# AWS CodeDeploy

## User Guide

## API Version 2014-10-06

# AWS CodeDeploy: User Guide

# Table of Contents

# What is AWS CodeDeploy?

AWS CodeDeploy is an AWS service that coordinates application deployments to instances. Instances are groups of one or more Amazon EC2 instances or on-premises instances or both. (On-premises instances are physical devices that are not Amazon EC2 instances.)

> **Tip**
> To view a video about this topic, see Introduction to AWS CodeDeploy on YouTube.

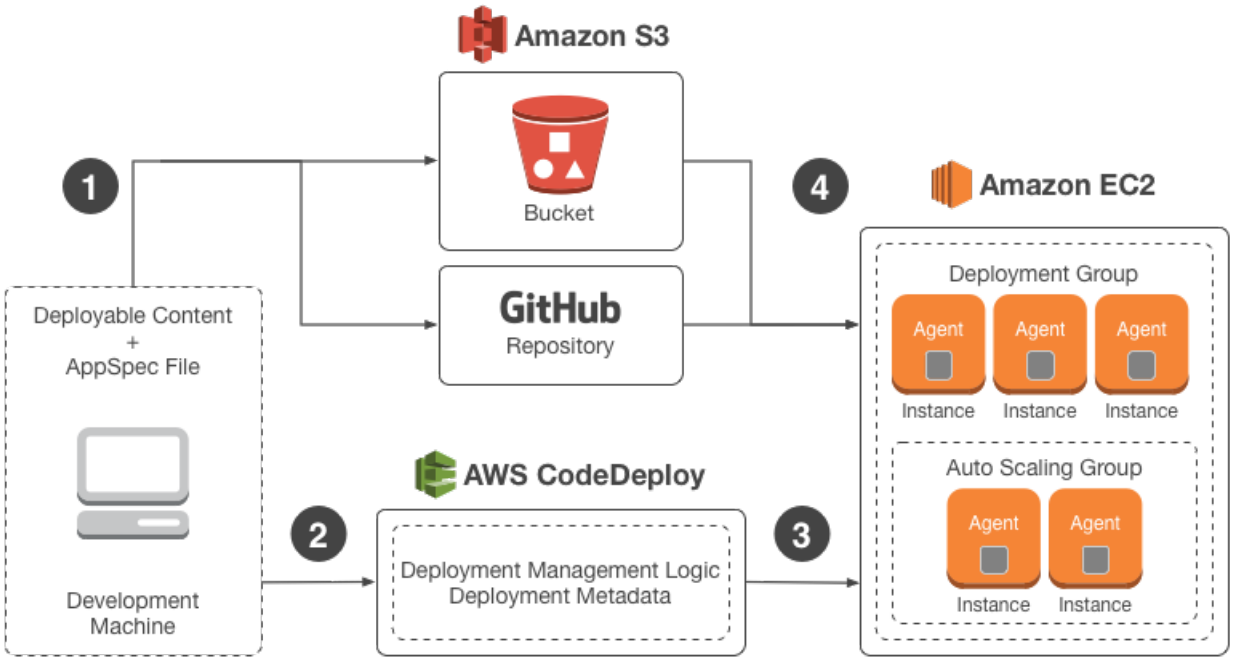
AWS CodeDeploy provides these benefits:

- **Automated deployments** – AWS CodeDeploy fully automates your application deployments, allowing you to deploy reliably and rapidly. You can consistently deploy your application across your development, test, and production environments. AWS CodeDeploy scales with your infrastructure so that you can deploy to one instance or thousands.
- **Minimize downtime** – AWS CodeDeploy helps maximize your application availability by performing rolling updates across your instances and tracking application health according to configurable rules. Deployments can easily be stopped and rolled back if there are errors.
- **Centralized control** – AWS CodeDeploy allows you to easily launch and track the status of your deployments through the AWS CodeDeploy console (a member of the AWS Management Console) or the AWS Command Line Interface (AWS CLI). AWS CodeDeploy gives you a detailed report that lists when each application revision was deployed and to which instances.
- **Easy to adopt** – AWS CodeDeploy is platform agnostic and works with any application. You can easily reuse your existing setup code. AWS CodeDeploy can also integrate with your existing software release process or continuous delivery toolchain.

An application can contain a variety of deployable content: code files, web files, configuration files, executables, packages, scripts, and so on. AWS CodeDeploy deploys applications from Amazon Simple Storage Service (Amazon S3) buckets and GitHub repositories.

## Overview of a Deployment

The following diagram illustrates the flow of a typical AWS CodeDeploy deployment:




Here's how it works:

1. First, you create deployable content (such as web pages, executable files, setup scripts, and similar) on your local development machine (or similar environment), and then you add an *Application Specification File* (AppSpec file) alongside the content. (The AppSpec file is unique to AWS CodeDeploy; it defines a series of deployment actions that you want AWS CodeDeploy to execute.) You bundle together your deployable content and the AppSpec file into an archive file, and then you upload the archive file to an Amazon S3 bucket or a GitHub repository. This archive file bundle is called an *application revision* (or simply a *revision*) in AWS CodeDeploy.
2. Next, you provide AWS CodeDeploy with information about your deployment, such as which Amazon S3 bucket or GitHub repository to pull from and which set of instances to deploy the revision to. AWS CodeDeploy calls a set of instances a *deployment group*. A deployment group contains individually-tagged instances, Amazon EC2 instances in Auto Scaling groups, or both.
3. Next, the AWS CodeDeploy Agent on each participating instance polls AWS CodeDeploy to determine what and when to pull the revision from the specified Amazon S3 bucket or GitHub repository.
4. Finally, the AWS CodeDeploy Agent on each participating instance pulls the revision from the specified Amazon S3 bucket or GitHub repository and starts deploying the contents to that instance, following the instructions in the AppSpec file that's provided.

Along the way, AWS CodeDeploy keeps a record of your deployments so that you can get information related to them, such as deployment status, deployment configuration parameters, instance health, and so on.

# Is AWS CodeDeploy the Right Solution for Me?

AWS CodeDeploy is part of a family of AWS deployment services that also includes Elastic Beanstalk, AWS CloudFormation and AWS OpsWorks.

AWS CodeDeploy extends this portfolio by providing a low-level building block service that is focused on software deployment. With AWS CodeDeploy, you can fully control software deployments across your

fleet, including controlling the pace of deployment across instances and defining the actions that will be taken at each stage in the deployment.

AWS CodeDeploy works with more than just AWS tools. AWS CodeDeploy works with a variety of configuration management systems, continuous integration and deployment systems, and source control systems. For more information, see the product integrations page. Using AWS CodeDeploy does not require that you make any changes to your existing code.

**Note**
For a complete list of AWS application management tools, see Application Management for AWS.

# Setting Up AWS CodeDeploy

Before you use AWS CodeDeploy for the first time, you must complete the following steps.

1. Sign Up for AWS (p. 4)
2. Provision an IAM User (p. 4)
3. Install or Upgrade, and Configure, the AWS CLI (p. 6)
4. Create an IAM Instance Profile and a Service Role (p. 6)

## Sign Up for AWS

Sign up for an AWS account, if you don't have one already. To sign up, go to http://aws.amazon.com/ and click **Sign Up**.

## Provision an IAM User

Follow these instructions to prepare an IAM user to use AWS CodeDeploy:

1. Create an IAM user, or use an existing one associated with your AWS account. To learn how to create a new IAM user, see Creating an IAM User (AWS Management Console).
2. Grant the IAM user access to AWS CodeDeploy—and AWS services and actions that AWS CodeDeploy depends on—by attaching the following policy to the IAM user:

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "autoscaling:*",
        "codedeploy:*",
        "ec2:*",
        "elasticloadbalancing:*",
        "iam:AddRoleToInstanceProfile",
        "iam:CreateInstanceProfile",
```

```
        "iam:CreateRole",
        "iam:DeleteInstanceProfile",
        "iam:DeleteRole",
        "iam:DeleteRolePolicy",
        "iam:GetInstanceProfile",
        "iam:GetRole",
        "iam:GetRolePolicy",
        "iam:ListInstanceProfilesForRole",
        "iam:ListRolePolicies",
        "iam:ListRoles",
        "iam:PassRole",
        "iam:PutRolePolicy",
        "iam:RemoveRoleFromInstanceProfile",
        "s3:*"
      ],
      "Resource" : "*"
    }
  ]
}
```

If you plan to use our AWS CloudFormation template to launch Amazon EC2 instances that are compatible with AWS CodeDeploy, you must grant the IAM user access to AWS CloudFormation—and AWS services and actions that AWS CloudFormation depends on—by attaching an *additional* policy to the IAM user, as follows:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:*"
      ],
      "Resource": "*"
    }
  ]
}
```

To learn how to attach a policy to a new or existing IAM user, see Managing Policies (AWS Management Console).

**Note**
The preceding policies provide access to many actions and resources associated with the IAM user. To learn how to restrict users to working with a limited set of AWS CodeDeploy actions and resources, see Access Permissions Reference (p. 226). For other AWS services listed in these statements, see the following:

- Overview of AWS IAM Policies
- Controlling User Access to Your Load Balancer
- Controlling Access to Your Auto Scaling Resources
- Controlling AWS CloudFormation Access with AWS Identity and Access Management

# Install or Upgrade, and Configure, the AWS CLI

To call AWS CodeDeploy commands from the AWS CLI on a local development machine, you must install the AWS CLI on that machine, if you haven't installed it already. If you have an older version of the AWS CLI already installed, you must upgrade it so that the AWS CodeDeploy commands will be available.

**Note**
AWS CodeDeploy commands first became available starting with version 1.6.1 of the AWS CLI. AWS CodeDeploy commands for working with on-premises instances became available starting with version 1.7.19 of the AWS CLI. If you have a version of the AWS CLI already installed, you can check its version by calling **aws --version**.

To install or upgrade the AWS CLI:

1. Follow the instructions in Installing the AWS Command Line Interface to install the AWS CLI for the first time or to upgrade an older version of the AWS CLI.
2. To configure the AWS CLI after you install or upgrade it, see Configuring the AWS Command Line Interface and Administering Access Keys for IAM Users.

   **Important**
   As you are configuring the AWS CLI, you will be prompted to specify an AWS region. You are limited to specifying one of the supported regions (p. 192):

   - US East (N. Virginia) region (us-east-1)
   - US West (Oregon) region (us-west-2)
   - EU (Ireland) region (eu-west-1)
   - Asia Pacific (Sydney) region (ap-southeast-2)

3. To verify the installation or upgrade, call the following command from the AWS CLI:

```
aws deploy help
```

   If successful, this command displays a list of available AWS CodeDeploy commands.

# Create an IAM Instance Profile and a Service Role

To enable AWS CodeDeploy to interact on your behalf with other AWS services that AWS CodeDeploy depends on, you must create a specific type of IAM role. To launch Amazon EC2 instances that are compatible with AWS CodeDeploy, you must create an additional IAM role. You need to create these 2 IAM roles only once for each AWS account.

The first IAM role is called a *service role*. To learn more about service roles and how to create one that is compatible with AWS CodeDeploy, see Create a Service Role (p. 140).

The second IAM role is called an *IAM instance profile*. To learn more about IAM instance profiles and how to create one that is compatible with AWS CodeDeploy, see Create an IAM Instance Profile (p. 84).

# Next Steps

Your IAM user, your AWS account, and your local development machine are now set up to work with AWS CodeDeploy. You can now use AWS CodeDeploy to set up instances and then deploy application revisions to them. To do this, follow the steps in Getting Started (p. 8).

# Getting Started with AWS CodeDeploy

To help you learn how to use AWS CodeDeploy, we provide an introductory walkthrough and three tutorials for you to experiment with.

We recommend that you go through the Create Deployment Walkthrough (p. 8) first. It assumes that you have no prior experience with AWS CodeDeploy, and it guides you through the minimum number of steps to deploy one of our sample application revisions to a set of Amazon EC2 instances.

After you complete the *Create Deployment Walkthrough*, you should try out the WordPress Deployment (Amazon Linux EC2) (p. 17) tutorial, the HelloWorld Deployment (Windows Server EC2) (p. 34) tutorial, or the On-Premises Instance Deployment (Windows Server or Ubuntu Server) (p. 50) tutorial. These tutorials help you complete many behind-the-scenes tasks so that you can gain more experience and confidence with AWS CodeDeploy.

Before you start the walkthrough or the tutorials, make sure that you have completed the prerequisites in Setting Up (p. 4), including setting up your IAM user, installing or upgrading the AWS CLI, and creating the necessary IAM instance profile and service role.

## Using the AWS CodeDeploy Create Deployment Walkthrough

The Create Deployment Walkthrough guides you through the minimum number of steps to deploy a revision to one or more Amazon EC2 instances. It assumes that you have no prior experience with AWS CodeDeploy and that you have not yet created any resources in AWS CodeDeploy such as applications, application revisions, or deployment groups.

If you want to practice deploying to on-premises instances instead of Amazon EC2 instances, see On-Premises Instance Deployment (Windows Server or Ubuntu Server) (p. 50).

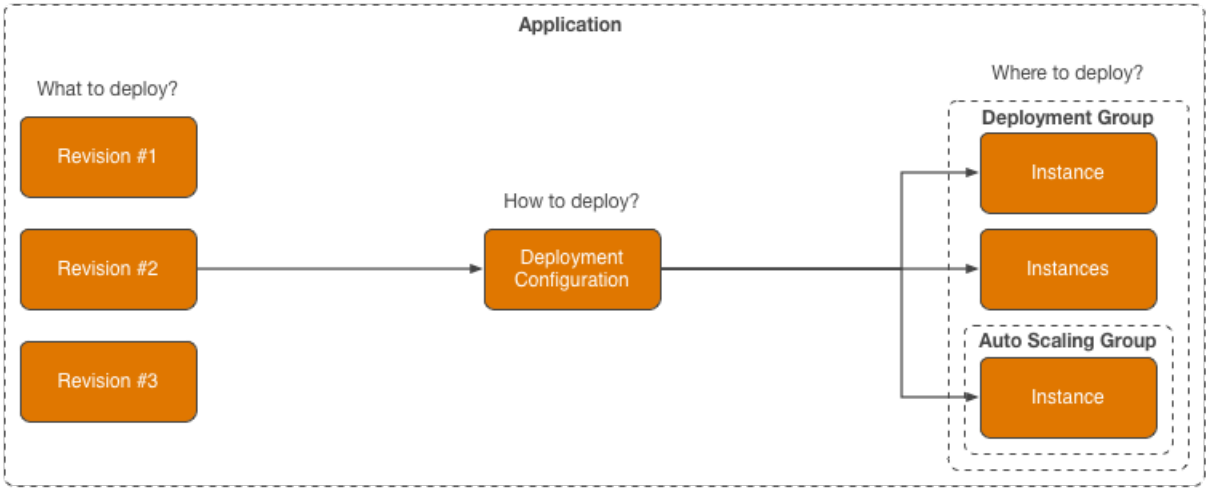
> **Tip**
> To view a video about this topic, see AWS CodeDeploy Deployment Walkthrough on YouTube.

**Topics**

- Terminology (p. 9)

# Terminology

The Create Deployment Walkthrough introduces some words and phrases that are unique to AWS CodeDeploy, which you may want to familiarize yourself with before you start. The following diagram and accompanying descriptions introduce you to these unique words and phrases and how they relate to each other:




**Application** – A name that uniquely identifies the application that you want to deploy. AWS CodeDeploy uses this name to ensure that the correct combination of revision, deployment configuration, and deployment group are being referenced during a deployment.

**Revision** – An archive file containing source content—such as source code, web pages, executable files, and deployment scripts—along with an Application Specification File (AppSpec file). Revisions are stored in Amazon S3 buckets or GitHub repositories. For Amazon S3, a revision is uniquely identified by its Amazon S3 object key and its ETag, version, or both. For GitHub, a revision is uniquely identified by its commit ID.

**Deployment configuration** – A set of deployment rules and deployment success and failure conditions that AWS CodeDeploy uses during a deployment.

**Deployment group** – A set of individual instances. A deployment group contains individually-tagged instances, Amazon EC2 instances in Auto Scaling groups, or both. For information on Amazon EC2 instance tags, see Working with Tags in the Console. For information on on-premises instances, see On-Premises Instances (p. 69). For information on Auto Scaling, see Auto Scaling Integration (p. 157).

# Prerequisites

Before you can use the various pages in the Create Deployment Walkthrough, you should review the following requirements.

- You must have an existing Amazon EC2 instance key pair, if you want AWS CodeDeploy to create some sample Amazon EC2 instances for you to deploy to, and you want to be able to connect to those Amazon EC2 instances after they are launched. To create a new Amazon EC2 instance key pair, follow the instructions at Creating Your Key Pair Using Amazon EC2. Be sure that your Amazon EC2 instance key pair is created in one of the supported regions (p. 192). If you create a new Amazon EC2 instance key pair, you must create it before you start this walkthrough, or else it will not show up in the **Key Pair Name** drop-down list in the walkthrough's **Instance Settings** page.
- You can't use this walkthrough to create a deployment that uses an existing application, application revision, deployment group, or custom deployment configuration in AWS CodeDeploy. Instead, follow the instructions in Deploy a Revision (p. 121).
- If you plan to use our AWS CloudFormation template to launch Amazon EC2 instances that are compatible with AWS CodeDeploy, the calling IAM user must have access to AWS CloudFormation and AWS services and actions that AWS CloudFormation depends on. If you have not yet followed the instructions in Setting Up (p. 4) to provision the calling IAM user, you must attach at least the following policy to the calling IAM user:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:*",
        "codedeploy:*",
        "ec2:*",
        "iam:AddRoleToInstanceProfile",
        "iam:CreateInstanceProfile",
        "iam:CreateRole",
        "iam:DeleteInstanceProfile",
        "iam:DeleteRole",
        "iam:DeleteRolePolicy",
        "iam:GetRole",
        "iam:PassRole",
        "iam:PutRolePolicy",
        "iam:RemoveRoleFromInstanceProfile"
      ],
      "Resource": "*"
    }
  ]
}
```

- If you plan to create a new *service role* (a special type of IAM role) that AWS CodeDeploy needs to work with other dependent AWS services on your behalf during the deployment, then the calling IAM user must have access to the IAM actions that it needs to create the new service role. If you have not yet followed the instructions in Setting Up (p. 4) to provision the calling IAM user, you must attach at least the following policy to the calling IAM user. Note that the following policy is a subset of the preceding policy for using AWS CloudFormation, so if you've already attached that preceding policy, you don't have to also attach the following policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateRole",
        "iam:PutRolePolicy"
      ],
      "Resource": "*"
    }
  ]
}
```

- To create applications and deployment groups, as well as to deploy applications, you must attach at least the following policy to the calling IAM user. Note that the following policy is a subset of the preceding policy for using AWS CloudFormation, so if you've already attached that preceding policy, you don't have to also attach the following policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codedeploy:*"
      ],
      "Resource": "*"
    }
  ]
}
```

# Start the Walkthrough

To start the Create Deployment Walkthrough:

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. Do one of the following:

   - If an introductory page appears, click **Get Started Now** in the middle of the page.
   - If the **Applications** page appears, then in the **Additional Information** area, click **Create Deployment Walkthrough**.

# Step 1: Welcome

The **Welcome** page is where you specify whether you want to deploy one of our sample application revisions or your own custom application revision.

To deploy one of our sample application revisions, select **Sample Deployment**, and then click **Next Step**. Then skip ahead to Step 2: Instance Settings (p. 12).

To deploy your own custom application revision, select **Custom Deployment**, and then click **Skip Walkthrough**. The walkthrough ends, and the **Create New Application** page is displayed instead. Continue on with the instructions in Create an Application (p. 111).

# Step 2: Instance Settings

The **Instance Settings** page is where you specify whether you want AWS CodeDeploy to launch a new set of Amazon EC2 instances to deploy one of our sample application revisions to, or whether you want to deploy one of our sample application revisions to your own Amazon EC2 instances.

If you have existing Amazon EC2 instances that you want to use, and they are already properly configured to participate in AWS CodeDeploy deployments, click **Skip This Step**, read and follow the on-screen instructions, and then skip ahead to Step 3: Application Name (p. 13). Otherwise:

1. Next to **Operating System**, select **Amazon Linux** or **Windows Server** to have AWS CodeDeploy configure and launch a set of new Amazon Linux or Windows Server Amazon EC2 instances that can participate in AWS CodeDeploy deployments.

   **Important**
   You may be billed for the Amazon EC2 instances that AWS CodeDeploy launches here. Be sure to terminate these Amazon EC2 instances after you're done using them. We use a special AWS CloudFormation template to launch these Amazon EC2 instances. For instructions on how to delete the AWS CloudFormation stack that we create to launch these Amazon EC2 instances, see Deleting a Stack on the AWS CloudFormation Console. The stack that you'll want to delete will start with the name **CodeDeploySampleStack**.

2. In the **Key Pair Name** drop-down list, select the name of an existing Amazon EC2 instance key pair that you will use to connect to the Amazon EC2 instances, if you want to, after they are launched.

   **Note**
   If the Amazon EC2 instance key pair that you want to use isn't visible in the **Key Pair Name** drop-down list, you'll need to create a new Amazon EC2 instance key pair. To create a new Amazon EC2 instance key pair, follow the instructions at Creating Your Key Pair Using Amazon EC2. Be sure that your Amazon EC2 instance key pair is created in one of the supported regions (p. 192). After you create the new Amazon EC2 instance key pair, you may need to start this walkthrough over again, as the new Amazon EC2 instance key pair may not automatically show up in the **Key Pair Name** drop-down list until you restart the walkthrough.

3. Next to **Tag Key and Value**, leave the proposed tag key and value. AWS CodeDeploy will use this tag key and value to find the Amazon EC2 instances during deployments.

   If you want to override the proposed tag key and value (for example, if you are running through this walkthrough multiple times without terminating any previously-created Amazon EC2 instances), we suggest that you leave the suggested tag key of `Name` in the **Key** box and type a different tag value in the **Value** box. To learn about Amazon EC2 instance tags, see Tagging Your Amazon EC2 Resources.

4. **Important**
   Before you click **Launch Instances** as described immediately following this note, make sure that the calling IAM user has the correct permissions to use the AWS CloudFormation

template. If the IAM user doesn't have the correct permissions, calls to AWS CloudFormation may fail. For more information, see the prerequisites (p. 10).

Click **Launch Instances**. AWS CodeDeploy displays a progress bar while AWS CodeDeploy uses an AWS CloudFormation template to configure and launch a set of new Amazon Linux or Windows Server Amazon EC2 instances that can participate in AWS CodeDeploy deployments.

5. If you want to see details about what is going on while you wait, click **See more details in AWS CloudFormation**. The AWS CloudFormation console opens in a separate web browser tab. Switch over to the new tab, and look for a stack that starts with the name **CodeDeploySampleStack**. When the **Status** column shows **CREATE_COMPLETE**, your Amazon EC2 instances are ready. (Note that this may take several minutes.)
6. After the Amazon EC2 instances are ready, on the Create Deployment Walkthrough web browser tab the progress bar will be replaced by a success message, and the **Next Step** button will be enabled. To continue, click **Next Step**.

# Step 3: Application Name

The **Application Name** page is where you specify a unique name that AWS CodeDeploy will use to ensure that it's referencing the correct combination of revision, deployment configuration, and deployment group during the deployment.

In the **Application Name** box, leave the proposed application name. If you want to override the proposed application name (for example, if you are running through this walkthrough multiple times without deleting any previously-created applications), type a different name. (This name must be unique across all of the applications that are created associated with the AWS account.) Then click **Next Step**.

# Step 4: Revision

The **Revision** page is where you review information about our sample application revision that you will be deploying.

If you clicked **Skip This Step** in Step 2 (p. 12), then in the **Revision Type** drop-down list, select the type of application revision that corresponds to the Amazon EC2 instances type (Amazon Linux or Windows Server) that you'll be deploying to. Otherwise, simply review information about our sample application revision. Then click **Next Step**.

**Tip**
If you want to inspect our sample revision's contents, click **Download Sample Bundle**, and follow your web browser's instructions to download and view the revision's contents.

# Step 5: Deployment Group

The **Deployment Group** page is where you provide information about the Amazon EC2 instances that will participate in the deployment.

1. In the **Deployment Group Name** box, leave the proposed deployment group name. If you want to override the proposed deployment group name (for example, if you are running through this walkthrough multiple times without deleting any previously-created deployment groups), type a different name. (This name must be unique across the application name that you specified in Step 3 (p. 13).)
2. If you clicked **Skip This Step** in Step 2 (p. 12), in the **Add Instances** area, overwrite the proposed values of the **Key** and **Value** boxes with the key and value of the key-value pair for the Amazon EC2 instances that you'll be deploying to. Otherwise, the key and value of the key-value pair that you specified in the **Instance Settings** page should already be visible (for example, `Name` and `CodeDeployDemo`).

Optionally, if your Amazon EC2 instances have multiple key-value pairs and you want to specify them here, you can type them into the next available blank row. As you begin adding key-value pair information, a new blank row appears for you to add another key-value pair if desired. You can repeat this step for up to 10 key-value pairs.

> **Tip**
> As AWS CodeDeploy finds Amazon EC2 instances that match each specified key-value pair, it displays the number of matching Amazon EC2 instances. To see more information about the Amazon EC2 instances, click the number.
> If the number is larger than the number that you are expecting to see here (for example, if you are running through this walkthrough multiple times), and if you are using our special AWS CloudFormation template to launch new Amazon EC2 instances, you must click **Cancel** and then restart the Create Deployment Walkthrough from the beginning, specifying a different Amazon EC2 instance tag value than the one that is proposed in Step 2 (p. 12). (And be sure to delete the AWS CloudFormation stack to terminate the related Amazon EC2 instances that you won't be deploying to now.) Or, if you are using your own Amazon EC2 instances, add a new tag key and value to your Amazon EC2 instances and then specify a different Amazon EC2 instance tag key and value than the one that is proposed in the **Add Instances** area.

If you make any mistakes here and need to remove a key-value pair from the list, click the corresponding remove icon.

3. To add an Auto Scaling group (if you have one available) to the deployment group, click **Search by Auto Scaling Group Names**, and then type the Auto Scaling group name. You can repeat this step for up to 10 Auto Scaling groups.

   > **Tip**
   > As AWS CodeDeploy finds Auto Scaling groups that match each specified Auto Scaling group name, it displays the number of matching Amazon EC2 instances in those Auto Scaling groups. To see more information about the Amazon EC2 instances, click the number.

   If you make any mistakes and need to remove an Auto Scaling group from the list, click the corresponding remove icon.
4. Click **Next Step**.

# Step 6: Service Role

The **Service Role** page is where you specify a *service role* (a special type of IAM role) that AWS CodeDeploy will use to work with other dependent AWS services on your behalf during the deployment.

In the **Service Role** list, select whether you want to **Create a new service role** or **Use an existing service role**. If you choose to create a new service role, be sure to read the following important note.

> **Important**
> If you choose to create a new service role, before you click **Create a new service role** as described immediately following this note, make sure that the calling IAM user has the correct permissions to create the new service role. If the IAM user doesn't have the correct permissions, the service role creation may fail. For more information, see the prerequisites (p. 10).

If you have never used this walkthrough before, we recommend that you click **Create a new service role**, accept the new service role's proposed name by clicking **Next Step**, and skip ahead to Step 7: Deployment Configuration (p. 15).

However, you may have an existing service role from previously using this walkthrough, or you may otherwise have an existing service role that trusts the AWS CodeDeploy service account with at minimum the trust and permissions that are described in Create a Service Role (p. 140). If so, you can select **Use**

**an existing service role**, select the service role in the **Role Name** drop-down list if it is not already selected, and then click **Next Step**.

# Step 7: Deployment Configuration

The **Deployment Configuration** page is where you specify how fast you want the deployment to be performed and what rules AWS CodeDeploy will follow to report back whether the overall deployment succeeds or fails.

1. Select the **Default Deployment Configurations** radio button to select a built-in configuration for this deployment, or select the **Create Custom Deployment Configuration** radio button to create your own configuration for this deployment.
2. If you selected **Default Deployment Configurations**, and you want to use a different configuration than the one that is already selected, click **Select** next to the desired configuration. Click **Next Step**, and skip ahead to Step 8: Review (p. 15).
3. If you selected **Create Custom Deployment Configuration**, complete the following:

   a. In the **Deployment Config Name** box, type a name that uniquely describes the configuration.
   b. In either the **Number** or **Percentage** box, type either the number or percentage of total Amazon EC2 instances that should be available at any time during the deployment.
   c. Click **Next Step**.

# Step 8: Review

The **Review** page lists the details of the deployment that you're about to create.

1. If you need to change any of the details, click one of the **Edit** links to get to the page containing the details that you want to change. When you're done making changes, keep clicking **Next Step** on each page to return to the **Step 8: Review** page.

   **Important**
   Before you click **Deploy Now** as described immediately following this note, make sure that the calling IAM user has the correct permissions to use AWS CodeDeploy to create applications and deployment groups, as well as to deploy applications. If the IAM user doesn't have the correct permissions, the creation of applications and deployment groups, as well as deployments, may fail. For more information, see the prerequisites (p. 11).
2. When you're satisfied with all of the details, click **Deploy Now**. The **Deployments** page appears, showing status information about the newly created deployment.

   To update the deployment's current status, click the refresh button next to the table.

   To get additional information about the deployment, see View Instance Details by Using the AWS CodeDeploy Console (p. 128).
3. After the deployment succeeds, you can verify the deployment through your web browser. Our sample revision deploys a single web page to each of the participating Amazon EC2 instances, which you can view in your web browser by going to `http://PublicDNS` for each Amazon EC2 instance, for example `http://ec2-01-234-567-890.compute-1.amazonaws.com`. (To get an Amazon EC2 instance's public DNS value, in the Amazon EC2 console, select the Amazon EC2 instance, and look for the **Public DNS** value in the **Description** tab.) The web page will display a simple message of congratulations.

# Clean Up

To keep from getting continually charged for resources that you used during the Create Deployment Walkthrough, you must clean up the associated resources. If you used our AWS CloudFormation template to launch Amazon EC2 instances, you should delete the associated AWS CloudFormation stack, which will terminate the Amazon EC2 instances and their associated resources. If you didn't use our AWS CloudFormation template, but you instead launched individual Amazon EC2 instances just for this walkthrough, you should terminate those Amazon EC2 instances. Optionally, you can delete the AWS CodeDeploy deployment component records that are associated with this walkthrough to remove visual clutter from the AWS CodeDeploy console.

## To delete the AWS CloudFormation stack

1. Sign in to the AWS Management Console and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. Check the box in the row next to the stack starting with `CodeDeploySampleStack` in the **Stack Name** column.
3. Click **Delete Stack**.
4. When prompted, click **Yes, Delete**. The associated Amazon EC2 instances are terminated, and the Amazon EC2 instances' associated IAM instance profile and service role are deleted.

## To terminate individual Amazon EC2 instances

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **Instances**, click **Instances**.
3. Check the box in the row that corresponds to each Amazon EC2 instance that you want to terminate.
4. Click **Actions**, point to **Instance State**, and then click **Terminate**.
5. When prompted, click **Yes, Terminate**. The selected Amazon EC2 instances are terminated.

## To delete AWS CodeDeploy deployment component records

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).
2. If the **Applications** page is not visible, in the AWS CodeDeploy service navigation bar, click **Applications**.
3. Click the name of the application that you want to delete.
4. Click **Delete Application**.
5. When prompted, click **Delete**. AWS CodeDeploy deletes from its system all records about the application and its associated deployment groups, revisions, and deployments.

# Getting Started by Deploying a WordPress Application with AWS CodeDeploy (Amazon Linux and Linux, OS X, or Unix)

This tutorial helps you build experience and confidence with AWS CodeDeploy by guiding you through the deployment of a sample application to a single Amazon EC2 instance. In this tutorial, you will deploy WordPress, an open source blogging tool and content management system based on PHP and MySQL, to a single Amazon Linux Amazon EC2 instance.

If you want to practice deploying to a Windows Server Amazon EC2 instance instead of an Amazon Linux Amazon EC2 instance, see HelloWorld Deployment (Windows Server EC2) (p. 34).

If you want to practice deploying to an on-premises instance instead of an Amazon EC2 instance, see On-Premises Instance Deployment (Windows Server or Ubuntu Server) (p. 50).

> **Note**
> This tutorial builds on concepts that were introduced in the Create Deployment Walkthrough (p. 8). If you have not yet completed the *Create Deployment Walkthrough*, you may want to start there first.
> This tutorial's steps are presented from the perspective of a local development machine running Linux, OS X, or Unix, as the accompanying tutorial targets Amazon Linux. While you can complete most of these steps on a local machine running Windows, you will need to adapt some of the steps that cover commands such as **chmod** and **wget**, applications such as sed, and directory paths such as `/tmp`.

Before you start this tutorial, you must complete the prerequisites in Setting Up (p. 4), including configuring your IAM user account, installing or upgrading the AWS CLI, and creating an IAM instance profile and a service role.



## Step 1: Launch an Amazon Linux Amazon EC2 Instance

To deploy the WordPress application with AWS CodeDeploy, you'll need an Amazon Linux Amazon EC2 instance to deploy to.

You can launch a new Amazon Linux Amazon EC2 instance that's properly configured to participate in AWS CodeDeploy deployments by following the instructions in Configure Instances (p. 72). In those instructions, when you get to the part about assigning an Amazon EC2 instance tag to the Amazon EC2 instance, be sure to specify the tag key of **`Name`** and the tag value of **`CodeDeployDemo`**. (If you specify a different tag key or tag value than this one for the Amazon EC2 instance that you'll use in this tutorial, then the instructions later in Step 4: Deploy Your Application (p. 25) may produce unexpected results.)

After you've followed those instructions to launch the new Amazon EC2 instance, return to this page, and continue on to the next section to practice connecting to your newly-launched Amazon EC2 instance, if you wish.

**Note**
The preceding Amazon EC2 instance launch instructions direct you to continue on to Create an Application (p. 111) as a next step. Ignore those directions and continue on instead with the next section to practice connecting to your Amazon EC2 instance after you launch it, if you wish.

## Connect to Your Amazon Linux Amazon EC2 Instance

After your new Amazon EC2 instance is launched, you can practice connecting to it, if you wish, by following these instructions.

1. Use the **ssh** command (or an SSH-capable terminal emulator such as PuTTY) to connect to your Amazon Linux Amazon EC2 instance. You will need the Amazon EC2 instance's public DNS address and the private key for the key pair that you selected or created when you started the Amazon EC2 instance. For more information, see Connect to Your Instance.

   For example, if your Amazon EC2 instance's public DNS address is **`ec2-01-234-567-890.compute-1.amazonaws.com`**, and your Amazon EC2 instance key pair for SSH access is named **`codedeploydemo.pem`**, you would type:

   ```
   ssh -i /path/to/codedeploydemo.pem ec2-user@ec2-01-234-567-890.compute-
   1.amazonaws.com
   ```

   Replace */path/to/*`codedeploydemo.pem` in this example with the path to your own `.pem` file, and the example DNS address with the address to your Amazon Linux Amazon EC2 instance.

   **Note**
   If you receive an error about your keyfile's permissions being too open, you will need to restrict its permissions to give only the current user (you) access. For example, with the **chmod** command on Linux, OS X, or Unix, type:

   ```
   chmod 400 /path/to/codedeploydemo.pem
   ```

2. Once you are successfully logged in, you will see your Amazon EC2 instance's AMI banner:

   ```
         __|  __|_  )
         _|  (     /   Amazon Linux AMI
        ___|\___|___|
   ```

3. You can now log out of the running Amazon EC2 instance, if you want to.

   **Caution**
   Do not stop or terminate the Amazon EC2 instance. Otherwise, AWS CodeDeploy won't be able to deploy to it.

# Step 2: Configure your Source Content to Deploy to the Amazon Linux Amazon EC2 Instance

Now that you have set up the Amazon EC2 instance, it's time to configure your application's source content so that you have something that you can deploy to the Amazon EC2 instance.

**Topics**

# Get the Source Code

For this tutorial, you'll deploy the WordPress content publishing platform from your development machine to the target Amazon EC2 instance. To get the WordPress source code, you can use built-in command-line calls. Or, if you have Git installed on your development machine, you can use that instead.

These steps assume that you'll download a copy of the WordPress source code into your development machine's `/tmp` directory. (You can choose any directory you like, but remember to substitute your chosen location for `/tmp` wherever it is specified throughout these steps.)

**Topics**



## To get a copy of the WordPress source code by using built-in command-line calls

1. Call the **wget** command to download a copy of the WordPress source code as a .zip file to the current directory:

```
wget https://github.com/WordPress/WordPress/archive/master.zip
```

2. Then call the **unzip**, **mkdir**, **cp**, and **rm** commands in the following order to unpack the `master` .zip file into the `/tmp/WordPress_Temp` directory (folder), copy its unzipped contents to the `/tmp/WordPress` destination folder, and then delete the temporary `/tmp/WordPress_Temp` folder as well as the `master` file itself:

```
unzip master -d /tmp/WordPress_Temp
mkdir -p /tmp/WordPress
cp -paf /tmp/WordPress_Temp/WordPress-master/* /tmp/WordPress
rm -rf /tmp/WordPress_Temp
rm -f master
```

   This leaves you with a clean set of WordPress source code files in the `/tmp/WordPress` folder to work with.

## To get a copy of the WordPress source code by using Git

1. If you don't already have Git installed, go to Git to download and install Git on your development machine.

2. In the `/tmp/WordPress` folder, call the **git init** command to prepare the folder to interact with Git.
3. Then, call the **git clone** command to clone the public WordPress repository, making your own copy of it in the `/tmp/WordPress` destination folder:

```
git clone https://github.com/WordPress/WordPress.git /tmp/WordPress
```

This leaves you with a clean set of WordPress source code files in the `/tmp/WordPress` folder to work with.

## Create Scripts to Run Your Application

Next, you will create a folder and a number of scripts in the directory that AWS CodeDeploy will use to set up and deploy your application revision on the target Amazon EC2 instance. (You can use any text editor to create the scripts.)

1. Create a scripts directory in your copy of the WordPress source code under the `/tmp/WordPress` folder, in a subfolder named `scripts`:

```
mkdir -p /tmp/WordPress/scripts
```

2. Create an `install_dependencies.sh` script in `/tmp/WordPress/scripts`. The `install_dependencies.sh` script will install Apache, MySQL, PHP and add MySQL support to PHP. Add the following lines to the file:

```
#!/bin/bash
yum groupinstall -y "Web Server" "MySQL Database" "PHP Support"
yum install -y php-mysql
```

3. Create a `stop_server.sh` script in `/tmp/WordPress/scripts` to stop Apache and MySQL. Add these lines:

```
#!/bin/bash
isExistApp=`pgrep httpd`
if [[ -n  \$isExistApp ]]; then
   service httpd stop
fi
isExistApp=`pgrep mysqld`
if [[ -n  \$isExistApp ]]; then
    service mysqld stop
fi
```

4. Create a `start_server.sh` script in `/tmp/WordPress/scripts` to start Apache and MySQL. Add these lines:

```
#!/bin/bash
service httpd start
service mysqld start
```

5. Finally, create a `change_permissions.sh` script in `/tmp/WordPress/scripts`. This will be used to change the folder permissions in Apache:

```
#!/bin/bash
chmod -R 644 /var/www/html/WordPress
```

6. Give all of the scripts executable permissions. On the command-line, type:

```
chmod +x /tmp/WordPress/scripts/*
```

## Add an Application Specification File

Next, you will add an Application Specification File alongside the source code. The Application Specification File (AppSpec file) is a YAML-formatted file that AWS CodeDeploy uses to describe what AWS CodeDeploy should install onto your Amazon EC2 instance, as well as what scripts must be run at various times during a deployment. The AppSpec file:

- Maps the source files in your application revision to their destinations on the target Amazon EC2 instance.
- Specifies custom permissions for deployed files.
- Specifies scripts to be run on the target Amazon EC2 instance at various stages of the deployment process.
- Must be named `appspec.yml` and must be placed in the application's source code's root directory.

With your text editor, create a file named `appspec.yml` and save it to the application's source code's root directory. This will be the AppSpec file. Add the following lines to the file:

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html/WordPress
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies.sh
      timeout: 300
      runas: root
  AfterInstall:
    - location: scripts/change_permissions.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

AWS CodeDeploy will use this AppSpec file to copy all of the files in the `/tmp/WordPress` folder on the development machine to the `/var/www/html/WordPress` folder on the target Amazon EC2 instance. During the deployment, AWS CodeDeploy will run the specified scripts as `root` in the `/var/www/html/WordPress/scripts` folder on the target Amazon EC2 instance at various specified events during the deployment lifecycle, such as **`BeforeInstall`** and **`AfterInstall`**. If any of these scripts take longer than 300 seconds (5 minutes) to run, AWS CodeDeploy will stop the deployment at that point and mark the deployment as failed.

For more information about each of these settings, see the AppSpec File Reference (p. 212).

**Important**
The locations and numbers of spaces between each of the items in this file is important. If the spacing is incorrect, AWS CodeDeploy will raise an error that may be difficult to debug. For more information, see AppSpec file Spacing (p. 223).

# Step 3: Upload Your WordPress Application to Amazon S3

Now that you've configured your source content, you will prepare and upload it to a location that AWS CodeDeploy can deploy the source content from. The following instructions show you how to provision an Amazon S3 bucket, prepare the application revision's files for the bucket, bundle the revision's files, and then push the revision to the bucket.

**Note**
You can also use AWS CodeDeploy to deploy applications from GitHub repositories to instances. However, deploying from GitHub repositories is not covered in this tutorial. To learn more, see GitHub Integration (p. 177).

**Topics**



## Provision an Amazon S3 Bucket

Create a new storage container, or *bucket*, in Amazon S3—or use an existing bucket. Also, make sure that the bucket allows both uploads and downloads, that you can upload the revision to the bucket, and that participating Amazon EC2 instances can download the revision from the bucket.

You can use the AWS CLI, the Amazon S3 console, or the Amazon S3 APIs to create a new Amazon S3 bucket. After you create the bucket, make sure to give both the bucket and your IAM user the proper access permissions.

**Topics**



### To use the AWS CLI to create a new Amazon S3 bucket

Call the **mb** command to create an Amazon S3 bucket named **`CodeDeployDemoBucket`**, for example:

```
aws s3 mb s3://CodeDeployDemoBucket
```

**Note**
Bucket names must be unique across Amazon S3 for all AWS accounts. If you aren't able to use **`CodeDeployDemoBucket`**, try a different bucket name, such as **`CodeDeployDemoBucket`** followed by a dash and your initials or some other unique identifier. Then be sure to substitute your bucket name for **`CodeDeployDemoBucket`** wherever you see it throughout the rest of this tutorial.

Also, note that the Amazon S3 bucket must be created in the same AWS region as the region that your target Amazon EC2 instances are launched in. For example, if you create the bucket in the US East (N. Virginia) region, then your target Amazon EC2 instances must be launched in the US East (N. Virginia) region too.

### To use the Amazon S3 console to create a new Amazon S3 bucket

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the Amazon S3 console, click **Create Bucket**.
3. In the **Bucket Name** box, type a name for the bucket, such as **`CodeDeployDemoBucket`**.

   **Note**
   Bucket names must be unique across Amazon S3 for all AWS accounts. If you aren't able to use **`CodeDeployDemoBucket`**, try a different bucket name, such as **`CodeDeployDemoBucket`** followed by a dash and your initials or some other unique identifier. Then be sure to substitute your bucket name for **`CodeDeployDemoBucket`** wherever you see it throughout the rest of this tutorial.
   Also, note that the Amazon S3 bucket must be created in the same AWS region as the region that your target Amazon EC2 instances are launched in. For example, if you create the bucket in the US East (N. Virginia) region, then your target Amazon EC2 instances must be launched in the US East (N. Virginia) region too.
4. In the **Region** list, select the target region.
5. Click **Create**. Amazon S3 creates the new bucket.

### Give the Amazon S3 bucket and your IAM user the proper permissions

The Amazon S3 bucket must allow you to upload to it. One way to specify this is through an Amazon S3 bucket policy. For example, the following Amazon S3 bucket policy allows AWS account `111122223333` to upload anywhere within the Amazon S3 bucket named `CodeDeployDemoBucket`:

```
{
  "Statement": [
    {
      "Action": ["s3:PutObject"],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::CodeDeployDemoBucket/*",
      "Principal": {
        "AWS": [
          "111122223333"
        ]
      }
    }
  ]
}
```

Although you will not be deploying the revision in this step, the Amazon S3 bucket must also allow download requests coming from each participating Amazon EC2 instance. While you are working with the bucket, now is a good time to verify this. One option is to specify this is through an Amazon S3 bucket policy. For example, the following Amazon S3 bucket policy allows any Amazon EC2 instance with an attached IAM instance profile containing the ARN `arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo` to download from anywhere within the Amazon S3 bucket named `CodeDeployDemoBucket`:

```
{
  "Statement": [
```

```
    {
      "Action": ["s3:Get*", "s3:List*"],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::CodeDeployDemoBucket/*",
      "Principal": {
        "AWS": [
          "arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo"
        ]
      }
    }
  ]
}
```

To learn how to generate and attach an Amazon S3 bucket policy, see Bucket Policy Examples.

Also, your account must have permission to upload the revision to the Amazon S3 bucket. One way to specify this is through an IAM policy. For example, the following custom IAM user policy allows your IAM user to upload revisions anywhere within the Amazon S3 bucket named `CodeDeployDemoBucket`:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":["s3:PutObject"],
      "Resource":"arn:aws:s3:::CodeDeployDemoBucket/*"
    }
  ]
}
```

To learn how to create and attach an IAM policy, see Managing IAM Policies (AWS Management Console).

## Prepare the Application's Files for the Bucket

Make sure that the WordPress application files that you downloaded earlier, along with the AppSpec file and the scripts that you created earlier, are organized on your development machine similar to the following:

```
/tmp/
  |--WordPress/
      |-- appspec.yml
      |-- scripts/
      |    |-- install_dependencies.sh
      |    |-- change_permissions.sh
      |    |-- start_server.sh
      |    |-- stop_server.sh
      |-- wp-admin/
      |    |-- (various files...)
      |-- wp-content/
      |    |-- (various files...)
      |-- wp-includes/
      |    |-- (various files...)
      |-- index.php
      |-- license.txt
```

```
    |-- readme.html
    |-- (various files ending with .php...)
```

## Bundle the Application's Files Into a Single Archive File, and Push the Archive File

Next, you need to bundle the WordPress application files, along with the AppSpec file, into an archive file (known as an application *revision*).

> **Note**
> You may be charged for storing objects in a bucket and for transferring application revisions into and out of a bucket. For more information, see Amazon S3 Pricing.

1. On the development machine, switch to the folder where the files are stored:

```
cd /tmp/WordPress
```

> **Note**
> If you don't switch to this folder, then when you bundle the files together, the file bundling will start at whatever point your current folder is. For example, if your current folder is `/tmp` instead of `/tmp/WordPress`, then you will start bundling files and subfolders that are in the `tmp` folder, which may include more than just the `WordPress` subfolder.

2. Call the **create-application** command to register a new application with AWS CodeDeploy named **`WordPress_App`**:

```
aws deploy create-application --application-name WordPress_App
```

3. Call the AWS CodeDeploy push command to bundle the files together, upload the revisions to Amazon S3, and register information with AWS CodeDeploy about the uploaded revision, all in one action. For example:

```
aws deploy push \
  --application-name WordPress_App \
  --s3-location s3://CodeDeployDemoBucket/WordPressApp.zip \
  --ignore-hidden-files
```

This command bundles the files from the current directory (excluding any hidden files) into a single archive file named **`WordPressApp.zip`**, uploads the revision to the **`CodeDeployDemoBucket`** bucket, and registers information with AWS CodeDeploy about the uploaded revision.

# Step 4: Deploy Your WordPress Application

Now you will deploy the sample WordPress application revision that you've previously uploaded to Amazon S3. You will use the AWS CLI or the AWS CodeDeploy console to deploy the revision and to monitor the deployment's progress. After the deployment succeeds, you will check the results.

**Topics**

# Deploy Your Application Revision with AWS CodeDeploy

**Topics**



## To use the AWS CLI to deploy your application revision

1. First, the deployment will need a corresponding deployment group. However, before you create the deployment group, you will need a service role ARN. A *service role* is a special type of IAM role that gives a service permission to act on your behalf. In this case, the service role will give AWS CodeDeploy permission to access your Amazon EC2 instances to expand (read) their Amazon EC2 instance tags.

   You should have already created a service role that compatible with AWS CodeDeploy by following the instructions in Use the AWS CLI to Create a Service Role that is Compatible with AWS CodeDeploy (p. 142). (If not, go there now and create the service role.) To get the service role ARN, see Use the AWS CLI to Get the Service Role ARN (p. 144).
2. Now that you have a service role ARN, call the **create-deployment-group** command to create a deployment group named **`WordPress_DepGroup`**, associated with the application named **`WordPress_App`**, using the Amazon EC2 tag named **`CodeDeployDemo`** and deployment configuration named **`CodeDeployDefault.OneAtATime`**, with the specified service role ARN:

   ```
   aws deploy create-deployment-group \
     --application-name WordPress_App \
     --deployment-group-name WordPress_DepGroup \
     --deployment-config-name CodeDeployDefault.OneAtATime \
     --ec2-tag-filters Key=Name,Value=CodeDeployDemo,Type=KEY_AND_VALUE \
     --service-role-arn serviceRoleARN
   ```

   In the preceding command, *`serviceRoleARN`* is the service role ARN that you noted earlier.
3. Now call the **create-deployment** command to create a deployment associated with the application named **`WordPress_App`**, the deployment configuration named **`CodeDeployDefault.OneAtATime`** and the deployment group named **`WordPress_DepGroup`**, using the application revision named **`WordPressApp.zip`** in the bucket named **`CodeDeployDemoBucket`**:

   ```
   aws deploy create-deployment \
     --application-name WordPress_App \
     --deployment-config-name CodeDeployDefault.OneAtATime \
     --deployment-group-name WordPress_DepGroup \
     --s3-location bucket=CodeDeployDemoBucket,bundleType=zip,key=WordPress
   App.zip
   ```

## To use the AWS CodeDeploy console to deploy your application revision

1. Before you start using the AWS CodeDeploy console to deploy your application revision, you will need a service role ARN. A *service role* is a special type of IAM role that gives a service permission to act on your behalf. In this case, the service role will give AWS CodeDeploy permission to access your Amazon EC2 instances to expand (read) their Amazon EC2 instance tags.

   You should have already created a service role that compatible with AWS CodeDeploy by following the instructions in Use the IAM Console to Create a Service Role that is Compatible with AWS

CodeDeploy (p. 140). (If not, go there now and create the service role.) To get the service role ARN, see Use the IAM Console to Get the Service Role ARN (p. 143).

2. Now that you have the service role ARN, you can start using the AWS CodeDeploy console to deploy your application revision:

   Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

3. If the **Applications** page is not visible, in the AWS CodeDeploy service navigation drop-down list, select **Applications**.
4. In the list of applications, click **WordPress_App**.
5. Under **Deployment Groups**, click **Create New Deployment Group**.
6. In the **Deployment Group Name** box, type **`WordPress_DepGroup`**.
7. In the list of tags, select **Amazon EC2** in the **Tag Type** drop-down list.
8. In the **Key** box, type **`Name`**.
9. In the **Value** box, type **`CodeDeployDemo`**.

   **Note**
   After you type **`CodeDeployDemo`**, a **1** should appear under **Instances** to confirm that AWS CodeDeploy found 1 matching Amazon EC2 instance.

10. In the **Deployment Config** drop-down list, select **CodeDeployDefault.OneAtATime**, if it is not already selected.
11. In the **Service Role ARN** drop-down list, select the service role ARN that you noted earlier, and then click **Create Deployment Group**.
12. In the AWS CodeDeploy service navigation drop-down list, select **Deployments**.
13. Click **Create New Deployment**.
14. In the **Application** drop-down list, select **WordPress_App**.
15. In the **Deployment Group** drop-down list, select **WordPress_DepGroup**.
16. With the option **My application is stored in Amazon S3** already selected next to **Revision Type**, in the **Revision Location** box, type the location of the sample WordPress application revision that you previously uploaded to Amazon S3. To get the location:

    1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
    2. In the **All Buckets** list, select **CodeDeployDemoBucket** (or whatever the name is of the bucket that you uploaded your application revision to earlier). A list of objects in that bucket appears.
    3. In the list of objects, select **WordPressApp.zip**.
    4. If the **Properties** pane is not visible, click the **Properties** button.
    5. In the **Properties** pane, copy the value of the **Link** field to your clipboard.

       It might look something like this:

       **`https://s3.amazonaws.com/CodeDeployDemoBucket/WordPressApp.zip`**
    6. Return to the AWS CodeDeploy console, and in the **Revision Location** box, paste the **Link** field value that you copied to your clipboard.

17. After you type the revision location and leave the **Revision Location** box, the **File Type** list appears.
18. In the **File Type** list, if a message appears stating that the file type could not be detected, select **.zip** in the list of file types.

19. If you want to type a comment about the deployment, enter the comment into the **Deployment Description** box. Otherwise, you can leave it blank.
20. In the **Deployment Config** drop-down list, select **CodeDeployDefault.OneAtATime**, if it is not already selected.
21. Click **Deploy Now**. The **Deployments** page appears, showing information about your newly created deployment.

    **Tip**
    To update the deployment's current status, click the refresh button next to the table.

# Monitor (and Troubleshoot) Your Deployment

**Topics**



## To use the AWS CLI to monitor (and troubleshoot) your deployment

1. Get the deployment's ID by calling the **list-deployments** command against the application named **`WordPress_App`** and the deployment group named **`WordPress_DepGroup`**:

```
aws deploy list-deployments --application-name WordPress_App --deployment-
group-name WordPress_DepGroup --query 'deployments' --output text
```

2. Then call the **get-deployment** command to see if the deployment succeeded, supplying the deployment ID:

```
aws deploy get-deployment --deployment-id deploymentID --query 'deploy
mentInfo.status' --output text
```

3. The **get-deployment** command will return the deployment's overall status. If successful, the value will be `Succeeded`.

   If, however, the overall status is `Failed`, you can try further troubleshooting by calling commands such as list-deployment-instances and get-deployment-instance.

   For even more troubleshooting options, see Investigating Deployment Failures on Instances (p. 199).

## To use the AWS CodeDeploy console to monitor (and troubleshoot) your deployment

On the **Deployments** page in the AWS CodeDeploy console, you can monitor your deployment's current status by looking at the value of the **Status** column for your deployment.

**Tip**
To update the deployment's current status, click the refresh button next to the table.

To get more information about your deployment, especially if the **Status** column value has any value other than **Succeeded**:

1. In the **Deployments** table, click the arrow next to the deployment ID. **Details** and **Instances** areas appear for the deployment. After a deployment fails, a message should appear in the **Details** area that describes why the deployment failed.

**Note**
If you don't see an **Instances** area, click the refresh button next to the table. After the **Status** column changes from **InProgress** to **Created**, the **Instances** area should then appear.

2. In the **Instances** area, click **View All Instances**. Additional information about the deployment appears. After a deployment fails, you may be able to determine on which Amazon EC2 instances the deployment failed and at which step the deployment failed on each failed Amazon EC2 instance.
3. If you want to do additional troubleshooting about an unsuccessful deployment, you can use a technique like View Instance Details (p. 128). You can also try analyzing the deployment log files on an Amazon EC2 instance. For more information, see Investigating Deployment Failures on Instances (p. 199).

## Verify Your Deployment

Once your deployment has succeeded, verify that your WordPress installation is now working. To verify, use the Amazon EC2 instance's public DNS address, followed by `/WordPress`, to view your site in a web browser. (To get an Amazon EC2 instance's public DNS value, in the Amazon EC2 console, select the Amazon EC2 instance, and look for the **Public DNS** value in the **Description** tab.)

For example, if your Amazon EC2 instance's public DNS address is **`ec2-01-234-567-890.compute-1.amazonaws.com`**, you would use the following URL:

```
http://ec2-01-234-567-890.compute-1.amazonaws.com/WordPress
```

# Step 5: Update and Redeploy Your WordPress Application

Now that you've successfully deployed your application revision, try making an update on the development machine to the WordPress code and then redeploy the site using AWS CodeDeploy. After redeployment, on the Amazon EC2 instance you should be able to see the changes that you made in the code on the development machine.

**Topics**



## Set Up the WordPress Site

To see the effects of the code change, first finish setting up the WordPress site so that you have a fully-functional installation:

1. Enter your site's URL into your web browser if you haven't already done so. The URL is the Amazon EC2 instance's public DNS address plus a `/WordPress` extension. In the case of this example WordPress site (and example Amazon EC2 instance public DNS address), the URL was **`http://ec2-01-234-567-890.compute-1.amazonaws.com/WordPress`**.
2. If you haven't set the site up yet, you'll be presented with the WordPress default configuration page. Click **Create a Configuration File** to continue with setup.
3. When you get to the database configuration page, enter the following values to use the default MySQL database:

   - **Database Name**: **`test`**

- **User Name**: `root`
- **Password**: (leave blank)
- **Database Host**: `localhost`
- **Table Prefix**: `wp_`

Click **Submit** to finish setting up the database.

4. Follow the on-screen directions to continue setting up the site. When you get to the **Welcome** page, fill in any values you want, and click **Install WordPress**. When installation completes, you will be able to log in to your dashboard.

   That's all the WordPress configuration you'll need to do right now. Next, update the code.

## Modify the Site

To modify the WordPress site, go back to your development machine, and go to your application's folder. For example:

```
cd /tmp/WordPress
```

Modify some of the site's colors by changing `#000` to `#768331` in the `wp-content/themes/twentyfourteen/style.css` file. You can do this using a text editor or with **sed**.

On Linux or other systems with GNU **sed**, use:

```
sed -i 's/#000/#768331/g' wp-content/themes/twentyfourteen/style.css
```

On Mac OS X, Unix or other systems with BSD **sed**, use:

```
sed -i '' 's/#000/#768331/g' wp-content/themes/twentyfourteen/style.css
```

## Redeploy the Site

Now that you've modified the site's code, redeploy the site using Amazon S3 and AWS CodeDeploy.

Bundle and upload the changes to Amazon S3 as explained in Bundle the Application's Files Into a Single Archive File, and Push the Archive File (p. 25) (note that as you follow those instructions, you do not need to create a new application, as you already did this earlier). Give the new revision the same key as before (**`WordPressApp.zip`**). Upload it to the same Amazon S3 bucket that you created earlier (for example, **`CodeDeployDemoBucket`**).

Then redeploy the site as follows, using the AWS CLI, the AWS Management Console, or the AWS CodeDeploy APIs.

**Topics**

## To use the AWS CLI to redeploy the site

Call the **create-deployment** command to create a new deployment based on the newly-uploaded revision, again using the application named **`WordPress_App`**, the deployment configuration named **`CodeDeployDefault.OneAtATime`**, the deployment group named **`WordPress_DepGroup`**, and the revision named **`WordPressApp.zip`** in the bucket named **`CodeDeployDemoBucket`**:

```
aws deploy create-deployment \
 --application-name WordPress_App \
 --deployment-config-name CodeDeployDefault.OneAtATime \
 --deployment-group-name WordPress_DepGroup \
 --s3-location bucket=CodeDeployDemoBucket,bundleType=zip,key=WordPressApp.zip
```

You can check the status of the new deployment as described in Monitor (and Troubleshoot) Your Deployment (p. 28).

When AWS CodeDeploy has finished redeploying the site, revisit the site in your web browser to verify that the colors have been changed. (You may need to refresh your browser.) If the colors have been changed, then congratulations, you've modified and redeployed your site!

## To use the AWS Management Console to redeploy the site

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).
2. In the AWS CodeDeploy service navigation bar, click **Deployments**.
3. Click **Create New Deployment**. The **Create New Deployment** page appears.
4. On the **Create New Deployment** page:

   1. In the **Choose Application** list, select **WordPress_App**.

      **Tip**
      If no entries are displaying but you expect to see some, make sure that the correct corresponding region is selected. On the navigation bar, in the region selector, click one of the supported regions (p. 192). AWS CodeDeploy supports only certain regions.
   2. In the **Choose Deployment Group** list, select **WordPress_DepGroup**.
   3. With the option **My application is stored in Amazon S3** already selected next to **Revision Type**, copy your revision's Amazon S3 link into the **Revision Location** box. To find the link value:

      1. In a separate browser tab, sign in to the Amazon S3 console:

         Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

         Browse to and open **CodeDeployDemoBucket**. Then select your new revision, named **`WordPressApp.zip`**, in the Amazon S3 console.
      2. If the **Properties** pane is not visible in the Amazon S3 console, click the **Properties** button.
      3. In the **Properties** pane, copy the value of the **Link** field into the **Revision Location** box in the AWS CodeDeploy console.

      After you type the revision location and leave the **Revision Location** box, the **File Type** list appears.

4. In the **File Type** list, if a message appears stating that the file type could not be detected, select **.zip**.
5. Leave the **Deployment Description** box blank.
6. In the **Deployment Config** list, select **CodeDeployDefault.OneAtATime**, and then click **Deploy Now**. The **Deployments** page appears, showing information about the newly created deployment.

   To update the deployment's current status, click the refresh button next to the table.

   You can check the status of the new deployment as described in Monitor (and Troubleshoot) Your Deployment (p. 28).

   When AWS CodeDeploy has finished redeploying the site, revisit the site in your web browser to verify that the colors have been changed. (You may need to refresh your browser.) If the colors have been changed, then congratulations, you've modified and redeployed your site!

# Step 6: Clean Up Your WordPress Application and Related Resources

You've now successfully made an update on the development machine to the WordPress code and then redeployed the site using AWS CodeDeploy. If you want to avoid ongoing charges for resources that you created to complete this tutorial, you should delete any AWS CloudFormation stacks (or terminate any Amazon EC2 instances, if you manually created them outside of AWS CloudFormation) that you created just for this tutorial. You should also delete any Amazon S3 buckets that you created just for this tutorial, as well the `WordPress_App` application in AWS CodeDeploy itself.

To perform the cleanup, you can use the AWS CLI, the AWS Management Console, or the AWS APIs.

**Topics**



## To use the AWS CLI to clean up

1. If you used our AWS CloudFormation template for this tutorial, delete the associated AWS CloudFormation stack by calling the **delete-stack** command against the stack named **`CodeDeployDemoStack`**. This also terminates all accompanying Amazon EC2 instances and then deletes all accompanying IAM roles that the stack originally created:

   ```
   aws cloudformation delete-stack --stack-name CodeDeployDemoStack
   ```

2. To delete the Amazon S3 bucket, call the **rm** command with the **--recursive** switch against the bucket named **`CodeDeployDemoBucket`**, which also deletes all objects in the bucket as well as the bucket itself:

   ```
   aws s3 rm s3://CodeDeployDemoBucket --recursive
   ```

3. To delete the `WordPress_App` application from AWS CodeDeploy, call the **delete-application** command, which also deletes all associated deployment group records and deployment records in AWS CodeDeploy for the application:

```
aws deploy delete-application --application-name WordPress_App
```

4. If you did *not* use the AWS CloudFormation stack for this tutorial, to terminate any individual Amazon EC2 instances that you may have manually created for this tutorial, call the **terminate-instances** command, supplying the ID of the Amazon EC2 instance to terminate:

```
aws ec2 terminate-instances --instance-ids instanceId
```

## To use the AWS Management Console to clean up

If you used our AWS CloudFormation template for this tutorial, delete the associated AWS CloudFormation stack:

1. Sign in to the AWS Management Console and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. In the **By Name** box, type the AWS CloudFormation stack name that you created earlier, such as **`CodeDeployDemoStack`**.
3. Click the stack name.
4. Click **Delete Stack**. AWS CloudFormation deletes the stack, terminates all accompanying Amazon EC2 instances, and deletes all accompanying IAM roles.

To terminate any individual Amazon EC2 instances that you may have created outside of an AWS CloudFormation stack:

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Instances** area, click **Instances**.
3. In the **Search Instances** box, type the name of any Amazon EC2 instance that you want to terminate, such as **`CodeDeployDemo`**, and then press Enter.
4. Click the Amazon EC2 instance name.
5. Click **Actions**, point to **Instance State**, and then click **Terminate**. When prompted, click **Yes, Terminate**. Amazon EC2 terminates the Amazon EC2 instance. Repeat these steps for any additional Amazon EC2 instances.

To delete the Amazon S3 bucket:

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **All Buckets** list, browse to and click the name of the Amazon S3 bucket that you created earlier, such as **`CodeDeployDemoBucket`**.
3. Before you can delete a bucket, you must first delete its contents: select all of the files in the bucket, such as **`WordPressApp.zip`**. Click **Actions**, and then click **Delete**. When prompted to confirm the deletion, click **OK**. Amazon S3 deletes the bucket's contents.
4. After the bucket is empty, you can delete the bucket: click **All Buckets**. In the **All Buckets** list, right-click the bucket name, such as **`CodeDeployDemoBucket`**, and then click **Delete**. When prompted to confirm the deletion, click **OK**. Amazon S3 deletes the bucket.

To delete the `WordPress_App` application from AWS CodeDeploy:

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).
2. In the AWS CodeDeploy service navigation bar, click **Applications**. The **Applications** page is displayed.
3. In the list of applications, click **WordPress_App**.
4. Click **Delete Application**. When prompted to confirm the deletion, click **Delete**. AWS CodeDeploy deletes the `WordPress_App` application and returns you to the **Applications** page.

## What's Next?

If you've arrived here, then you should have successfully completed a deployment with AWS CodeDeploy and have even updated your site's code and then redeployed it using AWS CodeDeploy. Congratulations!

To learn even more about AWS CodeDeploy, see AWS CodeDeploy Concepts (p. 58).

# Getting Started by Deploying a "Hello, World!" Application with AWS CodeDeploy (Windows Server and Windows)

This tutorial helps you build experience and confidence with AWS CodeDeploy by guiding you through the deployment of a sample application to a single Amazon EC2 instance. In this tutorial, you will deploy a single web page to a single Windows Server Amazon EC2 instance running Internet Information Server (IIS) as its web server. This web page will display a simple "Hello, World!" message.

If you want to practice deploying to an Amazon Linux Amazon EC2 instance instead of a Windows Server Amazon EC2 instance, see WordPress Deployment (Amazon Linux EC2) (p. 17).

If you want to practice deploying to an on-premises instance instead of an Amazon EC2 instance, see On-Premises Instance Deployment (Windows Server or Ubuntu Server) (p. 50).

**Note**
This tutorial builds on concepts that were introduced in the Create Deployment Walkthrough (p. 8). If you have not yet completed the *Create Deployment Walkthrough*, you may want to start there first.
This tutorial's steps are presented from a Windows perspective, as the accompanying tutorial targets Windows Server. While you can complete most of these steps on a local machine running Linux, OS X, or Unix, you will need to adapt some of the steps that cover Windows-based directory paths such as `c:\temp`. Also, if you want to connect to the Windows Server Amazon EC2 instance, you will need a client application that is capable of connecting through the Remote Desktop Protocol (RDP) to the Windows Server Amazon EC2 instance. (Windows includes an RDP connection client application by default.)

Before you start this tutorial, you must complete the prerequisites in Setting Up (p. 4), including configuring your IAM user, installing or upgrading the AWS CLI, and creating an IAM instance profile and a service role.

- Step 1: Launch an Amazon EC2 Instance (p. 35)
- Step 2: Configure your Source Content (p. 36)

# Step 1: Launch a Windows Server Amazon EC2 Instance

To deploy the "Hello, World!" application with AWS CodeDeploy, you'll need a Windows Server Amazon EC2 instance to deploy to.

You can launch a new Windows Server Amazon EC2 instance that's properly configured to participate in AWS CodeDeploy deployments by following the instructions in Configure Instances (p. 72). In those instructions, when you get to the part about assigning an Amazon EC2 instance tag to the instance, be sure to specify the tag key of **`Name`** and the tag value of **`CodeDeployDemo`**. (If you specify a different tag key or tag value than this one for the Amazon EC2 instance that you'll use in this tutorial, then the instructions later in Step 4: Deploy Your Application (p. 41) may produce unexpected results.)

After you've followed those instructions to launch the new Amazon EC2 instance, return to this page, and continue on to the next section to practice connecting to your newly-launched Amazon EC2 instance, if you wish.

> **Note**
> The preceding Amazon EC2 instance launch instructions direct you to continue on to Create an Application (p. 111) as a next step. Ignore those directions and continue on instead with the next section to practice connecting to your Amazon EC2 instance after you launch it, if you wish.

## Connect to Your Windows Server Amazon EC2 Instance

After your new Amazon EC2 instance is launched, you can practice connecting to it, if you wish, by following these instructions. (See also Connecting to Your Windows Instance Using RDP.)

> **Note**
> The following instructions assume that you are running Windows and the Windows Desktop Connection client application. You may need to adapt these instructions for other operating systems or other RDP connection client applications.

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **Instances**, click **Instances**.
3. Browse to and select your Windows Server Amazon EC2 instance in the list of Amazon EC2 instances.
4. Click **Connect**.
5. Click **Get Password**.
6. Click **Browse**. Browse to and select your Amazon EC2 instance key pair file that is associated with the Windows Server Amazon EC2 instance, and then click **Open**.
7. Click **Decrypt Password**. Make a note of the password that is displayed.
8. Click **Download Remote Desktop File**, and then open the file.
9. If prompted to connect even though the publisher of the remote connection can't be identified, click to proceed.
10. When prompted for a password, enter the password that you noted earlier, and then click to proceed. (If your RDP connection client application prompts you for a user name, enter **`Administrator`**.)
11. If prompted to connect even though the identify of the remote computer cannot be verified, click to proceed.

12. Once you are successfully connected, the Windows Server Amazon EC2 instance displays its Desktop.
13. You can now log out of the running Amazon EC2 instance, if you want to.

    **Caution**
    Do not stop or terminate the Amazon EC2 instance. Otherwise, AWS CodeDeploy won't be able to deploy to it.

# Step 2: Configure Your Source Content to Deploy to the Windows Server Amazon EC2 Instance

Now that you have set up the Amazon EC2 instance, it's time to configure your application's source content so that you have something that you can deploy to the Amazon EC2 instance. For this tutorial, you'll deploy a single web page to the Windows Server Amazon EC2 instance, which will run Internet Information Server (IIS) as its web server. This web page will display a simple "Hello, World!" message.

**Topics**



## Create the Web Page

1. Create a subdirectory (subfolder) named `HelloWorldApp` in your `c:\temp` folder. Then switch to that folder:

```
mkdir c:\temp\HelloWorldApp
cd c:\temp\HelloWorldApp
```

   **Note**
   You don't have to use the location of `c:\temp` or the subfolder name of `HelloWorldApp`. You can use whatever location or subfolder name that you want. However, if you use a different location or subfolder name, be sure to substitute your location or subfolder name for ours throughout this tutorial.

2. Use a text editor (such as Notepad) to create a file inside of the folder. Name the file `index.html`:

```
notepad index.html
```

3. Add the following HTML code to the file, and then save the file:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "ht
tp://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <title>Hello, World!</title>
  <style>
    body {
      color: #ffffff;
      background-color: #0188cc;
      font-family: Arial, sans-serif;
```

```
      font-size:14px;
    }
  </style>
</head>
<body>
  <div align="center"><h1>Hello, World!</h1></div>
  <div align="center"><h2>You have successfully deployed an application using
 AWS CodeDeploy</h2></div>
  <div align="center">
    <p>What to do next? Take a look through the <a href="ht
tp://docs.aws.amazon.com/codedeploy">AWS CodeDeploy Documentation</a>.</p>
  </div>
</body>
</html>
```

## Create a Script to Run Your Application

Next, you will create a script that AWS CodeDeploy will use to set up the web server on the target Amazon EC2 instance.

1. In the same subfolder as the `index.html` file, use a text editor (such as Notepad) to create another file. Name the file `before-install.bat`:

```
notepad before-install.bat
```

2. Add the following batch script code to the file, and then save the file:

```
REM Install Internet Information Server (IIS).
c:\Windows\Sysnative\WindowsPowerShell\v1.0\powershell.exe -Command Import-
Module -Name ServerManager
c:\Windows\Sysnative\WindowsPowerShell\v1.0\powershell.exe -Command Install-
WindowsFeature Web-Server
```

## Add an Application Specification File

Next, you will add an Application Specification File alongside the web page and the batch script file. The Application Specification File (AppSpec file) is a YAML-formatted file that AWS CodeDeploy uses to describe what AWS CodeDeploy should install onto your Amazon EC2 instance, as well as what scripts must be run at various times during a deployment. The AppSpec file:

- Maps the source files in your application revision to their destinations on the instance.
- Specifies scripts to be run on the instance at various stages of the deployment process.
- Must be named `appspec.yml` and must be placed in the application's source code's root folder.

1. In the same subfolder as the `index.html` file and the `before-install.bat` file, use a text editor (such as Notepad) to create another file. Name the file `appspec.yml`:

```
notepad appspec.yml
```

2. Add the following YAML code to the file, and then save the file:

```
version: 0.0
os: windows
files:
  - source: \index.html
    destination: c:\inetpub\wwwroot
hooks:
  BeforeInstall:
    - location: \before-install.bat
      timeout: 900
```

AWS CodeDeploy will use this AppSpec file to copy the `index.html` file in the application source code's root folder to the `c:\inetpub\wwwroot` folder on the target Amazon EC2 instance. During the deployment, AWS CodeDeploy will run the `before-install.bat` batch script on the target Amazon EC2 instance during the **`BeforeInstall`** deployment lifecycle event. If this script takes longer than 900 seconds (15 minutes) to run, AWS CodeDeploy will stop the deployment at that point and mark the deployment to the Amazon EC2 instance as failed.

For more information about each of these settings, see the AppSpec File Reference (p. 212).

> **Important**
> The locations and numbers of spaces between each of the items in this file is important. If the spacing is incorrect, AWS CodeDeploy will raise an error that may be difficult to debug. For more information, see AppSpec file Spacing (p. 223).

# Step 3: Upload Your "Hello, World!" Application to Amazon S3

Now that you've configured your source content, you will prepare and upload it to a location that AWS CodeDeploy can deploy the source content from. The following instructions show you how to provision an Amazon S3 bucket, prepare the application revision's files for the bucket, bundle the revision's files, and then push the revision to the bucket.

> **Note**
> You can also use AWS CodeDeploy to deploy applications from GitHub repositories to instances. However, deploying from GitHub repositories is not covered in this tutorial. To learn more, see GitHub Integration (p. 177).

**Topics**



## Provision an Amazon S3 Bucket

Create a new storage container, or *bucket*, in Amazon S3—or use an existing bucket. Also, make sure that the bucket allows both uploads and downloads, that you can upload the revision to the bucket, and that participating Amazon EC2 instances can download the revision from the bucket.

Use the AWS CLI, the Amazon S3 console, or the Amazon S3 APIs to create a new Amazon S3 bucket. After you create the bucket, make sure to give both the bucket and your IAM user the proper access permissions.

**Topics**



## To use the AWS CLI to create a new Amazon S3 bucket

Call the **mb** command to create an Amazon S3 bucket named **`CodeDeployDemoBucket`**, for example:

```
aws s3 mb s3://CodeDeployDemoBucket
```

> **Note**
> Bucket names must be unique across Amazon S3 for all AWS accounts. If you aren't able to use **`CodeDeployDemoBucket`**, try a different bucket name, such as **`CodeDeployDemoBucket`** followed by a dash and your initials or some other unique identifier. Then be sure to substitute your bucket name for **`CodeDeployDemoBucket`** wherever you see it throughout the rest of this tutorial.
> Also, note that the Amazon S3 bucket must be created in the same AWS region as the region that your target Amazon EC2 instances are launched in. For example, if you create the bucket in the US East (N. Virginia) region, then your target Amazon EC2 instances must be launched in the US East (N. Virginia) region too.

## To use the Amazon S3 console to create a new Amazon S3 bucket

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the Amazon S3 console, click **Create Bucket**.
3. In the **Bucket Name** box, type a name for the bucket, such as **`CodeDeployDemoBucket`**.

   > **Note**
   > Bucket names must be unique across Amazon S3 for all AWS accounts. If you aren't able to use **`CodeDeployDemoBucket`**, try a different bucket name, such as **`CodeDeployDemoBucket`** followed by a dash and your initials or some other unique identifier. Then be sure to substitute your bucket name for **`CodeDeployDemoBucket`** wherever you see it throughout the rest of this tutorial.
   > Also, note that the Amazon S3 bucket must be created in the same AWS region as the region that your target Amazon EC2 instances are launched in. For example, if you create the bucket in the US East (N. Virginia) region, then your target Amazon EC2 instances must be launched in the US East (N. Virginia) region too.
4. In the **Region** list, select the target region.
5. Click **Create**. Amazon S3 creates the new bucket.

## Give the Amazon S3 bucket and your IAM user the proper permissions

The Amazon S3 bucket must allow you to upload to it. One way to specify this is through an Amazon S3 bucket policy. For example, the following Amazon S3 bucket policy allows AWS account `111122223333` to upload anywhere within the Amazon S3 bucket named `CodeDeployDemoBucket`:

```
{
  "Statement": [
    {
      "Action": ["s3:PutObject"],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::CodeDeployDemoBucket/*",
```

```
      "Principal": {
        "AWS": [
          "111122223333"
        ]
      }
    }
  ]
}
```

Although you will not be deploying the revision in this step, the Amazon S3 bucket must also allow download requests coming from each participating Amazon EC2 instance. While you are working with the bucket, now is a good time to verify this. One option is to specify this is through an Amazon S3 bucket policy. For example, the following Amazon S3 bucket policy allows any Amazon EC2 instance with an attached IAM instance profile containing the ARN `arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo` to download from anywhere within the Amazon S3 bucket named `CodeDeployDemoBucket`:

```
{
  "Statement": [
    {
      "Action": ["s3:Get*", "s3:List*"],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::CodeDeployDemoBucket/*",
      "Principal": {
        "AWS": [
          "arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo"
        ]
      }
    }
  ]
}
```

To learn how to generate and attach an Amazon S3 bucket policy, see Bucket Policy Examples.

Also, your account must have permission to upload the revision to the Amazon S3 bucket. One way to specify this is through an IAM policy. For example, the following IAM policy allows your IAM user to upload revisions anywhere within the Amazon S3 bucket named `CodeDeployDemoBucket`:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":["s3:PutObject"],
      "Resource":"arn:aws:s3:::CodeDeployDemoBucket/*"
    }
  ]
}
```

To learn how to create and attach an IAM policy, see Managing IAM Policies (AWS Management Console).

## Prepare the Application's Files for the Bucket

Make sure that the web page that you created earlier, along with the AppSpec file and the script that you also created earlier, are organized on your development machine similar to the following:

```
c:\
  |-- temp\
        |--HelloWorldApp\
             |-- appspec.yml
             |-- before-install.bat
             |-- index.html
```

## Bundle the Application's Files Into a Single Archive File, and Push the Archive File

Next, you need to bundle these 3 files into an archive file (known as an application *revision*).

**Note**
You may be charged for storing objects in a bucket and for transferring application revisions into and out of a bucket. For more information, see Amazon S3 Pricing.

1. On the development machine, switch to the folder where the files are stored, if you aren't there already:

   ```
   cd c:\temp\HelloWorldApp
   ```

   **Note**
   If you don't switch to this folder, then when you bundle the files together, the file bundling will start at whatever point your current folder is. For example, if your current folder is `c:\temp` instead of `c:\temp\HelloWorldApp`, then you will start bundling files and subfolders that are in the `c:\temp` folder, which may include more than just the `HelloWorldApp` subfolder.

2. Call the **create-application** command to register a new application with AWS CodeDeploy named **`HelloWorld_App`**:

   ```
   aws deploy create-application --application-name HelloWorld_App
   ```

3. Call the AWS CodeDeploy push command to bundle the files together, upload the revisions to Amazon S3, and register information with AWS CodeDeploy about the uploaded revision, all in one action. For example:

   ```
   aws deploy push --application-name HelloWorld_App --s3-location
   s3://CodeDeployDemoBucket/HelloWorld_App.zip --ignore-hidden-files
   ```

   This command bundles the files from the current directory (excluding any hidden files) into a single archive file named `HelloWorld_App.zip`, uploads the revision to the **`CodeDeployDemoBucket`** bucket, and registers information with AWS CodeDeploy about the uploaded revision.

# Step 4: Deploy Your "Hello, World!" Application

Now you will deploy the sample "Hello, World!" application revision that you've previously uploaded to Amazon S3. You will use the AWS CLI or the AWS CodeDeploy console to deploy the revision and to monitor the deployment's progress. After the deployment succeeds, you will check the results.

**Topics**

# Deploy Your Application Revision with AWS CodeDeploy

**Topics**



## To use the AWS CLI to deploy your application revision

1. First, the deployment will need a corresponding deployment group. However, before you create the deployment group, you will need a service role ARN. A *service role* is a special type of IAM role that gives a service permission to act on your behalf. In this case, the service role will give AWS CodeDeploy permission to access your Amazon EC2 instances to expand (read) their Amazon EC2 instance tags.

   You should have already created a service role that compatible with AWS CodeDeploy by following the instructions in Use the AWS CLI to Create a Service Role that is Compatible with AWS CodeDeploy (p. 142). (If not, go there now and create the service role.) To get the service role ARN, see Use the AWS CLI to Get the Service Role ARN (p. 144).
2. Now that you have a service role ARN, call the **create-deployment-group** command to create a deployment group named **`HelloWorld_DepGroup`**, associated with the application named **`HelloWorld_App`**, using the Amazon EC2 instance tag named **`CodeDeployDemo`** and deployment configuration named **`CodeDeployDefault.OneAtATime`**, with the specified service role ARN:

   ```
   aws deploy create-deployment-group --application-name HelloWorld_App --de
   ployment-group-name HelloWorld_DepGroup --deployment-config-name
   CodeDeployDefault.OneAtATime --ec2-tag-filters Key=Name,Value=CodeDeploy
   Demo,Type=KEY_AND_VALUE --service-role-arn serviceRoleARN
   ```

   In the preceding command, *`serviceRoleARN`* is the service role ARN that you noted earlier.
3. Now call the **create-deployment** command to create a deployment associated with the application named **`HelloWorld_App`**, the deployment configuration named **`CodeDeployDefault.OneAtATime`** and the deployment group named **`HelloWorld_DepGroup`**, using the application revision named **`HelloWorld_App.zip`** in the bucket named **`CodeDeployDemoBucket`**:

   ```
   aws deploy create-deployment --application-name HelloWorld_App --deployment-
   config-name CodeDeployDefault.OneAtATime --deployment-group-name HelloWorld_De
   pGroup --s3-location bucket=CodeDeployDemoBucket,bundleType=zip,key=Hello
   World_App.zip
   ```

## To use the AWS CodeDeploy console to deploy your application revision

1. Before you start using the AWS CodeDeploy console to deploy your application revision, you will need a service role ARN. A *service role* is a special type of IAM role that gives a service permission to act on your behalf. In this case, the service role will give AWS CodeDeploy permission to access your Amazon EC2 instances to expand (read) their Amazon EC2 instance tags.

   You should have already created a service role that compatible with AWS CodeDeploy by following the instructions in Use the IAM Console to Create a Service Role that is Compatible with AWS

CodeDeploy (p. 140). (If not, go there now and create the service role.) To get the service role ARN, see Use the IAM Console to Get the Service Role ARN (p. 143).

2. Now that you have the service role ARN, you can start using the AWS CodeDeploy console to deploy your application revision:

   Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

3. If the **Applications** page is not visible, in the AWS CodeDeploy service navigation drop-down list, select **Applications**.
4. In the list of applications, click **HelloWorld_App**.
5. Under **Deployment Groups**, click **Create New Deployment Group**.
6. In the **Deployment Group Name** box, type **`HelloWorld_DepGroup`**.
7. In the list of tags, select **Amazon EC2** in the **Tag Type** drop-down list.
8. In the **Key** box, type **`Name`**.
9. In the **Value** box, type **`CodeDeployDemo`**.

   **Note**
   After you type **`CodeDeployDemo`**, a **1** should appear under **Instances** to confirm that AWS CodeDeploy found 1 matching Amazon EC2 instance.

10. In the **Deployment Config** drop-down list, select **CodeDeployDefault.OneAtATime**, if it is not already selected.
11. In the **Service Role ARN** drop-down list, select the service role ARN that you noted earlier, and then click **Create Deployment Group**.
12. In the AWS CodeDeploy service navigation drop-down list, select **Deployments**.
13. Click **Create New Deployment**.
14. In the **Application** drop-down list, select **HelloWorld_App**.
15. In the **Deployment Group** drop-down list, select **HelloWorld_DepGroup**.
16. With the option **My application is stored in Amazon S3** already selected next to **Revision Type**, in the **Revision Location** box, type the location of the sample "Hello, World!" application revision that you previously uploaded to Amazon S3. To get the location:

    1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
    2. In the **All Buckets** list, select **CodeDeployDemoBucket** (or whatever the name is of the bucket that you uploaded your application revision to earlier). A list of objects in that bucket appears.
    3. In the list of objects, select **HelloWorld_App.zip**.
    4. If the **Properties** pane is not visible, click the **Properties** button.
    5. In the **Properties** pane, copy the value of the **Link** field to your clipboard.

       It might look something like this:

       **`https://s3.amazonaws.com/CodeDeployDemoBucket/HelloWorld_App.zip`**
    6. Return to the AWS CodeDeploy console, and in the **Revision Location** box, paste the **Link** field value that you copied to your clipboard.

17. After you type the revision location and leave the **Revision Location** box, the **File Type** list appears.
18. In the **File Type** list, if a message appears stating that the file type could not be detected, select **.zip** in the list of file types.

19. If you want to type a comment about the deployment, enter the comment into the **Deployment Description** box. Otherwise, you can leave it blank.
20. In the **Deployment Config** drop-down list, select **CodeDeployDefault.OneAtATime**, if it is not already selected.
21. Click **Deploy Now**. The **Deployments** page appears, showing information about your newly created deployment.

    **Tip**
    To update the deployment's current status, click the refresh button next to the table.

# Monitor (and Troubleshoot) Your Deployment

**Topics**



## To use the AWS CLI to monitor (and troubleshoot) your deployment

1. Get the deployment's ID by calling the **list-deployments** command against the application named **`HelloWorld_App`** and the deployment group named **`HelloWorld_DepGroup`**:

```
aws deploy list-deployments --application-name HelloWorld_App --deployment-
group-name HelloWorld_DepGroup --query "deployments" --output text
```

2. Then call the **get-deployment** command to see if the deployment succeeded, supplying the deployment ID:

```
aws deploy get-deployment --deployment-id deploymentID --query "deploy
mentInfo.status" --output text
```

3. The **get-deployment** command will return the deployment's overall status. If successful, the value will be `Succeeded`.

   If, however, the overall status is `Failed`, you can try further troubleshooting by calling commands such as list-deployment-instances and get-deployment-instance.

   For even more troubleshooting options, see Investigating Deployment Failures on Instances (p. 199).

## To use the AWS CodeDeploy console to monitor (and troubleshoot) your deployment

On the **Deployments** page in the AWS CodeDeploy console, you can monitor your deployment's current status by looking at the value of the **Status** column for your deployment.

**Tip**
To update the deployment's current status, click the refresh button next to the table.

To get more information about your deployment, especially if the **Status** column value has any value other than **Succeeded**:

1. In the **Deployments** table, click the arrow next to the deployment ID. **Details** and **Instances** areas appear for the deployment. After a deployment fails, a message should appear in the **Details** area that describes why the deployment failed.

**Note**
If you don't see an **Instances** area, click the refresh button next to the table. After the **Status** column changes from **InProgress** to **Created**, the **Instances** area should then appear.

2. In the **Instances** area, click **View All Instances**. Additional information about the deployment appears. After a deployment fails, you may be able to determine on which Amazon EC2 instances the deployment failed and at which step the deployment failed on each failed Amazon EC2 instance.
3. If you want to do additional troubleshooting about an unsuccessful deployment, you can use a technique like View Instance Details (p. 128). You can also try analyzing the deployment log files on a instance. For more information, see Investigating Deployment Failures on Instances (p. 199).

## Verify Your Deployment

Once your deployment has succeeded, verify that your WordPress installation is now working. To verify, use the Amazon EC2 instance's public DNS address to view the web page in a web browser. (To get an Amazon EC2 instance's public DNS value, in the Amazon EC2 console, select the Amazon EC2 instance, and look for the **Public DNS** value in the **Description** tab.)

For example, if your Amazon EC2 instance's public DNS address is **`ec2-01-234-567-890.compute-1.amazonaws.com`**, you would use the following URL:

```
http://ec2-01-234-567-890.compute-1.amazonaws.com/WordPress
```

If successful, you should see a "Hello, World!" web page.

# Step 5: Update and Redeploy Your "Hello, World!" Application

Now that you've successfully deployed your application revision, try making an update on the development machine to the web page's code and then redeploy the site using AWS CodeDeploy. After redeployment, on the Amazon EC2 instance you should be able to see the changes that you made in the code on the development machine.

**Topics**



## Modify the Web Page

1. Go to your `c:\temp\HelloWorldApp` subfolder, if you're not there already. Once there, use a text editor (such as Notepad) to modify the `index.html` file:

```
cd c:\temp\HelloWorldApp
notepad index.html
```

2. Revise the contents of the `index.html` file to change the web page's background color and some of the text on the web page, and then save the file:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "ht
tp://www.w3.org/TR/html4/loose.dtd">
<html>
```

```
<head>
  <title>Hello Again, World!</title>
  <style>
    body {
      color: #ffffff;
      background-color: #66cc00;
      font-family: Arial, sans-serif;
      font-size:14px;
    }
  </style>
</head>
<body>
  <div align="center"><h1>Hello Again, World!</h1></div>
  <div align="center"><h2>You have successfully deployed a revision of an
application using AWS CodeDeploy</h2></div>
  <div align="center">
    <p>What to do next? Take a look through the <a href="ht
tp://docs.aws.amazon.com/codedeploy">AWS CodeDeploy Documentation</a>.</p>
  </div>
</body>
</html>
```

# Redeploy the Site

Now that you've modified the web page's code, redeploy the web page using Amazon S3 and AWS CodeDeploy.

Bundle and upload the changes to Amazon S3 as explained in Bundle the Application's Files Into a Single Archive File, and Push the Archive File (p. 41) (note that as you follow those instructions, you do not need to create a new application, as you already did this earlier). Give the new revision the same key as before (**`HelloWorld_App.zip`**). Upload it to the same Amazon S3 bucket that you created earlier (for example, **`CodeDeployDemoBucket`**).

Then redeploy the site as follows, using the AWS CLI or the AWS Management Console.

**Topics**



## To use the AWS CLI to redeploy the site

Call the **create-deployment** command to create a new deployment based on the newly-uploaded revision, again using the application named **`HelloWorld_App`**, the deployment configuration named **`CodeDeployDefault.OneAtATime`**, the deployment group named **`HelloWorld_DepGroup`**, and the revision named **`HelloWorld_App.zip`** in the bucket named **`CodeDeployDemoBucket`**:

```
 aws deploy create-deployment --application-name HelloWorld_App --deployment-
config-name CodeDeployDefault.OneAtATime --deployment-group-name HelloWorld_De
pGroup --s3-location bucket=CodeDeployDemoBucket,bundleType=zip,key=Hello
World_App.zip
```

You can check the status of the new deployment as described in Monitor (and Troubleshoot) Your Deployment (p. 44).

When AWS CodeDeploy has finished redeploying the site, revisit the site in your web browser to verify that the background color and the text on the web page have been changed. (You may need to refresh your browser.) If the background color and the text has been changed, then congratulations, you've modified and redeployed your site!

## To use the AWS Management Console to redeploy the site

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. In the AWS CodeDeploy service navigation bar, click **Deployments**.
3. Click **Create New Deployment**. The **Create New Deployment** page appears.
4. On the **Create New Deployment** page:

   1. In the **Choose Application** list, select **HelloWorld_App**.

      **Tip**
      If no entries are displaying but you expect to see some, make sure that the correct corresponding region is selected. On the navigation bar, in the region selector, click one of the supported regions (p. 192). AWS CodeDeploy supports only certain regions.

   2. In the **Choose Deployment Group** list, select **HelloWorld_DepGroup**.
   3. With the option **My application is stored in Amazon S3** already selected next to **Revision Type**, copy your revision's Amazon S3 link into the **Revision Location** box. To find the link value:

      1. In a separate browser tab, sign in to the Amazon S3 console:

         Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

         Browse to and open **CodeDeployDemoBucket**. Then select your new revision, named **`HelloWorld_App.zip`**, in the Amazon S3 console.
      2. If the **Properties** pane is not visible in the Amazon S3 console, click the **Properties** button.
      3. In the **Properties** pane, copy the value of the **Link** field into the **Revision Location** box in the AWS CodeDeploy console.

      After you type the revision location and leave the **Revision Location** box, the **File Type** list appears.
   4. In the **File Type** list, if a message appears stating that the file type could not be detected, select **.zip**.
   5. Leave the **Deployment Description** box blank.
   6. In the **Deployment Config** list, select **CodeDeployDefault.OneAtATime**, and then click **Deploy Now**. The **Deployments** page appears, showing information about the newly created deployment.

      To update the deployment's current status, click the refresh button next to the table.

      You can check the status of the new deployment as described in Monitor (and Troubleshoot) Your Deployment (p. 44).

      When AWS CodeDeploy has finished redeploying the site, revisit the site in your web browser to verify that the background color and the text on the web page have been changed. (You may need to refresh your browser.) If the background color and the text has been changed, then congratulations, you've modified and redeployed your site!

# Step 6: Clean Up Your "Hello, World!" Application and Related Resources

You've now successfully made an update on the development machine to the "Hello, World!" code and then redeployed the site using AWS CodeDeploy. If you want to avoid ongoing charges for resources that you created to complete this tutorial, you should delete any AWS CloudFormation stacks (or terminate any Amazon EC2 instances, if you manually created them outside of AWS CloudFormation) that you created just for this tutorial. You should also delete any Amazon S3 buckets that you created just for this tutorial, as well the `HelloWorld_App` application in AWS CodeDeploy itself.

To perform the cleanup, you can use the AWS CLI, the AWS Management Console, or the AWS APIs.

**Topics**



## To use the AWS CLI to clean up

1. If you used the AWS CloudFormation stack for this tutorial, delete the stack by calling the **delete-stack** command against the stack named **`CodeDeployDemoStack`**. This also terminates all accompanying Amazon EC2 instances and then deletes all accompanying IAM roles that the stack originally created:

   ```
   aws cloudformation delete-stack --stack-name CodeDeployDemoStack
   ```

2. To delete the Amazon S3 bucket, call the **rm** command with the **--recursive** switch against the bucket named **`CodeDeployDemoBucket`**, which also deletes all objects in the bucket as well as the bucket itself:

   ```
   aws s3 rm s3://CodeDeployDemoBucket --recursive
   ```

3. To delete the `HelloWorld_App` application from AWS CodeDeploy, call the **delete-application** command, which also deletes all associated deployment group records and deployment records in AWS CodeDeploy for the application:

   ```
   aws deploy delete-application --application-name HelloWorld_App
   ```

4. If you did *not* use the AWS CloudFormation stack for this tutorial, to terminate any individual Amazon EC2 instances that you may have manually created for this tutorial, call the **terminate-instances** command, supplying the ID of the Amazon EC2 instance to terminate:

   ```
   aws ec2 terminate-instances --instance-ids instanceId
   ```

## To use the AWS Management Console to clean up

If you used our AWS CloudFormation template for this tutorial, delete the associated AWS CloudFormation stack:

1. Sign in to the AWS Management Console and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. In the **By Name** box, type the AWS CloudFormation stack name that you created earlier, such as **`CodeDeployDemoStack`**.
3. Click the stack name.
4. Click **Delete Stack**. AWS CloudFormation deletes the stack, terminates all accompanying Amazon EC2 instances, and deletes all accompanying IAM roles.

To terminate any individual Amazon EC2 instances that you may have created outside of an AWS CloudFormation stack:

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Instances** area, click **Instances**.
3. In the **Search Instances** box, type the name of any Amazon EC2 instance that you want to terminate, such as **`CodeDeployDemo`**, and then press Enter.
4. Click the Amazon EC2 instance name.
5. Click **Actions**, point to **Instance State**, and then click **Terminate**. When prompted, click **Yes, Terminate**. Amazon EC2 terminates the Amazon EC2 instance. Repeat these steps for any additional Amazon EC2 instances.

To delete the Amazon S3 bucket:

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **All Buckets** list, browse to and click the name of the Amazon S3 bucket that you created earlier, such as **`CodeDeployDemoBucket`**.
3. Before you can delete a bucket, you must first delete its contents: select all of the files in the bucket, such as **`HelloWorld_App.zip`**. Click **Actions**, and then click **Delete**. When prompted to confirm the deletion, click **OK**. Amazon S3 deletes the bucket's contents.
4. After the bucket is empty, you can delete the bucket: click **All Buckets**. In the **All Buckets** list, right-click the bucket name, such as **`CodeDeployDemoBucket`**, and then click **Delete**. When prompted to confirm the deletion, click **OK**. Amazon S3 deletes the bucket.

To delete the `HelloWorld_App` application from AWS CodeDeploy:

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).
2. In the AWS CodeDeploy service navigation bar, click **Applications**. The **Applications** page is displayed.
3. In the list of applications, click **HelloWorld_App**.
4. Click **Delete Application**. When prompted to confirm the deletion, click **Delete**. AWS CodeDeploy deletes the `HelloWorld_App` application and returns you to the **Applications** page.

# What's Next?

If you've arrived here, then you should have successfully completed a deployment with AWS CodeDeploy and have even updated your site's code and then redeployed it using AWS CodeDeploy. Congratulations!

To learn even more about AWS CodeDeploy, see AWS CodeDeploy Concepts (p. 58).

# Getting Started by Deploying an Application to an On-Premises Instance with AWS CodeDeploy (Windows Server or Ubuntu Server)

This tutorial helps you build experience and confidence with AWS CodeDeploy by guiding you through the deployment of a sample application revision to a single on-premises instance—that is, a physical device that is not an Amazon EC2 instance—running Windows Server or Ubuntu Server. For information about on-premises instances and how they work with AWS CodeDeploy, see On-Premises Instances (p. 69).

If you want to practice deploying to an Amazon Linux Amazon EC2 instance instead of an on-premises instance, see WordPress Deployment (Amazon Linux EC2) (p. 17).

If you want to practice deploying to a Windows Server Amazon EC2 instance instead of an on-premises instance, see HelloWorld Deployment (Windows Server EC2) (p. 34).

> **Note**
> This tutorial builds on concepts that were introduced in the Create Deployment Walkthrough (p. 8). If you have not yet completed the *Create Deployment Walkthrough*, you may want to start there first.

**Topics**



## Prerequisites

Before you start this tutorial, you must complete the prerequisites in Setting Up (p. 4), including configuring your IAM user, installing or upgrading the AWS CLI, and creating a service role. (Note that you do not have to create an IAM instance profile as described in the prerequisites, as on-premises instances do not use IAM instance profiles.)

Also, the physical device that you want to configure as an on-premises instance must be running one of the supported operating systems (p. 206).

## Step 1: Configure the On-Premises Instance

Before you can try deploying to your on-premises instance, you must configure it so that AWS CodeDeploy can identify the correct on-premises instance to try deploying to. To configure your on-premises instance, follow the instructions in Use an Existing On-Premises Instance (p. 91), and then return to this page.

# Step 2: Create a Sample Application Revision

In this step, you'll create a sample application revision to deploy to your on-premises instance. While you can certainly try deploying any compatible application revision that you want to, the following instructions show you how to get started with a very simple application revision.

Since it is difficult to know which software and features are already installed—or that may be allowed to be installed by your organization's policies—on your on-premises instance, the sample application revision that we offer here simply uses a set of batch scripts (for Windows Server) or shell scripts (for Ubuntu Server) to write a series of text files to a specific location on your on-premises instance. One file is written for each of several AWS CodeDeploy deployment lifecycle events, including **Install**, **AfterInstall**, **ApplicationStart**, and **ValidateService**. Prior to these 4 deployment lifecycle events, during the **BeforeInstall** deployment lifecycle event, a script runs to make sure that any old files that were written during previous deployments with this sample are removed, as well as creating a predictable location on your on-premises instance to write the new files to.

**Note**
This sample application revision may fail to be deployed if any of the following are true:

- The user account that starts the AWS CodeDeploy Agent on the on-premises instance doesn't have permission to execute scripts on your on-premises instance.
- The environment variables that are described in this step either do not exist on your on-premises instance, are set to invalid values, or are set to directories (folders) that the user account that starts the AWS CodeDeploy Agent on the on-premises instance doesn't have read-write permissions to.
- The user account that starts the AWS CodeDeploy Agent on the on-premises instance doesn't have permission to create or delete folders in the locations that are listed in the scripts that are described in this step.
- The user account that starts the AWS CodeDeploy Agent on the on-premises instance doesn't have permission to create new text files in the locations that are listed in the scripts that are described in this step.

**Tip**
If you configured a Windows Server instance for use with this tutorial, and you want to deploy a different sample than the one that is listed in this section, you may want to try using the sample in Step 2: Configure your Source Content (p. 36) as part of the HelloWorld Deployment (Windows Server EC2) (p. 34) tutorial.
Currently there is no alternative sample that we provide for Ubuntu Server.

1. Switch to a directory (folder) on your development machine that's convenient for you, create a subdirectory (subfolder) named `CodeDeployDemo-OnPrem` to store the sample application revision's files, and then switch to the newly-created subfolder. For this example, we'll assume that you'll use the `c:\temp` folder as the root folder for Windows Server or the `/tmp` folder as the root folder for Ubuntu Server. If you choose to use a different folder than `c:\temp` for Windows Server or `/tmp` for Ubuntu Server, be sure to substitute your folder for ours throughout the rest of this tutorial:

   For Windows:

   ```
   mkdir c:\temp\CodeDeployDemo-OnPrem
   cd c:\temp\CodeDeployDemo-OnPrem
   ```

   For Linux, OS X, or Unix:

   ```
   mkdir /tmp/CodeDeployDemo-OnPrem
   cd /tmp/CodeDeployDemo-OnPrem
   ```

2. In the root of the `CodeDeployDemo-OnPrem` subfolder, use your preferred text editor to create 2 files named `appspec.yml` and `install.txt`, as follows:

   `appspec.yml` for Windows Server:

```
version: 0.0
os: windows
files:
  - source: .\install.txt
    destination: c:\temp\CodeDeployExample
hooks:
  BeforeInstall:
    - location: .\scripts\before-install.bat
      timeout: 900
  AfterInstall:
    - location: .\scripts\after-install.bat
      timeout: 900
  ApplicationStart:
    - location: .\scripts\application-start.bat
      timeout: 900
  ValidateService:
    - location: .\scripts\validate-service.bat
      timeout: 900
```

   `appspec.yml` for Ubuntu Server:

```
version: 0.0
os: linux
files:
  - source: ./install.txt
    destination: /tmp/CodeDeployExample
hooks:
  BeforeInstall:
    - location: ./scripts/before-install.sh
      timeout: 900
  AfterInstall:
    - location: ./scripts/after-install.sh
      timeout: 900
  ApplicationStart:
    - location: ./scripts/application-start.sh
      timeout: 900
  ValidateService:
    - location: ./scripts/validate-service.sh
      timeout: 900
```

   **Tip**
   For more information on AppSpec files, see Add an AppSpec File (p. 115) and AppSpec File Reference (p. 212).

   `install.txt`:

```
The Install deployment lifecycle event successfully completed.
```

3. Under the root of the `CodeDeployDemo-OnPrem` subfolder, create a `scripts` subfolder, and then switch to the newly-created subfolder:

For Windows:

```
mkdir c:\temp\CodeDeployDemo-OnPrem\scripts
cd c:\temp\CodeDeployDemo-OnPrem\scripts
```

For Linux, OS X, or Unix:

```
mkdir -p /tmp/CodeDeployDemo-OnPrem/scripts
cd /tmp/CodeDeployDemo-OnPrem/scripts
```

4. In the root of the `scripts` subfolder, use your preferred text editor to create 4 files named `before-install.bat`, `after-install.bat`, `application-start.bat`, and `validate-service.bat` for Windows Server, or `before-install.sh`, `after-install.sh`, `application-start.sh`, and `validate-service.sh` for Ubuntu Server, as follows:

   For Windows Server:

   `before-install.bat`:

   ```
   set FOLDER=%HOMEDRIVE%\temp\CodeDeployExample

   if exist %FOLDER% (
     rd /s /q "%FOLDER%"
   )

   mkdir %FOLDER%
   ```

   `after-install.bat`:

   ```
   cd %HOMEDRIVE%\temp\CodeDeployExample

   echo The AfterInstall deployment lifecycle event successfully completed. >
    after-install.txt
   ```

   `application-start.bat`:

   ```
   cd %HOMEDRIVE%\temp\CodeDeployExample

   echo The ApplicationStart deployment lifecycle event successfully completed.
    > application-start.txt
   ```

   `validate-service.bat`:

   ```
   cd %HOMEDRIVE%\temp\CodeDeployExample

   echo The ValidateService deployment lifecycle event successfully completed.
    > validate-service.txt
   ```

   For Ubuntu Server:

   `before-install.sh`:

```
#!/bin/bash
export FOLDER=/tmp/CodeDeployExample

if [ -d $FOLDER ]
then
 rm -rf $FOLDER
fi

mkdir -p $FOLDER
```

`after-install.sh`:

```
#!/bin/bash
cd /tmp/CodeDeployExample

echo "The AfterInstall deployment lifecycle event successfully completed."
 > after-install.txt
```

`application-start.sh`:

```
#!/bin/bash
cd /tmp/CodeDeployExample

echo "The ApplicationStart deployment lifecycle event successfully completed."
 > application-start.txt
```

`validate-service.sh`:

```
#!/bin/bash
cd /tmp/CodeDeployExample

echo "The ValidateService deployment lifecycle event successfully completed."
 > validate-service.txt

unset FOLDER
```

5. For Ubuntu Server only, make sure that the 4 shell scripts in the `scripts` folder have execute permissions:

```
chmod +x ./scripts/*
```

# Step 3: Bundle and Upload Your Application Revision to Amazon S3

Before you can try deploying your application revision, you'll need to bundle the files into the application revision and then upload the file bundle to an Amazon S3 bucket. To do this, complete the instructions in the Create an Application (p. 111) topic and then the Push a Revision (p. 118) topic. (Although you can name the application and deployment group anything you want, we recommend using `CodeDeploy-OnPrem-App` for the application name and `CodeDeploy-OnPrem-DG` for the deployment

group name so that you can more easily refer to them later.) After you have completed those instructions, return to this page.

**Note**
Alternatively, you could upload the file bundle to a GitHub repository and then deploy it from there. To learn how, see GitHub Integration (p. 177).

# Step 4: Deploy Your Application Revision

After you've uploaded your application revision to an Amazon S3 bucket, you can try deploying it to your on-premises instance. To do this, complete the instructions in Deploy a Revision (p. 121), and then return to this page.

**Note**
Alternatively, you could deploy the application revision from a GitHub repository to your on-premises instance. To learn how, see GitHub Integration (p. 177).

# Step 5: Validate and Verify Your Deployment

After you've tried deploying the application revision to your on-premises instance, you can validate whether the deployment succeeded. To do this, complete the instructions in View Deployment Details (p. 127), and then return to this page.

If the deployment succeeded, you can verify the deployment by logging on to the on-premises instance and checking for the presence of 4 text files in the `c:\temp\CodeDeployExample` folder (for Windows Server) or `/tmp/CodeDeployExample` (for Ubuntu Server).

If the deployment failed, you can try following the troubleshooting steps in View Instance Details (p. 128) and Troubleshooting Instance Issues (p. 198); then return to this page, make any needed fixes, re-bundle and re-upload your application revision, and then try deploying again.

# Step 6: Clean Up

If you want to avoid ongoing charges for resources that you created to complete this tutorial, you should delete the Amazon S3 bucket that you deployed your application revision from, if you'll no longer be using it. You can also clean up associated resources such as the application and deployment group records in AWS CodeDeploy and the on-premises instance itself.

To clean up the resources that you used for this tutorial, you can use the AWS CLI by itself or a combination of the AWS console and the AWS CLI.

## Use the AWS CLI to Clean Up

**To use the AWS CLI to delete the Amazon S3 bucket**

- Call the **rm** command along with the `--recursive` switch against the bucket represented here by *`your-bucket-name`* (for example, `CodeDeployDemoBucket`), which also deletes all objects in the bucket as well as the bucket itself:

```
aws s3 rm s3://your-bucket-name --recursive
```

### To use the AWS CLI to delete the application and deployment group records in AWS CodeDeploy

- Call the **delete-application** command against the application represented here by *your-application-name* (for example, `CodeDeploy-OnPrem-App`), which also deletes all associated deployment group records and deployment records in AWS CodeDeploy for the application:

```
aws deploy delete-application --application-name your-application-name
```

### To use the AWS CLI to deregister the on-premises instance and to delete the associated IAM user

- Call the **deregister** command against the on-premises instance represented here by *your-instance-name*, and the associated region represented here by *your-region*:

```
aws deploy deregister --instance-name your-instance-name --delete-iam-user
 --region your-region
```

> **Note**
> If you do not want to delete the IAM user that is associated with this on-premises instance, use the `--no-delete-iam-user` option instead of the `--delete-iam-user` option.

### To use the AWS CLI to uninstall the AWS CodeDeploy Agent and remove the special configuration file from the on-premises instance

- From the on-premises instance, call the **uninstall** command:

```
aws deploy uninstall
```

You have now completed all of the steps to clean up the resources that you used for this tutorial.

## Use the AWS Console to Clean Up

### To use the Amazon S3 console to delete the Amazon S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Click the icon next to the bucket that you want to delete (for example, `CodeDeployDemoBucket`). (Do not click the name of the bucket itself.)
3. Click **Actions**, and then click **Delete**.
4. When prompted about whether you really want to delete the bucket, click **OK**. The bucket is deleted.

### To use the AWS CodeDeploy console to delete the application and deployment group records in AWS CodeDeploy

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

**Note**
Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. If the list of applications is not visible, then on the service navigation bar, click **Applications**.
3. In the list of applications, click the name of the application that you want to delete (for example, `CodeDeploy-OnPrem-App`).
4. In the **Delete Application** area, click **Delete Application**.
5. When prompted about whether you really want to delete the application, click **Delete**. The application record and its associated deployment group records are deleted.

### To use the AWS CodeDeploy console to deregister the on-premises instance and to delete the associated IAM user

- You cannot use the AWS CodeDeploy console to deregister the on-premises instance. Use the AWS CLI instead, which also deletes the associated IAM user. For instructions, see To use the AWS CLI to deregister the on-premises instance and to delete the associated IAM user (p. 56).

### To uninstall the AWS CodeDeploy Agent and remove the special configuration file from the on-premises instance

- From the on-premises instance, use the AWS CLI to call the **uninstall** command:

```
aws deploy uninstall
```

# AWS CodeDeploy Concepts

This section provides conceptual details that are important to understand when using AWS CodeDeploy.

**Topics**



# AWS CodeDeploy Deployments

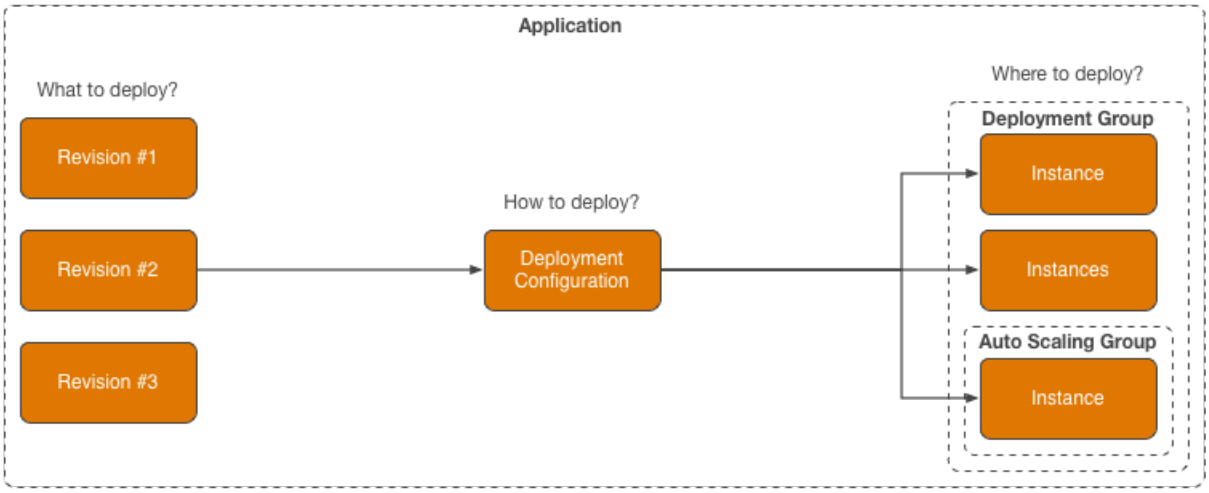
This page provides information about what deployments are in AWS CodeDeploy and how they work.

**Topics**



## Deployment Components

Deployments in AWS CodeDeploy consist of several components. Here is a diagram of these components and how they relate to each other:




**Application** – A name that uniquely identifies the application that you want to deploy. AWS CodeDeploy uses this name to ensure that the correct combination of revision, deployment configuration, and deployment group are being referenced during a deployment.

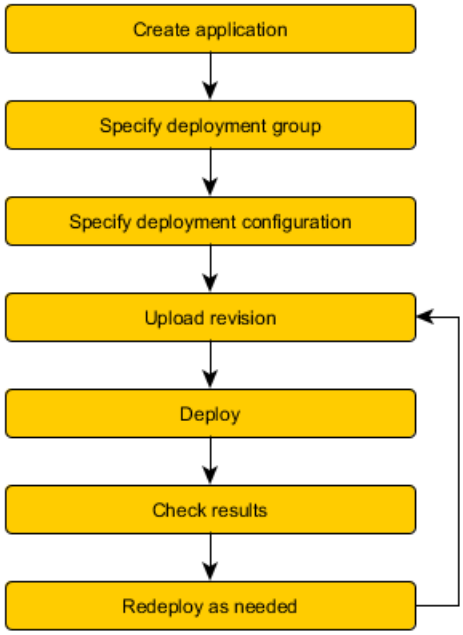
**Revision** – An archive file containing source content—such as source code, web pages, executable files, and deployment scripts—along with an Application Specification File (AppSpec file). Revisions are stored in Amazon S3 buckets or GitHub repositories. For Amazon S3, a revision is uniquely identified by its Amazon S3 object key and its ETag, version, or both. For GitHub, a revision is uniquely identified by its commit ID.

**Deployment configuration** – A set of deployment rules and deployment success and failure conditions that AWS CodeDeploy uses during a deployment.

**Deployment group** – A set of individual instances. A deployment group contains individually-tagged instances, Amazon EC2 instances in Auto Scaling groups, or both. For information on Amazon EC2 instance tags, see Working with Tags in the Console. For information on on-premises instances, see On-Premises Instances (p. 69). For information on Auto Scaling, see Auto Scaling Integration (p. 157).

# Deployment Workflow

Using AWS CodeDeploy typically works out into a set of defined steps. Here is a diagram of the major steps that all application revisions go through when they are deployed through AWS CodeDeploy:




These steps include:

1. Creating an application in AWS CodeDeploy by specifying a name that uniquely represents the application revisions that you want to deploy. During a deployment, AWS CodeDeploy uses this name to make sure that it is referencing the correct deployment components, such as the correct deployment group, deployment configuration, and application revision. For more information, see Create an Application (p. 111).
2. Specifying a deployment group in AWS CodeDeploy by specifying the set of instances that you want to deploy your application revisions to. You do this by specifying either the tags that have been applied to the instances, or the Auto Scaling group names that the Amazon EC2 instances belong to, or a combination of both. If you specify tags, AWS CodeDeploy deploys to instances that have at least one of the specified tags applied. These instances must be properly configured to participate in a deployment, such as being tagged or belong to an Auto Scaling group, and have the AWS CodeDeploy Agent installed and running. We provide you with an AWS CloudFormation template that you can use to quickly set up a new Amazon EC2 instance based on Amazon Linux or Windows Server that is preconfigured for AWS CodeDeploy to deploy application revisions to right away. We also provide you with the standalone AWS CodeDeploy Agent so that you can install it on existing Amazon Linux, Ubuntu Server, or Windows Server instances. For more information, see Create a Deployment Group (p. 138).
3. Specifying a deployment configuration in AWS CodeDeploy by determining how fast to deploy your application revisions to the instances and describing the success and failure conditions for the overall deployment. For more information, see View Deployment Configuration Details (p. 133).
4. Uploading an application revision to Amazon S3 or GitHub so that it can be deployed by AWS CodeDeploy. Along with the files that you want to deploy and any scripts that you want to run during the deployment, you include a file unique to AWS CodeDeploy that we call an *Application Specification File* (AppSpec file). This file contains a set of unique deployment instructions for AWS CodeDeploy to execute, such as where to copy the files onto each instance and at what point in time to run each of any included deployment scripts. For more information, see Prepare a Revision (p. 114).
5. Deploying your application revision through AWS CodeDeploy to the specified deployment group. The AWS CodeDeploy Agent on each participating instance in the deployment group copies your application revision from Amazon S3 or GitHub to the instance. The AWS CodeDeploy Agent then unbundles the revision and uses the AppSpec file to copy the files into the specified locations on the instance and execute any deployment scripts on the instance in the specified order. For more information, see Deploy a Revision (p. 121).

6. Checking the deployment results to see if the deployment completed as intended. For more information, see Monitor a Deployment (p. 127).
7. Redeploying a revision as needed. Let's say you want to redeploy an application revision through AWS CodeDeploy to the same deployment group, but you need to revise either the original source content, the AppSpec file, or the deployment scripts. You may want to do this for example if you need to fix a bug in the source content, or run the deployment scripts in a different order or in a different way, or otherwise address a failed deployment that you know was traceable to your end. To do this, you rebundle your revised source content, any deployment scripts, and the AppSpec file into a new revision, and then upload your new revision to the Amazon S3 bucket or GitHub repository. Then execute a new deployment to the same deployment group but by using the new revision. For more information, see Deploy a Revision (p. 121).

# Setting Up Instances

Before you can deploy application revisions to instances, you need to set up those instances. Let's say the application revision needs 3 production servers and 2 backup servers. You will therefore launch or use 5 instances. To use the instances with AWS CodeDeploy, the following requirements need to be satisfied:

1. Install the AWS CodeDeploy Agent on the instances. We support Amazon Linux, Ubuntu Server, and Windows Server instances for installing the AWS CodeDeploy Agent.
2. Enable tagging, if you are using tags to identify instances in a deployment group. AWS CodeDeploy relies on tags to identify and group instances into AWS CodeDeploy deployment groups. While we used both a key and a value in the *Getting Started* tutorials to define tags, AWS CodeDeploy allows the flexibility of choosing either a key or a value to define a tag for a deployment group as well.
3. Launch Amazon EC2 instances with a conforming IAM instance profile attached. The AWS CodeDeploy Agent needs an IAM instance profile to be attached to an Amazon EC2 instance as it is launched to verify the identity of the Amazon EC2 instance.
4. Create a service role. Ensure that AWS CodeDeploy has access to expand the tags in your AWS account by providing AWS CodeDeploy service access.

The preceding requirements must be done for manually provisioning existing instances for use by AWS CodeDeploy. For Amazon EC2 instances, our AWS CloudFormation template does all of this for your automatically; it creates and configures new, single Amazon EC2 instances based on Amazon Linux or Windows Server with the AWS CodeDeploy Agent preinstalled, for immediate use by AWS CodeDeploy. For more information, see Configure Instances (p. 72).

# Uploading Your Application Revision

After setting up the infrastructure, you can now focus on the application revision to be deployed:

1. In order to use AWS CodeDeploy, an AppSpec file must be placed under the root folder in your application's source content folder structure. AWS CodeDeploy relies on the AppSpec file to determine what to install onto the instances from your revision, as well as what user hooks to run in response to various events during a deployment. To learn more about the AppSpec file, see AppSpec Files (p. 64).
2. Bundle the application's source content folder structure into an archive file format such as zip, tar, or compressed tar. Then upload the archive file (known as an application *revision*) to an Amazon S3 bucket or GitHub repository of your choice.

   **Note**
   The tar and compressed tar archive file formats (.tar and .tar.gz) are not supported for Windows Server instances.

# Creating Your Application and Deployment Groups

An AWS CodeDeploy *deployment group* identifies a collection of instances based on their tags, or Auto Scaling group names, or both. Multiple application revisions can be deployed to the same instance, and an application revision can be deployed to multiple instances. For example, a tag of "Prod" could be added for the 3 production servers and "Backup" for the 2 backup servers. These two tags can be used to create two different deployment groups within the AWS CodeDeploy application, giving the ability to choose which set of servers (or both) should participate in a given deployment.

# Deploying Your Application Revision

Now you're ready to deploy your application revision from Amazon S3 or GitHub to the specified deployment group. The AWS CodeDeploy console makes this step rather simple. However, if the command line is more your style, the AWS CodeDeploy command create-deployment deploys a given revision to the specified instances. There are a number of parameters that you can specify to control your deployment including which revision to deploy, which deployment group to deploy to, and which deployment configuration to use.

# Updating Your Application

After the initial steps required for the first deployment, any subsequent updates to your application can be accomplished by pushing out a new revision and using the AWS CodeDeploy console or calling the **create-deployment** command again. This makes it easy to keep your application updated.

# Stopped and Failed Deployments

After a deployment starts, you can attempt to stop the deployment. You do this by using the AWS CodeDeploy console or by calling the stop-deployment command. When you attempt to stop the deployment, one of three things will happen:

- The deployment will successfully stop, and the operation will return a status of succeeded. In this case, no more deployment lifecycle events (such as before and after files are copied over to the instance, and so on) will be run on the deployment group for the stopped deployment. Because some deployment lifecycle events may have already run up to the time of the stop request, this means that some files may have already been copied to, and some scripts may have already run on, one or more of the instances in the deployment group.
- The deployment will not immediately stop, and the operation will return a status of pending. In this case, some deployment lifecycle events may still be running on the deployment group involved in the deployment. Because some deployment lifecycle events may have already run up to the time of the stop request, this means that some files may have already been copied to, and some scripts may have already run on, one or more of the instances in the deployment group. After the pending operation completes, subsequent calls to stop the deployment will return a status of succeeded.
- The deployment cannot successfully stop, and the operation will return an error. For a list of error conditions, see the "Error Codes" section of stop-deployment.

Deployment failures are similar to stopped deployments, in that overall failed deployments may result in some deployment lifecycle events having already been run on one or more of the instances in the deployment group before the deployment failed, depending on the rules of the accompanying deployment configuration. Because of this, some files may have already been copied to, and some scripts may have already run on, one or more of the instances in the deployment group, even though AWS CodeDeploy reports the overall deployment as a failure. To find out why a deployment might have failed for an instance in the deployment group, you can use the AWS CodeDeploy console, call the get-deployment-instance command, or analyze the related log files on the failed instance. For more information on AWS CodeDeploy log files, see Agent Cleanup (p. 69).

# Redeployments and Deployment Rollbacks

AWS CodeDeploy treats a redeployment as a brand new deployment of a previously-deployed revision. To redeploy a revision, see Deploy a Revision (p. 121).

AWS CodeDeploy does not directly support the concept of an automatic rollback of a deployment. That is, AWS CodeDeploy does not provide a way to completely uninstall anything that might have already been deployed to any of the instances in a deployment group and then redeploy some previous application revision to that deployment group. However, you can simulate a rollback with AWS CodeDeploy by creating a brand new deployment of a previously-deployed revision. To deploy a previously-deployed revision, see Deploy a Revision (p. 121).

If you remove an instance from a deployment group, AWS CodeDeploy does not uninstall anything that might have already been installed on that instance.

When you instruct AWS CodeDeploy to use the same application and deployment group information to do either a redeployment or a simulated rollback, AWS CodeDeploy first tries to remove from each participating instance all files that were last successfully installed. AWS CodeDeploy does this by checking the `/opt/codedeploy-agent/deployment-root/deployment-instructions/`*`deployment-group-ID`*`-cleanup` file (for Amazon Linux and Ubuntu Server instances) or the `C:\ProgramData\Amazon\CodeDeploy\deployment-instructions\`*`deployment-group-ID`*`-cleanup` file (for Windows Server instances). If such a cleanup file exists, AWS CodeDeploy uses the cleanup file's contents to remove from the instance all files listed in the cleanup file before starting the new deployment.

For example, a Windows Server Amazon EC2 instance may have the following first 2 text files and 2 script files already deployed to it, and the scripts created 2 more text files during various deployment lifecycle events:

```
c:\temp\a.txt (previously deployed by AWS CodeDeploy)
c:\temp\b.txt (previously deployed by AWS CodeDeploy)
c:\temp\c.bat (previously deployed by AWS CodeDeploy)
c:\temp\d.bat (previously deployed by AWS CodeDeploy)
c:\temp\e.txt (previously created by c.bat)
c:\temp\f.txt (previously created by d.bat)
```

However, the cleanup file will list only the first 2 text files and 2 script files, as they were installed by AWS CodeDeploy:

```
c:\temp\a.txt
c:\temp\b.txt
c:\temp\c.bat
c:\temp\d.bat
```

Before AWS CodeDeploy does the new deployment, AWS CodeDeploy will remove only the first 2 text files and the script files, leaving the last 2 text files untouched, as they were not installed by AWS CodeDeploy:

```
c:\temp\a.txt will be removed
c:\temp\b.txt will be removed
c:\temp\c.bat will be removed
c:\temp\d.bat will be removed
c:\temp\e.txt will remain
c:\temp\f.txt will remain
```

It's worth noting that as part of this process, AWS CodeDeploy will not try to revert or otherwise reconcile any actions taken by any scripts in previous deployments during subsequent redeployments or simulated rollbacks. For example, if the files `c.bat` and `d.bat` previously mentioned contain logic to not recreate the files `e.txt` and `f.txt` if those files already exist, then the old versions of `e.txt` and `f.txt` will remain untouched whenever AWS CodeDeploy runs `c.bat` and `d.bat` in subsequent deployments. To address this issue, you could add logic to `c.bat` and `d.bat` to always check whether old versions of `e.txt` and `f.txt` already exist, and if so, to delete those old files before creating new ones.

# AWS CodeDeploy Application Specification Files

An Application Specification File (AppSpec file) is a YAML-formatted file that is used in AWS CodeDeploy during a deployment to:

- Map the source files in your application revision to their destinations on the instance.
- Specify custom permissions for deployed files.
- Specify scripts to be run on each instance at various stages of the deployment process.

The AppSpec file is unique to AWS CodeDeploy. The AppSpec file is used to manage each deployment as a series of *deployment lifecycle events*. *Deployment lifecycle event hooks* allow you to optionally run one or more scripts on an instance following most deployment lifecycle events. You can run any type of script that is supported by the instance's operating system. You define deployment lifecycle event hooks in the AppSpec file. AWS CodeDeploy runs only those scripts that are specified in that file, but they can call other scripts on the instance.

> **Important**
> The AppSpec file must be present, well-formed, located at the root of your source content folder structure, and be named `appspec.yml`.

For information about how to create a well-formed AppSpec file and an example of one, see AppSpec File Reference (p. 212).

After you add an AppSpec file to your revision, you should be ready to push the revision and then deploy it from there. For instructions, see Push a Revision (p. 118) and Deploy a Revision (p. 121).

## How the AWS CodeDeploy Agent Uses the AppSpec file

If the AWS CodeDeploy Agent that is installed on the operating system doesn't match what is listed in the AppSpec file, the deployment will fail. Otherwise, it is used as follows:

- During the deployment steps, the AWS CodeDeploy Agent will look up the current event's name in the AppSpec file's **hooks** section. If it is not present, the AWS CodeDeploy Agent can move on to the next step. If the event is found in the **hooks** section, the AWS CodeDeploy Agent will retrieve the list of scripts to execute for the current step. The scripts are run sequentially in order until either all the scripts are complete, or one of the scripts exits with an error. The status of each script is logged in the AWS CodeDeploy Agent log file on the instance. For information about AWS CodeDeploy Agent log files, see AWS CodeDeploy Agent (p. 67).
- During the *Install* event, the AWS CodeDeploy Agent will use the mappings defined in the AppSpec file's **files** section to determine which folders or files to copy from the revision to which locations on the instance.

# AWS CodeDeploy Instance Health

AWS CodeDeploy monitors the health status of the instances in a deployment group, and fails deployments if the deployment group lacks a specified minimum number of healthy instances, referred to as the *minimum healthy instances* value.

## Health Status

AWS CodeDeploy assigns two health status values to each instance, *revision health* and *instance health*.

Revision health
: Revision health is based on the instance's currently installed application revision, and has the following status values:
  - Current: The instance's currently installed revision matches the revision for the deployment group's last successful deployment.
  - Old: The instance's currently installed revision matches an older version of the application.
  - Unknown: The instance has not successfully installed the application revision.

Deployment target health
: Deployment target health is based on whether deployments to an instance have succeeded or not, and has the following values:
  - Healthy: The last deployment to the instance succeeded.
  - Unhealthy: The attempt to deploy a revision to the instance failed, or a revision has not yet been deployed to the instance.

AWS CodeDeploy uses revision and instance health to schedule the deployment to the deployment group's instances in the following order:

1. Unhealthy instance health
2. Unknown revision health
3. Old revision health
4. Current revision health

If the overall deployment succeeds, the revision is updated and the deployment group's health status values are updated to reflect the latest deployment.

- All Current instances that had a successful deployment remain Current.

  Otherwise, they become Unknown.
- All Old or Unknown instances that had a successful deployment become Current.

  Otherwise, they remain Old or Unknown.
- All healthy instances that had a successful deployment remain healthy.

  Otherwise, they become unhealthy.
- All unhealthy instances that had a successful deployment become healthy.

  Otherwise, they remain unhealthy.

If the overall deployment fails or is stopped:

- Each instance that AWS CodeDeploy attempted to deploy the application revision to has its instance health set to healthy or unhealthy depending on whether the deployment attempt for that instance succeeded or failed.
- Each instance that AWS CodeDeploy did not attempt to deploy the application revision to retains its current instance health value.
- The deployment group's revision remains the same.

# Minimum Healthy Instances and Deployments

AWS CodeDeploy allow you to specify a minimum number of healthy instances for the deployment. Minimum healthy instances is used for two main purposes:

- To determine whether the overall deployment succeeds or fails. Deployment succeeds if the application revision was successfully deployed to at least the minimum number of healthy instances.
- To determine the number of instances that must be healthy during a deployment to allow the deployment to progress.

You can specify a minimum healthy instances value for your deployment group as the number of instances or for a percentage of the total. If you specify a percentage, then at the start of the deployment, AWS CodeDeploy converts the percentage to the equivalent number of instances, rounding up any fractional instances.

> **Note**
> For more information about how to set your deployment configuration, see create-deployment-config.

AWS CodeDeploy tracks the health status of the deployment group's instances during the deployment process and uses the deployment's minimum healthy instances value to determine whether to continue the deployment. The basic principle is: a deployment must never cause the number of healthy instances to fall below minimum healthy instances. The one exception to this rule is when a deployment group initially has less than minimum healthy instances. In that case, the deployment process does not reduce the number of healthy instances any further.

AWS CodeDeploy starts the deployment process by attempting to deploy the application revision to the deployment group's unhealthy instances. For each successful deployment, AWS CodeDeploy changes the instance's health status to healthy and adds it to the deployment group's healthy instances. AWS CodeDeploy then compares the current number of healthy instances to minimum healthy instances.

- If the number of healthy instances is less than or equal to minimum healthy instances, AWS CodeDeploy cancels the deployment to ensure that the number of healthy instances doesn't decrease with further deployments.
- If the number of healthy instances is greater than minimum healthy instances by at least 1, AWS CodeDeploy begins deploying the application revision to the original set of healthy instances.

If a deployment to a healthy instance fails, AWS CodeDeploy changes that instance's health status to unhealthy. As the deployment progresses, AWS CodeDeploy updates the current number of healthy instances and compares it to minimum healthy instances. If the number of healthy instances falls to minimum healthy instances at any point in the deployment process, AWS CodeDeploy stops the deployment. This practice avoids the possibility that the next deployment would fail, dropping the number of healthy instances below minimum healthy instances.

> **Tip**
> Make sure that your minimum healthy instances value is less than the total number of instances in the deployment group. If you specify a percentage value, remember that it will be rounded up. Otherwise, when the deployment starts, the number of healthy instances will already be less

than or equal to minimum healthy instances, and AWS CodeDeploy will immediately fail the deployment.

AWS CodeDeploy also uses minimum healthy instances and the number of healthy instances to determine whether and how to deploy the application revision to multiple instances. By default, AWS CodeDeploy deploys the application revision to as many instances as it can without any risk of having the number of healthy instances fall below minimum healthy instances. For example:

- If your deployment group has 10 instances and you set minimum healthy instances to 9, AWS CodeDeploy deploys to one instance at a time.
- If your deployment group has 10 instances and you set minimum healthy instances to 0, AWS CodeDeploy deploys to every instance at the same time.

Here are some specific examples, all of which assume a deployment group with 10 instances.

Minimum healthy instances: 95%
: AWS CodeDeploy rounds the minimum healthy instances value up to 10 instances, which equals the number of healthy instances. The overall deployment immediately fails without deploying the revision to any instances.

Minimum healthy instances: 9
: AWS CodeDeploy deploys the revision to one instance at a time. If any of these deployments fail, AWS CodeDeploy immediately fails the overall deployment.

Minimum healthy instances: 8
: AWS CodeDeploy deploys the revision to two instances at a time. If two of these deployment fail, AWS CodeDeploy immediately fails the overall deployment.

Minimum healthy instances: 0
: AWS CodeDeploy deploys the revision to the entire deployment group at once. The deployment group can't have less than 0 healthy instances, so the overall deployment cannot fail.

Minimum healthy instances: 9
: AWS CodeDeploy first deploys the revision to the unhealthy instance.
  - If any deployment fails, the number of healthy instances equals minimum healthy instances, so the overall deployment immediately fails.
  - If any deployment succeeds, the deployment group now has 10 healthy instances. AWS CodeDeploy continues the deployment, one instance at a time, until any deployment fails or the overall deployment is complete.

# AWS CodeDeploy Agent

The AWS CodeDeploy Agent is a software package that, when installed on an instance, enables that instance to participate in deployments through AWS CodeDeploy. The AWS CodeDeploy Agent is configurable, and it cleans up deployment artifacts and log files that it produces as part of its work.

To install the AWS CodeDeploy Agent on an existing instance, or to troubleshoot problems with the AWS CodeDeploy Agent on an instance, see AWS CodeDeploy Agent Operations (p. 205).

**Topics**



## Agent Configuration

When the AWS CodeDeploy Agent is installed on an instance, a configuration file is placed on the instance.

For Amazon Linux and Ubuntu Server instances, the configuration file is named `codedeployagent.yml` and is placed into the `/etc/codedeploy-agent/conf` directory.

For Windows Server instances, the configuration file is named `conf.yml` and is placed into the `C:\ProgramData\Amazon\CodeDeploy` directory.

This configuration file specifies certain directory paths for AWS CodeDeploy to use and behaviors for AWS CodeDeploy to follow as it interacts with the instance.

The configuration settings include:

- **:log_aws_wire:** – Set to `true` for the AWS CodeDeploy Agent to capture wire logs from Amazon S3 and write them to a file named `codedeploy-agent.wire.log` in the location pointed to by the **:logdir:** setting. The default setting is `false`. This setting applies to all instance types, however; you must manually add this configuration setting to Windows Server instances to be able to use it.

  **Caution**
  You should set **:log_aws_wire:** to `true` for only the amount of time that you need to actually capture wire logs. The `codedeploy-agent.wire.log` file can grow to a very large size in a relatively short amount of time. Also, the wire log output in the `codedeploy-agent.wire.log` file may contain sensitive information, including the plain-text contents of various files that were transferred into, or out of, Amazon S3 during the time that this setting was set to `true`. The wire logs contain information about all Amazon S3 activity associated with the AWS account while this setting was set to `true`, not just the Amazon S3 activity that relates to AWS CodeDeploy deployments.
- **:log_dir:** – The folder that the AWS CodeDeploy Agent uses to store related log files on the instance. The default setting is `'/var/log/aws/codedeploy-agent'` for Amazon Linux and Ubuntu Server instances, and `C:\ProgramData\Amazon\CodeDeploy\log` for Windows Server instances.
- **:pid_dir:** – The folder that the AWS CodeDeploy Agent uses to store a file named `codedeploy-agent.pid`. This file contains the process ID (PID) of the AWS CodeDeploy Agent. The default setting is `'/opt/codedeploy-agent/state/.pid'`. This setting applies only to Amazon Linux and Ubuntu Server instances.
- **:program_name:** – The AWS CodeDeploy Agent program name. The default setting is `codedeploy-agent`. This setting applies only to Amazon Linux and Ubuntu Server instances.
- **:root_dir:** – The folder that the AWS CodeDeploy Agent uses to store related revisions, deployment history, and deployment scripts on the instance. The default setting is `'/opt/codedeploy-agent/deployment-root'` for Amazon Linux and Ubuntu Server instances, and `C:\ProgramData\Amazon\CodeDeploy` for Windows Server instances.
- **:verbose:** – Set to `true` for the AWS CodeDeploy Agent to print debug messages to related log files on the instance. The default setting is `false` for Amazon Linux and Ubuntu Server instances, and `true` for Windows Server instances.
- **:wait_after_error:** – The number of seconds that the AWS CodeDeploy Agent waits after a polling error to poll AWS CodeDeploy again to see if there are pending deployments to run. The default setting is `1`.
- **:wait_between_runs:** – The number of seconds between times that the AWS CodeDeploy Agent polls AWS CodeDeploy to see if there are pending deployments to run. The default setting is `1`.
- **:on_premises_config_file:** – For on-premises instances, the path to an alternate location for the configuration file named `codedeploy.onpremises.yml` (for Ubuntu Server) or `conf.onpremises.yml` (for Windows Server). By default, these files are stored in `/etc/codedeploy-agent/conf/codedeploy.onpremises.yml` (for Ubuntu Server) or `C:\ProgramData\Amazon\CodeDeploy\conf.onpremises.yml` (for Windows Server).

## Agent Cleanup

As part of its work, the AWS CodeDeploy Agent archives revisions and log files on instances. The AWS CodeDeploy Agent cleans up these artifacts over time to conserve the instances' disk space.

Here's how the AWS CodeDeploy Agent cleans up these artifacts.

### Deployment Archive Cleanup

The AWS CodeDeploy Agent archives a maximum of 5 recent revisions per deployment group and deletes the rest. If there has been at least one successful deployment, one of these revisions is guaranteed to represent the last successfully installed revision to the instance.

### Log File Cleanup

For Amazon Linux and Ubuntu Server instances, the AWS CodeDeploy Agent rotates the log files under the `/var/log/aws/codedeploy-agent` folder. For Windows Server instances, the AWS CodeDeploy Agent rotates the log files under the `C:\ProgramData\Amazon\CodeDeploy\log` folder. The log file is rotated at 00:00:00 (instance time) daily. Log files older than 1 week are deleted. The rotated log files are named following the pattern `codedeploy-agent.`*`YYYYMMDD`*`.log` (for Amazon Linux and Ubuntu Server instances) or `codedeploy-agent-log.`*`YYYYMMDD`*`.txt` (for Windows Server instances).

# Using On-Premises Instances with AWS CodeDeploy

AWS CodeDeploy supports *on-premises instances*. An on-premises instance is any physical device that is not an Amazon EC2 instance, is able to successfully run the AWS CodeDeploy Agent, and can connect to public AWS service endpoints. You can use AWS CodeDeploy to deploy applications to on-premises instances. (This is in addition to being able to deploy applications to Amazon EC2 instances.) For example, you can use AWS CodeDeploy to simultaneously deploy an application to a set of Amazon EC2 instances running in the cloud along with a set of desktop PCs running in your office. (You could also deploy to just the Amazon EC2 instances or just the desktop PCs.)

## Comparing On-Premises Instances to Amazon EC2 Instances with AWS CodeDeploy

There are several similarities and differences between on-premises instances and Amazon EC2 instances to consider when using AWS CodeDeploy:

| Subject | On-Premises Instances | Amazon EC2 Instances |
|---|---|---|
| Requires you to install and run a version of the AWS CodeDeploy Agent that's compatible with the operating system that's running on the instance. | Yes | Yes |
| Requires the instance to be able to connect to the AWS CodeDeploy service. | Yes | Yes |

| Subject | On-Premises Instances | Amazon EC2 Instances |
|---|---|---|
| Requires an IAM instance profile (p. 84) to be attached to the instance, and the IAM instance profile must have the correct permissions to participate in AWS CodeDeploy deployments. | No | Yes |
| Requires you to create a separate IAM user for each participating instance, and requires you to store the IAM user's account credentials in plain text on the corresponding instance. | Yes | No |
| Requires you to register each participating instance with AWS CodeDeploy before you can deploy to it. | Yes | No |
| Requires you to tag each participating instance before AWS CodeDeploy can deploy to it. | Yes (by using on-premises instance tags) | Yes (by using Amazon EC2 tags) |
| Can participate in Auto Scaling and Elastic Load Balancing scenarios as part of AWS CodeDeploy deployments. | No | Yes |
| Can be deployed from both Amazon S3 buckets and GitHub repositories. | Yes | Yes |
| Is subject to being billed for associated deployments. | Yes | No |

# Deploying Applications with AWS CodeDeploy to On-Premises Instances

To deploy an AWS CodeDeploy application revision to an on-premises instance, you need to do just a few things:

1. Configure each on-premises instance, register each configured on-premises instance with AWS CodeDeploy, and then tag each configured on-premises instance. You can do this with the least amount of effort through a few special commands that the AWS Command Line Interface (AWS CLI) provides. Or you can also do this mostly on your own, following the steps that the special commands would have done for you. To configure an on-premises instance, register it with AWS CodeDeploy, and then tag it, follow the instructions in Use an Existing On-Premises Instance (p. 91).
2. Begin deploying application revisions to the on-premises instance. To experiment with creating and deploying a sample application revision to a properly configured and registered on-premises instance, see On-Premises Instance Deployment (Windows Server or Ubuntu Server) (p. 50).
3. If you don't want an on-premises instance to participate in deployments anymore, you could simply remove the corresponding on-premises instance tags from the associated deployment groups. However, a more robust approach is to remove the on-premises instance tags from the individual on-premises instance. You can also explicitly deregister an on-premises instance from AWS

CodeDeploy so that it is no longer capable of participating in any deployments at all, regardless of the on-premises instance tags that are associated with the on-premises instance. For instructions on removing on-premises instance tags and deregistering on-premises instances, see Next Steps / Advanced Tasks (p. 105).

# Launch or Configure Instances for AWS CodeDeploy

AWS CodeDeploy supports deployments to Amazon Linux, Ubuntu Server, and Windows Server instances. To launch or configure these types of instances to participate in deployments, follow these instructions:

| | |
|---|---|
| I want to launch a new Amazon Linux or Windows Server Amazon EC2 instance. | To launch the Amazon EC2 instance with the least amount of effort, see Use an AWS CloudFormation Template (p. 73).<br><br>To launch the Amazon EC2 instance mostly on your own, see Set Up a New Amazon EC2 Instance (p. 77). |
| I want to launch a new Ubuntu Server Amazon EC2 instance. | See Set Up a New Amazon EC2 Instance (p. 77). |
| I want to configure an existing Amazon Linux, Windows Server, or Ubuntu Server Amazon EC2 instance. | See Use an Existing Amazon EC2 Instance (p. 88). |
| I want to configure an existing Windows Server or Ubuntu Server on-premises instance (physical devices that are not Amazon EC2 instances). | See Use an Existing On-Premises Instance (p. 91). |

**Note**
To prepare Amazon EC2 instances in any Auto Scaling groups that you want to deploy to, you must follow some additional steps. For details, see Auto Scaling Integration (p. 157).

# Use an AWS CloudFormation Template to Launch a New Amazon EC2 Instance for AWS CodeDeploy

You can use our AWS CloudFormation template to quickly launch a new Amazon Linux or Windows Server Amazon EC2 instance that is properly configured to participate in AWS CodeDeploy deployments. (We currently do not provide an AWS CloudFormation template for Ubuntu Server Amazon EC2 instances.) You can use the AWS CLI, the AWS CodeDeploy console, or the AWS APIs to launch the new Amazon EC2 instance with the template. The template not only launches the new Amazon EC2 instance, but the template also instructs AWS CloudFormation to give the Amazon EC2 instance the correct permission to participate in AWS CodeDeploy deployments, tags the Amazon EC2 instance so that AWS CodeDeploy can find it during a deployment, and installs and runs the AWS CodeDeploy Agent on the Amazon EC2 instance so that it can actually participate in deployments.

**Note**
You don't have to use our AWS CloudFormation to set up an Amazon EC2 instance that AWS CodeDeploy can deploy applications to. For alternatives, see Configure Instances (p. 72).

**Important**
If you plan to use our AWS CloudFormation template to launch Amazon EC2 instances that are compatible with AWS CodeDeploy, the calling IAM user must have access to AWS CloudFormation and AWS services and actions that AWS CloudFormation depends on. If you have not yet followed the instructions in Setting Up (p. 4) to provision the calling IAM user, you must attach at least the following policy to the calling IAM user:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:*",
        "codedeploy:*",
        "ec2:*",
        "iam:AddRoleToInstanceProfile",
        "iam:CreateInstanceProfile",
        "iam:CreateRole",
        "iam:DeleteInstanceProfile",
        "iam:DeleteRole",
        "iam:DeleteRolePolicy",
        "iam:GetRole",
        "iam:PassRole",
        "iam:PutRolePolicy",
        "iam:RemoveRoleFromInstanceProfile"
      ],
      "Resource": "*"
    }
  ]
}
```

You can use either the AWS CLI or the AWS Management Console to launch a new Amazon EC2 instance that is properly configured to participate in AWS CodeDeploy deployments.

**Topics**

# Use the AWS CLI to Launch a New Amazon EC2 Instance With the AWS CloudFormation Template

The following instructions assume that you have already followed the instructions in Setting Up (p. 4) to install and configure the AWS CLI for use with AWS CodeDeploy.

1. Use our AWS CloudFormation template in a call to the **create-stack** command. This stack will launch a new Amazon EC2 instance with the AWS CodeDeploy Agent installed on it.

   Before you call the **create-stack** command, you must have an existing Amazon EC2 instance key pair to enable SSH access to the Amazon Linux Amazon EC2 instance or RDP access to the Windows Server Amazon EC2 instance, as indicated by placeholder *keyName* below. (Type only the Amazon EC2 instance key pair name in the command, such as ***myKeyName***. Do not type any Amazon EC2 instance key pair file extension, such as ***myKeyName*****.pem**.) To find an Amazon EC2 instance key pair name, open the Amazon EC2 console at https://console.aws.amazon.com/ec2. In the navigation pane, under **Network & Security**, click **Key Pairs**, and note the Amazon EC2 instance key pair name in the list. To generate a new Amazon EC2 instance key pair, see Creating Your Key Pair Using Amazon EC2. Be sure that your Amazon EC2 instance key pair is created in one of the supported regions (p. 192). Otherwise, you won't be able to use the Amazon EC2 instance key pair with AWS CodeDeploy.

   Now, call the **create-stack** command, as follows:

   To launch an Amazon Linux Amazon EC2 instance:

```
aws cloudformation create-stack \
  --stack-name CodeDeployDemoStack \
  --template-url templateURL \
  --parameters ParameterKey=InstanceCount,ParameterValue=1 ParameterKey=In
stanceType,ParameterValue=t1.micro \
    ParameterKey=KeyPairName,ParameterValue=keyName ParameterKey=Operating
System,ParameterValue=Linux \
   ParameterKey=SSHLocation,ParameterValue=0.0.0.0/0 ParameterKey=TagKey,Para
meterValue=Name \
    ParameterKey=TagValue,ParameterValue=CodeDeployDemo \
  --capabilities CAPABILITY_IAM
```

   To launch a Windows Server Amazon EC2 instance:

```
aws cloudformation create-stack --stack-name CodeDeployDemoStack --template-
url templateURL --parameters ParameterKey=InstanceCount,ParameterValue=1
ParameterKey=InstanceType,ParameterValue=t1.micro ParameterKey=KeyPair
Name,ParameterValue=keyName ParameterKey=OperatingSystem,ParameterValue=Win
dows ParameterKey=SSHLocation,ParameterValue=0.0.0.0/0 Parameter
Key=TagKey,ParameterValue=Name ParameterKey=TagValue,ParameterValue=CodeDeploy
Demo --capabilities CAPABILITY_IAM
```

   In the preceding commands, *templateURL* is one of the following:

- http://s3.amazonaws.com/aws-codedeploy-us-east-1/templates/latest/CodeDeploy_SampleCF_Template.json (for the US East (N. Virginia) region)
- http://s3-us-west-2.amazonaws.com/aws-codedeploy-us-west-2/templates/latest/CodeDeploy_SampleCF_Template.json (for the US West (Oregon) region)
- http://s3-eu-west-1.amazonaws.com/aws-codedeploy-eu-west-1/templates/latest/CodeDeploy_SampleCF_Template.json (for the EU (Ireland) region)
- http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/templates/latest/CodeDeploy_SampleCF_Template.json (for the Asia Pacific (Sydney) region)

This command creates a AWS CloudFormation stack named **`CodeDeployDemoStack`**, using the AWS CloudFormation template in the specified Amazon S3 bucket. (You can name the stack anything you want.) The Amazon EC2 instance is based on the t1.micro Amazon EC2 instance type (you can use any Amazon EC2 instance type you want), is tagged with the value **`CodeDeployDemo`** (you can tag the Amazon EC2 instance with any value that you want), and has the specified Amazon EC2 instance key pair applied.

2. Call the **describe-stacks** command to make sure that the AWS CloudFormation stack named **`CodeDeployDemoStack`** was successfully created:

```
aws cloudformation describe-stacks --stack-name CodeDeployDemoStack --query
 "Stacks[0].StackStatus" --output text
```

Do not proceed until the value `CREATE_COMPLETE` is returned.

To verify that the AWS CodeDeploy Agent is running on the Amazon EC2 instance and is ready to participate in deployments, see AWS CodeDeploy Agent Operations (p. 205). After you do this, the Amazon EC2 instance will be ready to participate in AWS CodeDeploy deployments. The next step is to proceed to Create an Application (p. 111).

# Use the AWS Management Console to Launch a New Amazon EC2 Instance With the AWS CloudFormation Template

Before you begin the following instructions, make sure that you have an existing Amazon EC2 instance key pair to enable SSH access to the Amazon Linux Amazon EC2 instance or RDP access to the Windows Server Amazon EC2 instance. (When you're prompted to specify the Amazon EC2 instance key pair name, type only the Amazon EC2 instance key pair name, such as ***`myKeyName`***. Do not type any Amazon EC2 instance key pair file extension, such as ***`myKeyName`*****`.pem`**.) To find an Amazon EC2 instance key pair name, open the Amazon EC2 console at https://console.aws.amazon.com/ec2. In the navigation pane, under **Network & Security**, click **Key Pairs**, and note the Amazon EC2 instance key pair name in the list. To generate a new Amazon EC2 instance key pair, see Creating Your Key Pair Using Amazon EC2. Be sure that your Amazon EC2 instance key pair is created in one of the supported regions (p. 192). Otherwise, you won't be able to use the Amazon EC2 instance key pair with AWS CodeDeploy.

1. Sign in to the AWS Management Console and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.

   **Important**
   Make sure that you are signed in to the AWS Management Console using the *same* account information that you used when setting up the AWS CLI in Setting Up (p. 4).
   Also, make sure that the correct region is selected. On the navigation bar, in the region selector, click one of the supported regions (p. 192). AWS CodeDeploy supports only certain regions.

2. Click **Create Stack**.
3. In the **Stack** area, in the **Name** box, type a name for the stack, such as **`CodeDeployDemoStack`** (you can name the stack anything you want).
4. In the **Template** area, click **Specify an Amazon S3 template URL**. In the box, type one of the following, and then click **Next**:

   - `http://s3.amazonaws.com/aws-codedeploy-us-east-1/templates/latest/CodeDeploy_SampleCF_Template.json` (for the US East (N. Virginia) region)
   - `http://s3-us-west-2.amazonaws.com/aws-codedeploy-us-west-2/templates/latest/CodeDeploy_SampleCF_Template.json` (for the US West (Oregon) region)
   - `http://s3-eu-west-1.amazonaws.com/aws-codedeploy-eu-west-1/templates/latest/CodeDeploy_SampleCF_Template.json` (for the EU (Ireland) region)
   - `http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/templates/latest/CodeDeploy_SampleCF_Template.json` (for the Asia Pacific (Sydney) region)

5. In the **Parameters** area, type the following, and then click **Next**:

   - In the **InstanceCount** box, type the number of Amazon EC2 instance that you want to launch (we recommend leaving the default of **1**).
   - In the **InstanceType** box, type the Amazon EC2 instance type that you want to launch (or leave the default of **t1.micro**).
   - In the **KeyPairName** box, type the existing Amazon EC2 instance key name.
   - In the **OperatingSystem** box, type **`Windows`** to launch Windows Server Amazon EC2 instances (or leave the default of **Linux** to launch Amazon Linux Amazon EC2 instances).
   - In the **SSHLocation** box, type the IP address range to use for connecting to the Amazon EC2 instance using SSH or RDP (or leave the default of **0.0.0.0/0**).

     **Important**
     As a best practice, we recommend restricting access to SSH (and HTTP) ports. Typing **`0.0.0.0/0`** here is for demonstration purposes only. AWS CodeDeploy does not require Amazon EC2 instances to have unrestricted access to ports. For more information, see Tips for Securing Your EC2 Instance.

   - In the **TagKey** box, type the Amazon EC2 instance tag key that AWS CodeDeploy will use to identify the Amazon EC2 instances during deployment (or leave the default of **Name**).
   - In the **TagValue** box, type the Amazon EC2 instance tag value that AWS CodeDeploy will use to identify the Amazon EC2 instances during deployment (or leave the default of **CodeDeployDemo**).

6. On the **Options** page, in the **Tags** area, leave the **Key** and **Value** boxes blank, and click **Next**.

   **Important**
   AWS CloudFormation tags are different from AWS CodeDeploy tags. AWS CloudFormation uses tags to simplify administration of your infrastructure. AWS CodeDeploy uses tags to identify Amazon EC2 instances. You already specified AWS CodeDeploy tags in the **Specify Parameters** page.

7. On the **Review** page, review the summary. In the **Capabilities** area, check the **I acknowledge that this template might cause AWS CloudFormation to create IAM resources** box, and then click **Create**.

   After AWS CloudFormation has created the stack and launched the Amazon EC2 instances, in the AWS CloudFormation console the **Status** column for your Amazon EC2 instances will show **CREATE_COMPLETE**. Note that this process could take several minutes.

To verify that the AWS CodeDeploy Agent is running on the Amazon EC2 instances and is ready to participate in deployments, see AWS CodeDeploy Agent Operations (p. 205). After you do this, the Amazon EC2 instances will be ready to participate in AWS CodeDeploy deployments. The next step is to proceed to Create an Application (p. 111).

# Set Up a New Amazon EC2 Instance to Work with AWS CodeDeploy

These instructions show you how to launch a new Amazon EC2 instance that is properly configured to start participating in AWS CodeDeploy deployments.

> **Note**
> You can use our AWS CloudFormation template to simplify launching a new Amazon Linux or Windows Server Amazon EC2 instance that is already configured to start participating in AWS CodeDeploy deployments. (We currently do not supply an AWS CloudFormation template to simply launching a new Ubuntu Server Amazon EC2 instance.) To learn about this and other alternatives, see Configure Instances (p. 72).

You can use the AWS CLI, the Amazon EC2 console, or the Amazon EC2 APIs to launch a new Amazon EC2 instance that is properly configured to start participating in AWS CodeDeploy deployments. As part of this process, you will create an IAM instance profile.

**Topics**



## Use the AWS CLI to Launch a New Amazon EC2 Instance

The following instructions assume that you have already followed the instructions in Setting Up (p. 4), including setting up and configuring the AWS CLI, as well as creating an IAM instance profile named `CodeDeployDemo-EC2-Instance-Profile`.

1. If you are creating a Windows Server Amazon EC2 instance, call the **create-security-group** and **authorize-security-group-ingress** commands to create a security group that allows RDP access (which is not allowed by default) and, alternatively, HTTP access:

```
aws ec2 create-security-group --group-name CodeDeployDemo-Windows-Security-
Group --description "For launching Windows Server images for use with AWS
CodeDeploy"
aws ec2 authorize-security-group-ingress --group-name CodeDeployDemo-Windows-
Security-Group --to-port 3389 --ip-protocol tcp --cidr-ip 0.0.0.0/0 --from-
port 3389
aws ec2 authorize-security-group-ingress --group-name CodeDeployDemo-Windows-
Security-Group --to-port 80 --ip-protocol tcp --cidr-ip 0.0.0.0/0 --from-
port 80
```

> **Tip**
> The preceding commands create a security group that allows unrestricted access to a Windows Server Amazon EC2 instance for RDP through port 3389 and alternatively HTTP

through port 80. This is for demonstration purposes only. As a best practice, we recommend restricting access to the RDP and HTTP ports. AWS CodeDeploy does not require unrestricted port access and does not require HTTP access. For more information, see Tips for Securing Your EC2 Instance.

2. On your development machine, create a file named `instance-setup.sh` (for Amazon Linux or Ubuntu Server Amazon EC2 instances) or `instance-setup.txt` (for Windows Server Amazon EC2 instances) with the following contents (you can name this file anything you want). As the Amazon EC2 instance is launching, this script will automatically download the corresponding AWS CodeDeploy Agent from the specified Amazon S3 location to the Amazon EC2 instance and then install the AWS CodeDeploy Agent on the Amazon EC2 instance.

   Here are the contents of the `instance-setup.sh` file for Amazon Linux Amazon EC2 instances:

```
#!/bin/bash
yum -y update
yum install -y aws-cli
cd /home/ec2-user
aws s3 cp s3://bucket-name/latest/install . --region region-name
chmod +x ./install
./install auto
```

   In the previous list of commands, *bucket-name* represents one of the following:

   - `aws-codedeploy-us-east-1` for instances in the US East (N. Virginia) region
   - `aws-codedeploy-us-west-2` for instances in the US West (Oregon) region
   - `aws-codedeploy-eu-west-1` for instances in the EU (Ireland) region
   - `aws-codedeploy-ap-southeast-2` for instances in the Asia Pacific (Sydney) region

   In the previous list of commands, *region-name* represents one of the following:

   - `us-east-1` for instances in the US East (N. Virginia) region
   - `us-west-2` for instances in the US West (Oregon) region
   - `eu-west-1` for instances in the EU (Ireland) region
   - `ap-southeast-2` for instances in the Asia Pacific (Sydney) region

   Here are the contents of the `instance-setup.sh` file for Ubuntu Server Amazon EC2 instances:

```
#!/bin/bash
apt-get -y update
apt-get -y install awscli
apt-get -y install ruby2.0
cd /home/ubuntu
aws s3 cp s3://bucket-name/latest/install . --region region-name
chmod +x ./install
./install auto
```

   In the previous list of commands, *bucket-name* represents one of the following:

   - `aws-codedeploy-us-east-1` for instances in the US East (N. Virginia) region
   - `aws-codedeploy-us-west-2` for instances in the US West (Oregon) region
   - `aws-codedeploy-eu-west-1` for instances in the EU (Ireland) region
   - `aws-codedeploy-ap-southeast-2` for instances in the Asia Pacific (Sydney) region

In the previous list of commands, *region-name* represents one of the following:

- `us-east-1` for instances in the US East (N. Virginia) region
- `us-west-2` for instances in the US West (Oregon) region
- `eu-west-1` for instances in the EU (Ireland) region
- `ap-southeast-2` for instances in the Asia Pacific (Sydney) region

Here are the contents of the `instance-setup.txt` file for Windows Server Amazon EC2 instances:

```
<powershell>
New-Item -Path c:\temp -ItemType "directory" -Force
Read-S3Object -BucketName bucket-name/latest -Key codedeploy-agent.msi -File
 c:\temp\codedeploy-agent.msi
Start-Process -Wait -FilePath c:\temp\codedeploy-agent.msi -WindowStyle
Hidden
</powershell>
```

In the previous list of commands, *bucket-name* represents one of the following:

- `aws-codedeploy-us-east-1` for instances in the US East (N. Virginia) region
- `aws-codedeploy-us-west-2` for instances in the US West (Oregon) region
- `aws-codedeploy-eu-west-1` for instances in the EU (Ireland) region
- `aws-codedeploy-ap-southeast-2` for instances in the Asia Pacific (Sydney) region

3. From the same directory where you created the `instance-setup.sh` or `instance-setup.txt` file, call the **run-instances** command to create and launch the new Amazon EC2 instance.

   Before you call this command, you will need the ID of an Amazon Machine Image (AMI) that you want to use for an Amazon Linux, Ubuntu Server, or Windows Server Amazon EC2 instance, represented by the placeholder *amiID* below. To get the ID, see Finding a Suitable AMI.

   You will also need the name of an existing Amazon EC2 instance key pair to enable SSH access to an Amazon Linux or Ubuntu Server Amazon EC2 instance or RDP access to a Windows Server Amazon EC2 instance, represented by the placeholder *keyName* below.

   **Important**
   Type only the key pair name, such as ***myKeyName***. Do not type any key pair file extension, such as ***myKeyName*****.pem**.

   To find an Amazon EC2 instance key pair name, open the Amazon EC2 console at https://console.aws.amazon.com/ec2. In the navigation pane, under **Network & Security**, click **Key Pairs**, and note the Amazon EC2 instance key pair name in the list. Be sure that the Amazon EC2 instance key pair is in one of the supported regions (p. 192). Otherwise, you won't be able to use the Amazon EC2 instance key pair with AWS CodeDeploy.

   To generate a new Amazon EC2 instance key pair, see Creating Your Key Pair Using Amazon EC2. Be sure that your Amazon EC2 instance key pair is created in one of the supported regions (p. 192). Otherwise, you won't be able to use the Amazon EC2 instance key pair with AWS CodeDeploy.

   Finally, you will need the name of the type of Amazon EC2 instance that you want to create, such as **`t1.micro`** (you can specify any Amazon EC2 instance type that you want). This is represented by the placeholder *instanceType* below. For a full list, see Amazon EC2 Instances.

To call the **run-instances** command to launch an Amazon Linux or Ubuntu Server Amazon EC2 instance (attaching the IAM instance profile that you created through Create an IAM Instance Profile (p. 84)):

```
aws ec2 run-instances \
  --image-id amiID \
  --key-name keyName \
  --user-data file://instance-setup.sh \
  --count 1 \
  --instance-type instanceType \
  --iam-instance-profile Name=CodeDeployDemo-EC2-Instance-Profile
```

**Tip**

The preceding command creates a default security group for the Amazon EC2 instance that allows access to several ports, including unrestricted access to the Amazon EC2 instance for SSH through port 22 and alternatively HTTP through port 80. This is for demonstration purposes only. As a best practice, we recommend restricting access, including restricting access to the SSH and HTTP ports only. AWS CodeDeploy does not require unrestricted port access and does not require HTTP port access. For more information, see Tips for Securing Your EC2 Instance.

To call the **run-instances** command to launch a Windows Server Amazon EC2 instance (attaching the IAM instance profile that you created through Create an IAM Instance Profile (p. 84)):

```
aws ec2 run-instances --image-id amiID --key-name keyName --user-data
file://instance-setup.txt --count 1 --instance-type instanceType --iam-in
stance-profile Name=CodeDeployDemo-EC2-Instance-Profile --security-groups
CodeDeployDemo-Windows-Security-Group
```

These commands launch a single Amazon EC2 instance with the specified AMI, Amazon EC2 instance key pair, and Amazon EC2 instance type, in the specified availability zone, with the specified IAM instance profile, and runs the specified script during launch.

4. In the output of the **run-instances** command, note the value of the `InstanceID`. If you forget this value, you can get it later by calling the **describe-instances** command against the specified Amazon EC2 instance key pair, for example:

```
aws ec2 describe-instances --filters "Name=key-name,Values=keyName" --query
 "Reservations[*].Instances[*].[InstanceId]" --output text
```

Finally, use the Amazon EC2 instance ID to call the **create-tags** command, which tags the Amazon EC2 instance so that AWS CodeDeploy can find it later during a deployment, for example using the Amazon EC2 instance tag named **`CodeDeployDemo`** (you can specify any Amazon EC2 instance tag that you want):

```
aws ec2 create-tags --resources instanceID --tags
Key=Name,Value=CodeDeployDemo
```

Do not proceed until the Amazon EC2 instance is launched and has passed all checks successfully. To verify this, use the Amazon EC2 instance ID to call the **describe-instance-status** command. After the Amazon EC2 instance has launched and has passed all checks successfully, the output will show as `ok`:

```
aws ec2 describe-instance-status --instance-ids instanceID --query "In
stanceStatuses[*].InstanceStatus.[Status]" --output text
```

To verify that the AWS CodeDeploy Agent is running on the Amazon EC2 instance and is ready to participate in deployments, see AWS CodeDeploy Agent Operations (p. 205), and then return to this page. After you do this, the Amazon EC2 instance will be ready to participate in AWS CodeDeploy deployments. The next step is to proceed to Create an Application (p. 111).

# Use the Amazon EC2 Console to Launch a New Amazon EC2 Instance

The following instructions assume that you have already followed the instructions in Setting Up (p. 4), including creating an IAM instance profile named `CodeDeployDemo-EC2-Instance-Profile`.

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

   **Important**
   Make sure that you are signed in to the AWS Management Console using the same account information (and using one of the supported regions (p. 192)) that you used when setting up the AWS CLI in Setting Up (p. 4).

2. In the navigation pane, under **Instances**, click **Instances**. Then click **Launch Instance**.
3. On the **Step 1: Choose an Amazon Machine Image** page, with the **Quick Start** tab already selected, choose an Amazon Machine Image (AMI). For Amazon Linux, next to the latest 64-bit Amazon Linux AMI, click **Select**. For Ubuntu Server, next to the latest 64-bit Ubuntu Server AMI, click **Select**. For Windows Server, we currently recommend an AMI such as Microsoft Windows Server 2012 R2; next to that AMI, click **Select**.
4. On the **Step 2: Choose an Instance Type** page, select any available Amazon EC2 instance type, and then click **Next: Configure Instance Details**.
5. On the **Step 3: Configure Instance Details** page, in the **IAM role** list, select the IAM instance profile that you created through Create an IAM Instance Profile (p. 84).

   **Note**
   If the **Network** list does not show **Launch into EC2-Classic**, and you are not able to select **Launch into EC2-Classic** or a default Virtual Private Cloud (VPC) in the list, or if you are not able to select a different Amazon EC2 instance type that supports launching into EC2-Classic, you must select an existing Amazon Virtual Private Cloud (VPC) and subnet, or click **Create new VPC** or **Create new subnet** or both to create a new VPC or subnet or both. For more information, see Your VPC and Subnets.

6. Expand **Advanced Details**.
7. Next to **User data**, with the **As text** option already selected, type the following in the box next to it to install the AWS CodeDeploy Agent as the Amazon EC2 instance is launching:

   To install the AWS CodeDeploy Agent for Amazon Linux, type this in the box:

```
#!/bin/bash
yum -y update
yum install -y aws-cli
cd /home/ec2-user
aws s3 cp s3://bucket-name/latest/install . --region region-name
chmod +x ./install
./install auto
```

In the previous list of commands, *bucket-name* represents one of the following:

- aws-codedeploy-us-east-1 for instances in the US East (N. Virginia) region
- aws-codedeploy-us-west-2 for instances in the US West (Oregon) region
- aws-codedeploy-eu-west-1 for instances in the EU (Ireland) region
- aws-codedeploy-ap-southeast-2 for instances in the Asia Pacific (Sydney) region

In the previous list of commands, *region-name* represents one of the following:

- us-east-1 for instances in the US East (N. Virginia) region
- us-west-2 for instances in the US West (Oregon) region
- eu-west-1 for instances in the EU (Ireland) region
- ap-southeast-2 for instances in the Asia Pacific (Sydney) region

To install the AWS CodeDeploy Agent for Ubuntu Server, type this in the box:

```
#!/bin/bash
apt-get -y update
apt-get -y install awscli
apt-get -y install ruby2.0
cd /home/ubuntu
aws s3 cp s3://bucket-name/latest/install . --region region-name
chmod +x ./install
./install auto
```

In the previous list of commands, *bucket-name* represents one of the following:

- aws-codedeploy-us-east-1 for instances in the US East (N. Virginia) region
- aws-codedeploy-us-west-2 for instances in the US West (Oregon) region
- aws-codedeploy-eu-west-1 for instances in the EU (Ireland) region
- aws-codedeploy-ap-southeast-2 for instances in the Asia Pacific (Sydney) region

In the previous list of commands, *region-name* represents one of the following:

- us-east-1 for instances in the US East (N. Virginia) region
- us-west-2 for instances in the US West (Oregon) region
- eu-west-1 for instances in the EU (Ireland) region
- ap-southeast-2 for instances in the Asia Pacific (Sydney) region

To install the AWS CodeDeploy Agent for Windows Server, type this in the box:

```
<powershell>
New-Item -Path c:\temp -ItemType "directory" -Force
Read-S3Object -BucketName bucket-name/latest -Key codedeploy-agent.msi -File
 c:\temp\codedeploy-agent.msi
Start-Process -Wait -FilePath c:\temp\codedeploy-agent.msi -WindowStyle
Hidden
</powershell>
```

In the previous list of commands, *bucket-name* represents one of the following:

- `aws-codedeploy-us-east-1` for instances in the US East (N. Virginia) region
- `aws-codedeploy-us-west-2` for instances in the US West (Oregon) region
- `aws-codedeploy-eu-west-1` for instances in the EU (Ireland) region
- `aws-codedeploy-ap-southeast-2` for instances in the Asia Pacific (Sydney) region

8. Leave the rest of the items on this page unchanged by clicking **Next: Add Storage**.
9. Leave the **Step 4: Add Storage** page unchanged by clicking **Next: Tag Instance**.
10. On the **Step 5: Tag Instance** page, with the **Key** box already showing **Name**, type **`CodeDeployDemo`** in the **Value** box (you can type any tag name that you want). This is referred to by AWS CodeDeploy in your deployment group. Then click **Next: Configure Security Group**.

    **Important**
    The contents of the **Key** and **Value** boxes are case-sensitive.

11. On the **Step 6: Configure Security Group** page, leave the **Create a new security group** option selected. For Amazon Linux or Ubuntu Server Amazon EC2 instances, a default SSH role will be configured; for Windows Server Amazon EC2 instances, a default RDP role will be configured.
12. If you want to open the HTTP port, click the **Add Rule** button, and in the **Type** drop-down list, select **`HTTP`**. Accept the default **Source** value of **Anywhere 0.0.0.0/0** and then click **Review and Launch**.

    **Tip**
    In a production environment, we recommend restricting access to the SSH, RDP, and HTTP ports, instead of specifying **Anywhere 0.0.0.0/0**. AWS CodeDeploy does not require unrestricted port access and does not require HTTP access; this is for demonstration purposes only. For more information, see Tips for Securing Your EC2 Instance.

    If a **Boot from General Purpose (SSD)** dialog box appears, follow the on-screen instructions, and then click **Next**.
13. Leave the contents of the **Step 7: Review Instance Launch** page unchanged by clicking **Launch**.
14. In the **Select an existing key pair or create a new key pair** dialog box, select either **Choose an existing key pair** or **Create a new key pair**. If you've already configured an Amazon EC2 instance key pair, you can select it here.

    If you don't already have an Amazon EC2 instance key pair configured, select **Create a new key pair** and give it a name, such as **`codedeploydemo`** (you can give it any name you want). Click **Download Key Pair** to download the Amazon EC2 instance key pair to your computer.

    **Important**
    You must have an Amazon EC2 instance key pair if you want to access your Amazon EC2 instance with SSH or RDP.

15. Click **Launch Instances**.
16. The next screen provides a link to your Amazon EC2 instance by its ID. Click it to see your Amazon EC2 instance launching in the Amazon EC2 console. Do not proceed until the Amazon EC2 instance is launched and has passed all checks successfully.

To verify that the AWS CodeDeploy Agent is running on the Amazon EC2 instance and is ready to participate in deployments, see AWS CodeDeploy Agent Operations (p. 205), and then return to this page. After you do this, the Amazon EC2 instance will be ready to participate in AWS CodeDeploy deployments. The next step is to proceed to Create an Application (p. 111).

# Create an IAM Instance Profile for Your Amazon EC2 Instances

Your Amazon EC2 instances need permission to access the Amazon S3 buckets or GitHub repositories where you're storing your applications for AWS CodeDeploy to deploy. These instructions show you how to create a special type of IAM role—known as an *IAM instance profile*—to attach to your Amazon EC2 instances that gives them this permission.

You can create an IAM instance profile with the AWS CLI, the IAM console, or the IAM APIs.

> **Note**
> You must attach an IAM instance profile to an Amazon EC2 instance as you launch it. You cannot attach an IAM instance profile to an Amazon EC2 instance that has already been launched. For more information, see Instance Profiles.

**Topics**



## Use the AWS CLI to Create an IAM Instance Profile for Your Amazon EC2 Instances

The following instructions assume that you have already followed the instructions in Setting Up (p. 4), especially for installing and configuring the AWS CLI.

1. On your development machine, create a text file named `CodeDeployDemo-EC2-Trust.json` (you can name the file anything you want) with the following contents, which allows Amazon EC2 to work on your behalf:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. In the same directory, create a text file named `CodeDeployDemo-EC2-Permissions.json` (you can name the file anything you want) with the following contents:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:Get*",
```

```
        "s3:List*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

**Note**

We recommend that you restrict this policy to only the specific Amazon S3 buckets that your Amazon EC2 instances need to access. If you do restrict this policy, make sure to also give access to our Amazon S3 buckets that contain the AWS CodeDeploy Agent. Otherwise, an error may occur whenever the AWS CodeDeploy Agent is installed or updated on the associated Amazon EC2 instances. For example:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::CodeDeployDemoBucket/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*"
      ]
    }
  ]
}
```

3. From the same directory, call the **create-role** command to create an IAM role named **`CodeDeployDemo-EC2`** (you can name the IAM role anything you want), based on the information in the first file:

```
aws iam create-role --role-name CodeDeployDemo-EC2 --assume-role-policy-
document file://CodeDeployDemo-EC2-Trust.json
```

4. From the same directory, call the **put-role-policy** command to give the role named **`CodeDeployDemo-EC2`** the permissions based on the information in the second file:

```
aws iam put-role-policy --role-name CodeDeployDemo-EC2 --policy-name
CodeDeployDemo-EC2-Permissions --policy-document file://CodeDeployDemo-EC2-
Permissions.json
```

5. Call the **create-instance-profile** command followed by the **add-role-to-instance-profile** command to create an IAM instance profile named **`CodeDeployDemo-EC2-Instance-Profile`** (you can name the IAM instance profile anything you want), which allows Amazon EC2 to pass the IAM role named **`CodeDeployDemo-EC2`** on to an Amazon EC2 instance when the Amazon EC2 instance first launches:

```
aws iam create-instance-profile --instance-profile-name CodeDeployDemo-EC2-
Instance-Profile
aws iam add-role-to-instance-profile --instance-profile-name CodeDeployDemo-
EC2-Instance-Profile --role-name CodeDeployDemo-EC2
```

If you need the name of the IAM instance profile later, see Use the AWS CLI to Get the IAM Instance Profile Name (p. 87).

You've now created an IAM instance profile to attach to your Amazon EC2 instances that gives them permission to access the Amazon S3 buckets or GitHub repositories where you're storing your applications for AWS CodeDeploy to deploy.

For more information about setting up IAM instance profiles for Amazon EC2 instances, see IAM Roles for Amazon EC2 in the *Amazon EC2 User Guide*.

## Use the IAM Console to Create an IAM Instance Profile for Your Amazon EC2 Instances

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

   **Important**
   Make sure that you are signed in to the AWS Management Console using the same account information that you used when setting up the AWS CLI in Setting Up (p. 4).
2. In the IAM console, in the navigation pane, click **Policies**, and then click **Create Policy**. (If a **Get Started** button appears, click it, and then click **Create Policy**.)
3. Next to **Create Your Own Policy**, click **Select**.
4. In the **Policy Name** box, type **`CodeDeployDemo-EC2-Permissions`**.
5. In the **Policy Document** box, paste the following:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

**Note**
We recommend that you restrict this policy to only the specific Amazon S3 buckets that your Amazon EC2 instances need to access. If you do restrict this policy, make sure to also give access to our Amazon S3 buckets that contain the AWS CodeDeploy Agent. Otherwise, an error may occur whenever the AWS CodeDeploy Agent is installed or updated on the associated Amazon EC2 instances. For example:

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::CodeDeployDemoBucket/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*"
      ]
    }
  ]
}
```

6. Click **Create Policy**.
7. In the navigation pane, click **Roles**, and then click **Create New Role**.
8. In the **Role Name** box, give the IAM instance profile a name, such as **`CodeDeployDemo-EC2`** (you can name the IAM instance profile anything you want). Then click **Next Step**.
9. On the **Select Role Type** page, with **AWS Service Roles** already selected, click **Select** next to **Amazon EC2**.
10. On the **Attach Policy** page, check the box next to **`CodeDeployDemo-EC2-Permissions`**, and then click **Next Step**.
11. Click **Create Role**.

You've now created an IAM instance profile to attach to your Amazon EC2 instances that gives them permission to access the Amazon S3 buckets or GitHub repositories where you're storing your applications for AWS CodeDeploy to deploy.

For more information about setting up IAM instance profiles for Amazon EC2 instances, see IAM Roles for Amazon EC2 in the *Amazon EC2 User Guide*.

## Use the AWS CLI to Get the IAM Instance Profile Name

To get the name of the IAM instance profile that you created through either Use the IAM Console to Create an IAM Instance Profile for Your Amazon EC2 Instances (p. 86) or Use the AWS CLI to Create an IAM Instance Profile for Your Amazon EC2 Instances (p. 84), call the **list-instance-profiles-for-role** command against the IAM role named **`CodeDeployDemo-EC2`**:

```
aws iam list-instance-profiles-for-role --role-name CodeDeployDemo-EC2 --query
 "InstanceProfiles[0].InstanceProfileName" --output text
```

The value that is returned is the IAM instance profile name.

> **Note**
> You cannot use the IAM console to get the IAM instance profile name.

# Use an Existing Amazon EC2 Instance to Work with AWS CodeDeploy

These instructions show you how to configure an existing Amazon Linux, Ubuntu Server, or Windows Server Amazon EC2 instance to participate in AWS CodeDeploy deployments.

> **Note**
> If you do not have an existing Amazon EC2 instance, you can use our AWS CloudFormation template to simplify launching a new Amazon Linux or Windows Server Amazon EC2 instance that is already configured to start participating in AWS CodeDeploy deployments. (We currently do not provide an AWS CloudFormation template to simplify launching a new Ubuntu Server Amazon EC2 instance that is configured for AWS CodeDeploy.) To learn about this and other alternatives, see Configure Instances (p. 72).

Your Amazon EC2 instance must have an IAM instance profile attached, and that IAM instance profile must have the correct permissions to participate in AWS CodeDeploy deployments. If the Amazon EC2 instance does not have an IAM instance profile attached, or if the attached IAM instance profile does not have the correct permissions, then the Amazon EC2 instance cannot participate in AWS CodeDeploy deployments, and you cannot follow the steps on this page. Instead, you must stop and create a new Amazon Linux, Ubuntu Server, or Windows Server Amazon EC2 instance through one of the alternatives at Configure Instances (p. 72).

Your Amazon EC2 instance must also be tagged, and the AWS CodeDeploy Agent must also be installed and running on the Amazon EC2 instance. If the AWS CodeDeploy Agent is not installed and running, any deployments that are intended for the Amazon EC2 instance will appear to be stalled in a pending state.

**Topics**



## Step 1: Verify Whether an IAM Instance Profile is Attached to Your Amazon EC2 Instance

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **Instances**, click **Instances**.
3. Browse to and select your existing Amazon Linux, Ubuntu Server, or Windows Server Amazon EC2 instance in the list.
4. In the details pane, on the **Description** tab, note the value of the **IAM role** field, and proceed to the next section.

   If this field is empty, you cannot use this Amazon EC2 instance. Instead, you must stop and create a new Amazon Linux, Ubuntu Server, or Windows Server Amazon EC2 instance through one of the alternatives at Configure Instances (p. 72).

# Step 2: Verify Whether the Attached IAM Instance Profile Has the Correct Access Permissions

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, click **Roles**.
3. Browse to and select the IAM role name that you noted from the previous section.

   **Tip**
   If you used our AWS CloudFormation template to generate an Amazon EC2 instance that's compatible with AWS CodeDeploy, and you want to use the service role that the AWS CloudFormation template automatically generated for you instead of one that you created by following the instructions in Create a Service Role (p. 140), then keep reading this tip.
   In some versions of our AWS CloudFormation template, the display name of the IAM instance profile that is generated and attached to the Amazon EC2 instances is not the same as the display name in the IAM console. For example, the IAM instance profile could have a display name of `CodeDeploySampleStack-expnyi6-InstanceRoleInstanceProfile-IK8J8A9123EX`, while the corresponding IAM instance profile in the IAM console could have a display name of `CodeDeploySampleStack-expnyi6-InstanceRole-C5P33V1L64EX`.
   To help you identify the correct IAM instance profile in the IAM console, notice in this case that the prefix of `CodeDeploySampleStack-expnyi6-InstanceRole` is the same for both. For information about why these display names may be different, see Instance Profiles.
4. Expand **Trust Relationships**, if it is not already expanded. In the **Trusted Entities** area, If there is *not* an entry that reads **The identity provider ec2.amazonaws.com**, you cannot use this Amazon EC2 instance. Instead, you must stop and create a new Amazon Linux, Ubuntu Server, or Windows Server Amazon EC2 instance through one of the alternatives at Configure Instances (p. 72).
5. If there is an entry that reads **The identity provider ec2.amazonaws.com**, and you are or will be storing your applications only in GitHub repositories to deploy, then skip ahead to Step 3 (p. 90).
6. If there is an entry that reads **The identity provider ec2.amazonaws.com**, and you are—or will be—storing your applications in Amazon S3 buckets to deploy, then expand **Permissions**, if it is not already expanded.
7. If there is a policy in the **Managed Policies** area, click the policy's name, and then click **Edit** next to **Policy Document**; if there is a policy in the **Inline Policies** area, click **Edit Policy** under **Actions**.
8. If you are—or will be—storing your applications in Amazon S3 buckets to deploy, in the **Policy Document** box, make sure that `"s3:Get*"` and `"s3:List*"` are included among the list of specified actions. For example, it may look something like this:

```
{"Statement":[{"Resource":"*","Action":[
  ... Some actions may already be listed here ...
  "s3:Get*","s3:List*"
  ... Some more actions may already be listed here ...
  ],"Effect":"Allow"}]}
```

   Or it may look something like this:

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": [
          ... Some actions may already be listed here ...
          "s3:Get*",
          "s3:List*"
```

```
        ... Some more actions may already be listed here ...
      ],
      ...
    }
  ]
}
```

If `"s3:Get*"` and `"s3:List*"` are not already included in the list of specified actions, add them, and then save or apply your changes. (If `"s3:Get*"` or `"s3:List*"` is not the last action in the list, be sure to add a comma after it, so that the policy document validates.)

**Note**
We recommend that you restrict this policy to only the specific Amazon S3 buckets that your Amazon EC2 instances need to access. If you do restrict this policy, make sure to also give access to our Amazon S3 buckets that contain the AWS CodeDeploy Agent. Otherwise, an error may occur whenever the AWS CodeDeploy Agent is installed or updated on the associated Amazon EC2 instances. For example:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::CodeDeployDemoBucket/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*"
      ]
    }
  ]
}
```

# Step 3: Tag the Amazon EC2 Instance

For instructions on how to tag the Amazon EC2 instance so that AWS CodeDeploy can find the Amazon EC2 instance during a deployment, go to Working with Tags in the Console, and then return to this page.

**Note**
You can tag the Amazon EC2 instance with whatever key and value you want. Just make sure to specify this key and value when you attempt to deploy to it.

# Step 4: Install the AWS CodeDeploy Agent on the Amazon EC2 Instance

For instructions on how to install the AWS CodeDeploy Agent on the Amazon EC2 instance and verify that the AWS CodeDeploy Agent is running, go to AWS CodeDeploy Agent Operations (p. 205). After you

do this, the Amazon EC2 instance will be ready to participate in AWS CodeDeploy deployments. The next step is to proceed to Create an Application (p. 111).

# Configure Existing On-Premises Instances by Using AWS CodeDeploy

These instructions show you how to configure an existing on-premises instance—that is, a physical device that is not an Amazon EC2 instance—and then register and tag it with AWS CodeDeploy so that it can participate in deployments. These instructions also show you how to use AWS CodeDeploy to get information about on-premises instances, as well as how to deregister an on-premises instance after you're no longer planning to deploy to it.

For information about on-premises instances and how they work with AWS CodeDeploy, see On-Premises Instances (p. 69).

**Topics**



## Prerequisites for Configuring an On-Premises Instance

The IAM user that you will be using to register the on-premises instance with AWS CodeDeploy must have the correct permissions to complete the registration (and to deregister the on-premises instance as needed). To do this, make sure that the calling IAM user has the policy attached as described in Setting Up (p. 4) *and* also has the following additional policy attached:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateAccessKey",
        "iam:CreateUser",
        "iam:DeleteAccessKey",
        "iam:DeleteUser",
        "iam:DeleteUserPolicy",
        "iam:ListAccessKeys",
        "iam:ListUserPolicies",
        "iam:PutUserPolicy",
        "iam:GetUser"
      ],
      "Resource": "*"
    }
  ]
}
```

Also, the device that you want to prepare, register, and tag as an on-premises instance with AWS CodeDeploy must meet the following minimum requirements:

1. The on-premises instance must be running a supported operating system. For a list of supported operating systems, see Operating Systems Supported by the AWS CodeDeploy Agent (p. 206).

   For unsupported operating systems, the AWS CodeDeploy Agent is available as open source for you to adapt to your needs. For more information, see the AWS CodeDeploy Agent repository in GitHub.
2. The on-premises instance must be able to connect to public AWS service endpoints so that it can communicate with the AWS CodeDeploy service.

   **Note**
   The AWS CodeDeploy Agent communicates outbound using HTTPS over port 443.
3. The local or network account that is being used on the on-premises instance to configure the on-premises instance must be able to run either as `sudo` or `root` (for Ubuntu Server) or as an administrator (for Windows Server).

If your device meets the preceding requirements, continue with the following steps.

# Configure and Register an On-Premises Instance with the AWS CLI

To configure an on-premises instance and register and tag it with AWS CodeDeploy with the least amount of effort, follow these instructions. Alternatively, if you want to configure an on-premises instance and register and tag it with AWS CodeDeploy mostly on your own, follow the instructions in Manually Configure and Register an On-Premises Instance (p. 96).

**Topics**



## Step 1: Install and Configure the AWS CLI on the On-Premises Instance

1. Install the AWS CLI on the on-premises instance, if you have not done so already. To do this, follow the instructions in Getting Set Up with the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

   **Note**
   AWS CodeDeploy commands for working with on-premises instances became available starting with version 1.7.19 of the AWS CLI. If you have a version of the AWS CLI already installed, you can check its version by calling **aws --version**.
2. Configure the AWS CLI on the on-premises instance by following the instructions in Configuring the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

   **Important**
   As you configure the AWS CLI (for example, by calling the **aws configure** command), be sure to specify the secret key ID and secret access key of an IAM user that has at minimum the following AWS access permissions *in addition* to the access permissions that were

specified in the prerequisites (p. 91). This establishes the correct permissions for downloading and installing the AWS CodeDeploy Agent on the on-premises instance. The complete set of access permissions should look similar to this:

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:*",
        "iam:CreateAccessKey",
        "iam:CreateUser",
        "iam:DeleteAccessKey",
        "iam:DeleteUser",
        "iam:DeleteUserPolicy",
        "iam:ListAccessKeys",
        "iam:ListUserPolicies",
        "iam:PutUserPolicy",
        "iam:GetUser"
      ],
      "Resource" : "*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource" : [
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*"

      ]
    }
  ]
}
```

## Step 2: Call the Register Command

**Note**
The instructions in this step assume that you are registering the on-premises instance from the on-premises instance itself. You can also register an on-premises instance from a separate device or instance that has the AWS CLI installed and configured as described in the previous step.

Use the AWS CLI to call the **register** command, specifying:

- A name that uniquely identifies the on-premises instance to AWS CodeDeploy (with the `--instance-name` option).

  **Important**
  To help identify the on-premises instance later, especially for debugging purposes, we strongly recommend that you specify a name that maps to some unique characteristic of the on-premises instance that can be easily referenced, for example the serial number, or some unique internal

asset identifier if applicable. If you specify a name such as a MAC address, note that MAC addresses contain characters that AWS CodeDeploy does not allow, such as colon (`:`) characters. For a list of allowed characters, see Limits (p. 240).

- Optionally, the ARN of an existing IAM user that you want to associate with this on-premises instance (with the `--iam-user-arn` option). To get the ARN of an existing IAM user, call the **get-user** command, or click the IAM user name in the **Users** section of the IAMconsole and then find the **User ARN** value in the **Summary** section. If this option is not specified, AWS CodeDeploy will create a new IAM user on your behalf within your AWS account and associate it with the on-premises instance.

  **Important**
  If you specify the `--iam-user-arn` option, you must also manually create the necessary on-premises instance configuration file, as described in Step 4 (p. 101) of the manual configuration instructions.
  You can associate only one IAM user with only one on-premises instance. Trying to associate a single IAM user with multiple on-premises instances can result in errors, failed deployments to those on-premises instances, or deployments to those on-premises instances that are stuck in a perpetual pending state.

- Optionally, a set of on-premises instance tags that AWS CodeDeploy will use to identify the correct set of instances to deploy to (with the `--tags` option). Specify each tag with `Key=`*`tag-key`*`,Value=`*`tag-value`*, for example `Key=Name,Value=Beta` `Key=Name,Value=WestRegion`. If this option is not specified, no tags will be registered at this time. To register tags later, call the **add-tags-to-on-premises-instances** command.
- Optionally, the AWS region where the on-premises instance will be registered with AWS CodeDeploy (with the `--region` option). This must be one of the supported regions (p. 192). For example, `us-west-2`. If this option is not specified, the default AWS region that is associated with the calling IAM user will be used.

For example:

```
aws deploy register --instance-name AssetTag12010298EX --iam-user-arn
arn:aws:iam::80398EXAMPLE:user/CodeDeployUser-OnPrem --tags
Key=Name,Value=CodeDeployDemo-OnPrem --region us-west-2
```

The **register** command does the following:

1. If no existing IAM user is specified, creates a new IAM user, attaches the necessary permissions to it, and generates a corresponding secret key and secret access key. The on-premises instance will use this IAM user and its permissions and credentials to authenticate and interact with AWS CodeDeploy.
2. Registers the on-premises instance with AWS CodeDeploy.
3. If specified, associates in the AWS CodeDeploy system the tags that are specified with the `--tags` option with the registered on-premises instance name.
4. If a new IAM user was created, also creates the necessary configuration file in the same directory that the **register** command was called from.

If this command encounters any errors, an error message appears, describing how you can manually complete the remaining steps. Otherwise, a success message appears, describing how to call the **install** command as listed in the next step.

## Step 3: Call the Install Command

From the on-premises instance, use the AWS CLI to call the **install** command, specifying:

- The path to the configuration file (with the `--config-file` option).

- Optionally, whether to replace the existing configuration file that already exists on the on-premises instance (with the `--override-config` option). If not specified, the existing configuration file will not be replaced.
- Optionally, the AWS region where the on-premises instance will be registered with AWS CodeDeploy (with the `--region` option). This must be one of the supported regions (p. 192). For example, `us-west-2`. If this option is not specified, the default AWS region that is associated with the calling IAM user will be used.
- Optionally, a custom location to install the AWS CodeDeploy Agent from (with the `--agent-installer` option). This option is useful for installing a custom version of the AWS CodeDeploy Agent that AWS CodeDeploy does not officially support (such as a custom version based on the AWS CodeDeploy Agent repository in GitHub). The value must be the path to an Amazon S3 bucket that contains either an AWS CodeDeploy Agent installation script (for Linux- or Unix-based operating systems, similar to the install file in the AWS CodeDeploy Agent repository in GitHub), or to an AWS CodeDeploy Agent installer package (.msi) file (for Windows-based operating systems). If this option is not specified, AWS CodeDeploy will make its best attempt to install from its own location an officially-supported version of the AWS CodeDeploy Agent that is compatible with the current operating system of the on-premises instance.

For example:

```
aws deploy install --override-config --config-file /tmp/codedeploy.onpremises.yml
 --region us-west-2 --agent-installer s3://aws-codedeploy-us-west-
2/latest/codedeploy-agent.msi
```

The **install** command does the following:

1. Checks whether the on-premises instance is an Amazon EC2 instance. If it is, an error message appears.
2. If the on-premises instance configuration file does not already exist with the expected file name and location on the on-premises instance (for Ubuntu Server, this is `/etc/codedeploy-agent/conf/codedeploy.onpremises.yml`. For Windows Server, this is `C:\ProgramData\Amazon\CodeDeploy\conf.onpremises.yml`), or if the `--override-config` option was specified, creates or overwrites the file.
3. Installs the AWS CodeDeploy Agent on the on-premises instance and then starts it.

## Step 4: Deploy Existing Application Revisions to the On-Premises Instance

You are now ready to deploy application revisions to the registered and tagged on-premises instance.

You deploy application revisions to on-premises instances similar to how you deploy application revisions to properly configured Amazon EC2 instances. For instructions, see Deploy a Revision (p. 121). Note that those instructions link to prerequisites including creating an application, creating a deployment group, and preparing an application revision. If you need a simple, sample application revision to deploy, you can create the one that is described in Step 2 (p. 51) of the On-Premises Instance Deployment (Windows Server or Ubuntu Server) (p. 50).

> **Important**
> If you choose to reuse an existing AWS CodeDeploy service role as part of creating a deployment group that targets on-premises instances, you must include `Tag:get*` to the `Action` portion of the service role's policy statement. For more information, see Create a Service Role (p. 140).

## Step 5: Track Deployments to the On-Premises Instance

After you deploy an application revision to registered and tagged on-premises instances, you can track the deployment's progress.

You track deployments to on-premises instances similar to how you track deployments to properly configured Amazon EC2 instances. For instructions, see View Deployment Details (p. 127).

For additional options, skip down to Next Steps / Advanced Tasks (p. 105).

# Manually Configure and Register an On-Premises Instance

To configure an on-premises instance and register and tag it with AWS CodeDeploy mostly on your own, instead of following the instructions in the previous section, follow these instructions.

**Topics**



## Step 1: Create a New IAM User on Behalf of the On-Premises Instance

Create a new IAM user that the on-premises instance will use to authenticate and interact with AWS CodeDeploy. You can create a new IAM user with the AWS Command Line Interface (AWS CLI) or with the IAM console.

> **Important**
> You must create a separate IAM user for each and every participating on-premises instance. If you try to reuse an individual IAM user for multiple on-premises instances, you may not be able to successfully resister or tag those on-premises instances with AWS CodeDeploy, and deployments to those on-premises instances may be stuck in a perpetual pending state or may fail altogether.

### To use the AWS CLI to create the new IAM user

1. Call the **create-user** command, specifying a name for the IAM user, for example **`CodeDeployUser-OnPrem`** (with the `--user-name` option). For example:

```
aws iam create-user --user-name CodeDeployUser-OnPrem
```

2. In the output of the call to the **create-user** command, note the value of the `Arn` field, as you will need the user ARN later in Step 8 (p. 103).
3. Call the **create-access-key** command, specifying the name of the newly created user (with the `--user-name` option). For example:

```
aws iam create-access-key --user-name CodeDeployUser-OnPrem
```

4. In the output of the call to the **create-access-key** command, note the value of the `AccessKeyId` and `SecretAccessKey` fields, as you will need them later in Step 4 (p. 101).

   **Important**
   This will be the only time that you will have access to this particular secret access key. If you forget or lose access to this particular secret access key, you will need to generate a new one, which you can learn how to do later in Step 3 (p. 99).

### To use the IAM console to create the new IAM user

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, click **Users**.
3. Click **Create New Users**.
4. In the first **Enter User Names** box, type a name for the IAM user, for example **`CodeDeployUser-OnPrem`**.
5. With the **Generate an access key for each user** box already checked, click **Create**. IAM creates the new IAM user and then displays a confirmation message.
6. Click **Show User Security Credentials** and make a note of the access key ID and the secret access key, as you will need this information later in Step 4 (p. 101). Alternatively, you can click **Download Credentials** to save a copy of the access key ID and the secret access key to a convenient location.

   **Important**
   Unless you make a note of or download the credentials, this will be the only time that you will have access to this particular secret access key. If you forget or lose access to this particular secret access key, you will need to generate a new one, which you can learn how to do later in Step 3 (p. 99).
7. After you have noted or downloaded the credentials, click **Close** to return to the list of users.
8. In the list of users, click the name of the newly-created IAM user.
9. In the **Summary** area, note the value of the **User ARN** field, as you will need this information later in Step 4 (p. 101) and Step 8 (p. 103).

## Step 2: Assign the Correct Permissions to the IAM User

If your on-premises instance will be deploying application revisions from Amazon S3 buckets, you must assign to the IAM user the correct permissions to interact with those buckets. You can assign permissions with the AWS CLI or with the IAM console.

**Note**
If you will be deploying application revisions only from GitHub repositories, skip this step and go directly to Step 3 (p. 99). (You will still need information about the IAM user that you created earlier in Step 1 (p. 96), as it will be used in later steps.)

### To use the AWS CLI to assign the permissions

1. Create a file on the Amazon EC2 instance or device that you are using to call the AWS CLI. This file will express the needed permissions. Name the file something like

**`CodeDeploy-OnPrem-Permissions.json`** with the following policy contents, and then save the file:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

**Note**

We recommend that you restrict this policy to only the specific Amazon S3 buckets that your on-premises instance need to access. If you do restrict this policy, make sure to also give access to our Amazon S3 buckets that contain the AWS CodeDeploy Agent. Otherwise, an error may occur whenever the AWS CodeDeploy Agent is installed or updated on the associated on-premises instance. For example:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::CodeDeployDemoBucket/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*"
      ]
    }
  ]
}
```

2. Call the **put-user-policy** command, specifying the name of the IAM user (with the `--user-name` option), a name for the policy (with the `--policy-name` option), and the path to the newly-created policy document (with the `--policy-document` option). For example, assuming that the **`CodeDeploy-OnPrem-Permissions.json`** file is in the same directory (folder) that you're calling this command from:

```
aws iam put-user-policy --user-name CodeDeployUser-OnPrem --policy-name
CodeDeploy-OnPrem-Permissions --policy-document file://CodeDeploy-OnPrem-
Permissions.json
```

### To use the IAM console to assign the permissions

1. If you don't already have the IAM console open:

   Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the IAM console, in the navigation pane, click **Policies**, and then click **Create Policy**. (If a **Get Started** button appears, click it, and then click **Create Policy**.)
3. Next to **Create Your Own Policy**, click **Select**.
4. In the **Policy Name** box, type a name for this policy, for example **`CodeDeploy-OnPrem-Permissions`**.
5. In the **Policy Document** box, type or paste the following permissions expression, which allows AWS CodeDeploy to deploy application revisions from any permitted Amazon S3 bucket to the on-premises instance on behalf of the IAM user account:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

6. Click **Create Policy**.
7. In the navigation pane, click **Users**.
8. In the list of users, browse to and click the name of the IAM user that you created earlier in Step 1 (p. 96). The **User > *`IAM User Name`*** page appears.
9. Expand the **Permissions** area, if it is not already expanded.
10. In the **Managed Policies** area, click **Attach Policy**.
11. Select the policy named **`CodeDeploy-OnPrem-Permissions`**, and then click **Attach Policy**. IAM applies the policy and then redisplays the **User > *`IAM User Name`*** page.

## Step 3: Get the IAM User Credentials

Get the secret key ID and the secret access key for the IAM user, which you will need later for Step 4 (p. 101). You can get the secret key ID and the secret access key with the AWS CLI or with the IAM console.

> **Note**
> If you already have the secret key ID and the secret access key available, skip this step and go directly to Step 4 (p. 101).

### To use the AWS CLI to get the credentials

1. Call the **list-access-keys** command, specifying the name of the IAM user (with the `--user-name` option), and querying for just the access key IDs (with the `--query` and `--output` options). For example:

```
aws iam list-access-keys --user-name CodeDeployUser-OnPrem --query "Access
KeyMetadata[*].AccessKeyId" --output text
```

2. If no keys are output or information about only one key is output, call the **create-access-key** command, specifying the name of the IAM user (with the `--user-name` option). For example:

```
aws iam create-access-key --user-name CodeDeployUser-OnPrem
```

In the output of the call to the **create-access-key** command, note the value of the `AccessKeyId` and `SecretAccessKey` fields, as you will need them later in Step 4 (p. 101).

**Important**
This will be the only time that you will have access to this particular secret access key. If you forget or lose access to this particular secret access key, you will need to generate a new one by following the steps in this step (Step 3 (p. 99)) from the beginning.

3. If, however, two access keys are already listed, you must delete at least one of them by calling the **delete-access-key** command, specifying the name of the IAM user (with the `--user-name` option) and the ID of the access key to delete (with the `--access-key-id` option). Then call the **create-access-key** command, as described earlier in this step. Here's an example of calling the **delete-access-key** command:

```
aws iam delete-access-key --user-name CodeDeployUser-OnPrem --access-key-id
 access-key-ID
```

**Important**
If you call the **delete-access-key** command to delete one of these access keys, and an on-premises instance is already using this access key as described in Step 4 (p. 101), you will need to follow the instructions in Step 4 (p. 101) again to specify a different access key ID and secret access key that is associated with this IAM user. Otherwise, any deployments to that on-premises instance may be stuck in a perpetual pending state or may fail altogether.

### To use the IAM console to get the credentials

1. If you don't already have the **User > *IAM User Name*** page displayed in the IAM console earlier from Step 2 (p. 97):
   1. Open the IAM console at https://console.aws.amazon.com/iam/.
   2. If you don't already have the list of users displayed, then in the navigation pane, click **Users**.
   3. In the list of users, browse to and click the name of the IAM user that you created earlier in Step 1 (p. 96). The **User > *IAM User Name*** page appears.
2. Expand the **Security Credentials** area, if it is not already expanded.
3. In the **Security Credentials** area, within the **Access Credentials** area, click **Manage Access Keys**. The **Manage Access Keys** dialog box appears.
4. If no keys or only one key is listed, click **Create Access Key**.
5. If, however, two access keys are already listed, you must delete at least one of them by clicking **Delete** next to one of the access keys. Click **Confirm** when prompted. Then click **Manage Access Keys** again, followed by clicking **Create Access Key**.

**Important**
If you click **Delete** next to one of these access keys, and an on-premises instance is already using this access key as described in Step 4 (p. 101), you will need to follow the instructions

in Step 4 (p. 101) again to specify a different access key ID and secret access key that is associated with this IAM user. Otherwise, any deployments to that on-premises instance may be stuck in a perpetual pending state or may fail altogether.

6. Click **Show User Security Credentials** and make a note of the access key ID and the secret access key, as you will need this information for the next step. Alternatively, you can click **Download Credentials** to save a copy of the access key ID and the secret access key to a convenient location.

   **Important**
   Unless you make a note of or download the credentials, this will be the only time that you will have access to this particular secret access key. If you forget or lose access to this particular secret access key, you will need to generate a new one by following the steps in this step (Step 3 (p. 99)) from the beginning.

7. After you have noted or downloaded the credentials, click **Close** to return to the **User > *IAM User Name*** page.

## Step 4: Add a Configuration File to the On-Premises Instance

Add a configuration file to the on-premises instance, using root or Administrator permissions. This configuration file will be used to declare the IAM user credentials and the target AWS region to be used for AWS CodeDeploy. The file must be added to a specific location on the on-premises instance; the file must include the IAM user's ARN, secret key ID, secret access key, and the target AWS region; and the file must follow a specific format, as follows:

1. Create a file named `codedeploy.onpremises.yml` (for an Ubuntu Server on-premises instance) or `conf.onpremises.yml` (for a Windows Server on-premises instance) in the following location on the on-premises instance:

   - For Ubuntu Server: `/etc/codedeploy-agent/conf`
   - For Windows Server: `C:\ProgramData\Amazon\CodeDeploy`

2. Use a text editor of your choice to add the following information to the newly-created `codedeploy.onpremises.yml` or `conf.onpremises.yml` file:

```
---
aws_access_key_id: secret-key-id
aws_secret_access_key: secret-access-key
iam_user_arn: IAM-user-ARN
region: supported-region
```

   Where:

   - *`secret-key-id`* is the corresponding IAM user's secret key ID that you noted earlier in Step 1 (p. 96) or Step 3 (p. 99).
   - *`secret-access-key`* is the corresponding IAM user's secret access key that you noted earlier in Step 1 (p. 96) or Step 3 (p. 99).
   - *`IAM-user-ARN`* is the corresponding IAM user's ARN that you noted earlier in Step 1 (p. 96).
   - *`supported-region`* is the identifier of a region that AWS CodeDeploy supports and where your corresponding AWS CodeDeploy applications, deployment groups, and application revisions for your deployments are located (for example, `us-west-2`). For a list of regions that AWS CodeDeploy supports, see AWS CodeDeploy Deployment Resources Are Supported in Only Certain Regions (p. 192).

**Important**
If you clicked **Delete** next to one of the access keys in Step 3 (p. 99), and your on-premises instance is already using the associated access key ID and secret access key, you will need to follow the instructions in this step (Step 4 (p. 101)) from the beginning to specify a different access key ID and secret access key that is associated with this IAM user. Otherwise, any deployments to your on-premises instance may be stuck in a perpetual pending state or may fail altogether.

## Step 5: Install and Configure the AWS CLI

Install and configure the AWS CLI on the on-premises instance as follows. (The AWS CLI is needed in Step 7 (p. 103) to download and install the AWS CodeDeploy Agent on the on-premises instance.)

1. Install the AWS CLI on the on-premises instance, if you have not done so already. To do this, follow the instructions in Getting Set Up with the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

   **Note**
   AWS CodeDeploy commands for working with on-premises instances became available starting with version 1.7.19 of the AWS CLI. If you have a version of the AWS CLI already installed, you can check its version by calling **aws --version**.

2. Configure the AWS CLI on the on-premises instance by following the instructions in Configuring the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

   **Important**
   As you configure the AWS CLI (for example, by calling the **aws configure** command), be sure to specify the secret key ID and secret access key of an IAM user that has at minimum the following AWS access permissions *in addition* to the access permissions that were specified in the prerequisites (p. 91). This establishes the correct permissions for you to download and install the AWS CodeDeploy Agent on the on-premises instance:

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:*"
      ],
      "Resource" : "*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource" : [
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*"
      ]
    }
  ]
}
```

These access permissions can be assigned to either the new IAM user that you created earlier in Step 1 (p. 96), or they can be assigned to a different IAM user. To assign these permissions to an IAM user, follow the instructions in Step 2 (p. 96), assigning these access permissions instead of the ones that are listed in that step.

## Step 6: Set the AWS_REGION Environment Variable (Ubuntu Server Only)

If you are not running Ubuntu Server on your on-premises instance, skip this step and go directly to Step 7 (p. 103).

Otherwise, prepare to install the AWS CodeDeploy Agent on an Ubuntu Server on-premises instance, as well as enable the Ubuntu Server on-premises instance to automatically update the AWS CodeDeploy Agent whenever a new version becomes available. You do this by setting the `AWS_REGION` environment variable on the Ubuntu Server on-premises instance to the identifier of one of the regions that AWS CodeDeploy supports. We recommend that you set the value to the region where your corresponding AWS CodeDeploy applications, deployment groups, and application revisions for your deployments are located (for example, `us-west-2`). For a list of regions that AWS CodeDeploy supports, see AWS CodeDeploy Deployment Resources Are Supported in Only Certain Regions (p. 192).

To set the environment variable, call the following from the terminal:

```
export AWS_REGION=supported-region
```

Where *`supported-region`* is the region identifier (for example, `us-west-2`).

## Step 7: Install the AWS CodeDeploy Agent

Install the AWS CodeDeploy Agent on the on-premises instance by completing one of the following sets of instructions:

- To install the AWS CodeDeploy Agent on an Ubuntu Server on-premises instance, follow the instructions in Install, Uninstall, or Reinstall the AWS CodeDeploy Agent for Ubuntu Server (p. 209), and then return to this page.
- To install the AWS CodeDeploy Agent on a Windows Server on-premises instance, follow the instructions in Install, Uninstall, or Reinstall the AWS CodeDeploy Agent for Windows Server (p. 210), and then return to this page.

## Step 8: Register the On-Premises Instance with AWS CodeDeploy

**Note**
The instructions in this step assume that you are registering the on-premises instance from the on-premises instance itself. You can also register an on-premises instance from a separate device or instance that has the AWS CLI installed and configured as described in Step 5 (p. 102).

Use the AWS CLI to register the on-premises instance with AWS CodeDeploy so that it can participate in deployments as follows:

1. Before you can use the AWS CLI, you will need the user ARN of the IAM user that you created in earlier in Step 1 (p. 96). If you don't already have the user ARN, call the **get-user** command, specifying the name of the IAM user (with the `--user-name` option) and querying for just the user ARN (with the `--query` and `--output` options). For example:

```
aws iam get-user --user-name CodeDeployUser-OnPrem --query "User.Arn" --
output text
```

2. Call the **register-on-premises-instance** command, specifying:

   - A name that uniquely identifies the on-premises instance to AWS CodeDeploy (with the `--instance-name` option).

     **Important**
     To help identify the on-premises instance later, especially for debugging purposes, we strongly recommend that you specify a name that maps to some unique characteristic of the on-premises instance that can be easily referenced, for example the serial number, or some unique internal asset identifier if applicable. If you specify a name such as a MAC address, note that MAC addresses contain characters that AWS CodeDeploy does not allow, such as colon (`:`) characters. For a list of allowed characters, see Limits (p. 240).

   - The user ARN of the IAM user that you created in Step 1 (p. 96) (with the `--iam-user-arn` option).

     For example:

```
aws deploy register-on-premises-instance --instance-name AssetTag12010298EX
 --iam-user-arn arn:aws:iam::80398EXAMPLE:user/CodeDeployUser-OnPrem
```

## Step 9: Tag the On-Premises Instance

You can use either the AWS CLI or the AWS CodeDeploy console to tag the on-premises instance. (AWS CodeDeploy uses on-premises instance tags to identify the correct sets of deployment targets during a deployment.)

### To use the AWS CLI to tag the on-premises instance

- Call the **add-tags-to-on-premises-instances** command, specifying:

  - The name that uniquely identifies the on-premises instance (with the `--instance-names` option).
  - The name of the on-premises instance tag key and tag value that you want to use (with the `--tags` option). You must specify both a name and value; AWS CodeDeploy does not allow on-premises instance tags that have only values.

    For example:

```
aws deploy add-tags-to-on-premises-instances --instance-names As
setTag12010298EX --tags Key=Name,Value=CodeDeployDemo-OnPrem
```

### To use the AWS CodeDeploy console to tag the on-premises instance

1. If you don't already have the AWS CodeDeploy console displayed:

   Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

**Note**
Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. If you don't already have the **On-Premises Instances** page displayed, then in the service navigation bar, select **On-Premises Instances**.
3. In the list of on-premises instances, click the arrow next to the name of the on-premises instance that you want to tag.
4. In the list of tags, select or type the desired tag key and tag value. After you type the tag key and tag value, another row appears. You can repeat this for up to 10 tags. (To remove a tag, click the delete icon (the character `X` inside of a circle) in the row for the tag that you want to remove.)
5. After you have added all of your desired tags, click **Update Tags**.

## Step 10: Deploy Existing Application Revisions to the On-Premises Instance

You are now ready to deploy application revisions to the registered and tagged on-premises instance.

You deploy application revisions to on-premises instances similar to how you deploy application revisions to properly configured Amazon EC2 instances. For instructions, see Deploy a Revision (p. 121). Note that these instructions link to prerequisites including creating an application, creating a deployment group, and preparing an application revision. If you need a simple, sample application revision to deploy, you can create the one that is described in Step 2 (p. 51) of the On-Premises Instance Deployment (Windows Server or Ubuntu Server) (p. 50).

**Important**
If you choose to reuse an existing AWS CodeDeploy service role as part of creating a deployment group that targets on-premises instances, you must include `Tag:get*` to the `Action` portion of the service role's policy statement. For more information, see Create a Service Role (p. 140).

## Step 11: Track Deployments to the On-Premises Instance

After you deploy an application revision to registered and tagged on-premises instances, you can track the deployment's progress.

You track deployments to on-premises instances similar to how you track deployments to properly configured Amazon EC2 instances. For instructions, see View Deployment Details (p. 127).

# Next Steps / Advanced Tasks

Follow the instructions in this section for additional tasks with on-premises instances, such as getting more information about specific on-premises instances, removing tags from on-premises instances, and uninstalling and deregistering on-premises instances after you're no longer planning to deploy to them.

**Topics**

# Get Information about a Single On-Premises Instance

Typically you get information about a single on-premises instance by following the instructions in View Deployment Details (p. 127). However, you can get more specific information about a single on-premises instance through the AWS CLI or the AWS CodeDeploy console, as follows:

### To use the AWS CLI to get information about a single on-premises instance

- Call the **get-on-premises-instance** command, specifying the name that uniquely identifies the on-premises instance (with the `--instance-name` option). For example:

```
aws deploy get-on-premises-instance --instance-name AssetTag12010298EX
```

### To use the AWS CodeDeploy console to get information about a single on-premises instance

1. If you don't already have the AWS CodeDeploy console displayed:

   Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).
2. If you don't already have the **On-Premises Instances** page displayed, then in the service navigation bar, select **On-Premises Instances**.
3. In the list of on-premises instances, click the arrow next to the name of the on-premises instance that you want to get information about. Details about the on-premises instance is displayed.

# Get Information about Multiple On-Premises Instances

Typically you can get information about on-premises instances by following the instructions in View Deployment Details (p. 127). However, you can get more specific information about on-premises instances through the AWS CLI or the AWS CodeDeploy console, as follows:

### To use the AWS CLI to get information about multiple on-premises instances

1. For a list of on-premises instance names, call the **list-on-premises-instances** command, specifying:

   - Whether to get information about all registered or deregistered on-premises instances (with the `--registration-status` option along with `Registered` or `Deregistered`, respectively). If you omit whether to get information about either registered or deregistered on-premises instances, then both registered and deregistered on-premises instance names are returned.
   - Whether to get information only about on-premises instances that are tagged with specific on-premises instance tags (with the `--tag-filters` option). For each on-premises instance tag, specify the `Key`, `Value`, and `Type` (which should always be `KEY_AND_VALUE`). Separate multiple on-premises instance tags with spaces between each `Key`, `Value`, and `Type` triplet.

   For example:

```
aws deploy list-on-premises-instances --registration-status Registered --
tag-filters Key=Name,Value=CodeDeployDemo-OnPrem,Type=KEY_AND_VALUE
Key=Name,Value=CodeDeployDemo-OnPrem-Beta,Type=KEY_AND_VALUE
```

2. For more detailed information, call the **batch-get-on-premises-instances** command, with the specific names of the on-premises instances to get information about (with the `--instance-names` option). For example:

```
aws deploy batch-get-on-premises-instances --instance-names AssetTag12010298EX
 AssetTag09920444EX
```

### To use the AWS CodeDeploy console to get information about multiple on-premises instances

1. If you don't already have the AWS CodeDeploy console displayed:

   Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. If you don't already have the **On-Premises Instances** page displayed, then in the service navigation bar, select **On-Premises Instances**. Information about the on-premises instances is displayed.

## Automatically Deregister an On-Premises Instance

Typically, you deregister an on-premises instance after you're no longer planning to deploy to it. For example, when you deregister an on-premises instance, then even though the on-premises instance may be part of a deployment group's on-premises instance tags, the on-premises instance will not be included in any associated deployments.You can automatically deregister on-premises instances through the AWS CLI, as follows.

**Note**
You cannot use the AWS CodeDeploy console to automatically deregister an on-premises instance. Also, automatically deregistering an on-premises instance does not disassociate any on-premises instance tags that are associated with the on-premises instance, nor uninstalls the AWS CodeDeploy Agent from the on-premises instance, nor removes the on-premises instance configuration file from the on-premises instance.
To use the AWS CodeDeploy console to perform some (but not all) of the activities in this section, see the AWS CodeDeploy console section of Manually Deregister an On-Premises Instance (p. 110).
To manually disassociate any associated on-premises instance tags, see Manually Remove On-Premises Instance Tags from an On-Premises Instance (p. 109).
To automatically uninstall the AWS CodeDeploy Agent and remove the configuration file from the on-premises instance, see Automatically Uninstall the AWS CodeDeploy Agent and Remove the Configuration File from an On-Premises Instance (p. 108).
To manually uninstall just the AWS CodeDeploy Agent from the on-premises instance, see AWS CodeDeploy Agent Operations (p. 205).

Use the AWS CLI to call the **deregister** command, specifying:

- The name that uniquely identifies the on-premises instance to AWS CodeDeploy (with the `--instance-name` option).

- Optionally, whether to delete the IAM user that is associated with the on-premises instance (with the `--delete-iam-user` option, the default). To not delete the IAM user that is associated with the on-premises instance, explicitly specify the `--no-delete-iam-user` option.
- Optionally, the AWS region where the on-premises instance was registered with AWS CodeDeploy (with the `--region` option). This must be one of the supported regions (p. 192). For example, `us-west-2`. If this option is not specified, the default AWS region that is associated with the calling IAM user will be used.

For example:

```
aws deploy deregister --instance-name AssetTag12010298EX --delete-iam-user --
region us-west-2
```

The **deregister** command does the following:

1. Deregisters the on-premises instance with AWS CodeDeploy.
2. If specified, deletes the IAM user that is associated with the on-premises instance.

After you deregister an on-premises instance, you cannot create a replacement on-premises instance with the same name or the same associated IAM user name until AWS CodeDeploy deletes its records about the deregistered on-premises instance. This typically takes about 24 hours.

If this command encounters any errors, an error message appears, describing how you can manually complete the remaining steps. Otherwise, a success message appears, describing how to call the **uninstall** command as listed in the next step.

## Automatically Uninstall the AWS CodeDeploy Agent and Remove the Configuration File from an On-Premises Instance

Typically, you uninstall the AWS CodeDeploy Agent and remove the configuration file from an on-premises instance after you're no longer planning to deploy to it, as follows.

> **Note**
> Automatically uninstalling the AWS CodeDeploy Agent and removing the configuration file from an on-premises instance does not deregister an on-premises instance nor disassociate any on-premises instance tags that are associated with the on-premises instance, nor deletes the IAM user that is associated with the on-premises instance.
> To automatically deregister the on-premises instance, see Automatically Deregister an On-Premises Instance (p. 107).
> To manually deregister the on-premises instance, see Manually Deregister an On-Premises Instance (p. 110).
> To manually disassociate any associated on-premises instance tags, see Manually Remove On-Premises Instance Tags from an On-Premises Instance (p. 109).
> To manually uninstall the AWS CodeDeploy Agent from the on-premises instance, see AWS CodeDeploy Agent Operations (p. 205).
> To manually delete the associated IAM user, see Deleting an IAM User from Your AWS Account.

From the on-premises instance, use the AWS CLI to call the **uninstall** command.

For example:

```
aws deploy uninstall
```

The **uninstall** command does the following:

1. Stops the running AWS CodeDeploy Agent on the on-premises instance.
2. Uninstalls the AWS CodeDeploy Agent from the on-premises instance.
3. Removes the configuration file from the on-premises instance. (For Ubuntu Server, this is `/etc/codedeploy-agent/conf/codedeploy.onpremises.yml`. For Windows Server, this is `C:\ProgramData\Amazon\CodeDeploy\conf.onpremises.yml`.)

# Manually Remove On-Premises Instance Tags from an On-Premises Instance

Typically, you remove an on-premises instance tag from an on-premises instance when that on-premises instance tag is no longer being used, or you want to remove the on-premises instance from any deployment groups that rely on that on-premises instance tag. You can remove on-premises instance tags from on-premises instances through the AWS CLI or the AWS CodeDeploy console, as follows.

> **Note**
> You do not need to remove the on-premises instance tags from an on-premises instance before you deregister it.
> Manually removing on-premises instance tags from an on-premises instance does not deregister the on-premises instance, nor uninstall the AWS CodeDeploy Agent from the on-premises instance, nor remove the configuration file from the on-premises instance, nor deletes the IAM user that is associated with the on-premises instance.
> To automatically deregister the on-premises instance, see Automatically Deregister an On-Premises Instance (p. 107).
> To manually deregister the on-premises instance, see Manually Deregister an On-Premises Instance (p. 110).
> To automatically uninstall the AWS CodeDeploy Agent and remove the configuration file from the on-premises instance, see Automatically Uninstall the AWS CodeDeploy Agent and Remove the Configuration File from an On-Premises Instance (p. 108).
> To manually uninstall just the AWS CodeDeploy Agent from the on-premises instance, see AWS CodeDeploy Agent Operations (p. 205).
> To manually delete the associated IAM user, see Deleting an IAM User from Your AWS Account.

### To use the AWS CLI to remove on-premises instance tags from an on-premises instance

- Call the **remove-tags-from-on-premises-instances**, specifying:
  - The names that uniquely identifies the on-premises instance (with the `--instance-names` option).
  - The names and values of the tags that you want to remove (with the `--tags` option).

    For example:

```
aws deploy remove-tags-from-on-premises-instances --instance-names As
setTag12010298EX --tags Key=Name,Value=CodeDeployDemo-OnPrem
```

### To use the AWS CodeDeploy console to remove on-premises instance tags from an on-premises instance

1. If you don't already have the AWS CodeDeploy console displayed:

   Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

**Note**
Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. If you don't already have the **On-Premises Instances** page displayed, then in the service navigation bar, select **On-Premises Instances**.
3. In the list of on-premises instances, click the arrow next to the name of the on-premises instance that you want to remove tags from.
4. In the **Tags** area, click the delete icon (the character X inside of a circle) in the row next to each tag that you want to remove.
5. After you have deleted all of the tags that you want to remove, click **Update Tags**.

## Manually Deregister an On-Premises Instance

Typically, you deregister an on-premises instance after you're no longer planning to deploy to it. You manually deregister on-premises instances through the AWS CLI, as follows.

**Note**
Manually deregistering an on-premises instance neither uninstalls the AWS CodeDeploy Agent nor removes the configuration file from the on-premises instance, nor deletes the IAM user that is associated with the on-premises instance, nor removes any on-premises instance tags that are associated with the on-premises instance.
To automatically uninstall the AWS CodeDeploy Agent and remove the configuration file the on-premises instance, see Automatically Uninstall the AWS CodeDeploy Agent and Remove the Configuration File from an On-Premises Instance (p. 108).
To manually uninstall just the AWS CodeDeploy Agent, see AWS CodeDeploy Agent Operations (p. 205).
To manually delete the associated IAM user, see Deleting an IAM User from Your AWS Account.
To manually remove just the associated on-premises instance tags, see Manually Remove On-Premises Instance Tags from an On-Premises Instance (p. 109).

- Call the **deregister-on-premises-instance** command, specifying the name that uniquely identifies the on-premises instance (with the `--instance-name` option). For example:

```
aws deploy deregister-on-premises-instance --instance-name AssetTag12010298EX
```

After you deregister an on-premises instance, you cannot create a replacement on-premises instance with the same name or the same associated IAM user name until AWS CodeDeploy deletes its records about the deregistered on-premises instance. This typically takes about 24 hours.

# Create an Application with AWS CodeDeploy

After you Configure Instances (p. 72), but before you can deploy a revision to any of those instances, you must create an application in AWS CodeDeploy. An *application* in AWS CodeDeploy is simply a unique identifier that AWS CodeDeploy uses to make sure it deploys the correct revision to the correct set of instances with the correct deployment configuration.

Use the following information to determine how to proceed next:

| | |
|---|---|
| I haven't created an application yet. | Follow the instructions on this page. |
| I have already created an application, but I haven't created a deployment group yet. | Skip these instructions. See Create a Deployment Group (p. 138) instead. |
| I have already created an application and a deployment group, but I haven't created an application revision yet. | Skip these instructions. See Prepare a Revision (p. 114) instead. |
| I have already created an application and a deployment group, and I have already uploaded my application revision. I'm ready to deploy. | Skip these instructions. See Deploy a Revision (p. 121) instead. |

You can use the AWS CodeDeploy console, the AWS CLI, or the AWS CodeDeploy APIs to create applications.

> **Note**
> To view a list of existing applications that are already registered to your AWS account, see View Application Details (p. 130).

**Topics**

- Create an Application by Using the AWS CodeDeploy Console (p. 112)
- Create an Application by Using the AWS CLI (p. 113)

# Create an Application by Using the AWS CodeDeploy Console

To create a new application by using the AWS CodeDeploy console, follow these steps:

**Caution**
You *cannot* start following these steps if:

- Your instances are not prepared to participate in AWS CodeDeploy deployments. Although this not a strict requirement before you create an application, we recommend that you set up your instances before you go through the process of deploying revisions to them. To set up your instances, follow the instructions in Configure Instances (p. 72), and then follow the steps below.
- You want to create a new application that uses a custom deployment configuration, but you have not yet created the custom deployment configuration. Instead, follow the instructions in Create a Deployment Configuration (p. 148), and then follow the steps below.
- You do not have an existing service role that trusts AWS CodeDeploy with at minimum the trust and permissions as described in Create a Service Role (p. 140). To create and configure a service role with the correct trust and permissions, follow the instructions in Create a Service Role (p. 140), and then follow the steps below.

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. If the **Applications** page is not already visible, in the AWS CodeDeploy service navigation bar, click **Applications**.
3. On the **Applications** page, click **Create New Application**.
4. In the **Application Name** box, type the application's name. (This name must be unique across all of the applications that you create with AWS CodeDeploy associated with your AWS account.)
5. In the **Deployment Group Name** box, type a name that describes the deployment group that you'll be deploying the application to.

   **Tip**
   If you want to use the same settings as an existing deployment group (including the deployment group name; tags, Auto Scaling group names, or both; and the deployment configuration), simply specify those settings on this page. Although this new deployment group and the existing deployment group will have the same name, AWS CodeDeploy treats them as separate deployment groups, because they are each associated with separate applications.

6. In the list of tags, select the tag type and fill in the **Key** and **Value** boxes with the value of the key-value pair that you have tagged (or will later tag) the instances with.

   As you begin adding key-value pair information, a new row appears for you to add another key-value pair if desired. You can repeat this step for up to 10 key-value pairs.

   **Tip**
   As AWS CodeDeploy finds instances that match each specified key-value pair, it displays the number of matching instances. To see more information about the instances, click the number.

   To remove a key-value pair from the list, click the corresponding remove icon.
7. In the **Deployment Config** list, select the desired deployment configuration.

8. In the **Service Role ARN** box, select an existing service role Amazon Resource Name (ARN) that trusts AWS CodeDeploy with at minimum the trust and permissions that are described in Create a Service Role (p. 140). To get the service role ARN, see Use the IAM Console to Get the Service Role ARN (p. 143).
9. Click **Create Application**. The application is created with AWS CodeDeploy, and the corresponding deployment group is created. A page with information about the newly-created application then appears.

The next step is to prepare a revision to deploy to the specified application and deployment group. For instructions, see Prepare a Revision (p. 114).

# Create an Application by Using the AWS CLI

To create a new application by using the AWS CLI, call the create-application command, specifying a name that uniquely represents the application. This name must be unique across all of the applications that you create with AWS CodeDeploy associated with your AWS account.

After you create a new application, you must then create a deployment group, which specifies the instances to deploy revisions to. For instructions, follow the instructions in Create a Deployment Group (p. 138).

After you create the deployment group, the next step is to prepare a revision to deploy to the specified application and deployment group. For instructions, see Prepare a Revision (p. 114).

# Prepare a Revision for AWS CodeDeploy

A revision in AWS CodeDeploy contains a certain version of the source files that you want AWS CodeDeploy to deploy to your instances, scripts that you want AWS CodeDeploy to run on your instances, or both.

To prepare a revision for deployment, you plan the revision, add an AppSpec file to the revision, and then push the revision to Amazon S3 or GitHub. After you push the revision, you can deploy it.

**Topics**



## Plan a Revision for AWS CodeDeploy

The first step in creating a revision to deploy is to plan the revision. Good planning makes deploying revisions to instances much easier.

Regardless of the type of revision that you want to create, you start by creating an empty root directory (folder) on the development machine. This folder will store your source files (such as text and binary files, executables, packages, and so on) to be deployed to the instances, scripts to be run on the instances, or both, as follows:

| | |
|---|---|
| I want to only copy source files onto the instances. | In the root folder, add the source files that you want to copy onto the instances. |
| I want to only run scripts on the instances. | In the root folder, add the scripts that you want to run on the instances. |
| I want to copy source files to the instances *and* run scripts on the instances. | In the root folder, add the source files that you want to copy onto the instances and the scripts that you want to run on the instances. |

For example, your root folder may contain various files like this, starting for example at the `/tmp/` root folder in Linux, OS X, or Unix or the `c:\temp` root folder in Windows:

```
/tmp/ or c:\temp (root folder)
  |--content (subfolder)
  |    |--myTextFile.txt
  |    |--mySourceFile.rb
  |    |--myExecutableFile.exe
  |    |--myInstallerFile.msi
  |    |--myPackage.rpm
  |    |--myImageFile.png
  |--scripts (subfolder)
  |    |--myShellScript.sh
  |    |--myBatchScript.bat
  |    |--myPowerShellScript.ps1
  |--appspec.yml
```

After you have added your source files, scripts, or both to the root folder, the next step is to add an Application Specification File (AppSpec file) in the root folder as shown above. For instructions, see Add an AppSpec File (p. 115).

# Add an AppSpec File to a Revision for AWS CodeDeploy

After you plan your revision as described in Plan a Revision (p. 114), the next step is to add an Application Specification File (AppSpec file) to the revision. The AppSpec file is unique to AWS CodeDeploy. It instructs AWS CodeDeploy what to do and when with files, scripts, or both as it deploys the revision to the instances. Without an AppSpec file, AWS CodeDeploy won't know what to do with the files or scripts in the revision during a deployment.

Each revision must contain one and only one AppSpec file. The file must be named `appspec.yml`. The AppSpec file must be added to the root directory of the application's source content's directory structure.

To add an AppSpec file to a revision, copy the following template into your preferred text editor. Modify the template as needed, and save the file as `appspec.yml` within the root directory of the revision. For help completing this template, see the AppSpec File Reference (p. 212).

**Note**
The lines in the template starting with "#" are instructional comments and can be safely left in the file or removed.

```
# This is an appspec.yml template file for use with AWS CodeDeploy.
# The lines in this template starting with the hashtag symbol are
#   instructional comments and can be safely left in the file or
#   ignored.
# For help completing this file, see the "AppSpec File Reference" in the
#   "AWS CodeDeploy User Guide" at
#   http://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html
version: 0.0
# Specify "os: linux" if this revision targets Amazon Linux or Ubuntu Server
#   instances.
# Specify "os: windows" if this revision targets Windows Server instances.
# (You cannot specify both "os: linux" and "os: windows".)
os: linux
```

```
# os: windows
# During the Install deployment lifecycle event (which occurs between the
#   BeforeInstall and AfterInstall events), copy the specified files
#   in "source" starting from the root of the revision's file bundle
#   to "destination" on the instance.
# Specify multiple "source" and "destination" pairs if you want to copy
#   from multiple sources or to multiple destinations.
# If you are not copying any files to the instance, then remove the
#   "files" section altogether. A blank or incomplete "files" section
#   may cause associated deployments to fail.
files:
  - source:
    destination:
  - source:
    destination:
# For deployments to Amazon Linux and Ubuntu Server instances,
#   you can specify a "permissions"
#   section here that describes special permissions to apply to the files
#   in the "files" section as they are being copied over to
#   the instance.
#   For more information, see the documentation.
# If you are deploying to Windows Server instances,
#   then remove the
#   "permissions" section altogether. A blank or incomplete "permissions"
#   section may cause associated deployments to fail.
permissions:
  - object:
    pattern:
    except:
    owner:
    group:
    mode:
    acls:
      -
    context:
      user:
      type:
      range:
    type:
      -
# If you are not running any commands on the instance, then remove
#   the "hooks" section altogether. A blank or incomplete "hooks" section
#   may cause associated deployments to fail.
hooks:
# For each deployment lifecycle event, specify multiple "location" entries
#   if you want to run multiple scripts during that event.
# You can specify "timeout" as the number of seconds to wait until failing the
 deployment
#   if the specified scripts do not run within the specified time limit for the

#   specified event. For example, 900 seconds is 15 minutes. If not specified,

#   the default is 1800 seconds (30 minutes).
#   Note that the maximum amount of time that all scripts must finish executing

#   for each individual deployment lifecycle event is 3600 seconds (1 hour).
#   Otherwise, the deployment will stop and AWS CodeDeploy will consider the
deployment
```

```
#   to have failed to the instance. Make sure that the total number of seconds

#  that are specified in "timeout" for all scripts in each individual deployment

#   lifecycle event does not exceed a combined 3600 seconds (1 hour).
# For deployments to Amazon Linux and Ubuntu Server  instances,
#   you can specify "runas" in an event to
#   run as the specified user. For more information, see the documentation.
#   If you are deploying to Windows Server instances,
#   remove "runas" altogether.
# If you do not want to run any commands during a particular deployment
#   lifecycle event, remove that event declaration altogether. Blank or
#   incomplete event declarations may cause associated deployments to fail.
# During the ApplicationStop deployment lifecycle event, run the commands
#   in the script specified in "location" starting from the root of the
#   revision's file bundle.
  ApplicationStop:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the BeforeInstall deployment lifecycle event, run the commands
#   in the script specified in "location".
  BeforeInstall:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the AfterInstall deployment lifecycle event, run the commands
#   in the script specified in "location".
  AfterInstall:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the ApplicationInstall deployment lifecycle event, run the commands
#   in the script specified in "location".
  ApplicationStart:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the ValidateService deployment lifecycle event, run the commands
#   in the script specified in "location".
  ValidateService:
    - location:
      timeout:
      runas:
    - location:
```

```
    timeout:
    runas:
```

You should verify that you have placed your AppSpec file in the root directory of the application's source content's directory structure. To do this, run one of the following commands:

For Linux, OS X, or Unix:

```
find /path/to/root/directory -name appspec.yml
```

If the AppSpec file is *not* located there, there will be no output.

For Windows:

```
dir path\to\root\directory\appspec.yml
```

If the AppSpec file is *not* located there, a **File Not Found** error displays.

Also, you should check the validity of your AppSpec file by using a YAML validator. If you don't have one, search the Internet for "YAML validator."

After you add the AppSpec file to the revision, you're ready to push the revision to Amazon S3 or GitHub. For instructions, see Push a Revision (p. 118).

# Push a Revision for AWS CodeDeploy to Amazon S3

After you plan your revision as described in Plan a Revision (p. 114) and add an AppSpec file to the revision as described in Add an AppSpec File (p. 115), you are ready to bundle the component files together and then push the revision to Amazon S3. After you push the revision, you can use AWS CodeDeploy to deploy the revision from Amazon S3 to the instances.

> **Note**
> AWS CodeDeploy also supports deploying revisions that have been pushed to GitHub. To push a revision to a GitHub repository, skip this page and see your GitHub documentation.

The following instructions assume that you have already followed the instructions in Setting Up (p. 4) to set up the AWS CLI. This is especially important for calling the **push** command described later.

Also, be sure that you have an Amazon S3 bucket to push the revision to. If you do not have a bucket yet, follow the instructions in Create a Bucket.

The target Amazon S3 bucket must be created or exist in the same region as the target instances. For example, if you want to deploy a particular revision to some instances in the US East (N. Virginia) region and other instances in the US West (Oregon) region, then you must have one bucket in the US East (N. Virginia) region with one copy of the revision and another bucket in the US West (Oregon) region with another copy of the same revision. In this scenario, you would then need to create 2 separate deployments, one in the US East (N. Virginia) region and another in the US West (Oregon) region, even though the revision is the same in both regions and buckets.

The Amazon S3 bucket must allow you to upload to it. One way to specify this is through an Amazon S3 bucket policy. For example, the following Amazon S3 bucket policy allows AWS account `111122223333` to upload anywhere within the Amazon S3 bucket named `CodeDeployDemoBucket`:

```
{
  "Statement": [
    {
      "Action": ["s3:PutObject"],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::CodeDeployDemoBucket/*",
      "Principal": {
        "AWS": [
          "111122223333"
        ]
      }
    }
  ]
}
```

To learn how to generate and attach an Amazon S3 bucket policy, see Bucket Policy Examples.

The IAM user who is calling the **push** command must at minimum have permissions to upload the revision to each target Amazon S3 bucket. One way to specify this is through an IAM policy. For example, the following policy allows the IAM user to upload revisions anywhere within the Amazon S3 bucket named `CodeDeployDemoBucket`:

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":["s3:PutObject"],
      "Resource":"arn:aws:s3:::CodeDeployDemoBucket/*"
    }
  ]
}
```

To learn how to create and attach an IAM policy, see Managing IAM Policies (AWS Management Console).

To bundle and push the revision in a single command, from the command line, switch to the root directory (folder) of the revision, and then call the **push** command.

For example, to bundle the component files together into a revision starting from the current directory, associated with the application named `WordPress_App`, to an Amazon S3 bucket named `CodeDeployDemoBucket`, with a revision name of `WordPressApp.zip`, call the **push** command as follows:

In Linux, OS X, or Unix:

```
aws deploy push \
  --application-name WordPress_App \
  --description "This is a revision for the application WordPress_App" \
  --ignore-hidden-files \
  --s3-location s3://CodeDeployDemoBucket/WordPressApp.zip \
  --source .
```

In Windows:

```
aws deploy push --application-name WordPress_App --description "This is a revi
sion for the application WordPress_App" --ignore-hidden-files --s3-location
s3://CodeDeployDemoBucket/WordPressApp.zip --source .
```

After the push succeeds, you can use the AWS CLI or the AWS CodeDeploy console to deploy the revision from Amazon S3 to the instances. For instructions, see Deploy a Revision (p. 121).

# Deploy a Revision with AWS CodeDeploy

After you have prepared your instances to deploy to as described in Configure Instances (p. 72), created the application as described in Create an Application (p. 111), and prepared your revision to deploy as described in Prepare a Revision (p. 114), you are finally ready to deploy your revision to the instances.

You can use the AWS CodeDeploy console, the AWS CLI, or the AWS CodeDeploy APIs to deploy revisions that you have already pushed to Amazon S3 or GitHub.

**Caution**
You *cannot* start following these steps if:

- You want to deploy the revision using a custom deployment configuration, but you have not yet created the new custom deployment configuration. Instead, follow the instructions in Create a Deployment Configuration (p. 148), and then follow the steps below.
- You want to deploy a revision from an Amazon S3 bucket, and the Amazon S3 bucket that you are deploying from does not allow the target instances to download the revision from the bucket. One way to specify this is through an Amazon S3 bucket policy. For example, the following Amazon S3 bucket policy allows any Amazon EC2 instance with an attached IAM instance profile containing the ARN `arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo` to download from anywhere within the Amazon S3 bucket named `CodeDeployDemoBucket`:
- 

```
{
  "Statement": [
    {
      "Action": ["s3:Get*", "s3:List*"],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::CodeDeployDemoBucket/*",
      "Principal": {
        "AWS": [
          "arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo"
        ]
      }
    }
  ]
}
```

And the following Amazon S3 bucket policy allows any on-premises instance with an associated IAM user containing the ARN `arn:aws:iam::80398EXAMPLE:user/CodeDeployUser` to download from anywhere within the Amazon S3 bucket named `CodeDeployDemoBucket`:

```
{
  "Statement": [
    {
      "Action": ["s3:Get*", "s3:List*"],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::CodeDeployDemoBucket/*",
      "Principal": {
        "AWS": [
          "arn:aws:iam::80398EXAMPLE:user/CodeDeployUser"
        ]
      }
    }
  ]
}
```

To learn how to generate and attach an Amazon S3 bucket policy, see Bucket Policy Examples.

- You are deploying your own application revision from an Amazon S3 bucket, and the Amazon S3 bucket that you are deploying from is in a different AWS region than the region that your target instances are in. To proceed, you would first need to copy the revision over to an Amazon S3 bucket that is in the same region that your target instances are in, and then you can follow the steps below.

**Topics**



# Deploy a Revision by Using the AWS CodeDeploy Console

To deploy a revision by using the AWS CodeDeploy console:

1. Prepare the instances, create the application, and push the revision, as referred to at the beginning of this page.
2. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).
3. In the AWS CodeDeploy service navigation bar, click **Applications**.
4. In the list of applications, click the name of the application that corresponds to the revision that you want to deploy.
5. On the application details page, in the **Deployment Groups** area, click the arrow next to the deployment group that you want to deploy the new revision to. The box expands to show deployment group details.

**Tip**
If you already have a revision associated with this application that you've previously deployed, then on the right side of the application details page, in the **Revisions** area, click the drop-down area next to the revision; click **Deploy This Revision**; follow the on-screen directions; and skip the rest of the steps on this page. To track the status of your deployment, see View Deployment Details (p. 127).

6. In the deployment group details area, click **Deploy New Revision**.
7. On the **Create New Deployment** page, if the revision that you want to deploy is stored in an Amazon S3 bucket, click **My application revision is stored in Amazon S3**. Otherwise, click **My application revision is stored in GitHub**. Then complete one of the following sets of instructions to specify information about the revision and then deploy the revision.

# To specify information about a revision that is stored in an Amazon S3 bucket

1. Copy your revision's Amazon S3 link into the **Revision Location** box. To find the link value:

   1. In a separate browser tab, sign in to the Amazon S3 console:

      Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

      Then browse to and select your revision in the Amazon S3 console.
   2. If the **Properties** pane is not visible in the Amazon S3 console, click the **Properties** button.
   3. In the **Properties** pane, copy the value of the **Link** field into the **Revision Location** box in the AWS CodeDeploy console.

   If you want to specify an ETag as part of the revision location:

   - If the **Link** field value ends in **`?versionId=`*`versionId`***, add **`&etag=`** and the ETag to the end of the **Link** field value after you copy it into the **Revision Location** box in the AWS CodeDeploy console.
   - If the **Link** field value does not specify a version ID, add **`?etag=`** and the ETag to the end of the **Link** field value after you copy it into the **Revision Location** box in the AWS CodeDeploy console.

   **Note**
   Although it's not as easy as copying the value of the **Link** field in the Amazon S3 console, the **Revision Location** box also accepts any of the following formats:

   - **`s3://`*`bucketName`*`/`*`folders`*`/`*`objectName`***
   - **`s3://`*`bucketName`*`/`*`folders`*`/`*`objectName`*`?versionId=`*`versionId`***
   - **`s3://`*`bucketName`*`/`*`folders`*`/`*`objectName`*`?etag=`*`etag`***
   - **`s3://`*`bucketName`*`/`*`folders`*`/`*`objectName`*`?versionId=`*`versionId`*`&etag=`*`etag`***
   - ***`bucketName`*`.s3.amazonaws.com/`*`folders`*`/`*`objectName`***

   After you paste or type the revision location and leave the **Revision Location** box, the **File Type** list appears.
2. In the **File Type** list, if a message appears stating that the file type could not be detected, select the revision's file type.
3. Optionally, in the **Deployment Description** box, type any description that you want to associate with this deployment.

4. In the **Deployment Config** list, select the desired deployment configuration.
5. Click **Deploy Now**. AWS CodeDeploy deploys the revision and then displays the **Deployments** page.

   To track the status of your deployment, see View Deployment Details (p. 127).

# To specify information about a revision that is stored in a GitHub repository

1. Click **Connect with GitHub**. A new web page appears, prompting you to authorize AWS CodeDeploy to interact with GitHub on behalf of your GitHub account.

   **Note**
   If you see a **Reconnect with GitHub** link instead of a **Connect with GitHub** button, this is expected behavior. Do not click the link; continue to the next step. For information about why this happens, see GitHub Authentication Behaviors with Applications in AWS CodeDeploy (p. 178).
   If you see a blank web page that briefly appears and then disappears, and you see neither a **Reconnect with GitHub** link or a **Connect with GitHub** button, this also expected behavior; continue to the next step. For information about why this happens, see GitHub Authentication Behaviors with Applications in AWS CodeDeploy (p. 178).
2. If you are prompted to sign in to GitHub, follow the on-screen instructions on the **Sign in** page to sign in with your preferred GitHub user name or email and password.
3. If an **Authorize application** page appears, click **Authorize application**. The web page disappears.
4. Back in the **Create New Deployment** page, in the **Repository Name** box, type the GitHub user or organization name that is associated with the repository that contains the revision, followed by a forward slash character (/), followed by the name of the repository that contains the revision. If you are unsure of the value to type:

   1. In a separate web browser tab, go to your GitHub dashboard.
   2. In the **Your repositories** area, hover your mouse pointer over the target repository name. A tooltip appears, displaying the GitHub user or organization name, followed by a forward slash character (/), followed by the name of the repository. Type this displayed value into the **Repository Name** box.

      **Tip**
      If the target repository name is not visible in the **Your repositories** area, use the **Search GitHub** box to find the target repository name and the corresponding GitHub user or organization name.

5. In the **Commit ID** box, type the ID of the commit that refers to the revision in the repository. If you are unsure of the value to type:

   1. In a separate web browser tab, go to your GitHub dashboard.
   2. In the **Your repositories** area, click the repository name that contains the target commit.

      **Tip**
      If the target repository name is not visible in the **Your repositories** area, use the **Search GitHub** box to find the target repository and corresponding GitHub user or organization name, and then click on the target repository's name.

   3. In the list of commits, find and copy the commit ID that refers to the revision in the repository. This ID is typically 40 characters in length and consists of both letters and numbers. (Do not use

the shorter version of the commit ID, which is typically the first 10 characters of the longer version of the commit ID.)

4. Paste the commit ID into the **Commit ID** box.

6. Optionally, in the **Deployment Description** box, type any description that you want to associate with this deployment.
7. In the **Deployment Config** list, select the desired deployment configuration.
8. Click **Deploy Now**. AWS CodeDeploy deploys the revision and then displays the **Deployments** page.

   To track the status of your deployment, see View Deployment Details (p. 127).

# Deploy a Revision by Using the AWS CLI

To deploy a revision by using the AWS CLI:

1. Prepare the instances, create the application, and push the revision, as referred to at the beginning of this page.
2. If you want to deploy a revision from an Amazon S3 bucket, skip ahead to the next step to call the **create-deployment** command. Otherwise, if you want to deploy a revision from a GitHub repository by using the AWS CLI, you must first give AWS CodeDeploy permission to interact with GitHub on behalf of your GitHub account for the specified application. Currently, the only way to do this is through the AWS CodeDeploy console, and you will only need to do it once for an application:

   1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

      **Note**
      Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

   2. On the AWS CodeDeploy service navigation bar, click **Deployments**.
   3. Click **Create New Deployment**.

      **Note**
      You will not be creating a new deployment through this page. However, this is currently the only way to give AWS CodeDeploy permission to interact with GitHub on behalf of your GitHub user account for the specified application.

   4. In the **Application** drop-down list, select the application that you want to link to your preferred GitHub user account.
   5. In the **Deployment Group** drop-down list, select any available deployment group.
   6. Next to **Revision Type**, click **My application revision is stored in GitHub**.
   7. Click **Connect With GitHub**.

      **Note**
      If you see a **Reconnect with GitHub** link instead of a **Connect with GitHub** button, you may have already authorized AWS CodeDeploy to interact with GitHub on behalf of a different GitHub account for the application. Or you may have revoked access for AWS CodeDeploy to interact with GitHub on behalf of the signed-in GitHub account for all applications that the account is linked to in AWS CodeDeploy. For more information, see GitHub Authentication Behaviors with Applications in AWS CodeDeploy (p. 178).

   8. If you are not already signed in to GitHub, follow the on-screen instructions on the **Sign in** page to sign in with your preferred GitHub user name or email and password.
   9. On the **Authorize application** page, click **Authorize application**. The web page disappears.

10. Now that AWS CodeDeploy has the necessary permission, click **Cancel** to stop using the **Create New Deployment** page, and continue using the AWS CLI.

3. Call the create-deployment command, specifying:
   - An existing application name. To view a list of existing application names, call the list-applications command.
   - An existing Amazon EC2 deployment group name. To view a list of existing deployment group names, call the list-deployment-groups command.
   - Information about the revision to be deployed:

     For revisions stored in Amazon S3:
     - The Amazon S3 bucket name containing the revision.
     - The name and file type of the uploaded revision.

       **Note**
       The tar and compressed tar archive file formats (.tar and .tar.gz) are not supported for Windows Server instances.
     - Optionally, the revision's Amazon S3 version identifier. (If the version identifier is not specified, AWS CodeDeploy will use the most recent version.)
     - Optionally, the revision's ETag. (If the ETag is not specified, AWS CodeDeploy will skip object validation.)

     For revisions stored in GitHub:
     - The GitHub user or group name assigned to the repository that contains the revision, followed by a forward slash character (`/`), followed by the repository name that contains the revision.
     - The ID of the commit that references the revision in the repository.
   - Optionally, the name of an existing deployment configuration to use. To view a list of existing deployment configurations, call the list-deployment-configs command. (If not specified, AWS CodeDeploy will use a specific default deployment configuration.)
   - Optionally, if the deployment causes the **ApplicationStop** deployment lifecycle event to fail to a specific instance, whether the deployment will be considered to have failed to that instance at that point and will continue on to the **BeforeInstall** deployment lifecycle event.
   - Optionally, any description that you want to associated with the deployment.

   **Tip**
   To specify information about a revision in Amazon S3 directly on the command line, use this syntax as part of the **create-deployment** call (`version` and `eTag` are optional):

   ```
   --s3-location bucket=string,key=string,bundleType=tar|tgz|zip,ver
   sion=string,eTag=string
   ```

   To specify information about a revision in GitHub directly on the command line, use this syntax as part of the **create-deployment** call :

   ```
   --github-location repository=string,commitId=string
   ```

   To get information about pushed revisions that have also been preregistered with AWS CodeDeploy, call the list-application-revisions command.

To track the status of your deployment, see View Deployment Details (p. 127).

# Monitor a Deployment with AWS CodeDeploy

You can use both the AWS CodeDeploy console and the AWS CLI to keep track of your deployments and their various components.

**Topics**



## View Deployment Details with AWS CodeDeploy

You can use the AWS CodeDeploy console, the AWS CLI, or the AWS CodeDeploy APIs to view details about existing deployments that are associated with your AWS account.

**Topics**



### View Deployment Details by Using the AWS CodeDeploy Console

To view deployment details by using the AWS CodeDeploy console:

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

**Note**
Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. In the AWS CodeDeploy service navigation bar, click **Deployments**. The **Deployments** section shows a list of deployments and their details.

   **Tip**
   If no entries are displaying but you expect to see some, make sure that the correct corresponding region is selected. On the navigation bar, in the region selector, click one of the supported regions (p. 192). AWS CodeDeploy supports only certain regions.

3. To see additional details for a single deployment, in the **Deployments** section, click the deployment ID.

## View Deployment Details by Using the AWS CLI

To view deployment details by using the AWS CLI, you can call the **get-deployment** command or the **batch-get-deployments** command to view details about single or multiple deployments, respectively. You can call the **list-deployments** command to get a list of unique deployment IDs to use as inputs to the **get-deployment** command and the **batch-get-deployments** command.

To view details about a single deployment, call the get-deployment command, specifying the unique deployment identifier. To get the deployment ID, call the list-deployments command.

To view details about multiple deployments, call the batch-get-deployments command, specifying multiple unique deployment identifiers. To get the deployment IDs, call the list-deployments command.

To view a list of existing deployment IDs, call the list-deployments command, specifying:

- The name of the existing application that is associated with the deployment that you want to find. To view a list of existing application names, call the list-applications command.
- The name of the existing deployment group that is associated with the deployment that you want to find. To view a list of existing deployment group names, call the list-deployment-groups command.
- Optionally, whether to include details about specific deployments by their deployment status. (If not specified, all matching deployments will be listed, regardless of their deployment status.)
- Optionally, whether to include details about specific deployments by their deployment creation start times or end times, or both. (If not specified, all matching deployments will be listed, regardless of their creation times.)

# View Instance Details with AWS CodeDeploy

You can use the AWS CodeDeploy console, the AWS CLI, or the AWS CodeDeploy APIs to view details about instances that participate in a specific deployment that is associated with your AWS account.

**Topics**



## View Instance Details by Using the AWS CodeDeploy Console

To view instance details by using the AWS CodeDeploy console:

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. In the AWS CodeDeploy service navigation bar, click **Deployments**. The **Deployments** section shows a list of deployments and their details.

   **Tip**
   If no entries are displaying but you expect to see some, make sure that the correct corresponding region is selected. On the navigation bar, in the region selector, click one of the supported regions (p. 192). AWS CodeDeploy supports only certain regions.

3. In the **Deployments** section, click the arrow next to the deployment ID for the deployment that corresponds to the instances. Additional deployment details are displayed.
4. In the **Instances** section, click **View All Instances**. A deployment details page is displayed, including a list of instances that are participating in the deployment, along with a few details about those instances.

   **Tip**
   The page's details will occasionally refresh. If you want to update the details faster, use your web browser's page refresh command.

5. To see information about individual deployment lifecycle events for a participating instance, on the deployment details page, click **View Events** in the **Events** column for the corresponding instance. An instance details page appears.

   **Tip**
   If any of the deployment lifecycle events shows as **Failed**, you can try to determine why by clicking **View Logs**, **View in EC2**, or both in the instance details page. If you click **View in EC2**, a new web browser tab opens and displays the Amazon EC2 console's **Instances** page; to get back to the instance details page, switch back to the original web browser tab.
   You can also try following the instructions in Troubleshooting Instance Issues (p. 198).

6. If **View in EC2** is not visible on the instance details page and you want to see additional information about a participating Amazon EC2 instance, return to the deployment details page, and then click the corresponding Amazon EC2 instance's ID in the **Instance ID** column.

   **Note**
   After you click the Amazon EC2 instance ID, the deployment details page is replaced by the Amazon EC2 console's **Instances** page on the same web browser tab. To get back to the deployment details page, repeat the steps in this section.

# View Instance Details by Using the AWS CLI

To view instance details by using the AWS CLI, you can call either the **get-deployment-instance** command or the **list-deployment-instances** command to view details about single or multiple instances, respectively.

To view details about a single instance, call the get-deployment-instance command, specifying:

- The unique deployment identifier. To get the deployment ID, call the list-deployments command.
- The unique instance identifier. To get the instance ID, call the list-deployment-instances command, as described next.

To view a list of IDs for instances that participate in a deployment, call the list-deployment-instances command, specifying:

- The unique deployment ID. To get the deployment ID, call the list-deployments command.

- Optionally, whether to include only specific instance IDs by their deployment status. (If not specified, all matching instance IDs will be listed, regardless of their deployment status.)

# View Application Details with AWS CodeDeploy

You can use the AWS CodeDeploy console, the AWS CLI, or the AWS CodeDeploy APIs to view details about all existing applications that are associated with your AWS account.

**Topics**



## View Application Details by Using the AWS CodeDeploy Console

To view application details by using the AWS CodeDeploy console:

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. In the AWS CodeDeploy service navigation bar, click **Applications**. A list of existing application names appears.

   **Tip**
   If no entries are displaying but you expect to see some, make sure that the correct corresponding region is selected. On the navigation bar, in the region selector, click one of the supported regions (p. 192). AWS CodeDeploy supports only certain regions.

3. To view additional application details, click the application name in the list.

## View Application Details by Using the AWS CLI

To view application details by using the AWS CLI, you can call the **get-application** command, the **batch-get-application** command, or the **list-applications** command to view information about applications.

To view details about a single application, call the get-application command, specifying the application name.

To view details about multiple applications, call the batch-get-applications command, specifying multiple application names.

To view a list of application names, call the list-applications command.

# View Deployment Group Details with AWS CodeDeploy

You can use the AWS CodeDeploy console, the AWS CLI, or the AWS CodeDeploy APIs to view details about all deployment groups that are associated with a specific application that is associated with your AWS account.

**Topics**



## View Deployment Group Details by Using the AWS CodeDeploy Console

To view deployment group details by using the AWS CodeDeploy console:

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. If the **Applications** page is not visible, on the AWS CodeDeploy service navigation bar, click **Applications**.
3. On the **Applications** page, click the application name that is associated with the deployment group that you want to view information about. The application details page appears, displaying a list of associated deployment groups in the **Deployment Groups** area.

   **Tip**
   If no entries are displaying but you expect to see some, make sure that the correct corresponding region is selected. On the navigation bar, in the region selector, click one of the supported regions (p. 192). AWS CodeDeploy supports only certain regions.

4. To view details about an individual deployment group, in the **Deployment Groups** area, click the arrow next to the deployment group that you want to see details about. Then click **Deployment Group Details**.

## View Deployment Group Details by Using the AWS CLI

To view deployment group details by using the AWS CLI, you can call either the **get-deployment-group** command or the **list-deployment-groups** command to view details about single or multiple deployment groups, respectively.

To view details about a single deployment group, call the get-deployment-group command, specifying:

- The existing application name that is associated with the deployment group. To get the application name, call the list-applications command.
- The existing deployment group name. To get the deployment group name, call the list-deployment-groups command, as described next.

To view a list of deployment group names, call the list-deployment-groups command, specifying the existing application name that is associated with the deployment groups. To get the application name, call the list-applications command.

# View Application Revision Details with AWS CodeDeploy

You can use the AWS CodeDeploy console, the AWS CLI, or the AWS CodeDeploy APIs to view details about all application revisions that are registered to your AWS account for a specific application.

**Topics**



## View Application Revision Details by Using the AWS CodeDeploy Console

To view application revision details by using the AWS CodeDeploy console:

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. In the AWS CodeDeploy service navigation bar, click **Applications**.
3. On the **Applications** page, click the name of the application that you want to view revision details about.

   **Tip**
   If no entries are displaying but you expect to see some, make sure that the correct corresponding region is selected. On the navigation bar, in the region selector, click one of the supported regions (p. 192). AWS CodeDeploy supports only certain regions.

4. On the application details page, the **Revisions** section displays a list of revisions that are associated with the application. To view additional details about a specific revision, click the arrow next to that revision.

## View Application Revision Details by Using the AWS CLI

To view application revision details by using the AWS CLI, you can call either the **get-application-revision** command or the **list-application-revisions** command to view details about single or multiple application revisions, respectively.

To view details about a single application revision, call the get-application-revision command, specifying:

- The existing application name. To get the application name, call the list-applications command.
- For a revision stored in GitHub, the GitHub repository name and the ID of the commit that references the application revision that was pushed to the repository.

- For a revision stored in Amazon S3, the Amazon S3 bucket name containing the revision; the name and file type of the uploaded archive file; and optionally the archive file's Amazon S3 version identifier and ETag. If the version identifier, the ETag, or both were specified when originally registering the revision during a call to register-application-revision, they must be specified here.

To view details about multiple application revisions, call the list-application-revisions command, specifying:

- The existing application name. To get the application name, call the list-applications command.
- Optionally, to view details only for Amazon S3 application revisions, the Amazon S3 bucket name containing the revisions.
- Optionally, whether to list revision details based on whether each revision is the target revision of a deployment group. (If not specified, AWS CodeDeploy will list all matching revisions.)
- Optionally, to view details only for Amazon S3 application revisions, a prefix string to limit the search for Amazon S3 application revisions. (If not specified, AWS CodeDeploy will list all matching Amazon S3 application revisions.)
- Optionally, the column name and order by which to sort the list of revision details. (If not specified, AWS CodeDeploy will list results in an arbitrary order.)

You can list all revisions or revisions stored only in Amazon S3. You cannot list revisions stored only in GitHub.

# View Deployment Configuration Details with AWS CodeDeploy

You can use the AWS CodeDeploy console, the AWS CLI, or the AWS CodeDeploy APIs to view details about all existing deployment configurations that are associated with your AWS account.

**Topics**



## View Deployment Configuration Details by Using the AWS CodeDeploy Console

To view a list of existing deployment configuration names by using the AWS CodeDeploy console:

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. In the AWS CodeDeploy service navigation bar, click **Deployment Configurations**. A list of available deployment configuration names appears, along with the criteria for each deployment configuration.

   **Tip**
   If no entries are displaying but you expect to see some, make sure that the correct corresponding region is selected. On the navigation bar, in the region selector, click one of the supported regions (p. 192). AWS CodeDeploy supports only certain regions.

# View Deployment Configuration Details by Using the AWS CLI

To view deployment configuration details by using the AWS CLI, you can call either the **get-deployment-config** command or the **list-deployment-configs** command to view details about single or multiple deployment configurations, respectively.

To view details about a single deployment configuration, call the get-deployment-config command, specifying the unique deployment configuration name.

To view a list of existing deployment configuration names, call the list-deployment-configs command.

# Advanced Tasks for AWS CodeDeploy

After you are comfortable with preparing instances to participate in AWS CodeDeploy deployments, creating applications in AWS CodeDeploy, preparing revisions to deploy to those instances, and then using AWS CodeDeploy to deploy those revisions, you can get the most out of AWS CodeDeploy through completing the following advanced tasks.

**Topics**



## Create a New Deployment with AWS CodeDeploy

Typically, to create a new deployment, you would follow the instructions in Deploy a Revision (p. 121). However, the AWS CodeDeploy console provides another way to create a new deployment through the **Create New Deployment** page. You may want to use that page, for example, if you want to create a new deployment on one page, and you already have instances, an application, a deployment group, a revision, and a deployment configuration readily available.

Use the following instructions to create a new deployment through the **Create New Deployment** page.

> **Note**
> These instructions assume that you have already followed the instructions in Configure Instances (p. 72), Create an Application (p. 111), and Prepare a Revision (p. 114).

**Tip**
You can also use the AWS CLI to create a deployment by calling the create-deployment command.

**Caution**
You *cannot* start following these steps if you want to create a new deployment that uses a custom deployment configuration, but you have not yet created the new custom deployment configuration. Instead, follow the instructions in Create a Deployment Configuration (p. 148), and then follow the steps below.
Also, you *cannot* start following these steps if you are using your own application to deploy your own revision from an Amazon S3 bucket, and the Amazon S3 bucket that you are deploying from is in a different AWS region than the region that your target instances are in. To proceed, you would first need to copy the revision over to an Amazon S3 bucket that is in the same region that your target instances are in, and then you can follow the steps below.

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. In the AWS CodeDeploy service navigation bar, click **Deployments**.
3. On the **Deployments** page, click **Create New Deployment**.
4. In the **Application** list, select the name of the existing application that you want to use for this deployment.
5. In the **Deployment Group** box, select the name of the existing deployment group associated with the application that you want to use for this deployment.
6. Next to **Revision Type**, if the revision that you want to deploy is stored in an Amazon S3 bucket, click **My application revision is stored in Amazon S3**. Otherwise, click **My application revision is stored in GitHub**. Then complete one of the following sets of instructions to specify information about the revision and then deploy the revision.

# To specify information about a revision that is stored in an Amazon S3 bucket

1. Copy your revision's Amazon S3 link into the **Revision Location** box. To find the link value:

   1. In a separate browser tab, sign in to the Amazon S3 console:

      Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

      Then browse to and select your revision in the Amazon S3 console.
   2. If the **Properties** pane is not visible in the Amazon S3 console, click the **Properties** button.
   3. In the **Properties** pane, copy the value of the **Link** field into the **Revision Location** box in the AWS CodeDeploy console.

   If you want to specify an ETag (a file checksum) as part of the revision location:

   - If the **Link** field value ends in **`?versionId=`*`versionId`***, add **`&etag=`** and the ETag to the end of the **Link** field value after you copy it into the **Revision Location** box in the AWS CodeDeploy console.
   - If the **Link** field value does not specify a version ID, add **`?etag=`** and the ETag to the end of the **Link** field value after you copy it into the **Revision Location** box in the AWS CodeDeploy console.

**Note**
Although it's not as easy as copying the value of the **Link** field in the Amazon S3 console, the **Revision Location** box also accepts any of the following formats:

- `s3://bucketName/folders/objectName`
- `s3://bucketName/folders/objectName?versionId=versionId`
- `s3://bucketName/folders/objectName?etag=etag`
- `s3://bucketName/folders/objectName?versionId=versionId&etag=etag`
- `bucketName.s3.amazonaws.com/folders/objectName`

After you paste or type the revision location and leave the **Revision Location** box, the **File Type** list appears.

2. In the **File Type** list, if a message appears stating that the file type could not be detected, select the revision's file type. Otherwise, simply accept the detected file type.
3. Optionally, in the **Deployment Description** box, type any description that you want to associate with this deployment.
4. In the **Deployment Config** list, select the desired deployment configuration.
5. Click **Deploy Now**. AWS CodeDeploy deploys the revision and then displays the **Deployments** page.

To track the status of your deployment, see View Deployment Details (p. 127).

# To specify information about a revision that is stored in a GitHub repository

1. Click **Connect with GitHub**. A new web page appears, prompting you to authorize AWS CodeDeploy to interact with GitHub on behalf of your GitHub account.

   **Note**
   If you see a **Reconnect with GitHub** link instead of a **Connect with GitHub** button, this is expected behavior. Do not click the link; continue to the next step. For information about why this happens, see GitHub Authentication Behaviors with Applications in AWS CodeDeploy (p. 178).
   If you see a blank web page that briefly appears and then disappears, and you see neither a **Reconnect with GitHub** link or a **Connect with GitHub** button, this also expected behavior; continue to the next step. For information about why this happens, see GitHub Authentication Behaviors with Applications in AWS CodeDeploy (p. 178).
2. If you are prompted to sign in to GitHub, follow the on-screen instructions on the **Sign in** page to sign in with your preferred GitHub user name or email and password.
3. If an **Authorize application** page appears, click **Authorize application**. The web page disappears.
4. Back in the **Create New Deployment** page, in the **Repository Name** box, type the GitHub user or organization name that is associated with the repository that contains the revision, followed by a forward slash character (`/`), followed by the name of the repository that contains the revision. If you are unsure of the value to type:

   1. In a separate web browser tab, go to your GitHub dashboard.
   2. In the **Your repositories** area, hover your mouse pointer over the target repository name. A tooltip appears, displaying the GitHub user or organization name, followed by a forward slash character (`/`), followed by the name of the repository. Type this displayed value into the **Repository Name** box.

**Tip**
If the target repository name is not visible in the **Your repositories** area, use the **Search GitHub** box to find the target repository name and the corresponding GitHub user or organization name.

5. In the **Commit ID** box, type the ID of the commit that refers to the revision in the repository. If you are unsure of the value to type:
   1. In a separate web browser tab, go to your GitHub dashboard.
   2. In the **Your repositories** area, click the repository name that contains the target commit.

      **Tip**
      If the target repository name is not visible in the **Your repositories** area, use the **Search GitHub** box to find the target repository and corresponding GitHub user or organization name, and then click on the target repository's name.
   3. In the list of commits, find and copy the commit ID that refers to the revision in the repository. This ID is typically 40 characters in length and consists of both letters and numbers. (Do not use the shorter version of the commit ID, which is typically the first 10 characters of the longer version of the commit ID.)
   4. Paste the commit ID into the **Commit ID** box.
6. Optionally, in the **Deployment Description** box, type any description that you want to associate with this deployment.
7. In the **Deployment Config** list, select the desired deployment configuration.
8. Click **Deploy Now**. AWS CodeDeploy deploys the revision and then displays the **Deployments** page.

   To track the status of your deployment, see View Deployment Details (p. 127).

# Create a Deployment Group with AWS CodeDeploy

After you Create an Application (p. 111) in AWS CodeDeploy, you must specify an existing deployment group so that AWS CodeDeploy knows which instances to deploy your revisions to.

You can use the AWS CodeDeploy console, the AWS CLI, or the AWS CodeDeploy APIs to create deployment groups.

As part of creating a deployment group, you must specify a service role. For more information, see Create a Service Role (p. 140).

**Note**
To view a list of existing deployment groups that are already associated with your AWS account and that you can use instead of creating a new one here, see View Deployment Group Details (p. 131).

**Caution**
You *cannot* start following these steps if:

- Your instances are not prepared to participate in AWS CodeDeploy deployments. Although this not a strict requirement before you create a deployment group, we recommend that you set up your instances before you go through the process of deploying revisions to them. To

set up your instances, follow the instructions in Configure Instances (p. 72), and then follow the steps below.

- You want to create a new deployment group that uses a custom deployment configuration, but you have not yet created the new custom deployment configuration. Instead, follow the instructions in Create a Deployment Configuration (p. 148), and then follow the steps below.
- You do not have an existing service role that trusts AWS CodeDeploy with at minimum the trust and permissions that are described in Create a Service Role (p. 140). To create and configure a new service role with the correct permissions, follow the instructions in Create a Service Role (p. 140), and then follow the steps below.

**Topics**



# Create a Deployment Group by Using the AWS CodeDeploy Console

To create a deployment group by using the AWS CodeDeploy console:

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).
2. In the AWS CodeDeploy service navigation bar, click **Applications**.
3. On the **Applications** page, click the name of the application that you want to associate a deployment group with.
4. Click **Create New Deployment Group**.
5. In the **Deployment Group Name** box, type a name that describes the deployment group that you'll be deploying application revisions to.
6. In the list of tags, select the tag type and fill in the **Key** and **Value** boxes with the value of the key-value pair that you have tagged (or will later tag) the instances with.

   As you begin adding key-value pair information, a new row appears for you to add another key-value pair if desired. You can repeat this step for up to 10 key-value pairs.

   **Tip**
   As AWS CodeDeploy finds instances that match each specified key-value pair, it displays the number of matching instances. To see more information about the instances, click the number.

   To remove a key-value pair from the list, click the corresponding remove icon.
7. In the **Deployment Config** list, select the desired deployment configuration.
8. In the **Service Role ARN** box, select an existing service role Amazon Resource Name (ARN) that trusts AWS CodeDeploy with at minimum the trust and permissions that are described in Create a Service Role (p. 140). To get the service role ARN, see Use the IAM Console to Get the Service Role ARN (p. 143).
9. Click **Create Deployment Group**. The deployment group is created in AWS CodeDeploy, and a page with information about the newly-created deployment group then appears.

# Create a Deployment Group by Using the AWS CLI

To create a deployment group by using the AWS CLI, call the create-deployment-group command, specifying:

- The existing application name. To view a list of existing application names, call the list-applications command.
- A name that uniquely represents the deployment group. This name must be unique for each application that you associate the deployment group with.
- Information about the tags, Auto Scaling group names, or both, that identify the instances to be included in the deployment group.
- The service role Amazon Resource Name (ARN) identifier that allows AWS CodeDeploy to act on your AWS account's behalf when interacting with other related AWS services. To get the service role ARN, see Use the AWS CLI to Get the Service Role ARN (p. 144). For more information on service roles, see Roles - Terms and Concepts.
- Optionally, the name of an existing deployment configuration. To view a list of existing deployment configurations, see View Deployment Configuration Details (p. 133). (If not specified, AWS CodeDeploy uses a specific default deployment configuration.)

# Create a Service Role for AWS CodeDeploy

AWS CodeDeploy relies on tags or Auto Scaling group names to identify instances that it can deploy applications to. To do this, AWS CodeDeploy needs access to your instances to expand (read) their tags or to identify your Amazon EC2 instances by the Auto Scaling group names that they're associated with. These instructions show how to create a special type of IAM role—called a *service role*—to give AWS CodeDeploy permission to do this.

You can create a service role with the IAM console, the AWS CLI, or the IAM APIs.

**Topics**



## Use the IAM Console to Create a Service Role that is Compatible with AWS CodeDeploy

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

   **Important**
   Make sure that you are signed in to the AWS Management Console using the same account information that you used when setting up the AWS CLI in Setting Up (p. 4).
2. In the IAM console, in the navigation pane, click **Policies**, and then click **Create Policy**. (If a **Get Started** button appears, click it, and then click **Create Policy**.)
3. Next to **Create Your Own Policy**, click **Select**.
4. In the **Policy Name** box, type `CodeDeployAccess` (you can use any name you want).
5. In the **Policy Document** box, paste the following:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "autoscaling:PutLifecycleHook",
        "autoscaling:DeleteLifecycleHook",
        "autoscaling:RecordLifecycleActionHeartbeat",
        "autoscaling:CompleteLifecycleAction",
        "autoscaling:DescribeAutoscalingGroups",
        "autoscaling:PutInstanceInStandby",
        "autoscaling:PutInstanceInService",
        "ec2:Describe*",
        "Tag:get*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

6. Click **Create Policy**.
7. In the navigation pane, click **Roles**, and then click **Create New Role**.
8. In the **Role Name** box, give the service role a name, such as **`CodeDeployDemo`** (you can use any name you want). Then click **Next Step**.
9. On the **Select Role Type** page, with **AWS Service Roles** already selected, next to **Amazon EC2**, click **Select**.

   **Note**
   In the end, you will not be establishing trust for the Amazon EC2 service. In a later step, you will change this trust relationship to trust the AWS CodeDeploy service instead of the Amazon EC2 service.

10. On the **Attach Policy** page, check the box next to **`CodeDeployAccess`**, and then click **Next Step**.
11. Click **Create Role**.
12. In the list of roles, browse to and click the role that you just created.
13. Under **Trust Relationships**, click **Edit Trust Relationship**.
14. Replace the entire contents of the **Policy Document** box with the following policy, and then click **Update Trust Policy**:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "codedeploy.us-east-1.amazonaws.com",
          "codedeploy.us-west-2.amazonaws.com",
          "codedeploy.eu-west-1.amazonaws.com",
          "codedeploy.ap-southeast-2.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
```

```
    }
  ]
}
```

**Note**
If you don't care about restricting the service role by region, you can simply specify `codedeploy.amazonaws.com` as the service principal instead of `codedeploy.us-east-1.amazonaws.com`, `codedeploy.us-west-2.amazonaws.com`, and so on. This allows the service role to be used with AWS CodeDeploy in any supported region.
Also, after you click **Update Trust Policy**, you may notice that under **Trust Relationships**, the **Trusted Entities** area may continue to show that the Amazon EC2 service is still trusted. However, if you leave this page and then open the page again, the **Trusted Entities** area will update to show the new trusted entities.

15. Note the value of the **Role ARN** field, as you will need it to create deployment groups later. If you forget the value, you can get it later by following the instructions in Use the IAM Console to Get the Service Role ARN (p. 143).

You've now created a service role that allows AWS CodeDeploy to access your instances to expand (read) their tags or to identify your Amazon EC2 instances by the Auto Scaling group names that they're associated with.

For more general information about creating service roles, see Creating a Role to Delegate Permissions to an AWS Service in the *IAM User Guide*.

## Use the AWS CLI to Create a Service Role that is Compatible with AWS CodeDeploy

1. On your development machine, create a text file named `CodeDeployDemo-Trust.json` (you can name the file anything you want) with the following contents, which allows AWS CodeDeploy to work on your behalf:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "codedeploy.us-east-1.amazonaws.com",
          "codedeploy.us-west-2.amazonaws.com",
          "codedeploy.eu-west-1.amazonaws.com",
          "codedeploy.ap-southeast-2.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. In the same directory, create a text file named `CodeDeployDemo-Permissions.json` (you can name the file anything you want) with the following contents, which allows AWS CodeDeploy to

specifically find instances by their tags or Amazon EC2 instances in Auto Scaling group names on your behalf:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "autoscaling:PutLifecycleHook",
        "autoscaling:DeleteLifecycleHook",
        "autoscaling:RecordLifecycleActionHeartbeat",
        "autoscaling:CompleteLifecycleAction",
        "autoscaling:DescribeAutoscalingGroups",
        "autoscaling:PutInstanceInStandby",
        "autoscaling:PutInstanceInService",
        "ec2:Describe*",
        "Tag:get*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

3. From the same directory, call the **create-role** command to create a service role named **`CodeDeployDemo`** (you can name the service role anything you want), based on the information in the first file:

```
aws iam create-role --role-name CodeDeployDemo --assume-role-policy-document
 file://CodeDeployDemo-Trust.json
```

   In the command's output, note the value of the `Arn` entry under the `Role` object, as you'll need it to create a deployment group. If you forget the value, you can get it again later by following the instructions in Use the AWS CLI to Get the Service Role ARN (p. 144).

4. From the same directory, call the **put-role-policy** command to give the service role named **`CodeDeployDemo`** the permissions based on the information in the second file:

```
aws iam put-role-policy --role-name CodeDeployDemo --policy-name
CodeDeployDemo-Permissions --policy-document file://CodeDeployDemo-Permis
sions.json
```

You've now created a service role that allows AWS CodeDeploy to access your instances to expand (read) their tags or to identify your Amazon EC2 instances by the Auto Scaling group names that they're associated with.

For more general information about creating service roles, see Creating a Role for an AWS Service in the *IAM User Guide*.

## Use the IAM Console to Get the Service Role ARN

To use the IAM console to get the ARN of the service role that was created through Use the IAM Console to Create a Service Role that is Compatible with AWS CodeDeploy (p. 140):

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, click **Roles**.
3. In the **Search** box, type **`CodeDeployDemo`** and then press Enter.
4. Click **CodeDeployDemo**.
5. Note the value of the **Role ARN** field.

### Use the AWS CLI to Get the Service Role ARN

To use the AWS CLI to get the ARN of the service role that was created through Use the AWS CLI to Create a Service Role that is Compatible with AWS CodeDeploy (p. 142), call the **get-role** command against the service role named **`CodeDeployDemo`**:

```
aws iam get-role --role-name CodeDeployDemo --query "Role.Arn" --output text
```

The value that is returned is the service role ARN.

# Change Deployment Group Settings with AWS CodeDeploy

You can use the AWS CodeDeploy console, the AWS CLI, or the AWS CodeDeploy APIs to change the settings of an existing deployment group.

**Caution**
You *cannot* start following these steps if:

- You want the existing deployment group to use a different deployment configuration, and it is a custom deployment configuration, but you have not yet created the new custom deployment configuration. Instead, follow the instructions in Create a Deployment Configuration (p. 148), and then follow the steps below.
- You want the existing deployment group to use a different service role, but you have not yet created the new service role. Note that the new service role must trust AWS CodeDeploy with at minimum the permissions that are described in Create a Service Role (p. 140). To create and configure a new service role with the correct permissions, follow the instructions in Create a Service Role (p. 140), and then follow the steps below.

**Topics**



## Change Deployment Group Settings by Using the AWS CodeDeploy Console

To change deployment group settings by using the AWS CodeDeploy console:

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

**Note**
Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. On the AWS CodeDeploy service navigation bar, click **Applications**.
3. In the list of applications, click the application that matches the deployment group that you want to change.

   **Tip**
   If no entries are displaying but you expect to see some, make sure that the correct corresponding region is selected. On the navigation bar, in the region selector, click one of the supported regions (p. 192). AWS CodeDeploy supports only certain regions.

4. In the application details page, in the **Deployment Groups** area, click the arrow next to the deployment group that you want to change. Then click **Deployment Group Details**.
5. In the deployment group details page, click **Edit**.
6. In the **Deployment Group Name** box, type a different name, if you want to change it to something else. Otherwise, leave the default, which is the deployment group's current name.

   **Note**
   Deployment group names must be unique per application.

7. In the list of tags, in the **Key** and **Value** boxes, change or add tags types and key-value pair values, if you want to change them to something else. Otherwise, leave the default, which is the current set of key-value pairs. (For information on Amazon EC2 tags, see Working with Tags in the Console.)

   If you add any new key-value pairs, a new row appears for you to add another key-value pair if desired. You can repeat this step for up to 10 key-value pairs.

   **Tip**
   As AWS CodeDeploy finds instances that match each specified key-value pair, it displays the number of matching instances. To see more information about the instances, click the number.

   To remove a key-value pair from the list, click the corresponding remove icon.
8. In the **Deployment Config** list, select a different deployment configuration, if you want to change it to something else. Otherwise, leave the default, which is the deployment group's current deployment configuration.
9. In the **Service Role ARN** box, select a different existing Amazon Resource Name (ARN) corresponding to a service role that trusts AWS CodeDeploy with at minimum the trust and permissions that are described in Create a Service Role (p. 140), if you want to change it to something else. Otherwise, leave the default, which is the deployment group's current service role ARN. (To get the service role ARN, see Use the IAM Console to Get the Service Role ARN (p. 143).)
10. If you want to deploy the last successful revision to the deployment group based on your changes, check the box labelled **Deploy changes made to *`deployment group name`***, and then click **Save & Deploy Now**. When prompted, click **Deploy Now**. AWS CodeDeploy updates the deployment group's information, starts a deployment of the last successful revision to the deployment group based on the changes that you specified, and displays the **Deployments** page.

    **Note**
    This check box will only appear if—and checking this box will only work if—there was a last successful deployment to this deployment group.

11. If you only want to update the deployment group's information with your changes, but you do not want to deploy any applications to the deployment group at this time, leave the box unchecked that is labelled **Deploy changes made to *`deployment group name`***, and then click **Save**. AWS CodeDeploy updates the deployment group's information, but it does not deploy any applications to the deployment group based on your changes.

## Change Deployment Group Settings by Using the AWS CLI

To change deployment group settings by using the AWS CLI, call the update-deployment-group command, specifying:

- The existing application name. To view a list of existing application names, call the list-applications command.
- The current deployment group name. To view a list of existing deployment group names, call the list-deployment-groups command.
- Optionally, a different deployment group name, if you want to change the current deployment group's name.
- Optionally, replacement tags that uniquely identify the instances to be included in the deployment group, if you want to change the current tags.
- Optionally, a different Amazon Resource Name (ARN) corresponding to a service role that allows AWS CodeDeploy to act on your AWS account's behalf when interacting with other related AWS services, if you want to change the current service role ARN to another one that has the correct permissions. To get the service role ARN, see Use the AWS CLI to Get the Service Role ARN (p. 144). For more information on service roles, see Cross-Account API Access Using IAM Roles.
- Optionally, the names of replacement Auto Scaling groups to be added to the deployment group, if you want to change the current Auto Scaling group names.
- Optionally, the name of a different existing deployment configuration, if you want to change the current deployment configuration. To view a list of existing deployment configurations, see View Deployment Configuration Details (p. 133). (If not specified, AWS CodeDeploy uses a specific default deployment configuration.)

# Register an Application Revision in Amazon S3 with AWS CodeDeploy

If you've already called the push command to push an application revision to Amazon S3 to be deployed by AWS CodeDeploy, you don't also need to register the revision. However, if you upload a revision to Amazon S3 through other means, and you want the revision to appear in the **Revisions** section of the application details page of the AWS CodeDeploy console or through the AWS CLI, you must follow the steps on this page to register the revision first.

If you've pushed an application revision to a GitHub repository and you want the revision to appear in the **Revisions** section of the application details page of the AWS CodeDeploy console or through the AWS CLI, you must also follow the steps on this page to register the revision first.

You can use only the AWS CLI or the AWS CodeDeploy APIs to register application revisions in Amazon S3 or GitHub with AWS CodeDeploy.

**Topics**

# To register a revision in Amazon S3 with AWS CodeDeploy by using the AWS CLI

1. Upload the revision to Amazon S3.
2. Call the register-application-revision command, specifying:
   - The existing application name. To view a list of existing application names, call the list-applications command.
   - Information about the revision to be registered:
     - The Amazon S3 bucket name containing the revision.
     - The name and file type of the uploaded revision.

       **Note**
       The tar and compressed tar archive file formats (.tar and .tar.gz) are not supported for Windows Server instances.
     - Optionally, the revision's Amazon S3 version identifier. (If the version identifier is not specified, AWS CodeDeploy will use the most recent version.)
     - Optionally, the revision's ETag. (If the ETag is not specified, AWS CodeDeploy will skip object validation.)
   - Optionally, any description that you may want to associate with the revision.

**Tip**
Information about a revision in Amazon S3 can be specified directly on the command line, using this syntax as part of the **register-application-revision** call (`version` and `eTag` are optional):

```
--s3-location bucket=string,key=string,bundleType=tar|tgz|zip,version=string,eTag=string
```

# To register a revision in GitHub with AWS CodeDeploy by using the AWS CLI

1. Upload the revision to your GitHub repository.
2. Call the register-application-revision command, specifying:
   - The existing application name. To view a list of existing application names, call the list-applications command.
   - Information about the revision to be registered:
     - The GitHub user or group name assigned to the repository that contains the revision, followed by a forward slash character (`/`), followed by the repository name.
     - The ID of the commit that references the revision in the repository.
   - Optionally, any description that you may want to associate with the revision.

**Tip**
Information about a revision in GitHub can be specified directly on the command line, using this syntax as part of the **register-application-revision** call:

```
--github-location repository=string,commitId=string
```

# Create a Deployment Configuration with AWS CodeDeploy

When you create a deployment, you can specify an existing deployment configuration. (If you don't specify a deployment configuration, AWS CodeDeploy uses a specific default deployment configuration.) If none of the existing deployment configurations meet your requirements, you can create your own deployment configuration.

You can use only the AWS CLI or the AWS CodeDeploy APIs to create deployment configurations.

> **Note**
> To view a list of existing deployment configurations that are already registered to your AWS account and that you can use instead of creating a new one here, see View Deployment Configuration Details (p. 133).

To create a new deployment configuration by using the AWS CLI, call the create-deployment-config command, specifying:

- A name that uniquely represents the deployment configuration. This name must be unique across all of the deployment configurations that you create with AWS CodeDeploy associated with your AWS account.
- The minimum number or percentage of healthy instances that should be available at any time during the deployment. For more information, see Instance Health (p. 65).

# Stop a Deployment with AWS CodeDeploy

You can use the AWS CodeDeploy console, the AWS CLI, or the AWS CodeDeploy APIs to stop deployments that are associated with your AWS account.

> **Caution**
> Stopping a deployment may leave some or all of the instances in your deployment groups in an indeterminate deployment state. For more information, see Stopped and Failed Deployments (p. 62).

**Topics**



## Stop a Deployment by Using the AWS CodeDeploy Console

To stop a deployment by using the AWS CodeDeploy console:

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

**Note**
Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. In the AWS CodeDeploy service navigation bar, click **Deployments**. The **Deployments** section shows a list of deployments and their details.

   **Tip**
   If no entries are displaying but you expect to see some, make sure that the correct corresponding region is selected. On the navigation bar, in the region selector, click one of the supported regions (p. 192). AWS CodeDeploy supports only certain regions.

3. In the **Actions** column for the deployment that you want to stop, click **Stop**.

   **Note**
   If the **Actions** column does not show a **Stop** button, then you cannot stop the deployment. This is because the deployment has gone past the point where it can be stopped.

## Stop a Deployment by Using the AWS CLI

To stop a deployment by using the AWS CLI, call the stop-deployment command, specifying the unique deployment ID. To view a list of existing deployment IDs, call the list-deployments command.

# Delete a Deployment Group with AWS CodeDeploy

You can use the AWS CodeDeploy console, the AWS CLI, or the AWS CodeDeploy APIs to delete deployment groups that are associated with your AWS account.

**Caution**
If you delete a deployment group, then all existing deployment details that are associated with that deployment group will also be deleted from the AWS CodeDeploy system. The instances that were participating in the deployment group will remain unchanged. **This action cannot be undone.**

**Topics**



## Delete a Deployment Group by Using the AWS CodeDeploy Console

To delete a deployment group by using the AWS CodeDeploy console:

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. In the AWS CodeDeploy service navigation bar, click **Applications**.

3. In the list of applications, click the name of the application that is associated with the deployment group that you want to delete.
4. On the application details page, in the **Deployment Groups** area, click the arrow next to the name of the deployment group that you want to delete. Then click **Deployment Group Details**.
5. On the deployment group details page, click **Delete Deployment Group**. When prompted, click **Delete Deployment Group**. The deployment group is deleted, and the application details page is displayed.

## Delete a Deployment Group by Using the AWS CLI

To delete a deployment group by using the AWS CLI, call the delete-deployment-group command, specifying:

- The name of the existing application that is associated with the deployment group. To view a list of application names, call the list-applications command.
- The name of the existing deployment group that is associated with the application. To view a list of deployment group names, call the list-deployment-groups command.

# Delete a Deployment Configuration with AWS CodeDeploy

You can use only the AWS CLI or the AWS CodeDeploy APIs to delete custom deployment configurations that are associated with your AWS account.

> **Note**
> You cannot delete deployment configurations that are built in to AWS CodeDeploy, such as `CodeDeployDefault.AllAtOnce`, `CodeDeployDefault.HalfAtATime`, and `CodeDeployDefault.AllAtOnce`.

> **Caution**
> You cannot delete a custom deployment configuration that is still in use. If you delete an unused, custom deployment configuration, you will no longer be able to associate the deleted deployment configuration with new deployments and new deployment groups. **This action cannot be undone.**

To delete a deployment configuration by using the AWS CLI, call the delete-deployment-config command, specifying the existing deployment configuration name. To view a list of existing deployment configuration names, call the list-deployment-configs command.

# Delete an Application with AWS CodeDeploy

You can use the AWS CodeDeploy console, the AWS CLI, or the AWS CodeDeploy APIs to delete applications that are associated with your AWS account.

> **Caution**
> Deleting an application removes information about the application from the AWS CodeDeploy system, including all related deployment group information and all related deployment details. It does not uninstall any related application revisions from any instances that the revisions may be installed on, nor does it delete any related revisions from any Amazon S3 buckets where the revisions may be stored. It also does not terminate any associated Amazon EC2 instances or deregister any associated on-premises instances. **This action cannot be undone.**

**Topics**

## Delete an Application by Using the AWS CodeDeploy Console

To delete an application by using the AWS CodeDeploy console:

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).
2. If the **Applications** page is not visible, in the AWS CodeDeploy service navigation bar, click **Applications**. The **Applications** page is displayed.
3. In the list of applications, click the name of the application that you want to delete.
4. Click **Delete Application**. When prompted to confirm the deletion, click **Delete**. AWS CodeDeploy deletes the application and returns you to the **Applications** page.

## Delete an Application by Using the AWS CLI

To delete an application by using the AWS CLI, call the delete-application command, specifying the existing application name. To view a list of existing application names, call the list-applications command.

# Change Application Settings with AWS CodeDeploy

You can use only the AWS CLI or the AWS CodeDeploy APIs to change the settings of an existing application.

To change application settings by using the AWS CLI, call the update-application command, specifying:

- The existing application name. To view a list of existing application names, call the list-applications command.
- The new application name.

# Redeploy and Roll Back Deployments with AWS CodeDeploy

You can use AWS CodeDeploy to redeploy a revision that you previously deployed. You may decide to do this if, for example, a user has unknowingly gotten an application into an unknown state on an instance. Rather than spending a lot of time troubleshooting what happened, you can simply redeploy the application that you originally deployed to get back to a known working state.

AWS CodeDeploy does not support the concept of an automatic rollback of a deployment. However, you can simulate a rollback by redeploying a previous known good revision of the application. You may decide

to do this if, for example, you just deployed an application that turns out to be broken in the same way across a collection of instances. Rather than spending a lot of time making fixes to the broken application on each instance, you can simply redeploy a previous known good revision of the application that you know worked in the past.

To redeploy or simulate a deployment rollback by using AWS CodeDeploy, see Deploy a Revision (p. 121).

For more information about how AWS CodeDeploy handles redeployments and simulated rollbacks, see Redeployments and Deployment Rollbacks (p. 63).

# Service and Product Integrations with AWS CodeDeploy

AWS CodeDeploy integrates with various AWS services and with products such as the following:

## AWS Services

### Auto Scaling

| Resource | Description |
| --- | --- |
| Auto Scaling Integration (p. 157) | Describes how AWS CodeDeploy integrates with Auto Scaling, an AWS web service that can automatically launch Amazon EC2 instances based on criteria that you specify. |
| Tutorial: Deploy to an Auto Scaling Group (p. 158) | Practice using AWS CodeDeploy and Auto Scaling to automatically launch Amazon EC2 instances and then deploy application revisions to the newly-launched instances. |

### AWS CloudTrail

| Resource | Description |
| --- | --- |
| Logging AWS CodeDeploy API Calls By Using AWS CloudTrail (p. 174) | Describes how AWS CodeDeploy integrates with AWS CloudTrail, an AWS service that captures API calls made by or on behalf of AWS CodeDeploy in your AWS account and delivers the log files to an Amazon S3 bucket that you specify. |

## Elastic Load Balancing

| Resource | Description |
| --- | --- |
| Elastic Load Balancing Integration (p. 176) | Describes how AWS CodeDeploy integrates with Elastic Load Balancing, which enables you to automatically distribute incoming application traffic across multiple Amazon EC2 instances. |
| ELB and ASG lifecycle event scripts | Provides a set of deployment lifecycle event scripts that, during a deployment, deregisters the target Amazon EC2 instance with the target load balancers, waits for the connection to drain, and then re-registers the Amazon EC2 instance with the load balancers after the deployment is done. |

# Products

## Ansible

| Resource | Description |
| --- | --- |
| Ansible and AWS CodeDeploy | If you already have a set of Ansible playbooks, but just need somewhere to run them, the template for Ansible and AWS CodeDeploy demonstrates how a couple of simple deployment hooks will ensure Ansible is available on the local deployment instance and run the given playbooks. Alternatively, if you already have a process for building and maintaining your inventory, there's also an Ansible module that you can use to install and run the AWS CodeDeploy Agent. |

## Chef

| Resource | Description |
| --- | --- |
| Chef and AWS CodeDeploy | AWS provides two template samples for integrating Chef and AWS CodeDeploy. The first is a Chef cookbook that will install and start the AWS CodeDeploy Agent, giving you the ability to continue managing your host infrastructure with Chef while also being able to take advantage of the power of AWS CodeDeploy. The second sample template demonstrates how to use AWS CodeDeploy to orchestrate running cookbooks and recipes with chef-solo on each node. |

# Circle CI

| Resource | Description |
| --- | --- |
| Continuous Deployment with AWS CodeDeploy | Describes how to use CircleCI to automatically create new application revisions, upload them to Amazon S3, and both trigger and watch deployments when you get a green build. |

# CloudBees

| Resource | Description |
| --- | --- |
| AWS CodeDeploy Jenkins Plugin Now Available Including on DEV@cloud | Announces the general availability and location of an AWS CodeDeploy Jenkins plugin, available on CloudBees DEV@cloud. |

# CodeShip

| Resource | Description |
| --- | --- |
| Deploy to AWS CodeDeploy | Provides a high-level description of how to use Codeship to deploy application revisions through AWS CodeDeploy. |
| The AWS CodeDeploy Integration on Codeship | Describes in more detail how to use the Codeship UI to add AWS CodeDeploy to an existing deployment pipeline for a branch. |

# GitHub

| Resource | Description |
| --- | --- |
| GitHub Integration (p. 177) | Learn how to use AWS CodeDeploy to deploy application revisions from GitHub repositories. |
| Tutorial: Deploy from GitHub (p. 180) | Practice using AWS CodeDeploy to deploy a sample application revision from a GitHub repository to a single Amazon Linux or Windows Server Amazon EC2 instance. |
| Automatically Deploy from GitHub Using AWS CodeDeploy | Describes how to automatically trigger a deployment from a GitHub repository whenever the source code in that repository changes. |

# Jenkins

| Resource | Description |
| --- | --- |
| AWS CodeDeploy Jenkins Plugin | The AWS CodeDeploy Jenkins plugin provides a post-build step for your Jenkins project. Upon a successful build, it will zip the workspace, upload to Amazon S3, and start a new deployment. Optionally, you can set it to wait for the deployment to finish, making the final success contingent on the success of the deployment. |

# Puppet Labs

| Resource | Description |
| --- | --- |
| Puppet and AWS CodeDeploy | AWS provides a couple of sample templates to get you working with Puppet and AWS CodeDeploy faster. The first is a Puppet module that will install and start the AWS CodeDeploy Agent, giving you the ability to continue managing your host infrastructure with Puppet while also being able to take advantage of the power of AWS CodeDeploy. The second sample template demonstrates how to use AWS CodeDeploy to orchestrate running modules and manifests with masterless puppet on each node. |

# SaltStack

| Resource | Description |
| --- | --- |
| SaltStack and AWS CodeDeploy | The two sample templates that AWS provides offer examples for how you can integrate existing Salt infrastructure with AWS CodeDeploy. On the one hand, you can use the AWS CodeDeploy module to install and run the AWS CodeDeploy Agent on your minions. On the other hand, you can use AWS CodeDeploy with a couple of simple deployment hooks to orchestrate running your Salt States. Either way, you get to take advantage of the power of both Salt and AWS CodeDeploy. |

# Solano Labs

| Resource | Description |
| --- | --- |
| AWS CodeDeploy | Describes how to set up automatic AWS CodeDeploy deployment from your Solano CI build. |

## Travis CI

| Resource | Description |
| --- | --- |
| AWS CodeDeploy | Describes how Travis CI can automatically trigger a new deployment on AWS CodeDeploy after a successful build. |

# AWS CodeDeploy Integration with Auto Scaling

AWS CodeDeploy supports Auto Scaling. Auto Scaling is an AWS web service that can automatically launch Amazon EC2 instances based on criteria that you specify. These criteria can include limits that are exceeded for specified CPU utilization, disk reads or writes, or inbound or outbound network traffic, over a specified time interval. This enables you to scale up a group of Amazon EC2 instances whenever you need them and then use AWS CodeDeploy to deploy application revisions to the additional Amazon EC2 instances automatically. When the Amazon EC2 instances are no longer needed, Auto Scaling automatically terminates those Amazon EC2 instances. For more information about Auto Scaling, see What is Auto Scaling.

With Auto Scaling integration, AWS CodeDeploy can deploy revisions to an Auto Scaling group. Whenever new Amazon EC2 instances are launched as part of an Auto Scaling group, AWS CodeDeploy can automatically deploy your revisions to the new Amazon EC2 instances.

> **Tip**
> You can also coordinate deployments in AWS CodeDeploy with Amazon EC2 instances that are registered with Elastic Load Balancing load balancers. For more information, see Elastic Load Balancing Integration (p. 176).

## Deploying AWS CodeDeploy Applications to Auto Scaling Groups

To deploy an AWS CodeDeploy application revision to an Amazon EC2 Auto Scaling group, you need to do just a few things:

1. Create or locate an IAM instance profile that allows the Auto Scaling group to work with Amazon S3.

   > **Note**
   > You can also use AWS CodeDeploy to deploy revisions from GitHub repositories to Auto Scaling groups. While participating Amazon EC2 instances still require an IAM instance profile, the profile doesn't need any specific permissions. For more information, see Create an IAM Instance Profile (p. 84).

2. Create an Auto Scaling group, or use an existing one, specifying the IAM instance profile.
3. Create or locate a service role that allows AWS CodeDeploy to create a deployment group that contains the Auto Scaling group.
4. Create a deployment group with AWS CodeDeploy, specifying the Auto Scaling group name and the service role.
5. Use AWS CodeDeploy to deploy your revision to the deployment group that contains the Auto Scaling group.

To experiment with completing these activities, see Tutorial: Deploy to an Auto Scaling Group (p. 158).

# Auto Scaling Behaviors with AWS CodeDeploy

## The execution order of custom lifecycle hook events for Auto Scaling groups cannot be predetermined

You can add your own lifecycle hooks to Auto Scaling groups that AWS CodeDeploy deploys to. However, the order that those custom lifecycle hook events execute cannot be predetermined in relation to AWS CodeDeploy default deployment lifecycle events. For example, if you add a custom lifecycle hook named `ReadyForSoftwareInstall` to an Auto Scaling group, you cannot know beforehand whether your `ReadyForSoftwareInstall` custom lifecycle hook will execute before the first, or after the last, AWS CodeDeploy default deployment lifecycle event after an Amazon EC2 instance is added to the Auto Scaling group.

To learn how to add custom lifecycle hooks to an Auto Scaling group, see Adding lifecycle hooks.

## Deleting Auto Scaling groups cause deployment failures for associated deployment groups

If you add an Auto Scaling group to a deployment group, and then you delete the Auto Scaling group, all future deployments to that deployment group will fail.

# Using a Custom AMI with AWS CodeDeploy and Auto Scaling

When you create an Auto Scaling configuration for use with AWS CodeDeploy, you have 2 options when specifying which base AMI to use when new Amazon EC2 instances are launched into an Auto Scaling group:

- You can specify a base custom AMI that already has the AWS CodeDeploy Agent installed by you or through other means. Because the AWS CodeDeploy Agent is preinstalled, this option launches new Amazon EC2 instances slightly faster than the other option. However, this option provides a small probability of initial deployments failing for new Amazon EC2 instances, especially if the AWS CodeDeploy Agent is out of date. For this option, we recommend that you regularly update the AWS CodeDeploy Agent in your base custom AMI to reduce the probability of initial deployments failing for new Amazon EC2 instances.
- You can specify a base AMI that doesn't have the AWS CodeDeploy Agent installed, and specify the AWS CodeDeploy Agent to be installed as each new Amazon EC2 instance is launched into an Auto Scaling group. Because the AWS CodeDeploy Agent is not preinstalled, this option launches new Amazon EC2 instances slightly slower than the other option. However, this option provides a greater probability of initial deployments succeeding for new Amazon EC2 instances, as this option uses the most recent version of the AWS CodeDeploy Agent.

# Tutorial: Using AWS CodeDeploy to Deploy an Application to an Auto Scaling Group

In this tutorial, you'll practice using AWS CodeDeploy to deploy an application revision to an Auto Scaling group. For general information about Auto Scaling integration with AWS CodeDeploy, see Auto Scaling Integration (p. 157).

**Topics**

# Prerequisites

This tutorial assumes that you have already completed all of the steps in Setting Up (p. 4), including setting up and configuring the AWS CLI, as well as creating an IAM instance profile and a service role. This tutorial assumes that the name of your IAM instance profile is **`CodeDeployDemo-EC2-Instance-Profile`**, and the name of your service role is **`CodeDeployDemo`**.

If you want to deploy an application revision to an Auto Scaling group that is comprised of Ubuntu Server Amazon EC2 instances, you can create and use the sample revision that is described in Step 2 (p. 51) of the On-Premises Instance Deployment (Windows Server or Ubuntu Server) (p. 50) tutorial, or you will need to create and use your own revision that is compatible with both an Ubuntu Server instance and AWS CodeDeploy. (We also provide sample revisions for Amazon Linux and Windows Server Amazon EC2 instances.) To create a revision on your own, see Prepare a Revision (p. 114).

# Step 1: Create and Configure the Auto Scaling Group

In this step, you'll create an Auto Scaling group for AWS CodeDeploy to deploy your revision to. This Auto Scaling group will contain a single Amazon Linux or Windows Server Amazon EC2 instance to start with. (In a later step, you will instruct Auto Scaling to add one more Amazon EC2 instance; AWS CodeDeploy will then automatically deploy your revision to the new Amazon EC2 instance.)

**Topics**



## To use the AWS CLI to create and configure the Auto Scaling group

1. Call the **create-launch-configuration** command to create an Auto Scaling launch configuration.

   Before you call this command, you'll need the ID of an AMI that works for this tutorial, represented by the placeholder *`imageID`* below. You'll also need the name of an existing Amazon EC2 instance key pair to enable access to the Amazon EC2 instance, represented by the placeholder *`keyName`* below. Finally, you will need instructions to install the latest version of the AWS CodeDeploy Agent.

   To get the ID of an AMI that works with this tutorial, do the following:

   1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
   2. In the navigation pane, under **Instances**, click **Instances**. Then click **Launch Instance**.
   3. On the **Step 1: Choose an Amazon Machine Image** page, with the **Quick Start** tab already selected, note the ID of the AMI next to **Amazon Linux AMI**, **Ubuntu Server 14.04 LTS**, or

**Microsoft Windows Server 2012 R2**. Then click **Cancel and Exit**, as you will no longer need this wizard to use the AWS CLI to create and configure the Auto Scaling group.

> **Note**
> If you have a custom version of an AMI that is fully compatible with AWS CodeDeploy, select your AMI here instead of browsing through the **Quick Start** tab, and note the ID of the custom AMI. For considerations regarding using a custom AMI with AWS CodeDeploy and Auto Scaling, see Using a Custom AMI with AWS CodeDeploy and Auto Scaling (p. 158).

For the Amazon EC2 instance key pair, use the name of your Amazon EC2 instance key pair.

For the instructions to install the latest version of the AWS CodeDeploy Agent, on your development machine, create a file named `instance-setup.sh` (for an Amazon Linux or Ubuntu Server Amazon EC2 instance) or `instance-setup.txt` (for a Windows Server Amazon EC2 instance) with the following contents:

> **Note**
> If you have a custom version of an AMI that is fully compatible with AWS CodeDeploy, you don't need to create the `instance-setup.sh` or `instance-setup.txt` file. For considerations regarding using a custom AMI with AWS CodeDeploy and Auto Scaling, see Using a Custom AMI with AWS CodeDeploy and Auto Scaling (p. 158).

For Amazon Linux Amazon EC2 instances:

```
#!/bin/bash
yum -y update
yum install -y aws-cli
cd /home/ec2-user
aws s3 cp s3://bucket-name/latest/install . --region region-name
chmod +x ./install
./install auto
```

In the previous list of commands, *`bucket-name`* represents one of the following:

- `aws-codedeploy-us-east-1` for instances in the US East (N. Virginia) region
- `aws-codedeploy-us-west-2` for instances in the US West (Oregon) region
- `aws-codedeploy-eu-west-1` for instances in the EU (Ireland) region
- `aws-codedeploy-ap-southeast-2` for instances in the Asia Pacific (Sydney) region

In the previous list of commands, *`region-name`* represents one of the following:

- `us-east-1` for instances in the US East (N. Virginia) region
- `us-west-2` for instances in the US West (Oregon) region
- `eu-west-1` for instances in the EU (Ireland) region
- `ap-southeast-2` for instances in the Asia Pacific (Sydney) region

For Ubuntu Server Amazon EC2 instances:

```
#!/bin/bash
apt-get -y update
apt-get -y install awscli
apt-get -y install ruby2.0
cd /home/ubuntu
```

```
aws s3 cp s3://bucket-name/latest/install . --region region-name
chmod +x ./install
./install auto
```

In the previous list of commands, *bucket-name* represents one of the following:

- aws-codedeploy-us-east-1 for instances in the US East (N. Virginia) region
- aws-codedeploy-us-west-2 for instances in the US West (Oregon) region
- aws-codedeploy-eu-west-1 for instances in the EU (Ireland) region
- aws-codedeploy-ap-southeast-2 for instances in the Asia Pacific (Sydney) region

In the previous list of commands, *region-name* represents one of the following:

- us-east-1 for instances in the US East (N. Virginia) region
- us-west-2 for instances in the US West (Oregon) region
- eu-west-1 for instances in the EU (Ireland) region
- ap-southeast-2 for instances in the Asia Pacific (Sydney) region

For Windows Server Amazon EC2 instances:

```
<powershell>
New-Item -Path c:\temp -ItemType "directory" -Force
Read-S3Object -BucketName bucket-name/latest -Key codedeploy-agent.msi -File
 c:\temp\codedeploy-agent.msi
Start-Process -Wait -FilePath c:\temp\codedeploy-agent.msi -WindowStyle
Hidden
</powershell>
```

In the previous list of commands, *bucket-name* represents one of the following:

- aws-codedeploy-us-east-1 for instances in the US East (N. Virginia) region
- aws-codedeploy-us-west-2 for instances in the US West (Oregon) region
- aws-codedeploy-eu-west-1 for instances in the EU (Ireland) region
- aws-codedeploy-ap-southeast-2 for instances in the Asia Pacific (Sydney) region

Now call the **create-launch-configuration** command.

For Linux, OS X, or Unix:

```
aws autoscaling create-launch-configuration \
  --launch-configuration-name CodeDeployDemo-AS-Configuration \
  --image-id imageID \
  --key-name keyName \
  --iam-instance-profile CodeDeployDemo-EC2-Instance-Profile \
  --instance-type t1.micro \
  --user-data file://path/to/instance-setup.sh
```

For Windows:

```
aws autoscaling create-launch-configuration --launch-configuration-name
CodeDeployDemo-AS-Configuration --image-id imageID --key-name keyName --iam-
```

```
instance-profile CodeDeployDemo-EC2-Instance-Profile --instance-type t1.micro
 --user-data file://path/to/instance-setup.sh
```

**Note**
If you have a custom version of an AMI that is fully compatible with AWS CodeDeploy, omit the **--user-data** option in the preceding command. For considerations regarding using a custom AMI with AWS CodeDeploy and Auto Scaling, see Using a Custom AMI with AWS CodeDeploy and Auto Scaling (p. 158).

These commands create an Auto Scaling launch configuration named **CodeDeployDemo-AS-Configuration**, based on the specified image ID, applying the specified IAM instance profile and Amazon EC2 instance key pair, and running the specified instructions to install the latest version of the AWS CodeDeploy Agent. This launch configuration is based on the t1.micro Amazon EC2 instance type.

2. Create an Auto Scaling group with the **create-auto-scaling-group** command. You will need the name of one of the availability zones in one of the supported regions (p. 192), represented by the placeholder *availabilityZone* below.

   **Tip**
   To view a list of available availability zones within a region, call:

   ```
   aws ec2 describe-availability-zones --region regionName
   ```

   For example, to view a list of available availability zones within the US West (Oregon) region, call:

   ```
   aws ec2 describe-availability-zones --region us-west-2
   ```

   For Linux, OS X, or Unix:

   ```
   aws autoscaling create-auto-scaling-group \
     --auto-scaling-group-name CodeDeployDemo-AS-Group \
     --launch-configuration-name CodeDeployDemo-AS-Configuration \
     --min-size 1 \
     --max-size 1 \
     --desired-capacity 1 \
     --availability-zones availabilityZone
   ```

   For Windows:

   ```
   aws autoscaling create-auto-scaling-group --auto-scaling-group-name
   CodeDeployDemo-AS-Group --launch-configuration-name CodeDeployDemo-AS-Con
   figuration --min-size 1 --max-size 1 --desired-capacity 1 --availability-
   zones availabilityZone
   ```

   These commands create an Auto Scaling group named **CodeDeployDemo-AS-Group** based on the Auto Scaling launch configuration named **CodeDeployDemo-AS-Configuration**. This Auto Scaling group has only 1 Amazon EC2 instance, and it is created in the specified availability zone.

3. Make sure that your Auto Scaling group is ready by calling the **describe-auto-scaling-groups** command against **CodeDeployDemo-AS-Group**:

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].[Health
Status, LifecycleState]" --output text
```

Do not proceed until the returned values show `Healthy` and `InService`.

## To use the Amazon EC2 console to create and configure the Auto Scaling group

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the global navigation bar, make sure one of supported regions (p. 192) is selected. This is because Auto Scaling resources are tied to the region that you specify, and AWS CodeDeploy is currently supported only in certain regions.
3. In the navigation bar, under **Auto Scaling**, click **Launch Configurations**.
4. Click **Create launch configuration**.
5. On the **1. Choose AMI** page, with the **Quick Start** tab already selected, next to **Amazon Linux AMI**, **Ubuntu Server 14.04 LTS**, or **Microsoft Windows Server 2012 R2**, click **Select**.

   **Note**
   If you have a custom version of an AMI that you want to use that already has the AWS CodeDeploy Agent installed, select your AMI here instead. For considerations regarding using a custom AMI with AWS CodeDeploy and Auto Scaling, see Using a Custom AMI with AWS CodeDeploy and Auto Scaling (p. 158).
6. On the **2. Choose Instance Type** page, leave the defaults by clicking **Next: Configure details**.
7. On the **3. Configure details** page, in the **Name** box, type **`CodeDeployDemo-AS-Configuration`**. In the **IAM role** box, select the IAM instance profile that you created earlier: **`CodeDeployDemo-EC2-Instance-Profile`**.

   Expand **Advanced Details**, and in the **User data** box, with **As text** already selected, type the following:

   **Note**
   If you have a custom version of an Amazon Linux AMI that you want to use that already has the AWS CodeDeploy Agent installed, don't type anything into the **User data** box, and continue on. For considerations regarding using a custom AMI with AWS CodeDeploy and Auto Scaling, see Using a Custom AMI with AWS CodeDeploy and Auto Scaling (p. 158).

   For Amazon Linux Amazon EC2 instances:

```
#!/bin/bash
yum -y update
yum install -y aws-cli
cd /home/ec2-user
aws s3 cp s3://bucket-name/latest/install . --region region-name
chmod +x ./install
./install auto
```

In the previous list of commands, *`bucket-name`* represents one of the following:

- `aws-codedeploy-us-east-1` for instances in the US East (N. Virginia) region
- `aws-codedeploy-us-west-2` for instances in the US West (Oregon) region
- `aws-codedeploy-eu-west-1` for instances in the EU (Ireland) region
- `aws-codedeploy-ap-southeast-2` for instances in the Asia Pacific (Sydney) region

In the previous list of commands, *region-name* represents one of the following:

- us-east-1 for instances in the US East (N. Virginia) region
- us-west-2 for instances in the US West (Oregon) region
- eu-west-1 for instances in the EU (Ireland) region
- ap-southeast-2 for instances in the Asia Pacific (Sydney) region

For Ubuntu Server Amazon EC2 instances:

```
#!/bin/bash
apt-get -y update
apt-get -y install awscli
apt-get -y install ruby2.0
cd /home/ubuntu
aws s3 cp s3://bucket-name/latest/install . --region region-name
chmod +x ./install
./install auto
```

In the previous list of commands, *bucket-name* represents one of the following:

- aws-codedeploy-us-east-1 for instances in the US East (N. Virginia) region
- aws-codedeploy-us-west-2 for instances in the US West (Oregon) region
- aws-codedeploy-eu-west-1 for instances in the EU (Ireland) region
- aws-codedeploy-ap-southeast-2 for instances in the Asia Pacific (Sydney) region

In the previous list of commands, *region-name* represents one of the following:

- us-east-1 for instances in the US East (N. Virginia) region
- us-west-2 for instances in the US West (Oregon) region
- eu-west-1 for instances in the EU (Ireland) region
- ap-southeast-2 for instances in the Asia Pacific (Sydney) region

For Windows Server Amazon EC2 instances:

```
<powershell>
New-Item -Path c:\temp -ItemType "directory" -Force
Read-S3Object -BucketName bucket-name/latest -Key codedeploy-agent.msi -File
 c:\temp\codedeploy-agent.msi
Start-Process -Wait -FilePath c:\temp\codedeploy-agent.msi -WindowStyle
Hidden
</powershell>
```

In the previous list of commands, *bucket-name* represents one of the following:

- aws-codedeploy-us-east-1 for instances in the US East (N. Virginia) region
- aws-codedeploy-us-west-2 for instances in the US West (Oregon) region
- aws-codedeploy-eu-west-1 for instances in the EU (Ireland) region
- aws-codedeploy-ap-southeast-2 for instances in the Asia Pacific (Sydney) region

Leave the rest of the defaults by clicking **Skip to review**.

8. On the **6. Review** page, click **Create launch configuration**.

**Note**
In a production environment, we recommend restricting access to Amazon EC2 instances. For more information, see Tips for Securing Your EC2 Instance.

9. In the **Select an existing key pair or create a new key pair** dialog box, select **Choose an existing key pair** if it is not already selected. In the **Select a key pair** drop-down list, select the Amazon EC2 instance key pair that you created or used in previous steps. Then check the box labeled **I acknowledge that I have access to the selected private key file (*key-file-name*.pem), and that without this file, I won't be able to log into my instance**. Finally, click **Create launch configuration**. After the launch configuration is created, a confirmation message appears.
10. Click **Create an Auto Scaling group using this launch configuration**.
11. On the **1. Configure Auto Scaling group details** page, in the **Group name** box, type **`CodeDeployDemo-AS-Group`**. In the **Group size** box, leave the default number **1**. In the **Availability Zone(s)** box, select an availability zone in one of the supported regions (p. 192). Leave the rest of the defaults by clicking **Next: Configure scaling policies**.

    **Note**
    If the **Network** list does not show **Launch into EC2-Classic**, and you are not able to select **Launch into EC2-Classic** or a default Virtual Private Cloud (VPC) in the list, you must select an existing Amazon Virtual Private Cloud (VPC) and subnet, or click **Create new VPC** or **Create new subnet** or both to create a new VPC or subnet or both. For more information, see Your VPC and Subnets.
12. On the **2. Configure scaling policies** page, leave **Keep this group at its initial size** selected by clicking **Next: Configure Notifications**.
13. Skip the step for configuring notifications by clicking **Review**.
14. Click **Create Auto Scaling group**. After the Auto Scaling group is created, a confirmation message appears. Click **Close**.
15. In the navigation bar, with **Auto Scaling Groups** selected under **Auto Scaling**, click the **`CodeDeployDemo-AS-Group`** entry, if it is not already selected. Then click the **Instances** tab. Do not proceed until the Amazon EC2 instance shows the **Lifecycle** column value of **InService** and the **Health Status** column value of **Healthy**.

# Step 2: Deploy the Application to the Auto Scaling Group

In this step, you'll deploy the revision to the single Amazon EC2 instance in the Auto Scaling group.

**Topics**



## To use the AWS CLI to create the deployment

1. The deployment will first need a corresponding application. Call the **create-application** command to create an application named **`SimpleDemoApp`**:

```
aws deploy create-application --application-name SimpleDemoApp
```

2. Next, the deployment will need a corresponding deployment group. To do this, you will need a service role ARN. A *service role* is a special type of IAM role that gives a service permission to act on your

behalf. In this case, the service role will give AWS CodeDeploy permission to access your instances to expand (read) their tags.

You should have already created a service role by following the instructions in Create a Service Role (p. 140). If you didn't follow the instruction in Create a Service Role (p. 140), follow them now. To get the service role ARN, follow the instructions in Use the AWS CLI to Get the Service Role ARN (p. 144), and then return to this page.

3. Now that you have a service role ARN, call the **create-deployment-group** command to create a new deployment group named **`SimpleDemoDG`**, associated with the application named **`SimpleDemoApp`**, using the Auto Scaling group named **`CodeDeployDemo-AS-Group`** and deployment configuration named **`CodeDeployDefault.OneAtATime`**, with the specified service role ARN.

   For Linux, OS X, or Unix:

```
aws deploy create-deployment-group \
  --application-name SimpleDemoApp \
  --auto-scaling-groups CodeDeployDemo-AS-Group \
  --deployment-group-name SimpleDemoDG \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --service-role-arn serviceRoleARN
```

   For Windows:

```
aws deploy create-deployment-group --application-name SimpleDemoApp --auto-
scaling-groups CodeDeployDemo-AS-Group --deployment-group-name SimpleDemoDG
 --deployment-config-name CodeDeployDefault.OneAtATime --service-role-arn
serviceRoleARN
```

4. Call the **create-deployment** command to create a deployment associated with the application named **`SimpleDemoApp`**, the deployment configuration named **`CodeDeployDefault.OneAtATime`**, the deployment group named **`SimpleDemoDG`**, using the revision at the specified location.

   For Amazon Linux Amazon EC2 instances, calling from Linux, OS X, or Unix:

```
aws deploy create-deployment \
  --application-name SimpleDemoApp \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --deployment-group-name SimpleDemoDG \
  --s3-location bucket=bucket,bundleType=zip,key=samples/latest/Sample
App_Linux.zip
```

   In this command, *`bucket`* is one of the following:

   - `aws-codedeploy-us-east-1` (for the US East (N. Virginia) region)
   - `aws-codedeploy-us-west-2` (for the US West (Oregon) region)
   - `aws-codedeploy-eu-west-1` (for the EU (Ireland) region)
   - `aws-codedeploy-ap-southeast-2` (for the Asia Pacific (Sydney) region)

   For Amazon Linux Amazon EC2 instances, calling from Windows:

```
aws deploy create-deployment --application-name SimpleDemoApp --deployment-
config-name CodeDeployDefault.OneAtATime --deployment-group-name SimpleDemoDG
```

```
 --s3-location bucket=bucket,bundleType=zip,key=samples/latest/Sample
App_Linux.zip
```

For Windows Server Amazon EC2 instances, calling from Linux, OS X, or Unix:

```
aws deploy create-deployment \
  --application-name SimpleDemoApp \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --deployment-group-name SimpleDemoDG \
  --s3-location bucket=bucket,bundleType=zip,key=samples/latest/SampleApp_Win
dows.zip
```

In this command, *bucket* is one of the following:

- `aws-codedeploy-us-east-1` (for the US East (N. Virginia) region)
- `aws-codedeploy-us-west-2` (for the US West (Oregon) region)
- `aws-codedeploy-eu-west-1` (for the EU (Ireland) region)
- `aws-codedeploy-ap-southeast-2` (for the Asia Pacific (Sydney) region)

For Windows Server Amazon EC2 instances, calling from Windows:

```
aws deploy create-deployment --application-name SimpleDemoApp --deployment-
config-name CodeDeployDefault.OneAtATime --deployment-group-name SimpleDemoDG
 --s3-location bucket=bucket,bundleType=zip,key=samples/latest/SampleApp_Win
dows.zip
```

**Note**
Currently, we do not provide a sample revision to deploy to Ubuntu Server Amazon EC2 instances. To create a revision on your own, see Prepare a Revision (p. 114).

5. Call the **get-deployment** command to make sure that the deployment succeeded.

   Before you call this command, you will need the ID of the deployment, which should have been returned by the call to the **create-deployment** command. If you need to get the deployment ID again, call the **list-deployments** command against the application named **`SimpleDemoApp`** and the deployment group named **`SimpleDemoDG`**:

```
aws deploy list-deployments --application-name SimpleDemoApp --deployment-
group-name SimpleDemoDG --query "deployments" --output text
```

   Now, call the **get-deployment** command, using the deployment ID represented by *deploymentID* below:

```
aws deploy get-deployment --deployment-id deploymentID --query "deploy
mentInfo.status" --output text
```

   Do not continue until the returned value is `Succeeded`.

## To use the AWS CodeDeploy console to create the deployment

1. Before you start using the AWS CodeDeploy console to deploy your application revision, you will need a service role ARN. A *service role* is a special type of IAM role that gives a service permission to act on your behalf. In this case, the service role will give AWS CodeDeploy permission to access your Amazon EC2 instances to expand (read) their tags.

   You should have already created a service role by following the instructions in Create a Service Role (p. 140). If you didn't follow the instruction in Create a Service Role (p. 140), follow them now. To get the service role ARN, follow the instructions in Use the IAM Console to Get the Service Role ARN (p. 143), and then return to this page.
2. Now that you have the service role ARN, you can start using the AWS CodeDeploy console to deploy your application revision:

   Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).
3. If the **Applications** page is not visible, in the AWS CodeDeploy service navigation drop-down list, select **Applications**.
4. Click **Create New Application**.
5. In the **Application Name** box, type **`SimpleDemoApp`**.
6. In the **Deployment Group Name** box, type **`SimpleDemoDG`**.
7. In the list of tags, select **Auto Scaling Group** in the **Tag Type** drop-down list.
8. In the box next to **Auto Scaling Group**, type **`CodeDeployDemo-AS-Group`**.
9. In the **Deployment Config** drop-down list, select **CodeDeployDefault.OneAtATime**, if it isn't already selected.
10. In the **Service Role ARN** drop-down list, select the service role ARN that you noted earlier.
11. Click **Create Application**. AWS CodeDeploy creates the application and then displays the application details page.
12. In the **Deployment Groups** area, next to **SimpleDemoDG**, click the arrow to see the deployment group details area.
13. Click **Deploy New Revision**.
14. With the option **My application is stored in Amazon S3** already selected next to **Revision Type**, in the **Revision Location** box, type one of the following:

    For Amazon Linux Amazon EC2 instances:

    - For Amazon EC2 instances launched in the US East (N. Virginia) region:
      `http://s3.amazonaws.com/aws-codedeploy-us-east-1/samples/latest/SampleApp_Linux.zip`
    - For Amazon EC2 instances launched in the US West (Oregon) region:
      `http://s3-us-west-2.amazonaws.com/aws-codedeploy-us-west-2/samples/latest/SampleApp_Linux.zip`
    - For Amazon EC2 instances launched in the EU (Ireland) region:
      `http://s3-eu-west-1.amazonaws.com/aws-codedeploy-eu-west-1/samples/latest/SampleApp_Linux.zip`
    - For Amazon EC2 instances launched in the Asia Pacific (Sydney) region:
      `http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Linux.zip`

    For Windows Server Amazon EC2 instances:

- For Amazon EC2 instances launched in the US East (N. Virginia) region:
  `http://s3.amazonaws.com/aws-codedeploy-us-east-1/samples/latest/SampleApp_Windows.zip`
- For Amazon EC2 instances launched in the US West (Oregon) region:
  `http://s3-us-west-2.amazonaws.com/aws-codedeploy-us-west-2/samples/latest/SampleApp_Windows.zip`
- For Amazon EC2 instances launched in the EU (Ireland) region:
  `http://s3-eu-west-1.amazonaws.com/aws-codedeploy-eu-west-1/samples/latest/SampleApp_Windows.zip`
- For Amazon EC2 instances launched in the Asia Pacific (Sydney) region:
  `http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Windows.zip`

For Ubuntu Server Amazon EC2 instances, type the location of your custom application revision that is stored in Amazon S3.

15. Leave the **Deployment Description** box blank.
16. With the **Deployment Config** drop-down list already showing **CodeDeployDefault.OneAtATime**, click **Deploy Now**. The **Deployments** page appears, showing information about your newly created deployment.

    **Tip**
    To update the deployment's current status, use your browser's page refresh command. If the deployment status shows **Failed** instead of **Succeeded**, you may want to try some of the techniques at Monitor (and Troubleshoot) Your Deployment (p. 28) (using the application name of **`SimpleDemoApp`** and the deployment group name of **`SimpleDemoDG`**) to try to determine what went wrong.

# Step 3: Check Your Results

In this step, you'll check to see that AWS CodeDeploy installed the SimpleDemoApp revision on the single Amazon EC2 instance in the Auto Scaling group.

**Note**
If you're deploying to Ubuntu Server Amazon EC2 instances, follow whatever instructions are appropriate to check your results, and then skip ahead to Step 4: Increase the Number of Amazon EC2 Instances in the Auto Scaling Group (p. 170).

**Topics**



## To use the AWS CLI and a web browser to check the results

First, you'll need the public DNS of the Amazon EC2 instance.

Use the AWS CLI to get the public DNS of the Amazon EC2 instance in the Auto Scaling group by calling the **describe-instances** command.

Before you call this command, you will need the ID of the Amazon EC2 instance, represented by the placeholder *`instanceID`* below. To get the ID, call the **describe-auto-scaling-groups** against **`CodeDeployDemo-AS-Group`** as you did before, for example:

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].InstanceId"
 --output text
```

Now call the **describe-instances** command:

```
aws ec2 describe-instances --instance-id instanceID --query "Reservations[0].In
stances[0].PublicDnsName" --output text
```

The returned value is the public DNS of the Amazon EC2 instance.

Once you have the public DNS, using a web browser, show the SimpleDemoApp revision that AWS CodeDeploy automatically deployed to that Amazon EC2 instance, using a URL such as the following:

```
http://ec2-01-234-567-890.compute-1.amazonaws.com
```

A web page should appear showing the congratulations page. If so, you've successfully used AWS CodeDeploy to deploy a revision to a single Amazon EC2 instance in an Auto Scaling group!

Next, you'll add an Amazon EC2 instance to the Auto Scaling group. After Auto Scaling adds the Amazon EC2 instance, AWS CodeDeploy will automatically deploy your revision to the new Amazon EC2 instance without any further work on your part.

## To use the Amazon EC2 console and a web browser to check the results

First, you'll need the public DNS of the Amazon EC2 instance.

Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

In the Amazon EC2 navigation pane, under **Auto Scaling**, click **Auto Scaling Groups** (and click **Auto Scaling Groups** again if needed). Then click the **`CodeDeployDemo-AS-Group`** entry.

In the **Instances** tab, click the Amazon EC2 instance ID in the list.

On the **Instances** page, in the **Description** tab, note the **Public DNS** value. It should look something like this: **`ec2-01-234-567-890.compute-1.amazonaws.com`**.

Once you have the public DNS, using a web browser, show the SimpleDemoApp revision that AWS CodeDeploy automatically deployed to that Amazon EC2 instance, using a URL such as the following:

```
http://ec2-01-234-567-890.compute-1.amazonaws.com
```

A web page should appear showing the congratulations page. If so, you've successfully used AWS CodeDeploy to deploy a revision to a single Amazon EC2 instance in an Auto Scaling group!

Next, you'll add an Amazon EC2 instance to the Auto Scaling group. After Auto Scaling adds the Amazon EC2 instance, AWS CodeDeploy will automatically deploy your revision to the new Amazon EC2 instance without any further work on your part.

# Step 4: Increase the Number of Amazon EC2 Instances in the Auto Scaling Group

In this step, you'll instruct the Auto Scaling group to create an additional Amazon EC2 instance. After Auto Scaling creates the new Amazon EC2 instance, AWS CodeDeploy will automatically deploy your revision to it.

**Topics**

- To use the Amazon EC2 console to scale up the number of Amazon EC2 instances in the deployment group (p. 171)

## To use the AWS CLI to scale up with the number of Amazon EC2 instances in the Auto Scaling group

1. Call the **update-auto-scaling-group** command to increase the Amazon EC2 instances in the Auto Scaling group named **`CodeDeployDemo-AS-Group`** from 1 to 2.

   For Linux, OS X, or Unix:

   ```
   aws autoscaling update-auto-scaling-group \
     --auto-scaling-group-name CodeDeployDemo-AS-Group \
     --min-size 2 \
     --max-size 2 \
     --desired-capacity 2
   ```

   For Windows:

   ```
   aws autoscaling update-auto-scaling-group --auto-scaling-group-name
   CodeDeployDemo-AS-Group --min-size 2 --max-size 2 --desired-capacity 2
   ```

2. Make sure that the Auto Scaling group now has 2 Amazon EC2 instances and that both Amazon EC2 instances are ready by calling the **describe-auto-scaling-groups** command against **`CodeDeployDemo-AS-Group`**:

   ```
   aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
   CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].[Health
   Status, LifecycleState]" --output text
   ```

   Do not proceed until both of the returned values show `Healthy` and `InService`.

## To use the Amazon EC2 console to scale up the number of Amazon EC2 instances in the deployment group

1. In the Amazon EC2 navigation bar, under **Auto Scaling**, click **Auto Scaling Groups** (and click **Auto Scaling Groups** again if needed). Then click the **`CodeDeployDemo-AS-Group`** entry.
2. Click **Actions**, and then click **Edit**.
3. On the **Details** tab, In the **Desired**, **Min**, and **Max** boxes, type **2**, and then click **Save**.
4. Click the **Instances** tab. The new Amazon EC2 instance should appear in the list of Amazon EC2 instances. (If the new Amazon EC2 instance does not appear at first, you may need to click the **Refresh** button a few times to get it to appear.) Do not proceed until the Amazon EC2 instance shows the **Lifecycle** column value of **InService** and the **Health Status** column value of **Healthy**.

# Step 5: Check Your Results Again

In this step, you'll check to see that AWS CodeDeploy automatically installed the SimpleDemoApp revision on the new Amazon EC2 instance in the Auto Scaling group.

**Note**
If you're deploying to Ubuntu Server Amazon EC2 instances, follow whatever instructions are appropriate to check your results, and then skip ahead to Step 6: Clean Up (p. 173).

**Topics**



# To use the AWS CLI to check automatic deployment results

1. Call the **get-deployment** command to get information about the automatic deployment.

   Before you call this command, you will need the ID of the automatic deployment, represented by the placeholder *deploymentID* below. To get the ID, call the **list-deployments** command against the application named **SimpleDemoApp** and the deployment group named **SimpleDemoDG**:

   ```
   aws deploy list-deployments --application-name SimpleDemoApp --deployment-
   group-name SimpleDemoDG --query "deployments" --output text
   ```

   There should be 2 deployment IDs showing this time. Use the deployment ID that you have not yet used in a call to the **get-deployment** command:

   ```
   aws deploy get-deployment --deployment-id deploymentID --query "deploy
   mentInfo.[status, creator]" --output text
   ```

   In the **get-deployment** command output, you should the deployment status as well as seeing `autoScaling` for the automatic deployment (which means that Auto Scaling automatically created the deployment).

   Do not proceed until the deployment status shows `Succeeded`.
2. Now call the **describe-instances** command to get the public DNS of the new Amazon EC2 instance.

   Before you call this command, you will need the ID of the new Amazon EC2 instance, represented by the placeholder *instanceID* below. To get this ID, make another call to the **describe-auto-scaling-groups** command against **CodeDeployDemo-AS-Group**:

   ```
   aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
   CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].InstanceId"
    --output text
   ```

   Now make a call to the **describe-instances** command:

   ```
   aws ec2 describe-instances --instance-id instanceID --query "Reserva
   tions[0].Instances[0].PublicDnsName" --output text
   ```

   In the output of the **describe-instances** command, note the public DNS for the new Amazon EC2 instance.
3. Once you have the public DNS, using a web browser, show the `SimpleDemoApp` revision that AWS CodeDeploy automatically deployed to that Amazon EC2 instance, using a URL such as the following:

   ```
   http://ec2-01-234-567-890.compute-1.amazonaws.com
   ```

A web page should appear, showing the congratulations page. If so, you've used AWS CodeDeploy to automatically deploy a revision to a scaled-up Amazon EC2 instance in an Auto Scaling group!

## To use the AWS CodeDeploy console to check automatic deployment results

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. In the AWS CodeDeploy service navigation bar, click **Deployments**. The **Deployments** page appears, showing information about the deployment that Auto Scaling created. Normally, you would create a new deployment on your own. However, Auto Scaling automatically created a deployment so that it could automatically deploy your revision to the new Amazon EC2 instance on your behalf.

   **Tip**
   To update the deployment's current status, use your browser's page refresh command.

3. After the deployment status shows **Succeeded**, verify the results on the new Amazon EC2 instance, first by getting the public DNS of the new Amazon EC2 instance:
4. In the Amazon EC2 navigation pane, under **Auto Scaling**, click **Auto Scaling Groups** (and click **Auto Scaling Groups** again if needed). Then click the **`CodeDeployDemo-AS-Group`** entry.
5. In the **Instances** tab, in the list, click the ID of the new Amazon EC2 instance.
6. On the **Instances** page, in the **Description** tab, note the **Public DNS** value. It should look something like this: **`ec2-01-234-567-890.compute-1.amazonaws.com`**.

Now that you have the public DNS of the new Amazon EC2 instance, show the `SimpleDemoApp` revision that AWS CodeDeploy automatically deployed to that Amazon EC2 instance, using a URL such as the following:

```
http://ec2-01-234-567-890.compute-1.amazonaws.com
```

A web page should appear, showing the congratulations page. If so, you've used AWS CodeDeploy to automatically deploy a revision to a scaled-up Amazon EC2 instance in an Auto Scaling group!

# Step 6: Clean Up

In this step, to keep from getting continually charged after this tutorial for resources that you used during this tutorial, you'll delete the Auto Scaling group. Optionally, you can delete the associated Auto Scaling configuration and associated AWS CodeDeploy deployment component records.

**Topics**



## To use the AWS CLI to clean up

1. Delete the Auto Scaling group, which also terminates the associated Amazon EC2 instances, by calling the **delete-auto-scaling-group** command against **`CodeDeployDemo-AS-Group`**:

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name
CodeDeployDemo-AS-Group --force-delete
```

2. Optionally, delete the associated Auto Scaling launch configuration by calling the **delete-launch-configuration** command against the launch configuration named **`CodeDeployDemo-AS-Configuration`**:

```
aws autoscaling delete-launch-configuration --launch-configuration-name
CodeDeployDemo-AS-Configuration
```

3. Optionally, delete the application from AWS CodeDeploy, which also deletes all associated deployment, deployment group, and revision records by calling the **delete-application** command against the application named **`SimpleDemoApp`**:

```
aws deploy delete-application --application-name SimpleDemoApp
```

### To use the AWS console to clean up

1. Delete the Auto Scaling group, which also terminates the associated Amazon EC2 instances:

   Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the Amazon EC2 navigation pane, under **Auto Scaling**, click **Auto Scaling Groups** (and click **Auto Scaling Groups** again if needed). Then click the **`CodeDeployDemo-AS-Group`** entry.
3. Click **Actions**, click **Delete**, and then click **Yes, Delete**.
4. Optionally, delete the associated launch configuration: in the navigation bar, under **Auto Scaling**, click **Launch Configurations** (and click **Launch Configurations** again if needed). Then click the **`CodeDeployDemo-AS-Configuration`** entry.
5. Click **Actions**, click **Delete launch configuration**, and then click **Yes, Delete**.
6. Optionally, delete the application from AWS CodeDeploy, which also deletes all associated deployment, deployment group, and revision records: open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.
7. In the AWS CodeDeploy service navigation bar, click **Applications**.
8. In the list of applications, click **SimpleDemoApp**.
9. Click **Delete Application**, and then click **Delete**.

# Logging AWS CodeDeploy API Calls By Using AWS CloudTrail

AWS CodeDeploy is integrated with CloudTrail, a service that captures API calls made by or on behalf of AWS CodeDeploy in your AWS account and delivers the log files to an Amazon S3 bucket that you specify. CloudTrail captures API calls from the AWS CodeDeploy console, from AWS CodeDeploy commands through the AWS CLI, or from the AWS CodeDeploy APIs directly. Using the information collected by CloudTrail, you can determine what request was made to AWS CodeDeploy, the source IP address from which the request was made, who made the request, when it was made, and so on. To learn more about CloudTrail, including how to configure and enable it, see the AWS CloudTrail User Guide.

# AWS CodeDeploy Information in CloudTrail

When CloudTrail logging is enabled in your AWS account, API calls made to AWS CodeDeploy actions are tracked in log files. AWS CodeDeploy records are written together with other AWS service records in a log file. CloudTrail determines when to create and write to a new file based on a time period and file size.

All of the AWS CodeDeploy actions are logged and are documented in the AWS CodeDeploy Command Line Reference and the AWS CodeDeploy API Reference. For example, calls to create deployments, delete applications, and register application revisions generate entries in CloudTrail log files.

Every log entry contains information about who generated the request. The user identity information in the log helps you determine whether the request was made with root or IAM user credentials, with temporary security credentials for a role or federated user, or by another AWS service. For more information, see the **userIdentity** field in the CloudTrail Event Reference.

You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted by using Amazon S3 server-side encryption (SSE).

You can choose to have CloudTrail publish Amazon SNS notifications when new log files are delivered if you want to take quick action upon log file delivery. For more information, see Configuring Amazon SNS Notifications.

You can also aggregate AWS CodeDeploy log files from multiple AWS regions and multiple AWS accounts into a single Amazon S3 bucket. For more information, see Aggregating CloudTrail Log Files to a Single Amazon S3 Bucket.

# Understanding AWS CodeDeploy Log File Entries

CloudTrail log files can contain one or more log entries where each entry is made up of multiple JSON-formatted events. A log entry represents a single request from any source and includes information about the requested action, any parameters, the date and time of the action, and so on. The log entries are not guaranteed to be in any particular order. That is, they are not an ordered stack trace of the public API calls.

The following example shows a CloudTrail log entry that demonstrates the AWS CodeDeploy create deployment group action:

```
{
 "Records": [{
  "eventVersion": "1.02",
  "userIdentity": {
   "type": "AssumedRole",
   "principalId": "AKIAI44QH8DHBEXAMPLE:203.0.113.11",
   "arn": "arn:aws:sts::123456789012:assumed-role/example-role/203.0.113.11",
   "accountId": "123456789012",
   "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
   "sessionContext": {
    "attributes": {
     "mfaAuthenticated": "false",
     "creationDate": "2014-11-27T03:57:36Z"
    },
    "sessionIssuer": {
     "type": "Role",
     "principalId": "AKIAI44QH8DHBEXAMPLE",
     "arn": "arn:aws:iam::123456789012:role/example-role",
```

```
     "accountId": "123456789012",
     "userName": "example-role"
    }
   }
  },
  "eventTime": "2014-11-27T03:57:36Z",
  "eventSource": "codedeploy.amazonaws.com",
  "eventName": "CreateDeploymentGroup",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.11",
  "userAgent": "example-user-agent-string",
  "requestParameters": {
   "applicationName": "ExampleApplication",
   "serviceRoleArn": "arn:aws:iam::123456789012:role/example-instance-group-
role",
   "deploymentGroupName": "ExampleDeploymentGroup",
   "ec2TagFilters": [{
                "value": "CodeDeployDemo",
    "type": "KEY_AND_VALUE",
    "key": "Name"
           }],
           "deploymentConfigName": "CodeDeployDefault.HalfAtATime"
  },
  "responseElements": {
   "deploymentGroupId": "7d64e680-e6f4-4c07-b10a-9e117EXAMPLE"
  },
  "requestID": "86168559-75e9-11e4-8cf8-75d18EXAMPLE",
  "eventID": "832b82d5-d474-44e8-a51d-093ccEXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
 },
    ... additional entries ...
    ]
}
```

# AWS CodeDeploy Integration with Elastic Load Balancing

AWS CodeDeploy supports Elastic Load Balancing. Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances. (For more information about Elastic Load Balancing, see What is Elastic Load Balancing.)

When you run an application such as a web service, you will likely have your corresponding Amazon EC2 instances registered with Elastic Load Balancing load balancers. When you're ready to use AWS CodeDeploy to deploy new code to one of those Amazon EC2 instances, you don't want the load balancers to continue sending traffic to the Amazon EC2 instance during the deployment. You can coordinate this kind of deployment by using deployment lifecycle event scripts in AWS CodeDeploy to deploy new code to Amazon EC2 instances that are registered with load balancers.

> **Tip**
> You can also coordinate deployments in AWS CodeDeploy with Amazon EC2 instances that are part of an Auto Scaling group. For more information, see Auto Scaling Integration (p. 157).

We provide instructions and a sample that you can adapt to use AWS CodeDeploy with Elastic Load Balancing. For details, see the Elastic Load Balancing section of the AWS CodeDeploy Samples repository

on GitHub. This repository includes three sample scripts—`register_with_elb.sh`, `deregister_from_elb.sh`, and `common_functions.sh`—that provide all of the code that you need to get going. Simply edit the placeholders within these three scripts, and then reference these scripts from your application revision's `appspec.yml` file.

To coordinate deployments in AWS CodeDeploy with Amazon EC2 instances that are registered with Elastic Load Balancing load balancers, do the following:

1. Make sure that each of your target Amazon EC2 instances has the AWS CLI installed.
2. Make sure that each of your target Amazon EC2 instances has an IAM instance profile attached with at minimum the permissions **elasticloadbalancing:*** and **autoscaling:***.
3. Include in your application's source code directory the deployment lifecycle event scripts from the sample named `register_with_elb.sh`, `deregister_from_elb.sh`, and `common_functions.sh`.
4. In your application revision's AppSpec File, instruct AWS CodeDeploy to run the `register_with_elb.sh` script during the **ApplicationStart** event and to run the `deregister_from_elb.sh` script during the **ApplicationStop** event.
5. In the `common_functions.sh` script from the sample, specify the names of the Elastic Load Balancing load balancers, along with some other miscellaneous deployment settings.
6. Bundle your application's source code, your AppSpec File, and the deployment lifecycle event scripts into an application revision; and then upload the revision. When you're ready, deploy the revision to the Amazon EC2 instances. During the deployment, the deployment lifecycle event scripts will deregister the Amazon EC2 instance with the load balancers, wait for the connection to drain, and then re-register the Amazon EC2 instance with the load balancers after the deployment is done.

# AWS CodeDeploy Integration with GitHub

AWS CodeDeploy supports GitHub. GitHub is a web-based code hosting and sharing service. For more information about GitHub, see About GitHub.

With GitHub integration, AWS CodeDeploy can deploy application revisions that are stored in GitHub repositories to instances. This is in addition to AWS CodeDeploy supporting the deployment of revisions from Amazon S3 buckets to instances.

> **Tip**
> To view a video about this topic, see AWS CodeDeploy & GitHub Integration - Demo on YouTube.

**Topics**



## Deploying AWS CodeDeploy Revisions from GitHub

To deploy an application revision from a GitHub repository to instances, you need to do just a few things:

1. Create a revision that's compatible with AWS CodeDeploy. Of course, make sure that your revision is compatible with the instance type that you want to deploy to.

   To create a compatible revision, follow the instructions in Plan a Revision (p. 114) and Add an AppSpec File (p. 115).
2. Use a new or existing GitHub account to add your revision to a new or existing GitHub repository.

To create a new GitHub account, see Join GitHub.

To create a new GitHub repository, see Create A Repo.

3. Use a tool such as the AWS CodeDeploy console's **Create New Deployment** page or the AWS CLI **create-deployment** command to deploy your revision from your GitHub repository to target instances that are configured to participate in AWS CodeDeploy deployments.

   If you want to call the **create-deployment** command, you must first briefly use the AWS CodeDeploy console's **Create New Deployment** page to give AWS CodeDeploy permission to interact with GitHub on behalf of your preferred GitHub account for the specified application in AWS CodeDeploy. You only need to do this once per application.

   To learn how to use the **Create New Deployment** page to deploy from a GitHub repository, see Create a New Deployment (p. 135).

   To learn how to call the **create-deployment** command to deploy from a GitHub repository, see Deploy a Revision by Using the AWS CLI (p. 125).

   To learn how to prepare instances to participate in AWS CodeDeploy deployments, see Configure Instances (p. 72).

To experiment with completing these activities, see Tutorial: Deploy from GitHub (p. 180).

# GitHub Behaviors with AWS CodeDeploy

**Topics**



## GitHub Authentication Behaviors with Applications in AWS CodeDeploy

After you give AWS CodeDeploy permission to interact with GitHub on behalf of a GitHub account for an application in AWS CodeDeploy, the linkage between that specific GitHub account and application is stored in the AWS CodeDeploy system. You can change this by linking the application to a different GitHub account. You can also revoke permission for AWS CodeDeploy to interact with GitHub on behalf of a GitHub account for all applications that are linked to that account in the AWS CodeDeploy system.

### To link a different GitHub account to an application in AWS CodeDeploy

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. On the AWS CodeDeploy service navigation bar, click **Deployments**.
3. Click **Create New Deployment**.

   **Note**
   You do not have to go through with creating a new deployment. However, currently this is the only way to link a different GitHub account to an application.

4. In the **Application** drop-down list, select the application that you want to link to a different GitHub account.
5. Next to **Revision Type**, select the **My application is stored in GitHub** option.
6. Click **Reconnect to GitHub**. A new web page appears, prompting you to authorize AWS CodeDeploy to interact with GitHub on behalf of your GitHub account for the selected application.

   **Note**
   If you do not see a **Reconnect to GitHub** link, then you have not yet given AWS CodeDeploy permission to interact with GitHub. To give permission for the first time, click **Connect to GitHub**, and then skip ahead to the next step.
   If you see a blank web page that briefly appears and then disappears, and you see neither a **Reconnect with GitHub** link or a **Connect with GitHub** button, this is because you have already given AWS CodeDeploy permission to interact with GitHub, and you are currently signed in to GitHub. To show the **Reconnect with GitHub** link, sign out of GitHub, and then repeat this section's steps from the beginning.
7. If you are not already signed in to GitHub, follow the on-screen instructions on the **Sign in** page to sign in with a different GitHub account that you want to link to the application.
8. On the **Authorize application** page, click **Authorize application**. GitHub gives AWS CodeDeploy permission to interact with GitHub on behalf of the signed-in GitHub account for the selected application. The web page then disappears.
9. If you do not want to go through with creating a new deployment at this time, click **Cancel**.

**To revoke permission for AWS CodeDeploy to interact with GitHub on behalf of a GitHub account for all applications that the account is linked to in the AWS CodeDeploy system**

1. Go to your GitHub dashboard for the GitHub account that you want to revoke AWS CodeDeploy permission for.
2. Click the **Settings** (gear) icon.
3. Click **Applications**.
4. In the **Authorized applications** box, next to **AWS CodeDeploy**, click **Revoke**.
5. When prompted, click **I understand, revoke access**. GitHub revokes permission for AWS CodeDeploy to interact with GitHub on behalf of the signed-in GitHub account for all applications that the account is linked to in the AWS CodeDeploy system.

## AWS CodeDeploy Interaction Behaviors with Private and Public GitHub Repositories

AWS CodeDeploy supports deploying applications from both private and public GitHub repositories. When you give AWS CodeDeploy permission to access GitHub on your behalf, AWS CodeDeploy will have read-write access to all of the private GitHub repositories that your GitHub account has access to. However, note that AWS CodeDeploy only reads from GitHub repositories. AWS CodeDeploy will not write to any of your private GitHub repositories.

## Automatically Deploy from GitHub with AWS CodeDeploy

You can automatically trigger a deployment from a GitHub repository whenever the source code in that repository changes. For instructions, see Automatically Deploy from GitHub Using AWS CodeDeploy.

# Tutorial: Using AWS CodeDeploy to Deploy an Application from GitHub

In this tutorial, you'll practice using AWS CodeDeploy to deploy one of our sample application revisions from GitHub to a single Amazon Linux Amazon EC2 instance or Windows Server instance. For general information about GitHub integration with AWS CodeDeploy, see GitHub Integration (p. 177).

> **Note**
> You can also use AWS CodeDeploy to deploy an application revision from GitHub to an Ubuntu Server instance. You can create and use the sample revision that is described in Step 2 (p. 51) of the On-Premises Instance Deployment (Windows Server or Ubuntu Server) (p. 50) tutorial, or you would need to create and use your own revision that is compatible with both an Ubuntu Server instance and AWS CodeDeploy. To create a revision on your own, see Plan a Revision (p. 114) and Add an AppSpec File (p. 115).

**Topics**



## Prerequisites

Before you can start using this tutorial, you should review the following requirements.

- Install Git on your local machine, if you haven't done so already. To install Git, see Git Downloads.
- Complete the steps in Setting Up (p. 4), if you haven't done so already, including installing and configuring the AWS CLI. This is especially important if you want to use the AWS CLI to deploy a revision from GitHub to the instance.

## Step 1: Set Up a GitHub Account

In this step, you will set up a GitHub account. You will need a GitHub account so that you can create a GitHub repository to store the revision that you want to deploy.

If you already have a GitHub account, skip ahead to Step 2: Create a GitHub Repository (p. 181).

1. Go to https://github.com/join.
2. Enter a user name, your email address, and a password.
3. Click **Create an account**, and then follow the on-screen instructions to complete the sign up process.
4. On the final page of the sign up process, click **Take me to my dashboard**. If your GitHub dashboard does not appear, go to https://github.com/dashboard.

# Step 2: Create a GitHub Repository

In this step, you will create a GitHub repository. You will need a GitHub repository to store the revision that you want to deploy.

If you already have a GitHub repository that you want to use, be sure to substitute your repository's name for **`CodeDeployGitHubDemo`** throughout this tutorial, and then skip ahead to Step 3: Upload a Sample Application to Your GitHub Repository (p. 182).

1. From your GitHub dashboard, do one of the following:
   - In the **Your repositories** area, click **New repository**.
   - On the navigation bar, click **Create new** (the plus symbol (**+**)), and then click **New repository**.
2. In the **Repository name** box, type **`CodeDeployGitHubDemo`**.
3. With the **Public** option selected, check the box labelled **Initialize this repository with a README**, and then click **Create repository**.

   **Note**
   Leaving the default **Public** option selected means that anyone can see this repository, which is fine for this tutorial. Optionally, you can select the **Private** option to limit who can see and commit to the repository; however, selecting the **Private** option may result in additional charges from GitHub.
4. Follow the on-screen instructions to use the command line to create the new repository. For example:

   For Linux, OS X, or Unix:

   1. From the terminal, run the following commands, one at a time, where *`user-name`* is your GitHub user name:

      ```
      mkdir /tmp/CodeDeployGitHubDemo
      cd /tmp/CodeDeployGitHubDemo
      touch README.md
      git init
      git add README.md
      git commit -m "My first commit"
      git remote add origin https://github.com/user-name/CodeDeployGitHub
      Demo.git
      git push -u origin master
      ```

   2. Leave the command prompt open in the `/tmp/CodeDeployGitHubDemo` location.

   For Windows:

   1. From a command prompt running as an administrator, run the following commands, one at a time:

      ```
      mkdir c:\temp\CodeDeployGitHubDemo
      cd c:\temp\CodeDeployGitHubDemo
      notepad README.md
      ```

   2. In Notepad, save the `README.md` file. Close Notepad, and then run the following commands, one at a time, where *`user-name`* is your GitHub user name:

```
git init
git add README.md
git commit -m "My first commit"
git remote add origin https://github.com/user-name/CodeDeployGitHub
Demo.git
git push -u origin master
```

3. Leave the command prompt open in the `c:\temp\CodeDeployGitHubDemo` location.

# Step 3: Upload a Sample Application to Your GitHub Repository

In this step, you will copy one of our sample revisions from one of our public Amazon S3 buckets over to your GitHub repository.

> **Note**
> You can substitute one of your own revisions for our sample revision. If so, your revision must follow the guidelines in Plan a Revision (p. 114) and Add an AppSpec File (p. 115), it must work with the corresponding instance type, and you must be able to access it from your GitHub dashboard. If your revision meets these requirements, then skip ahead to Step 5: Deploy the Application to the Instance (p. 183).
> If you're deploying to an Ubuntu Server instance, you'll need to upload to your GitHub repository your own revision that is compatible with both an Ubuntu Server instance and AWS CodeDeploy. For more information, see Plan a Revision (p. 114) and Add an AppSpec File (p. 115).

With your terminal or administrative command prompt still open in, for example, the `/tmp/CodeDeployGitHubDemo` location (for Linux, OS X, or Unix) or `c:\temp\CodeDeployGitHubDemo` (for Windows), run the following commands, one at a time:

To push our sample revision that can be deployed to an Amazon Linux Amazon EC2 instance:

```
(Amazon S3 copy command)
git add SampleApp_Linux.zip
git commit -m "Added Linux sample app"
git push
```

Where *(Amazon S3 copy command)* is one of the following:

- `aws s3 cp s3://aws-codedeploy-us-east-1/samples/latest/SampleApp_Linux.zip . --region us-east-1` for the US East (N. Virginia) region
- `aws s3 cp s3://aws-codedeploy-us-west-2/samples/latest/SampleApp_Linux.zip . --region us-west-2` for the US West (Oregon) region
- `aws s3 cp s3://aws-codedeploy-eu-west-1/samples/latest/SampleApp_Linux.zip . --region eu-west-1` for the EU (Ireland) region
- `aws s3 cp s3://aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Linux.zip . --region ap-southeast-2` for the Asia Pacific (Sydney) region

To push our sample revision that can be deployed to a Windows Server instance:

```
(Amazon S3 copy command)
git add SampleApp_Windows.zip
git commit -m "Added Windows sample app"
git push
```

Where *(Amazon S3 copy command)* is one of the following:

- `aws s3 cp s3://aws-codedeploy-us-east-1/samples/latest/SampleApp_Windows.zip . --region us-east-1` for the US East (N. Virginia) region
- `aws s3 cp s3://aws-codedeploy-us-west-2/samples/latest/SampleApp_Windows.zip . --region us-west-2` for the US West (Oregon) region
- `aws s3 cp s3://aws-codedeploy-eu-west-1/samples/latest/SampleApp_Windows.zip . --region eu-west-1` for the EU (Ireland) region
- `aws s3 cp s3://aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Windows.zip . --region ap-southeast-2` for the Asia Pacific (Sydney) region

To push your own revision that can be deployed to an Ubuntu Server instance, copy your revision into your local repo, and then call the following:

```
git add your-revision-file-name
git commit -m "Added Ubuntu app"
git push
```

# Step 4: Provision an Instance

In this step, you will create a new Amazon Linux Amazon EC2 instance or Windows Server or Ubuntu Server instance—or use an existing Amazon Linux Amazon EC2 instance or Windows Server or Ubuntu Server instance—that is properly configured to participate in AWS CodeDeploy deployments.

To do this, follow the instructions in Configure Instances (p. 72), and then return to this page. After you have successfully launched the instance and verified that the AWS CodeDeploy Agent is running on the instance, proceed to Step 5: Deploy the Application to the Instance (p. 183).

If you already have an instance that is properly configured to participate in AWS CodeDeploy deployments, skip ahead to Step 5: Deploy the Application to the Instance (p. 183).

# Step 5: Deploy the Application to the Instance

In this step, you will use the AWS CodeDeploy console or the AWS CLI to deploy the sample revision from your GitHub repository to your instance. If you're using one of our sample revisions to deploy to an Amazon Linux Amazon EC2 instance or Windows Server instance, our sample revision deploys a single web page to the instance.

### To use the AWS CodeDeploy console to deploy the revision

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. If the **Applications** page is not visible, then on the AWS CodeDeploy service navigation bar, click **Applications**.

3. On the **Applications** page, click **Create New Application**.
4. In the **Application Name** box, type **`CodeDeployGitHubDemo-App`**.
5. In the **Deployment Group Name** box, type **`CodeDeployGitHubDemo-DepGrp`**.
6. If you're deploying to an Amazon EC2 instance, then in the list of tags, select **Amazon EC2** in the **Tag Type** drop-down list. In the **Key** and **Value** boxes, type the Amazon EC2 instance tag key and value that was applied to your Amazon EC2 instance as part of Step 4: Provision an Instance (p. 183).
7. If you're deploying to an on-premises instance, then in the list of tags, select **On-Premises Instance** in the **Tag Type** drop-down list. In the **Key** and **Value** boxes, type the on-premises instance tag key and value that was applied to your on-premises instance as part of Step 4: Provision an Instance (p. 183).
8. In the **Deployment Config** drop-down list, leave the default of **CodeDeployDefault.OneAtATime**.
9. In the **Service Role ARN** drop-down list, select an appropriate service role ARN. (If you don't know what the service role ARN is, follow the instructions in Use the IAM Console to Get the Service Role ARN (p. 143) to find it.)
10. Click **Create Application**. AWS CodeDeploy creates the application and then displays the application details page for `CodeDeployGitHubDemo-App`.
11. On the application details page, in the **Deployment Groups** area, click the drop-down arrow next to `CodeDeployGitHubDemo-DepGrp`.
12. Click **Deploy New Revision**. The **Create New Deployment** page appears.
13. Next to **Revision Type**, select the **My application is stored in GitHub** option.
14. Click **Connect with GitHub**. A new web page appears, prompting you to authorize AWS CodeDeploy to interact with GitHub on behalf of your GitHub account for the application named `CodeDeployGitHubDemo-App`.

    **Note**
    If you see a **Reconnect with GitHub** link instead of a **Connect with GitHub** button, this is expected behavior. Do not click the link; continue to the next step. For information about why this happens, see GitHub Authentication Behaviors with Applications in AWS CodeDeploy (p. 178).
    If you see a blank web page that briefly appears and then disappears, and you see neither a **Reconnect with GitHub** link or a **Connect with GitHub** button, this also expected behavior; continue to the next step. For information about why this happens, see GitHub Authentication Behaviors with Applications in AWS CodeDeploy (p. 178).
15. If you are not already signed in to GitHub, follow the on-screen instructions on the **Sign in** page to sign in with your GitHub account.
16. On the **Authorize application** page, click **Authorize application**. The web page disappears.
17. Back in the **Create New Deployment** page, in the **Repository Name** box, type the GitHub user name that you are signed in with, followed by a forward slash character (`/`), followed by the name of the repository that you pushed your application revision to; for example ***`My-GitHub-User-Name`*`/CodeDeployGitHubDemo`**. If you are unsure of the value to type, or if you want to specify a different repository:

    1. In a separate web browser tab, go to your GitHub dashboard.
    2. In the **Your repositories** area, hover your mouse pointer over the target repository name. A tooltip appears, displaying the GitHub user or organization name, followed by a forward slash character (`/`), followed by the name of the repository. Type this displayed value into the **Repository Name** box.

       **Tip**
       If the target repository name is not visible in the **Your repositories** area, use the **Search GitHub** box to find the target repository and corresponding GitHub user or organization name.

18. In the **Commit ID** box, type the ID of the commit that is associated with the push of your application revision to GitHub. If you are unsure of the value to type:

    1. In a separate web browser tab, go to your GitHub dashboard.
    2. In the **Your repositories** area, click **CodeDeployGitHubDemo**.

       **Tip**
       If the target repository name is not visible in the **Your repositories** area, use the **Search GitHub** box to find the target repository and corresponding GitHub user or organization name, and then click on the target repository's name.

    3. In the list of commits, find and copy the commit ID that is associated with the push of your application revision to GitHub. This ID is typically 40 characters in length and consists of both letters and numbers. (Do not use the shorter version of the commit ID, which is typically the first 10 characters of the longer version of the commit ID.)
    4. Paste the commit ID into the **Commit ID** box.

19. Leave the **Deployment Description** box blank.
20. With the **Deployment Config** drop-down list showing the default of **CodeDeployDefault.OneAtATime**, click **Deploy Now**. The **Deployments** page appears, showing status information about the newly created deployment.

Now that you've used the AWS CodeDeploy console to deploy the revision to the instance, you can skip ahead to Step 6: Monitor and Verify the Deployment (p. 187).

### To use the AWS CLI to deploy the revision

1. Call the **create-application** command to create an application in AWS CodeDeploy named `CodeDeployGitHubDemo-App`:

```
aws deploy create-application --application-name CodeDeployGitHubDemo-App
```

2. Call the **create-deployment-group** command to create a deployment group named `CodeDeployGitHubDemo-DepGrp`, where:

   - If you're deploying to an Amazon EC2 instance, *`EC2-tag-key`* is the Amazon EC2 instance tag key that was applied to your Amazon EC2 instance as part of Step 4: Provision an Instance (p. 183).
   - If you're deploying to an Amazon EC2 instance, *`EC2-tag-value`* is the Amazon EC2 instance tag value that was applied to your Amazon EC2 instance as part of Step 4: Provision an Instance (p. 183).
   - If you're deploying to an on-premises instance, *`on-premises-tag-key`* is the on-premises instance tag key that was applied to your on-premises instance as part of Step 4: Provision an Instance (p. 183).
   - If you're deploying to an on-premises instance, *`on-premises-tag-value`* is the on-premises instance tag value that was applied to your on-premises instance as part of Step 4: Provision an Instance (p. 183).
   - *`service-role-ARN`* is an appropriate service role ARN. (If you don't know what the service role ARN is, follow the instructions in Use the AWS CLI to Get the Service Role ARN (p. 144) to find it.)

```
aws deploy create-deployment-group --application-name CodeDeployGitHubDemo-
App --ec2-tag-filters Key=EC2-tag-key,Type=KEY_AND_VALUE,Value=EC2-tag-value
 --on-premises-tag-filters Key=on-premises-tag-
```

```
key,Type=KEY_AND_VALUE,Value=on-premises-tag-value --deployment-group-name
 CodeDeployGitHubDemo-DepGrp --service-role-arn service-role-ARN
```

3. Before you can call any AWS CLI commands that interact with GitHub for the specified application in AWS CodeDeploy (such as the **create-deployment** command, which you will call next), you must give AWS CodeDeploy permission to interact with GitHub on behalf of your GitHub user account for the `CodeDeployGitHubDemo-App` application. Currently, the only way to do this is through the AWS CodeDeploy console.

   Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

4. On the AWS CodeDeploy service navigation bar, click **Deployments**.
5. Click **Create New Deployment**.

   **Note**
   You will not be creating a new deployment through this page. However, this is currently the only way to give AWS CodeDeploy permission to interact with GitHub on behalf of your GitHub user account for the specified application.

6. In the **Application** drop-down list, select **CodeDeployGitHubDemo-App**.
7. In the **Deployment Group** drop-down list, select **CodeDeployGitHubDemo-DepGrp**.
8. Next to **Revision Type**, click **My application revision is stored in GitHub**.
9. Click **Connect With GitHub**.

   **Note**
   If you see a **Reconnect with GitHub** link instead of a **Connect with GitHub** button, this is expected behavior. Do not click the link; continue to the next step. For information about why this happens, see GitHub Authentication Behaviors with Applications in AWS CodeDeploy (p. 178).
   If you see a blank web page that briefly appears and then disappears, and you see neither a **Reconnect with GitHub** link or a **Connect with GitHub** button, this also expected behavior; continue to the next step. For information about why this happens, see GitHub Authentication Behaviors with Applications in AWS CodeDeploy (p. 178).

10. If you are not already signed in to GitHub, follow the on-screen instructions on the **Sign in** page to sign in with your preferred GitHub user name or email and password.
11. On the **Authorize application** page, click **Authorize application**. The web page disappears.
12. Now that AWS CodeDeploy has the necessary permission, click **Cancel** to stop using the **Create New Deployment** page, and continue using the AWS CLI.
13. Call the **create-deployment** command to deploy the revision from your GitHub repository to the instance, where:

    - *`repository`* is your GitHub account name, followed by a forward-slash character (`/`), followed by the name of your repository (`CodeDeployGitHubDemo`); for example, `MyGitHubUserName/CodeDeployGitHubDemo`. If you are unsure of the value to use, or if you want to specify a different repository:

      1. In a separate web browser tab, go to your GitHub dashboard.
      2. In the **Your repositories** area, hover your mouse pointer over the target repository name. A tooltip appears, displaying the GitHub user or organization name, followed by a forward slash character (`/`), followed by the name of the repository. Use this displayed value.

**Tip**
If the target repository name is not visible in the **Your repositories** area, use the **Search GitHub** box to find the target repository and corresponding GitHub user or organization name.

- *commitId* is the ID of the commit in your repository that references the version of the application revision that you pushed to your repository; for example, `f835159a...528eb76f`. If you are unsure of the value to use:

  1. In a separate web browser tab, go to your GitHub dashboard.
  2. In the **Your repositories** area, click **CodeDeployGitHubDemo**.

     **Tip**
     If the target repository name is not visible in the **Your repositories** area, use the **Search GitHub** box to find the target repository and corresponding GitHub user or organization name, and then click on the target repository's name.

  3. In the list of commits, find the commit ID that is associated with the push of your application revision to GitHub. This ID is typically 40 characters in length and consists of both letters and numbers. (Do not use the shorter version of the commit ID, which is typically the first 10 characters of the longer version of the commit ID.) Use this value.

For Linux, OS X, or Unix:

```
aws deploy create-deployment \
  --application-name CodeDeployGitHubDemo-App \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --deployment-group-name CodeDeployGitHubDemo-DepGrp \
  --description "My GitHub deployment demo" \
  --github-location repository=repository,commitId=commitId
```

For Windows:

```
aws deploy create-deployment --application-name CodeDeployGitHubDemo-App -
-deployment-config-name CodeDeployDefault.OneAtATime --deployment-group-name
 CodeDeployGitHubDemo-DepGrp --description "My GitHub deployment demo" --
github-location repository=repository,commitId=commitId
```

# Step 6: Monitor and Verify the Deployment

In this step, you will use either the AWS CodeDeploy console or the AWS CLI to see if the deployment succeeded. If the deployment succeeded, you will use your web browser to view the web page that was deployed to each participating Amazon Linux Amazon EC2 instance or Windows Server instance.

**Note**
If you're deploying to an Ubuntu Server instance, use your own testing strategy to determine whether the deployed revision works on the instance as expected, and then skip ahead to Step 7: Clean Up (p. 188).

### To use the AWS CodeDeploy console to monitor and verify the deployment

1. In the AWS CodeDeploy console, if the **Deployments** page is not visible, in the AWS CodeDeploy service navigation bar, click **Deployments**.
2. In the list of deployments, look for the row with an **Application** value of **CodeDeployGitHubDemo-App** and a **Deployment Group** value of **CodeDeployGitHubDemo-DepGrp**. If the **Status** column doesn't show either **Succeeded** or **Failed** for this row, periodically click the refresh button next to the table until you see either **Succeeded** or **Failed**.
3. If you see **Failed**, you can try troubleshooting the deployment by following the instructions in View Instance Details by Using the AWS CodeDeploy Console (p. 128).
4. If you see **Succeeded**, you can now try verifying the deployment through your web browser. Our sample revision deploys a single web page to the instance which, if you're deploying to an Amazon EC2 instance, you can view in your web browser by going to `http://`*`PublicDNS`* for the instance, for example `http://ec2-01-234-567-890.compute-1.amazonaws.com`.
5. If you can see the web page, then congratulations! You have successfully used AWS CodeDeploy to deploy from your GitHub repository.

Now that you've successfully used AWS CodeDeploy to deploy a revision from GitHub, you can skip ahead to Step 7: Clean Up (p. 188).

### To use the AWS CLI to monitor and verify the deployment

1. Call the **list-deployments** command to get the deployment ID for the application named `CodeDeployGitHubDemo-App` and the deployment group named `CodeDeployGitHubDemo-DepGrp`:

```
aws deploy list-deployments --application-name CodeDeployGitHubDemo-App --
deployment-group-name CodeDeployGitHubDemo-DepGrp --query "deployments" --
output text
```

2. Call the **get-deployment** command, supplying the ID of the deployment that was output from the **list-deployments** command:

```
aws deploy get-deployment --deployment-id deployment-ID --query "deploy
mentInfo.[status, creator]" --output text
```

3. If **Failed** is returned, you can try troubleshooting the deployment by following the instructions in View Instance Details by Using the AWS CodeDeploy Console (p. 128).
4. If **Succeeded** is returned, you can now try verifying the deployment through your web browser. Our sample revision deploys a single web page to the instance which, if you're deploying to an Amazon EC2 instance, you can view in your web browser by going to `http://`*`PublicDNS`* for the Amazon EC2 instance, for example `http://ec2-01-234-567-890.compute-1.amazonaws.com`.
5. If you can see the web page, then congratulations! You have successfully used AWS CodeDeploy to deploy from your GitHub repository.

# Step 7: Clean Up

To keep you from getting continually charged for resources that you used during this tutorial, you must clean up the associated resources. To do so, you must terminate any Amazon EC2 instance that you launched and its associated resources. Optionally, you can delete the AWS CodeDeploy deployment component records that are associated with this tutorial. If you were using a GitHub repository just for this tutorial, you can delete that repository now, too.

## To delete a AWS CloudFormation stack (if you used our AWS CloudFormation template to create an Amazon EC2 instance)

1. Sign in to the AWS Management Console and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. Check the box in the row next to the stack starting with `CodeDeploySampleStack` in the **Stack Name** column.
3. Click **Delete Stack**.
4. When prompted, click **Yes, Delete**. The associated Amazon EC2 instance is terminated, and the Amazon EC2 instance's associated IAM instance profile and service role are also deleted.

## To manually deregister and clean up an on-premises instance (if you provisioned an on-premises instance)

1. Use the AWS CLI to call the **deregister** command against the on-premises instance represented here by *your-instance-name*, and the associated region represented here by *your-region*:

```
aws deploy deregister --instance-name your-instance-name --delete-iam-user
 --region your-region
```

2. From the on-premises instance, call the **uninstall** command:

```
aws deploy uninstall
```

## To manually terminate an Amazon EC2 instance (if you manually launched an Amazon EC2 instance)

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **Instances**, click **Instances**.
3. Check the box next to the Amazon EC2 instance that you want to terminate. Click **Actions**, point to **Instance State**, and then click **Terminate**.
4. When prompted, click **Yes, Terminate**. The Amazon EC2 instance is terminated.

## To delete the AWS CodeDeploy deployment component records

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   **Note**
   Make sure that you sign in using the same account or IAM user information that you used when setting up the AWS CLI in Setting Up (p. 4).

2. If the **Applications** page is not visible, in the AWS CodeDeploy service navigation bar, click **Applications**.
3. Click the application named **CodeDeployGitHubDemo-App**.

4. Click **Delete Application**.
5. When prompted, click **Delete**. AWS CodeDeploy deletes from its system all records about the application and its associated deployment groups, revisions, and deployments.

## To delete your GitHub repository

1. Open your GitHub dashboard, if it is not already visible.
2. In the **Your repositories** area, click **CodeDeployGitHubDemo**.
3. In the repository navigation pane, click **Settings** (next to the tools icon).
4. In the **Danger Zone** area, click **Delete this repository**.
5. When prompted, type **`CodeDeployGitHubDemo`**, and then click **I understand the consequences, delete this repository**. GitHub permanently deletes the repository from its system.
6. On your local computer, delete the `/tmp/CodeDeployGitHubDemo` folder and its contents (for Linux, OS X, or Unix) or the `c:\temp\CodeDeployGitHubDemo` folder and its contents (for Windows).

# Troubleshooting AWS CodeDeploy

**Topics**



## General Troubleshooting Issues

**Topics**



### General Troubleshooting Checklist

To troubleshoot a failed deployment, you can use the following checklist of common areas to investigate:

1. Determine why the deployment may have failed. For more information, see View Deployment Details (p. 127) and View Instance Details (p. 128). If you are unable to determine why the deployment may have failed, continue on with the rest of the items in this checklist.
2. Check whether you have correctly configured each of the instances that you want to deploy to:
   - An Amazon EC2 instance needs to have been launched with an Amazon EC2 key pair specified. For more information, see Amazon EC2 Key Pairs.
   - An Amazon EC2 instance needs to have been launched with the correct IAM instance profile. For more information, see Use an Existing Amazon EC2 Instance (p. 88) and Create an IAM Instance Profile (p. 84).

- An Amazon EC2 instance must be tagged with the tags that you anticipate using during your deployments. For more information, see Working with Tags in the Console.
- The AWS CodeDeploy Agent is installed and running on the instance. For more information, see AWS CodeDeploy Agent Operations (p. 205).

3. Check whether the application and deployment group that you created in AWS CodeDeploy for the deployment contains the settings that you expect:
   - To check your application settings, see View Application Details (p. 130).
   - To check your deployment group settings, see View Deployment Group Details (p. 131).
4. Confirm that the application revision that you want to deploy through AWS CodeDeploy is properly configured and contains the settings that you expect:
   - To check your AppSpec file's format, see Add an AppSpec File (p. 115) and AppSpec File Reference (p. 212).
   - Check your Amazon S3 bucket or GitHub repository to verify that your application revision is in the expected location.
   - To make sure that your application revision is correctly registered with AWS CodeDeploy, see View Application Revision Details (p. 132).
   - If you're deploying from Amazon S3, check your Amazon S3 bucket to verify that AWS CodeDeploy has been granted the correct permissions for downloading the application revision. For more information, see the discussion about Amazon S3 bucket policies in Deploy a Revision (p. 121).
   - If you're deploying from GitHub, check your GitHub repository to verify that AWS CodeDeploy has been granted the correct permissions for downloading the application revision. For more information, see the discussion about GitHub in Deploy a Revision (p. 121) as well as GitHub Authentication Behaviors with Applications in AWS CodeDeploy (p. 178).
5. Check whether the service role that you're using is properly configured. For more information, see Create a Service Role (p. 140).
6. Confirm that you followed the steps in Setting Up (p. 4), especially the steps that describe how to attach the necessary policies to the IAM user that is calling AWS CodeDeploy; install or upgrade, and configure, the AWS CLI; and create an IAM instance profile and a service role. See also Access Permissions Reference (p. 226).
7. Confirm that you are using AWS CLI version 1.6.1 or later. To check the version that you have installed, call **aws --version**.

If you are still unable to troubleshoot your failed deployment, explore the other issues that are described in this topic.

# AWS CodeDeploy Deployment Resources Are Supported in Only Certain Regions

If you do not see or cannot access specific applications, deployment groups, instances, or other deployment resources from either the AWS CLI or the AWS CodeDeploy console, and you expect to see them, make sure that you're referencing one of the supported regions.

AWS CodeDeploy is currently supported in only the following regions:

- US East (N. Virginia) region (us-east-1)
- US West (Oregon) region (us-west-2)
- EU (Ireland) region (eu-west-1)
- Asia Pacific (Sydney) region (ap-southeast-2)

If you're launching Amazon EC2 instances, and you want them to participate in AWS CodeDeploy deployments, launch the Amazon EC2 instances in one of the supported regions.

If you're creating Auto Scaling groups, and you want them to participate in AWS CodeDeploy deployments, create the Auto Scaling groups in one of the supported regions.

If you're using the AWS CLI, the best way to ensure that you're consistently using one of the supported regions is to set it as the default region. To do this, run the `aws configure` command from the AWS CLI. Then you can interactively view and set your default region, and this choice will remain in effect until you change it again.

If you're using the AWS CodeDeploy console, on the navigation bar, in the region selector, select one of the supported regions.

# Required IAM Roles Are Not Available

You may rely on a specific IAM role (such as an IAM instance profile or a service role) in conjunction with AWS CodeDeploy, and that role was created as part of an AWS CloudFormation stack. After you delete the stack, the IAM role will no longer be visible in the IAM console, and AWS CodeDeploy may no longer work as expected. This is because when you delete an AWS CloudFormation stack, all IAM roles that the stack created are deleted as well. To fix this problem, you must manually recreate the deleted IAM role.

# Using Certain Text Editors with AppSpec Files and Shell Scripts May Cause Associated Deployments to Fail

If you use certain text editors to create or modify AppSpec files, or if you use certain text editors to create or modify shell script files to run on Amazon Linux or Ubuntu Server instances, then any deployments that rely on these AppSpec files and shell script files may fail. This is because certain text editors can automatically or randomly introduce certain non-conforming, non-printing characters into these files. When AWS CodeDeploy uses these types of files during a deployment, these types of characters can result in hard-to-troubleshoot AppSpec file validation failures and script execution failures.

Although there is not an exact approach to determining whether any non-conforming, non-printing characters exist in a failed deployment's AppSpec file or shell script files, one approach to identifying this issue is to use the AWS CodeDeploy console: on the event details page for the deployment, click **View Logs**. (Alternatively, you use the AWS CLI to call the **get-deployment-instance** command.) There should be errors that typically contain phrases such as "invalid character," "command not found," or "file not found."

To address this issue, we recommend the following:

- Do not use text editors that automatically or randomly introduce non-printing characters such as carriage returns (`^M` characters) into your AppSpec files and shell script files.
- Use text editors that display non-printing characters such as carriage returns (`^M` characters) in your AppSpec files and shell script files, so that you can identify and remove any that may be automatically or randomly introduced. For examples of these types of text editors, search the Internet for "text editor show carriage returns."
- Use text editors running on Amazon Linux or Ubuntu Server instances to create shell script files that are designed to run on Amazon Linux or Ubuntu Server instances. For examples of these types of text editors, search the Internet for "Linux shell script editor" or "Ubuntu shell script editor."
- If you must use a text editor in Windows or MacOS to create shell script files to run on Amazon Linux or Ubuntu Server instances, use a program or utility that converts text in Windows or MacOS format to Unix format. For examples of such programs and utilities, search the Internet for "DOS to UNIX" or "Mac to UNIX." After conversion, you should test to make sure that the converted shell script files run in the way that you expect on the target operating systems.

# Using Finder in MacOS to Manually Bundle an Application Revision May Cause Associated Deployments to Fail

If you use the Finder graphical user interface (GUI) application on a Mac to manually bundle (zip) an AppSpec file and related files and scripts into an application revision archive (.zip) file, and then you try to deploy that .zip file, the deployment may fail. This is because Finder creates an intermediate `__MACOSX` folder within the .zip file and then places the component files into this intermediate folder. Because AWS CodeDeploy does not expect this intermediate folder to exist within the .zip file, it cannot find any of the component files, and this results in AWS CodeDeploy failing the deployment.

To address this issue, we recommend that you use the AWS CLI to call the **push** command, which automatically zips the component files in the layout that AWS CodeDeploy expects. Alternatively, you can use Terminal instead of the GUI to manually zip the component files; using Terminal does not create an intermediate `__MACOSX` folder.

# Troubleshooting Deployment Issues

**Topics**



## Troubleshooting a Failed ApplicationStop Deployment Lifecycle Event

A deployment may fail during the **ApplicationStop** deployment lifecycle event for one of the following reasons:

- The AWS CodeDeploy Agent finds the corresponding *`deployment-group-id`*`_last_successful_install` file in the correct location; however, the location that is listed in the *`deployment-group-id`*`_last_successful_install` file does not exist. (Note that this file must exist within the `/opt/codedeploy-agent/deployment-root/deployment-instructions` on Amazon Linux and Ubuntu Server instances or within the `C:\ProgramData\Amazon\CodeDeploy\deployment-instructions` folder on Windows Server instances.)
- In the location that is listed in the *`deployment-group-id`*`_last_successful_install` file, either the AppSpec file is invalid, or the scripts fail to run successfully during the **ApplicationStop** deployment lifecycle event.

To investigate why a deployment may have failed during this event, you can use the AWS CodeDeploy console: on the event details page for the deployment, in the **ApplicationStop** row, click **View Logs**. Alternatively, you use the AWS CLI to call the **get-deployment-instance** command.

To recover from a deployment that failed during the **ApplicationStop** deployment lifecycle event, use the AWS CodeDeploy to deploy the application revision again by calling the **create-deployment** command.

(You cannot use the AWS CodeDeploy console to try to recover from deploying an application revision that fails during the **ApplicationStop** deployment lifecycle event.) This time, when you call the **create-deployment** command, set the `--ignore-application-stop-failures` option. During this deployment, the **ApplicationStop** deployment lifecycle event will still execute; but this time, the deployment will still continue even if the **ApplicationStop** deployment lifecycle event fails again.

# Troubleshooting a Failed DownloadBundle Deployment Lifecycle Event with "UnknownError: Not Opened for Reading"

If you are trying to deploy an application revision from Amazon S3, and the deployment fails during the **DownloadBundle** deployment lifecycle event with the error "UnknownError: not opened for reading," this can happen for one of the following reasons:

- There was some internal error with the Amazon S3 service. To address this issue, simply try to deploy the application revision again.
- The IAM instance profile on your Amazon EC2 instance does not have the correct permissions to access the application revision in Amazon S3. To address this issue, make sure that the IAM instance profile has the correct permissions to access and deploy the application revision from the associated Amazon S3 bucket. For more information, see the discussion about Amazon S3 bucket policies in Push a Revision (p. 118) and Deploy a Revision (p. 121).
- The instances that you want to deploy to are associated with one region (for example, US West (Oregon) (us-west-2)), and you are trying to deploy an application revision from an Amazon S3 bucket that is associated with another region (for example, US East (N. Virginia) (us-east-1)). To address this issue, make sure that the application revision is in an Amazon S3 bucket that is associated with the same region as the instances that you are trying to deploy to.

To confirm whether this particular error has occurred, on the event details page for the deployment, in the **DownloadBundle** row, click **View Logs**. (Alternatively, you use the AWS CLI to call the **get-deployment-instance** command.) There should be an error that contains the error code "UnknownError" and the error message "not opened for reading."

To determine for which particular reason this error may have occurred, do the following:

1. If you do not have wire logging enabled, enable it on at least one of the instances that you are trying to deploy to, and then try deploying the application revision again.
2. If you have wire logging enabled, then after the deployment fails, examine the wire logging file to determine the specific error. Common error messages for this type of issue include the phrase "access denied."
3. After you have examined the log files, we recommend that you disable wire logging to both reduce log file size and to reduce the amount of sensitive information that may be output as plain text on the instance in the future.

To learn how to enable and disable wire logging as well as to learn how to find the wire logging file, see the discussion about **:log_aws_wire:** in AWS CodeDeploy Agent (p. 67).

# Windows PowerShell Scripts Fail to Use the 64-Bit Version of Windows PowerShell by Default

When a Windows PowerShell script runs as part of a deployment, and the script relies on 64-bit functionality to work (for example, consuming more memory than a 32-bit application will allow, or calling libraries that are only offered in a 64-bit version), the script may not run as expected or may crash. This is because

AWS CodeDeploy uses the 32-bit version of Windows PowerShell by default to run Windows PowerShell scripts that are part of an application revision. To address this issue, add code similar to the following to the beginning of any script that needs to run with the 64-bit version of Windows PowerShell:

```
# Are you running in 32-bit mode?
#   (\SysWOW64\ = 32-bit mode)

if ($PSHOME -like "*SysWOW64*")
{
  Write-Warning "Restarting this script under 64-bit Windows PowerShell."

  # Restart this script under 64-bit Windows PowerShell.
  #   (\SysNative\ redirects to \System32\ for 64-bit mode)

  & (Join-Path ($PSHOME -replace "SysWOW64", "SysNative") powershell.exe) -File
  `
    (Join-Path $PSScriptRoot $MyInvocation.MyCommand) @args

  # Exit 32-bit script.

  Exit
}

# Was restart successful?
Write-Warning "Hello from $PSHOME"
Write-Warning "  (\SysWOW64\ = 32-bit mode, \System32\ = 64-bit mode)"
Write-Warning "Original arguments (if any): $args"

# Your 64-bit script code follows here...
# ...
```

Although the file path information in the preceding code may seem counterintuitive, 32-bit Windows PowerShell uses a path such as `c:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe`, while 64-bit Windows PowerShell uses a path such as `c:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe`.

# Long-Running Processes Can Cause Deployments to Fail

For deployments to Amazon Linux and Ubuntu Server instances, when you have a deployment script that starts a long-running process, AWS CodeDeploy may spend a long time waiting in the associated deployment lifecycle event and then stop and fail the deployment. This is because if the long-running process keeps running for a long enough time, it could keep running past the deadline for when AWS CodeDeploy expects all of the foreground and background processes in that event to stop running. Once that deadline is passed, AWS CodeDeploy stops and fails the deployment, even though the long-running process may still be running as expected.

To demonstrate, for example, let's pretend that an application revision contains two files in its root, `after-install.sh` and `sleep.sh`, and its AppSpec file contains the following instructions:

```
version: 0.0
os: linux
files:
  - source: ./sleep.sh
```

```
    destination: /tmp
hooks:
  AfterInstall:
    - location: after-install.sh
      timeout: 60
```

The file named `after-install.sh` runs during the **AfterInstall** application lifecycle event, with the contents of the `after-install.sh` file being the following:

```
#!/bin/bash
/tmp/sleep.sh
```

The file named `sleep.sh` file contains the following, which suspends program execution for 3 minutes (180 seconds), simulating some long-running process:

```
#!/bin/bash
sleep 180
```

In the preceding example, when `after-install.sh` calls `sleep.sh`, `sleep.sh` will start running and then keep running for 3 minutes (180 seconds), which is 2 minutes (120 seconds) past the deadline for when AWS CodeDeploy expects `sleep.sh` (and, by relation, `after-install.sh`) to stop running. Once the deadline of 1 minute (60 seconds) passes, AWS CodeDeploy stops and fails the deployment at the **AfterInstall** application lifecycle event, even though `sleep.sh` successfully continues to run as expected, and the following error displays: `Script at specified location: after-install.sh failed to complete in 60 seconds`.

You cannot address this issue simply by adding an ampersand (`&`) character after `sleep.sh` in `after-install.sh` to try running `sleep.sh` in the background, as follows:

```
#!/bin/bash
# Do not do this.
/tmp/sleep.sh &
```

Doing so can leave the deployment in a pending state for up to the default one-hour deployment lifecycle event timeout period, after which AWS CodeDeploy stops and fails the deployment at the **AfterInstall** application lifecycle event as before.

To properly address this issue, in `after-install.sh`, you should call `sleep.sh` as follows, which enables AWS CodeDeploy to continue on after the process starts running:

```
#!/bin/bash
/tmp/sleep.sh > /dev/null 2> /dev/null < /dev/null &
```

In the preceding call, `sleep.sh` is the name of the process that you want to start running in the background, redirecting stdout, stderr, and stdin to `/dev/null`.

# Troubleshooting Deployment Group Issues

## Tagging an Instance to Be Part Of an Existing Deployment Group Does Not Automatically Deploy Your Application to the New Instance

After you tag an instance so that it can be part of an existing deployment group, AWS CodeDeploy does not automatically deploy your application to the newly-tagged instance. To have AWS CodeDeploy deploy your application to the newly-tagged instance, create a new deployment to the existing deployment group.

You can use AWS CodeDeploy to enable automatic deployments to new Amazon EC2 instances in Auto Scaling groups. For more information, see Auto Scaling Integration (p. 157).

# Troubleshooting Instance Issues

**Topics**



## Tags Must Be Set Correctly

Verify that your instances are tagged correctly. You can use the **list-deployment-instances** command to confirm the instances that are being used for a specific deployment. If any Amazon EC2 is missing in the output, but you expect it to be listed, you can use the Amazon EC2 console to confirm the tags that have been set on those Amazon EC2 instances. (For more information, see Working with Tags in the Console.)

> **Note**
> If you tag an instance and immediately use AWS CodeDeploy to deploy an application to it, the instance may not be included in the deployment. This is because it may take several minutes for the tags to be visible to AWS CodeDeploy. We recommend waiting at least 5 minutes between the time that you tag an instance and try deploying to it.

# The AWS CodeDeploy Agent for Amazon Linux Must Be Installed and Running on Amazon Linux Instances

To verify whether the AWS CodeDeploy Agent is installed and running on an instance, see Verify That the AWS CodeDeploy Agent for Amazon Linux is Running (p. 206).

# Installing, Uninstalling, or Reinstalling the AWS CodeDeploy Agent for Amazon Linux

To install, uninstall, or reinstall the AWS CodeDeploy Agent, see Install, Uninstall, or Reinstall the AWS CodeDeploy Agent for Amazon Linux (p. 208).

# The AWS CodeDeploy Agent for Ubuntu Server Must Be Installed and Running on Ubuntu Server Instances

To verify whether the AWS CodeDeploy Agent is installed and running on an instance, see Verify That the AWS CodeDeploy Agent for Ubuntu Server is Running (p. 207).

# Installing, Uninstalling, or Reinstalling the AWS CodeDeploy Agent for Ubuntu Server

To install, uninstall, or reinstall the AWS CodeDeploy Agent, see Install, Uninstall, or Reinstall the AWS CodeDeploy Agent for Ubuntu Server (p. 209).

# The AWS CodeDeploy Agent for Windows Server Must Be Installed and Running on Windows Server Instances

To verify whether the AWS CodeDeploy Agent is installed and running on an instance, see Verify That the AWS CodeDeploy Agent for Windows Server is Running (p. 207).

# Installing, Uninstalling, or Reinstalling the AWS CodeDeploy Agent for Windows Server

To install, uninstall, or reinstall the AWS CodeDeploy Agent, see Install, Uninstall, or Reinstall the AWS CodeDeploy Agent for Windows Server (p. 210).

# Investigating Deployment Failures on Instances

If a deployment fails, you can investigate the instances that participated in the deployment, by using techniques such as the following:

- View the status of each instance in the deployment. Specifically, look for any status other than `Succeeded` to determine which instances failed to be deployed to. For instructions, see View Instance Details (p. 128).
- Analyze the deployment log file on an instance that failed to be deployed to. To do this, sign in to the instance that failed to be deployed to, and then do one of the following:
  - For an Amazon Linux or Ubuntu Server instance, type the following command to open the AWS CodeDeploy Agent log file:

```
less /var/log/aws/codedeploy-agent/codedeploy-agent.log
```

    Type the following commands to browse the log file for error messages:
    - Type **`& ERROR`** (notice the single space before and after the word **ERROR**) to show just the error messages in the log file.

      Type **`/ ERROR`** (notice the single space before and after the word **ERROR**) to search for the next error message, or type **`? ERROR`** (notice the single space before and after the word **ERROR**) to search for the previous error message.

      After you type **`/ ERROR`**, type **n** for the next error message forward, or type **`N`** for the previous error message backward.

      After you type **`? ERROR`**, type **n** for the next error message backward, or type **`N`** for the previous error message forward.

      Type **`G`** to go to the end of the log file, or type **g** to go to the start.

      Type **q** to exit the log file.

      Type **h** to learn about additional commands that you can use.

    You can also type the following command to open an AWS CodeDeploy scripts log file:

```
less /opt/codedeploy-agent/deployment-root/deployment-group-ID/deployment-ID/logs/scripts.log
```

    Type the following commands to browse the log file for error messages:
    - Type **`/stderr`** to show just the error messages in the log file.

      Type **`/stderr`** to search for the next error message, or type **`?stderr`** to search for the previous error message.

      After you type **`/stderr`**, type **n** for the next error message forward, or type **`N`** for the previous error message backward.

      After you type **`? ERROR`**, type **n** for the next error message backward, or type **`N`** for the previous error message forward.

      Type **`G`** to go to the end of the log file, or type **g** to go to the start.

      Type **q** to exit the log file.

      Type **h** to learn about additional commands that you can use.
  - For a Windows Server instance, type the following command to open the AWS CodeDeploy Agent log file:

```
notepad C:\ProgramData\Amazon\CodeDeploy\log\codedeploy-agent-log.txt
```

To browse the log file for error messages, press CTRL+F, type **ERROR [**, and then press Enter to find the first error. Click **Find Next** to find the next error, and so on.

You can also type the following command to open an AWS CodeDeploy scripts log file:

```
notepad C:\ProgramData\Amazon\CodeDeploy\deployment-group-ID\deployment-ID\logs\scripts.log
```

To browse the log file for error messages, press CTRL+F, type **stderr**, and then press Enter to find the first error. Click **Find Next** to find the next error, and so on.

# Create a New AWS CodeDeploy Log File If It Was Accidentally Deleted

If you accidentally delete the deployment log file on an instance, AWS CodeDeploy does not automatically create a replacement log file. To create a new log file, sign in to the instance, and then run these commands:

For an Amazon Linux or Ubuntu Server instance, run these commands in this order, one at a time:

```
sudo service codedeploy-agent stop
sudo service codedeploy-agent
```

For a Windows Server instance:

```
powershell.exe -Command Restart-Service -Name codedeployagent
```

# Deploying or Redeploying the Same Files to the Same Locations on Instances May Fail Under Certain Conditions

By default, if AWS CodeDeploy tries to copy files to specific locations onto an instance, and any one of those files already exists in the specified location, the deployment will stop for that instance, and the deployment will be considered to have failed for that instance. Also, if you try to redeploy files with the same names and locations onto instances, and you specify the application name and the deployment group with the *same* underlying deployment group ID that you used before, then the redeployment should have a better probability of succeeding. This is because AWS CodeDeploy uses the underlying deployment group ID to try to identify files to forcibly remove before a redeployment. Because of these behaviors, deploying new files or redeploying the same files to the same locations on instances may fail for seemingly unanticipated reasons; for example:

- *You specify a different application name in AWS CodeDeploy for a redeployment of the same revision to the same instances.* The redeployment will fail because redeploying under a different application name uses a different underlying deployment group ID, even if the deployment group name is the same under both the current and other application names in AWS CodeDeploy.
- *You delete a deployment group in AWS CodeDeploy for a given application; under that same application you recreate a new deployment group with the same name as the deployment group that you just deleted; and then you try to redeploy the same revision to that deployment group.* The redeployment will fail because AWS CodeDeploy will reference a different underlying deployment group ID for the new deployment group, even if the deployment group name is the same as the one before.

- *You delete an application in AWS CodeDeploy; you recreate a new application and deployment group with the same name as the application and deployment group that you just deleted; and then you try to redeploy the same revision to that deployment group.* The redeployment will fail because AWS CodeDeploy will reference a different underlying deployment group IDs for the new deployment group, even if the application and deployment group names are the same.
- *You deploy a revision to one deployment group in AWS CodeDeploy; and then you deploy the exact same revision to another deployment group in AWS CodeDeploy to the same instances.* The second deployment will fail because AWS CodeDeploy will refer to a different underlying deployment group ID.
- *You deploy a revision to one deployment group in AWS CodeDeploy; and then you deploy a different revision in AWS CodeDeploy to the same instances; but there is at least one file with the same name and in the same location that the second deployment tries to deploy.* The second deployment will fail because AWS CodeDeploy will not remove the existing file before the second deployment starts, as both deployments will reference different deployment group IDs.
- *You deploy a revision in AWS CodeDeploy; but there is at least one file with the same name and in the same location that exists for some other reason.* The deployment will fail because by default AWS CodeDeploy will not forcibly remove the existing file before the deployment starts.

To address these situations, do one of the following:

- Manually remove the files from the locations that were previously deployed to each of the instances, and then try the deployment again.
- In your revision's AppSpec file, in either the **ApplicationStop** or **BeforeInstall** deployment lifecycle events, specify a custom script to run that forcibly deletes any files in any of the locations that correspond to the files that your revision is about to install.
- Deploy or redeploy the files to locations or instances that were not part of previous deployments.
- Before you delete an application or a deployment group, deploy a revision that contains an AppSpec file that specifies no files to copy to the instances. For the deployment, specify the application name and deployment group name that use the *same* underlying application and deployment group IDs as the ones that you are about to delete. AWS CodeDeploy will use the underlying deployment group ID and AppSpec file to try to forcibly remove all of the files that it installed in the previous successful deployment.

# Troubleshooting Auto Scaling Issues

**Topics**



## General Auto Scaling Troubleshooting

Deployments to Amazon EC2 instances in an Auto Scaling group may generally fail for the following reasons:

- **Auto Scaling continuously launches and terminates Amazon EC2 instances.** If AWS CodeDeploy cannot automatically deploy your application revision, Auto Scaling will continuously launch and terminate Amazon EC2 instances. One of the ways to address this issue includes:

  - Manually stop Auto Scaling from launching any more Amazon EC2 instances either by disassociating the Auto Scaling group from the corresponding AWS CodeDeploy deployment group, or by changing your Auto Scaling group's configuration so that the desired number of instances matches the current number of instances (thus preventing Auto Scaling from launching any more Amazon EC2 instances). To learn how to do this, see Change Deployment Group Settings (p. 144) or Configuring Your Auto Scaling Groups.
- **The AWS CodeDeploy Agent is unresponsive.** If any initialization scripts (for example, cloud-init scripts) that you run immediately after an Amazon EC2 instance launches or starts take more than 1 hour, then the AWS CodeDeploy Agent may not install properly or not install at all. This is because AWS CodeDeploy currently has a 1-hour timeout for the AWS CodeDeploy Agent to begin responding to any pending deployments. To address this issue, move your initialization scripts into your AWS CodeDeploy application revision.
- **An Amazon EC2 instance in an Auto Scaling group reboots during a deployment.** Rebooting an Amazon EC2 instance during a deployment, or shutting down the AWS CodeDeploy Agent while it is processing a deployment command, can cause your deployment to fail. To address this issue, make sure that Amazon EC2 instances do not reboot during deployments or otherwise shut down the AWS CodeDeploy Agent during deployments. For more information, see Terminating or Rebooting an Auto Scaling Instance May Cause Associated Deployments to Fail (p. 203).
- **Multiple application revisions are deployed simultaneously to the same Amazon EC2 instance in an Auto Scaling group.** Deploying multiple application revisions to the same Amazon EC2 instance in an Auto Scaling group at the same time can fail if one of the deployments has scripts that run for more than a few minutes. To address this issue, do not deploy multiple application revisions to the same Amazon EC2 instances in an Auto Scaling group.
- **A deployment fails for new Amazon EC2 instances that are launched as part of an Auto Scaling group.** Typically in this scenario, various issues with running the scripts in a deployment can cause no more Amazon EC2 instances to be launched in the corresponding Auto Scaling group (although existing Amazon EC2 instances in the Auto Scaling group may appear to be running normally). To address this issue, fix any scripts in a deployment that may fail to run as expected.

# Terminating or Rebooting an Auto Scaling Instance May Cause Associated Deployments to Fail

If an Amazon EC2 instance is launched through Auto Scaling, and then that instance terminates or reboots, any associated deployments to that instance may fail due to the following reasons:

- During an in-progress deployment, a scale-in event—or any other termination event—will cause the instance to detach from the Auto Scaling group and then terminate. Because the deployment cannot complete, the deployment therefore fails.
- The instance reboots, but the elapsed time between the previous and current instance starts is greater than 5 minutes. AWS CodeDeploy considers this to be a time-out situation. Because of this, the service will fail all current and future deployments to the instance.

To address this issue:

- In general, make sure that all deployments finish before the associated instance begins terminating or rebooting. Also, make sure that all deployments start after the associated instance finishes starting or rebooting.
- If you specify a Windows Server base Amazon Machine Image (AMI) for an Auto Scaling configuration, and you use the EC2Config service to set the instance's computer name, this behavior can cause deployments to fail. To disable this behavior, in the Windows Server base AMI, in the **Ec2 Service Properties** dialog box's **General** tab, uncheck the **Set Computer Name** box. Note that after you uncheck this box, this behavior will then be disabled for all new Windows Server Auto Scaling instances that are launched with that Windows Server base AMI. For existing Windows Server Auto Scaling

instances that have this behavior enabled, you do not need to uncheck this box; simply redeploy any failed deployments to those instances after they have finished rebooting.

# Amazon EC2 Instances in an Auto Scaling Group Fail to Launch with the Error "Heartbeat Timeout"

An Auto Scaling group may fail to launch new Amazon EC2 instances, generating a message similar to the following: `Launching a new EC2 instance <instance-Id>. Status Reason: Instance failed to complete user's Lifecycle Action: Lifecycle Action with token<token-Id> was abandoned: Heartbeat Timeout`. This is typically caused by an application being deleted in AWS CodeDeploy before its associated deployment groups are updated or deleted, and those deployment groups contain references to one or more Auto Scaling groups with stray Auto Scaling lifecycle hooks. To determine whether this is the cause, you can examine the output of a call to either the **update-deployment-group** or **delete-deployment-group** command for the associated deployment group. The output should contain a `hooksNotCleanedUp` structure containing a list of stray Auto Scaling lifecycle hooks that are likely causing the problem.

To address this issue, do the following:

1. Call the **describe-lifecycle-hooks** command, specifying the name of the Auto Scaling group that is associated with the Amazon EC2 instances that fail to launch. In the list of Auto Scaling lifecycle hooks that are output, look for any Auto Scaling lifecycle hook names that correspond to the `hooksNotCleanedUp` structure as output from a call to the **update-deploy-group** or **delete-deployment-group** commands. (Alternatively, you can look for any Auto Scaling lifecycle hooks names that contain the corresponding deployment group's name.)
2. For each stray Auto Scaling lifecycle hook, call the **delete-lifecycle-hook** command, specifying the name of the Auto Scaling group and the name of the stray Auto Scaling lifecycle hook.

Note that if you delete from an Auto Scaling group all of the Auto Scaling lifecycle hooks that were created by AWS CodeDeploy, then AWS CodeDeploy will no longer automatically deploy to Amazon EC2 instances that are scaled up as part of that Auto Scaling group.

# Mismatched Auto Scaling Lifecycle Hooks May Cause Automatic Deployments to Auto Scaling Groups to Stop or Fail

Auto Scaling and AWS CodeDeploy use lifecycle hooks to determine which application revisions should be automatically deployed to which Amazon EC2 instances after they are newly launched in Auto Scaling groups. Automatic deployments can stop or fail if specific lifecycle hooks, and information about these hooks, do not match exactly in both Auto Scaling and AWS CodeDeploy.

If deployments to a specific Auto Scaling group are not succeeding as expected, one troubleshooting approach is to see if both Auto Scaling and AWS CodeDeploy agree on the Auto Scaling lifecycle hook names that are being used for the associated deployments. If they do not match, you can get them to match with just a few AWS CLI command calls.

First, get the list of lifecycle hook names for both the Auto Scaling group and the deployment group by doing the following:

1. Call the **describe-lifecycle-hooks** command, specifying the name of the Auto Scaling group that is associated with the deployment group in AWS CodeDeploy. In the output, within the `LifecycleHooks` list, make a note of each `LifecycleHookName` value.

2. Then call the **get-deployment-group** command, specifying the name of the deployment group that is associated with the Auto Scaling group. In the output, within the `autoScalingGroups` list, find each item whose name value matches the Auto Scaling group name, and then make a note of the corresponding `hook` value.

Now compare the two sets of lifecycle hook names. If they match exactly, character for character, then this is not the issue, and you may want to try other Auto Scaling troubleshooting steps that are described elsewhere in this section.

However, if the two sets of lifecycle hook names do not match exactly, character for character, do one or both of the following sets of steps to get them to match:

1. If there are lifecycle hook names that are in the **describe-lifecycle-hooks** command output but are not also in the **get-deployment-group** command output, then do the following:

    1. For each of the lifecycle hook names that are only in the **describe-lifecycle-hooks** command output, call the **delete-lifecycle-hook** command.
    2. Then call the **update-deployment-group** command, specifying the name of the original Auto Scaling group. AWS CodeDeploy will create new, replacement lifecycle hooks in the Auto Scaling group, and will associate the newly-created lifecycle hooks with the deployment group, so that automatic deployments should now resume as new instances are added to the Auto Scaling group.

2. If there are lifecycle hook names that are in the **get-deployment-group** command output but are not also in the **describe-lifecycle-hooks** command output, then do the following:

    1. Call the **update-deployment-group** command, but do not specify the name of the original Auto Scaling group.
    2. Then call the **update-deployment-group** command again, but this time specify the name of the original Auto Scaling group. AWS CodeDeploy will recreate the missing lifecycle hooks in the Auto Scaling group so that automatic deployments should now resume as new instances are added to the Auto Scaling group.

Note that once you get the two sets of lifecycle hook names to match exactly, character for character, application revisions should begin to automatically deploy again but only to new instances as they are added to the Auto Scaling group. However, deployments will not automatically occur to instances that may already exist in the Auto Scaling group.

# AWS CodeDeploy Agent Operations

These instructions show you how to perform various functions with the AWS CodeDeploy Agent, such as verifying that the AWS CodeDeploy Agent is running, and installing, uninstalling, reinstalling, or updating the AWS CodeDeploy Agent.

**Topics**

# Operating Systems Supported by the AWS CodeDeploy Agent

The AWS CodeDeploy Agent has been tested on the following Amazon EC2 AMI operating systems:

- Amazon Linux 2014.09.1, 2015.03
- Ubuntu Server 14.04 LTS
- Windows Server 2008 R2 and 2012 R2

To use the AWS CodeDeploy Agent with other Amazon EC2 AMI operating systems, the AWS CodeDeploy Agent is available as open source for you to adapt to your needs. For more information, see the AWS CodeDeploy Agent repository in GitHub.

The AWS CodeDeploy Agent has been tested on the following on-premises instance operating systems:

- Ubuntu Server 14.04 LTS
- Windows Server 2008 R2 and 2012 R2

To use the AWS CodeDeploy Agent with other on-premises instance operating systems, the AWS CodeDeploy Agent is available as open source for you to adapt to your needs. For more information, see the AWS CodeDeploy Agent repository in GitHub.

# Communication Protocol and Port for the AWS CodeDeploy Agent

The AWS CodeDeploy Agent communicates outbound using HTTPS over port 443.

# Verify That the AWS CodeDeploy Agent for Amazon Linux is Running

To see if the AWS CodeDeploy Agent is installed and running, sign in to the instance, and run the following command:

```
sudo service codedeploy-agent status
```

If the command returns an error, the AWS CodeDeploy Agent is not installed. Install it as described in Install, Uninstall, or Reinstall the AWS CodeDeploy Agent for Amazon Linux (p. 208).

If the AWS CodeDeploy Agent is installed and running, you should see a message such as `The AWS CodeDeploy agent is running`.

If you see a message such as `error: No AWS CodeDeploy agent running`, you can start the service and verify that it is running with the following two commands, one at a time:

```
sudo service codedeploy-agent start
sudo service codedeploy-agent status
```

# Verify That the AWS CodeDeploy Agent for Ubuntu Server is Running

To see if the AWS CodeDeploy Agent is installed and running, sign in to the instance, and run the following command:

```
sudo service codedeploy-agent status
```

If the command returns an error, the AWS CodeDeploy Agent is not installed. Install it as described in Install, Uninstall, or Reinstall the AWS CodeDeploy Agent for Ubuntu Server (p. 209).

If the AWS CodeDeploy Agent is installed and running, you should see a message such as `The AWS CodeDeploy agent is running`.

If you see a message such as `error: No AWS CodeDeploy agent running`, you can start the service and verify that it is running with the following two commands, one at a time:

```
sudo service codedeploy-agent start
sudo service codedeploy-agent status
```

# Verify That the AWS CodeDeploy Agent for Windows Server is Running

To see if the AWS CodeDeploy Agent is installed and running, sign in to the instance, and run the following command:

```
powershell.exe -Command Get-Service -Name codedeployagent
```

You should see output similar to the following:

```
Status   Name               DisplayName
------   ----               -----------
Running codedeployagent    CodeDeploy Host Agent Service
```

If the command returns an error, the AWS CodeDeploy Agent is not installed. Install it as described in Install, Uninstall, or Reinstall the AWS CodeDeploy Agent for Windows Server (p. 210).

If `Status` shows anything other than `Running`, you can start the service with the following command:

```
powershell.exe -Command Start-Service -Name codedeployagent
```

If needed, you can restart the service with the following command:

```
powershell.exe -Command Restart-Service -Name codedeployagent
```

And if needed, you can stop the service with the following command:

```
powershell.exe -Command Stop-Service -Name codedeployagent
```

# Install, Uninstall, or Reinstall the AWS CodeDeploy Agent for Amazon Linux

To determine the version of the AWS CodeDeploy Agent that's installed on an instance, sign in to the instance and run the following command:

```
sudo yum info codedeploy-agent
```

If you suspect that the AWS CodeDeploy Agent is missing or broken on an instance, you can install or reinstall it by signing in to the instance and then running the following commands, one at a time:

1. 
```
sudo yum update
sudo yum install aws-cli
cd /home/ec2-user
aws s3 cp s3://bucket-name/latest/install . --region region-name
chmod +x ./install
sudo ./install auto
```

   (Type **y** if prompted, and do not forget to include the period (.) character in the **aws s3 cp** command here.)

   In the previous list of commands, *`bucket-name`* represents one of the following:

   - `aws-codedeploy-us-east-1` for instances in the US East (N. Virginia) region
   - `aws-codedeploy-us-west-2` for instances in the US West (Oregon) region
   - `aws-codedeploy-eu-west-1` for instances in the EU (Ireland) region
   - `aws-codedeploy-ap-southeast-2` for instances in the Asia Pacific (Sydney) region

   In the previous list of commands, *`region-name`* represents one of the following:

   - `us-east-1` for instances in the US East (N. Virginia) region
   - `us-west-2` for instances in the US West (Oregon) region
   - `eu-west-1` for instances in the EU (Ireland) region
   - `ap-southeast-2` for instances in the Asia Pacific (Sydney) region

2. 
```
sudo service codedeploy-agent status
```

   If the AWS CodeDeploy Agent is installed and running, you should see a message such as `The AWS CodeDeploy agent is running`.

   If you see a message such as `error: No AWS CodeDeploy agent running`, you can start the service and verify that it is running with the following two commands, one at a time:

```
sudo service codedeploy-agent start
sudo service codedeploy-agent status
```

To uninstall the AWS CodeDeploy Agent, sign in to the instance and run the following command:

```
sudo yum erase codedeploy-agent
```

# Install, Uninstall, or Reinstall the AWS CodeDeploy Agent for Ubuntu Server

To determine the version of the AWS CodeDeploy Agent that's installed on an instance, sign in to the instance and run the following command:

```
sudo dpkg -s codedeploy-agent
```

If you suspect that the AWS CodeDeploy Agent is missing or broken on an instance, you can install or reinstall it by signing in to the instance and then running the following commands, one at a time:

1. 
```
sudo apt-get update
sudo apt-get install awscli
sudo apt-get install ruby2.0
cd /home/ubuntu
sudo aws s3 cp s3://bucket-name/latest/install . --region region-name
sudo chmod +x ./install
sudo ./install auto
```

   (Type **y** if prompted, and do not forget to include the period (.) character in the **aws s3 cp** command here.)

   In the previous list of commands, *`bucket-name`* represents one of the following:

   - `aws-codedeploy-us-east-1` for instances in the US East (N. Virginia) region
   - `aws-codedeploy-us-west-2` for instances in the US West (Oregon) region
   - `aws-codedeploy-eu-west-1` for instances in the EU (Ireland) region
   - `aws-codedeploy-ap-southeast-2` for instances in the Asia Pacific (Sydney) region

   In the previous list of commands, *`region-name`* represents one of the following:

   - `us-east-1` for instances in the US East (N. Virginia) region
   - `us-west-2` for instances in the US West (Oregon) region
   - `eu-west-1` for instances in the EU (Ireland) region
   - `ap-southeast-2` for instances in the Asia Pacific (Sydney) region

2. 
```
sudo service codedeploy-agent status
```

   If the AWS CodeDeploy Agent is installed and running, you should see a message such as `The AWS CodeDeploy agent is running`.

If you see a message such as `error: No AWS CodeDeploy agent running`, you can start the service and verify that it is running with the following two commands, one at a time:

```
sudo service codedeploy-agent start
sudo service codedeploy-agent status
```

To uninstall the AWS CodeDeploy Agent, sign in to the instance and run the following command:

```
sudo dpkg -r codedeploy-agent
```

# Install, Uninstall, or Reinstall the AWS CodeDeploy Agent for Windows Server

To determine the version of the AWS CodeDeploy Agent that's installed on an instance, sign in to the instance and run the following command:

```
sc qdescription codedeployagent
```

If you suspect that the AWS CodeDeploy Agent is missing or broken on an instance, you can install or reinstall it by signing in to the instance and then running the following commands, one at a time:

```
if not exist "c:\temp" mkdir c:\temp
powershell.exe -Command Read-S3Object -BucketName bucket-name/latest -Key
codedeploy-agent.msi -File c:\temp\codedeploy-agent.msi
c:\temp\codedeploy-agent.msi /quiet /l c:\temp\host-agent-install-log.txt
powershell.exe -Command Get-Service -Name codedeployagent
```

In the previous list of commands, *`bucket-name`* represents one of the following:

- `aws-codedeploy-us-east-1` for instances in the US East (N. Virginia) region
- `aws-codedeploy-us-west-2` for instances in the US West (Oregon) region
- `aws-codedeploy-eu-west-1` for instances in the EU (Ireland) region
- `aws-codedeploy-ap-southeast-2` for instances in the Asia Pacific (Sydney) region

If the AWS CodeDeploy Agent is installed and running, you should see output similar to the following after the **Get-Service** command call:

```

Status   Name               DisplayName
------   ----               -----------
Running  codedeployagent    CodeDeploy Host Agent Service
```

To uninstall the AWS CodeDeploy Agent, sign in to the instance and run the following three commands, one at a time:

```
wmic
product where name="CodeDeploy Host Agent" call uninstall /nointeractive
exit
```

Or, sign in to the instance, open the **Programs and Features** applet in **Control Panel**, click the **CodeDeploy Host Agent** entry, and then click **Uninstall**.

# Update the AWS CodeDeploy Agent for Amazon Linux

After you install the AWS CodeDeploy Agent (`codedeploy-agent.noarch.rpm`) on an instance, if a new version is released, the AWS CodeDeploy Agent will automatically update itself to the new version within 24 hours of the new version being released. The update will happen at an arbitrarily-determined time within the 24-hour time period. The update time cannot easily be cancelled or rescheduled. If a deployment is in progress during the update, the current deployment lifecycle event will finish first. After the update finishes, the deployment will resume with the next deployment lifecycle event.

If you want to immediately force an update of the AWS CodeDeploy Agent (`codedeploy-agent.noarch.rpm`), sign in to the instance, and then run the following command:

```
sudo /opt/codedeploy-agent/bin/install auto
```

# Update the AWS CodeDeploy Agent for Ubuntu Server

After you install the AWS CodeDeploy Agent (`codedeploy-agent_all.deb`) on an instance, if a new version is released, the AWS CodeDeploy Agent will automatically update itself to the new version within 24 hours of the new version being released. The update will happen at an arbitrarily-determined time within the 24-hour time period. The update time cannot easily be cancelled or rescheduled. If a deployment is in progress during the update, the current deployment lifecycle event will finish first. After the update finishes, the deployment will resume with the next deployment lifecycle event.

If you want to immediately force an update of the AWS CodeDeploy Agent (`codedeploy-agent_all.deb`), sign in to the instance, and then run the following command:

```
sudo /opt/codedeploy-agent/bin/install auto
```

# Update the AWS CodeDeploy Agent for Windows Server

The AWS CodeDeploy Agent (`codedeploy-agent.msi`) currently does not automatically update itself if a new version is released. To update the AWS CodeDeploy Agent after it has been installed on an instance, follow the instructions in Install, Uninstall, or Reinstall the AWS CodeDeploy Agent for Windows Server (p. 210).

# AWS CodeDeploy AppSpec File Reference

The Application Specification File (AppSpec file) is a YAML-formatted file that AWS CodeDeploy uses to determine what it should install onto your instances from your application revision in Amazon S3 or GitHub, as well as what deployment lifecycle event hooks to run in response to various deployment lifecycle events.

An AppSpec file must be named `appspec.yml`, and it must be placed in the root of an application's source code's directory structure. AppSpec files that do not follow this requirement will cause associated deployments to fail.

After you have a completed AppSpec file, you bundle it, along with your deployable content, into an archive file of type zip, tar, or compressed tar. For more information, see Prepare a Revision (p. 114).

> **Note**
> The tar and compressed tar archive file formats (.tar and .tar.gz) are not supported for Windows Server instances.

After you have a bundled archive file (known in AWS CodeDeploy as a *revision*) ready, you upload it to an Amazon S3 bucket or Git repository of your choice. Then you use AWS CodeDeploy to deploy the revision from there. For instructions, see Deploy a Revision (p. 121).

> **Note**
> This topic serves as a reference only. For a conceptual overview of the AppSpec file, see AppSpec Files (p. 64).

**Topics**



## AppSpec file Structure

The AppSpec file has the following high-level structure:

```
version: 0.0
os: operating-system-name
files:
  source-destination-files-mappings
permissions:
  permissions-specifications
hooks:
  deployment-lifecycle-event-mappings
```

In this structure:

- The **version** section specifies the version of the AppSpec file.

  **Note**
  You should not change this value. It is reserved by AWS CodeDeploy for future use.
- The **os** section specifies the operating system value for the instance.
- The **files** section specifies the names of files that should be copied to the instance during the deployment's *Install* event.
- The **permissions** section specifies how any special permissions should be applied to the files in the **files** section as they are being copied over to the instance. This section applies only to Amazon Linux and Ubuntu Server instances.
- The **hooks** section specifies scripts to run at specific deployment lifecycle events during the deployment.

**Topics**



# version Section

The **version** section of the AppSpec file indicates the version of your application. It is required. Currently the only allowed value is **`0.0`**.

# os Section

The **os** section of the AppSpec file indicates the operating system of the instance that you will deploy to. It is required. The following values can be specified:

- **linux** – The instance is an Amazon Linux or Ubuntu Server instance.
- **windows** – The instance is a Windows Server instance.

# files Section

The *files* section provides information to AWS CodeDeploy about which files from your application revision should be installed on the instance during the deployment's *Install* event. This section is required only if you will be copying files from your revision to specific locations on the instance during deployment.

This section has the following structure:

```
files:
  - source: source-file-location
    destination: destination-file-location
```

Multiple **source** and **destination** pairs can be set.

The **source** instruction identifies a file or directory from your revision to copy to the instance:

- If **source** refers to a file, only the specified file will be copied to the instance.
- If **source** refers to a directory, then all files in the directory will be copied to the instance.
- If **source** is a single slash (/), then all of the files from your revision will be copied to the instance.

The paths used in **source** are relative paths, starting from the root of your revision.

The **destination** instruction identifies the location on the instance to copy the files to. This must be a fully-qualified path.

Here's an example **files** section for an Amazon Linux or Ubuntu Server instance.

```
files:
  - source: Config/config.txt
    destination: /webapps/Config
  - source: source
    destination: /webapps/myApp
```

In this example, the following two operations will be performed during the *Install* event:

1. Copy the file `Config/config.txt` in your revision to the path `/webapps/Config/config.txt` on the instance.
2. Recursively copy all the files in your revision's `source` directory to the `/webapps/myApp` directory on the instance.

## files Examples

The following examples shows how to specify the **files** section. Although these examples describe Windows Server file and directory (folder) structures, they can easily be adapted for Amazon Linux and Ubuntu Server instances.

The following set of examples assume that in the root of `source` there is an `appspec.yml` file along with 3 files named `my-file.txt`, `my-file-2.txt`, and `my-file-3.txt`, like this:

- `appspec.yml`
- `my-file.txt`
- `my-file-2.txt`
- `my-file-3.txt`

```
# 1) Copy only my-file.txt to the destination folder c:\temp.
#
files:
  - source: .\my-file.txt
    destination: c:\temp
#
```

```
# Result:
#   c:\temp\my-file.txt
#
# ---------------------
#
# 2) Copy only my-file-2.txt and my-file-3.txt to the destination folder c:\temp.
#
files:
  - source: my-file-2.txt
    destination: c:\temp
  - source: my-file-3.txt
    destination: c:\temp
#
# Result:
#   c:\temp\my-file-2.txt
#   c:\temp\my-file-3.txt
#
# ---------------------
#
# 3) Copy my-file.txt, my-file-2.txt, and my-file-3.txt (along with the
appspec.yml file) to the destination folder c:\temp.
#
files:
  - source: \
    destination: c:\temp
#
# Result:
#   c:\temp\appspec.yml
#   c:\temp\my-file.txt
#   c:\temp\my-file-2.txt
#   c:\temp\my-file-3.txt
```

The following set of examples assume that in the root of `source` there is an `appspec.yml` file along with a folder named `my-folder` and 3 files inside of `my-folder` named `my-file.txt`, `my-file-2.txt`, and `my-file-3.txt`, like this:

- `appspec.yml`
- `my-folder\my-file.txt`
- `my-folder\my-file-2.txt`
- `my-folder\my-file-3.txt`

```
# 4) Copy the 3 files in my-folder (but do not copy my-folder itself) to the
destination folder c:\temp.
#
files:
  - source: .\my-folder
    destination: c:\temp
#
# Result:
#   c:\temp\my-file.txt
#   c:\temp\my-file-2.txt
#   c:\temp\my-file-3.txt
#
# ---------------------
#
```

```
# 5) Copy my-folder and its 3 files to my-folder within the destination folder
 c:\temp.
#
files:
  - source: .\my-folder
    destination: c:\temp\my-folder
#
# Result:
#   c:\temp\my-folder\my-file.txt
#   c:\temp\my-folder\my-file-2.txt
#   c:\temp\my-folder\my-file-3.txt
#
# ---------------------
#
# 6) Copy the 3 files in my-folder to other-folder within the destination folder
 c:\temp.
#
files:
  - source: .\my-folder
    destination: c:\temp\other-folder
#
# Result:
#   c:\temp\other-folder\my-file.txt
#   c:\temp\other-folder\my-file-2.txt
#   c:\temp\other-folder\my-file-3.txt
#
# ---------------------
#
# 7) Copy only my-file-2.txt and my-file-3.txt to my-folder within the destina
tion folder c:\temp.
#
files:
  - source: .\my-folder\my-file-2.txt
    destination: c:\temp\my-folder
  - source: .\my-folder\my-file-3.txt
    destination: c:\temp\my-folder
#
# Result:
#   c:\temp\my-folder\my-file-2.txt
#   c:\temp\my-folder\my-file-3.txt
#
# ---------------------
#
# 8) Copy only my-file-2.txt and my-file-3.txt to other-folder within the des
tination folder c:\temp.
#
files:
  - source: .\my-folder\my-file-2.txt
    destination: c:\temp\other-folder
  - source: .\my-folder\my-file-3.txt
    destination: c:\temp\other-folder
#
# Result:
#   c:\temp\other-folder\my-file-2.txt
#   c:\temp\other-folder\my-file-3.txt
#
# ---------------------
#
```

```
# 9) Copy my-folder and its 3 files (along with the appspec.yml file) to the
destination folder c:\temp.
#
files:
  - source: \
    destination: c:\temp
#
# Result:
#   c:\temp\appspec.yml
#   c:\temp\my-folder\my-file.txt
#   c:\temp\my-folder\my-file-2.txt
#   c:\temp\my-folder\my-file-3.txt
```

# permissions Section

The **permissions** section specifies how any special permissions should be applied to the files and directories/folders in the **files** section after they are copied over to the instance. Multiple **object** instructions can be specified. This section is optional, and it applies only to Amazon Linux and Ubuntu Server instances.

This section has the following structure:

```
permissions:
  - object: object-specification
    pattern: pattern-specification
    except: exception-specification
    owner: owner-account-name
    group: group-name
    mode: mode-specification
    acls:
      - acls-specification
    context:
      user: user-specification
      type: type-specification
      range: range-specification
    type:
      - object-type
```

The instructions are as follows:

- **object** – *Required*. This is a set of file system objects (files or directories/folders) that the specified permissions will be applied to after the file system objects are copied to the instance.
- **pattern** – *Optional*. Specifies a pattern to apply permissions. If specified with the special characters **"***"**, or if not specified altogether, the specified permissions will be applied to all matching files or directories, depending on the **type**.
- **except** – *Optional*. Specifies any exceptions to **pattern**.
- **owner** – *Optional*. The name of the owner of **object**. If not specified, all existing owners applied to the original file or directory/folder structure will remain unchanged after the copy operation.
- **group** – *Optional*. The name of the group for **object**. If not specified, all existing groups applied to the original file or directory/folder structure will remain unchanged after the copy operation.
- **mode** – *Optional*. An integer specifying the octal mode for the permissions to be applied to **object**. For example, **644** represents read and write permissions for the owner, read-only permissions for the group, and read-only permissions for all other users; while **4755** represents the setuid attribute being set, full control permissions for the owner, read and execute permissions for the group, and read and execute permissions for all other users. (For additional examples, see the Linux **chmod** command

documentation.) If **mode** is not specified, all existing modes applied to the original file or directory/folder structure will remain unchanged after the copy operation.

- **acls** – *Optional*. A list of character strings representing one or more Access Control List (ACL) entries applied to **object**. For example, **`u:bob:rw`** represents read and write permissions for user **bob**. (For additional examples, see ACL entry format examples in the Linux **setfacl** command documentation.) Multiple ACL entries an be specified. If **acls** is not specified, any existing ACLs applied to the original file or directory/folder structure will remain unchanged after the copy operation. These will replace any existing ACLs.

  **Note**
  Setting unnamed users, unnamed groups, or other similar ACL entries will cause the AppSpec file to fail. Use **mode** to specify these types of permissions instead.

- **context** – *Optional*. For Security-Enhanced Linux (SELinux)-enabled instances, a list of security-relevant context labels to apply to the copied objects. Labels are specified as keys containing **user**, **type**, and **range**. (For more information, see the SELinux documentation.) If not specified, any existing labels applied to the original file or directory/folder structure will remain unchanged after the copy operation.
  - **user** – *Optional*. The SELinux user.
  - **type** – *Optional*. The SELinux type name.
  - **range** – *Optional*. The SELinux range specifier. This has no effect unless Multi-Level Security (MLS) and Multi-Category Security (MCS) is enabled on the machine. If MLS/MCS is not enabled, **range** defaults to **`s0`**.
- **type** – *Optional*. The types of objects to apply the specified permissions to. This can be set to **`file`** or **`directory`**. If **`file`** is specified, the permissions will be applied only to files that are immediately contained within **object** after the copy operation (and not to **object** itself). If **directory** is specified, the permissions will be recursively applied to all directories/folders that are anywhere within **object** after the copy operation (but not to **object** itself).

## permissions Example

The following example shows how to specify the **permissions** section with the **object**, **pattern**, **except**, **owner**, **mode**, and **type** instructions. This example applies only to Amazon Linux and Ubuntu Server instances. In this example, assume that the following files and folders are copied over to the instance in this hierarchy:

```
/tmp
  `-- my-app
       |-- my-file-1.txt
       |-- my-file-2.txt
       |-- my-file-3.txt
       |-- my-folder-1
       |     |-- my-file-4.txt
       |     |-- my-file-5.txt
       |     `-- my-file-6.txt
       `-- my-folder-2
             |-- my-file-7.txt
             |-- my-file-8.txt
             |-- my-file-9.txt
          `-- my-folder-3
```

The following AppSpec file shows how to set various permissions on these files and folders after they are copied over:

```
version: 0.0
os: linux
```

```
# Copy over all of the folders and files with whatever permissions they
#  were originally assigned.
files:
  - source: ./my-file-1.txt
    destination: /tmp/my-app
  - source: ./my-file-2.txt
    destination: /tmp/my-app
  - source: ./my-file-3.txt
    destination: /tmp/my-app
  - source: ./my-folder-1
    destination: /tmp/my-app/my-folder-1
  - source: ./my-folder-2
    destination: /tmp/my-app/my-folder-2
# 1) For all the files in the /tmp/my-app folder ending in -3.txt
#  (for example, just my-file-3.txt), owner = adm, group = wheel, and
#  mode = 464 (-r--rw-r--).
permissions:
  - object: /tmp/my-app
    pattern: "*-3.txt"
    owner: adm
    group: wheel
    mode: 464
    type:
      - file
# 2) For all of the files ending in .txt in the /tmp/my-app
#  folder, but not for the file my-file-3.txt (for example,
#  just my-file-1.txt and my-file-2.txt),
#  owner = ec2-user and mode = 444 (-r--r--r--).
  - object: /tmp/my-app
    pattern: "*.txt"
    except: [my-file-3.txt]
    owner: ec2-user
    mode: 444
    type:
      - file
# 3) For all the files in the /tmp/my-app/my-folder-1 folder except
#  for my-file-4.txt and my-file-5.txt, (for example,
#  just my-file-6.txt), owner = operator and mode = 646 (-rw-r--rw-).
  - object: /tmp/my-app/my-folder-1
    pattern: "**"
    except: [my-file-4.txt, my-file-5.txt]
    owner: operator
    mode: 646
    type:
      - file
# 4) For all of the files that are immediately under
#  the /tmp/my-app/my-folder-2 folder except for my-file-8.txt,
#  (for example, just my-file-7.txt and
#  my-file-9.txt), owner = ec2-user and mode = 777 (-rwxrwxrwx).
  - object: /tmp/my-app/my-folder-2
    pattern: "**"
    except: [my-file-8.txt]
    owner: ec2-user
    mode: 777
    type:
      - file
# 5) For all folders at any level under /tmp/my-app that contain
#  the name my-folder but not
```

```
#  /tmp/my-app/my-folder-2/my-folder-3 (for example, just
#  /tmp/my-app/my-folder-1 and /tmp/my-app/my-folder-2),
#  owner = ec2-user and mode = 555 (dr-xr-xr-x).
  - object: /tmp/my-app
    pattern: "*my-folder*"
    except: [tmp/my-app/my-folder-2/my-folder-3]
    owner: ec2-user
    mode: 555
    type:
      - directory
# 6) For the folder /tmp/my-app/my-folder-2/my-folder-3,
#  group = wheel and mode = 564 (dr-xrw-r--).
  - object: /tmp/my-app/my-folder-2
    group: wheel
    mode: 564
    type:
      - directory
```

The resulting permissions are as follows:

```
-r--r--r-- ec2-user root  my-file-1.txt
-r--r--r-- ec2-user root  my-file-2.txt
-r--rw-r-- adm      wheel my-file-3.txt

dr-xr-xr-x ec2-user root  my-folder-1
-rw-r--r-- root     root  my-file-4.txt
-rw-r--r-- root     root  my-file-5.txt
-rw-r--rw- operator root  my-file-6.txt

dr-xr-xr-x ec2-user root  my-folder-2
-rwxrwxrwx ec2-user root  my-file-7.txt
-rw-r--r-- root     root  my-file-8.txt
-rwxrwxrwx ec2-user root  my-file-9.txt

dr-xrw-r-- root     wheel my-folder-3
```

The following example shows how to specify the **permissions** section with the addition of the **acls** and **context** instructions. This example applies only to Amazon Linux and Ubuntu Server instances.

```
permissions:
  - object: /var/www/html/WordPress
    pattern: "**"
    except: [/var/www/html/WordPress/ReadMe.txt]
    owner: bob
    group: writers
    mode: 644
    acls:
      - u:mary:rw
      - u:sam:rw
      - m::rw
    context:
      user: unconfined_u
      type: httpd_sys_content_t
      range: s0
    type:
      - file
```

# hooks Section

The **hooks** section of the AppSpec file contains mappings that link deployment lifecycle event hooks to one or more scripts. If an event hook is not present, then no operation is executed for that event. This section is required only if you will be running scripts as part of the deployment.

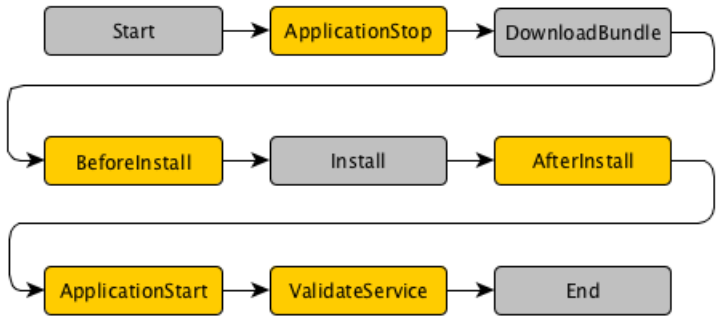
The available event hooks are:

1. **ApplicationStop** – this is the first deployment lifecycle event that occurs even before the application revision gets downloaded. You can use the **ApplicationStop** deployment lifecycle event if you want to gracefully stop the application or remove currently installed packages in preparation of a deployment. The AppSpec file and scripts used for this deployment lifecycle event are from the last successfully deployed application revision. To determine the location of the last successfully deployed application revision, the AWS CodeDeploy Agent looks up the location that is listed in the *deployment-group-id*_last_successful_install file. This file is located within the /opt/codedeploy-agent/deployment-root/deployment-instructions folder on Amazon Linux and Ubuntu Server instances and within the C:\ProgramData\Amazon\CodeDeploy\deployment-instructions folder on Windows Server instances.

   To troubleshoot a deployment that fails during the **ApplicationStop** deployment lifecycle event, see Troubleshooting a Failed ApplicationStop Deployment Lifecycle Event (p. 194).
2. **DownloadBundle** – during this deployment lifecycle event, the AWS CodeDeploy Agent copies the application revision files to a temporary location: the /opt/codedeploy-agent/deployment-root/*deployment-group-id*/*deployment-id*/deployment-archive folder on Amazon Linux and Ubuntu Server instances or the C:\ProgramData\Amazon\CodeDeploy*deployment-group-id**deployment-id*\deployment-archive folder on Windows Server instances. This deployment lifecycle event is reserved for the AWS CodeDeploy Agent and cannot be used to run scripts.

   To troubleshoot a deployment that fails during the **DownloadBundle** deployment lifecycle event, see Troubleshooting a Failed DownloadBundle Deployment Lifecycle Event with "UnknownError: Not Opened for Reading" (p. 195).
3. **BeforeInstall** – you can use the **BeforeInstall** deployment lifecycle event for preinstall tasks such as decrypting files and creating a backup of the current version.
4. **Install** – during this deployment lifecycle event, the AWS CodeDeploy Agent copies the revision files from the temporary location to the final destination folder. This deployment lifecycle event is reserved for the AWS CodeDeploy Agent and cannot be used to run scripts.
5. **AfterInstall** – you can use the **AfterInstall** deployment lifecycle event for tasks such as configuring your application or changing file permissions.
6. **ApplicationStart** – you typically use the **ApplicationStart** deployment lifecycle event to restart services that were stopped during **ApplicationStop**.
7. **ValidateService** – **ValidateService** is the last deployment lifecycle event and is an opportunity to verify that the deployment completed successfully.

These event hooks occur in order during the deployment process, which is illustrated here:




**Note**
The *Start*, *DownloadBundle*, *Install* and *End* events in the deployment cannot be scripted, as represented in gray in this diagram. However, you can affect what's installed during the *Install* event, and where, by editing the **files** section of the AppSpec file.

This section has the following structure:

```
hooks:
   deployment-lifecycle-event-name
     - location: script-location
       timeout: timeout-in-seconds
       runas: user-name
```

You can include the following elements in a **hook** entry after the specific deployment lifecycle event name:

- **location** – *Required*. The location of the script file from the revision to run.
- **timeout** – *Optional*. The number of seconds to allow the script to execute before it is considered to be failed. The default is 1800 seconds (30 minutes).

  **Note**
  The maximum amount of time that all scripts must finish executing for each individual deployment lifecycle event is 3600 seconds (1 hour). Otherwise, the deployment will stop and AWS CodeDeploy will consider the deployment to have failed to the instance. Make sure that the total number of seconds that are specified in **timeout** for all scripts in each individual deployment lifecycle event does not exceed a combined 3600 seconds (1 hour).
- **runas** – *Optional*. The user to impersonate when running the script. The default is the same user that the AWS CodeDeploy Agent on the instance is running as. AWS CodeDeploy does not store passwords, so this will fail if the *runas* user needs a password. This element applies only to Amazon Linux and Ubuntu Server instances.

## hooks Example

An example of a **hooks** entry is:

```
hooks:
   AfterInstall:
     - location: Scripts/RunResourceTests.sh
       timeout: 180
```

In this entry, the script `Scripts/RunResourceTests.sh` will be run during the *AfterInstall* stage of the deployment process, but will be considered unsuccessful if it takes the script more than 180 seconds (3 minutes) to run.

# AppSpec file Example

Here is an example of an AppSpec file for an Amazon Linux or Ubuntu Server instances.

For a Windows Server instance, you would change `os: linux` to `os: windows`. Also, you must fully qualify the `destination` paths (for example, something like `c:\temp\webapps\Config` and `c:\temp\webapps\myApp`), and do not include the `runas` element.

An explanation of the order of the operations on the instance follows the example.

```
os: linux
files:
  - source: Config/config.txt
    destination: webapps/Config
  - source: source
    destination: /webapps/myApp
hooks:
  BeforeInstall:
    - location: Scripts/UnzipResourceBundle.sh
    - location: Scripts/UnzipDataBundle.sh
  AfterInstall:
    - location: Scripts/RunResourceTests.sh
      timeout: 180
  ApplicationStart:
    - location: Scripts/RunFunctionalTests.sh
      timeout: 3600
  ValidateService:
    - location: Scripts/MonitorService.sh
      timeout: 3600
      runas: codedeployuser
```

This file will cause the following sequence of events during deployment:

1. Run the script located at `Scripts/UnzipResourceBundle.sh`.
2. Run the script located at `Scripts/UnzipDataBundle.sh` if the previous script returned an exit code of 0 (success).
3. Copy the file from the path of `Config/config.txt` to the path `/webapps/Config/config.txt`.
4. Recursively copy all the files in the `source` directory to the `/webapps/myApp` directory.
5. Run the script located at `Scripts/RunResourceTests.sh` with a timeout of 180 seconds (3 minutes).
6. Run the script located at `Scripts/RunFunctionalTests.sh` with a timeout of 3600 seconds (1 hour).
7. Run the script located at `Scripts/MonitorService.sh` as the user *codedeploy* with a timeout of 3600 seconds (1 hour).

# AppSpec file Spacing

The locations and numbers of spaces between various items in an AppSpec file is important. AWS CodeDeploy expects these spaces to be in the proper positions in the AppSpec file. Otherwise, AWS CodeDeploy will raise an error that may be difficult to debug.

The following is the correct format for AppSpec file spacing. The numbers in square brackets indicate the number of spaces that must occur between items. For example, `[4]` means to insert four spaces between the items.

```
version:[1]version-number
os:[1]operating-system-name
files:
[2]-[1]source:[1]source-files-location
[4]destination:[1]destination-files-location
permissions:
[2]-[1]object:[1]object-specification
[4]pattern:[1]pattern-specification
[4]except:[1]exception-specification
[4]owner:[1]owner-account-name
[4]group:[1]group-name
[4]mode:[1]mode-specification
[4]acls:
[6]-[1]acls-specification
[4]context:
[6]user:[1]user-specification
[6]type:[1]type-specification
[6]range:[1]range-specification
[4]type:
[6]-[1]object-type
hooks:
[2]deployment-lifecycle-event-name:
[4]-[1]location:[1]script-location
[6]timeout:[1]timeout-in-seconds
[6]runas:[1]user-name
```

Here is an example of a conforming AppSpec file:

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html/WordPress
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies.sh
      timeout: 300
      runas: root
  AfterInstall:
    - location: scripts/change_permissions.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

For more information on spacing, see the YAML specification.

# Validating Your AppSpec File

To check the validity of your AppSpec file, you can use a YAML validator. If you don't have one, search the Internet for "YAML validator."

To verify that you have placed your AppSpec file in the root directory of the application's source content's directory structure, run one of the following commands:

For Linux, OS X, or Unix:

```
find /path/to/root/directory -name appspec.yml
```

If the AppSpec file is *not* located there, there will be no output.

For Windows:

```
dir path\to\root\directory\appspec.yml
```

If the AppSpec file is *not* located there, a **File Not Found** error displays.

# AWS CodeDeploy User Access Permissions Reference

You can use IAM to enable IAM users to work with only certain AWS CodeDeploy resources and perform only certain actions against those resources. You may want to do this, for example, if you have a set of IAM users that you want to give read-only access to certain information in AWS CodeDeploy; you may have another set of IAM users that you want to give the ability to only deploy applications to certain deployment groups; and so on.

In the Setting Up (p. 4) instructions, you attached a policy to an IAM user that looks similar to this:

```
{
 "Version": "2012-10-17",
 "Statement" : [
   {
     "Effect" : "Allow",
     "Action" : "codedeploy:*",
     "Resource" : "*"
   },
   ...
 ]
}
```

The first statement here allows the IAM user to perform all available actions in AWS CodeDeploy with all available AWS CodeDeploy resources that are associated with the IAM user. In practice, you may not want to give all IAM users this much access.

The following information shows how you can attach a policy to an IAM user that restricts the actions in AWS CodeDeploy that the user can perform as well as restricts the AWS CodeDeploy resources that the IAM user can take action against.

## Attach a Policy to an IAM User

To attach a policy to an IAM user that restricts the IAM user to certain actions and resources in AWS CodeDeploy, do the following:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the IAM console, in the navigation pane, click **Policies**, and then click **Create Policy**. (If a **Get Started** button appears, click it, and then click **Create Policy**.)
3. Next to **Create Your Own Policy**, click **Select**.
4. In the **Policy Name** box, you can type any value that makes it easy for you to refer to later as needed.
5. In the **Policy Document** box, type a policy that follows this format, and then click **Create Policy**:

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "action-statement"
      ],
      "Resource" : [
        "resource-statement"
      ]
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "action-statement"
      ],
      "Resource" : [
        "resource-statement"
      ]
    }
  ]
}
```

   In the preceding statement, substitute *action-statement* and *resource-statement* as needed, and add additional statements as needed, to specify the AWS CodeDeploy actions and resources that you want to allow for the IAM user. (By default, the IAM user will not have the desired permissions unless a corresponding `Allow` statement is explicitly stated.) The following sections describe the format of allowed actions and resources for AWS CodeDeploy.
6. In the navigation pane, click **Users**.
7. Click the name of the IAM user that you want to attach the policy to.
8. Expand the **Permissions** area, if it is not already expanded.
9. In the **Managed Policies** area, click **Attach Policy**.
10. Select the policy that you just created, and then click **Attach Policy**.

# Action and Resource Syntax

The following sections describe the format for specifying actions and resources.

Actions follow this general format:

```
codedeploy:action
```

Where *action* is an available AWS CodeDeploy operation, such as `CreateApplication` or `DeleteDeploymentGroup`.

Resources follow this general format:

```
arn:aws:codedeploy:region:account:resource-type:resource-specifier
```

Where *region* is a target region (such as **us-east-1**), *account* is the AWS account ID, *resource-type* is the target type of resource (such as **deploymentconfig** for deployment configurations), and *resource-specifier* is the specific target resource (such as **WordPress_App** for an application or ***** for all resources of that resource type).

For example, the following specifies the **RegisterApplicationRevision** action:

```
codedeploy:RegisterApplicationRevision
```

While the following specifies the application named **WordPress_App** registered to the AWS account **80398EXAMPLE** in the region **us-east-1**:

```
arn:aws:codedeploy:us-east-1:80398EXAMPLE:application:WordPress_App
```

**Topics**



# Applications

Allowed actions include:

- **BatchGetApplications**, to get information about multiple applications associated with the IAM user.
- **CreateApplication**, to create an application associated with the IAM user.
- **DeleteApplication**, to delete an application associated with the IAM user.
- **GetApplication**, to get information about a single application associated with the IAM user.
- **ListApplications**, to get information about all applications associated with the IAM user.
- **UpdateApplication**, to change information about an application associated with the IAM user.

> **Note**
> For **UpdateApplication**, you must have **UpdateApplication** permissions for both the old application name and the new application name.

Allowed resources include:

- **application:*application-name*** (valid for all of the preceding application actions, except **BatchGetApplications** and **ListApplications**), where *application-name* is the complete name of a specific application.

- **`application:`*`partial-application-name`*`*`** (valid for all of the preceding application actions, except **`BatchGetApplications`** and **`ListApplications`**), where *`partial-application-name`* is the partial name of an application, and where **`*`** represents any series of remaining characters.
- **`application:*`** (valid for all of the preceding application actions, including **`BatchGetApplications`** and **`ListApplications`**), where **`*`** represents all applications.

The following example allows the specified user to get information about the application named **`WordPress_App`** in the **`us-east-1`** region:

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetApplication"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-east-1:80398EXAMPLE:application:WordPress_App"
      ]
    }
  ]
}
```

# Application Revisions

Allowed actions include:

- **`GetApplicationRevision`**, to get information about a single application revision for an application associated with the IAM user.
- **`ListApplicationRevisions`**, to get information about all application revisions for an application associated with the IAM user.
- **`RegisterApplicationRevision`**, to register information about an application revision for an application associated with the IAM user.

Allowed resources include:

- **`application:`*`application-name`*** (valid for all of the preceding application revision actions, except **`ListApplicationRevisions`**), where *`application-name`* is the complete name of a specific application.
- **`application:`*`partial-application-name`*`*`** (valid for all of the preceding application revision actions, except **`ListApplicationRevisions`**), where *`partial-application-name`* is the partial name of an application, and where **`*`** represents any series of remaining characters.
- **`application:*`** (valid for all of the preceding application revision actions, including **`ListApplicationRevisions`**), where **`*`** represents all applications.

The following example allows the specified user to register application revisions for the application named **`WordPress_App`** in the **`us-east-1`** region:

```
{
  "Version": "2012-10-17",
  "Statement" : [
```

```
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:RegisterApplicationRevision"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-east-1:80398EXAMPLE:application:WordPress_App"
      ]
    }
  ]
}
```

# Deployments

Allowed actions include:

- **`AbortDeployment`**, to stop a deployment to a deployment group for an application associated with the IAM user.
- **`BatchGetDeployments`**, to get information about multiple deployments associated with the IAM user.
- **`CreateDeployment`**, to create a deployment with a deployment configuration to a deployment group for an application associated with the IAM user.
- **`GetDeployment`**, to get information about a single deployment to a deployment group for an application associated with the IAM user.
- **`ListDeployments`**, to get information about all deployments to a deployment group associated with the IAM user, or to get all deployments associated with the IAM user.

> **Note**
> When specifying **`CreateDeployment`** permissions, you must also specify corresponding **`GetDeploymentConfig`** permissions for the deployment configuration, and **`GetApplicationRevision`** or **`RegisterApplicationRevision`** permissions for the existing or new application revision, respectively, as appropriate.

Allowed resources include:

- **`deploymentgroup:`***`application-name`*`/`***`deployment-group-name`** (valid for all of the preceding deployment actions), where *`application-name`* is the complete name of a specific application, and *`deployment-group-name`* is the complete name of a specific deployment group associated with the matching application.
- **`deploymentgroup:`***`partial-application-name`*`*/`***`deployment-group-name`** (valid for all of the preceding deployment actions), where *`partial-application-name`* is the partial name of an application, and where **`*`** represents any series of remaining characters; and *`deployment-group-name`* is the complete name of a specific deployment group associated with any matching application.
- **`deploymentgroup:`***`application-name`*`/`***`partial-deployment-group-name`*`*`** (valid for all of the preceding deployment actions), where *`application-name`* is the complete name of a specific application; and *`partial-deployment-group-name`* is the partial name of a deployment group associated with the matching application, and where **`*`** represents any series of remaining characters.
- **`deploymentgroup:`***`partial-application-name`*`*/`***`partial-deployment-group-name`*`*`** (valid for all of the preceding deployment actions), where *`partial-application-name`* is the partial name of an application, and where **`*`** represents any series of remaining characters; and *`partial-deployment-group-name`* is the partial name of a deployment group associated with any matching application, and where **`*`** represents any series of remaining characters.
- **`deploymentgroup:`***`application-name`*`/*`** (valid for all of the preceding deployment actions except for **`BatchGetDeployments`**; also valid for **`ListDeployments`** when providing a specific deployment group, not when listing all of the deployments associated with the IAM user), where *`application-name`*

is the name of a specific application, and ***** represents any deployment group associated with the matching application.

- **`deploymentgroup:`*`partial-application-name`*`*/*`** (valid for all of the preceding deployment actions), where *`partial-application-name`* is the partial name of an application, and where ***** represents any series of remaining characters; and where ***** represents any deployment group associated with the matching applications.
- **`deploymentgroup:*`** (valid for all of the preceding deployment actions, including **`BatchGetDeployments`**), where ***** represents all deployments.

The following example allows the specified user to create deployments for the deployment group named **`WordPress_DepGroup`** associated with the application named **`WordPress_App`**, the custom deployment configuration named **`ThreeQuartersHealthy`**, and any existing application revisions associated with the application named **`WordPress_App`**. All of these resources are associated with the **`us-east-1`** region.

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:CreateDeployment"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-east-1:80398EXAMPLE:deploymentgroup:Word
Press_App/WordPress_DepGroup"
      ]
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetDeploymentConfig"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-east-1:80398EXAMPLE:deploymentconfig:ThreeQuar
tersHealthy"
      ]
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetApplicationRevision"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-east-1:80398EXAMPLE:application:WordPress_App"
      ]
    }
  ]
}
```

# Deployment Configurations

Allowed actions include:

- **`CreateDeploymentConfig`**, to create a custom deployment configuration associated with the IAM user.

- **`DeleteDeploymentConfig`**, to delete a custom deployment configuration associated with the IAM user.
- **`GetDeploymentConfig`**, to get information about a single deployment configuration associated with the IAM user.
- **`ListDeploymentConfigs`**, to get information about all deployment configurations associated with the IAM user.

Allowed resources include:

- **`deploymentconfig:`*`custom-deployment-configuration-name`*** (valid for all of the preceding deployment configuration actions, except **`ListDeploymentConfigs`**), where *`custom-deployment-configuration-name`* is the complete name of a custom deployment configuration.
- **`deploymentconfig:`*`partial-custom-deployment-configuration-name`*`*`** (valid for all of the preceding deployment configuration actions, except **`ListDeploymentConfigs`**), where *`partial-custom-deployment-configuration-name`* is the partial name of a custom deployment configuration, and where **`*`** represents any series of remaining characters.
- **`deploymentconfig:`*`predefined-deployment-configuration-name`*** (valid for all of the preceding deployment configuration actions, except **`ListDeploymentConfigs`**), where *`predefined-deployment-configuration-name`* is the name of a specific built-in deployment configuration, such as **`CodeDeployDefault.OneAtATime`**.
- **`deploymentconfig:`*`partial-predefined-deployment-configuration-name`*`*`** (valid for all of the preceding deployment configuration actions, except **`ListDeploymentConfigs`**), where *`partial-predefined-deployment-configuration-name`* is the partial name of a built-in deployment configuration, and where **`*`** represents any series of remaining characters.
- **`deploymentconfig:*`** (valid for all of the preceding deployment configuration actions, including **`ListDeploymentConfigs`**), where **`*`** represents all deployment configurations.

The following example allows the specified user to get information about the custom deployment configuration named **`ThreeQuartersHealthy`** in the **`us-east-1`** region.

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetDeploymentConfig"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-east-1:80398EXAMPLE:deploymentconfig:ThreeQuar
tersHealthy"
      ]
    }
  ]
}
```

# Deployment Groups

Allowed actions include:

- **`CreateDeploymentGroup`**, to create a deployment group for an application associated with the IAM user.

- **DeleteDeploymentGroup**, to delete a deployment group for an application associated with the IAM user.
- **GetDeploymentGroup**, to get information about a single deployment group for an application associated with the IAM user.
- **ListDeploymentGroups**, to get information about all deployment groups for an application associated with the IAM user.
- **UpdateDeploymentGroup**, to change information about a single deployment group for an application associated with the IAM user.

> **Note**
> For **UpdateDeploymentGroup** actions that involve changing a deployment group's name, you must have **UpdateDeploymentGroup** permissions for both the old deployment group name and the new deployment group name.

Allowed resources include:

- **deploymentgroup:*application-name*/*deployment-group-name*** (valid for all of the preceding deployment group actions, except **ListDeploymentGroups**), where *application-name* is the complete name of a specific application, and *deployment-group-name* is the complete name of a specific deployment group associated with the matching application.
- **deploymentgroup:*partial-application-name**/*deployment-group-name*** (valid for all of the preceding deployment group actions, except **ListDeploymentGroups**), where *partial-application-name* is the partial name of an application, and where ***** represents any series of remaining characters; and *deployment-group-name* is the complete name of a specific deployment group associated with any matching application.
- **deploymentgroup:*application-name*/*partial-deployment-group-name**** (valid for all of the preceding deployment group actions, except **ListDeploymentGroups**), where *application-name* is the complete name of a specific application; and *partial-deployment-group-name* is the partial name of a deployment group associated with the matching application, and where ***** represents any series of remaining characters.
- **deploymentgroup:*partial-application-name**/*partial-deployment-group-name**** (valid for all of the preceding deployment group actions, except **ListDeploymentGroups**), where *partial-application-name* is the partial name of an application, and where ***** represents any series of remaining characters; and *partial-deployment-group-name* is the partial name of a deployment group associated with any matching application, and where ***** represents any series of remaining characters.
- **deploymentgroup:*application-name*/*** (valid for all of the preceding deployment group actions, including **ListDeploymentGroups**), where *application-name* is the name of a specific application, and ***** represents any deployment group associated with the matching application.
- **deploymentgroup:*partial-application-name**/*** (valid for all of the preceding deployment group actions, including **ListDeploymentGroups**), where *partial-application-name* is the partial name of an application, and where ***** represents any series of remaining characters; and where ***** represents any deployment groups associated with the matching applications.

The following example allows the user to delete the deployment group named **WordPress_DepGroup** associated with the application named **WordPress_App**in the **us-east-1** region.

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:DeleteDeploymentGroup"
```

```
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-east-1:80398EXAMPLE:deploymentgroup:Word
Press_App/WordPress_DepGroup"
      ]
    }
  ]
}
```

# Instances

Allowed actions include:

- **`GetDeploymentInstance`**, to get information about a single instance in a deployment associated with the IAM user.
- **`ListDeploymentInstances`**, to get information about all instances in a deployment associated with the IAM user.

> **Note**
> To get information specific to on-premises instances, see the next section.

Allowed resources include:

- **`deploymentgroup:`*`application-name`*`/`*`deployment-group-name`** (valid for all of the preceding instance actions), where *`application-name`* is the complete name of a specific application, and *`deployment-group-name`* is the complete name of a specific deployment group associated with the matching application.
- **`deploymentgroup:`*`partial-application-name`*`*/`*`deployment-group-name`** (valid for all of the preceding instance actions), where *`partial-application-name`* is the partial name of an application, and where **`*`** represents any series of remaining characters; and *`deployment-group-name`* is the complete name of a specific deployment group associated with any matching application.
- **`deploymentgroup:`*`application-name`*`/`*`partial-deployment-group-name`*`*`** (valid for all of the preceding instance actions), where *`application-name`* is the complete name of a specific application; and *`partial-deployment-group-name`* is the partial name of a deployment group associated with the matching application, and where **`*`** represents any series of remaining characters.
- **`deploymentgroup:`*`partial-application-name`*`*/`*`partial-deployment-group-name`*`*`** (valid for all of the preceding instance actions), where *`partial-application-name`* is the partial name of an application, and where **`*`** represents any series of remaining characters; and *`partial-deployment-group-name`* is the partial name of a deployment group associated with any matching application, and where **`*`** represents any series of remaining characters.
- **`deploymentgroup:`*`application-name`*`/*`** (valid for all of the preceding instance actions), where *`application-name`* is the name of a specific application, and **`*`** represents any deployment group associated with the matching application.
- **`deploymentgroup:`*`partial-application-name`*`*/*`** (valid for all of the preceding instance actions), where *`partial-application-name`* is the partial name of an application, and where **`*`** represents any series of remaining characters; and where **`*`** represents any deployment groups associated with the matching applications.

The following example allows the user to get information about all of the instances in deployments associated with the deployment group named **`WordPress_DepGroup`** associated with the application named **`WordPress_App`**in the **`us-east-1`** region.

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:ListDeploymentInstances"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-east-1:80398EXAMPLE:deploymentgroup:Word
Press_App/WordPress_DepGroup"
      ]
    }
  ]
}
```

# On-Premises Instances

Allowed actions include:

- **`AddTagsToOnPremisesInstances`**, to add tags to one or more on-premises instances.
- **`BatchGetOnPremisesInstances`**, to get information about one or more on-premises instances.
- **`DeregisterOnPremisesInstance`**, to deregister an on-premises instance.
- **`GetOnPremisesInstance`**, to get information about a single on-premises instance.
- **`ListOnPremisesInstances`**, to get a list of one or more on-premises instance names.
- **`RegisterOnPremisesInstance`**, to register an on-premises instance.
- **`RemoveTagsFromOnPremisesInstances`**, to remove tags from one or more on-premises instances.

Allowed resources include:

- **`instance/`*`instance-ID`*** (valid for all of the preceding on-premises instance actions except for **`BatchGetOnPremisesInstances`** and **`ListOnPremisesInstances`**), where *`instance-ID`* is the complete ID of a specific on-premises instance.
- **`instance/`*`partial-instance-ID`*`*`** (valid for all of the preceding on-premises instance actions except for **`BatchGetOnPremisesInstances`** and **`ListOnPremisesInstances`**), where *`partial-instance-ID`* is a partial on-premises instance ID, and where **`*`** represents any series of remaining characters.
- **`instance/*`** (valid for all of the preceding on-premises instance actions except for **`BatchGetOnPremisesInstances`** and **`ListOnPremisesInstances`**), where **`*`** represents any available on-premises instance.
- **`*`** (valid for all of the preceding on-premises instance actions), where **`*`** represents any available on-premises instance.

The following example allows the user to get information about any single on-premises instance that begins with the ID of **`AssetTag`** in the **`us-east-1`** region.

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
```

```
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetOnPremisesInstance"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/AssetTag*"
      ]
    }
  ]
}
```

# AWS CodeDeploy Resource Kit

We're storing many of the supporting files that AWS CodeDeploy relies on in a set of publicly-available, AWS region-specific, Amazon S3 buckets. We call this collection of files the *AWS CodeDeploy Resource Kit*. Several of the procedures, walkthroughs, and tutorials in this *User Guide* rely on these files. Here's a list of the kit's files, how to browse through them, and how to download them, if you wish.

**Topics**



## Resource Kit File List

| | |
|---|---|
| `VERSION` | A special file that the various AWS CodeDeploy Agents use to automatically update themselves as they are running on instances. |
| `codedeploy-agent.noarch.rpm` | The AWS CodeDeploy Agent for Amazon Linux. Note that there may be several files here with the same base file name but with different versions (such as `-1.0-0`). |
| `codedeploy-agent_all.deb` | The AWS CodeDeploy Agent for Ubuntu Server. Note that there may be several files here with the same base file name but with different versions (such as `_1.0-0`). |
| `codedeploy-agent.msi` | The AWS CodeDeploy Agent for Windows Server. Note that there may be several files here with the same base file name but with different versions (such as `-1.0-0`). |
| `install` | A special file that you can use to more easily install the AWS CodeDeploy Agent. |

| | |
|---|---|
| `CodeDeploy_SampleCF_Template.json` | An AWS CloudFormation template that you can use to launch from 1 to 3 Amazon Linux or Windows Server Amazon EC2 instances with the proper IAM instance profile attached and with the AWS CodeDeploy Agent installed so that AWS CodeDeploy can deploy applications to them right away. Note that there may be several files here with the same base file name but with different versions (such as `-1.0.0`). |
| `SampleApp_Linux.zip` | A sample deployable application revision that you can deploy to an Amazon Linux Amazon EC2 instance. Note that there may be several files here with the same base file name but with different versions (such as `-1.0`). |
| `SampleApp_Windows.zip` | A sample deployable application revision that you can deploy to a Windows Server Amazon EC2 instance. Note that there may be several files here with the same base file name but with different versions (such as `-1.0`). |

# Listing the Resource Kit's Files

To view a list of the kit's files, use one of the following AWS CLI **aws s3 ls** commands for your region.

**Note**
The files in each bucket are designed to work with resources in the corresponding region.

```
aws s3 ls --recursive s3://aws-codedeploy-us-east-1
```

```
aws s3 ls --recursive s3://aws-codedeploy-us-west-2
```

```
aws s3 ls --recursive s3://aws-codedeploy-eu-west-1
```

```
aws s3 ls --recursive s3://aws-codedeploy-ap-southeast-2
```

# Downloading the Resource Kit's Files

To download a particular file, use one of the following **aws s3 cp** commands for your region. For example, the following commands download a single file named `SampleApp_Linux.zip` from one of the buckets' `/samples/latest/` folders:

```
aws s3 cp s3://aws-codedeploy-us-east-1/samples/latest/SampleApp_Linux.zip . --region us-east-1
```

```
aws s3 cp s3://aws-codedeploy-us-west-2/samples/latest/SampleApp_Linux.zip . --region us-west-2
```

```
aws s3 cp s3://aws-codedeploy-eu-west-1/samples/latest/SampleApp_Linux.zip . --region eu-west-1
```

```
aws s3 cp s3://aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Linux.zip . --region ap-southeast-2
```

**Note**
Don't forget the period character (`.`) near the end. This downloads the file to your current directory.

To download all of the files, use one of the following commands for your region:

```
aws s3 cp --recursive s3://aws-codedeploy-us-east-1 . --region us-east-1
```

```
aws s3 cp --recursive s3://aws-codedeploy-us-west-2 . --region us-west-2
```

```
aws s3 cp --recursive s3://aws-codedeploy-eu-west-1 . --region eu-west-1
```

```
aws s3 cp --recursive s3://aws-codedeploy-ap-southeast-2 . --region ap-southeast-2
```

# Limits in AWS CodeDeploy

The following table describes current limits within AWS CodeDeploy.

**Topics**



## Applications

| | |
|---|---|
| Number of applications associated with an AWS account in a single region | 40 |
| Number of characters in an application name | 100 |
| Characters allowed in an application name | The letter characters a through z, inclusive.<br><br>The letter characters A through Z, inclusive.<br><br>The number characters 0 through 9, inclusive.<br><br>The special characters + (plus sign), = (equals sign), , (comma), . (period), @ (at sign), - (minus sign), and _ (underscore). |

## Application Revisions

| | |
|---|---|
| Number of characters in an application revision name | 100 |

| | |
|---|---|
| Allowed file types for deployable application revisions | Archive files with the extension `.zip` or `.tar` and compressed archive files with the extension `.tar.gz`.<br><br>Note that an archive or compressed archive file that is compatible with AWS CodeDeploy must contain a single Application Specification File (AppSpec file) with the file name `appspec.yml` but can also contain any type of file that you want to deploy. |

# Deployments

| | |
|---|---|
| Number of sequential deployments to a deployment group* | 1 |
| Number of sequential deployments associated with an AWS account** | 10 |
| Number of hours until a deployment fails if not completed | 8 |
| Number of seconds until an individual deployment lifecycle event fails if not completed | 3600 |
| Number of characters in a deployment description | 100 |

* This limit is intended to prevent accidental deployments of the same application to the same deployment group sequentially.

** Each deployment to a scaled-up Amazon EC2 instance in an Auto Scaling group counts as a single sequential deployment. If the scaled-up Amazon EC2 instance is associated with multiple applications, then this would generate an additional sequential deployment for each application. For example, an Auto Scaling group that scales up by 5 Amazon EC2 instances and is associated with a single application would generate 5 sequential deployments. If the same 5 scaled-up Amazon EC2 instances are associated with 2 additional applications, then this would generate 10 additional sequential deployments.

# Deployment Configurations

| | |
|---|---|
| Number of custom deployment configurations associated with an AWS account | 25 |
| Allowed values for a minimum healthy instances setting of **HOST_COUNT** | Any positive integer. |
| Allowed values for a minimum healthy instances setting of **FLEET_PERCENT** | 0 (zero), or any positive integer less than 100. |
| Number of characters in a custom deployment configuration name | 100 |

| | |
|---|---|
| Characters allowed in a custom deployment configuration name | The letter characters a through z, inclusive.<br><br>The letter characters A through Z, inclusive.<br><br>The number characters 0 through 9, inclusive.<br><br>The special characters + (plus sign), = (equals sign), , (comma), . (period), @ (at sign), - (minus sign), and _ (underscore). |
| Disallowed prefixes in a custom deployment configuration name | `CodeDeployDefault.` |

# Deployment Groups

| | |
|---|---|
| Number of deployment groups associated with a single application | 10 |
| Number of tags in a deployment group | 10 |
| Number of Auto Scaling groups in a deployment group | 10 |
| Number of characters in a deployment group name | 100 |
| Characters allowed in a deployment group name | The letter characters a through z, inclusive.<br><br>The letter characters A through Z, inclusive.<br><br>The number characters 0 through 9, inclusive.<br><br>The special characters + (plus sign), = (equals sign), , (comma), . (period), @ (at sign), - (minus sign), and _ (underscore). |

# Instances

| | |
|---|---|
| Number of instances in a single deployment | 50 |
| Number of characters in a tag key | 128 |
| Number of characters in a tag value | 256 |

# AWS CodeDeploy Resources

The following related resources can help you as you work with AWS CodeDeploy.

## Reference Guides and Support Resources

- **AWS CodeDeploy API Reference** — Descriptions, syntax, and usage examples about AWS CodeDeploy actions and data types, including common parameters and error codes.
- **AWS CodeDeploy Technical FAQs** — Top questions that customers have asked about AWS CodeDeploy.
- **AWS CodeDeploy Release Notes** — A high-level overview of the current and past releases, specifically notes about new features, corrections, and known issues.
- **AWS Support Center** — The hub for creating and managing your AWS Support cases. Also includes links to other helpful resources, such as forums, technical FAQs, service health status, and AWS Trusted Advisor.
- **AWS Premium Support** — The primary web page for information about AWS Support, a one-on-one, fast-response support channel to help you build and run applications in the cloud.
- **Contact Us** — A central contact point for inquiries concerning AWS billing, account, events, abuse, and other issues.
- **AWS Site Terms** — Detailed information about our copyright and trademark; your account, license, and site access; and other topics.

## Samples

- **AWS CodeDeploy Samples on GitHub** — Samples and template scenarios for AWS CodeDeploy.
- **AWS CodeDeploy Jenkins Plugin** — Jenkins plugin for AWS CodeDeploy.
- **AWS CodeDeploy Agent** — Open-source version of the AWS CodeDeploy Agent.

## Blogs

- **AWS Application Management Blog** — Insights for developers, sysadmins, and architects.

# AWS Software Development Kits and Tools

The following AWS SDKs and tools support solution development with AWS CodeDeploy:

- **AWS SDK for Java**
- **AWS SDK for JavaScript**
- **AWS SDK for .NET**
- **AWS SDK for PHP**
- **AWS SDK for Python (Boto)**
- **AWS SDK for Ruby**
- **AWS Toolkit for Eclipse** — Parts 1, 2, and 3.
- **AWS Tools for Windows PowerShell** — A set of Windows PowerShell cmdlets that expose the functionality of the AWS SDK for .NET in the PowerShell environment.
- **AWS CodeDeploy Cmdlets in the AWS Tools for PowerShell** — A set of Windows PowerShell cmdlets that expose the functionality of AWS CodeDeploy in the PowerShell environment.
- **AWS Command Line Interface** — A uniform command line syntax for accessing AWS services. The AWS CLI uses a single setup process to enable access for all supported services.
- **AWS CodeDeploy Command Line Reference** — A set of AWS CodeDeploy commands that can be run from the AWS CLI.
- **AWS Developer Tools** — Links to developer tools and resources that provide documentation, code samples, release notes, and other information to help you build innovative applications with AWS CodeDeploy and AWS.

# Document History

The following table describes the important changes to the documentation since the last release of the *AWS CodeDeploy User Guide*.

- **API version:** 2014-10-06
- **Latest documentation update:** May 7, 2015

| Change | Description | Date Changed |
| --- | --- | --- |
| Topic updates | AWS CodeDeploy is now available in the EU (Ireland) region (eu-west-1) and in the Asia Pacific (Sydney) region (ap-southeast-2). Several topics were updated to reflect the availability of these new regions, especially in the instructions for setting up the AWS CodeDeploy Agent. | May 7, 2015 |
| New topics | AWS CodeDeploy now supports deploying to on-premises instances as well as Amazon EC2 instances. The following topics were added to describe this new support:<br>• On-Premises Instances (p. 69)<br>• On-Premises Instance Deployment (Windows Server or Ubuntu Server) (p. 50)<br>• Use an Existing On-Premises Instance (p. 91) | April 2, 2015 |
| New topic | A new AWS CodeDeploy Resources (p. 243) topic was added. | April 2, 2015 |

| Change | Description | Date Changed |
| --- | --- | --- |
| Topic update | The Troubleshooting (p. 191) was updated:<br>• A new Long-Running Processes Can Cause Deployments to Fail (p. 196) section describes steps that you can take to identify and address deployment failures due to long-running processes.<br>• The General Auto Scaling Troubleshooting (p. 202) section was updated to show that AWS CodeDeploy has increased its Auto Scaling timeout logic for the AWS CodeDeploy Agent from 5 minutes to 1 hour.<br>• A new Mismatched Auto Scaling Lifecycle Hooks May Cause Automatic Deployments to Auto Scaling Groups to Stop or Fail (p. 204) section describes steps that you can take to identify and address failed automatic deployments to Auto Scaling groups by identifying information about Auto Scaling lifecycle hooks that are mismatched between AWS CodeDeploy and Auto Scaling. | April 2, 2015 |
| Topic updates | The following topics were updated to reflect new recommendations for creating your own custom policies and then attaching them to users and roles in IAM:<br>• Use an Existing Amazon EC2 Instance (p. 88)<br>• Create an IAM Instance Profile (p. 84)<br>• Create a Service Role (p. 140)<br>• Access Permissions Reference (p. 226)<br><br>Two new sections were added to Troubleshooting (p. 191):<br>• General Troubleshooting Checklist (p. 191)<br>• Windows PowerShell Scripts Fail to Use the 64-Bit Version of Windows PowerShell by Default (p. 195)<br><br>The hooks Section (p. 221) section in the AppSpec File Reference (p. 212) was updated to more accurately describe the available deployment lifecycle events. | February 12, 2015 |
| Topic updates | A new section was added to Troubleshooting (p. 191): Amazon EC2 Instances in an Auto Scaling Group Fail to Launch with the Error "Heartbeat Timeout" (p. 204).<br><br>A new CloudBees section was added to Service and Product Integrations with AWS CodeDeploy (p. 153). | January 28, 2015 |

| Change | Description | Date Changed |
| --- | --- | --- |
| Topic updates | New sections were added to Troubleshooting (p. 191), including the following:<br>• Using Certain Text Editors with AppSpec Files and Shell Scripts May Cause Associated Deployments to Fail (p. 193)<br>• Using Finder in MacOS to Manually Bundle an Application Revision May Cause Associated Deployments to Fail (p. 194)<br>• Troubleshooting a Failed ApplicationStop Deployment Lifecycle Event (p. 194)<br>• Troubleshooting a Failed DownloadBundle Deployment Lifecycle Event with "UnknownError: Not Opened for Reading" (p. 195)<br>• General Auto Scaling Troubleshooting (p. 202)<br><br>Information was added to the Create Deployment Walkthrough (p. 8) to clarify that certain permissions are needed for the calling IAM user to successfully complete the Create Deployment Walkthrough, specifically:<br>• Step 2: Instance Settings (p. 12) notes that certain permissions are needed to use the walkthrough's AWS CloudFormation template.<br>• Step 6: Service Role (p. 14) notes that certain permissions are needed to create a service role as part of the walkthrough.<br>• Step 8: Review (p. 15) notes that certain permissions are needed to create applications and deployment groups, as well as to deploy applications, as part of the walkthrough.<br><br>Each of these steps references permissions that are detailed as part of the walkthrough's prerequisites (p. 10). | January 20, 2015 |
| New topics | The new section Service and Product Integrations with AWS CodeDeploy (p. 153) was added, and six existing topics were moved from the existing AWS CodeDeploy Concepts (p. 58) section into this new section:<br>• Auto Scaling Integration (p. 157)<br>• Tutorial: Deploy to an Auto Scaling Group (p. 158)<br>• Logging AWS CodeDeploy API Calls By Using AWS CloudTrail (p. 174)<br>• Auto Scaling Integration (p. 176)<br>• GitHub Integration (p. 177)<br>• Tutorial: Deploy from GitHub (p. 180) | January 9, 2015 |

| Change | Description | Date Changed |
| --- | --- | --- |
| Topic updates | • The new section Automatically Deploy from GitHub with AWS CodeDeploy (p. 179) was added to GitHub Integration (p. 177). You can now automatically trigger a deployment from a GitHub repository whenever the source code in that repository changes.<br>• The new section Troubleshooting Auto Scaling Issues (p. 202) was added to Troubleshooting (p. 191). This new section describes how to troubleshoot common issues with deploying to Auto Scaling groups.<br>• The new subsection "files Examples" was added to the files Section (p. 213) section of AppSpec File Reference (p. 212). This new subsection includes several examples of how to use the `files` section of an AppSpec file to instruct AWS CodeDeploy to copy specific files or folders to specific locations on an instance during a deployment. | January 8, 2015 |
| New topic | The new topic Logging AWS CodeDeploy API Calls By Using AWS CloudTrail (p. 174) was added. AWS CodeDeploy is integrated with AWS CloudTrail, a service that captures API calls made by or on behalf of AWS CodeDeploy in your AWS account and delivers the log files to an Amazon S3 bucket that you specify. | December 17, 2014 |
| Topic update | The Step 2: Instance Settings (p. 12) section in Create Deployment Walkthrough (p. 8) was updated. The Create Deployment Walkthrough now shows AWS CloudFormation status by using a progress bar. Also, the **Next Step** button on the associated page in the walkthrough is disabled until the corresponding AWS CloudFormation stack is successfully created. | December 3, 2014 |
| Initial public release | This is the initial public release of the *AWS CodeDeploy User Guide*. | November 12, 2014 |

# AWS Glossary

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