
AWS CodeDeploy

User Guide

API Version 2014-10-06



AWS CodeDeploy: User Guide

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

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What Is AWS CodeDeploy?	1
Video Introduction to AWS CodeDeploy	1
Benefits of AWS CodeDeploy	1
Overview of AWS CodeDeploy Compute Platforms	2
Overview of AWS CodeDeploy Deployment Types	3
Overview of an In-Place Deployment	4
Overview of a Blue/Green Deployment	5
We Want to Hear from You	7
Primary Components	7
Deployments	9
Deployments on an AWS Lambda Compute Platform	9
Deployments on an EC2/On-Premises Compute Platform	13
Application Specification Files	17
AppSpec Files on an AWS Lambda Compute Platform	17
AppSpec Files on an EC2/On-Premises Compute Platform	17
How the AWS CodeDeploy Agent Uses the AppSpec File	18
Getting Started	19
Step 1: Provision an IAM User	19
Step 2: Install or Upgrade and Then Configure the AWS CLI	20
Step 3: Create a Service Role	21
Create a Service Role (Console)	21
Create a Service Role (CLI)	23
Get the Service Role ARN (Console)	24
Get the Service Role ARN (CLI)	25
Step 4: Create an IAM Instance Profile	25
Create an IAM Instance Profile for Your Amazon EC2 Instances (CLI)	25
Create an IAM Instance Profile for Your Amazon EC2 Instances (Console)	27
Step 5: Try the Sample Deployment Wizard	29
Prerequisites	30
Try a Sample Blue/Green Deployment in AWS CodeDeploy	31
Try a Sample In-Place Deployment in AWS CodeDeploy	35
Product and Service Integrations	40
Integration with Other AWS Services	40
Auto Scaling	44
Elastic Load Balancing	46
Integration with Partner Products and Services	49
GitHub	53
Integration Examples from the Community	55
Blog posts	56
Videos	56
Tutorials	57
Tutorial: Deploy WordPress to a Non-Windows Instance	57
Step 1: Launch an Amazon EC2 Instance	58
Step 2: Configure Your Source Content	60
Step 3: Upload Your Application to Amazon S3	63
Step 4: Deploy Your Application	67
Step 5: Update and Redeploy Your Application	71
Step 6: Clean Up	73
Tutorial: Deploy a HelloWorld Application to a Windows Server Instance	75
Step 1: Launch an Amazon EC2 Instance	76
Step 2: Configure Your Source Content	77
Step 3: Upload Your Application to Amazon S3	79
Step 4: Deploy Your Application	82
Step 5: Update and Redeploy Your Application	86

Step 6: Clean Up	88
Tutorial: Deploy an Application to an On-Premises Instance	90
Prerequisites	90
Step 1: Configure the On-Premises Instance	91
Step 2: Create a Sample Application Revision	91
Step 3: Bundle and Upload Your Application Revision to Amazon S3	94
Step 4: Deploy Your Application Revision	94
Step 5: Verify Your Deployment	95
Step 6: Clean Up Resources	95
Tutorial: Deploy to an Auto Scaling Group	96
Prerequisites	97
Step 1: Create and Configure the Auto Scaling Group	97
Step 2: Deploy the Application to the Auto Scaling Group	102
Step 3: Check Your Results	107
Step 4: Increase the Number of Amazon EC2 Instances in the Auto Scaling Group	108
Step 5: Check Your Results Again	109
Step 6: Clean Up	111
Tutorial: Deploy an Application from GitHub	112
Prerequisites	112
Step 1: Set Up a GitHub Account	113
Step 2: Create a GitHub Repository	113
Step 3: Upload a Sample Application to Your GitHub Repository	115
Step 4: Provision an Instance	117
Step 5: Create an Application and Deployment Group	118
Step 6: Deploy the Application to the Instance	119
Step 7: Monitor and Verify the Deployment	122
Step 8: Clean Up	123
Working with the AWS CodeDeploy Agent	125
Operating Systems Supported by the AWS CodeDeploy Agent	125
Supported Amazon EC2 AMI Operating Systems	125
Supported On-Premises Operating Systems	125
Communication Protocol and Port for the AWS CodeDeploy Agent	126
AWS SDK for Ruby (aws-sdk-core) Support for the AWS CodeDeploy Agent	126
Version History of the AWS CodeDeploy Agent	126
Application Revision and Log File Cleanup	129
Files Installed by the AWS CodeDeploy Agent	129
Managing AWS CodeDeploy Agent Operations	132
Verify the AWS CodeDeploy Agent Is Running	132
Determine the Version of the AWS CodeDeploy Agent	133
Install or Reinstall the AWS CodeDeploy Agent	134
Update the AWS CodeDeploy Agent	141
Uninstall the AWS CodeDeploy Agent	145
Working with Instances	147
Comparing Amazon EC2 Instances to On-Premises Instances	147
Instance Tasks for AWS CodeDeploy	148
Tagging Instances for AWS CodeDeploy Deployments	149
Example 1: Single Tag Group, Single Tag	150
Example 2: Single Tag Group, Multiple Tags	150
Example 3: Multiple Tag Groups, Single Tags	151
Example 4: Multiple Tag Groups, Multiple Tags	153
Working with Amazon EC2 Instances	156
Create an Amazon EC2 Instance (AWS CLI or Amazon EC2 Console)	156
Create an Amazon EC2 Instance (AWS CloudFormation Template)	162
Configure an Amazon EC2 Instance	168
Working with On-Premises Instances	171
Prerequisites for Configuring an On-Premises Instance	172
Register an On-Premises Instance	173

Managing On-Premises Instances Operations	193
View Instance Details	197
View Instance Details (Console)	198
View Instance Details (CLI)	198
Instance Health	198
Health Status	199
Minimum Healthy Instances and Deployments	200
Working with Deployment Configurations	203
Deployment Configurations on an EC2/On-Premises Compute Platform	203
Predefined Deployment Configurations	203
Deployment Configurations on an AWS Lambda Compute Platform	205
Predefined Deployment Configurations	206
.....	206
Create a Deployment Configuration	207
View Deployment Configuration Details	207
View Deployment Configuration Details (Console)	208
View Deployment Configuration (CLI)	208
Delete a Deployment Configuration	208
Working with Applications	209
Create an Application	209
Create an Application for an In-Place Deployment (Console)	211
Create an Application for a Blue/Green Deployment (Console)	212
Create an Application for an AWS Lambda Function Deployment (Console)	215
Create an Application (CLI)	216
View Application Details	216
View Application Details (Console)	217
View Application Details (CLI)	217
Rename an Application	217
Delete an Application	217
Delete an Application (Console)	218
Delete an Application (AWS CLI)	218
Working with Deployment Groups	219
Deployment Groups in AWS Lambda Compute Platform Deployments	219
Deployment Groups in EC2/On-Premises Compute Platform Deployments	219
.....	220
Create a Deployment Group	220
Create a Deployment Group for an In-Place Deployment (Console)	220
Create a Deployment Group for a Blue/Green Deployment (Console)	222
Set Up a Load Balancer in Elastic Load Balancing for AWS CodeDeploy Deployments	224
Create a Deployment Group (CLI)	225
View Deployment Group Details	226
View Deployment Group Details (Console)	226
View Deployment Group Details (CLI)	226
Change Deployment Group Settings	227
Change Deployment Group Settings (Console)	227
Change Deployment Group Settings (CLI)	228
Configure Advanced Options for a Deployment Group	228
Delete a Deployment Group	230
Delete a Deployment Group (Console)	230
Delete a Deployment Group (CLI)	230
Working with Application Revisions	232
Plan a Revision	232
Add an AppSpec File	233
Add an AppSpec File for an AWS Lambda Deployment	233
Add an AppSpec File for an EC2/On-Premises Deployment	234
Choose a Repository Type	237
Push a Revision	238

Push a Revision Using the AWS CLI	239
View Application Revision Details	241
View Application Revision Details (Console)	241
View Application Revision Details (CLI)	241
Register an Application Revision	242
Register a revision in Amazon S3 with AWS CodeDeploy (CLI)	242
Register a revision in GitHub with AWS CodeDeploy (CLI)	243
Working with Deployments	244
Create a Deployment	245
Deployment Prerequisites	245
Create an AWS Lambda Compute Platform Deployment (Console)	247
Create an EC2/On-Premises Compute Platform Deployment (Console)	248
Create an AWS Lambda Compute Platform Deployment (CLI)	251
Create an EC2/On-Premises Compute Platform Deployment (CLI)	252
View Deployment Details	254
View Deployment Details (Console)	254
View Deployment Details (CLI)	255
View Deployment Log Data	255
View Log File Data in the Amazon CloudWatch Console	256
View Log Files on an Instance	256
Stop a Deployment	258
Stop a deployment (console)	258
Stop a deployment (CLI)	258
Redeploy and Roll Back a Deployment	259
Automatic Rollbacks	259
Manual Rollbacks	259
Rollback and Redeployment Workflow	259
Rollback Behavior with Existing Content	260
Deploy an Application in a Different AWS Account	262
Step 1: Create an S3 Bucket in Either Account	263
Step 2: Grant Amazon S3 Bucket Permissions to the Production Account's IAM Instance Profile ..	263
Step 3: Create Resources and a Cross-Account Role in the Production Account	264
Step 4: Upload the Application Revision to Amazon S3 Bucket	265
Step 5: Assume the Cross-Account Role and Deploy Applications	265
Validate a Deployment Package on a Local Machine	265
Prerequisites	266
Create a Local Deployment	267
Examples	269
Monitoring Deployments	271
Automated Tools	271
Manual Tools	272
Monitoring Deployments with Amazon CloudWatch Tools	273
Monitoring Deployments with CloudWatch Alarms	273
Monitoring Deployments with Amazon CloudWatch Events	274
Monitoring Deployments with AWS CloudTrail	277
AWS CodeDeploy Information in CloudTrail	277
Understanding AWS CodeDeploy Log File Entries	277
Monitoring Deployments with Amazon SNS Event Notifications	278
Grant Amazon SNS Permissions to a Service Role	279
Create a Trigger for an AWS CodeDeploy Event	280
Edit a Trigger in a Deployment Group	285
Delete a Trigger from a Deployment Group	286
JSON Data Formats for Triggers	287
Authentication and Access Control	289
Authentication	289
Access Control	290
Overview of Managing Access	290

Resources and Operations	291
Understanding Resource Ownership	292
Managing Access to Resources	292
Specifying Policy Elements: Actions, Effects, and Principals	294
Specifying Conditions in a Policy	295
Using Identity-Based Policies (IAM Policies)	295
Permissions Required to Use the AWS CodeDeploy Console	296
AWS Managed (Predefined) Policies for AWS CodeDeploy	296
Customer Managed Policy Examples	297
AWS CodeDeploy Permissions Reference	299
AppSpec File Reference	306
AppSpec Files on an AWS Lambda Compute Platform	306
AppSpec Files on an EC2/On-Premises Compute Platform	306
AppSpec File Structure	307
AppSpec File Structure for AWS LambdaDeployments	307
AppSpec File Structure for EC2/On-Premises Deployments	308
AppSpec 'files' Section (EC2/On-Premises Deployments Only)	309
AppSpec 'resources' Section (AWS Lambda Deployments Only)	312
AppSpec 'permissions' Section (EC2/On-Premises Deployments Only)	312
AppSpec 'hooks' Section	316
AppSpec File Example	325
AppSpec File Example for an AWS Lambda Deployment	326
AppSpec File Example for an EC2/On-Premises Deployment	327
AppSpec File Spacing	327
Validate Your AppSpec File and File Location	328
Agent Configuration Reference	330
Related Topics	332
AWS CloudFormation Template Reference	333
Resource Kit Reference	335
Resource Kit Bucket Names by Region	335
Resource Kit Contents	336
Display a List of the Resource Kit Files	337
Download the Resource Kit Files	338
Limits	340
Applications	340
Application Revisions	340
Deployments	341
Deployment Configurations	342
Deployment Groups	342
Instances	343
Troubleshooting	344
General Troubleshooting Issues	344
General Troubleshooting Checklist	344
AWS CodeDeploy deployment resources are supported in certain regions only	345
Required IAM roles are not available	346
Avoid concurrent deployments to the same Amazon EC2 instance	346
Using some text editors to create AppSpec files and shell scripts can cause deployments to fail	347
Using Finder in macOS to bundle an application revision can cause deployments to fail	347
Troubleshoot EC2/On-Premises Deployment Issues	347
Deployment fails with the message "Validation of PKCS7 signed message failed"	348
Deployment or redeployment of the same files to the same instance locations fail with the error "The deployment failed because a specified file already exists at this location"	348
Troubleshooting a failed AllowTraffic lifecycle event with no error reported in the deployment logs	350
Troubleshooting failed ApplicationStop, BeforeBlockTraffic, and AfterBlockTraffic deployment lifecycle events	350

Troubleshooting a failed DownloadBundle deployment lifecycle event with "UnknownError: not opened for reading"	351
Windows PowerShell scripts fail to use the 64-bit version of Windows PowerShell by default	352
Long-running processes can cause deployments to fail	352
Troubleshoot AWS Lambda Deployment Issues	353
AWS Lambda deployments fail after manually stopping a Lambda deployment that does not have configured rollbacks	354
Troubleshoot Deployment Group Issues	354
Tagging an instance as part of a deployment group does not automatically deploy your application to the new instance	354
Troubleshoot Instance Issues	354
Tags must be set correctly	355
AWS CodeDeploy agent must be installed and running on instances	355
Deployments do not fail for up to an hour when an instance is terminated during a deployment	355
Analyzing log files to investigate deployment failures on instances	355
Create a new AWS CodeDeploy log file if it was accidentally deleted	355
Troubleshooting "InvalidSignatureException – Signature expired: [time] is now earlier than [time]" deployment errors	356
Troubleshoot GitHub Token Issues	356
Invalid GitHub OAuth token	356
Maximum Number of GitHub OAuth Tokens Exceeded	356
Troubleshoot Auto Scaling Issues	357
General Auto Scaling troubleshooting	357
Instances in an Auto Scaling group are continuously provisioned and terminated before a revision can be deployed	358
Terminating or rebooting an Auto Scaling instance may cause deployments to fail	358
Avoid associating multiple deployment groups with a single Auto Scaling group	359
Amazon EC2 instances in an Auto Scaling group fail to launch and receive the error "Heartbeat Timeout"	359
Mismatched Auto Scaling lifecycle hooks might cause automatic deployments to Auto Scaling groups to stop or fail	360
Error Codes	361
Related Topics	363
Resources	364
Reference Guides and Support Resources	364
Samples	364
Blogs	364
AWS Software Development Kits and Tools	364
Document History	366
Earlier Updates	366
AWS Glossary	380

What Is AWS CodeDeploy?

AWS CodeDeploy is a deployment service that automates application deployments to Amazon EC2 instances, on-premises instances, or serverless Lambda functions.

You can deploy a nearly unlimited variety of application content, such as code, serverless AWS Lambda functions, web and configuration files, executables, packages, scripts, multimedia files, and so on. AWS CodeDeploy can deploy application content that runs on a server and is stored in Amazon S3 buckets, GitHub repositories, or Bitbucket repositories. AWS CodeDeploy can also deploy a serverless Lambda function. You do not need to make changes to your existing code before you can use AWS CodeDeploy.

AWS CodeDeploy makes it easier for you to:

- Rapidly release new features.
- Update AWS Lambda function versions.
- Avoid downtime during application deployment.
- Handle the complexity of updating your applications, without many of the risks associated with error-prone manual deployments.

The service scales with your infrastructure so you can easily deploy to one instance or thousands.

AWS CodeDeploy works with various systems for configuration management, source control, [continuous integration](#), [continuous delivery](#), and continuous deployment. For more information, see [Product Integrations](#).

Topics

- [Video Introduction to AWS CodeDeploy \(p. 1\)](#)
- [Benefits of AWS CodeDeploy \(p. 1\)](#)
- [Overview of AWS CodeDeploy Compute Platforms \(p. 2\)](#)
- [Overview of AWS CodeDeploy Deployment Types \(p. 3\)](#)
- [We Want to Hear from You \(p. 7\)](#)
- [AWS CodeDeploy Primary Components \(p. 7\)](#)
- [AWS CodeDeploy Deployments \(p. 9\)](#)
- [AWS CodeDeploy Application Specification Files \(p. 17\)](#)

Video Introduction to AWS CodeDeploy

This short video (2:10) describes how AWS CodeDeploy automates code deployments to Amazon EC2 instances, making it easier for you to rapidly release new features, eliminate downtime during deployment, and avoid the need for error-prone, manual operations.

[Video Walkthrough of an AWS CodeDeploy Deployment.](#)

Benefits of AWS CodeDeploy

AWS CodeDeploy offers these benefits:

- **Server and serverless applications.** AWS CodeDeploy lets you deploy both traditional applications on servers and applications that deploy a serverless AWS Lambda function version.

- **Automated deployments.** AWS CodeDeploy fully automates your application deployments 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. During an in-place deployment, AWS CodeDeploy performs a rolling update across Amazon EC2 instances. You can specify the number of instances to be taken offline at a time for updates. During a blue/green deployment, the latest application revision is installed on replacement instances, and traffic is rerouted to these instances when you choose, either immediately or as soon as you are done testing the new environment. For both deployment types, AWS CodeDeploy tracks application health according to rules you configure.
- **Stop and roll back.** You can automatically or manually stop and roll back deployments if there are errors.
- **Centralized control.** You can launch and track the status of your deployments through the AWS CodeDeploy console or the AWS CLI. You receive a report that lists when each application revision was deployed and to which Amazon EC2 instances.
- **Easy to adopt.** AWS CodeDeploy is platform-agnostic and works with any application. You can easily reuse your setup code. AWS CodeDeploy can also integrate with your software release process or continuous delivery toolchain.

Overview of AWS CodeDeploy Compute Platforms

AWS CodeDeploy is able to deploy applications to two compute platforms:

- **EC2/On-Premises:** Describes instances of physical servers that can be Amazon EC2 cloud instances, on-premises servers, or both. Applications created using the EC2/On-Premises compute platform can be composed of executable files, configuration files, images, and more.

Deployments that use the EC2/On-Premises compute platform manage the way in which traffic is directed to instances by using an in-place or blue/green deployment type. For more information, see [Overview of AWS CodeDeploy Deployment Types \(p. 3\)](#).

- **AWS Lambda:** Used to deploy applications that consist of updated versions of Lambda functions. AWS Lambda manages the Lambda functions in a serverless compute environment made up of a high-availability compute structure. All administration of the compute resources is performed by AWS Lambda. For more information, see [Serverless Computing and Applications](#). For more information about AWS Lambda and Lambda functions, see [AWS Lambda](#).

Applications created using the AWS Lambda compute platform can manage the way in which traffic is directed to the updated Lambda function versions during a deployment by choosing a canary, linear, or all-at-once configuration.

The following table describes how AWS CodeDeploy components are used with each compute platform.

AWS CodeDeploy Component	EC2/On-Premises	AWS Lambda
Deployment group	Deploys a set of instances to which a new revision is deployed.	Deploys a Lambda function version on a high-availability compute infrastructure.
Deployment	Deploys a new revision that consists of an application and AppSpec file. The AppSpec specifies how to deploy the application to the instances in a deployment group.	Deploys a new revision that consists of an AppSpec file. The AppSpec specifies which Lambda function version to deploy.

AWS CodeDeploy Component	EC2/On-Premises	AWS Lambda
Deployment configuration	Settings that determine the deployment speed and the minimum number of instances that must be healthy at any point during a deployment.	Settings that determine how traffic is shifted to the updated Lambda function versions.
Revision	A combination of an AppSpec file and application files, such as executables, configuration files, and so on.	An AppSpec file that specifies which Lambda functions to deploy and update.
Application	A collection of deployment groups and revisions. An EC2/On-Premises application uses the EC2/On-Premises compute platform.	A collection of revisions. A Lambda application uses the AWS Lambda compute platform.

Overview of AWS CodeDeploy Deployment Types

AWS CodeDeploy provides two deployment type options:

- **In-place deployment:** The application on each instance in the deployment group is stopped, the latest application revision is installed, and the new version of the application is started and validated. You can use a load balancer so that each instance is deregistered during its deployment and then restored to service after the deployment is complete. Only deployments that use the EC2/On-Premises compute platform can use in-place deployments. For more information about in-place deployments, see [Overview of an In-Place Deployment \(p. 4\)](#).

Note

AWS Lambda compute platform deployments cannot use an in-place deployment type.

- **Blue/green deployment:** The behavior of your deployment depends on which compute platform you use:
 - **Blue/green on an EC2/On-Premises compute platform:** The instances in a deployment group (the original environment) are replaced by a different set of instances (the replacement environment) using these steps:
 - Instances are provisioned for the replacement environment.
 - The latest application revision is installed on the replacement instances.
 - An optional wait time occurs for activities such as application testing and system verification.
 - Instances in the replacement environment are registered with an Elastic Load Balancing load balancer, causing traffic to be rerouted to them. Instances in the original environment are deregistered and can be terminated or kept running for other uses.

Note

When using an EC2/On-Premises compute platform, blue/green deployments work with Amazon EC2 instances only.

- **Blue/green on an AWS Lambda compute platform:** Traffic is shifted from your current serverless environment to one with your updated Lambda function versions. You can specify Lambda functions that perform validation tests and choose the way in which the traffic shift occurs. All AWS Lambda compute platform deployments are blue/green deployments. For this reason, you do not need to specify a deployment type.

For more information about blue/green deployments, see [Overview of a Blue/Green Deployment \(p. 5\)](#).

Note

Using the AWS CodeDeploy agent, you can perform a deployment on an instance you are logged on to without the need for an application, deployment group, or even an AWS account. For information, see [Use the AWS CodeDeploy Agent to Validate a Deployment Package on a Local Machine \(p. 265\)](#).

Topics

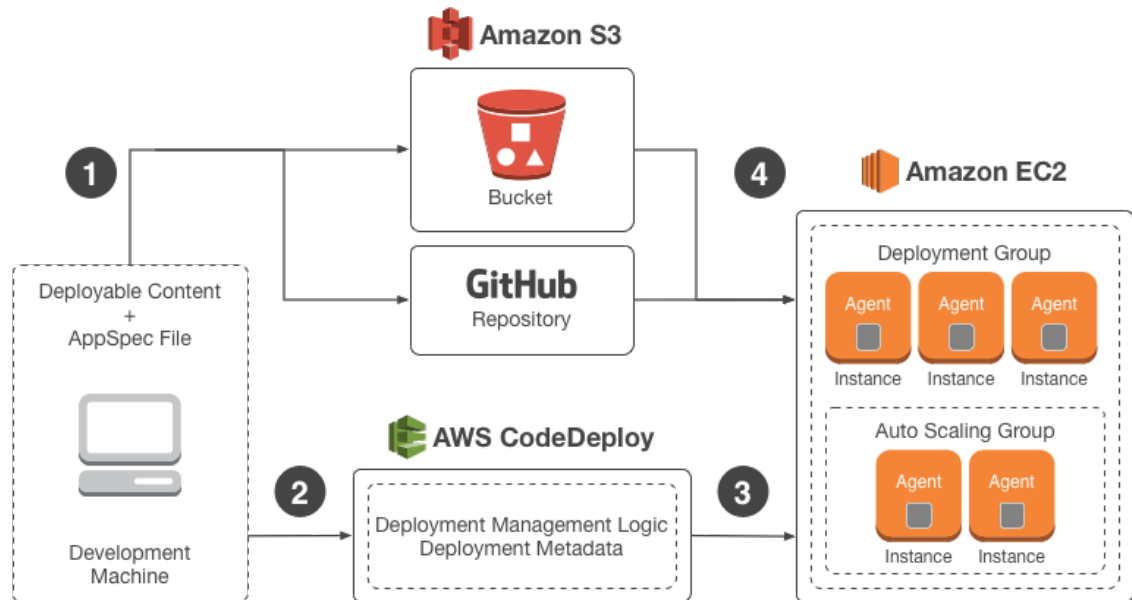
- [Overview of an In-Place Deployment \(p. 4\)](#)
- [Overview of a Blue/Green Deployment \(p. 5\)](#)

Overview of an In-Place Deployment

The following diagram shows the flow of a typical AWS CodeDeploy in-place deployment.

Note

AWS Lambda compute platform deployments cannot use an in-place deployment type.



Here's how it works:

1. First, you create deployable content on your local development machine or similar environment, and then you add an *application specification file* (AppSpec file). The AppSpec file is unique to AWS CodeDeploy. It defines the deployment actions you want AWS CodeDeploy to execute. You bundle your deployable content and the AppSpec file into an archive file, and then upload it to an Amazon S3 bucket or a GitHub repository. This archive file is called an *application revision* (or simply a *revision*).
2. Next, you provide AWS CodeDeploy with information about your deployment, such as which Amazon S3 bucket or GitHub repository to pull the revision from and to which set of Amazon EC2 instances to deploy its contents. AWS CodeDeploy calls a set of Amazon EC2 instances a *deployment group*. A deployment group contains individually tagged Amazon EC2 instances, Amazon EC2 instances in Auto Scaling groups, or both.

Each time you successfully upload a new application revision that you want to deploy to the deployment group, that bundle is set as the *target revision* for the deployment group. In other words, the application revision that is currently targeted for deployment is the target revision. This is also the revision that will be pulled for automatic deployments.

3. Next, the AWS CodeDeploy agent on each instance polls AWS CodeDeploy to determine what and when to pull from the specified Amazon S3 bucket or GitHub repository.
4. Finally, the AWS CodeDeploy agent on each instance pulls the target revision from the specified Amazon S3 bucket or GitHub repository and, using the instructions in the AppSpec file, deploys the contents to the instance.

AWS CodeDeploy keeps a record of your deployments so that you can get deployment status, deployment configuration parameters, instance health, and so on.

Overview of a Blue/Green Deployment

A blue/green deployment, in which traffic is rerouted from one set of instances (the original environment) to a different set (the replacement environment), offers a number of advantages over an in-place deployment:

- An application can be installed and tested on the new instances ahead of time and deployed to production simply by switching traffic to the new servers.
- Switching back to the most recent version of an application is faster and more reliable because traffic can be routed back to the original instances as long as they have not been terminated. With an in-place deployment, versions must be rolled back by redeploying the previous version of the application.
- If you're using the EC2/On-Premises compute platform, new instances are provisioned for a blue/green deployment and reflect the most up-to-date server configurations. This helps you avoid the types of problems that sometimes occur on long-running instances.
- If you're using the AWS Lambda compute platform, you control how traffic is shifted from your original AWS Lambda function versions to your new AWS Lambda function versions.

How you configure a blue/green deployment depends on which compute platform your deployment is using.

Blue/Green Deployment on an AWS Lambda Compute Platform

If you're using the AWS Lambda compute platform, you must choose one of the following deployment configuration types to specify how traffic is shifted from the original AWS Lambda function version to the new AWS Lambda function version:

- **Canary:** Traffic is shifted in two increments. You can choose from predefined canary options that specify the percentage of traffic shifted to your updated Lambda function version in the first increment and the interval, in minutes, before the remaining traffic is shifted in the second increment.
- **Linear:** Traffic is shifted in equal increments with an equal number of minutes between each increment. You can choose from predefined linear options that specify the percentage of traffic shifted in each increment and the number of minutes between each increment.
- **All-at-once:** All traffic is shifted from the original Lambda function to the updated Lambda function version at once.

For more information about AWS Lambda deployment configurations, see [Predefined Deployment Configurations for an AWS Lambda Compute Platform](#) (p. 206).

Blue/Green Deployment on an EC2/On-Premises Compute Platform

Note

You must use Amazon EC2 instances for blue/green deployments on the EC2/On-Premises compute platform. On-premises instances are not supported for the blue/green deployment type.

If you're using the EC2/On-Premises compute platform, the following applies:

You must have one or more Amazon EC2 instances with identifying Amazon EC2 tags or an Auto Scaling group. The instances must meet these additional requirements:

- Each Amazon EC2 instance must have the correct IAM instance profile attached.
- The AWS CodeDeploy agent must be installed and running on each instance.

Note

You typically also have an application revision running on the instances in your original environment, but this is not a requirement for a blue/green deployment.

When you create a deployment group that is used in blue/green deployments, you can choose how your replacement environment is specified:

Copy an existing Auto Scaling group: During the blue/green deployment, AWS CodeDeploy creates the instances for your replacement environment automatically during the deployment. With this option, AWS CodeDeploy uses the Auto Scaling group you specify as a template for the replacement environment, including the same number of running instances and many other configuration options.

Choose instances manually: You can specify the instances to be counted as your replacement using Amazon EC2 instance tags, Auto Scaling group names, or both. If you choose this option, you do not need to specify the instances for the replacement environment until you create a deployment.

Here's how it works:

1. You already have instances or an Auto Scaling group that will serve as your original environment. The first time you run a blue/green deployment, you'll typically use instances that were already used in an in-place deployment.
2. In an existing AWS CodeDeploy application, you create a blue/green deployment group where, in addition to the options required for an in-place deployment, you specify the following:
 - The load balancer that will route traffic from your original environment to your replacement environment during the blue/green deployment process.
 - Whether to reroute traffic to the replacement environment immediately or wait for you to reroute it manually.
 - The rate at which traffic is routed to the replacement instances.
 - Whether the instances that are replaced are terminated or kept running.
3. You create a deployment for this deployment group during which the following occur:
 - a. If you chose to copy an Auto Scaling group, instances are provisioned for your replacement environment.
 - b. The application revision you specify for the deployment is installed on the replacement instances.
 - c. If you specified a wait time in the deployment group settings, the deployment is paused. This is the time when you can run tests and verifications in your replacement environment. If you don't manually reroute the traffic before the end of the wait period, the deployment is stopped.
 - d. Instances in the replacement environment are registered with an Elastic Load Balancing load balancer and traffic begins to be routed to them.

- e. Instances in the original environment are deregistered and handled according to your specification in the deployment group, either terminated or kept running.

We Want to Hear from You

We welcome your feedback. To contact us, visit [the AWS CodeDeploy forum](#).

Topics

- [Primary Components \(p. 7\)](#)
- [Deployments \(p. 9\)](#)
- [Application Specification Files \(p. 17\)](#)

AWS CodeDeploy Primary Components

Before you start working with the service, you should familiarize yourself with the major components of the AWS CodeDeploy deployment process.

Application: A name that uniquely identifies the application you want to deploy. AWS CodeDeploy uses this name, which functions as a container, to ensure the correct combination of revision, deployment configuration, and deployment group are referenced during a deployment.

Compute platform: The platform on which AWS CodeDeploy deploys an application.

- **EC2/On-Premises:** Describes instances of physical servers that can be Amazon EC2 cloud instances, on-premises servers, or both. Applications created using the EC2/On-Premises compute platform can be composed of executable files, configuration files, images, and more.

Deployments that use the EC2/On-Premises compute platform manage the way in which traffic is directed to instances by using an in-place or blue/green deployment type. For more information, see [Overview of AWS CodeDeploy Deployment Types \(p. 3\)](#).

- **AWS Lambda:** Used to deploy applications that consist of updated versions of Lambda functions. AWS Lambda manages the Lambda functions in a serverless compute environment made up of a high-availability compute structure. All administration of the compute resources is performed by AWS Lambda. For more information, see [Serverless Computing and Applications](#). For more information about AWS Lambda and Lambda functions, see [AWS Lambda](#).

Applications created using the AWS Lambda compute platform can manage the way in which traffic is directed to the updated Lambda function versions during a deployment by choosing a canary, linear, or all-at-once configuration.

Deployment configuration: A set of deployment rules and deployment success and failure conditions used by AWS CodeDeploy during a deployment. If your deployment uses the EC2/On-Premises compute platform, you can specify the minimum number of healthy instances for the deployment. If your deployment uses the AWS Lambda compute platform, you can specify how traffic is routed to your updated Lambda function versions.

For more information about specifying the minimum number of healthy hosts for a deployment that uses the EC2/On-Premises compute platform, see [Minimum Healthy Instances and Deployments \(p. 200\)](#).

These are the deployment configurations that specify how traffic is routed during a deployment that uses the AWS Lambda compute platform:

- **Canary:** Traffic is shifted in two increments. You can choose from predefined canary options that specify the percentage of traffic shifted to your updated Lambda function version in the first increment and the interval, in minutes, before the remaining traffic is shifted in the second increment.
- **Linear:** Traffic is shifted in equal increments with an equal number of minutes between each increment. You can choose from predefined linear options that specify the percentage of traffic shifted in each increment and the number of minutes between each increment.
- **All-at-once:** All traffic is shifted from the original Lambda function to the updated Lambda function version at once.

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 about Amazon EC2 instance tags, see [Working with Tags Using the Console](#). For information about on-premises instances, see [Working with On-Premises Instances \(p. 171\)](#). For information about Auto Scaling, see [Integrating AWS CodeDeploy with Auto Scaling \(p. 44\)](#).

Deployment type: The method used to make the latest application revision available on instances in a deployment group.

- **In-place deployment:** The application on each instance in the deployment group is stopped, the latest application revision is installed, and the new version of the application is started and validated. You can use a load balancer so that each instance is deregistered during its deployment and then restored to service after the deployment is complete. Only deployments that use the EC2/On-Premises compute platform can use in-place deployments. For more information about in-place deployments, see [Overview of an In-Place Deployment \(p. 4\)](#).
- **Blue/green deployment:** The behavior of your deployment depends on which compute platform you use:
 - **Blue/green on an EC2/On-Premises compute platform:** The instances in a deployment group (the original environment) are replaced by a different set of instances (the replacement environment) using these steps:
 - Instances are provisioned for the replacement environment.
 - The latest application revision is installed on the replacement instances.
 - An optional wait time occurs for activities such as application testing and system verification.
 - Instances in the replacement environment are registered with an Elastic Load Balancing load balancer, causing traffic to be rerouted to them. Instances in the original environment are deregistered and can be terminated or kept running for other uses.

Note

When using an EC2/On-Premises compute platform, blue/green deployments work with Amazon EC2 instances only.

- **Blue/green on an AWS Lambda compute platform:** Traffic is shifted from your current serverless environment to one with your updated Lambda function versions. You can specify Lambda functions that perform validation tests and choose the way in which the traffic shift occurs. All AWS Lambda compute platform deployments are blue/green deployments. For this reason, you do not need to specify a deployment type.

For more information about blue/green deployments, see [Overview of a Blue/Green Deployment \(p. 5\)](#).

IAM instance profile: An IAM role that you attach to your Amazon EC2 instances. This profile includes the permissions required to access the Amazon S3 buckets or GitHub repositories where the applications that will be deployed by AWS CodeDeploy are stored. For more information, see [Step 4: Create an IAM Instance Profile for Your Amazon EC2 Instances \(p. 25\)](#).

Revision: An AWS Lambda deployment revision is a YAML-formatted or JSON-formatted file that specifies information about the Lambda function to deploy. An EC2/On-Premises deployment revision is

an archive file that contains source content—source code, web pages, executable files, and deployment scripts—along with an application specification file (AppSpec file). AWS Lambda revisions can be stored in Amazon S3 buckets. EC2/On-Premises 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.

Service role: An IAM role that grants permissions to an AWS service so it can access AWS resources. The policies you attach to the service role determine which AWS resources the service can access and the actions it can perform with those resources. For AWS CodeDeploy, a service role is used for the following:

- To read either the tags applied to the instances or the Amazon EC2 Auto Scaling group names associated with the instances. This enables AWS CodeDeploy to identify instances to which it can deploy applications.
- To perform operations on instances, Auto Scaling groups, and Elastic Load Balancing load balancers.
- To publish information to Amazon SNS topics so that notifications can be sent when specified deployment or instance events occur.
- To retrieve information about CloudWatch alarms in order to set up alarm monitoring for deployments.

For more information, see [Step 3: Create a Service Role for AWS CodeDeploy \(p. 21\)](#).

Target revision: The most recent version of the application revision that you have uploaded to your repository and want to deploy to the instances in a deployment group. In other words, the application revision currently targeted for deployment is the target revision. This is also the revision that will be pulled for automatic deployments.

For information about other major components in the AWS CodeDeploy workflow, see the following topics:

- [Choose an AWS CodeDeploy Repository Type \(p. 237\)](#)
- [Deployments \(p. 9\)](#)
- [Application Specification Files \(p. 17\)](#)
- [Instance Health \(p. 198\)](#)
- [Working with the AWS CodeDeploy Agent \(p. 125\)](#)
- [Working with On-Premises Instances \(p. 171\)](#)

AWS CodeDeploy Deployments

This topic provides information about the components and workflow of deployments in AWS CodeDeploy. The deployment process varies, depending on the compute platform (EC2/On-Premises or Lambda) you use for your deployments.

Topics

- [Deployments on an AWS Lambda Compute Platform \(p. 9\)](#)
- [Deployments on an EC2/On-Premises Compute Platform \(p. 13\)](#)

Deployments on an AWS Lambda Compute Platform

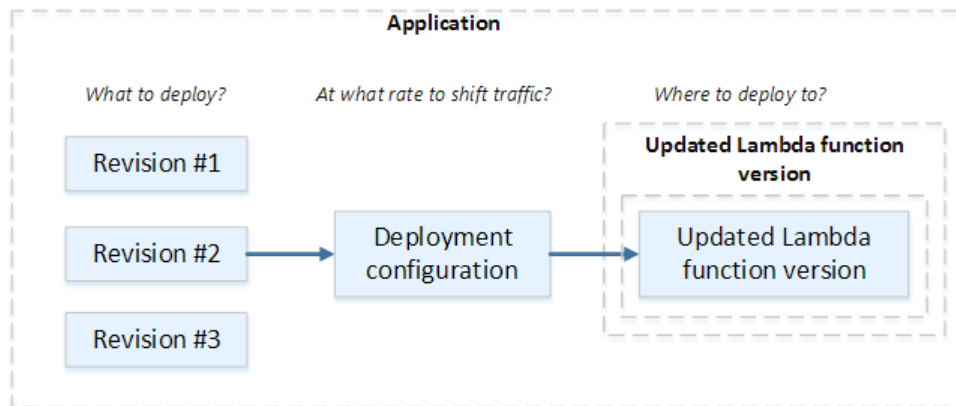
This topic provides information about the components and workflow of AWS CodeDeploy deployments that use the AWS Lambda compute platform.

Topics

- [Deployment Components on an AWS Lambda Compute Platform \(p. 10\)](#)
- [Deployment Workflow on an AWS Lambda Compute Platform \(p. 10\)](#)
- [Uploading Your Application Revision \(p. 11\)](#)
- [Creating Your Application and Deployment Groups \(p. 12\)](#)
- [Deploying Your Application Revision \(p. 12\)](#)
- [Updating Your Application \(p. 12\)](#)
- [Stopped and Failed Deployments \(p. 12\)](#)
- [Redeployments and Deployment Rollbacks \(p. 12\)](#)

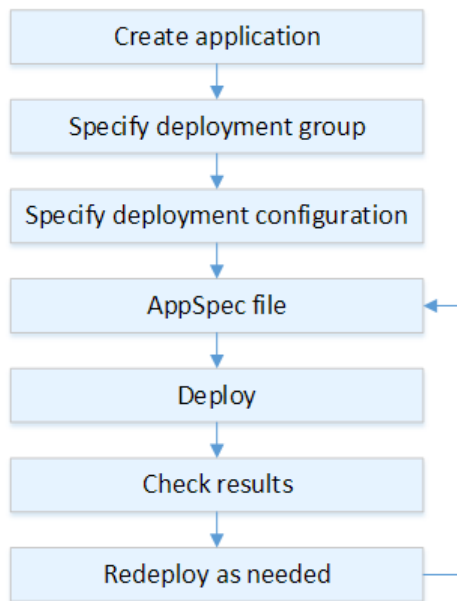
Deployment Components on an AWS Lambda Compute Platform

The following diagram shows the components in an AWS CodeDeploy deployment on an AWS Lambda compute platform.



Deployment Workflow on an AWS Lambda Compute Platform

The following diagram shows the major steps in the deployment of new and updated AWS Lambda functions.



These steps include:

1. Creating an application by specifying a name that uniquely represents the application revisions you want to deploy. To deploy Lambda functions, you specify the AWS Lambda compute platform when you create your application. AWS CodeDeploy uses this name during a deployment to make sure it is referencing the correct deployment components, such as the deployment group, deployment configuration, and application revision. For more information, see [Create an Application with AWS CodeDeploy \(p. 209\)](#).
2. Setting up a deployment group by specifying your deployment group's name.
3. Choosing a deployment configuration to specify how traffic is shifted from your original AWS Lambda function version to your new Lambda function version. For more information, see [View Deployment Configuration Details \(p. 207\)](#).
4. Uploading an *application specification file* (AppSpec file) to Amazon S3. The AppSpec file specifies a Lambda function version and Lambda functions used to validate your deployment. If you don't want to create an AppSpec file, you can specify a Lambda function version and Lambda deployment validation functions directly in the console using YAML or JSON. For more information, see [Working with Application Revisions for AWS CodeDeploy \(p. 232\)](#).
5. Deploying your application revision to the deployment group. AWS CodeDeploy deploys the Lambda function revision you specified. The traffic is shifted to your Lambda function revision using the deployment AppSpec file you chose when you created your application. For more information, see [Create a Deployment with AWS CodeDeploy \(p. 245\)](#).
6. Checking the deployment results. For more information, see [Monitoring Deployments in AWS CodeDeploy \(p. 271\)](#).
7. Redeploying a revision. You might want to do this if you need to fix a bug in a Lambda function in your deployment or address a failed deployment. If you redeploy the same Lambda function version, then you simply execute a new deployment to the same deployment group with the same AppSpec file. If you rename, add, or remove a Lambda function before you redeploy, you must update your AppSpec file. For more information, see [Create a Deployment with AWS CodeDeploy \(p. 245\)](#).

Uploading Your Application Revision

Place an AppSpec file in Amazon S3 or enter it directly into the console or AWS CLI. For more information, see [Application Specification Files \(p. 17\)](#).

Creating Your Application and Deployment Groups

An AWS CodeDeploy deployment group on an AWS Lambda compute platform identifies a collection of one or more AppSpec files. Each AppSpec file can deploy one Lambda function version. A deployment group also defines a set of configuration options for future deployments, such as alarms and rollback configurations.

Deploying Your Application Revision

Now you're ready to deploy the function revision specified in the AppSpec file to the deployment group. You can use the AWS CodeDeploy console or the [create-deployment](#) command. There are parameters you can specify to control your deployment, including the revision, deployment group, and deployment configuration.

Updating Your Application

You can make updates to your application and then use the AWS CodeDeploy console or call the [create-deployment](#) command to push a revision.

Stopped and Failed Deployments

You can use the AWS CodeDeploy console or the [stop-deployment](#) command to stop a deployment. When you attempt to stop the deployment, one of three things happens:

- The deployment stops, and the operation returns a status of succeeded. In this case, no more deployment lifecycle events are run on the deployment group for the stopped deployment.
- The deployment does not immediately stop, and the operation returns a status of pending. In this case, some deployment lifecycle events might still be running on the deployment group. After the pending operation is complete, subsequent calls to stop the deployment return a status of succeeded.
- The deployment cannot stop, and the operation returns an error. For more information, see [ErrorInformation](#) and [Common Errors](#) in the AWS CodeDeploy API Reference.

Like stopped deployments, failed deployments might result in some deployment lifecycle events having already been run. To find out why a deployment failed, you can use the AWS CodeDeploy console or analyze the log file data from the failed deployment. For more information, see [Application Revision and Log File Cleanup](#) (p. 129) and [View Log Data for AWS CodeDeploy Deployments](#) (p. 255).

Redeployments and Deployment Rollbacks

AWS CodeDeploy implements rollbacks by redeploying, as a new deployment, a previously deployed revision.

You can configure a deployment group to automatically roll back deployments when certain conditions are met, including when a deployment fails or an alarm monitoring threshold is met. You can also override the rollback settings specified for a deployment group in an individual deployment.

You can also choose to roll back a failed deployment by manually redeploying a previously deployed revision.

In all cases, the new or rolled-back deployment is assigned its own deployment ID. The list of deployments you can view in the AWS CodeDeploy console shows which ones are the result of an automatic deployment.

For more information, see [Redeploy and Roll Back a Deployment with AWS CodeDeploy](#) (p. 259).

Deployments on an EC2/On-Premises Compute Platform

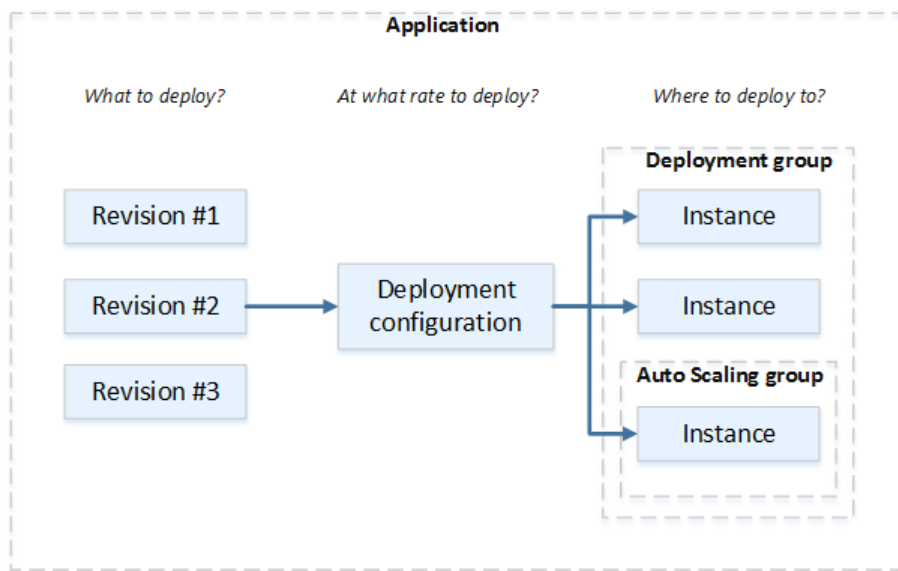
This topic provides information about the components and workflow of AWS CodeDeploy deployments that use the EC2/On-Premises compute platform. For information about blue/green deployments, see [Overview of a Blue/Green Deployment \(p. 5\)](#)

Topics

- [Deployment Components on an EC2/On-Premises Compute Platform \(p. 13\)](#)
- [Deployment Workflow on an EC2/On-Premises Compute Platform \(p. 13\)](#)
- [Setting Up Instances \(p. 15\)](#)
- [Uploading Your Application Revision \(p. 16\)](#)
- [Creating Your Application and Deployment Groups \(p. 16\)](#)
- [Deploying Your Application Revision \(p. 16\)](#)
- [Updating Your Application \(p. 16\)](#)
- [Stopped and Failed Deployments \(p. 16\)](#)
- [Redeployments and Deployment Rollbacks \(p. 17\)](#)

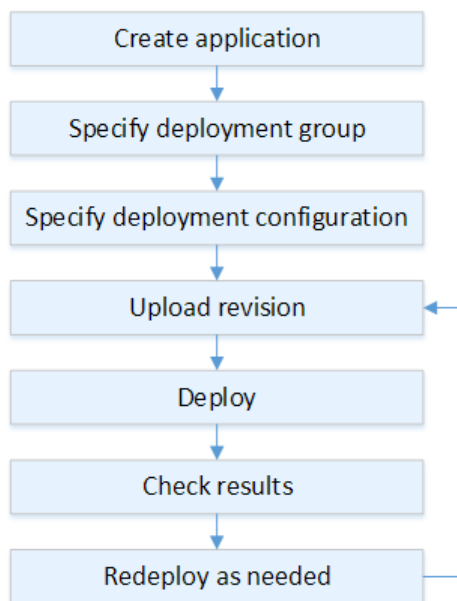
Deployment Components on an EC2/On-Premises Compute Platform

The following diagram shows the components in an AWS CodeDeploy deployment on an EC2/On-Premises compute platform.



Deployment Workflow on an EC2/On-Premises Compute Platform

The following diagram shows the major steps in the deployment of application revisions:



These steps include:

1. Creating an application by specifying a name that uniquely represents the application revisions you want to deploy and the compute platform for your application. AWS CodeDeploy uses this name during a deployment to make sure it is referencing the correct deployment components, such as the deployment group, deployment configuration, and application revision. For more information, see [Create an Application with AWS CodeDeploy \(p. 209\)](#).
2. Setting up a deployment group by specifying a deployment type and the instances to which you want to deploy your application revisions. An in-place deployment updates instances with the latest application revision. A blue/green deployment registers a replacement set of instances for the deployment group with a load balancer and deregisters the original instances.

You can specify the tags applied to the instances, the Auto Scaling group names, or both.

If you specify one group of tags in a deployment group, AWS CodeDeploy deploys to instances that have at least one of the specified tags applied. If you specify two or more tag groups, AWS CodeDeploy deploys only to the instances that meet the criteria for each of the tag groups. For more information, see [Tagging Instances for AWS CodeDeploy Deployments \(p. 149\)](#).

In all cases, the instances must be configured to be used in a deployment (that is, they must be 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 an Amazon EC2 instance based on Amazon Linux or Windows Server. We also provide you with the standalone AWS CodeDeploy agent so that you can install it on Amazon Linux, Ubuntu Server, Red Hat Enterprise Linux (RHEL), or Windows Server instances. For more information, see [Create a Deployment Group with AWS CodeDeploy \(p. 220\)](#).

You can also specify the following options:

- **Amazon SNS notifications** — Create triggers that will send notifications to subscribers of an Amazon SNS topic when specified events, such as success or failure events, occur in deployments and instances. For more information, see [Monitoring Deployments with Amazon SNS Event Notifications \(p. 278\)](#).
- **Alarm-based deployment management** — Implement Amazon CloudWatch alarm monitoring to stop deployments when your metrics exceed or fall below the thresholds set in CloudWatch.

- **Automatic deployment rollbacks** — Configure a deployment to roll back automatically to the previously known good revision when a deployment fails or an alarm threshold is met.
3. Specifying a deployment configuration to indicate to how many instances to simultaneously deploy your application revisions and describing the success and failure conditions for the deployment. For more information, see [View Deployment Configuration Details \(p. 207\)](#).
 4. Uploading an application revision to Amazon S3 or GitHub. In addition to the files you want to deploy and any scripts you want to run during the deployment, you must include an *application specification file* (AppSpec file). This file contains deployment instructions, such as where to copy the files onto each instance and when to run deployment scripts. For more information, see [Working with Application Revisions for AWS CodeDeploy \(p. 232\)](#).
 5. Deploying your application revision to the deployment group. The AWS CodeDeploy agent on each 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 using the AppSpec file, copies the files into the specified locations and executes any deployment scripts. For more information, see [Create a Deployment with AWS CodeDeploy \(p. 245\)](#).
 6. Checking the deployment results. For more information, see [Monitoring Deployments in AWS CodeDeploy \(p. 271\)](#).
 7. Redeploying a revision. You might want to do this if you need to fix a bug in the source content, or run the deployment scripts in a different order, or address a failed deployment. To do this, you rebundle your revised source content, any deployment scripts, and the AppSpec file into a new revision, and then upload the revision to the Amazon S3 bucket or GitHub repository. You then execute a new deployment to the same deployment group with the new revision. For more information, see [Create a Deployment with AWS CodeDeploy \(p. 245\)](#).

Setting Up Instances

You need to set up instances before you can deploy application revisions for the first time. If an application revision requires three production servers and two backup servers, you will launch or use five instances.

To manually provision instances:

1. Install the AWS CodeDeploy agent on the instances. The AWS CodeDeploy agent can be installed on Amazon Linux, Ubuntu Server, RHEL, and Windows Server instances.
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. Although the Getting Started tutorials used both, you can simply use a key or a value to define a tag for a deployment group.
3. Launch Amazon EC2 instances with an IAM instance profile attached. The IAM instance profile must be attached to an Amazon EC2 instance as it is launched in order for the AWS CodeDeploy agent to verify the identity of the instance.
4. Create a service role. Provide service access so that AWS CodeDeploy can expand the tags in your AWS account.

For an initial deployment, the AWS CloudFormation template does all of this for you. It creates and configures new, single Amazon EC2 instances based on Amazon Linux or Windows Server with the AWS CodeDeploy agent already installed. For more information, see [Working with Instances for AWS CodeDeploy \(p. 147\)](#).

Note

For a blue/green deployment, you can choose between using instances you already have for the replacement environment or letting AWS CodeDeploy provision new instances for you as part of the deployment process.

Uploading Your Application Revision

Place an AppSpec file under the root folder in your application's source content folder structure. For more information, see [Application Specification Files \(p. 17\)](#).

Bundle the application's source content folder structure into an archive file format such as zip, tar, or compressed tar. Upload the archive file (the *revision*) to an Amazon S3 bucket or GitHub repository.

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, Auto Scaling group names, or both. Multiple application revisions can be deployed to the same instance. An application revision can be deployed to multiple instances.

For example, you could add a tag of "Prod" to the three production servers and "Backup" to the two backup servers. These two tags can be used to create two different deployment groups in the AWS CodeDeploy application, allowing you to choose which set of servers (or both) should participate in a deployment.

You can use multiple tag groups in a deployment group to restrict deployments to a smaller set of instances. For information, see [Tagging Instances for AWS CodeDeploy Deployments \(p. 149\)](#).

Deploying Your Application Revision

Now you're ready to deploy your application revision from Amazon S3 or GitHub to the deployment group. You can use the AWS CodeDeploy console or the [create-deployment](#) command. There are parameters you can specify to control your deployment, including the revision, deployment group, and deployment configuration.

Updating Your Application

You can make updates to your application and then use the AWS CodeDeploy console or call the [create-deployment](#) command to push a revision.

Stopped and Failed Deployments

You can use the AWS CodeDeploy console or the [stop-deployment](#) command to stop a deployment. When you attempt to stop the deployment, one of three things will happen:

- The deployment stops, and the operation returns a status of succeeded. In this case, no more deployment lifecycle events are run on the deployment group for the stopped deployment. Some files might have already been copied to, and some scripts might have already been run on, one or more of the instances in the deployment group.
- The deployment does not immediately stop, and the operation returns a status of pending. In this case, some deployment lifecycle events might still be running on the deployment group. Some files might have already been copied to, and some scripts might have already been run on, one or more of the instances in the deployment group. After the pending operation is complete, subsequent calls to stop the deployment return a status of succeeded.
- The deployment cannot stop, and the operation returns an error. For more information, see [ErrorInformation](#) and [Common Errors](#) in the AWS CodeDeploy API Reference.

Like stopped deployments, failed deployments might result in some deployment lifecycle events having already been run on one or more of the instances in the deployment group. To find out why

a deployment failed, you can use the AWS CodeDeploy console, call the [get-deployment-instance](#) command, or analyze the log file data from the failed deployment. For more information, see [Application Revision and Log File Cleanup](#) (p. 129) and [View Log Data for AWS CodeDeploy Deployments](#) (p. 255).

Redeployments and Deployment Rollbacks

AWS CodeDeploy implements rollbacks by redeploying, as a new deployment, a previously deployed revision.

You can configure a deployment group to automatically roll back deployments when certain conditions are met, including when a deployment fails or an alarm monitoring threshold is met. You can also override the rollback settings specified for a deployment group in an individual deployment.

You can also choose to roll back a failed deployment by manually redeploying a previously deployed revision.

In all cases, the new or rolled-back deployment is assigned its own deployment ID. The list of deployments you can view in the AWS CodeDeploy console shows which ones are the result of an automatic deployment.

For more information, see [Redeploy and Roll Back a Deployment with AWS CodeDeploy](#) (p. 259).

AWS CodeDeploy Application Specification Files

An application specification file (AppSpec file), which is unique to AWS CodeDeploy, is a [YAML](#)-formatted or [JSON](#)-formatted file. The AppSpec file is used to manage each deployment as a series of lifecycle event hooks, which are defined in the file.

For information about how to create a well-formed AppSpec file, see [AWS CodeDeploy AppSpec File Reference](#) (p. 306).

Topics

- [AppSpec Files on an AWS Lambda Compute Platform](#) (p. 17)
- [AppSpec Files on an EC2/On-Premises Compute Platform](#) (p. 17)
- [How the AWS CodeDeploy Agent Uses the AppSpec File](#) (p. 18)

AppSpec Files on an AWS Lambda Compute Platform

If your application uses the AWS Lambda compute platform, the AppSpec file can be formatted with either YAML or JSON. It can also be typed directly into an editor in the console. The AppSpec file is used to specify:

- The AWS Lambda function version to deploy.
- The functions to be used as validation tests.

You can run validation Lambda functions after deployment lifecycle events. For more information, see [AppSpec 'hooks' Section for an AWS Lambda Deployment](#) (p. 317).

AppSpec Files on an EC2/On-Premises Compute Platform

If your application uses the EC2/On-Premises compute platform, the AppSpec file is always YAML-formatted. The AppSpec file is used 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.

You can run scripts on an instance after many of the individual deployment lifecycle events. AWS CodeDeploy runs only those scripts specified in the file, but those scripts can call other scripts on the instance. You can run any type of script as long as it is supported by the operating system running on the instances. For more information, see [AppSpec 'hooks' Section for an EC2/On-Premises Deployment \(p. 319\)](#).

How the AWS CodeDeploy Agent Uses the AppSpec File

During deployment, the AWS CodeDeploy agent looks up the name of the current event in the **hooks** section of the AppSpec file. If the event is not found, the AWS CodeDeploy agent moves on to the next step. If the event is found, the AWS CodeDeploy agent retrieves the list of scripts to execute. The scripts are run sequentially, in the order in which they appear in the file. The status of each script is logged in the AWS CodeDeploy agent log file on the instance.

If a script runs successfully, it returns an exit code of 0 (zero).

Note

The AWS CodeDeploy agent is not used in an AWS Lambda deployment.

During the **Install** event, the AWS CodeDeploy agent uses the mappings defined in the **files** section of the AppSpec file to determine which folders or files to copy from the revision to the instance.

If the AWS CodeDeploy agent installed on the operating system doesn't match what's listed in the AppSpec file, the deployment fails.

For information about AWS CodeDeploy agent log files, see [Working with the AWS CodeDeploy Agent \(p. 125\)](#).

Getting Started with AWS CodeDeploy

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

To begin, you must sign up for an AWS account. To sign up, go to <https://aws.amazon.com/> and choose **Create an AWS Account**.

Then you can continue with the rest of the setup tasks in this section.

Topics

- [Step 1: Provision an IAM User \(p. 19\)](#)
- [Step 2: Install or Upgrade and Then Configure the AWS CLI \(p. 20\)](#)
- [Step 3: Create a Service Role for AWS CodeDeploy \(p. 21\)](#)
- [Step 4: Create an IAM Instance Profile for Your Amazon EC2 Instances \(p. 25\)](#)
- [Step 5: Try the AWS CodeDeploy Sample Deployment Wizard \(p. 29\)](#)

Step 1: 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. For more information, see [Creating an IAM User](#) in *IAM User Guide*.
2. Grant the IAM user access to AWS CodeDeploy—and AWS services and actions AWS CodeDeploy depends on—by copying the following policy and attaching it to the IAM user:

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "autoscaling:*",
        "codedeploy:*",
        "ec2:*",
        "lambda:*",
        "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" : "*"
  }
}
```

The preceding policy grants the IAM user the access required to deploy to both an AWS Lambda compute platform and an EC2/On-Premises compute platform.

To learn how to attach a policy to an IAM user, see [Working with Policies](#). To learn how to restrict users to a limited set of AWS CodeDeploy actions and resources, see [Authentication and Access Control for AWS CodeDeploy \(p. 289\)](#).

You can use the AWS CloudFormation templates provided in this documentation to launch Amazon EC2 instances that are compatible with AWS CodeDeploy. To use AWS CloudFormation templates to create applications, deployment groups, or deployment configurations, 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": "*"
    }
  ]
}
```

For information about other AWS services listed in these statements, see:

- [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](#)

Step 2: Install or Upgrade and Then Configure the AWS CLI

To call AWS CodeDeploy commands from the AWS CLI on a local development machine, you must install the AWS CLI. AWS CodeDeploy commands first became available in version 1.6.1 of the AWS CLI. AWS CodeDeploy commands for working with on-premises instances became available in 1.7.19 of the AWS CLI.

If you have an older version of the AWS CLI installed, you must upgrade it so the AWS CodeDeploy commands will be available. You can call **aws --version** to check the version.

To install or upgrade the AWS CLI:

1. Follow the instructions in [Installing the AWS Command Line Interface](#) to install or upgrade the AWS CLI.
2. To configure the AWS CLI, see [Configuring the AWS Command Line Interface](#) and [Managing Access Keys for IAM Users](#).

Important

When you configure the AWS CLI, you will be prompted to specify an AWS region. Specify one of the supported regions listed in [Region and Endpoints](#) in the *AWS General Reference*.

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.

Step 3: Create a Service Role for AWS CodeDeploy

In AWS, service roles are used to grant permissions to an AWS service so it can access AWS resources. The policies that you attach to the service role determine which AWS resources the service can access and what it can do with those resources.

The service role you create for AWS CodeDeploy must be granted the permissions to access the instances to which you will deploy applications. These permissions enable AWS CodeDeploy to read the tags applied to the instances or the Auto Scaling group names associated with the instances.

The permissions you add to the service role specify the operations AWS CodeDeploy can perform when it accesses your Amazon EC2 instances and Auto Scaling groups. To add these permissions, attach an AWS-supplied policy, `AWSCodeDeployRole`, to the service role.

As part of setting up the service role, you also update its trust relationship to specify the endpoints to which you want to grant it access.

You can create a service role with the IAM console, the AWS CLI, or the IAM APIs.

Topics

- [Create a Service Role \(Console\)](#) (p. 21)
- [Create a Service Role \(CLI\)](#) (p. 23)
- [Get the Service Role ARN \(Console\)](#) (p. 24)
- [Get the Service Role ARN \(CLI\)](#) (p. 25)

Create a Service Role (Console)

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, choose **Roles**, and then choose **Create role**.
3. On the **Create role** page, choose **AWS service**, and from the **Choose the service that will use this role** list, choose **CodeDeploy**.
4. From **Select your use case**, choose **CodeDeploy**.
5. Choose **Next: Permissions**.
6. On the **Attached permissions policy** page, if there is a box next to `AWSCodeDeployRole`, select it, and then choose **Next: Review**.

The **AWSCodeDeployRole** policy provides the permissions required for your service role to:

- Read the tags on your instances or identify your Amazon EC2 instances by Auto Scaling group names.
 - Publish information to Amazon SNS topics.
 - Retrieve information about CloudWatch alarms.
 - Retrieve information about Elastic Load Balancing.
7. On the **Review** page, in **Role name**, type a name for the service role (for example **CodeDeployServiceRole**), and then choose **Create role**.

You can also type a description for this service role in the **Role description** box.

8. If you want this service role to have permission to access all currently supported endpoints, you are finished with this procedure.

If you want to restrict this service role from access to some endpoints, in the list of roles, browse to and choose the role you just created, and continue to the next step.

9. On the **Trust relationships** tab, choose **Edit trust relationship**.
10. You should see the following policy, which provides the service role permission to access all supported endpoints:

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

To grant the service role access to only some supported endpoints, replace the contents of the **Policy Document** box with the following policy, remove the lines for the endpoints you want to prevent access to, and then choose **Update Trust Policy**.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "codedeploy.us-east-2.amazonaws.com",
          "codedeploy.us-east-1.amazonaws.com",
          "codedeploy.us-west-1.amazonaws.com",
          "codedeploy.us-west-2.amazonaws.com",
          "codedeploy.eu-west-3.amazonaws.com",
          "codedeploy.ca-central-1.amazonaws.com",
          "codedeploy.eu-west-1.amazonaws.com",
          "codedeploy.eu-west-2.amazonaws.com",
          "codedeploy.eu-central-1.amazonaws.com",
          "codedeploy.ap-northeast-1.amazonaws.com",

```

```
        "codedeploy.ap-northeast-2.amazonaws.com",
        "codedeploy.ap-southeast-1.amazonaws.com",
        "codedeploy.ap-southeast-2.amazonaws.com",
        "codedeploy.ap-south-1.amazonaws.com",
        "codedeploy.sa-east-1.amazonaws.com"
    ],
    },
    "Action": "sts:AssumeRole"
}
]
```

For more information about creating service roles, see [Creating a Role to Delegate Permissions to an AWS Service](#) in the *IAM User Guide*.

Create a Service Role (CLI)

1. On your development machine, create a text file named, for example, `CodeDeployDemo-Trust.json`. This file is used to allow AWS CodeDeploy to work on your behalf.

Do one of the following:

- To grant access to all supported regions, save the following content in the file:

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

- To grant access to only some supported regions, type the following content into the file, and remove the lines for the regions to which you want to exclude access:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "codedeploy.us-east-2.amazonaws.com",
          "codedeploy.us-east-1.amazonaws.com",
          "codedeploy.us-west-1.amazonaws.com",
          "codedeploy.us-west-2.amazonaws.com",
          "codedeploy.eu-west-3.amazonaws.com",
          "codedeploy.ca-central-1.amazonaws.com",
          "codedeploy.eu-west-1.amazonaws.com",
          "codedeploy.eu-west-2.amazonaws.com",
          "codedeploy.eu-central-1.amazonaws.com",
          "codedeploy.ap-northeast-1.amazonaws.com",

```

```
        "codedeploy.ap-northeast-2.amazonaws.com",  
        "codedeploy.ap-southeast-1.amazonaws.com",  
        "codedeploy.ap-southeast-2.amazonaws.com",  
        "codedeploy.ap-south-1.amazonaws.com",  
        "codedeploy.sa-east-1.amazonaws.com"  
    ],  
    },  
    "Action": "sts:AssumeRole"  
  }  
]  
}
```

Note

Do not use a comma after the last endpoint in the list.

2. From the same directory, call the **create-role** command to create a service role named **CodeDeployServiceRole** based on the information in the text file you just created:

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

Important

Be sure to include `file://` before the file name. It is required in this command.

In the command's output, make a note of the value of the `Arn` entry under the `Role` object. You need it later when you create deployment groups. If you forget the value, follow the instructions in [Get the Service Role ARN \(CLI\)](#) (p. 25).

3. The managed policy you use depends on the compute platform.

- If your deployment is to an EC2/On-Premises compute platform:

Call the **attach-role-policy** command to give the service role named **CodeDeployServiceRole** the permissions based on the IAM managed policy named **AWSCodeDeployRole**. For example:

```
aws iam attach-role-policy --role-name CodeDeployServiceRole --policy-arn  
arn:aws:iam::aws:policy/service-role/AWSCodeDeployRole
```

- If your deployment is to an AWS Lambda compute platform:

Call the **attach-role-policy** command to give the service role named **CodeDeployServiceRole** the permissions based on the IAM managed policy named **AWSCodeDeployRoleForLambda**. For example:

```
aws iam attach-role-policy --role-name CodeDeployServiceRole --policy-arn  
arn:aws:iam::aws:policy/service-role/AWSCodeDeployRoleForLambda
```

For more information about creating service roles, see [Creating a Role for an AWS Service](#) in the *IAM User Guide*.

Get the Service Role ARN (Console)

To use the IAM console to get the ARN of the service role:

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, choose **Roles**.
3. In the **Filter** box, type **CodeDeployServiceRole**, and then press Enter.

4. Choose **CodeDeployServiceRole**.
5. Make a note of the value of the **Role ARN** field.

Get the Service Role ARN (CLI)

To use the AWS CLI to get the ARN of the service role, call the **get-role** command against the service role named **CodeDeployServiceRole**:

```
aws iam get-role --role-name CodeDeployServiceRole --query "Role.Arn" --output text
```

The value returned is the ARN of the service role.

Step 4: Create an IAM Instance Profile for Your Amazon EC2 Instances

Note

If you are using the AWS Lambda compute platform, skip this step. AWS Lambda deployments deploy a serverless Lambda function version, so an instance profile for Amazon EC2 instances is not required.

Your Amazon EC2 instances need permission to access the Amazon S3 buckets or GitHub repositories where the applications that will be deployed by AWS CodeDeploy are stored. To launch Amazon EC2 instances that are compatible with AWS CodeDeploy, you must create an additional IAM role, an *instance profile*. These instructions show you how to create an IAM instance profile to attach to your Amazon EC2 instances. This role gives AWS CodeDeploy permission to access the Amazon S3 buckets or GitHub repositories where your applications are stored.

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

Note

You can attach an IAM instance profile to an Amazon EC2 instance as you launch it or to a previously launched instance. For more information, see [Instance Profiles](#).

Topics

- [Create an IAM Instance Profile for Your Amazon EC2 Instances \(CLI\)](#) (p. 25)
- [Create an IAM Instance Profile for Your Amazon EC2 Instances \(Console\)](#) (p. 27)

Create an IAM Instance Profile for Your Amazon EC2 Instances (CLI)

In these steps, we assume you have already followed the instructions in the first three steps of [Getting Started with AWS CodeDeploy](#) (p. 19).

1. On your development machine, create a text file named `CodeDeployDemo-EC2-Trust.json`. Paste the following content, 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`. Paste the following content:

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

Note

We recommend that you restrict this policy to only those Amazon S3 buckets your Amazon EC2 instances must access. Make sure to give access to the Amazon S3 buckets that contain the AWS CodeDeploy agent. Otherwise, an error may occur when the AWS CodeDeploy agent is installed or updated on the instances. To grant the IAM instance profile access to only some AWS CodeDeploy resource kit buckets in Amazon S3, use the following policy but remove the lines for buckets you want to prevent access to:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::replace-with-your-s3-bucket-name/*",
        "arn:aws:s3:::aws-codedeploy-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
      ]
    }
  ]
}
```

```
]
}
```

3. From the same directory, call the **create-role** command to create an IAM role named **CodeDeployDemo-EC2-Instance-Profile**, based on the information in the first file:

Important

Be sure to include `file://` before the file name. It is required in this command.

```
aws iam create-role --role-name CodeDeployDemo-EC2-Instance-Profile --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-Instance-Profile** the permissions based on the information in the second file:

Important

Be sure to include `file://` before the file name. It is required in this command.

```
aws iam put-role-policy --role-name CodeDeployDemo-EC2-Instance-Profile --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**. The instance profile allows Amazon EC2 to pass the IAM role named **CodeDeployDemo-EC2-Instance-Profile** to an Amazon EC2 instance when the instance is first launched:

```
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-Instance-Profile
```

If you need to get the name of the IAM instance profile, see [list-instance-profiles-for-role](#) in the IAM section of the *AWS CLI Reference*.

You've now created an IAM instance profile to attach to your Amazon EC2 instances. For more information, see [IAM Roles for Amazon EC2](#) in the *Amazon EC2 User Guide*.

Create an IAM Instance Profile for Your Amazon EC2 Instances (Console)

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, choose **Policies**, and then choose **Create policy**. (If a **Get Started** button appears, choose it, and then choose **Create Policy**.)
3. On the **Create policy** page, paste the following in the **JSON** tab:

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

```
        "Resource": "*"
      }
    ]
  }
}
```

Note

We recommend that you restrict this policy to only those Amazon S3 buckets your Amazon EC2 instances must access. Make sure to give access to the Amazon S3 buckets that contain the AWS CodeDeploy agent. Otherwise, an error may occur when the AWS CodeDeploy agent is installed or updated on the instances. To grant the IAM instance profile access to only some AWS CodeDeploy resource kit buckets in Amazon S3, use the following policy but remove the lines for buckets you want to prevent access to:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::replace-with-your-s3-bucket-name/*",
        "arn:aws:s3:::aws-codedeploy-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
      ]
    }
  ]
}
```

4. Choose **Review policy**.
5. On the **Create policy** page, type **CodeDeployDemo-EC2-Permissions** in the **Policy Name** box.
6. (Optional) For **Description**, type a description for the policy.
7. Choose **Create Policy**.
8. In the navigation pane, choose **Roles**, and then choose **Create role**.
9. On the **Create role** page, choose **AWS service**, and from the **Choose the service that will use this role** list, choose **EC2**.
10. From the **Select your use case** list, choose **EC2**.
11. Choose **Next: Permissions**.
12. On the **Attached permissions policy** page, if there is a box next to **CodeDeployDemo-EC2-Permissions**, select it, and then choose **Next: Review**.
13. On the **Review** page, in **Role name**, type a name for the service role (for example **CodeDeployDemo-EC2-Instance-Profile**), and then choose **Create role**.

You can also type a description for this service role in the **Role description** box.

You've now created an IAM instance profile to attach to your Amazon EC2 instances. For more information, see [IAM Roles for Amazon EC2](#) in the *Amazon EC2 User Guide*.

Step 5: Try the AWS CodeDeploy Sample Deployment Wizard

Note

The AWS CodeDeploy Sample Deployment Wizard currently does not apply to deployments that use the AWS Lambda compute platform.

After you have completed the first four steps in [Getting Started with AWS CodeDeploy \(p. 19\)](#), try the Sample deployment wizard. It guides you through the steps for creating an AWS CodeDeploy deployment. The Sample deployment wizard lets you try an in-place deployment and a blue/green deployment.

- **In-place deployment:** The application on each instance in the deployment group is stopped, the latest application revision is installed, and the new version of the application is started and validated. You can use a load balancer so that each instance is deregistered during its deployment and then restored to service after the deployment is complete. Only deployments that use the EC2/On-Premises compute platform can use in-place deployments. For more information about in-place deployments, see [Overview of an In-Place Deployment \(p. 4\)](#).
- **Blue/green deployment:** The behavior of your deployment depends on which compute platform you use:
 - **Blue/green on an EC2/On-Premises compute platform:** The instances in a deployment group (the original environment) are replaced by a different set of instances (the replacement environment) using these steps:
 - Instances are provisioned for the replacement environment.
 - The latest application revision is installed on the replacement instances.
 - An optional wait time occurs for activities such as application testing and system verification.
 - Instances in the replacement environment are registered with an Elastic Load Balancing load balancer, causing traffic to be rerouted to them. Instances in the original environment are deregistered and can be terminated or kept running for other uses.

Note

When using an EC2/On-Premises compute platform, blue/green deployments work with Amazon EC2 instances only.

- **Blue/green on an AWS Lambda compute platform:** Traffic is shifted from your current serverless environment to one with your updated Lambda function versions. You can specify Lambda functions that perform validation tests and choose the way in which the traffic shift occurs. All AWS Lambda compute platform deployments are blue/green deployments. For this reason, you do not need to specify a deployment type.

For more information about blue/green deployments, see [Overview of a Blue/Green Deployment \(p. 5\)](#).

For both deployment type samples, we assume you have no prior experience with AWS CodeDeploy and have not yet created any resources, such as applications, application revisions, or deployment groups in AWS CodeDeploy.

These topics refer to resources and concepts that are unique to AWS CodeDeploy. To familiarize yourself with them before you start, see [Primary Components \(p. 7\)](#).

Prerequisites

If you want AWS CodeDeploy to create some sample Amazon EC2 instances, you must have an Amazon EC2 instance key pair. To create an Amazon EC2 instance key pair, follow the instructions in [Creating Your Key Pair Using Amazon EC2](#). Be sure your Amazon EC2 instance key pair is created in one of the regions listed in [Region and Endpoints](#) in the *AWS General Reference*. You must create an Amazon EC2 instance key pair before you start the wizard. Otherwise, it will not appear in the **Key pair name** drop-down list in the Sample deployment wizard.

If you use the AWS CloudFormation template to launch Amazon EC2 instances, the calling IAM user must have access to AWS CloudFormation and AWS services and actions on which AWS CloudFormation depends. If you have not followed the steps in [Step 1: Provision an IAM User \(p. 19\)](#) to provision the calling IAM user, you must at least attach the following policy:

```
{
  "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": "*"
    }
  ]
}
```

The following portion of the policy grants the calling IAM user access to the IAM actions required to create the service role.

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

The following portion of the policy grants the calling IAM user permission to create applications and deployment groups and deploy applications.

```
{
```

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

Not what you're looking for?

- To create a deployment that uses an existing application, revision, deployment group, or custom deployment configuration in AWS CodeDeploy, follow the instructions in [Create a Deployment with AWS CodeDeploy](#) (p. 245).
- To practice deploying to on-premises instances instead of Amazon EC2 instances, see [Tutorial: Deploy an Application to an On-Premises Instance with AWS CodeDeploy \(Windows Server, Ubuntu Server, or Red Hat Enterprise Linux\)](#) (p. 90).

Topics

- [Try a Sample Blue/Green Deployment in AWS CodeDeploy](#) (p. 31)
- [Try a Sample In-Place Deployment in AWS CodeDeploy](#) (p. 35)

Try a Sample Blue/Green Deployment in AWS CodeDeploy

This section guides you through the steps required to deploy a revision to one or more Amazon EC2 instances using the Sample deployment wizard, and then run a blue/green deployment to replace the original set of instances in a deployment group, the original environment, with a different set of instances, the replacement environment.

Topics

- [Start the wizard](#) (p. 31)
- [Step 1: Get started](#) (p. 32)
- [Step 2: Choose a deployment type](#) (p. 32)
- [Step 3: Create blue/green deployment](#) (p. 32)
- [Step 4: Monitor the blue/green deployment](#) (p. 33)
- [Step 5: Refresh the web application window](#) (p. 33)
- [Step 6: Clean up sample resources](#) (p. 34)

Start the wizard

To start the wizard:

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. If an introductory page appears, choose **Get Started Now**. If the **Applications** page appears, in **More info**, choose **Sample deployment wizard**.

Step 1: Get started

Choose **Sample deployment**, and then choose **Next**.

Step 2: Choose a deployment type

Choose **Blue/green deployment**, and then choose **Next**.

Step 3: Create blue/green deployment

On the **Step 3: Create blue/green deployment** page, we provide default values for most of the components you will need for a blue/green deployment.

1. Accept or change any of the default names in the following fields:

Note

Because the fields have different validation requirements, to save time, we recommend leaving the default names for your sample deployment.

Field	Description
Application name	A name that functions as a container to ensure the right content is deployed using the correct combination of deployment options.
Deployment group name	A name that represents the group of instances to deploy to.
Auto Scaling group name	A name for the Auto Scaling group that will provision the Amazon EC2 instances for the deployment environment.
Load balancer name	The Classic Load Balancer in Elastic Load Balancing that will route traffic to the instances in your deployment environment.
Service role name	The name of the service role that will provide AWS CodeDeploy with the permissions required to access other AWS services during the deployment process.

2. From the **Key pair name** drop-down list, choose the Amazon EC2 instance key pair you will use to connect to the Amazon EC2 instances.

Note

To create an Amazon EC2 instance key pair, follow the instructions in [Creating Your Key Pair Using Amazon EC2](#). Be sure your key pair is created in one of the regions listed in [Region and Endpoints](#) in the *AWS General Reference*. The new Amazon EC2 instance key pair may not appear in the **Key pair name** drop-down list until you restart the wizard.

3. Choose **Launch environment**.
4. After the environment is ready, review the details under **Congratulations! Your environment is ready**.

Note

For easy reference later, we recommend that you copy the entire text in the **Congratulations! Your environment is ready** area to a file on your computer.

- The name of the Classic Load Balancer created for you, such as `BlueGreenLoadBalancer-abcdefh`.
- The name of the Auto Scaling group created for your original environment, such as `CodeDeployBGStack-abcdefh-BlueGreenAutoScalingGroup-1IJKLMN234056`.
- The web address of the application that has been deployed in your original environment, such as `http://BlueGreenLoadBalancer-abcdefh-1234567890.us-east-2.elb.amazonaws.com`.

Important

We recommend that you open your web application in a different browser window now so you can refresh it later to review the background color change made during the blue/green deployment.

- The name of the AWS CloudFormation stack used to create your environment, such as `BlueGreenLoadBalancer-abcdefh`. When you are ready to clean up resources from your sample blue/green deployment, you will use the AWS CloudFormation console to delete this stack.
- The location of the sample application that will be installed in the original environment, such as:

https://s3.amazonaws.com/aws-codedeploy-us-east-2/samples/latest/SampleApp_Linux.zip

5. Choose **Start blue/green deployment**.

Step 4: Monitor the blue/green deployment

On the **Deployment** page, you can view the progress of the blue/green deployment in a dashboard format.

The **Deployment progress** area reports the progress of the four major steps in the deployment:

- Provisioning instances in the replacement environment.
- Installing the new application revision in the replacement environment.
- Rerouting traffic to the replacement environment.
- Terminating the instances in the original environment.

The **Instances receiving traffic** area reports the counts of instances in both the original and replacement environments that are currently registered with the load balancer.

The **Deployment details** area lists identifying information about both the deployment and the application revision that was installed in the replacement environment.

The **Instance activity** area provides details about each instance from both the original environment and replacement environment.

Step 5: Refresh the web application window

In the browser window where you previously opened a view of the application that was installed in the original environment, such as `http://BlueGreenLoadBalancer-abcdefh-1234567890.us-east-2.elb.amazonaws.com`, choose the browser's **Refresh** button.

If the background color of the web page changes from blue to green, traffic has been successfully routed from the instances we created for you in [Step 3: Create blue/green deployment \(p. 32\)](#) to the replacement instances created during the process in [Step 4: Monitor the blue/green deployment \(p. 33\)](#)

Step 6: Clean up sample resources

To avoid future charges, you must clean up the resources used in this wizard. The resources must be cleaned up in the following order:

- The Auto Scaling group that the instances for the replacement environment belong to. (The Auto Scaling group associated with the instances in the original environment will be deleted when you delete the AWS CloudFormation stack.)
- The AWS CloudFormation stack that the Sample deployment wizard created to provide the original environment for the blue/green deployment.
- The AWS CodeDeploy deployment group and application created by the Sample deployment wizard.

To delete the Auto Scaling group for the replacement environment

You will see two Auto Scaling groups associated with the sample blue/green deployment in the Amazon EC2 console. To avoid errors, be sure to delete the Auto Scaling group associated with the replacement environment in this step. You can distinguish the Auto Scaling groups by their formats:

- Delete the Auto Scaling with this format:

```
CodeDeploy_BlueGreenDemoFleet-9zyxwvut_d-ZY9XWVUTS8R
```

- You can delete the Auto Scaling group with this format now or let it be removed when the AWS CloudFormation stack is deleted:

```
CodeDeployBGStack-abcdefgh-BlueGreenAutoScalingGroup-1IJKLMN234056
```

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
3. On the **Auto Scaling** groups page, select the box by the Auto Scaling group created for the replacement environment. For example:

```
CodeDeploy_BlueGreenDemoFleet-9zyxwvut_d-ZY9XWVUTS8R.
```

From the **Actions** menu, choose **Delete**.

4. When prompted for confirmation, choose **Yes, Delete**.

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. Select the box next to the stack created for your blue/green deployment. For example:

```
CodeDeployBGStack-abcdefgh-BlueGreenAutoScalingGroup-1IJKLMN234056.
```

From the **Actions** menu, choose **Delete Stack**.

3. When prompted, choose **Yes, Delete**. The remaining resources that were created for this deployment in Amazon EC2, AWS Identity and Access Management, Amazon VPC, and Elastic Load Balancing will be deleted.

To delete AWS CodeDeploy blue/green deployment resource records

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. If the **Applications** page does not appear, on the AWS CodeDeploy menu, choose **Applications**.
3. On the **Applications** page, choose the application to delete, such as `BlueGreenDemoApplication`.
4. On the **Application details** page, in **Deployment groups**, choose the button next to the deployment group you want to delete, such as `BlueGreenDemoFleet-1abcdef`. On the **Actions** menu, choose **Delete**. When prompted, type the name of the deployment group to confirm you want to delete it, and then choose **Delete**.
5. At the bottom of the **Application details** page, choose **Delete application**.
6. When prompted, type the name of the application, and then choose **Delete**.

All records about the application and its associated deployment groups, revisions, and deployments will be deleted.

Try a Sample In-Place Deployment in AWS CodeDeploy

This section guides you through the steps required to deploy a revision to one or more Amazon EC2 instances using the Sample deployment wizard.

Topics

- [Video of a Sample AWS CodeDeploy In-Place Deployment](#) (p. 35)
- [Start the wizard](#) (p. 35)
- [Step 1: Get started](#) (p. 36)
- [Step 2: Choose a deployment type](#) (p. 36)
- [Step 3: Configure instances](#) (p. 36)
- [Step 4: Name your application](#) (p. 37)
- [Step 5: Select a revision](#) (p. 37)
- [Step 6: Create a deployment group](#) (p. 37)
- [Step 7: Select a service role](#) (p. 37)
- [Step 8: Choose a deployment configuration](#) (p. 38)
- [Step 9: Review deployment details](#) (p. 38)
- [Clean up sample in-place deployment resources](#) (p. 38)

Video of a Sample AWS CodeDeploy In-Place Deployment

This short video (5:01) walks you through a sample AWS CodeDeploy in-place deployment using the AWS CodeDeploy console.

[Video of an AWS CodeDeploy In-Place Deployment.](#)

Start the wizard

To start the wizard:

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. If an introductory page appears, choose **Get Started Now**. If the **Applications** page appears, in **More info**, choose **Sample deployment wizard**.

Step 1: Get started

Choose **Sample deployment**, and then choose **Next**.

Step 2: Choose a deployment type

Choose **In-place deployment**, and then choose **Next**.

Step 3: Configure instances

If you have Amazon EC2 instances that are already configured for use in AWS CodeDeploy deployments, choose **Skip**, read and follow the instructions, and then proceed to [Step 4: Name your application](#) (p. 37).

If you want AWS CodeDeploy to launch a new set of Amazon EC2 instances:

1. Next to **Operating system**, choose **Amazon Linux** or **Windows Server**.

Important

You may be billed for the Amazon EC2 instances launched by AWS CodeDeploy, so be sure to terminate them after you've completed the wizard. In this wizard, an AWS CloudFormation template is used to launch these Amazon EC2 instances. To delete the AWS CloudFormation stack created to launch the Amazon EC2 instances, see [Deleting a Stack on the AWS CloudFormation Console](#). The stack name will start with **CodeDeploySampleStack**.

2. From the **Key pair name** drop-down list, choose the Amazon EC2 instance key pair you will use to connect to the Amazon EC2 instances.

Note

To create an Amazon EC2 instance key pair, follow the instructions in [Creating Your Key Pair Using Amazon EC2](#). Be sure your key pair is created in one of the regions listed in [Region and Endpoints](#) in the *AWS General Reference*. The new Amazon EC2 instance key pair might not appear in the **Key pair name** drop-down list until you restart the wizard.

3. Leave the defaults for **Tag key and value**. AWS CodeDeploy will use this tag key and value to locate the instances during deployments.

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

4. Choose **Launch instances**.

If you choose **See more details in AWS CloudFormation**, the AWS CloudFormation console will open in a separate web browser tab. Look for a stack that starts with **CodeDeploySampleStack**. When **CREATE_COMPLETE** appears in the **Status** column, your Amazon EC2 instances have been launched. (This might take several minutes.)

5. To continue, choose **Next**.

Step 4: Name your application

In the **Application name** box, leave the proposed application name or, if you prefer, type a different name, and choose **Next**.

Step 5: Select a revision

Review the information about our sample application revision, and choose **Next**.

Note

If you want to examine the content of our sample revision, choose **Download sample bundle**, and follow your web browser's instructions to download and view the content.

If you chose **Skip** in [Step 3: Configure instances \(p. 36\)](#), from the **Revision type** drop-down list, choose the type of application revision that corresponds to the Amazon EC2 instances type (Amazon Linux or Windows Server).

Step 6: Create a deployment group

1. In the **Deployment group name** box, leave the proposed deployment group name or, if you prefer, type a different name.
2. The key and value of the key-value pair you specified in the **Configure instances** page (for example, Name and CodeDeployDemo) should appear.

If you chose **Skip** in [Step 3: Configure instances \(p. 36\)](#), in **Add instances**, overwrite the values of the **Key** and **Value** boxes with the key and value of the key-value pair for your Amazon EC2 instances.

Optionally, if your Amazon EC2 instances have multiple key-value pairs, you can type them into the blank row. A new blank row appears so you can add another key-value pair. You can add up to 10 key-value pairs. Choose the remove icon to remove a key-value pair from the list.

Note

AWS CodeDeploy displays the number of instances that match each key-value pair. To view instances in the Amazon EC2 console, click the number.

If you are using our AWS CloudFormation template to launch new Amazon EC2 instances, and the number is larger than you're expecting, choose **Cancel**, start the wizard from the beginning, and in [Step 3: Configure instances \(p. 36\)](#), specify a tag value different from the default . (Be sure to delete the AWS CloudFormation stack to terminate the Amazon EC2 instances.)

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 tag key and value different from the default in **Add instances**.

3. If you have an Auto Scaling group to add to the deployment group, choose **Search by Auto Scaling group names**, and then type the Auto Scaling group name. You can add up to 10 Auto Scaling groups. Choose the remove icon to remove an Auto Scaling group from the list.

Note

AWS CodeDeploy displays the number of Amazon EC2 instances that match each Auto Scaling group name. To view instances in the Amazon EC2 console, click the number.

4. Choose **Next**.

Step 7: Select a service role

Choose **Create a service role** or **Use an existing service role**.

If you are using this wizard for the first time, we recommend you choose **Create a service role**, choose **Next** to accept the default name, and then proceed to [Step 8: Choose a deployment configuration](#) (p. 38).

If you already have a service role, choose **Use an existing service role**, choose it from the **Role name** drop-down list, and then choose **Next**.

Step 8: Choose a deployment configuration

1. To use a built-in configuration for this deployment, choose **Use a default deployment configuration**. To create your own configuration for this deployment, choose **Create a custom deployment configuration**.
2. If you chose **Use a default deployment configuration** and want to use a configuration different from the one selected, next to the desired configuration, choose **Select**. Choose **Next**, and go to [Step 9: Review deployment details](#) (p. 38).
3. If you chose **Create a custom deployment configuration**:
 - a. In the **Deployment configuration name** box, type a unique name for the configuration.
 - b. Use the **Number** or **Percentage** box to type either the number or percentage of total Amazon EC2 instances that should be available during the deployment.
 - c. Choose **Next**.

Step 9: Review deployment details

1. If you need to make changes, choose one of the **Edit** links. After you've made your changes, choose **Next** until you return to the **Review deployment details** page, and then choose **Deploy**.
2. Choose the **Refresh** button next to the table to get deployment status. To get information about the deployment, see [View Instance Details \(Console\)](#) (p. 198).
3. Our sample revision deploys a single web page to each instance. You can use your web browser to verify the deployment was successful by going to `http://PublicDNS` for each instance (for example, `http://ec2-01-234-567-890.compute-1.amazonaws.com`). The web page will display a message of congratulations.

To get the public DNS value, in the Amazon EC2 console, choose the Amazon EC2 instance. On the **Description** tab, look for the value in **Public DNS**.

Clean up sample in-place deployment resources

To avoid future charges, you must clean up the resources used in this wizard. If you used our AWS CloudFormation template to launch Amazon EC2 instances, delete the AWS CloudFormation stack. This will terminate the instances and their associated resources.

If you launched your own Amazon EC2 instances just for this wizard, you should terminate them. Optionally, you can delete the deployment component records associated with this wizard 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. Choose the button next to the stack starting with `CodeDeploySampleStack`. On the **Actions** menu, choose **Delete Stack**.
3. When prompted, choose **Yes, Delete**. The Amazon EC2 instances will be terminated. The associated IAM instance profile and service role will be deleted.

To terminate 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**, choose **Instances**.
3. Select the box for each Amazon EC2 instance to terminate.
4. On the **Actions** menu, point to **Instance State**, and then choose **Terminate**.
5. When prompted, choose **Yes, Terminate**.

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

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. If the **Applications** page does not appear, on the AWS CodeDeploy menu, choose **Applications**.
3. On the **Applications** page, choose the application to delete.
4. On the **Application details** page, in **Deployment groups**, choose the button next to the deployment group you want to delete. On the **Actions** menu, choose **Delete**. When prompted, type the name of the deployment group to confirm you want to delete it, and then choose **Delete**.
5. At the bottom of the **Application details** page, choose **Delete application**.
6. When prompted, type the name of the application, and then choose **Delete**.

All records about the application and its associated deployment groups, revisions, and deployments will be deleted.

Product and Service Integrations with AWS CodeDeploy

By default, AWS CodeDeploy is integrated with a number of AWS services and partner products and services. The following information can help you configure AWS CodeDeploy to integrate with the products and services you use.

- [Integration with Other AWS Services \(p. 40\)](#)
- [Integration with Partner Products and Services \(p. 49\)](#)
- [Integration Examples from the Community \(p. 55\)](#)

Integration with Other AWS Services

AWS CodeDeploy is integrated with the following AWS services:

Amazon CloudWatch	<p>Amazon CloudWatch is a monitoring service for AWS cloud resources and the applications you run on AWS. You can use Amazon CloudWatch to collect and track metrics, collect and monitor log files, and set alarms. AWS CodeDeploy supports the following CloudWatch tools:</p> <ul style="list-style-type: none">• CloudWatch Alarms for monitoring your deployments and stopping them when your specified monitoring metrics exceed or fall below the thresholds you specify in a CloudWatch alarm rule. To use alarm monitoring, you first set up an alarm in CloudWatch, and then add it in AWS CodeDeploy to the application or deployment group where deployments should stop when the alarm is activated. <p>Learn more:</p> <ul style="list-style-type: none">• Creating CloudWatch Logs Alarms• Amazon CloudWatch Events for detecting and reacting to changes in the state of an instance or a deployment in your AWS CodeDeploy operations. Then, based on rules you create, CloudWatch Events invokes one or more target actions when a deployment or instance enters the state you specify in a rule. <p>Learn more:</p> <ul style="list-style-type: none">• Monitoring Deployments with Amazon CloudWatch Events (p. 274)• Amazon CloudWatch Logs for monitoring the three types of logs created by the AWS
--------------------------	---

	<p>CodeDeploy agent without having to sign in to instances one at a time.</p> <p>Learn more:</p> <ul style="list-style-type: none">• View AWS CodeDeploy Logs in CloudWatch Logs Console
Auto Scaling	<p>AWS CodeDeploy supports Amazon EC2 Auto Scaling, an AWS web service that can automatically launch Amazon EC2 instances based on criteria you specify (for example, limits 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. Auto Scaling terminates those Amazon EC2 instances when they are no longer needed.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Integrating AWS CodeDeploy with Auto Scaling (p. 44)• Tutorial: Use AWS CodeDeploy to Deploy an Application to an Auto Scaling Group (p. 96)• Under the Hood: AWS CodeDeploy and Auto Scaling Integration
AWS CloudTrail	<p>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 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 which 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.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Monitoring Deployments with AWS CloudTrail (p. 277)

AWS Cloud9

[AWS Cloud9](#) is an online, cloud-based integrated development environment (IDE) you can use to write, run, debug, and deploy code using just a browser from an internet-connected machine. AWS Cloud9 includes a code editor, debugger, terminal, and essential tools, such as the AWS CLI and Git.

- You can use the AWS Cloud9 IDE to run, debug, and build code that is in a GitHub repository. You can view, change, and save the code using its IDE **Environment** window and editor tabs. When you're ready, you can use Git in the AWS Cloud9 terminal session to push code changes to your GitHub repository and then use AWS CodeDeploy to deploy your updates. For more information about using AWS Cloud9 with GitHub, see [GitHub Sample for AWS Cloud9](#).
- You can use the AWS Cloud9 IDE to update an AWS Lambda function. You can then use AWS CodeDeploy create a deployment that shifts traffic to the new version of your AWS Lambda function. For more information, see [Working with AWS Lambda Functions in the AWS Cloud9 Integrated Development Environment \(IDE\)](#).

For more information about AWS Cloud9, see [What is AWS Cloud9](#) and [Getting Started with AWS Cloud9](#).

AWS CodePipeline	<p>AWS CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software in a continuous delivery process. You can use AWS CodePipeline to define your own release process so that the service builds, tests, and deploys your code every time there is a code change. For example, you might have three deployment groups for an application: Beta, Gamma, and Prod. You can set up a pipeline so that each time there is a change in your source code, the updates are deployed to each deployment group, one by one.</p> <p>You can configure AWS CodePipeline to use AWS CodeDeploy to deploy:</p> <ul style="list-style-type: none">• Code to Amazon EC2 instances, on-premises instances, or both.• Serverless AWS Lambda function versions. <p>You can create the AWS CodeDeploy application, deployment, and deployment group to use in a deploy action in a stage either before you create the pipeline or in the Create Pipeline wizard.</p> <p>Learn more:</p> <ul style="list-style-type: none">• AWS for DevOps Getting Started Guide — Learn how to use AWS CodePipeline with AWS CodeDeploy to continuously deliver and deploy source code in AWS CodeCommit repositories to Amazon EC2 instances.• Simple Pipeline Walkthrough (Amazon S3 Bucket)• Simple Pipeline Walkthrough (AWS CodeCommit Repository)• Four-Stage Pipeline Tutorial
AWS Serverless Application Model	<p>AWS Serverless Application Model (AWS SAM) is a model to define serverless applications. It extends AWS CloudFormation to provide a simplified way of defining AWS Lambda functions, Amazon API Gateway APIs, and Amazon DynamoDB tables required by a serverless application. If you already use AWS SAM, you can add deployment preferences to start using AWS CodeDeploy to manage the way in which traffic is shifted during an AWS Lambda application deployment.</p> <p>For more information, see the AWS Serverless Application Model.</p>

Elastic Load Balancing

AWS CodeDeploy supports [Elastic Load Balancing](#), a service that distributes incoming application traffic across multiple Amazon EC2 instances.

For AWS CodeDeploy deployments, load balancers also prevent traffic from being routed to instances when they are not ready, are currently being deployed to, or are no longer needed as part of an environment.

Learn more:

- [Integrating AWS CodeDeploy with Elastic Load Balancing \(p. 46\)](#)

Topics

- [Integrating AWS CodeDeploy with Auto Scaling \(p. 44\)](#)
- [Integrating AWS CodeDeploy with Elastic Load Balancing \(p. 46\)](#)

Integrating AWS CodeDeploy with Auto Scaling

AWS CodeDeploy supports Auto Scaling, an AWS service that can launch Amazon EC2 instances automatically according to conditions you define. These conditions can include limits exceeded in a specified time interval for CPU utilization, disk reads or writes, or inbound or outbound network traffic. Auto Scaling terminates the instances when they are no longer needed. For more information, see [What Is Auto Scaling?](#).

When new Amazon EC2 instances are launched as part of an Auto Scaling group, AWS CodeDeploy can deploy your revisions to the new instances automatically. You can also coordinate deployments in AWS CodeDeploy with Amazon EC2 instances registered with Elastic Load Balancing load balancers. For more information, see [Integrating AWS CodeDeploy with Elastic Load Balancing \(p. 46\)](#) and [Set Up a Load Balancer in Elastic Load Balancing for AWS CodeDeploy Deployments \(p. 224\)](#).

Note

Be aware that you might encounter issues if you associate multiple deployment groups with a single Auto Scaling group. If one deployment fails, for example, the instance will begin to shut down, but the other deployments that were running can take an hour to time out. For more information, see [Avoid associating multiple deployment groups with a single Auto Scaling group \(p. 359\)](#) and [Under the Hood: AWS CodeDeploy and Auto Scaling Integration](#).

Topics

- [Deploying AWS CodeDeploy Applications to Auto Scaling Groups \(p. 44\)](#)
- [Auto Scaling Behaviors with AWS CodeDeploy \(p. 45\)](#)
- [Using a Custom AMI with AWS CodeDeploy and Auto Scaling \(p. 46\)](#)

Deploying AWS CodeDeploy Applications to Auto Scaling Groups

To deploy an AWS CodeDeploy application revision to an Amazon EC2 Auto Scaling group:

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. Although Amazon EC2 instances still require an IAM instance profile, the profile doesn't need any additional permissions to deploy from a GitHub repository. For more information, see [Step 4: Create an IAM Instance Profile for Your Amazon EC2 Instances](#) (p. 25).

2. Create or use an Auto Scaling group, 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 service role.
5. Use AWS CodeDeploy to deploy your revision to the deployment group that contains the Auto Scaling group.

For more information, see [Tutorial: Use AWS CodeDeploy to Deploy an Application to an Auto Scaling Group](#) (p. 96).

Auto Scaling Behaviors with AWS CodeDeploy

The execution order of custom lifecycle hook events cannot be predetermined

You can add your own lifecycle hooks to Auto Scaling groups to which AWS CodeDeploy deploys. However, the order in which those custom lifecycle hook events are executed 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 it will be executed before the first, or after the last, AWS CodeDeploy default deployment lifecycle event.

To learn how to add custom lifecycle hooks to an Auto Scaling group, see [Adding Lifecycle Hooks](#).

Scale-up events during a deployment results in a mixed environment

If an Auto Scaling scale-up event occurs while a deployment is underway, the new instances will be updated with the application revision that was most recently deployed, not the application revision that is currently being deployed. If the deployment succeeds, the old instances and the newly scaled-up instances will be hosting different application revisions.

To resolve this problem after it occurs, you can redeploy the newer application revision to the affected deployment groups.

To avoid this problem, we recommend suspending the Auto Scaling scale-up processes while deployments are taking place. You can do this through a setting in the `common_functions.sh` script that is used for load balancing with AWS CodeDeploy. If `HANDLE_PROCS=true`, the following Auto Scaling events are suspended automatically during the deployment process:

- `AZRebalance`
- `AlarmNotification`
- `ScheduledActions`
- `ReplaceUnhealthy`

Important

Only the `CodeDeployDefault.OneAtATime` deployment configuration supports this functionality. If you are using other deployment configurations, the deployment group might still have different application revisions applied to its instances.

For more information about using `HANDLE_PROCS=true` to avoid deployment problems when using Auto Scaling, see [Important notice about handling AutoScaling processes](#) in [aws-codedeploy-samples](#) on GitHub.

The order of events must be controlled when using AWS CloudFormation `cfn-init` scripts

If you use `cfn-init` (or `cloud-init`) to run scripts on newly provisioned Linux-based instances, your deployments might fail unless you strictly control the order of events that occur after the instance starts.

That order must be:

1. The newly provisioned instance starts.
2. All `cfn-init` bootstrapping scripts run to completion.
3. The AWS CodeDeploy agent starts.
4. The latest application revision is deployed to the instance.

If the order of events is not carefully controlled, the AWS CodeDeploy agent might start a deployment before all the scripts have finished running.

To control the order of events, use one of these best practices:

- Install the AWS CodeDeploy agent through a `cfn-init` script, placing it after all other scripts.
- Include the AWS CodeDeploy agent in a custom AMI and use a `cfn-init` script to start it, placing it after all other scripts.

For information about using `cfn-init`, see [cfn-init](#) in *AWS CloudFormation User Guide*.

Using a Custom AMI with AWS CodeDeploy and Auto Scaling

You have two options for specifying the base AMI to use when new Amazon EC2 instances are launched in an Auto Scaling group:

- You can specify a base custom AMI that already has the AWS CodeDeploy agent installed. Because the agent is already installed, this option launches new Amazon EC2 instances more quickly than the other option. However, this option provides a greater likelihood that initial deployments of Amazon EC2 instances will fail, especially if the AWS CodeDeploy agent is out of date. If you choose this option, we recommend you regularly update the AWS CodeDeploy agent in your base custom AMI.
- You can specify a base AMI that doesn't have the AWS CodeDeploy agent installed and have the agent installed as each new instance is launched in an Auto Scaling group. Although this option launches new Amazon EC2 instances more slowly than the other option, it provides a greater likelihood that initial deployments of instances will succeed. This option uses the most recent version of the AWS CodeDeploy agent.

Integrating AWS CodeDeploy with Elastic Load Balancing

Elastic Load Balancing provides three types of load balancers that can be used in AWS CodeDeploy deployments: Classic Load Balancers, Application Load Balancers, and Network Load Balancers.

Classic Load Balancer

Routes and load balances either at the transport layer (TCP/SSL) or the application layer (HTTP/HTTPS). It supports either EC2-Classic or a VPC.

Application Load Balancer

Routes and load balances at the application layer (HTTP/HTTPS) and supports path-based routing. It can route requests to ports on each EC2 instance or container instance in your virtual private cloud (VPC).

Network Load Balancer

Routes and load balances at the transport layer (TCP/UDP Layer-4) based on address information extracted from the TCP packet header, not from packet content. Network Load Balancers can handle traffic bursts, retain the source IP of the client, and use a fixed IP for the life of the load balancer.

To learn more about Elastic Load Balancing load balancers, see the following topics:

- [What Is Elastic Load Balancing?](#)
- [What Is a Classic Load Balancer?](#)
- [What Is an Application Load Balancer?](#)
- [What Is a Network Load Balancer?](#)

The role of a load balancer in an AWS CodeDeploy deployment

During AWS CodeDeploy deployments, a load balancer prevents internet traffic from being routed to instances when they are not ready, are currently being deployed to, or are no longer needed as part of an environment. The exact role the load balancer plays, however, depends on whether it is used in a blue/green deployment or an in-place deployment.

Note

The use of Elastic Load Balancing load balancers is mandatory in blue/green deployments and optional in in-place deployments.

Blue/green deployments

Rerouting instance traffic behind an Elastic Load Balancing load balancer is fundamental to AWS CodeDeploy blue/green deployments.

During a blue/green deployment, the load balancer allows traffic to be routed to the new instances in a deployment group that the latest application revision has been deployed to (the replacement environment), according to the rules you specify, and then blocks traffic from the old instances where the previous application revision was running (the original environment).

After instances in a replacement environment are registered with a load balancer, instances from the original environment are deregistered and, if you choose, terminated.

For a blue/green deployment, you can specify a Classic Load Balancer, Application Load Balancer, or Network Load Balancer in your deployment group. You use the AWS CodeDeploy console or AWS CLI to add the load balancer to a deployment group.

For more information about load balancers in blue/green deployments, see the following topics:

- [Set Up a Load Balancer in Elastic Load Balancing for AWS CodeDeploy Deployments \(p. 224\)](#)
- [Create an Application for a Blue/Green Deployment \(Console\) \(p. 212\)](#)
- [Create a Deployment Group for a Blue/Green Deployment \(Console\) \(p. 222\)](#)

In-place deployments

During an in-place deployment, a load balancer prevents internet traffic from being routed to an instance while it is being deployed to, and then makes the instance available for traffic again after the deployment to that instance is complete.

If a load balancer isn't used during an in-place deployment, internet traffic may still be directed to an instance during the deployment process. As a result, your customers might encounter broken, incomplete, or outdated web applications. When you use an Elastic Load Balancing load balancer with an in-place deployment, instances in a deployment group are deregistered from a load balancer, updated with the latest application revision, and then reregistered with the load balancer as part of the same deployment group after the deployment is successful.

For an in-place deployment, you can specify a Classic Load Balancer, Application Load Balancer, or Network Load Balancer. You can specify the load balancer as part of the deployment group's configuration, or use a script provided by AWS CodeDeploy to implement the load balancer.

To add the load balancer to a deployment group, you use the AWS CodeDeploy console or AWS CLI. For information about specifying a load balancer in a deployment group for in-place deployments, see the following topics:

- [Create an Application for an In-Place Deployment \(Console\) \(p. 211\)](#)
- [Create a Deployment Group for an In-Place Deployment \(Console\) \(p. 220\)](#)
- [Set Up a Load Balancer in Elastic Load Balancing for AWS CodeDeploy Deployments \(p. 224\)](#)

For information about specifying a load balancer for in-place deployments using a script, see the following topic:

- [Use a script to set up a load balancer for an in-place deployment \(p. 48\)](#)

Use a script to set up a load balancer for an in-place deployment

Use the steps in the following procedure to use deployment lifecycle scripts to set up load balancing for in-place deployments.

Note

You should use the `CodeDeployDefault.OneAtATime` deployment configuration only when you are using a script to set up a load balancer for an in-place deployment. Concurrent runs are not supported, and the `CodeDeployDefault.OneAtATime` setting ensures a serial execution of the scripts. For more information about deployment configurations, see [Working with Deployment Configurations in AWS CodeDeploy \(p. 203\)](#).

In the AWS CodeDeploy Samples repository on GitHub, we provide instructions and samples you can adapt to use AWS CodeDeploy Elastic Load Balancing load balancers. These repositories include three sample scripts—`register_with_elb.sh`, `deregister_from_elb.sh`, and `common_functions.sh`—that provide all of the code you need to get going. Simply edit the placeholders in these three scripts, and then reference these scripts from your `appspec.yml` file.

To set up in-place deployments in AWS CodeDeploy with Amazon EC2 instances that are registered with Elastic Load Balancing load balancers, do the following:

1. Download the samples for the type of load balancer you want to use for an in-place deployment:
 - [Classic Load Balancer](#)
 - [Application Load Balancer or Network Load Balancer \(the same script can be used for either type\)](#)
2. Make sure each of your target Amazon EC2 instances has the AWS CLI installed.

3. Make sure each of your target Amazon EC2 instances has an IAM instance profile attached with, at minimum, the `elasticloadbalancing:*` and `autoscaling:*` permissions.
4. Include in your application's source code directory the deployment lifecycle event scripts (`register_with_elb.sh`, `deregister_from_elb.sh`, and `common_functions.sh`).
5. In the `appspec.yml` for the application revision, provide instructions for AWS CodeDeploy to run the `register_with_elb.sh` script during the **ApplicationStart** event and the `deregister_from_elb.sh` script during the **ApplicationStop** event.
6. If the instance is part of an Auto Scaling group, you can skip this step.

In the `common_functions.sh` script:

- If you are using the [Classic Load Balancer](#), specify the names of the Elastic Load Balancing load balancers in `ELB_LIST=""`, and make any changes you need to the other deployment settings in the file.
 - If you are using the [Application Load Balancer](#) or [Network Load Balancer](#), specify the names of the Elastic Load Balancing target group names in `TARGET_GROUP_LIST=""`, and make any changes you need to the other deployment settings in the file.
7. Bundle your application's source code, the `appspec.yml`, and the deployment lifecycle event scripts into an application revision, and then upload the revision. 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 complete.

Integration with Partner Products and Services

AWS CodeDeploy has built-in integration for the following partner products and services:

Ansible	<p>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 will run the playbooks. Alternatively, if you already have a process for building and maintaining your inventory, there's also an Ansible module you can use to install and run the AWS CodeDeploy agent.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Ansible and AWS CodeDeploy
Atlassian – Bamboo and Bitbucket	<p>The AWS CodeDeploy task for Bamboo compresses the directory that contains an AppSpec file into a .zip file, uploads the file to Amazon S3, and then starts the deployment according to the configuration provided in the AWS CodeDeploy application.</p> <p>Atlassian Bitbucket support for AWS CodeDeploy enables you to push code to Amazon EC2 instances directly from the Bitbucket UI, on demand, to any of your deployment groups. This means that after you update code in your</p>

	<p>Bitbucket repository, you do not have to sign in to your continuous integration (CI) platform or Amazon EC2 instances to run a manual deployment process.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Using the AWS CodeDeploy Task for Bamboo• Announcing Atlassian Bitbucket Support for AWS CodeDeploy
Chef	<p>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. This allows you to continue managing your host infrastructure with Chef while using AWS CodeDeploy. The second sample template demonstrates how to use AWS CodeDeploy to orchestrate the running of cookbooks and recipes with chef-solo on each node.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Chef and AWS CodeDeploy
CircleCI	<p>CircleCI provides an automated testing and continuous integration and deployment toolset. After you create an IAM role in AWS to use with CircleCI and configure your deployment parameters in your circle.yml file, you can use CircleCI with AWS CodeDeploy to create application revisions, upload them to an Amazon S3 bucket, and then initiate and monitor your deployments.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Continuous Deployment with AWS CodeDeploy
CloudBees	<p>You can use the AWS CodeDeploy Jenkins plugin, available on CloudBees DEV@cloud, as a post-build action. For example, at the end of a continuous delivery pipeline, you can use it to deploy an application revision to your fleet of servers.</p> <p>Learn more:</p> <ul style="list-style-type: none">• AWS CodeDeploy Jenkins Plugin Now Available on DEV@cloud

Codeship	<p>You can use Codeship to deploy application revisions through AWS CodeDeploy. You can use the Codeship UI to add AWS CodeDeploy to a deployment pipeline for a branch.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Deploy to AWS CodeDeploy• AWS CodeDeploy Integration on Codeship
GitHub	<p>You can use AWS CodeDeploy to deploy application revisions from GitHub repositories. You can also trigger a deployment from a GitHub repository whenever the source code in that repository is changed.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Integrating AWS CodeDeploy with GitHub (p. 53)• Tutorial: Use AWS CodeDeploy to Deploy an Application from GitHub (p. 112)• Automatically Deploy from GitHub Using AWS CodeDeploy
HashiCorp Consul	<p>You can use the open-source HashiCorp Consul tool to help ensure the health and stability of your application environment when you deploy applications in AWS CodeDeploy. You can use Consul to register applications to be discovered during deployment, put applications and nodes in maintenance mode to omit them from deployments, and stop deployments if target instances become unhealthy.</p> <p>Learn more:</p> <ul style="list-style-type: none">• AWS CodeDeploy Deployments with HashiCorp Consul
Jenkins	<p>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.</p> <p>Learn more:</p> <ul style="list-style-type: none">• AWS CodeDeploy Jenkins Plugin• Setting Up the Jenkins Plugin for AWS CodeDeploy

Puppet Labs	<p>AWS provides sample templates for Puppet and AWS CodeDeploy. The first is a Puppet module that will install and start the AWS CodeDeploy agent. This allows you to continue managing your host infrastructure with Puppet while using AWS CodeDeploy. The second sample template demonstrates how to use AWS CodeDeploy to orchestrate the running of modules and manifests with a masterless puppet on each node.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Puppet and AWS CodeDeploy
SaltStack	<p>You can integrate SaltStack infrastructure with AWS CodeDeploy. You can use the AWS CodeDeploy module to install and run the AWS CodeDeploy agent on your minions or, with a couple of simple deployment hooks, you can use AWS CodeDeploy to orchestrate the running of your Salt States.</p> <p>Learn more:</p> <ul style="list-style-type: none">• SaltStack and AWS CodeDeploy
Solano Labs	<p>After your build has passed its tests in Solano CI, a script will run to prepare your application for release. The <code>aws deploy push</code> command will package and push your application through AWS CodeDeploy, and then optionally deploy the application revision to a deployment group and confirm it has been deployed. You can also set up automatic AWS CodeDeploy deployments from your CI build.</p> <p>Learn more:</p> <ul style="list-style-type: none">• AWS CodeDeploy Deployments from Solano CI Builds
TeamCity	<p>You can use the AWS CodeDeploy Runner plugin to deploy applications directly from TeamCity. The plugin adds a TeamCity build step that prepares and uploads an application revision to an Amazon S3 bucket, registers the revision in an AWS CodeDeploy application, creates an AWS CodeDeploy deployment and, if you choose, waits for the deployment to be completed.</p> <p>Learn more:</p> <ul style="list-style-type: none">• AWS CodeDeploy Runner (Download)• AWS CodeDeploy Runner Plugin (Documentation)

Travis CI

You can configure [Travis CI](#) to trigger a deployment in AWS CodeDeploy after a successful build.

Learn more:

- [Travis CI and AWS CodeDeploy Deployments](#)

Topics

- [Integrating AWS CodeDeploy with GitHub \(p. 53\)](#)

Integrating AWS CodeDeploy with GitHub

AWS CodeDeploy supports [GitHub](#), a web-based code hosting and sharing service. AWS CodeDeploy can deploy application revisions stored in GitHub repositories or Amazon S3 buckets to instances.

Topics

- [Video Introduction to AWS CodeDeploy Integration with GitHub \(p. 53\)](#)
- [Deploying AWS CodeDeploy Revisions from GitHub \(p. 53\)](#)
- [GitHub Behaviors with AWS CodeDeploy \(p. 54\)](#)

Video Introduction to AWS CodeDeploy Integration with GitHub

This short video (5:20) demonstrates how to automate application deployments with AWS CodeDeploy from your existing GitHub workflows.

[Video Introduction to AWS CodeDeploy integration with GitHub.](#)

Deploying AWS CodeDeploy Revisions from GitHub

To deploy an application revision from a GitHub repository to instances:

1. Create a revision that's compatible with AWS CodeDeploy and the Amazon EC2 instance type to which you will deploy.

To create a compatible revision, follow the instructions in [Plan a Revision for AWS CodeDeploy \(p. 232\)](#) and [Add an Application Specification File to a Revision for AWS CodeDeploy \(p. 233\)](#).

2. Use a GitHub account to add your revision to a GitHub repository.

To create a GitHub account, see [Join GitHub](#). To create a GitHub repository, see [Create a Repo](#).

3. Use the **Create deployment** page in the AWS CodeDeploy console or the AWS CLI **create-deployment** command to deploy your revision from your GitHub repository to target instances configured for use in AWS CodeDeploy deployments.

If you want to call the **create-deployment** command, you must first use the **Create deployment** page of the console to give AWS CodeDeploy permission to interact with GitHub on behalf of your preferred GitHub account for the specified application. You only need to do this once per application.

To learn how to use the **Create deployment** page to deploy from a GitHub repository, see [Create a Deployment with AWS CodeDeploy \(p. 245\)](#).

To learn how to call the **create-deployment** command to deploy from a GitHub repository, see [Create an EC2/On-Premises Compute Platform Deployment \(CLI\)](#) (p. 252).

To learn how to prepare instances for use in AWS CodeDeploy deployments, see [Working with Instances for AWS CodeDeploy](#) (p. 147).

For more information, see [Tutorial: Use AWS CodeDeploy to Deploy an Application from GitHub](#) (p. 112).

GitHub Behaviors with AWS CodeDeploy

Topics

- [GitHub Authentication with Applications in AWS CodeDeploy](#) (p. 54)
- [AWS CodeDeploy Interaction with Private and Public GitHub Repositories](#) (p. 55)
- [AWS CodeDeploy Interaction with Organization-Managed GitHub Repositories](#) (p. 55)
- [Automatically Deploy from GitHub with AWS CodeDeploy](#) (p. 55)

GitHub Authentication with Applications in AWS CodeDeploy

After you give AWS CodeDeploy permission to interact with GitHub, the association between that GitHub account and application is stored in AWS CodeDeploy. You can link the application to a different GitHub account. You can also revoke permission for AWS CodeDeploy to interact with GitHub.

To link a 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

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. On the AWS CodeDeploy menu, choose **Deployments**.
3. Choose **Create deployment**.

Note

You don't have to create a new deployment. This is currently the only way to link a different GitHub account to an application.

4. From the **Application** drop-down list, choose the application you want to link to a different GitHub account.
5. Next to **Repository type**, choose **My application is stored in GitHub**.
6. In **Connect to GitHub**, do one of the following:
 - To create a connection for AWS CodeDeploy applications to a GitHub account, sign out of GitHub in a separate web browser tab. In **GitHub account**, type a name to identify this connection, and then choose **Connect to GitHub**. The web page prompts you to authorize AWS CodeDeploy to interact with GitHub for your application. Continue to step 2.
 - To use a connection you have already created, in **GitHub account**, select its name, and then choose **Connect to GitHub**. Continue to step 4.
 - To create a connection to a different GitHub account, sign out of GitHub in a separate web browser tab. Choose **Connect to a different GitHub account**, and then choose **Connect to GitHub**. Continue to step 2.
7. If you are not already signed in to GitHub, follow the instructions on the **Sign in** page to sign in with the GitHub account to which you want to link the application.

8. Choose **Authorize application**. GitHub gives AWS CodeDeploy permission to interact with GitHub on behalf of the signed-in GitHub account for the selected application.
9. If you do not want to create a deployment, choose **Cancel**.

To revoke permission for AWS CodeDeploy to interact with GitHub

1. Sign in to [GitHub](#) using credentials for the GitHub account in which you want to revoke AWS CodeDeploy permission.
2. Open the GitHub [Applications](#) page, locate **AWS CodeDeploy** in the list of authorized applications, and then follow the GitHub procedure for revoking authorization for an application.

AWS CodeDeploy Interaction with Private and Public GitHub Repositories

AWS CodeDeploy supports the deployment of applications from 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 to which your GitHub account has access. However, AWS CodeDeploy only reads from GitHub repositories. It will not write to any of your private GitHub repositories.

AWS CodeDeploy Interaction with Organization-Managed GitHub Repositories

By default, GitHub repositories that are managed by an organization (as opposed to your account's own private or public repositories) do not grant access to third-party applications, including AWS CodeDeploy. Your deployment will fail if an organization's third-party application restrictions are enabled in GitHub and you attempt to deploy code from its GitHub repository. There are two ways to resolve this issue.

- As an organization member, you can ask the organization owner to approve access to AWS CodeDeploy. The steps for requesting this access depend on whether you have already authorized AWS CodeDeploy for your individual account:
 - If you have authorized access to AWS CodeDeploy in your account, see [Requesting Organization Approval for Your Authorized Applications](#).
 - If you have not yet authorized access to AWS CodeDeploy in your account, see [Requesting Organization Approval for Third-Party Applications](#).
- The organization owner can disable all third-party application restrictions for the organization. For information, see [Disabling Third-Party Application Restrictions for Your Organization](#).

For more information, see [About Third-Party Application Restrictions](#).

Automatically Deploy from GitHub with AWS CodeDeploy

You can trigger a deployment from a GitHub repository whenever the source code changes. For instructions, see [Automatically Deploy from GitHub Using AWS CodeDeploy](#).

Integration Examples from the Community

The following sections provide links to blog posts, articles, and community-provided examples.

Note

These links are provided for informational purposes only, and should not be considered either a comprehensive list or an endorsement of the content of the examples. AWS is not responsible for the content or accuracy of external content.

Blog posts

- [Automating AWS CodeDeploy Provisioning in AWS CloudFormation](#)

Learn how to provision the deployment of an application in AWS CodeDeploy by using AWS CloudFormation.

Published January 2016

- [AWS Toolkit for Eclipse Integration with AWS CodeDeploy \(Part 1\)](#)

[AWS Toolkit for Eclipse Integration with AWS CodeDeploy \(Part 2\)](#)

[AWS Toolkit for Eclipse Integration with AWS CodeDeploy \(Part 3\)](#)

Learn how Java developers can use the AWS CodeDeploy plugin for Eclipse to deploy web applications to AWS directly from Eclipse development environments.

Published February 2015

- [Automatically Deploy from GitHub Using AWS CodeDeploy](#)

Learn how automatic deployments from GitHub to AWS CodeDeploy can be used to create an end-to-end pipeline — from source control to your testing or production environments.

Published December 2014

Videos

- **Hosting ASP.NET 5 Apps in AWS with Docker and AWS CodeDeploy**

Learn how AWS CodeDeploy can be used to deploy ASP.NET 5 applications to an Internet Information Services (IIS) server on Microsoft Windows operating systems.

[Hosting ASP.NET 5 Apps in AWS with Docker and AWS CodeDeploy](#)

Published October 2015

Duration: 47:37

- **Mastering AWS CodeDeploy with Jenkins and Puppet**

Learn how to use the open-source tools Jenkins and Puppet with AWS CodeDeploy.

[Mastering AWS CodeDeploy with Jenkins and Puppet](#)

Published May 2015

Duration: 49:31

AWS CodeDeploy Tutorials

This section includes some tutorials to help you learn how to use AWS CodeDeploy.

If you haven't completed it already, we recommend you start with [Step 5: Try the AWS CodeDeploy Sample Deployment Wizard \(p. 29\)](#). It requires no prior experience with AWS CodeDeploy. It guides you through the steps required to deploy one of our sample application revisions to Amazon EC2 instances.

Important

Before you begin, complete the prerequisites in [Getting Started with AWS CodeDeploy \(p. 19\)](#).

The procedures in these tutorials provide suggestions for the location in which to store files (for example, c:\temp) and the names to give to buckets, subfolders, or files (for example, codedeploydemobucket, HelloWorldApp, and CodeDeployDemo-EC2-Trust.json, respectively), but you are not required to use them. Just be sure to substitute your file locations and names as you perform the procedures.

Topics

- [Tutorial: Deploy WordPress to an Amazon EC2 Instance \(Amazon Linux or Red Hat Enterprise Linux and Linux, macOS, or Unix\) \(p. 57\)](#)
- [Tutorial: Deploy a "Hello, World!" Application with AWS CodeDeploy \(Windows Server\) \(p. 75\)](#)
- [Tutorial: Deploy an Application to an On-Premises Instance with AWS CodeDeploy \(Windows Server, Ubuntu Server, or Red Hat Enterprise Linux\) \(p. 90\)](#)
- [Tutorial: Use AWS CodeDeploy to Deploy an Application to an Auto Scaling Group \(p. 96\)](#)
- [Tutorial: Use AWS CodeDeploy to Deploy an Application from GitHub \(p. 112\)](#)

Tutorial: Deploy WordPress to an Amazon EC2 Instance (Amazon Linux or Red Hat Enterprise Linux and Linux, macOS, or Unix)

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 EC2 instance running Amazon Linux or Red Hat Enterprise Linux (RHEL).

Not what you're looking for?

- To practice deploying to an Amazon EC2 instance running Windows Server instead, see [Tutorial: Deploy a "Hello, World!" Application with AWS CodeDeploy \(Windows Server\) \(p. 75\)](#).
- To practice deploying to an on-premises instance instead of an Amazon EC2 instance, see [Tutorial: Deploy an Application to an On-Premises Instance with AWS CodeDeploy \(Windows Server, Ubuntu Server, or Red Hat Enterprise Linux\) \(p. 90\)](#).

This tutorial builds on concepts introduced in [Step 5: Try the AWS CodeDeploy Sample Deployment Wizard \(p. 29\)](#). If you have not yet completed it, you may want to start there first.

This tutorial's steps are presented from the perspective of a local development machine running Linux, macOS, or Unix. Although you can complete most of these steps on a local machine running Windows, you will need to adapt 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 [Getting Started with AWS CodeDeploy](#) (p. 19). These include configuring your IAM user account, installing or upgrading the AWS CLI, and creating an IAM instance profile and a service role.

Topics

- [Step 1: Launch and Configure an Amazon Linux or Red Hat Enterprise Linux Amazon EC2 Instance](#) (p. 58)
- [Step 2: Configure Your Source Content to Be Deployed to the Amazon Linux or Red Hat Enterprise Linux Amazon EC2 Instance](#) (p. 60)
- [Step 3: Upload Your WordPress Application to Amazon S3](#) (p. 63)
- [Step 4: Deploy Your WordPress Application](#) (p. 67)
- [Step 5: Update and Redeploy Your WordPress Application](#) (p. 71)
- [Step 6: Clean Up Your WordPress Application and Related Resources](#) (p. 73)

Step 1: Launch and Configure an Amazon Linux or Red Hat Enterprise Linux Amazon EC2 Instance

To deploy the WordPress application with AWS CodeDeploy, you'll need an Amazon EC2 instance running Amazon Linux or Red Hat Enterprise Linux (RHEL). The Amazon EC2 instance requires a new inbound security rule that allows HTTP connections. This rule is needed in order to view the WordPress page in a browser after it is successfully deployed.

Follow the instructions in [Working with Instances for AWS CodeDeploy](#) (p. 147). When you get to the part in those instructions 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, then the instructions in [Step 4: Deploy Your WordPress Application](#) (p. 67) may produce unexpected results.)

After you've followed the instructions to launch the Amazon EC2 instance, return to this page, and continue to the next section. Do not continue on to [Create an Application with AWS CodeDeploy](#) (p. 209) as a next step.

Connect to Your Amazon Linux or RHEL Amazon EC2 Instance

After your new Amazon EC2 instance is launched, follow these instructions to practice connecting to it.

1. Use the **ssh** command (or an SSH-capable terminal emulator like [PuTTY](#)) to connect to your Amazon Linux or RHEL Amazon EC2 instance. You will need the public DNS address of the instance and the private key for the key pair you used when you started the Amazon EC2 instance. For more information, see [Connect to Your Instance](#).

For example, if the 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
```

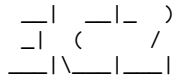
Replace **/path/to/codedeploydemo.pem** with the path to your **.pem** file and the example DNS address with the address to your Amazon Linux or RHEL Amazon EC2 instance.

Note

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

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

2. After you are signed in, you will see the AMI banner for the Amazon EC2 instance. For Amazon Linux, it should look like this:

 Amazon Linux AMI

3. Confirm the AWS CodeDeploy agent installed properly when you setup your Amazon EC2 instance by typing the following:

```
sudo service codedeploy-agent status
```

For more information about determining the status of the AWS CodeDeploy agent, see [Verify the AWS CodeDeploy Agent Is Running \(p. 132\)](#).

If the AWS CodeDeploy agent is not installed, follow the instructions in [Install or reinstall the AWS CodeDeploy agent for Amazon Linux or RHEL \(p. 135\)](#).

4. You can now sign out of the running Amazon EC2 instance.

Warning

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

Add Inbound Rule Allowing HTTP Traffic to Your Amazon Linux or RHEL Amazon EC2 Instance

The next step confirms your Amazon EC2 instance has an open HTTP port so you can see the deployed WordPress application's home page in a browser.

1. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. Choose **Instances** and select your instance.
3. Under **Security groups**, choose **view inbound rules**.

You should see a list of rules in your security group similar to the following:

Security Groups associated with i-1234567890abcdef0			
Ports	Protocol	Source	launch-wizard- N
22	tcp	0.0.0.0/0	#

4. Under **Security groups**, choose your Amazon EC2 instance's security group. It may be called **launch-wizard-**N****. The **N** in the name is a number assigned to your security group when your instance was created.

Choose the **Inbound** tab. Your instance's security group is properly configured if you see a rule with the following values:

- **Type:** HTTP
- **Protocol:** TCP
- **Port Range:** 80

- **Source:** Custom
5. If you do not see a rule with the values listed in the previous step, use the procedures in [Adding Rules to a Security Group](#) to add them to a new security rule.

Step 2: Configure Your Source Content to Be Deployed to the Amazon Linux or Red Hat Enterprise Linux Amazon EC2 Instance

Now it's time to configure your application's source content so you have something to deploy to the instance.

Topics

- [Get the Source Code](#) (p. 60)
- [Create Scripts to Run Your Application](#) (p. 61)
- [Add an Application Specification File](#) (p. 62)

Get the Source Code

For this tutorial, you 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.

For these steps, we assume you downloaded a copy of the WordPress source code to the `/tmp` directory on your development machine. (You can choose any directory you like, but remember to substitute your location for `/tmp` wherever it is specified in these steps.)

Choose one of the following two options to copy the WordPress source files to your development machine. The first option uses built-in command-line calls. The second option uses Git.

Topics

- [To get a copy of the WordPress source code \(built-in command-line calls\)](#) (p. 60)
- [To get a copy of the WordPress source code \(Git\)](#) (p. 61)

To get a copy of the WordPress source code (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. Call the **unzip**, **mkdir**, **cp**, and **rm** commands to:
 - Unpack the `master.zip` file into the `/tmp/WordPress_Temp` directory (folder).
 - Copy its unzipped contents to the `/tmp/WordPress` destination folder.
 - Delete the temporary `/tmp/WordPress_Temp` folder and `master` file.

Run the commands one at a time:

```
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 get a copy of the WordPress source code (Git)

1. Download and install [Git](#) on your development machine.
2. In the `/tmp/WordPress` folder, call the **git init** command.
3. 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.

Create Scripts to Run Your Application

Next, create a folder and scripts in the directory. AWS CodeDeploy uses these scripts 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:

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

2. Create an `install_dependencies.sh` file in `/tmp/WordPress/scripts`. Add the following lines to the file. This `install_dependencies.sh` script installs Apache, MySQL, and PHP. It also adds MySQL support to PHP.

```
#!/bin/bash  
sudo yum install -y httpd24 php70 mysql56-server php70-mysqld
```

3. Create a `start_server.sh` file in `/tmp/WordPress/scripts`. Add the following lines to the file. This `start_server.sh` script starts Apache and MySQL.

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

4. Create a `stop_server.sh` file in `/tmp/WordPress/scripts`. Add the following lines to the file. This `stop_server.sh` script stops Apache and MySQL.

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

5. Create a `create_test_db.sh` file in `/tmp/WordPress/scripts`. Add the following lines to the file. This `create_test_db.sh` script uses MySQL to create a `test` database for WordPress to use.

```
#!/bin/bash
mysql -uroot <<CREATE_TEST_DB
CREATE DATABASE test;
CREATE_TEST_DB
```

6. Finally, create a `change_permissions.sh` script in `/tmp/WordPress/scripts`. This is used to change the folder permissions in Apache.

Important

This script updated permissions on the `/tmp/WordPress` folder so that anyone can write to it. This is required so that WordPress can write to its database during [Step 5: Update and Redeploy Your WordPress Application \(p. 71\)](#). After the WordPress application is set up, run the following command to update permissions to a more secure setting:

```
chmod -R 755 /var/www/html/WordPress
```

```
#!/bin/bash
chmod -R 777 /var/www/html/WordPress
```

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

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

Add an Application Specification File

Next, add an application specification file (AppSpec file), a [YAML](#)-formatted file used by AWS CodeDeploy to:

- Map the source files in your application revision to their destinations on the target Amazon EC2 instance.
- Specify custom permissions for deployed files.
- Specify scripts to be run on the target Amazon EC2 instance during the deployment.

The AppSpec file must be named `appspec.yml`. It must be placed in the root directory of the application's source code. In this tutorial, the root directory is `/tmp/WordPress`

With your text editor, create a file named `appspec.yml`. 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
    - location: scripts/create_test_db.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

AWS CodeDeploy uses 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 runs the specified scripts as `root` in the `/var/www/html/WordPress/scripts` folder on the target Amazon EC2 instance at 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 stops the deployment and marks the deployment as failed.

For more information about these settings, see the [AWS CodeDeploy AppSpec File Reference \(p. 306\)](#).

Important

The locations and numbers of spaces between each of the items in this file are important. If the spacing is incorrect, AWS CodeDeploy raises an error that might be difficult to debug. For more information, see [AppSpec File Spacing \(p. 327\)](#).

Step 3: Upload Your WordPress Application to Amazon S3

Now you will prepare and upload your source content to a location from which AWS CodeDeploy can deploy it. 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

Although it's not covered in this tutorial, you can use AWS CodeDeploy to deploy applications from GitHub repositories to instances. For more information, see [Integrating AWS CodeDeploy with GitHub \(p. 53\)](#).

Topics

- [Provision an Amazon S3 Bucket \(p. 64\)](#)
- [Prepare the Application's Files for the Bucket \(p. 65\)](#)
- [Bundle the Application's Files into a Single Archive File and Push the Archive File \(p. 66\)](#)

Provision an Amazon S3 Bucket

Create a storage container or *bucket* in Amazon S3—or use an existing bucket. Make sure you can upload the revision to the bucket and that Amazon EC2 instances used in deployments can download the revision from the bucket.

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

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 this tutorial. The Amazon S3 bucket must be created in the same AWS region where your target Amazon EC2 instances are launched. 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.

Topics

- [To create an Amazon S3 bucket \(CLI\) \(p. 64\)](#)
- [To create an Amazon S3 bucket \(console\) \(p. 64\)](#)
- [Give permissions to the Amazon S3 bucket and your IAM user \(p. 64\)](#)

To create an Amazon S3 bucket (CLI)

Call the `mb` command to create an Amazon S3 bucket named `codedeploydemobucket`:

```
aws s3 mb s3://codedeploydemobucket
```

To create an Amazon S3 bucket (console)

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. In the Amazon S3 console, choose **Create bucket**.
3. In the **Bucket name** box, type a name for the bucket.
4. In the **Region** list, choose the target region, and then choose **Create**.

Give permissions to the Amazon S3 bucket and your IAM user

You must have permissions to upload to the Amazon S3 bucket. You can specify these permissions through an Amazon S3 bucket policy. For example, in the following Amazon S3 bucket policy, using the wildcard character (*) allows AWS account 111122223333 to upload files to any directory in the Amazon S3 bucket named `codedeploydemobucket`:

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



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

To view your AWS account ID, see [Finding Your AWS Account ID](#).

Now is a good time to verify the Amazon S3 bucket will allow download requests from each participating Amazon EC2 instance. You can specify this through an Amazon S3 bucket policy. For example, in the following Amazon S3 bucket policy, using the wildcard character (*) allows any Amazon EC2 instance with an attached IAM instance profile containing the ARN `arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo` to download files from any directory in 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"
        ]
      }
    }
  ]
}
```

For information about how to generate and attach an Amazon S3 bucket policy, see [Bucket Policy Examples](#).

Your account must have permission to upload the revision to the Amazon S3 bucket. One way to specify this is through an IAM policy. The following custom IAM user policy allows your IAM user to upload revisions anywhere in the Amazon S3 bucket named `codedeploydemobucket`:

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

For information about how to create and attach an IAM policy, see [Working with Policies](#).

Prepare the Application's Files for the Bucket

Make sure the WordPress application files, the AppSpec file, and the scripts are organized on your development machine similar to the following:

```
/tmp/  
|--WordPress/  
|  |-- appspec.yml  
|  |-- scripts/  
|      |-- change_permissions.sh  
|      |-- create_test_db.sh  
|      |-- install_dependencies.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

Bundle the WordPress application files and 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 the file bundling will start at your current folder. For example, if your current folder is `/tmp` instead of `/tmp/WordPress`, then the bundling will start with files and subfolders in the `tmp` folder, which may include more than the `WordPress` subfolder.

2. Call the **create-application** command to register a new application 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.

```
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 deploy the sample WordPress application revision you uploaded to Amazon S3. You can use the AWS CLI or the AWS CodeDeploy console to deploy the revision and monitor the deployment's progress. After the application revision is successfully deployed, you check the results.

Topics

- [Deploy Your Application Revision with AWS CodeDeploy \(p. 67\)](#)
- [Monitor and Troubleshoot Your Deployment \(p. 69\)](#)
- [Verify Your Deployment \(p. 70\)](#)

Deploy Your Application Revision with AWS CodeDeploy

Topics

- [To deploy your application revision \(CLI\) \(p. 67\)](#)
- [To deploy your application revision \(console\) \(p. 68\)](#)

To deploy your application revision (CLI)

1. The deployment needs a deployment group. However, before you create the deployment group, you need a service role ARN. A service role is an IAM role that gives a service permission to act on your behalf. In this case, the service role gives AWS CodeDeploy permission to access your Amazon EC2 instances to expand (read) their Amazon EC2 instance tags.

You should have already followed the instructions in [Create a Service Role \(CLI\) \(p. 23\)](#) to create a service role. To get the ARN of the service role, see [Get the Service Role ARN \(CLI\) \(p. 25\)](#).

2. Now that you have the 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**:

```
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
```

Note

The **create-deployment-group** command provides support for creating triggers that result in the sending of Amazon SNS notifications to topic subscribers about specified events in deployments and instances. The command also supports options for automatically rolling back deployments and setting up alarms to stop deployments when monitoring thresholds in Amazon CloudWatch alarms are met. Commands for these actions are not included in this tutorial.

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-group-name WordPress_DepGroup \  
  --deployment-config-name CodeDeployDefault.OneAtATime \  
  --revision-name WordPressApp.zip \  
  --s3-location bucket=codedeploydemobucket,key=WordPressApp.zip,object=WordPressApp.zip
```

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

To deploy your application revision (console)

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

You should have already followed the instructions in [Create a Service Role \(Console\)](#) (p. 21) to create a service role. To get the ARN of the service role, see [Get the Service Role ARN \(Console\)](#) (p. 24).

2. Now that you have the ARN, use 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

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

3. If the **Applications** page is not displayed, on the AWS CodeDeploy menu, choose **Applications**.
4. In the list of applications, choose **WordPress_App**.
5. Under **Deployment groups**, choose **Create deployment group**.
6. In **Deployment group name**, type **WordPress_DepGroup**.
7. Under **Deployment type**, choose **In-place deployment**.
8. In **Environment configuration**, choose the **Amazon EC2 instances** tab.
9. In the **Key** box, type **Name**.
10. In the **Value** box, type **CodeDeployDemo**.

Note

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

11. In the **Deployment configuration** drop-down list, choose **CodeDeployDefault.OneAtATime**.
12. In the **Service role ARN** drop-down list, choose the service role ARN, and then choose **Create deployment group**.
13. On the **Application details** page, select the button next to the new deployment group. From the **Actions** menu, choose **Deploy new revision**.
14. In the **Application** drop-down list, choose **WordPress_App**.
15. In the **Deployment group** drop-down list, choose **WordPress_DepGroup**.
16. Next to **Repository type**, choose **My application is stored in Amazon S3**. In **Revision location**, type the location of the sample WordPress application revision you previously uploaded to Amazon S3. To get the location:
 - a. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
 - b. In the list of buckets, choose **codedeploydemobucket** (or the name of the bucket where you uploaded your application revision).
 - c. In the list of objects, choose **WordPressApp.zip**.
 - d. On the **Overview** tab, copy the value of the **Link** field to your clipboard.

It might look something like this:

`https://s3.amazonaws.com/codedeploydemobucket/WordPressApp.zip`

- e. Return to the AWS CodeDeploy console, and in **Revision location**, paste the **Link** field value.
17. If a message appears in the **File type** list stating the file type could not be detected, choose **.zip**.
18. (Optional) Type a comment in the **Deployment description** box.
19. From the **Deployment configuration** drop-down list, choose **CodeDeployDefault.OneAtATime**.
20. Choose **Deploy**. Information about your newly created deployment appears on the **Deployments** page.

Note

To get the current status of the deployment, choose the **Refresh** button next to the table.

Monitor and Troubleshoot Your Deployment

Topics

- [To monitor and troubleshoot your deployment \(CLI\)](#) (p. 69)
- [To monitor and troubleshoot your deployment \(console\)](#) (p. 69)

To monitor and troubleshoot your deployment (CLI)

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. Call the **get-deployment** command with the deployment ID:

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

3. The command returns the deployment's overall status. If successful, the value is **Succeeded**.

If the overall status is **Failed**, you can call commands such as [list-deployment-instances](#) and [get-deployment-instance](#) to troubleshoot. For more troubleshooting options, see [Analyzing log files to investigate deployment failures on instances](#) (p. 355).

To monitor and troubleshoot your deployment (console)

On the **Deployments** page in the AWS CodeDeploy console, you can monitor your deployment's status in the **Status** column.

Note

To get the current status of the deployment, choose the **Refresh** button above 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, choose the arrow next to the deployment ID. After a deployment fails, a message that describes the reason for the failure appears in **Details**.
2. In **Instances**, choose **View all instances**. More information about the deployment is displayed. After a deployment fails, you might be able to determine on which Amazon EC2 instances and at which step the deployment failed.

Note

If you don't see **Instances**, choose the **Refresh** button above the table. After the **Status** column changes from **In progress** to **Created**, **Instances** should appear.

3. If you want to do more troubleshooting, you can use a technique like the one described in [View Instance Details \(p. 197\)](#). You can also analyze the deployment log files on an Amazon EC2 instance. For more information, see [Analyzing log files to investigate deployment failures on instances \(p. 355\)](#).

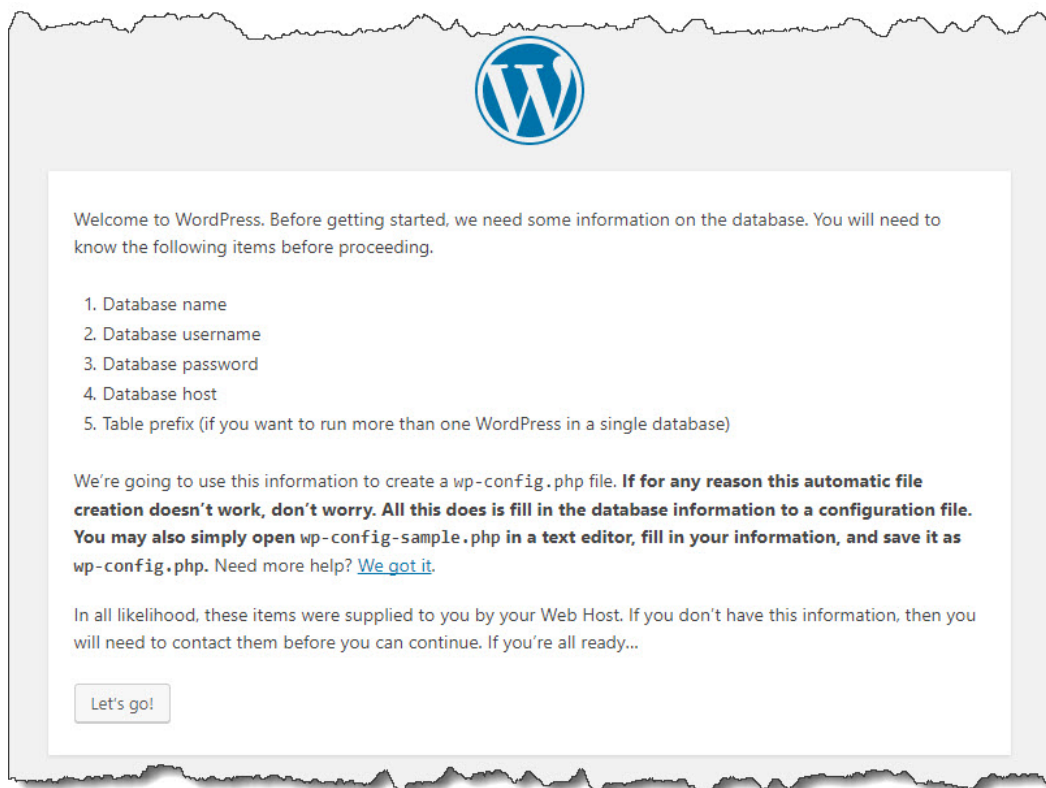
Verify Your Deployment

After your deployment is successful, verify your WordPress installation is working. Use the public DNS address of the Amazon EC2 instance, followed by `/wordpress`, to view your site in a web browser. (To get the public DNS value, in the Amazon EC2 console, choose the Amazon EC2 instance, and on the **Description** tab, look for the value of **Public DNS**.)

For example, if the public DNS address of your Amazon EC2 instance 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
```

When you view the site in your browser, you should see a WordPress welcome page that looks similar to the following:



If your Amazon EC2 instance does not have an HTTP inbound rule added to its security group, then the WordPress welcome page does not appear. If you see a message that says the remote server is not responding, make sure the security group for your Amazon EC2 instance has the inbound rule. For more

information, see [Add Inbound Rule Allowing HTTP Traffic to Your Amazon Linux or RHEL Amazon EC2 Instance](#) (p. 59).

Step 5: Update and Redeploy Your WordPress Application

Now that you've successfully deployed your application revision, update the WordPress code on the development machine, and then use AWS CodeDeploy to redeploy the site. Afterward, you should see the code changes on the Amazon EC2 instance.

Topics

- [Set Up the WordPress Site](#) (p. 71)
- [Modify the Site](#) (p. 71)
- [Redeploy the Site](#) (p. 72)

Set Up the WordPress Site

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

1. Type your site's URL into your web browser. The URL is the public DNS address of the Amazon EC2 instance plus a `/WordPress` extension. For this example WordPress site (and example Amazon EC2 instance public DNS address), the URL is **`http://ec2-01-234-567-890.compute-1.amazonaws.com/WordPress`**.
2. If you haven't set up the site yet, the WordPress default welcome page appears. Choose **Let's go!**.
3. To use the default MySQL database, on the database configuration page, type the following values:
 - **Database Name:** `test`
 - **User Name:** `root`
 - **Password:** Leave blank.
 - **Database Host:** `localhost`
 - **Table Prefix:** `wp_`

Choose **Submit** to set up the database.

4. Continue the site setup. On the **Welcome** page, fill in any values you want, and choose **Install WordPress**. When the installation is complete, you can sign in to your dashboard.

Important

During the deployment of the WordPress application, the `change_permissions.sh` script updated permissions of the `/tmp/WordPress` folder so anyone can write to it. Now is a good time to run the following command to restrict permissions so that only you, the owner, can write to it:

```
chmod -R 755 /var/www/html/WordPress
```

Modify the Site

To modify the WordPress site, go to the application's folder on your development machine:

```
cd /tmp/WordPress
```

To modify some of the site's colors, in the `wp-content/themes/twentyfifteen/style.css` file, use a text editor or **sed** to change `#fff` to `#768331`.

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

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

On macOS, Unix, or other systems with BSD **sed**, use:

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

Redeploy the Site

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

Bundle and upload the changes to Amazon S3, as described in [Bundle the Application's Files into a Single Archive File and Push the Archive File \(p. 66\)](#). (As you follow those instructions, remember that you do not need to create an application.) Give the new revision the same key as before (**WordPressApp.zip**). Upload it to the same Amazon S3 bucket you created earlier (for example, **codedeploydemobucket**).

Use the AWS CLI, the AWS CodeDeploy console, or the AWS CodeDeploy APIs to redeploy the site.

Topics

- [To redeploy the site \(CLI\) \(p. 72\)](#)
- [To redeploy the site \(console\) \(p. 72\)](#)

To redeploy the site (CLI)

Call the **create-deployment** command to create a deployment based on the newly uploaded revision. Use 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 deployment, as described in [Monitor and Troubleshoot Your Deployment \(p. 69\)](#).

After AWS CodeDeploy has redeployed the site, revisit the site in your web browser to verify the colors have been changed. (You might need to refresh your browser.) If the colors have been changed, congratulations! You have successfully modified and redeployed your site!

To redeploy the site (console)

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#).

2. On the AWS CodeDeploy menu, choose **Deployments**.
3. Choose **Create deployment**.
4. On the **Create deployment** page:
 - a. In the **Application** list, choose **WordPress_App**.

Note

If no entries are displayed, make sure the correct region is selected. On the navigation bar, in the region selector, choose one of the regions listed in [Region and Endpoints](#) in the *AWS General Reference*. AWS CodeDeploy is supported in these regions only.

- b. In the **Deployment group** list, choose **WordPress_DepGroup**.
- c. In the **Repository type** area, choose **My application is stored in Amazon S3**, and then copy your revision's Amazon S3 link into the **Revision location** box. To find the link value:
 - i. In a separate browser tab:

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**, and then choose your revision, **WordPressApp.zip**.
 - ii. If the **Properties** pane is not visible in the Amazon S3 console, choose the **Properties** button.
 - iii. In the **Properties** pane, copy the value of the **Link** field into the **Revision location** box in the AWS CodeDeploy console.
- d. If a message appears saying the file type could not be detected, choose **.zip**.
- e. Leave the **Deployment description** box blank.
- f. In the **Deployment configuration** list, choose **CodeDeployDefault.OneAtATime**, and then choose **Deploy**.

To update the deployment's status, choose the **Refresh** button above the table.

You can check the status of the deployment, as described in [Monitor and Troubleshoot Your Deployment \(p. 69\)](#).

After AWS CodeDeploy has redeployed the site, revisit the site in your web browser to verify the colors have been changed. (You might need to refresh your browser.) If the colors have been changed, congratulations! You have successfully modified and redeployed your site!

Step 6: Clean Up Your WordPress Application and Related Resources

You've now successfully made an update to the WordPress code and redeployed the site. To avoid ongoing charges for resources you created for this tutorial, you should delete:

- Any AWS CloudFormation stacks (or terminate any Amazon EC2 instances, if you created them outside of AWS CloudFormation).
- Any Amazon S3 buckets.
- The `WordPress_App` application in AWS CodeDeploy.

You can use the AWS CLI, the AWS CloudFormation, Amazon S3, Amazon EC2, and AWS CodeDeploy consoles, or the AWS APIs to perform the cleanup.

Topics

- [To clean up resources \(CLI\) \(p. 74\)](#)
- [To clean up resources \(console\) \(p. 74\)](#)
- [What's Next? \(p. 75\)](#)

To clean up resources (CLI)

1. If you used our AWS CloudFormation template for this tutorial, call the **delete-stack** command against the stack named **CodeDeployDemoStack**. This will terminate all accompanying Amazon EC2 instances and delete all accompanying IAM roles the stack 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**. This will delete the bucket and all objects in the bucket:

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

3. To delete the **WordPress_App** application, call the **delete-application** command. This will also delete all associated deployment group records and deployment records for the application:

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

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

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

To clean up resources (console)

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 **Filter** box, type the AWS CloudFormation stack name you created earlier (for example, **CodeDeployDemoStack**).
3. Select the box beside stack name. In the **Actions** menu, choose **Delete Stack**.

AWS CloudFormation deletes the stack, terminates all accompanying Amazon EC2 instances, and deletes all accompanying IAM roles.

To terminate Amazon EC2 instances you 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** list, choose **Instances**.

3. In the search box, type the name of the Amazon EC2 instance you want to terminate (for example, **CodeDeployDemo**), and then press Enter.
4. Choose the Amazon EC2 instance name.
5. In the **Actions** menu, point to **Instance State**, and then choose **Terminate**. When prompted, choose **Yes, Terminate**.

Repeat these steps for each instance.

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 list of buckets, browse to and choose the name of the Amazon S3 bucket you created earlier (for example, **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**. In the **Actions** menu, choose **Delete**. When prompted to confirm the deletion, choose **OK**.
4. After the bucket is empty, you can delete the bucket. In the list of buckets, choose the row of the bucket (but not the bucket name). Choose **Delete bucket**, and when prompted to confirm, choose **OK**.

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
Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).
2. On the AWS CodeDeploy menu, choose **Applications**.
3. In the list of applications, choose **WordPress_App**.
4. On the **Application details** page, in **Deployment groups**, choose the button next to the deployment group. On the **Actions** menu, choose **Delete**. When prompted, type the name of the deployment group to confirm you want to delete it, and then choose **Delete**.
5. At the bottom of the **Application details** page, choose **Delete application**.
6. When prompted, type the name of the application to confirm you want to delete it, and then choose **Delete**.

What's Next?

If you've arrived here, congratulations! You have successfully completed an AWS CodeDeploy deployment, and then updated your site's code and redeployed it.

Tutorial: Deploy a "Hello, World!" Application with AWS CodeDeploy (Windows Server)

In this tutorial, you will deploy a single web page to a single Windows Server Amazon EC2 instance running Internet Information Services (IIS) as its web server. This web page will display a simple "Hello, World!" message.

Not what you're looking for?

- To practice deploying to an Amazon Linux or Red Hat Enterprise Linux (RHEL) Amazon EC2 instance instead, see [Tutorial: Deploy WordPress to an Amazon EC2 Instance \(Amazon Linux or Red Hat Enterprise Linux and Linux, macOS, or Unix\)](#) (p. 57).
- To practice deploying to an on-premises instance instead, see [Tutorial: Deploy an Application to an On-Premises Instance with AWS CodeDeploy \(Windows Server, Ubuntu Server, or Red Hat Enterprise Linux\)](#) (p. 90).

This tutorial builds on concepts that were introduced in the [Step 5: Try the AWS CodeDeploy Sample Deployment Wizard](#) (p. 29). If you have not yet completed it, you may want to do that first.

This tutorial's steps are presented from a Windows perspective. Although you can complete most of these steps on a local machine running Linux, macOS, or Unix, you will need to adapt those that cover Windows-based directory paths such as `c:\temp`. Also, if you want to connect to the Amazon EC2 instance, you will need a client application that is capable of connecting through Remote Desktop Protocol (RDP) to the Amazon EC2 instance running Windows Server. (Windows includes an RDP connection client application by default.)

Before you start this tutorial, you must complete the prerequisites in [Getting Started with AWS CodeDeploy](#) (p. 19), including configuring your IAM user, installing or upgrading the AWS CLI, and creating an IAM instance profile and a service role.

Topics

- [Step 1: Launch a Windows Server Amazon EC2 Instance](#) (p. 76)
- [Step 2: Configure Your Source Content to Deploy to the Windows Server Amazon EC2 Instance](#) (p. 77)
- [Step 3: Upload Your "Hello, World!" Application to Amazon S3](#) (p. 79)
- [Step 4: Deploy Your "Hello, World!" Application](#) (p. 82)
- [Step 5: Update and Redeploy Your "Hello, World!" Application](#) (p. 86)
- [Step 6: Clean Up Your "Hello, World!" Application and Related Resources](#) (p. 88)

Step 1: Launch a Windows Server Amazon EC2 Instance

To deploy the "Hello, World!" application with AWS CodeDeploy, you need an Amazon EC2 instance running Windows Server.

Follow the instructions in [Working with Instances for AWS CodeDeploy](#) (p. 147). When you are ready to assign 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, then the instructions in [Step 4: Deploy Your "Hello, World!" Application](#) (p. 82) might produce unexpected results.)

After you've launched the Amazon EC2 instance, return to this page, and continue to the next section. Do not continue on to [Create an Application with AWS CodeDeploy](#) (p. 209) as a next step.

Connect to Your Amazon EC2 Instance

After your Amazon EC2 instance is launched, follow these instructions to practice connecting to it.

Note

In these instructions, we assume you are running Windows and the Windows Desktop Connection client application. For information, see [Connecting to Your Windows Instance Using RDP](#). You might 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**, choose **Instances**.
3. Browse to and choose your Windows Server instance in the list.
4. Choose **Connect**.
5. Choose **Get Password**.
6. Choose **Browse**. Browse to and choose the Amazon EC2 instance key pair file associated with the Windows Server Amazon EC2 instance, and then choose **Open**.
7. Choose **Decrypt Password**. Make a note of the password that is displayed. You need it in step 10.
8. Choose **Download Remote Desktop File**, and then open the file.
9. If you are prompted to connect even though the publisher of the remote connection can't be identified, proceed.
10. Type the password you noted in step 7, and then proceed. (If your RDP connection client application prompts you for a user name, type **Administrator**.)
11. If you are prompted to connect even though the identity of the remote computer cannot be verified, proceed.
12. After you are connected, the desktop of the Amazon EC2 instance running Windows Server is displayed.
13. You can now sign out of the running Amazon EC2 instance.

Warning

Do not stop or terminate the instance. Otherwise, AWS CodeDeploy can't deploy to it.

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

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

Topics

- [Create the Web Page \(p. 77\)](#)
- [Create a Script to Run Your Application \(p. 78\)](#)
- [Add an Application Specification File \(p. 78\)](#)

Create the Web Page

1. Create a subdirectory (subfolder) named `HelloWorldApp` in your `c:\temp` folder, and 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`. If you use a different location or subfolder name, be sure to use it throughout this tutorial.

2. Use a text editor 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" "http://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="https://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 where the `index.html` file is saved, use a text editor 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 (AppSpec file) in addition to the web page and batch script file. The AppSpec file is a [YAML](#)-formatted file used by AWS CodeDeploy to:

- Map the source files in your application revision to their destinations on the instance.
- Specify scripts to be run on the instance during the deployment.

The AppSpec file must be named `appspec.yml`. It must be placed in the application source code's root folder.

1. In the same subfolder where the `index.html` and `before-install.bat` files are saved, use a text editor 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 and mark the deployment to the Amazon EC2 instance as failed.

For more information about these settings, see the [AWS CodeDeploy AppSpec File Reference \(p. 306\)](#).

Important

The locations and numbers of spaces between each of the items in this file are 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 \(p. 327\)](#).

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

Now you will prepare and upload your source content to a location from which AWS CodeDeploy can deploy it. 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

Although it's not covered in this tutorial, you can use AWS CodeDeploy to deploy applications from GitHub repositories to instances. For more information, see [Integrating AWS CodeDeploy with GitHub \(p. 53\)](#).

Topics

- [Provision an Amazon S3 Bucket \(p. 80\)](#)
- [Prepare the Application's Files for the Bucket \(p. 81\)](#)
- [Bundle the Application's Files into a Single Archive File and Push the Archive File \(p. 82\)](#)

Provision an Amazon S3 Bucket

Create a storage container or *bucket* in Amazon S3—or use an existing bucket. Make sure you can upload the revision to the bucket and that Amazon EC2 instances used in deployments can download the revision from the bucket.

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

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 this tutorial. The Amazon S3 bucket must be created in the same AWS region in which your target Amazon EC2 instances are launched. 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.

Topics

- [To create an Amazon S3 bucket \(CLI\) \(p. 80\)](#)
- [To create an Amazon S3 bucket \(console\) \(p. 80\)](#)
- [Give Permissions to the Amazon S3 Bucket and Your IAM User \(p. 80\)](#)

To create an Amazon S3 bucket (CLI)

Call the `mb` command to create an Amazon S3 bucket named `codedeploydemobucket`:

```
aws s3 mb s3://codedeploydemobucket
```

To create an Amazon S3 bucket (console)

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. In the Amazon S3 console, choose **Create bucket**.
3. In the **Bucket name** box, type a name for the bucket.
4. In the **Region** list, choose the target region, and then choose **Create**.

Give Permissions to the Amazon S3 Bucket and Your IAM User

You must have permissions to upload to the Amazon S3 bucket. You can specify these permissions through an Amazon S3 bucket policy. For example, in the following Amazon S3 bucket policy, using the wildcard character (*) allows AWS account 111122223333 to upload files to any directory in the Amazon S3 bucket named `codedeploydemobucket`:

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



```
}
    ]
  }
}
```

To view your AWS account ID, see [Finding Your AWS Account ID](#).

Now is a good time to verify the Amazon S3 bucket will allow download requests from each participating Amazon EC2 instance. You can specify this through an Amazon S3 bucket policy. For example, in the following Amazon S3 bucket policy, using the wildcard character (*) allows any Amazon EC2 instance with an attached IAM instance profile containing the ARN `arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo` to download files from any directory in 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"
        ]
      }
    }
  ]
}
```

For information about how to generate and attach an Amazon S3 bucket policy, see [Bucket Policy Examples](#).

Your account must have permission to upload the revision to the Amazon S3 bucket. One way to specify this is through an IAM policy. The following IAM policy allows your IAM user to upload revisions anywhere in the Amazon S3 bucket named `codedeploydemobucket`:

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

For information about how to create and attach an IAM policy, see [Working with Policies](#).

Prepare the Application's Files for the Bucket

Make sure the web page, the AppSpec file, and the script are organized on your development machine like this:

```
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

Bundle the 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:

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

Note

If you don't switch to this folder, then the file bundling will start at your current folder. For example, if your current folder is `c:\temp` instead of `c:\temp\HelloWorldApp`, the bundling will start with files and subfolders in the `c:\temp` folder, which may include more than the `HelloWorldApp` subfolder.

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

```
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.

```
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 you uploaded to Amazon S3. You will use the AWS CLI or the AWS CodeDeploy console to deploy the revision and monitor the deployment's progress. After the application revision is successfully deployed, you will check the results.

Topics

- [Deploy Your Application Revision with AWS CodeDeploy \(p. 82\)](#)
- [Monitor and Troubleshoot Your Deployment \(p. 84\)](#)
- [Verify Your Deployment \(p. 85\)](#)

Deploy Your Application Revision with AWS CodeDeploy

Topics

- [To deploy your application revision \(CLI\) \(p. 83\)](#)
- [To deploy your application revision \(console\) \(p. 83\)](#)

To deploy your application revision (CLI)

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

You should have already followed the instructions in [Create a Service Role \(CLI\) \(p. 23\)](#) to create a service role. To get the ARN of the service role, see [Get the Service Role ARN \(CLI\) \(p. 25\)](#).

2. Now that you have the 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 service role ARN:

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

Note

The [create-deployment-group](#) command provides support for creating triggers that result in the sending of Amazon SNS notifications to topic subscribers about specified events in deployments and instances. The command also supports options for automatically rolling back deployments and setting up alarms to stop deployments when monitoring thresholds in Amazon CloudWatch alarms are met. Commands for these actions are not included in this tutorial.

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_DepGroup --s3-location  
bucket=codedeploydemobucket,bundleType=zip,key=HelloWorld_App.zip
```

To deploy your application revision (console)

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

You should have already followed the instructions in [Create a Service Role \(Console\) \(p. 21\)](#) to create a service role. To get the ARN of the service role, see [Get the Service Role ARN \(Console\) \(p. 24\)](#).

2. Now that you have the ARN, you can use 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

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

3. If the **Applications** page is not displayed, on the AWS CodeDeploy menu, choose **Applications**.
4. In the list of applications, choose **HelloWorld_App**.
5. Under **Deployment groups**, choose **Create deployment group**.
6. In **Deployment group name**, type **HelloWorld_DepGroup**.
7. In **Environment configuration**, choose the **Amazon EC2 instances** tab.
8. In the **Key** box, type **Name**.
9. In **Value**, type **CodeDeployDemo**.

Note

After you type **CodeDeployDemo**, a **1** should appear under **Matching instances** to confirm AWS CodeDeploy found one matching Amazon EC2 instance.

10. In the **Deployment configuration** drop-down list, choose **CodeDeployDefault.OneAtATime**.
11. In the **Service role ARN** drop-down list, choose the service role ARN, and then choose **Create deployment group**.
12. On the AWS CodeDeploy menu, choose **Deployments**.
13. Choose **Create deployment**.
14. In the **Application** drop-down list, choose **HelloWorld_App**.
15. In the **Deployment group** drop-down list, choose **HelloWorld_DepGroup**.
16. In **Repository type**, choose **My application is stored in Amazon S3**, and then in **Revision location**, type the location of the sample "Hello, World!" application revision 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 list of buckets, choose **codedeploydemobucket** (or the name of the bucket where you uploaded your application revision).
 3. In the list of objects, choose **HelloWorld_App.zip**.
 4. If the **Properties** pane is not displayed, choose 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 **Revision Location**, paste the **Link** field value.
17. If a message appears in the **File type** list stating the file type could not be detected, choose **.zip** in the list of file types.
18. (Optional) Type a comment in **Deployment description**.
19. From the **Deployment configuration** drop-down list, choose **CodeDeployDefault.OneAtATime**.
20. Choose **Deploy**. Information about your newly created deployment appears on the **Deployments** page.

Note

To update the deployment's current status, choose the **Refresh** button next to the table.

Monitor and Troubleshoot Your Deployment

Topics

- [To monitor and troubleshoot your deployment \(CLI\)](#) (p. 85)

- [To monitor and troubleshoot your deployment \(console\)](#) (p. 85)

To monitor and troubleshoot your deployment (CLI)

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. Call the **get-deployment** command with the deployment ID:

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

3. The command returns the deployment's overall status. If successful, the value is **Succeeded**.

If the overall status is **Failed**, you can call commands such as [list-deployment-instances](#) and [get-deployment-instance](#) to troubleshoot. For more troubleshooting options, see [Analyzing log files to investigate deployment failures on instances](#) (p. 355).

To monitor and troubleshoot your deployment (console)

On the **Deployments** page in the AWS CodeDeploy console, you can monitor your deployment's status in the **Status** column.

Note

To update the deployment's current status, choose the **Refresh** button.

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

1. In the **Deployments** table, choose the arrow next to the deployment ID. After a deployment fails, a message that describes the reason for the failure appears in **Details**.
2. In **Instances**, choose **View all instances**. More information about the deployment is displayed. After a deployment fails, you might be able to determine on which Amazon EC2 instances and at which step the deployment failed.

Note

If you don't see **Instances**, choose the **Refresh** button above the table. After the **Status** column changes from **In progress** to **Created**, **Instances** should appear.

3. If you want to do more troubleshooting, you can use a technique like [View Instance Details](#) (p. 197). You can also analyze the deployment log files on an Amazon EC2 instance. For more information, see [Analyzing log files to investigate deployment failures on instances](#) (p. 355).

Verify Your Deployment

After your deployment is successful, verify your installation is working. Use the public DNS address of the Amazon EC2 instance to view the web page in a web browser. (To get the public DNS value, in the Amazon EC2 console, choose the Amazon EC2 instance, and on the **Description** tab, look for the value in **Public DNS**.)

For example, if the public DNS address of your Amazon EC2 instance 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, on the development machine, make an update to the web page's code, and then use AWS CodeDeploy to redeploy the site. After redeployment, you should be able to see the changes on the Amazon EC2 instance.

Topics

- [Modify the Web Page \(p. 86\)](#)
- [Redeploy the Site \(p. 86\)](#)

Modify the Web Page

1. Go to your `c:\temp\HelloWorldApp` subfolder and use a text editor 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 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" "http://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="https://aws.amazon.com/
    codedeploy">AWS CodeDeploy Documentation</a>.</p>
  </div>
</body>
</html>
```

Redeploy the Site

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

Bundle and upload the changes to Amazon S3 as described in [Bundle the Application's Files into a Single Archive File and Push the Archive File \(p. 82\)](#). (As you follow those instructions, you do not need to

create a new application.) Give the revision the same key as before (**HelloWorld_App.zip**). Upload it to the same Amazon S3 bucket you created earlier (for example, **codedeploydemobucket**).

Use the AWS CLI or the AWS CodeDeploy console to redeploy the site.

Topics

- [To redeploy the site \(CLI\) \(p. 87\)](#)
- [To redeploy the site \(console\) \(p. 87\)](#)

To redeploy the site (CLI)

Call the **create-deployment** command to create a deployment based on the 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_DepGroup --s3-location
bucket=codedeploydemobucket,bundleType=zip,key=HelloWorld_App.zip
```

You can check the status of the new deployment, as described in [Monitor and Troubleshoot Your Deployment \(p. 84\)](#).

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

To redeploy the site (console)

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#).

2. On the AWS CodeDeploy menu, choose **Deployments**.
3. Choose **Create deployment**.
4. On the **Create deployment** page:
 1. In the **Application** list, choose **HelloWorld_App**.
 2. In the **Deployment group** list, choose **HelloWorld_DepGroup**.
 3. In the **Repository type** area, choose **My application is stored in Amazon S3**, and then copy the Amazon S3 link for your revision into the **Revision location** box.

To find the link value:

1. 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**, and then choose your revision, **HelloWorld_App.zip**, in the Amazon S3 console.

2. If the **Properties** pane is not visible in the Amazon S3 console, choose 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.

4. In the **File type** list, if a message appears stating that the file type could not be detected, choose **.zip**.
5. Leave the **Deployment description** box blank.
6. In the **Deployment configuration** list, choose **CodeDeployDefault.OneAtATime**, and then choose **Deploy**.

Choose the **Refresh** button above the table to get status on the deployment.

You can check the status of the deployment as described in [Monitor and Troubleshoot Your Deployment](#) (p. 84).

When AWS CodeDeploy has redeployed the site, revisit the site in your web browser to verify that the background color and text on the web page have been changed. (You may need to refresh your browser.) If the background color and text has been changed, 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 to the "Hello, World!" code and redeployed the site. To avoid ongoing charges for resources 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). You should also delete any Amazon S3 buckets that you created just for this tutorial, and the `HelloWorld_App` application in AWS CodeDeploy.

You can use the AWS CLI, the AWS CloudFormation, Amazon S3, Amazon EC2, and AWS CodeDeploy consoles, or the AWS APIs to clean up resources.

Topics

- [To use clean up resources \(CLI\)](#) (p. 88)
- [To clean up resources \(console\)](#) (p. 89)
- [What's Next?](#) (p. 90)

To use clean up resources (CLI)

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 will terminate all accompanying Amazon EC2 instances and delete all accompanying IAM roles originally created by the stack.

```
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**. This will delete the bucket and all objects in the bucket.

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

3. To delete the `HelloWorld_App` application from AWS CodeDeploy, call the **delete-application** command. This will delete all associated deployment group records and deployment records for the application.

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


4. If you did not use the AWS CloudFormation stack for this tutorial, call the **terminate-instances** command to terminate Amazon EC2 instances you manually created. Supply the ID of the Amazon EC2 instance to terminate.

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

To clean up resources (console)

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 search box, type the AWS CloudFormation stack name (for example, **CodeDeployDemoStack**).
3. Select the box beside the stack name.
4. In the **Actions** menu, choose **Delete Stack**. This will delete the stack, terminate all accompanying Amazon EC2 instances, and delete all accompanying IAM roles.

To terminate Amazon EC2 instances you 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, choose **Instances**.
3. In the search box, type the name of Amazon EC2 instance you want to terminate, and then press **Enter**.
4. Choose the Amazon EC2 instance.
5. Choose **Actions**, point to **Instance State**, and then choose **Terminate**. When prompted, choose **Yes, Terminate**. 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 list of buckets, browse to and choose the name of the Amazon S3 bucket (for example, **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**. In the **Actions** menu, choose **Delete**. When prompted to confirm the deletion, choose **OK**.
4. After the bucket is empty, you can delete the bucket. In the list of buckets, choose the row of the bucket (but not the bucket name). Choose **Delete bucket**, and when prompted to confirm, choose **OK**.

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
Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#).
2. On the AWS CodeDeploy menu, choose **Applications**.

3. On the **Application details** page, in **Deployment groups**, choose the button next to the deployment group. On the **Actions** menu, choose **Delete**. When prompted, type the name of the deployment group to confirm you want to delete it, and then choose **Delete**.
4. At the bottom of the **Application details** page, choose **Delete application**.
5. When prompted, type the name of the application to confirm you want to delete it, and then choose **Delete**.

What's Next?

If you've arrived here, you have successfully completed a deployment with AWS CodeDeploy. Congratulations!

Tutorial: Deploy an Application to an On-Premises Instance with AWS CodeDeploy (Windows Server, Ubuntu Server, or Red Hat Enterprise Linux)

This tutorial helps you gain experience 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, Ubuntu Server, or Red Hat Enterprise Linux (RHEL). For information about on-premises instances and how they work with AWS CodeDeploy, see [Working with On-Premises Instances](#) (p. 171).

Not what you're looking for?

- To practice deploying to an Amazon EC2 instance running Amazon Linux or RHEL, see [Tutorial: Deploy WordPress to an Amazon EC2 Instance \(Amazon Linux or Red Hat Enterprise Linux and Linux, macOS, or Unix\)](#) (p. 57).
- To practice deploying to an Amazon EC2 instance running Windows Server, see [Tutorial: Deploy a "Hello, World!" Application with AWS CodeDeploy \(Windows Server\)](#) (p. 75).

This tutorial builds on concepts introduced in [Try a Sample In-Place Deployment in AWS CodeDeploy](#) (p. 35). If you have not yet completed the Sample deployment wizard for an in-place deployment, you may want to do that first.

Topics

- [Prerequisites](#) (p. 90)
- [Step 1: Configure the On-Premises Instance](#) (p. 91)
- [Step 2: Create a Sample Application Revision](#) (p. 91)
- [Step 3: Bundle and Upload Your Application Revision to Amazon S3](#) (p. 94)
- [Step 4: Deploy Your Application Revision](#) (p. 94)
- [Step 5: Verify Your Deployment](#) (p. 95)
- [Step 6: Clean Up Resources](#) (p. 95)

Prerequisites

Before you start this tutorial, you must complete the prerequisites in [Getting Started with AWS CodeDeploy](#) (p. 19), which include configuring your IAM user, installing or upgrading the AWS CLI,

and creating a service role. You do not have to create an IAM instance profile as described in the prerequisites. On-premises instances do not use IAM instance profiles.

The physical device you will configure as an on-premises instance must be running one of the operating systems listed in [Operating Systems Supported by the AWS CodeDeploy Agent](#) (p. 125).

Step 1: Configure the On-Premises Instance

Before you can deploy to your on-premises instance, you must configure it. Follow the instructions in [Working with On-Premises Instances](#) (p. 171), 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.

Because it is difficult to know which software and features are already installed—or are allowed to be installed by your organization's policies—on your on-premises instance, the sample application revision we offer here simply uses batch scripts (for Windows Server) or shell scripts (for Ubuntu Server and RHEL) to write text files to a 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**. During the **BeforeInstall** deployment lifecycle event, a script will run to remove old files written during previous deployments of this sample and create a location on your on-premises instance to which to write the new files.

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 does not have permission to execute scripts.
- The user account does not have permission to create or delete folders in the locations listed in the scripts.
- The user account does not have permission to create text files in the locations listed in the scripts.

Note

If you configured a Windows Server instance and want to deploy a different sample, you may want to use the one in [Step 2: Configure Your Source Content to Deploy to the Windows Server Amazon EC2 Instance](#) (p. 77) in the [Tutorial: Deploy a "Hello, World!" Application with AWS CodeDeploy \(Windows Server\)](#) (p. 75) tutorial.

If you configured a RHEL instance and want to deploy a different sample, you may want to use the one in [Step 2: Configure Your Source Content to Be Deployed to the Amazon Linux or Red Hat Enterprise Linux Amazon EC2 Instance](#) (p. 60) in the [Tutorial: Deploy WordPress to an Amazon EC2 Instance \(Amazon Linux or Red Hat Enterprise Linux and Linux, macOS, or Unix\)](#) (p. 57) tutorial.

Currently, there is no alternative sample for Ubuntu Server.

1. On your development machine, create a subdirectory (subfolder) named `CodeDeployDemo-OnPrem` that will store the sample application revision's files, and then switch to the subfolder. For this example, we assume 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 and RHEL. If you use a different folder, be sure to substitute it for ours throughout this tutorial:

For Windows:

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

For Linux, macOS, or Unix:

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

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

`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 and RHEL:

```
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
```

For more information about AppSpec files, see [Add an Application Specification File to a Revision for AWS CodeDeploy \(p. 233\)](#) and [AWS CodeDeploy AppSpec File Reference \(p. 306\)](#).

`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 it:

For Windows:

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

For Linux, macOS, or Unix:

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

4. In the root of the `scripts` subfolder, use a text editor to create four 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 and RHEL:

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 and RHEL:

`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 and RHEL only, make sure the four shell scripts have execute permissions:

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

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

Before you can deploy your application revision, you'll need to bundle the files, and then upload the file bundle to an Amazon S3 bucket. Follow the instructions in [Create an Application with AWS CodeDeploy \(p. 209\)](#) and [Push a Revision for AWS CodeDeploy to Amazon S3 \(p. 238\)](#). (Although you can give the application and deployment group any name, we recommend you use CodeDeploy-OnPrem-App for the application name and CodeDeploy-OnPrem-DG for the deployment group name.) After you have completed those instructions, return to this page.

Note

Alternatively, you can upload the file bundle to a GitHub repository and deploy it from there. For more information, see [Integrating AWS CodeDeploy with GitHub \(p. 53\)](#).

Step 4: Deploy Your Application Revision

After you've uploaded your application revision to an Amazon S3 bucket, try deploying it to your on-premises instance. Follow the instructions in [Create a Deployment with AWS CodeDeploy \(p. 245\)](#), and then return to this page.

Step 5: Verify Your Deployment

To verify the deployment was successful, follow the instructions in [View Deployment Details with AWS CodeDeploy \(p. 254\)](#), and then return to this page.

If the deployment was successful, you'll find four text files in the `c:\temp\CodeDeployExample` folder (for Windows Server) or `/tmp/CodeDeployExample` (for Ubuntu Server and RHEL).

If the deployment failed, follow the troubleshooting steps in [View Instance Details \(p. 197\)](#) and [Troubleshoot Instance Issues \(p. 354\)](#). Make any required fixes, rebundle and upload your application revision, and then try the deployment again.

Step 6: Clean Up Resources

To avoid ongoing charges for resources you created for this tutorial, delete the Amazon S3 bucket 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.

You can use the AWS CLI or a combination of the AWS CodeDeploy and Amazon S3 consoles and the AWS CLI to clean up resources.

Clean Up Resources (CLI)

To delete the Amazon S3 bucket

- Call the `rm` command along with the `--recursive` switch against the bucket (for example, `codedeploydemobucket`). The bucket and all objects in the bucket will be deleted.

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

To delete the application and deployment group records in AWS CodeDeploy

- Call the `delete-application` command against the application (for example, `CodeDeploy-OnPrem-App`). The records for the deployment and deployment group will be deleted.

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

To deregister the on-premises instance and delete the IAM user

- Call the `deregister` command against the on-premises instance and 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 associated with this on-premises instance, use the `--no-delete-iam-user` option instead.

To uninstall the AWS CodeDeploy agent and remove the 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 used for this tutorial.

Clean Up Resources (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. Choose the icon next to the bucket you want to delete (for example, `codedeploydemobucket`), but do not choose the bucket itself.
3. Choose **Actions**, and then choose **Delete**.
4. When prompted to delete the bucket, choose **OK**.

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

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. If the list of applications does not appear, on the AWS CodeDeploy menu, choose **Applications**.
3. Choose the name of the application you want to delete (for example, `CodeDeploy-OnPrem-App`).
4. At the bottom of the **Application details** page, choose **Delete application**.
5. When prompted, type the name of the application to confirm you want to delete it, and then choose **Delete**.

You cannot use the AWS CodeDeploy console to deregister the on-premises instance or uninstall the AWS CodeDeploy agent. Follow the instructions in [To deregister the on-premises instance and delete the IAM user](#) (p. 95).

Tutorial: Use AWS CodeDeploy to Deploy an Application to an Auto Scaling Group

In this tutorial, you'll use AWS CodeDeploy to deploy an application revision to an Auto Scaling group. For information about Auto Scaling integration with AWS CodeDeploy, see [Integrating AWS CodeDeploy with Auto Scaling](#) (p. 44).

Topics

- [Prerequisites](#) (p. 97)
- [Step 1: Create and Configure the Auto Scaling Group](#) (p. 97)
- [Step 2: Deploy the Application to the Auto Scaling Group](#) (p. 102)
- [Step 3: Check Your Results](#) (p. 107)
- [Step 4: Increase the Number of Amazon EC2 Instances in the Auto Scaling Group](#) (p. 108)
- [Step 5: Check Your Results Again](#) (p. 109)
- [Step 6: Clean Up](#) (p. 111)

Prerequisites

For this tutorial, we assume you have already completed all of the steps in [Getting Started with AWS CodeDeploy \(p. 19\)](#), including setting up and configuring the AWS CLI and creating an IAM instance profile (**CodeDeployDemo-EC2-Instance-Profile**) and a service role (**CodeDeployDemo**). A *service role* is a special type of IAM role that gives a service permission to act on your behalf.

If you want to deploy an application revision to an Auto Scaling group of Ubuntu Server Amazon EC2 instances, you can create and use the sample revision in [Step 2: Create a Sample Application Revision \(p. 91\)](#) in the [Tutorial: Deploy an Application to an On-Premises Instance with AWS CodeDeploy \(Windows Server, Ubuntu Server, or Red Hat Enterprise Linux\) \(p. 90\)](#) tutorial. Otherwise, you need to create and use a revision that is compatible with an Ubuntu Server instance and AWS CodeDeploy. We also provide sample revisions for Amazon Linux, Windows Server, and Red Hat Enterprise Linux (RHEL) Amazon EC2 instances. To create a revision on your own, see [Working with Application Revisions for AWS CodeDeploy \(p. 232\)](#).

Step 1: Create and Configure the Auto Scaling Group

In this step, you'll create an Auto Scaling group that contains a single Amazon Linux, RHEL, or Windows Server Amazon EC2 instance. In a later step, you will instruct Auto Scaling to add one more Amazon EC2 instance, and AWS CodeDeploy will deploy your revision to it.

Topics

- [To create and configure the Auto Scaling group \(CLI\) \(p. 97\)](#)
- [To create and configure the Auto Scaling group \(console\) \(p. 100\)](#)

To create and configure the Auto Scaling group (CLI)

1. Call the **create-launch-configuration** command to create an Auto Scaling launch configuration.

Before you call this command, you need the ID of an AMI that works for this tutorial, represented by the placeholder *image-id*. You also need the name of an Amazon EC2 instance key pair to enable access to the Amazon EC2 instance, represented by the placeholder *key-name*. Finally, you need instructions to install the latest version of the AWS CodeDeploy agent.

AWS CodeDeployTo get the ID of an AMI that works with this tutorial:

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **Instances**, choose **Instances**, and then choose **Launch Instance**.
3. On the **Quick Start** tab of the **Choose an Amazon Machine Image** page, note the ID of the AMI next to **Amazon Linux AMI**, **Red Hat Enterprise Linux 7.1**, **Ubuntu Server 14.04 LTS**, or **Microsoft Windows Server 2012 R2**.

Note

If you have a custom version of an AMI that is compatible with AWS CodeDeploy, choose it here instead of browsing through the **Quick Start** tab. For information about using a custom AMI with AWS CodeDeploy and Auto Scaling, see [Using a Custom AMI with AWS CodeDeploy and Auto Scaling \(p. 46\)](#).

For the Amazon EC2 instance key pair, use the name of your Amazon EC2 instance key pair.

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, Ubuntu Server or RHEL 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 compatible with AWS CodeDeploy, you don't need to create the `instance-setup.sh` or `instance-setup.txt` file.

On Amazon Linux and RHEL Amazon EC2 instances

```
#!/bin/bash
yum -y update
yum install -y ruby
cd /home/ec2-user
curl -O https://bucket-name.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

bucket-name is the name of the Amazon S3 `sds-s3-latest-bucket-name` bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace *bucket-name* with `aws-codedeploy-us-east-2`. For a list of bucket names, see [Resource Kit Bucket Names by Region \(p. 335\)](#).

On Ubuntu Server Amazon EC2 instances

```
#!/bin/bash
apt-get -y update
apt-get -y install ruby
apt-get -y install wget
cd /home/ubuntu
wget https://bucket-name.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

bucket-name is the name of the Amazon S3 `sds-s3-latest-bucket-name` bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace *bucket-name* with `aws-codedeploy-us-east-2`. For a list of bucket names, see [Resource Kit Bucket Names by Region \(p. 335\)](#).

On Windows Server Amazon EC2 instances

```
<powershell>
New-Item -Path c:\temp -ItemType "directory" -Force
powershell.exe -Command 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>
```

bucket-name is the name of the Amazon S3 `sds-s3-latest-bucket-name` bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace *bucket-name* with `aws-codedeploy-us-east-2`. For a list of bucket names, see [Resource Kit Bucket Names by Region \(p. 335\)](#).

Call the **create-launch-configuration** command.

On local Linux, macOS, or Unix machines:

Important

Be sure to include `file://` before the file name. It is required in this command.

```
aws autoscaling create-launch-configuration \
  --launch-configuration-name CodeDeployDemo-AS-Configuration \
```

```
--image-id image-id \  
--key-name key-name \  
--iam-instance-profile CodeDeployDemo-EC2-Instance-Profile \  
--instance-type t1.micro \  
--user-data file://path/to/instance-setup.sh
```

On local Windows machines:

Important

Be sure to include `file://` before the file name. It is required in this command.

```
aws autoscaling create-launch-configuration --launch-configuration-name CodeDeployDemo-  
AS-Configuration --image-id image-id --key-name key-name --iam-instance-profile  
CodeDeployDemo-EC2-Instance-Profile --instance-type t1.micro --user-data file://path/  
to/instance-setup.txt
```

Note

If you have a custom version of an AMI that is compatible with AWS CodeDeploy, omit the `--user-data` option in the preceding command.

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 command to install the latest version of the AWS CodeDeploy agent. This launch configuration is based on the t1.micro Amazon EC2 instance type.

2. Call the **create-auto-scaling-group** command to create an Auto Scaling group. You will need the name of one of the Availability Zones in one of the regions listed in [Region and Endpoints](#) in the *AWS General Reference*, represented by the placeholder *availability-zone*.

Note

To view a list of Availability Zones in a region, call:

```
aws ec2 describe-availability-zones --region region-name
```

For example, to view a list of Availability Zones in the US West (Oregon) Region, call:

```
aws ec2 describe-availability-zones --region us-west-2
```

For a list of region name identifiers, see [Resource Kit Bucket Names by Region](#) (p. 335).

On local Linux, macOS, or Unix machines:

```
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 availability-zone
```

On local Windows machines:

```
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 availability-zone
```

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 one Amazon EC2 instance, and it is created in the specified Availability Zone.

3. Call 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[*].[HealthStatus, LifecycleState]" --output text
```

Do not proceed until the returned values show **Healthy** and **InService**.

To create and configure the Auto Scaling group (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the global navigation bar, make sure one of the regions listed in [Region and Endpoints](#) in the *AWS General Reference* is selected. Auto Scaling resources are tied to the region you specify, and AWS CodeDeploy is supported in select regions only.
3. In the navigation bar, under **Auto Scaling**, choose **Launch Configurations**.
4. Choose **Create launch configuration**.
5. On the **Quick Start** tab of the **Choose AMI** page, next to **Amazon Linux AMI**, **Red Hat Enterprise Linux 7.2**, **Ubuntu Server 14.04 LTS**, or **Microsoft Windows Server 2012 R2 Base**, choose **Select**.

Note

If you have a custom version of an AMI that already has the AWS CodeDeploy agent installed, choose it here instead. For information about using a custom AMI with AWS CodeDeploy and Auto Scaling, see [Using a Custom AMI with AWS CodeDeploy and Auto Scaling](#) (p. 46).

6. On the **Choose Instance Type** page, leave the defaults, and choose **Next: Configure details**.
7. On the **Configure details** page, in **Name**, type **CodeDeployDemo-AS-Configuration**. In **IAM role**, choose the IAM instance profile you created earlier (**CodeDeployDemo-EC2-Instance-Profile**).

Expand **Advanced Details**, and in **User data**, type the following.

Note

If you are using a custom version of an AMI that already has the AWS CodeDeploy agent installed, skip this step.

For Amazon Linux and RHEL Amazon EC2 instances

```
#!/bin/bash
yum -y update
yum install -y ruby
cd /home/ec2-user
curl -O https://bucket-name.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

bucket-name is the name of the Amazon S3 sds-s3-latest-bucket-name bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace *bucket-name* with `aws-codedeploy-us-east-2`. For a list of bucket names, see [Resource Kit Bucket Names by Region](#) (p. 335).

For Ubuntu Server Amazon EC2 instances

```
#!/bin/bash
apt-get -y update
apt-get -y install ruby
apt-get -y install wget
cd /home/ubuntu
wget https://bucket-name.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

bucket-name is the name of the Amazon S3 sds-s3-latest-bucket-name bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace *bucket-name* with aws-codedeploy-us-east-2. For a list of bucket names, see [Resource Kit Bucket Names by Region \(p. 335\)](#).

For Windows Server Amazon EC2 instances

```
<powershell>
New-Item -Path c:\temp -ItemType "directory" -Force
powershell.exe -Command 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>
```

bucket-name is the name of the Amazon S3 sds-s3-latest-bucket-name bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace *bucket-name* with aws-codedeploy-us-east-2. For a list of bucket names, see [Resource Kit Bucket Names by Region \(p. 335\)](#).

Leave the rest of the defaults, and choose **Skip to review**.

8. On the **Review** page, choose **Create launch configuration**.

Note

In a production environment, we recommend that you restrict 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**. In the **Select a key pair** drop-down list, choose the Amazon EC2 instance key pair you created or used in previous steps. Select **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**, and then choose **Create launch configuration**.
10. Choose **Create an Auto Scaling group using this launch configuration**.
11. On the **Configure Auto Scaling group details** page, in **Group name**, type **CodeDeployDemo-AS-Group**. In **Group size**, leave the default. In the **Availability Zone(s)** box, choose an Availability Zone in one of the regions listed in [Region and Endpoints](#) in the *AWS General Reference*. Leave the rest of the defaults, and choose **Next: Configure scaling policies**.

Note

If **Launch into EC2-Classical** does not appear in the **Network** list, and you are not able to select a default virtual private cloud (VPC), choose or create a VPC and subnet. 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, and choose **Next: Configure Notifications**.
13. Skip the step for configuring notifications, and choose **Review**.
14. Choose **Create Auto Scaling group**, and then choose **Close**.
15. In the navigation bar, with **Auto Scaling Groups** selected, choose **CodeDeployDemo-AS-Group**, and then choose the **Instances** tab. Do not proceed until the value of **InService** appears in the **Lifecycle** column and the value of **Healthy** appears in the **Health Status** column.

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 create the deployment \(CLI\) \(p. 102\)](#)
- [To create the deployment \(console\) \(p. 104\)](#)

To create the deployment (CLI)

1. Call the **create-application** command to create an application named **SimpleDemoApp**:

```
aws deploy create-application --application-name SimpleDemoApp
```

2. You should have already created a service role by following the instructions in [Step 3: Create a Service Role for AWS CodeDeploy \(p. 21\)](#). The service role will give AWS CodeDeploy permission to access your Amazon EC2 instances to expand (read) their tags. You will need the service role ARN. To get the service role ARN, follow the instructions in [Get the Service Role ARN \(CLI\) \(p. 25\)](#).
3. Now that you have a service role ARN, call the **create-deployment-group** command to create a 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.

Note

The [create-deployment-group](#) command provides support for creating triggers that result in the sending of Amazon SNS notifications to topic subscribers about specified events in deployments and instances. The command also supports options for automatically rolling back deployments and setting up alarms to stop deployments when monitoring thresholds in Amazon CloudWatch alarms are met. Commands for these actions are not included in this tutorial.

On local Linux, macOS, or Unix machines:

```
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 service-role-arn
```

On local Windows machines:

```
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 service-role-arn
```

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 and RHEL Amazon EC2 instances, calling from local Linux, macOS, or Unix machines

```
aws deploy create-deployment \  
  --application-name SimpleDemoApp \  
  --deployment-config-name CodeDeployDefault.OneAtATime \  
  --deployment-group-name SimpleDemoDG \  
  --s3-location bucket=bucket-name,bundleType=zip,key=samples/latest/  
SampleApp_Linux.zip
```

bucket-name is the name of the Amazon S3 sds-s3-latest-bucket-name bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace *bucket-name* with `aws-codedeploy-us-east-2`. For a list of bucket names, see [Resource Kit Bucket Names by Region](#) (p. 335).

For Amazon Linux and RHEL Amazon EC2 instances, calling from local Windows machines

```
aws deploy create-deployment --application-name SimpleDemoApp --deployment-config-  
name CodeDeployDefault.OneAtATime --deployment-group-name SimpleDemoDG --s3-location  
bucket=bucket-name,bundleType=zip,key=samples/latest/SampleApp_Linux.zip
```

bucket-name is the name of the Amazon S3 sds-s3-latest-bucket-name bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace *bucket-name* with `aws-codedeploy-us-east-2`. For a list of bucket names, see [Resource Kit Bucket Names by Region](#) (p. 335).

For Windows Server Amazon EC2 instances, calling from local Linux, macOS, or Unix machines

```
aws deploy create-deployment \  
  --application-name SimpleDemoApp \  
  --deployment-config-name CodeDeployDefault.OneAtATime \  
  --deployment-group-name SimpleDemoDG \  
  --s3-location bucket=bucket-name,bundleType=zip,key=samples/latest/  
SampleApp_Windows.zip
```

bucket-name is the name of the Amazon S3 sds-s3-latest-bucket-name bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace *bucket-name* with `aws-codedeploy-us-east-2`. For a list of bucket names, see [Resource Kit Bucket Names by Region](#) (p. 335).

For Windows Server Amazon EC2 instances, calling from local Windows machines

```
aws deploy create-deployment --application-name SimpleDemoApp --deployment-config-  
name CodeDeployDefault.OneAtATime --deployment-group-name SimpleDemoDG --s3-location  
bucket=bucket-name,bundleType=zip,key=samples/latest/SampleApp_Windows.zip
```

bucket-name is the name of the Amazon S3 sds-s3-latest-bucket-name bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace *bucket-name* with `aws-codedeploy-us-east-2`. For a list of bucket names, see [Resource Kit Bucket Names by Region](#) (p. 335).

Note

Currently, AWS CodeDeploy does not provide a sample revision to deploy to Ubuntu Server Amazon EC2 instances. To create a revision on your own, see [Working with Application Revisions for AWS CodeDeploy](#) (p. 232).

5. Call the **get-deployment** command to make sure the deployment was successful.

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:

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

Do not continue until the returned value is Succeeded.

To create the deployment (console)

1. You should have already created a service role by following the instructions in [Step 3: Create a Service Role for AWS CodeDeploy \(p. 21\)](#). The service role will give AWS CodeDeploy permission to access your instances to expand (read) their tags. Before you use the AWS CodeDeploy console to deploy your application revision, you will need the service role ARN. To get the service role ARN, follow the instructions in [Get the Service Role ARN \(Console\) \(p. 24\)](#).
2. Now that you have the service role ARN, you can use 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

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#).

3. If the **Applications** page does not appear, on the AWS CodeDeploy menu, choose **Applications**.
4. Choose **Create application**.
5. In the **Application name** box, type **SimpleDemoApp**.
6. In the **Deployment group name** box, type **SimpleDemoDG**.
7. In **Environment configuration**, on the **Auto Scaling groups** tab, type **CodeDeployDemo-AS-Group**.
8. From the **Deployment configuration** drop-down list, choose **CodeDeployDefault.OneAtATime**.
9. From the **Service role ARN** drop-down list, choose the service role ARN.
10. Choose **Create application**.
11. In the **Application details** page, in the **Deployment groups** area, next to **SimpleDemoDG**, choose the arrow to see the deployment group details.
12. Select the button next to **SimpleDemoDG**. In the **Actions** menu, choose **Deploy new revision**.
13. In the **Repository type** area, choose **My application is stored in Amazon S3**, and then in the **Revision location** box, type the location of the sample application for your operating system and region.

For Amazon Linux and RHEL Amazon EC2 instances

Region	Location of sample application
US East (Ohio) Region	<code>http://s3-us-east-2.amazonaws.com/aws-codedeploy-us-east-2/samples/latest/SampleApp_Linux.zip</code>

Region	Location of sample application
US East (N. Virginia) Region	http://s3.amazonaws.com/aws-codedeploy-us-east-1/samples/latest/SampleApp_Linux.zip
US West (N. California) Region	http://s3-us-west-1.amazonaws.com/aws-codedeploy-us-west-1/samples/latest/SampleApp_Linux.zip
US West (Oregon) Region	http://s3-us-west-2.amazonaws.com/aws-codedeploy-us-west-2/samples/latest/SampleApp_Linux.zip
Canada (Central) Region	http://s3-ca-central-1.amazonaws.com/aws-codedeploy-ca-central-1/samples/latest/SampleApp_Linux.zip
EU (Ireland) Region	http://s3-eu-west-1.amazonaws.com/aws-codedeploy-eu-west-1/samples/latest/SampleApp_Linux.zip
EU (London) Region	http://s3-eu-west-2.amazonaws.com/aws-codedeploy-eu-west-2/samples/latest/SampleApp_Linux.zip
EU (Paris) Region	http://s3-eu-west-3.amazonaws.com/aws-codedeploy-eu-west-3/samples/latest/SampleApp_Linux.zip
EU (Frankfurt) Region	http://s3-eu-central-1.amazonaws.com/aws-codedeploy-eu-central-1/samples/latest/SampleApp_Linux.zip
Asia Pacific (Tokyo) Region	http://s3-ap-northeast-1.amazonaws.com/aws-codedeploy-ap-northeast-1/samples/latest/SampleApp_Linux.zip
Asia Pacific (Seoul) Region	http://s3-ap-northeast-2.amazonaws.com/aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Linux.zip
Asia Pacific (Singapore) Region	http://s3-ap-southeast-1.amazonaws.com/aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Linux.zip
Asia Pacific (Sydney) Region	http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Linux.zip
Asia Pacific (Mumbai) Region	http://s3-ap-south-1.amazonaws.com/aws-codedeploy-ap-south-1/samples/latest/SampleApp_Linux.zip

Region	Location of sample application
South America (São Paulo) Region	http://s3-sa-east-1.amazonaws.com/ aws-codedeploy-sa-east-1/samples/ latest/SampleApp_Linux.zip

For Windows Server Amazon EC2 instances

Region	Location of sample application
US East (Ohio) Region	http://s3-us-east-2.amazonaws.com/ aws-codedeploy-us-east-2/samples/ latest/SampleApp_Windows.zip
US East (N. Virginia) Region	http://s3.amazonaws.com/aws- codedeploy-us-east-1/samples/latest/ SampleApp_Windows.zip
US West (N. California) Region	http://s3-us-west-1.amazonaws.com/ aws-codedeploy-us-west-1/samples/ latest/SampleApp_Windows.zip
US West (Oregon) Region	http://s3-us-west-2.amazonaws.com/ aws-codedeploy-us-west-2/samples/ latest/SampleApp_Windows.zip
Canada (Central) Region	http://s3-ca- central-1.amazonaws.com/aws- codedeploy-ca-central-1/samples/ latest/SampleApp_Windows.zip
EU (Ireland) Region	http://s3-eu-west-1.amazonaws.com/ aws-codedeploy-eu-west-1/samples/ latest/SampleApp_Windows.zip
EU (London) Region	http://s3-eu-west-2.amazonaws.com/ aws-codedeploy-eu-west-2/samples/ latest/SampleApp_Windows.zip
EU (Paris) Region	http://s3-eu-west-3.amazonaws.com/ aws-codedeploy-eu-west-3/samples/ latest/SampleApp_Windows.zip
EU (Frankfurt) Region	http://s3-eu- central-1.amazonaws.com/aws- codedeploy-eu-central-1/samples/ latest/SampleApp_Windows.zip
Asia Pacific (Tokyo) Region	http://s3-ap- northeast-1.amazonaws.com/aws- codedeploy-ap-northeast-1/samples/ latest/SampleApp_Windows.zip
Asia Pacific (Seoul) Region	http://s3-ap- northeast-2.amazonaws.com/aws- codedeploy-ap-northeast-2/samples/ latest/SampleApp_Windows.zip

Region	Location of sample application
Asia Pacific (Singapore) Region	<code>http://s3-ap-southeast-1.amazonaws.com/aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Windows.zip</code>
Asia Pacific (Sydney) Region	<code>http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Windows.zip</code>
Asia Pacific (Mumbai) Region	<code>http://s3-ap-south-1.amazonaws.com/aws-codedeploy-ap-south-1/samples/latest/SampleApp_Windows.zip</code>
South America (São Paulo) Region	<code>http://s3-sa-east-1.amazonaws.com/aws-codedeploy-sa-east-1/samples/latest/SampleApp_Windows.zip</code>

For Ubuntu Server Amazon EC2 instances

Type the location of your custom application revision stored in Amazon S3.

14. Leave the **Deployment description** box blank.
15. With **CodeDeployDefault.OneAtATime** selected in the **Deployment configuration** drop-down list, choose **Deploy**.

Note

To update the deployment's current status, refresh the page in your browser.

If **Failed** appears instead of **Succeeded**, you may want to try some of the techniques in [Monitor and Troubleshoot Your Deployment \(p. 69\)](#) (using the application name of **SimpleDemoApp** and the deployment group name of **SimpleDemoDG**).

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.

Topics

- [To check the results \(CLI\) \(p. 107\)](#)
- [To check the results \(console\) \(p. 108\)](#)

To check the results (CLI)

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. To get the ID, call the **describe-auto-scaling-groups** against **CodeDeployDemo-AS-Group** as you did before:

```
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 instance-id --query  
"Reservations[0].Instances[0].PublicDnsName" --output text
```

The returned value is the public DNS of the Amazon EC2 instance.

Using a web browser, show the SimpleDemoApp revision deployed to that Amazon EC2 instance, using a URL like the following:

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

If you see the congratulations page, 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 deploy your revision to the new instance.

To check the results (console)

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**, choose **Auto Scaling Groups**, and then choose the **CodeDeployDemo-AS-Group** entry.

On the **Instances** tab, choose the Amazon EC2 instance ID in the list.

On the **Instances** page, on the **Description** tab, note the **Public DNS** value. It should look something like this: **ec2-01-234-567-890.compute-1.amazonaws.com**.

Using a web browser, show the SimpleDemoApp revision deployed to that Amazon EC2 instance, using a URL like the following:

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

If you see the congratulations page, 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 deploy your revision to the new Amazon EC2 instance.

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 instance, AWS CodeDeploy will deploy your revision to it.

Topics

- [To scale up the number of Amazon EC2 instances in the Auto Scaling group \(CLI\) \(p. 109\)](#)
- [To scale up the number of Amazon EC2 instances in the deployment group \(console\) \(p. 109\)](#)

To scale up the number of Amazon EC2 instances in the Auto Scaling group (CLI)

1. Call the **update-auto-scaling-group** command to increase the Amazon EC2 instances in the Auto Scaling group named **CodeDeployDemo-AS-Group** from one to two.

On local Linux, macOS, or Unix machines:

```
aws autoscaling update-auto-scaling-group \
  --auto-scaling-group-name CodeDeployDemo-AS-Group \
  --min-size 2 \
  --max-size 2 \
  --desired-capacity 2
```

On local Windows machines:

```
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 the Auto Scaling group now has two Amazon EC2 instances. Call 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[*].[HealthStatus, LifecycleState]" --output text
```

Do not proceed until both of the returned values show **Healthy** and **InService**.

To scale up the number of Amazon EC2 instances in the deployment group (console)

1. In the Amazon EC2 navigation bar, under **Auto Scaling**, choose **Auto Scaling Groups**, and then choose **CodeDeployDemo-AS-Group**.
2. Choose **Actions**, and then choose **Edit**.
3. On the **Details** tab, in the **Desired**, **Min**, and **Max** boxes, type **2**, and then choose **Save**.
4. Choose the **Instances** tab. The new Amazon EC2 instance should appear in the list. (If the instance does not appear, you may need to choose the **Refresh** button a few times.) Do not proceed until the value of **InService** appears in the **Lifecycle** column and the value of **Healthy** appears in the **Health Status** column.

Step 5: Check Your Results Again

In this step, you'll check to see if AWS CodeDeploy installed the SimpleDemoApp revision on the new instance in the Auto Scaling group.

Topics

- [To check automatic deployment results \(CLI\) \(p. 110\)](#)
- [To check automatic deployment results \(console\) \(p. 110\)](#)

To check automatic deployment results (CLI)

1. Before you call the **get-deployment** command, you will need the ID of the automatic deployment. 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 two deployment IDs. Use the one you have not yet used in a call to the **get-deployment** command:

```
aws deploy get-deployment --deployment-id deployment-id --query "deploymentInfo.[status, creator]" --output text
```

In addition to the deployment status, you should see `autoScaling` in the command output. (`autoScaling` means Auto Scaling created the deployment.)

Do not proceed until the deployment status shows `Succeeded`.

2. Before you call the **describe-instances** command, you will need the ID of the new Amazon EC2 instance. 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 instance-id --query "Reservations[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. Using a web browser, show the **SimpleDemoApp** revision deployed to that Amazon EC2 instance, using a URL like the following:

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

If the congratulations page appears, you've used AWS CodeDeploy to deploy a revision to a scaled-up Amazon EC2 instance in an Auto Scaling group!

To check automatic deployment results (console)

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. On the AWS CodeDeploy menu, choose **Deployments**. The **Deployments** page displays information about the deployment Auto Scaling created. Normally, you would create a deployment on your own, but Auto Scaling created one on your behalf to deploy your revision to the new Amazon EC2 instance.

Note

To update the deployment's current status, use refresh the page in your browser.

3. After **Succeeded** is displayed for the deployment status, verify the results on the instance. You will first need to get the public DNS of the instance:
4. In the Amazon EC2 navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**, and then choose the **CodeDeployDemo-AS-Group** entry.
5. On the **Instances** tab, choose the ID of the new Amazon EC2 instance.
6. On the **Instances** page, on the **Description** tab, note the **Public DNS** value. It should look something like this: **ec2-01-234-567-890.compute-1.amazonaws.com**.

Show the SimpleDemoApp revision deployed to the instance using a URL like the following:

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

If the congratulations page appears, you've used AWS CodeDeploy to deploy a revision to a scaled-up Amazon EC2 instance in an Auto Scaling group!

Step 6: Clean Up

In this step, you'll delete the Auto Scaling group to avoid ongoing charges for resources you used during this tutorial. Optionally, you can delete the Auto Scaling configuration and AWS CodeDeploy deployment component records.

Topics

- [To clean up resources \(CLI\) \(p. 111\)](#)
- [To clean up resources \(console\) \(p. 111\)](#)

To clean up resources (CLI)

1. Delete the Auto Scaling group by calling the **delete-auto-scaling-group** command against **CodeDeployDemo-AS-Group**. This will also terminate the Amazon EC2 instances.

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name CodeDeployDemo-AS-Group --force-delete
```

2. Optionally, delete the 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 by calling the **delete-application** command against the application named **SimpleDemoApp**. This will also delete all associated deployment, deployment group, and revision records.

```
aws deploy delete-application --application-name SimpleDemoApp
```

To clean up resources (console)

1. Delete the Auto Scaling group. This will also terminate the 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**, choose **Auto Scaling Groups**, and then choose the **CodeDeployDemo-AS-Group** entry.
3. Choose **Actions**, choose **Delete**, and then choose **Yes, Delete**.
4. Optionally, delete the launch configuration. In the navigation bar, under **Auto Scaling**, choose **Launch Configurations**, and then choose **CodeDeployDemo-AS-Configuration**.
5. Choose **Actions**, choose **Delete launch configuration**, and then choose **Yes, Delete**.
6. Optionally, delete the application from AWS CodeDeploy. This will also delete all associated deployment, deployment group, and revision records. Open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.
7. On the AWS CodeDeploy menu, choose **Applications**.
8. In the list of applications, choose **SimpleDemoApp**.
9. On the **Application details** page, choose **Delete application**.
10. When prompted, type the name of the application to confirm you want to delete it, and then choose **Delete**.

Tutorial: Use AWS CodeDeploy to Deploy an Application from GitHub

In this tutorial, you use AWS CodeDeploy to deploy a sample application revision from GitHub to a single Amazon EC2 instance running Amazon Linux, a single Red Hat Enterprise Linux (RHEL) instance, or a single Windows Server instance. For information about GitHub integration with AWS CodeDeploy, see [Integrating AWS CodeDeploy with GitHub](#) (p. 53).

Note

You can also use AWS CodeDeploy to deploy an application revision from GitHub to an Ubuntu Server instance. You can use the sample revision described in [Step 2: Create a Sample Application Revision](#) (p. 91) in [Tutorial: Deploy an Application to an On-Premises Instance with AWS CodeDeploy \(Windows Server, Ubuntu Server, or Red Hat Enterprise Linux\)](#) (p. 90), or you can create a revision compatible with an Ubuntu Server instance and AWS CodeDeploy. To create your own revision, see [Plan a Revision for AWS CodeDeploy](#) (p. 232) and [Add an Application Specification File to a Revision for AWS CodeDeploy](#) (p. 233).

Topics

- [Prerequisites](#) (p. 112)
- [Step 1: Set Up a GitHub Account](#) (p. 113)
- [Step 2: Create a GitHub Repository](#) (p. 113)
- [Step 3: Upload a Sample Application to Your GitHub Repository](#) (p. 115)
- [Step 4: Provision an Instance](#) (p. 117)
- [Step 5: Create an Application and Deployment Group](#) (p. 118)
- [Step 6: Deploy the Application to the Instance](#) (p. 119)
- [Step 7: Monitor and Verify the Deployment](#) (p. 122)
- [Step 8: Clean Up](#) (p. 123)

Prerequisites

Before you start this tutorial, do the following:

- Install Git on your local machine. To install Git, see [Git Downloads](#).
- Complete the steps in [Getting Started with AWS CodeDeploy \(p. 19\)](#), 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

You will need a GitHub account to create a GitHub repository where the revision will be stored. If you already have a GitHub account, skip ahead to [Step 2: Create a GitHub Repository \(p. 113\)](#).

1. Go to <https://github.com/join>.
2. Type a user name, your email address, and a password.
3. Choose **Sign up for GitHub**, and then follow the instructions.

Step 2: Create a GitHub Repository

You will need a GitHub repository to store the revision.

If you already have a GitHub repository, be sure to substitute its name for **CodeDeployGitHubDemo** throughout this tutorial, and then skip ahead to [Step 3: Upload a Sample Application to Your GitHub Repository \(p. 115\)](#).

1. On the [GitHub home page](#), do one of the following:
 - In **Your repositories**, choose **New repository**.
 - On the navigation bar, choose **Create new (+)**, and then choose **New repository**.
2. In the **Create a new repository** page, do the following:
 - In the **Repository name** box, type **CodeDeployGitHubDemo**.
 - Select **Public**.

Note
Selecting the default **Public** option means that anyone can see this repository. Although you can select the **Private** option to limit who can see and commit to the repository, this option may result in additional charges from GitHub.

 - Clear the **Initialize this repository with a README** check box. You will create a `README.md` file manually in the next step instead.
 - Choose **Create repository**.
3. Follow the instructions for your local machine type to use the command line to create the repository.

Note

If you have enabled two-factor authentication on GitHub, make sure you enter your personal access token instead of your GitHub login password if prompted for a password. For information, see [Providing Your 2FA Authentication Code](#).

On local Linux, macOS, or Unix machines:

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/CodeDeployGitHubDemo.git
```

```
git push -u origin master
```

2. Leave the terminal open in the /tmp/CodeDeployGitHubDemo location.

On local Windows machines:

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. 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/CodeDeployGitHubDemo.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 a sample revision from a public Amazon S3 bucket to your GitHub repository. (For simplicity, the sample revisions provided for this tutorial are single web pages.)

Note

If you use one of your revisions instead of our sample revision, your revision must:

- Follow the guidelines in [Plan a Revision for AWS CodeDeploy \(p. 232\)](#) and [Add an Application Specification File to a Revision for AWS CodeDeploy \(p. 233\)](#).
- Work with the corresponding instance type.
- Be accessible from your GitHub dashboard.

If your revision meets these requirements, skip ahead to [Step 5: Create an Application and Deployment Group \(p. 118\)](#).

If you're deploying to an Ubuntu Server instance, you'll need to upload to your GitHub repository a revision compatible with an Ubuntu Server instance and AWS CodeDeploy. For more information, see [Plan a Revision for AWS CodeDeploy \(p. 232\)](#) and [Add an Application Specification File to a Revision for AWS CodeDeploy \(p. 233\)](#).

Topics

- [Push a sample revision from a local Linux, macOS, or Unix machine \(p. 115\)](#)
- [Push a sample revision from a local Windows machine \(p. 116\)](#)

Push a sample revision from a local Linux, macOS, or Unix machine

With your terminal still open in, for example, the `/tmp/CodeDeployGitHubDemo` location, run the following commands one at a time:

Note

If you plan to deploy to a Windows Server instance, substitute `SampleApp_Windows.zip` for `SampleApp_Linux.zip` in the commands.

```
(Amazon S3 copy command)
```

```
unzip SampleApp_Linux.zip
```

```
rm SampleApp_Linux.zip
```

```
git add .
```

```
git commit -m "Added sample app"
```

```
git push
```

Where *(Amazon S3 copy command)* is one of the following:

- `aws s3 cp s3://aws-codedeploy-us-east-2/samples/latest/SampleApp_Linux.zip . --region us-east-2` for the US East (Ohio) region
- `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-1/samples/latest/SampleApp_Linux.zip . --region us-west-1` for the US West (N. California) 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-ca-central-1/samples/latest/SampleApp_Linux.zip . --region ca-central-1` for the Canada (Central) 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-eu-west-2/samples/latest/SampleApp_Linux.zip . --region eu-west-2` for the EU (London) region
- `aws s3 cp s3://aws-codedeploy-eu-west-3/samples/latest/SampleApp_Linux.zip . --region eu-west-3` for the EU (Paris) region
- `aws s3 cp s3://aws-codedeploy-eu-central-1/samples/latest/SampleApp_Linux.zip . --region eu-central-1` for the EU (Frankfurt) Region
- `aws s3 cp s3://aws-codedeploy-ap-northeast-1/samples/latest/SampleApp_Linux.zip . --region ap-northeast-1` for the Asia Pacific (Tokyo) region
- `aws s3 cp s3://aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Linux.zip . --region ap-northeast-2` for the Asia Pacific (Seoul) region
- `aws s3 cp s3://aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Linux.zip . --region ap-southeast-1` for the Asia Pacific (Singapore) 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
- `aws s3 cp s3://aws-codedeploy-ap-south-1/samples/latest/SampleApp_Linux.zip . --region ap-south-1` for the Asia Pacific (Mumbai) region
- `aws s3 cp s3://aws-codedeploy-sa-east-1/samples/latest/SampleApp_Linux.zip . --region sa-east-1` for the South America (São Paulo) Region

Push a sample revision from a local Windows machine

With your command prompt still open in, for example, the `c:\temp\CodeDeployGitHubDemo` location, run the following commands one at a time:

Note

If you plan to deploy to an Amazon Linux or RHEL instance, substitute `SampleApp_Linux.zip` for `SampleApp_Windows.zip` in the commands.

(Amazon S3 copy command)

Unzip the contents of the ZIP file directly into the local directory (for example `c:\temp\CodeDeployGitHubDemo`), not into a new subdirectory.

```
git add .
```

```
git commit -m "Added sample app"
```

```
git push
```

Where (*Amazon S3 copy command*) is one of the following:

- `aws s3 cp s3://aws-codedeploy-us-east-2/samples/latest/SampleApp_Windows.zip . --region us-east-2` for the US East (Ohio) region
- `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-1/samples/latest/SampleApp_Windows.zip . --region us-west-1` for the US West (N. California) 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-ca-central-1/samples/latest/SampleApp_Windows.zip . --region ca-central-1` for the Canada (Central) 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-eu-west-2/samples/latest/SampleApp_Windows.zip . --region eu-west-2` for the EU (London) region
- `aws s3 cp s3://aws-codedeploy-eu-west-3/samples/latest/SampleApp_Windows.zip . --region eu-west-3` for the EU (Paris) region
- `aws s3 cp s3://aws-codedeploy-eu-central-1/samples/latest/SampleApp_Windows.zip . --region eu-central-1` for the EU (Frankfurt) Region
- `aws s3 cp s3://aws-codedeploy-ap-northeast-1/samples/latest/SampleApp_Windows.zip . --region ap-northeast-1` for the Asia Pacific (Tokyo) region
- `aws s3 cp s3://aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Windows.zip . --region ap-northeast-2` for the Asia Pacific (Seoul) region
- `aws s3 cp s3://aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Windows.zip . --region ap-southeast-1` for the Asia Pacific (Singapore) 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
- `aws s3 cp s3://aws-codedeploy-ap-south-1/samples/latest/SampleApp_Windows.zip . --region ap-south-1` for the Asia Pacific (Mumbai) region
- `aws s3 cp s3://aws-codedeploy-sa-east-1/samples/latest/SampleApp_Windows.zip . --region sa-east-1` for the South America (São Paulo) Region

To push your own revision to an Ubuntu Server instance, copy your revision into your local repo, and then call the following:

```
git add .  
git commit -m "Added Ubuntu app"  
git push
```

Step 4: Provision an Instance

In this step, you will create or configure the instance that you will deploy the sample application to. You can deploy to an Amazon EC2 instance or an on-premises instance that is running one of the operating systems supported by AWS CodeDeploy. For information see [Operating Systems Supported by the AWS CodeDeploy Agent \(p. 125\)](#). (If you already have an instance configured for use in AWS CodeDeploy deployments, skip to the next step.)

To create or configure an instance, see [Working with Instances for AWS CodeDeploy \(p. 156\)](#), and then return to this page.

Note

To quickly create a new instance for this tutorial, we recommend using the Amazon EC2 console. See [Launch an Amazon EC2 Instance \(Console\) \(p. 157\)](#).

To verify that the AWS CodeDeploy agent is running on the instance, see [Verify the AWS CodeDeploy Agent Is Running \(p. 132\)](#).

After you have successfully launched or configured the instance and verified the AWS CodeDeploy agent is running, go to the next step.

Step 5: Create an Application and Deployment Group

In this step, you will use the AWS CodeDeploy console or the AWS CLI to create an application and deployment group to use to deploy the sample revision from your GitHub repository.

Create an application and deployment group (console)

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#).

2. On the **Applications** page, choose **Create application**.

Note

If you haven't created any applications yet and the AWS CodeDeploy start page is displayed, choose **Get Started Now**, complete a deployment using the Sample deployment wizard, and then return to this topic.

3. In the **Application name** box, type **CodeDeployGitHubDemo-App**.
4. In the **Deployment group name** box, type **CodeDeployGitHubDemo-DepGrp**.
5. In **Deployment type**, choose **In-place deployment**.
6. In **Environment configuration**, depending on the type of instance you are using, choose the **Amazon EC2 instances** tab or the **On-premises instances** tab. In the **Key** and **Value** boxes, type the instance tag key and value that was applied to your instance as part of [Step 4: Provision an Instance \(p. 117\)](#).
7. In the **Service role ARN** drop-down list, choose your service role ARN. (Follow the instructions in [Get the Service Role ARN \(Console\) \(p. 24\)](#) if you need to find your service role ARN.)
8. Choose **Create application**, and continue to the next step.

Create an application and deployment group (CLI)

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**:
 - 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 \(p. 117\)](#).

- 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. 117).
- 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. 117).
- 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. 117).
- `service-role-arn` is a service role ARN. (Follow the instructions in [Get the Service Role ARN \(CLI\)](#) (p. 25) to find the service role ARN.)

```
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
```

Note

The `create-deployment-group` command provides support for creating triggers that result in the sending of Amazon SNS notifications to topic subscribers about specified events in deployments and instances. The command also supports options for automatically rolling back deployments and setting up alarms to stop deployments when monitoring thresholds in Amazon CloudWatch alarms are met. Commands for these actions are not included in this tutorial.

Step 6: 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.

To deploy the revision (console)

1. On the **Application details** page, in **Deployment groups**, choose the button next to `CodeDeployGitHubDemo-DepGrp`.
2. In the **Actions** menu, choose **Deploy new revision**.
3. On the **Create deployment** page, in the **Repository type** area, choose **My application is stored in GitHub**.
4. In **Connect to GitHub**, do one of the following:
 - To create a connection for AWS CodeDeploy applications to a GitHub account, sign out of GitHub in a separate web browser tab. In **GitHub account**, type a name to identify this connection, and then choose **Connect to GitHub**. The web page prompts you to authorize AWS CodeDeploy to interact with GitHub for the application named `CodeDeployGitHubDemo-App`. Continue to step 5.
 - To use a connection you have already created, in **GitHub account**, select its name, and then choose **Connect to GitHub**. Continue to step 7.
 - To create a connection to a different GitHub account, sign out of GitHub in a separate web browser tab. Choose **Connect to a different GitHub account**, and then choose **Connect to GitHub**. Continue to step 5.
5. Follow the instructions on the **Sign in** page to sign in with your GitHub account.
6. On the **Authorize application** page, choose **Authorize application**.

7. On the AWS CodeDeploy **Create deployment** page, in the **Repository name** box, type the GitHub user name you used to sign in, followed by a forward slash (/), followed by the name of the repository where you pushed your application revision (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 **Your repositories**, 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.

Note

If the target repository name is not displayed in **Your repositories**, use the **Search GitHub** box to find the target repository and corresponding GitHub user or organization name.

8. In the **Commit ID** box, type the ID of the commit 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 **Your repositories**, choose **CodeDeployGitHubDemo**.
 3. In the list of commits, find and copy the commit ID 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.)
 4. Paste the commit ID into the **Commit ID** box.
9. Choose **Deploy**, and continue to the next step.

To deploy the revision (CLI)

Before you can call any AWS CLI commands that interact with GitHub (such as the **create-deployment** command, which you will call next), you must give AWS CodeDeploy permission to use your GitHub user account to interact with GitHub for the CodeDeployGitHubDemo-App application. Currently, you must use the AWS CodeDeploy console to do this.

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. On the AWS CodeDeploy menu, choose **Deployments**.
3. Choose **Create deployment**.

Note

You will not be creating a new deployment. This is currently the only way to give AWS CodeDeploy permission to interact with GitHub on behalf of your GitHub user account.

4. From the **Application** drop-down list, choose **CodeDeployGitHubDemo-App**.
5. From the **Deployment group** drop-down list, choose **CodeDeployGitHubDemo-DepGrp**.
6. In the **Repository type** area, choose **My application is stored in GitHub**.
7. In **Connect to GitHub**, do one of the following:

- To create a connection for AWS CodeDeploy applications to a GitHub account, sign out of GitHub in a separate web browser tab. In **GitHub account**, type a name to identify this connection, and then choose **Connect to GitHub**. The web page prompts you to authorize AWS CodeDeploy to interact with GitHub for the application named CodeDeployGitHubDemo-App. Continue to step 8.
 - To use a connection you have already created, in **GitHub account**, select its name, and then choose **Connect to GitHub**. Continue to step 10.
 - To create a connection to a different GitHub account, sign out of GitHub in a separate web browser tab. Choose **Connect to a different GitHub account**, and then choose **Connect to GitHub**. Continue to step 8.
8. Follow the instructions on the **Sign in** page to sign in with your GitHub user name or email and password.
 9. On the **Authorize application** page, choose **Authorize application**.
 10. On the AWS CodeDeploy **Create deployment** page, choose **Cancel**.
 11. 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 (/), 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 **Your repositories**, hover your mouse pointer over the target repository name. A tooltip appears, displaying the GitHub user or organization name, followed by a forward slash (/), followed by the name of the repository. This is the value to use.

Note

If the target repository name does not appear in **Your repositories**, use the **Search GitHub** box to find the target repository and corresponding GitHub user or organization name.

- **commit-id** is the commit associated with the version of the application revision 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 **Your repositories**, choose **CodeDeployGitHubDemo**.
3. In the list of commits, find the commit ID 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.) Use this value.

If you are working on a local Linux, macOS, or Unix machine:

```
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=commit-id
```

If you are working on a local Windows machine:

```
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=commit-id
```

Step 7: Monitor and Verify the Deployment

In this step, you will use the AWS CodeDeploy console or the AWS CLI to verify the success of the deployment. You will use your web browser to view the web page that was deployed to the instance you created or configured.

Note

If you're deploying to an Ubuntu Server instance, use your own testing strategy to determine whether the deployed revision works as expected on the instance, and then go to the next step.

To monitor and verify the deployment (console)

1. If the **Deployments** page is not displayed, on the AWS CodeDeploy menu, choose **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 **Succeeded** or **Failed** do not appear in the **Status** column, choose the **Refresh** button periodically.
3. If **Failed** appears in the **Status** column, follow the instructions in [View Instance Details \(Console\)](#) (p. 198) to troubleshoot the deployment.
4. If **Succeeded** appears in the **Status** column, you can now verify the deployment through your web browser. Our sample revision deploys a single web page to the instance. If you're deploying to an Amazon EC2 instance, in your web browser, go to `http://public-dns` 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! Now that you've successfully used AWS CodeDeploy to deploy a revision from GitHub, you can skip ahead to [Step 8: Clean Up](#) (p. 123).

To monitor and verify the deployment (CLI)

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 in the output from the **list-deployments** command:

```
aws deploy get-deployment --deployment-id deployment-id --query "deploymentInfo.  
[status, creator]" --output text
```

3. If **Failed** is returned, follow the instructions in [View Instance Details \(Console\)](#) (p. 198) to troubleshoot the deployment.
4. If **Succeeded** is returned, you can now try verifying the deployment through your web browser. Our sample revision is a single web page deployed to the instance. If you're deploying to an Amazon EC2 instance, you can view this page in your web browser by going to `http://public-dns` 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 8: Clean Up

To avoid further charges for resources you used during this tutorial, you must terminate the Amazon EC2 instance and its associated resources. Optionally, you can delete the AWS CodeDeploy deployment component records associated with this tutorial. If you were using a GitHub repository just for this tutorial, you can delete it now, too.

To delete a AWS CloudFormation stack (if you used the 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. In the **Stack Name** column, select the box next to the stack starting with CodeDeploySampleStack.
3. Choose **Delete Stack**.
4. When prompted, choose **Yes, Delete**. The Amazon EC2 instance and the associated IAM instance profile and service role will be 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 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**, choose **Instances**.
3. Select the box next to the Amazon EC2 instance you want to terminate. In the **Actions** menu, point to **Instance State**, and then choose **Terminate**.
4. When prompted, choose **Yes, Terminate**.

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

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#).

2. If the **Applications** page is not displayed, on the AWS CodeDeploy menu, choose **Applications**.
3. Choose **CodeDeployGitHubDemo-App**.
4. On the **Application details** page, in **Deployment groups**, choose the button next to the deployment group. On the **Actions** menu, choose **Delete**. When prompted, type the name of the deployment group to confirm you want to delete it, and then choose **Delete**.
5. At the bottom of the **Application details** page, choose **Delete application**.
6. When prompted, type the name of the application to confirm you want to delete it, and then choose **Delete**.

To delete your GitHub repository

See [Deleting a repository](#) in [GitHub Help](#).

Working with the AWS CodeDeploy Agent

The AWS CodeDeploy agent is a software package that, when installed and configured on an instance, enables that instance to be used in AWS CodeDeploy deployments.

Note

The AWS CodeDeploy agent is required only if you deploy to an EC2/On-Premises compute platform. The agent is not required for deployments that use the AWS Lambda compute platform.

A configuration file is placed on the instance when the agent is installed. This file is used to specify how the agent works. This configuration file specifies directory paths and other settings for AWS CodeDeploy to use as it interacts with the instance. You can change some of the configuration options in the file. For information about working with the AWS CodeDeploy agent configuration file, see [AWS CodeDeploy Agent Configuration Reference \(p. 330\)](#).

For more information about working with the AWS CodeDeploy agent, such as steps for installing, updating, and verifying versions, see [Managing AWS CodeDeploy Agent Operations \(p. 132\)](#).

Topics

- [Operating Systems Supported by the AWS CodeDeploy Agent \(p. 125\)](#)
- [Communication Protocol and Port for the AWS CodeDeploy Agent \(p. 126\)](#)
- [AWS SDK for Ruby \(aws-sdk-core\) Support for the AWS CodeDeploy Agent \(p. 126\)](#)
- [Version History of the AWS CodeDeploy Agent \(p. 126\)](#)
- [Application Revision and Log File Cleanup \(p. 129\)](#)
- [Files Installed by the AWS CodeDeploy Agent \(p. 129\)](#)
- [Managing AWS CodeDeploy Agent Operations \(p. 132\)](#)

Operating Systems Supported by the AWS CodeDeploy Agent

Supported Amazon EC2 AMI Operating Systems

The AWS CodeDeploy agent has been tested on the following Amazon EC2 AMI operating systems:

- Amazon Linux 2017.03.x, 2016.09.0, 2016.03.1, 2016.03.0, 2015.03, 2014.09.1
- Ubuntu Server 16.04 LTS and 14.04 LTS
- Microsoft Windows Server 2016, 2012 R2, and 2008 R2
- Red Hat Enterprise Linux (RHEL) 7.x

The AWS CodeDeploy agent is available as open source for you to adapt to your needs. It can be used with other Amazon EC2 AMI operating systems. For more information, go to the [AWS CodeDeploy Agent repository in GitHub](#).

Supported On-Premises Operating Systems

The AWS CodeDeploy agent has been tested on the following on-premises operating systems:

- Ubuntu Server 14.04 LTS
- Microsoft Windows Server 2016, 2012 R2, and 2008 R2
- Red Hat Enterprise Linux (RHEL) 7.x

The AWS CodeDeploy agent is available as open source for you to adapt to your needs. It can be used with other on-premises instance operating systems. For more information, go to 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.

AWS SDK for Ruby (aws-sdk-core) Support for the AWS CodeDeploy Agent

Versions of the AWS CodeDeploy agent earlier than 1.0.1.880 are compatible only with version 2.1.2 and earlier versions of the AWS SDK for Ruby (aws-sdk-core 2.1.2). If you are using a version of the AWS CodeDeploy agent earlier than 1.0.1.880, we recommend that you update to the latest version. For information, see the following:

- [Determine the Version of the AWS CodeDeploy Agent \(p. 133\)](#)
- [Install or Reinstall the AWS CodeDeploy Agent \(p. 134\)](#)

The latest version of the AWS SDK for Ruby compatible the AWS CodeDeploy Agent is aws-sdk-core 2.3.

Version History of the AWS CodeDeploy Agent

Your instances must be running a supported version of the AWS CodeDeploy agent. The current minimum supported version is 1.0.1.1458. If you are running an earlier version, deployments to your instances might fail.

The following table lists all releases of the AWS CodeDeploy agent and the features and enhancements included with each version.

Version	Release date	Details
1.0.1.1518	June 12, 2018	<p>Enhancement: Fixed an issue that caused an error when the AWS CodeDeploy agent is closed while it is accepting poll requests.</p> <p>Enhancement: Added a deployment tracking feature that prevents the AWS CodeDeploy agent from being closed when a deployment is in progress.</p> <p>Enhancement: Improved performance when deleting files.</p>

Version	Release date	Details
1.0.1.1458	March 6, 2018	<p>Important The minimum supported version of the AWS CodeDeploy Agent is 1.0.1.1458. Use of an earlier AWS CodeDeploy agent may cause deployments to fail.</p> <p>Enhancement: Improved certificate validations to support more trusted authorities.</p> <p>Enhancement: Fixed an issue that caused the local CLI to fail during a deployment that includes a BeforeInstall lifecycle event.</p> <p>Enhancement: Fixed an issue that might cause an active deployment to fail when the AWS CodeDeploy agent is updated.</p>
1.0.1.1352	November 16, 2017	<p>Feature: Introduced a new feature for testing and debugging an EC2/On-Premises deployment on a local machine or instance where the AWS CodeDeploy Agent is installed.</p>
1.0.1.1106	May 16, 2017	<p>Feature: Introduced new support for handling content in a target location that wasn't part of the application revision from the most recent successful deployment. Deployments options for existing content now include retaining the content, overwriting the content, or failing the deployment.</p> <p>Enhancement: Made the AWS CodeDeploy agent compatible with version 2.9.2 of the AWS SDK for Ruby (aws-sdk-core 2.9.2).</p>
1.0.1.1095	March 29, 2017	<p>Enhancement: Introduced support for the AWS CodeDeploy agent in the China (Beijing) Region.</p> <p>Enhancement: Enabled Puppet to run on Windows Server instances when invoked by a lifecycle event hook.</p> <p>Enhancement: Improved the handling of untar operations.</p>
1.0.1.1067	January 6, 2017	<p>Enhancement: Revised many error messages to include more specific causes for deployment failures.</p> <p>Enhancement: Fixed an issue that prevented the AWS CodeDeploy agent from identifying the correct application revision to deploy during some deployments.</p> <p>Enhancement: Reverted the usage of <code>pushd</code> and <code>popd</code> before and after the <code>untar</code> operation.</p>
1.0.1.1045	November 21, 2016	<p>Enhancement: Made the AWS CodeDeploy agent compatible with version 2.6.11 of the AWS SDK for Ruby (aws-sdk-core 2.6.11).</p>
1.0.1.1037	October 19, 2016	<p>The AWS CodeDeploy agent for Amazon Linux, RHEL, and Ubuntu Server instances has been updated with the following change. For Windows Server instances, the latest version remains 1.0.1.998.</p> <p>Enhancement: The agent can now determine which version of Ruby is installed on an instance so it can invoke the <code>codedeploy-agent</code> script using that version.</p>

Version	Release date	Details
1.0.1.1011.1	August 17, 2016	Enhancement: Removed the changes introduced by version 1.0.1.1011 due to issues with shell support. This version of the agent is functionally equivalent to version 1.0.1.998 released on July 11, 2016.
1.0.1.1011	August 15, 2016	<p>The AWS CodeDeploy agent for Amazon Linux, RHEL, and Ubuntu Server instances has been updated with the following changes. For Windows Server instances, the latest version remains 1.0.1.998.</p> <p>Feature: Added support for invoking the AWS CodeDeploy agent using the bash shell on operating systems where the systemd init system is in use.</p> <p>Enhancement: Enabled support for all versions of Ruby 2.x in the AWS CodeDeploy agent and the AWS CodeDeploy agent updater. Updated AWS CodeDeploy agents are no longer dependent on Ruby 2.0 only. (Ruby 2.0 is still required for deb and rpm versions of the AWS CodeDeploy agent installer.)</p>
1.0.1.998	July 11, 2016	Enhancement: Fixed support for running the AWS CodeDeploy agent with user profiles other than <i>root</i> . The variable named <code>USER</code> is replaced by <code>CODEDEPLOY_USER</code> to avoid conflicts with environmental variables.
1.0.1.966	June 16, 2016	<p>Feature: Introduced support for running the AWS CodeDeploy agent with user profiles other than <i>root</i>.</p> <p>Enhancement: Fixed support for specifying the number of application revisions you want the AWS CodeDeploy agent to archive for a deployment group.</p> <p>Enhancement: Made the AWS CodeDeploy agent compatible with version 2.3 of the AWS SDK for Ruby (aws-sdk-core 2.3).</p> <p>Enhancement: Fixed issues with UTF-8 encoding during deployments.</p> <p>Enhancement: Improved accuracy when identifying process names.</p>
1.0.1.950	March 24, 2016	<p>Feature: Added installation proxy support.</p> <p>Enhancement: Updated the installation script to not download the AWS CodeDeploy agent if the latest version is already installed.</p>
1.0.1.934	February 11, 2016	Feature: Introduced support for specifying the number of application revisions you want the AWS CodeDeploy agent to archive for a deployment group.
1.0.1.880	January 11, 2016	Enhancement: Made the AWS CodeDeploy agent compatible with version 2.2 of the AWS SDK for Ruby (aws-sdk-core 2.2). Version 2.1.2 is still supported.

Version	Release date	Details
1.0.1.854	November 17, 2015	<p>Feature: Introduced support for the SHA-256 hash algorithm.</p> <p>Important After October 17, 2016, all installations of the AWS CodeDeploy agent must be updated, at minimum, to version 1.0.1.854 or deployments will fail. For more information see Deployment fails with the message "Validation of PKCS7 signed message failed" (p. 348).</p> <p>Feature: Introduced version tracking support in <code>.version</code> files.</p> <p>Feature: Made the deployment group ID available through the use of an environment variable.</p> <p>Enhancement: Added support for monitoring AWS CodeDeploy agent logs using Amazon CloudWatch Logs.</p>

For related information, see the following:

- [Determine the Version of the AWS CodeDeploy Agent](#) (p. 133)
- [Install or Reinstall the AWS CodeDeploy Agent](#) (p. 134)

For a history of AWS CodeDeploy agent versions, see the [Release Repository on GitHub](#).

Application Revision and Log File Cleanup

The AWS CodeDeploy agent archives revisions and log files on instances. The AWS CodeDeploy agent cleans up these artifacts to conserve disk space.

Application revision deployment logs: You can use the `:max_revisions:` option in the agent configuration file to specify the number of application revisions to archive by entering any positive integer. AWS CodeDeploy also archives the log files for those revisions. All others are deleted, with the exception of the log file of the last successful deployment. That log file will always be retained, even if the number of failed deployments exceeds the number of retained revisions. If no value is specified, AWS CodeDeploy will retain the five most recent revisions in addition to the currently deployed revision.

AWS CodeDeploy logs: For Amazon Linux, Ubuntu Server, and RHEL instances, the AWS CodeDeploy agent rotates the log files under the `/var/log/aws/codedeploy-agent` folder. The log file is rotated at 00:00:00 (instance time) daily. Log files are deleted after seven days. The naming pattern for rotated log files is `codedeploy-agent.YYYYMMDD.log`.

Files Installed by the AWS CodeDeploy Agent

The AWS CodeDeploy agent stores revisions, deployment history, and deployment scripts in its root directory on an instance. The default name and location of this directory is:

`'/opt/codedeploy-agent/deployment-root'` for Amazon Linux, Ubuntu Server, and RHEL instances.

`'C:\ProgramData\Amazon\CodeDeploy'` for Windows Server instances.

You can use the **root_dir** setting in the AWS CodeDeploy agent configuration file to configure the directory's name and location. For more information, see [AWS CodeDeploy Agent Configuration Reference \(p. 330\)](#).

The following is an example of the file and directory structure under the root directory. The structure assumes there are N number of deployment groups, and each deployment group contains N number of deployments.

```
|--deployment-root/
|-- deployment group 1 ID
|   |-- deployment 1 ID
|   |   |-- Contents and logs of the deployment's revision
|   |-- deployment 2 ID
|   |   |-- Contents and logs of the deployment's revision
|   |-- deployment N ID
|   |   |-- Contents and logs of the deployment's revision
|-- deployment group 2 ID
|   |-- deployment 1 ID
|   |   |-- bundle.tar
|   |   |-- deployment-archive
|   |   |   |-- contents of the deployment's revision
|   |   |-- logs
|   |   |   |-- scripts.log
|   |-- deployment 2 ID
|   |   |-- bundle.tar
|   |   |-- deployment-archive
|   |   |   |-- contents of the deployment's revision
|   |   |-- logs
|   |   |   |-- scripts.log
|   |-- deployment N ID
|   |   |-- bundle.tar
|   |   |-- deployment-archive
|   |   |   |-- contents of the deployment's revision
|   |   |-- logs
|   |   |   |-- scripts.log
|-- deployment group N ID
|   |-- deployment 1 ID
|   |   |-- Contents and logs of the deployment's revision
|   |-- deployment 2 ID
|   |   |-- Contents and logs of the deployment's revision
|   |-- deployment N ID
|   |   |-- Contents and logs of the deployment's revision
|-- deployment-instructions
|   |-- [deployment group 1 ID]_cleanup
|   |-- [deployment group 2 ID]_cleanup
|   |-- [deployment group N ID]_cleanup
|   |-- [deployment group 1 ID]_install.json
|   |-- [deployment group 2 ID]_install.json
|   |-- [deployment group N ID]_install.json
|   |-- [deployment group 1 ID]_last_successful_install
|   |-- [deployment group 2 ID]_last_successful_install
|   |-- [deployment group N ID]_last_successful_install
|   |-- [deployment group 1 ID]_most_recent_install
|   |-- [deployment group 2 ID]_most_recent_install
|   |-- [deployment group N ID]_most_recent_install
|-- deployment-logs
|   |-- codedeploy-agent-deployments.log
```

- **Deployment Group ID** folders represent each of your deployment groups. A deployment group directory's name is its ID (for example, acde1916-9099-7caf-fd21-012345abcdef). Each deployment group directory contains one subdirectory for each attempted deployment in that deployment group.

You can use the [batch-get-deployments](#) command to find a deployment group ID.

- **Deployment ID** folders represent each deployment in a deployment group. Each deployment directory's name is its ID. Each folder contains:
 - **bundle.tar**, a compressed file with the contents of the deployment's revision. Use a zip decompression utility if you want to view the revision.
 - **deployment-archive**, a directory that contains the contents of the deployment's revision.
 - **logs**, a directory that contains a `scripts.log` file. This file lists the output of all scripts specified in the deployment's AppSpec file.

If you want to find the folder for a deployment but don't know its deployment ID or deployment group ID, you can use the [AWS CodeDeploy console](#) or the AWS CLI to find them. For more information, see [View Deployment Details with AWS CodeDeploy \(p. 254\)](#).

The default maximum number of deployments that can be archived in a deployment group is five. When that number is reached, future deployments are archived and the oldest archive is deleted. You can use the **max_revisions** setting in the AWS CodeDeploy agent configuration file to change the default. For more information, see [AWS CodeDeploy Agent Configuration Reference \(p. 330\)](#).

Note

If you want to recover hard disk space used by archived deployments, update the **max_revisions** setting to a low number, such as one or two. The next deployment deletes archived deployments so that the number is equal to the you specified.

- **deployment-instructions** contains four text files for each deployment group:
 - **[Deployment Group ID]-cleanup**, a text file with an undo version of each command that is run during a deployment. An example file name is `acde1916-9099-7caf-fd21-012345abcdef-cleanup`.
 - **[Deployment Group ID]-install.json**, a JSON file created during the most recent deployment. It contains the commands run during the deployment. An example file name is `acde1916-9099-7caf-fd21-012345abcdef-install.json`.
 - **[Deployment Group ID]_last_successfull_install**, a text file that lists the archive directory of the last successful deployment. This file is created when the AWS CodeDeploy agent has copied all files in the deployment application to the instance. It is used by the AWS CodeDeploy agent during the next deployment to determine which `ApplicationStop` and `BeforeInstall` scripts to run. An example file name is `acde1916-9099-7caf-fd21-012345abcdef_last_successfull_install`.
 - **[Deployment Group ID]_most_recent_install**, a text file that lists the name of the archive directory of the most recent deployment. This file is created when the files in the deployment are successfully downloaded. The `[deployment group ID]_last_successfull_install` file is created after this file, when the downloaded files are copied to their final destination. An example file name is `acde1916-9099-7caf-fd21-012345abcdef_most_recent_install`.
- **deployment-logs** contains the following log files:
 - **codedeploy-agent.yyyymmdd.log** files are created for each day there is a deployment. Each log file contains information about the day's deployments. These log files might be useful for debugging problems like a permissions issue. The log file is initially named `codedeploy-agent.log`. The next day, the date of its deployments is inserted into the file name. For example, if today is January 3, 2018, you can see information about all of today's deployments in `codedeploy-agent.log`. Tomorrow, on January 4, 2018, the log file is renamed `codedeploy-agent.20180103.log`.
 - **codedeploy-agent-deployments.log** compiles the contents of `scripts.log` files for each deployment. The `scripts.log` files are located in the `logs` subfolder under each `Deployment ID` folder. The entries in this file are preceded by a deployment ID. For example, "[d-ABCDEF123]LifecycleEvent - BeforeInstall" might be written during a deployment with an ID of `d-ABCDEF123`. When `codedeploy-agent-deployments.log` reaches its maximum size, the AWS CodeDeploy agent continues to write to it while deleting old content.

Managing AWS CodeDeploy Agent Operations

The instructions in this section show you how to install, uninstall, reinstall, or update the AWS CodeDeploy agent and how to verify the AWS CodeDeploy agent is running.

Important

The minimum supported version of the AWS CodeDeploy Agent is 1.0.1.1458. Use of an earlier AWS CodeDeploy agent may cause deployments to fail.

Topics

- [Verify the AWS CodeDeploy Agent Is Running \(p. 132\)](#)
- [Determine the Version of the AWS CodeDeploy Agent \(p. 133\)](#)
- [Install or Reinstall the AWS CodeDeploy Agent \(p. 134\)](#)
- [Update the AWS CodeDeploy Agent \(p. 141\)](#)
- [Uninstall the AWS CodeDeploy Agent \(p. 145\)](#)

Verify the AWS CodeDeploy Agent Is Running

This section describes commands to run if you suspect the AWS CodeDeploy agent has stopped running on an instance.

Topics

- [Verify the AWS CodeDeploy agent for Amazon Linux or RHEL is running \(p. 132\)](#)
- [Verify the AWS CodeDeploy agent for Ubuntu Server is running \(p. 132\)](#)
- [Verify the AWS CodeDeploy agent for Windows Server is running \(p. 133\)](#)

Verify the AWS CodeDeploy agent for Amazon Linux or RHEL 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 or reinstall the AWS CodeDeploy agent for Amazon Linux or RHEL \(p. 135\)](#).

If the AWS CodeDeploy agent is installed and running, you should see a message like `The AWS CodeDeploy agent is running.`

If you see a message like `error: No AWS CodeDeploy agent running`, start the service and run the following two commands, one at a time:

```
sudo service codedeploy-agent start
```

```
sudo service codedeploy-agent status
```

Verify 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 or reinstall the AWS CodeDeploy agent for Ubuntu Server \(p. 136\)](#).

If the AWS CodeDeploy agent is installed and running, you should see a message like The AWS CodeDeploy agent is running.

If you see a message like error: No AWS CodeDeploy agent running, start the service and run the following two commands, one at a time:

```
sudo service codedeploy-agent start
```

```
sudo service codedeploy-agent status
```

Verify 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 or reinstall the AWS CodeDeploy agent for Windows Server \(p. 137\)](#).

If Status shows anything other than Running, start the service with the following command:

```
powershell.exe -Command Start-Service -Name codedeployagent
```

You can restart the service with the following command:

```
powershell.exe -Command Restart-Service -Name codedeployagent
```

You can stop the service with the following command:

```
powershell.exe -Command Stop-Service -Name codedeployagent
```

Determine the Version of the AWS CodeDeploy Agent

You can determine the version of the AWS CodeDeploy agent running on your instance in two ways.

First, starting with version 1.0.1.854 of the AWS CodeDeploy agent, you can view the version number in a .version file on the instance. The following table shows the location and sample version string for each of the supported operating systems.

Operating system	File location	Sample agent_version string
Amazon Linux and Red Hat Enterprise Linux (RHEL)	/opt/codedeploy-agent/.version	OFFICIAL_1.0.1.854_rpm
Ubuntu Server	/opt/codedeploy-agent/.version	OFFICIAL_1.0.1.854_deb
Windows Server	C:\ProgramData\Amazon\CodeDeploy\.version	OFFICIAL_1.0.1.854_msi

Second, you can run a command on an instance to determine the version of the AWS CodeDeploy agent.

Topics

- [Determine the version on Amazon Linux or RHEL \(p. 134\)](#)
- [Determine the version on Ubuntu Server \(p. 134\)](#)
- [Determine the version on Windows Server \(p. 134\)](#)

Determine the version on Amazon Linux or RHEL

Sign in to the instance and run the following command:

```
sudo yum info codedeploy-agent
```

Determine the version on Ubuntu Server

Sign in to the instance and run the following command:

```
sudo dpkg -s codedeploy-agent
```

Determine the version on Windows Server

Sign in to the instance and run the following command:

```
sc qdescription codedeployagent
```

Install or Reinstall the AWS CodeDeploy Agent

If you suspect the AWS CodeDeploy agent is missing or not working, you can run commands on an instance to install or reinstall it.

Important

An IAM instance profile with permission to access the Amazon S3 bucket that contains the agent installation files for your region must be attached to each Amazon EC2 instance. Without the permissions provided by the IAM instance profile, the instance will not be able to download AWS CodeDeploy agent installation files. For information, see [Step 4: Create an IAM Instance Profile for Your Amazon EC2 Instances \(p. 25\)](#) and [Configure an Amazon EC2 Instance to Work with AWS CodeDeploy \(p. 168\)](#).

Topics

- [Install or reinstall the AWS CodeDeploy agent for Amazon Linux or RHEL \(p. 135\)](#)
- [Install or reinstall the AWS CodeDeploy agent for Ubuntu Server \(p. 136\)](#)
- [Install or reinstall the AWS CodeDeploy agent for Windows Server \(p. 137\)](#)

Install or reinstall the AWS CodeDeploy agent for Amazon Linux or RHEL

Sign in to the instance, and run the following commands, one at a time.

Note

In the fourth command, `/home/ec2-user` represents the default user name for an Amazon Linux or RHEL Amazon EC2 instance. If your instance was created using a custom AMI, the AMI owner might have specified a different default user name.

```
sudo yum update
```

```
sudo yum install ruby
```

```
sudo yum install wget
```

```
cd /home/ec2-user
```

```
wget https://bucket-name.s3.amazonaws.com/latest/install
```

```
chmod +x ./install
```

```
sudo ./install auto
```

bucket-name is the name of the Amazon S3 `sds-s3-latest-bucket-name` bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace *bucket-name* with `aws-codedeploy-us-east-2`. For a list of bucket names, see [Resource Kit Bucket Names by Region \(p. 335\)](#).

To check that the service is running, run the following command:

```
sudo service codedeploy-agent status
```

If the AWS CodeDeploy agent is installed and running, you should see a message like `The AWS CodeDeploy agent is running.`

If you see a message like `error: No AWS CodeDeploy agent running`, start the service and run the following two commands, one at a time:

```
sudo service codedeploy-agent start
```

```
sudo service codedeploy-agent status
```

Install or reinstall the AWS CodeDeploy agent for Ubuntu Server

Sign in to the instance, and run the following commands, one at a time.

Note

In the fifth command, `/home/ubuntu` represents the default user name for an Ubuntu Server instance. If your instance was created using a custom AMI, the AMI owner might have specified a different default user name.

```
sudo apt-get update
```

On Ubuntu Server 14.04:

- ```
sudo apt-get install ruby2.0
```

On Ubuntu Server 16.04:

- ```
sudo apt-get install ruby
```

```
sudo apt-get install wget
```

```
cd /home/ubuntu
```

```
wget https://bucket-name.s3.amazonaws.com/latest/install
```

```
chmod +x ./install
```

```
sudo ./install auto
```

bucket-name is the name of the Amazon S3 `sds-s3-latest-bucket-name` bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace *bucket-name* with `aws-codedeploy-us-east-2`. For a list of bucket names, see [Resource Kit Bucket Names by Region](#) (p. 335).

To check that the service is running, run the following command:

```
sudo service codedeploy-agent status
```

If the AWS CodeDeploy agent is installed and running, you should see a message like `The AWS CodeDeploy agent is running.`

If you see a message like `error: No AWS CodeDeploy agent running`, start the service and run the following two commands, one at a time:

```
sudo service codedeploy-agent start
```

```
sudo service codedeploy-agent status
```


Install or reinstall the AWS CodeDeploy agent for Windows Server

On Windows Server instances, you can use one of these methods to download and install the AWS CodeDeploy agent:

- Run a series of Windows PowerShell commands
- Choose a direct download link
- Run an Amazon S3 copy command

Note

On both new and existing instances, we recommend installing the AWS CodeDeploy agent updater for Windows Server. The updater checks periodically for new versions of the agent and installs it when a new version is available. On new instances, you can install the updater instead of the agent, and the current version of the agent will be installed immediately after the updater. For more information, see [Update the AWS CodeDeploy Agent on Windows Server \(p. 141\)](#).

Topics

- [Use Windows PowerShell \(p. 137\)](#)
- [Use a Direct Link \(p. 138\)](#)
- [Use an Amazon S3 Copy Command \(p. 139\)](#)

Use Windows PowerShell

Sign in to the instance, and run the following commands in Windows PowerShell:

1. Require that all scripts and configuration files downloaded from the Internet be signed by a trusted publisher. If you are prompted to change the execution policy, type "Y."

```
Set-ExecutionPolicy RemoteSigned
```

2. Load the AWS Tools for Windows PowerShell.

```
Import-Module AWSPowerShell
```

3. Create a directory into where the AWS CodeDeploy agent installation file is downloaded.

```
New-Item -Path "c:\temp" -ItemType "directory" -Force
```

4. Download the AWS CodeDeploy agent installation file.

```
powershell.exe -Command Read-S3Object -BucketName bucket-name -Key latest/codedeploy-agent.msi -File c:\temp\codedeploy-agent.msi
```

5. Run the AWS CodeDeploy agent installation file.

```
c:\temp\codedeploy-agent.msi /quiet /l c:\temp\host-agent-install-log.txt
```

bucket-name is the name of the Amazon S3 sds-s3-latest-bucket-name bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace *bucket-name* with aws-codedeploy-us-east-2. For a list of bucket names, see [Resource Kit Bucket Names by Region \(p. 335\)](#).

To check that the service is running, run the following command:

```
powershell.exe -Command Get-Service -Name codedeployagent
```

If the AWS CodeDeploy agent was just installed and has not been started, then after running the **Get-Service** command, under **Status**, you should see **Start...**:

Status	Name	DisplayName
-----	----	-----
Start...	codedeployagent	CodeDeploy Host Agent Service

If the AWS CodeDeploy agent is already running, after running the **Get-Service** command, under **Status**, you should see **Running**:

Status	Name	DisplayName
-----	----	-----
Running	codedeployagent	CodeDeploy Host Agent Service

Use a Direct Link

If the browser security settings on the Windows Server instance provide the permissions (for example, to `http://*.s3.amazonaws.com`), you can use a direct link for your region to download the AWS CodeDeploy agent and then run the installer manually.

Region name	Download link
US East (Ohio)	https://aws-codedeploy-us-east-2.s3.amazonaws.com/latest/codedeploy-agent.msi
US East (N. Virginia)	https://aws-codedeploy-us-east-1.s3.amazonaws.com/latest/codedeploy-agent.msi
US West (N. California)	https://aws-codedeploy-us-west-1.s3.amazonaws.com/latest/codedeploy-agent.msi
US West (Oregon)	https://aws-codedeploy-us-west-2.s3.amazonaws.com/latest/codedeploy-agent.msi
Canada (Central)	https://aws-codedeploy-ca-central-1.s3.amazonaws.com/latest/codedeploy-agent.msi
EU (Ireland)	https://aws-codedeploy-eu-west-1.s3.amazonaws.com/latest/codedeploy-agent.msi
EU (London)	https://aws-codedeploy-eu-west-2.s3.amazonaws.com/latest/codedeploy-agent.msi

Region name	Download link
EU (Paris)	https://aws-codedeploy-eu-west-3.s3.amazonaws.com/latest/codedeploy-agent.msi
EU (Frankfurt)	https://aws-codedeploy-eu-central-1.s3.amazonaws.com/latest/codedeploy-agent.msi
Asia Pacific (Tokyo)	https://aws-codedeploy-ap-northeast-1.s3.amazonaws.com/latest/codedeploy-agent.msi
Asia Pacific (Seoul)	https://aws-codedeploy-ap-northeast-2.s3.amazonaws.com/latest/codedeploy-agent.msi
Asia Pacific (Singapore)	https://aws-codedeploy-ap-southeast-1.s3.amazonaws.com/latest/codedeploy-agent.msi
Asia Pacific (Sydney)	https://aws-codedeploy-ap-southeast-2.s3.amazonaws.com/latest/codedeploy-agent.msi
Asia Pacific (Mumbai)	https://aws-codedeploy-ap-south-1.s3.amazonaws.com/latest/codedeploy-agent.msi
South America (São Paulo)	https://aws-codedeploy-sa-east-1.s3.amazonaws.com/latest/codedeploy-agent.msi

Use an Amazon S3 Copy Command

If the AWS CLI is installed on the instance, you can use the Amazon S3 `cp` command to download the AWS CodeDeploy agent and then run the installer manually. For information, see [Install the AWS Command Line Interface on Microsoft Windows](#).

Region name	Amazon S3 copy command
US East (Ohio)	<pre>aws s3 cp s3://aws-codedeploy-us-east-2/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
US East (N. Virginia)	<pre>aws s3 cp s3://aws-codedeploy-us-east-1/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
US West (N. California)	<pre>aws s3 cp s3://aws-codedeploy-us-west-1/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>

Region name	Amazon S3 copy command
US West (Oregon)	<pre>aws s3 cp s3://aws-codedeploy-us-west-2/ latest/codedeploy-agent.msi codedeploy- agent.msi</pre>
Canada (Central)	<pre>aws s3 cp s3://aws-codedeploy-ca-central-1/ latest/codedeploy-agent.msi codedeploy- agent.msi</pre>
EU (Ireland)	<pre>aws s3 cp s3://aws-codedeploy-eu-west-1/ latest/codedeploy-agent.msi codedeploy- agent.msi</pre>
EU (London)	<pre>aws s3 cp s3://aws-codedeploy-eu-west-2/ latest/codedeploy-agent.msi codedeploy- agent.msi</pre>
EU (Paris)	<pre>aws s3 cp s3://aws-codedeploy-eu-west-3/ latest/codedeploy-agent.msi codedeploy- agent.msi</pre>
EU (Frankfurt)	<pre>aws s3 cp s3://aws-codedeploy-eu-central-1/ latest/codedeploy-agent.msi codedeploy- agent.msi</pre>
Asia Pacific (Tokyo)	<pre>aws s3 cp s3://aws-codedeploy-ap- northeast-1/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
Asia Pacific (Seoul)	<pre>aws s3 cp s3://aws-codedeploy-ap- northeast-2/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
Asia Pacific (Singapore)	<pre>aws s3 cp s3://aws-codedeploy-ap- southeast-1/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
Asia Pacific (Sydney)	<pre>aws s3 cp s3://aws-codedeploy-ap- southeast-2/latest/codedeploy-agent.msi codedeploy-agent.msi</pre>
Asia Pacific (Mumbai)	<pre>aws s3 cp s3://aws-codedeploy-ap-south-1/ latest/codedeploy-agent.msi codedeploy- agent.msi</pre>
South America (São Paulo)	<pre>aws s3 cp s3://aws-codedeploy-sa-east-1/ latest/codedeploy-agent.msi codedeploy- agent.msi</pre>

Update the AWS CodeDeploy Agent

For the Amazon Linux, RHEL, and Ubuntu Server operating systems, the AWS CodeDeploy agent is updated automatically when a new version is released. For Windows Server, you can install the AWS CodeDeploy agent updater for Windows Server after or instead of the AWS CodeDeploy agent. The agent will be updated whenever a new version is detected. You can also force updates on all supported operating systems by running a command on an instance.

Topics

- [Update the AWS CodeDeploy Agent on Amazon Linux or RHEL \(p. 141\)](#)
- [Update the AWS CodeDeploy Agent on Ubuntu Server \(p. 141\)](#)
- [Update the AWS CodeDeploy Agent on Windows Server \(p. 141\)](#)

Update the AWS CodeDeploy Agent on Amazon Linux or RHEL

After the AWS CodeDeploy agent (`codedeploy-agent.noarch.rpm`) is installed on an instance, it will be updated automatically within 24 hours of the release of a new version. The update time cannot be easily cancelled or rescheduled. If a deployment is in progress during the update, the current deployment lifecycle event will finish first. After the update is complete, the deployment will resume with the next deployment lifecycle event.

If you want to force an update of the AWS CodeDeploy agent, sign in to the instance, and run the following command:

```
sudo /opt/codedeploy-agent/bin/install auto
```

Update the AWS CodeDeploy Agent on Ubuntu Server

After the AWS CodeDeploy agent (`codedeploy-agent_all.deb`) is installed on an instance, it will be updated automatically within 24 hours of the release of a new version. The update time cannot be easily cancelled or rescheduled. If a deployment is in progress during the update, the current deployment lifecycle event finishes first. After the update is complete, the deployment resumes with the next deployment lifecycle event.

If you want to force an update of the AWS CodeDeploy agent, sign in to the instance, and run the following command:

```
sudo /opt/codedeploy-agent/bin/install auto
```

Update the AWS CodeDeploy Agent on Windows Server

To enable automatic updates of the AWS CodeDeploy agent whenever a new version is released, install the AWS CodeDeploy agent updater for Windows Server on new or existing instances. The updater checks periodically for new versions. When a new version is detected, the updater uninstalls the current version of the agent, if one is installed, before installing the newest version.

If a deployment is already underway when the updater detects a new version, the deployment continues to completion. If a deployment attempts to start during the update process, the deployment fails.

If you want to force an update of the AWS CodeDeploy agent, follow the instructions in [Install or reinstall the AWS CodeDeploy agent for Windows Server \(p. 137\)](#).

On Windows Server instances, you can download and install the AWS CodeDeploy agent updater by running a series of Windows PowerShell commands, using a direct download link, or running an Amazon S3 copy command.

Topics

- [Use Windows PowerShell \(p. 142\)](#)
- [Use a direct link \(p. 142\)](#)
- [Use an Amazon S3 copy command \(p. 144\)](#)

Use Windows PowerShell

Sign in to the instance, and run the following commands in Windows PowerShell, one at a time:

```
Set-ExecutionPolicy RemoteSigned
```

If you are prompted to change the execution policy, choose **Y** so Windows PowerShell requires all scripts and configuration files downloaded from the internet be signed by a trusted publisher.

```
Import-Module AWSPowerShell
```

```
New-Item -Path "c:\temp" -ItemType "directory" -Force
```

```
powershell.exe -Command Read-S3Object -BucketName bucket-name -Key latest/codedeploy-agent-updater.msi -File c:\temp\codedeploy-agent-updater.msi
```

```
c:\temp\codedeploy-agent-updater.msi /quiet /l c:\temp\host-agent-updater-log.txt
```

```
powershell.exe -Command Get-Service -Name codedeployagent
```

bucket-name is the name of the Amazon S3 sds-s3-latest-bucket-name bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace *bucket-name* with aws-codedeploy-us-east-2. For a list of bucket names, see [Resource Kit Bucket Names by Region \(p. 335\)](#).

If you need to troubleshoot an update process error, type the following command to open the AWS CodeDeploy agent updater log file:

```
notepad C:\ProgramData\Amazon\CodeDeployUpdater\log\codedeploy-agent.updater.log
```

Use a direct link

If the browser security settings on the Windows Server instance provide the necessary permissions (for example, to `http://*.s3.amazonaws.com`), you can use a direct link to download the AWS CodeDeploy agent updater by entering the following in your browser's address bar and then download and run the installer manually.

Region name	Download link
US East (Ohio)	https://aws-codedeploy-us-east-2.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
US East (N. Virginia)	https://aws-codedeploy-us-east-1.s3.amazonaws.com/latest/codedeploy-agent-updater.msi

Region name	Download link
US West (N. California)	https://aws-codedeploy-us-west-1.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
US West (Oregon)	https://aws-codedeploy-us-west-2.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
Canada (Central)	https://aws-codedeploy-ca-central-1.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
EU (Ireland)	https://aws-codedeploy-eu-west-1.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
EU (London)	https://aws-codedeploy-eu-west-2.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
EU (Paris)	https://aws-codedeploy-eu-west-3.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
EU (Frankfurt)	https://aws-codedeploy-eu-central-1.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
Asia Pacific (Tokyo)	https://aws-codedeploy-ap-northeast-1.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
Asia Pacific (Seoul)	https://aws-codedeploy-ap-northeast-2.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
Asia Pacific (Singapore)	https://aws-codedeploy-ap-southeast-1.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
Asia Pacific (Sydney)	https://aws-codedeploy-ap-southeast-2.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
Asia Pacific (Mumbai)	https://aws-codedeploy-ap-south-1.s3.amazonaws.com/latest/codedeploy-agent-updater.msi
South America (São Paulo)	https://aws-codedeploy-sa-east-1.s3.amazonaws.com/latest/codedeploy-agent-updater.msi

Use an Amazon S3 copy command

If the AWS CLI is installed on the instance, you can use the Amazon S3 [cp](#) command to download the AWS CodeDeploy agent updater and then run the installer manually. For information, see [Install the AWS Command Line Interface on Microsoft Windows](#).

Region name	Amazon S3 copy command
US East (Ohio)	<pre>aws s3 cp s3://aws-codedeploy-us-east-2/ latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
US East (N. Virginia)	<pre>aws s3 cp s3://aws-codedeploy-us-east-1/ latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
US West (N. California)	<pre>aws s3 cp s3://aws-codedeploy-us-west-1/ latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
US West (Oregon)	<pre>aws s3 cp s3://aws-codedeploy-us-west-2/ latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
Canada (Central)	<pre>aws s3 cp s3://aws-codedeploy-ca-central-1/ latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
EU (Ireland)	<pre>aws s3 cp s3://aws-codedeploy-eu-west-1/ latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
EU (London)	<pre>aws s3 cp s3://aws-codedeploy-eu-west-2/ latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
EU (Paris)	<pre>aws s3 cp s3://aws-codedeploy-eu-west-3/ latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
EU (Frankfurt)	<pre>aws s3 cp s3://aws-codedeploy-eu-central-1/ latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
Asia Pacific (Tokyo)	<pre>aws s3 cp s3://aws-codedeploy-ap- northeast-1/latest/codedeploy-agent- updater.msi codedeploy-agent-updater.msi</pre>

Region name	Amazon S3 copy command
Asia Pacific (Seoul)	<pre>aws s3 cp s3://aws-codedeploy-ap-northeast-2/latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
Asia Pacific (Singapore)	<pre>aws s3 cp s3://aws-codedeploy-ap-southeast-1/latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
Asia Pacific (Sydney)	<pre>aws s3 cp s3://aws-codedeploy-ap-southeast-2/latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
Asia Pacific (Mumbai)	<pre>aws s3 cp s3://aws-codedeploy-ap-south-1/latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>
South America (São Paulo)	<pre>aws s3 cp s3://aws-codedeploy-sa-east-1/latest/codedeploy-agent-updater.msi codedeploy-agent-updater.msi</pre>

Uninstall the AWS CodeDeploy Agent

You can remove the AWS CodeDeploy Agent from instances when it is no longer needed or when you want to perform a fresh installation.

Uninstall the AWS CodeDeploy Agent from Amazon Linux or RHEL

To uninstall the AWS CodeDeploy agent, sign in to the instance and run the following command:

```
sudo yum erase codedeploy-agent
```

Uninstall the AWS CodeDeploy Agent from Ubuntu Server

To uninstall the AWS CodeDeploy agent, sign in to the instance and run the following command:

```
sudo dpkg --purge codedeploy-agent
```

Uninstall the AWS CodeDeploy Agent from Windows Server

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
```

Alternatively, sign in to the instance, and in **Control Panel**, open **Programs and Features**, choose **CodeDeploy Host Agent**, and then choose **Uninstall**.

Working with Instances for AWS CodeDeploy

AWS CodeDeploy supports deployments to instances running Amazon Linux, Ubuntu Server, Red Hat Enterprise Linux (RHEL), and Windows Server.

You can use AWS CodeDeploy to deploy to both Amazon EC2 instances and on-premises instances. An on-premises instance is any physical device that is not an Amazon EC2 instance and that can run the AWS CodeDeploy agent and connect to public AWS service endpoints. You can use AWS CodeDeploy to simultaneously deploy an application to Amazon EC2 instances running in the cloud and to desktop PCs running in your office or servers in your own data center.

Comparing Amazon EC2 Instances to On-Premises Instances

The following table compares Amazon EC2 instances and on-premises instances:

Subject	Amazon EC2 Instances	On-Premises Instances
Requires you to install and run a version of the AWS CodeDeploy agent that's compatible with the operating system running on the instance.	Yes	Yes
Requires the instance to be able to connect to the AWS CodeDeploy service.	Yes	Yes
Requires an IAM instance profile to be attached to the instance. The IAM instance profile must have permissions to participate in AWS CodeDeploy deployments. For information, see Step 4: Create an IAM Instance Profile for Your Amazon EC2 Instances (p. 25).	Yes	No
Requires you to do one of the following to authenticate and register instances: <ul style="list-style-type: none">Create an IAM user for each instance and store the IAM	No	Yes

Subject	Amazon EC2 Instances	On-Premises Instances
user's account credentials in plain text on the instance. <ul style="list-style-type: none">Create an IAM role that can be assumed by an IAM user on each instance to retrieve periodically refreshed temporary credentials generated through AWS Security Token Service.		
Requires you to register each instance with AWS CodeDeploy before you can deploy to it.	No	Yes
Requires you to tag each instance before AWS CodeDeploy can deploy to it.	Yes	Yes
Can participate in Auto Scaling and Elastic Load Balancing scenarios as part of AWS CodeDeploy deployments.	Yes	No
Can be deployed from Amazon S3 buckets and GitHub repositories.	Yes	Yes
Can support triggers that prompt the sending of SMS or email notifications when specified events occur in deployments or instances.	Yes	Yes
Is subject to being billed for associated deployments.	No	Yes

Instance Tasks for AWS CodeDeploy

To launch or configure instances for use in deployments, choose from the following instructions:

I want to launch a new Amazon Linux or Windows Server Amazon EC2 instance.	<p>To launch the Amazon EC2 instance with the least amount of effort, see Create an Amazon EC2 Instance for AWS CodeDeploy (AWS CloudFormation Template) (p. 162).</p> <p>To launch the Amazon EC2 instance mostly on your own, see Create an Amazon EC2 Instance for AWS CodeDeploy (AWS CLI or Amazon EC2 Console) (p. 156).</p>
I want to launch a new Ubuntu Server or RHEL Amazon EC2 instance.	See Create an Amazon EC2 Instance for AWS CodeDeploy (AWS CLI or Amazon EC2 Console) (p. 156).

I want to configure an Amazon Linux, Windows Server, Ubuntu Server, or RHEL Amazon EC2 instance.	See Configure an Amazon EC2 Instance to Work with AWS CodeDeploy (p. 168).
I want to configure a Windows Server, Ubuntu Server, or RHEL on-premises instance (physical devices that are not Amazon EC2 instances).	See Working with On-Premises Instances (p. 171).
I want AWS CodeDeploy to provision a replacement fleet of instances during a blue/green deployment.	See Working with Deployments in AWS CodeDeploy (p. 244).

To prepare Amazon EC2 instances in Auto Scaling groups, you must follow some additional steps. For more information, see [Integrating AWS CodeDeploy with Auto Scaling](#) (p. 44).

Topics

- [Tagging Instances for AWS CodeDeploy Deployments](#) (p. 149)
- [Working with Amazon EC2 Instances](#) (p. 156)
- [Working with On-Premises Instances](#) (p. 171)
- [View Instance Details](#) (p. 197)
- [Instance Health](#) (p. 198)

Tagging Instances for Deployment Groups in AWS CodeDeploy

To help manage your Amazon EC2 instances and on-premises instances, you can use tags to assign your own metadata to each resource. Tags enable you to categorize your instances in different ways (for example, by purpose, owner, or environment). This is useful when you have many instances. You can quickly identify an instance or group of instances based on the tags you've assigned to them. Each tag consists of a key and an optional value, both of which you define. For more information, see [Tagging Your Amazon EC2 Resources](#).

To specify which instances are included in an AWS CodeDeploy deployment group, you specify tags in one or more *tag groups*. Instances that meet your tag criteria are the ones that the latest application revision is installed on when a deployment to that deployment group is created.

Note

You can also include Auto Scaling groups in deployment groups, but they are identified by their names rather than by tags applied to instances. For information, see [Integrating AWS CodeDeploy with Auto Scaling](#) (p. 44).

The criteria for instances in a deployment group can be as simple as a single tag in a single tag group, or as complex as 10 tags each in a maximum of three tag groups.

If you use a single tag group, any instance identified by at least one tag in the group is included in the deployment group. If you use multiple tag groups, only instances that are identified by at least one tag in *each* of the tag groups are included.

The following examples illustrate how tags and tag groups can be used to select the instances for a deployment group.

Topics

- [Example 1: Single Tag Group, Single Tag](#) (p. 150)

- [Example 2: Single Tag Group, Multiple Tags \(p. 150\)](#)
- [Example 3: Multiple Tag Groups, Single Tags \(p. 151\)](#)
- [Example 4: Multiple Tag Groups, Multiple Tags \(p. 153\)](#)

Example 1: Single Tag Group, Single Tag

You can specify a single tag in a single tag group:

Tag Group 1

Key	Value
Name	AppVersion-ABC

Each instance that is tagged with `Name=AppVersion-ABC` is part of the deployment group, even if it has other tags applied.

AWS CodeDeploy console setup view:

Environment configuration

Specify any combination of Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add instances to this deployment group.

Auto Scaling groups

Amazon EC2 instances

On-premises instances

You can add up to three groups of tags for EC2 instances to this deployment group. [Learn more](#)

One tag group : Any instance identified by the tag group will be deployed to.

Multiple tag groups : Only instances identified by all the tag groups will be deployed to.

Tag group 1

	Key	Value	Instances	
1	Name	AppVersion-ABC	7	✕
2				✕

[Add tag group](#)

JSON structure:

```
"ec2TagFilters": [  
  {  
    "Type": "KEY_AND_VALUE",  
    "Key": "Name",  
    "Value": "AppVersion-ABC"  
  },  
]
```

Example 2: Single Tag Group, Multiple Tags

You can also specify multiple tags in a single tag group:

Tag Group 1

Key	Value
Region	North

Key	Value
Region	South
Region	East

An instance that is tagged with any of these three tags is part of the deployment group, even if it has other tags applied. If, for example, you had other instances tagged with `Region=West`, they would not be included in the deployment group.

AWS CodeDeploy console setup view:

Environment configuration

Specify any combination of Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add instances to this deployment group.

Auto Scaling groups

Amazon EC2 instances

On-premises instances

You can add up to three groups of tags for EC2 instances to this deployment group. [Learn more](#)

One tag group : Any instance identified by the tag group will be deployed to.

Multiple tag groups : Only instances identified by all the tag groups will be deployed to.

Tag group 1

	Key	Value	Instances	
1	Region	North	4	✕
2	Region	South	2	✕
3	Region	East	2	✕
4				✕

[Add tag group](#)

JSON structure:

```
"ec2TagFilters": [  
  {  
    "Type": "KEY_AND_VALUE",  
    "Key": "Region",  
    "Value": "North"  
  },  
  {  
    "Type": "KEY_AND_VALUE",  
    "Key": "Region",  
    "Value": "South"  
  },  
  {  
    "Type": "KEY_AND_VALUE",  
    "Key": "Region",  
    "Value": "East"  
  }  
],
```

Example 3: Multiple Tag Groups, Single Tags

You can also use multiple sets of tag groups with a single key-value pair in each to specify the criteria for instances in a deployment group. When you use multiple tag groups in a deployment group, only instances that are identified by all the tag groups are included in the deployment group.

Tag Group 1

Key	Value
Name	AppVersion-ABC

Tag Group 2

Key	Value
Region	North

Tag Group 3

Key	Value
Type	t2.medium

You might have instances in many regions and of various instance types tagged with `Name=AppVersion-ABC`, but in this example, only the instances also tagged with `Region=North` and `Type=t2.medium` are part of the deployment group.

AWS CodeDeploy console setup view:

Environment configuration

Specify any combination of Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add instances to this deployment group.

Auto Scaling groups

Amazon EC2 instances

On-premises instances

You can add up to three groups of tags for EC2 instances to this deployment group. [Learn more](#)

One tag group : Any instance identified by the tag group will be deployed to.

Multiple tag groups : Only instances identified by all the tag groups will be deployed to.

Tag group 1

	Key	Value	Instances	
1	Name	AppVersion-ABC	7	✕
2				✕

Tag group 2

	Key	Value	Instances	
1	Region	North	4	✕
2				✕

✕ Remove tag group

Tag group 3

	Key	Value	Instances	
1	Type	t2.medium	3	✕
2				✕

✕ Remove tag group

JSON structure:

```
"ec2TagSet": {
  "ec2TagSetList": [
    [
      {
        "Key": "KEY_AND_VALUE",
        "Type": "Name",
        "Value": "AppVersion-ABC"
      }
    ],
    [
      {
        "Key": "KEY_AND_VALUE",
        "Type": "Region",
        "Value": "North"
      }
    ],
    [
      {
        "Key": "KEY_AND_VALUE",
        "Type": "Type",
        "Value": "t2.medium"
      }
    ]
  ]
},
```

Example 4: Multiple Tag Groups, Multiple Tags

When you use multiple tag groups with multiple tags in one or more of the groups, an instance must match at least one of the tags in each of the groups to be included in the deployment group.

Tag Group 1

Key	Value
Environment	Beta
Environment	Staging

Tag Group 2

Key	Value
Region	North
Region	South
Region	East

Tag Group 3

Key	Value
Type	t2.medium
Type	t2.large

In this example, to be included in the deployment group, an instance must be tagged with (1) `Environment=Beta` or `Environment=Staging`, with (2) `Region=North`, `Region=South`, or `Region=East`, and with (3) `Type=t2.medium` or `Type=t2.large`.

To illustrate, instances with the following tag groups *would* be among those included in the deployment group:

- `Environment=Beta, Region=North, Type=t2.medium`
- `Environment=Staging, Region=East, Type=t2.large`
- `Environment=Staging, Region=South, Type=t2.large`

Instances with the following tag groups would *not* be included in the deployment group. The **highlighted** key values cause the instances to be excluded:

- `Environment=Beta, Region=West, Type=t2.medium`
- `Environment=Staging, Region=East, Type=t2.micro`
- `Environment=Production, Region=South, Type=t2.large`

AWS CodeDeploy console setup view:

Environment configuration

Specify any combination of Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add instances to this deployment group.

Auto Scaling groups

Amazon EC2 instances

On-premises instances

You can add up to three groups of tags for EC2 instances to this deployment group. [Learn more](#)
One tag group : Any instance identified by the tag group will be deployed to.
Multiple tag groups : Only instances identified by all the tag groups will be deployed to.

Tag group 1

	Key	Value	Instances	
1	Environment	Staging	4	✕
2	Environment	Beta	6	✕
3				✕

Tag group 2

	Key	Value	Instances	
1	Region	South	2	✕
2	Region	North	2	✕
3	Region	East	2	✕
4				✕

✖ Remove tag group

Tag group 3

	Key	Value	Instances	
1	Type	t2.medium	3	✕
2	Type	t2.large	4	✕
3				✕

✖ Remove tag group

JSON structure:

```
"ec2TagSet": {
  "ec2TagSetList": [
    [
      {
        "Key": "KEY_AND_VALUE",
        "Type": "Environment",
        "Value": "Beta"
      },
      {
        "Key": "KEY_AND_VALUE",
        "Type": "Environment",
        "Value": "Staging"
      }
    ],
    [
      {
        "Key": "KEY_AND_VALUE",
        "Type": "Region",
        "Value": "North"
      }
    ]
  ]
}
```

```
{
  "Key": "KEY_AND_VALUE",
  "Type": "Region",
  "Value": "South"
},
{
  "Key": "KEY_AND_VALUE",
  "Type": "Region",
  "Value": "East"
}
],
[
  {
    "Key": "KEY_AND_VALUE",
    "Type": "Type",
    "Value": "t2.medium"
  },
  {
    "Key": "KEY_AND_VALUE",
    "Type": "Type",
    "Value": "t2.large"
  }
],
]
```

Working with Amazon EC2 Instances for AWS CodeDeploy

An Amazon EC2 instance is a virtual computing environment that you create and configure using the Amazon Elastic Compute Cloud service. Amazon EC2 provides scalable computing capacity in the Amazon Web Services (AWS) cloud. You can use Amazon EC2 to launch as many or as few virtual servers as you need for your AWS CodeDeploy deployments.

For more information about Amazon EC2, see [Amazon EC2 Getting Started Guide](#).

The instructions in this section show you how to create and configure Amazon EC2 instances for use in your AWS CodeDeploy deployments.

Topics

- [Create an Amazon EC2 Instance for AWS CodeDeploy \(AWS CLI or Amazon EC2 Console\) \(p. 156\)](#)
- [Create an Amazon EC2 Instance for AWS CodeDeploy \(AWS CloudFormation Template\) \(p. 162\)](#)
- [Configure an Amazon EC2 Instance to Work with AWS CodeDeploy \(p. 168\)](#)

Create an Amazon EC2 Instance for AWS CodeDeploy (AWS CLI or Amazon EC2 Console)

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

You can use our AWS CloudFormation template to launch an Amazon EC2 instance running Amazon Linux or Windows Server that is already configured for use in AWS CodeDeploy deployments. We do not provide an AWS CloudFormation template for Amazon EC2 instances running Ubuntu Server or Red Hat Enterprise Linux (RHEL). For alternatives to the use of the template, see [Working with Instances for AWS CodeDeploy \(p. 147\)](#).

You can use the Amazon EC2 console, AWS CLI, or Amazon EC2 APIs to launch an Amazon EC2 instance.

Launch an Amazon EC2 Instance (Console)

If you have not done so already, follow the instructions in [Getting Started with AWS CodeDeploy \(p. 19\)](#) to set up and configure the AWS CLI and create an IAM instance profile.

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, choose **Instances**, and then choose **Launch Instance**.
3. On the **Step 1: Choose an Amazon Machine Image (AMI)** page, from the **Quick Start** tab, locate the operating system and version you want to use, and then choose **Select**.
4. On the **Step 2: Choose an Instance Type** page, choose any available Amazon EC2 instance type, and then choose **Next: Configure Instance Details**.
5. On the **Step 3: Configure Instance Details** page, in the **IAM role** list, choose the IAM instance role you created in [Step 4: Create an IAM Instance Profile for Your Amazon EC2 Instances \(p. 25\)](#). If you used the suggested role name, then you will choose **CodeDeployDemo-EC2-Instance-Profile**. If you created your own role name, select that.

Note

If neither **Launch into EC2-Classical** nor a default virtual private cloud (VPC) is displayed in the **Network** list, and you are not able to select a different Amazon EC2 instance type that supports launching into EC2-Classical, you must choose an Amazon VPC and subnet, or choose **Create new VPC** or **Create new 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 selected, type the following to install the AWS CodeDeploy agent as the Amazon EC2 instance is launched.

For Amazon Linux and RHEL

```
#!/bin/bash
yum -y update
yum install -y ruby
cd /home/ec2-user
curl -O https://bucket-name.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

bucket-name is the name of the Amazon S3 sds-s3-latest-bucket-name bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace *bucket-name* with aws-codedeploy-us-east-2. For a list of bucket names, see [Resource Kit Bucket Names by Region \(p. 335\)](#).

For Ubuntu Server

Important

If you are installing the AWS CodeDeploy agent on Ubuntu Server 14.04, change the third line the following:

```
apt-get -y install ruby2.0
```

```
#!/bin/bash
apt-get -y update
apt-get -y install ruby
apt-get -y install wget
cd /home/ubuntu
wget https://bucket-name.s3.amazonaws.com/latest/install
chmod +x ./install
```

```
./install auto
```

bucket-name is the name of the Amazon S3 sds-s3-latest-bucket-name bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace **bucket-name** with aws-codedeploy-us-east-2. For a list of bucket names, see [Resource Kit Bucket Names by Region](#) (p. 335).

For Windows Server

```
<powershell>
New-Item -Path c:\temp -ItemType "directory" -Force
powershell.exe -Command 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>
```

bucket-name is the name of the Amazon S3 sds-s3-latest-bucket-name bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace **bucket-name** with aws-codedeploy-us-east-2. For a list of bucket names, see [Resource Kit Bucket Names by Region](#) (p. 335).

8. Leave the rest of the items on this page unchanged, and choose **Next: Add Storage**.
9. Leave the **Step 4: Add Storage** page unchanged, and choose **Next: Add Tags**.
10. On the **Step 5: Add Tags** page, choose **Add Tag**.
11. In the **Key** box, type **Name**. In the **Value** box type **CodeDeployDemo**.

Important

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

12. Choose **Next: Configure Security Group**.
13. On the **Step 6: Configure Security Group** page, leave the **Create a new security group** option selected.

A default SSH role will be configured for Amazon EC2 instances running Amazon Linux, Ubuntu Server, or RHEL. A default RDP role will be configured for Amazon EC2 instances running Windows Server.

14. If you want to open the HTTP port, choose the **Add Rule** button, and from the **Type** drop-down list, choose **HTTP**. Accept the default **Source** value of **Anywhere 0.0.0.0/0**, and then choose **Review and Launch**.

Note

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. For more information, see [Tips for Securing Your Amazon EC2 Instance](#).

If a **Boot from General Purpose (SSD)** dialog box appears, follow the instructions, and then choose **Next**.

15. Leave the **Step 7: Review Instance Launch** page unchanged, and choose **Launch**.
16. In the **Select an existing key pair or create a new key pair** dialog box, choose 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 choose it here.

If you don't already have an Amazon EC2 instance key pair, choose **Create a new key pair** and give it a recognizable name. Choose **Download Key Pair** to download the Amazon EC2 instance key pair to your computer.

Important

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

17. Choose **Launch Instances**.
18. Choose the ID for your Amazon EC2 instance. Do not continue until the instance has been launched and passed all checks.

To verify the AWS CodeDeploy agent is running on the instance, see [Verify the AWS CodeDeploy Agent Is Running \(p. 132\)](#), and then return to this page. After you do this, the Amazon EC2 instance will be ready for use in AWS CodeDeploy deployments.

Launch an Amazon EC2 Instance (CLI)

If you have not done so already, follow the instructions in [Getting Started with AWS CodeDeploy \(p. 19\)](#) to set up and configure the AWS CLI and create an IAM instance profile.

1. **For Windows Server only** If you are creating an Amazon EC2 instance running Windows Server, 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. For example, to create a security group named *CodeDeployDemo-Windows-Security-Group*, run the following commands, one at a time:

```
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
```

Note

For demonstration purposes, these commands create a security group that allows unrestricted access for RDP through port 3389 and, alternatively, HTTP through port 80. 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 Amazon EC2 Instance](#).

2. On your development machine, create a file named `instance-setup.sh` (for Amazon EC2 instances running Amazon Linux, Ubuntu Server, or RHEL) or `instance-setup.txt` (for Amazon EC2 instances running Windows Server) that contains the following contents.

As the Amazon EC2 instance is launched, this script will download the AWS CodeDeploy agent from the specified Amazon S3 location and then install it on the instance.

For Amazon Linux and RHEL

Here is the content of `instance-setup.sh` for Amazon Linux and RHEL:

```
#!/bin/bash
yum -y update
yum install -y ruby
cd /home/ec2-user
curl -O https://bucket-name.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

bucket-name is the name of the Amazon S3 `sds-s3-latest-bucket-name` bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region,

replace `bucket-name` with `aws-codedeploy-us-east-2`. For a list of bucket names, see [Resource Kit Bucket Names by Region \(p. 335\)](#).

For Ubuntu Server

Here is the content of `instance-setup.sh` for Ubuntu Server:

Important

If you are installing the AWS CodeDeploy agent on Ubuntu Server 14.04, change the third line of the file to the following:

```
apt-get -y install ruby2.0
```

```
#!/bin/bash
apt-get -y update
apt-get -y install ruby
apt-get -y install wget
cd /home/ubuntu
wget https://bucket-name.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```

`bucket-name` is the name of the Amazon S3 `sds-s3-latest-bucket-name` bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace `bucket-name` with `aws-codedeploy-us-east-2`. For a list of bucket names, see [Resource Kit Bucket Names by Region \(p. 335\)](#).

For Windows Server

Here is the content of `instance-setup.txt` for Windows Server:

```
<powershell>
New-Item -Path c:\temp -ItemType "directory" -Force
powershell.exe -Command 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>
```

`bucket-name` is the name of the Amazon S3 `sds-s3-latest-bucket-name` bucket that contains the AWS CodeDeploy Resource Kit files for your region. For example, for the US East (Ohio) Region, replace `bucket-name` with `aws-codedeploy-us-east-2`. For a list of bucket names, see [Resource Kit Bucket Names by Region \(p. 335\)](#).

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

Before you call this command, you will need to collect the following:

- The ID of an Amazon Machine Image (AMI) (`ami-id`) you will use for the instance. To get the ID, see [Finding a Suitable AMI](#).
- The name of the type of Amazon EC2 instance (`instance-type`) you will create, such as `t1.micro`. For a list, see [Amazon EC2 Instance Types](#).
- The name of an IAM instance profile with permission to access the Amazon S3 bucket where the AWS CodeDeploy agent installation files for your region are stored.

For information about creating an IAM instance profile, see [Step 4: Create an IAM Instance Profile for Your Amazon EC2 Instances \(p. 25\)](#).

- The name of an Amazon EC2 instance key pair (`key-name`) to enable SSH access to an Amazon EC2 instance running Amazon Linux, Ubuntu Server, or RHEL or RDP access to an Amazon EC2

instance running Windows Server

Important

Type the key pair name only, not the key pair file extension. For example, *my-keypair*, not *my-keypair.pem*.

To find a key pair name, open the Amazon EC2 console at <https://console.aws.amazon.com/ec2>. In the navigation pane, under **Network & Security**, choose **Key Pairs**, and note the key pair name in the list.

To generate a key pair, see [Creating Your Key Pair Using Amazon EC2](#). Be sure you create the key pair in one of the regions listed in [Region and Endpoints](#) in *AWS General Reference*. Otherwise, you won't be able to use the Amazon EC2 instance key pair with AWS CodeDeploy.

For Amazon Linux, RHEL, and Ubuntu Server

To call the **run-instances** command to launch an Amazon EC2 instance running Amazon Linux, Ubuntu Server, or RHEL and attach the IAM instance profile you created in [Step 4: Create an IAM Instance Profile for Your Amazon EC2 Instances \(p. 25\)](#). For example:

Important

Be sure to include `file://` before the file name. It is required in this command.

```
aws ec2 run-instances \
  --image-id ami-id \
  --key-name key-name \
  --user-data file://instance-setup.sh \
  --count 1 \
  --instance-type instance-type \
  --iam-instance-profile Name=iam-instance-profile
```

Note

This command creates a default security group for the Amazon EC2 instance that allows access to several ports, including unrestricted access for SSH through port 22 and, alternatively, HTTP through port 80. As a best practice, we recommend 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 Amazon EC2 Instance](#).

For Windows Server

To call the **run-instances** command to launch an Amazon EC2 instance running Windows Server and attach the IAM instance profile you created in [Step 4: Create an IAM Instance Profile for Your Amazon EC2 Instances \(p. 25\)](#), and specify the name of the security group you created in Step 1. For example:

Important

Be sure to include `file://` before the file name. It is required in this command.

```
aws ec2 run-instances --image-id ami-id --key-name key-name --user-data file://
instance-setup.txt --count 1 --instance-type instance-type --iam-instance-profile
Name=iam-instance-profile --security-groups CodeDeploy-Windows-Security-Group
```

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

4. Note the value of the `InstanceId` in the output. If you forget this value, you can get it later by calling the **describe-instances** command against the Amazon EC2 instance key pair.

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

Use the 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. In the following example, the tag is named **CodeDeployDemo**, but you can specify any Amazon EC2 instance tag you want.

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

You can apply multiple tags to an instance at the same time. For example:

```
aws ec2 create-tags --resources instance-id --tags Key=Name,Value=testInstance  
Key=Region,Value=West Key=Environment,Value=Beta
```

To verify the Amazon EC2 instance has been launched and passed all checks, use the instance ID to call the **describe-instance-status** command.

```
aws ec2 describe-instance-status --instance-ids instance-id --query  
"InstanceStatuses[*].InstanceStatus.[Status]" --output text
```

If the instance has been launched and passed all checks, `ok` will appear in the output:

To verify the AWS CodeDeploy agent is running on the instance, see [Verify the AWS CodeDeploy Agent Is Running \(p. 132\)](#), and then return to this page. After you do this, the Amazon EC2 instance will be ready for use in AWS CodeDeploy deployments.

Create an Amazon EC2 Instance for AWS CodeDeploy (AWS CloudFormation Template)

You can use our AWS CloudFormation template to quickly launch an Amazon EC2 instance running Amazon Linux or Windows Server. You can use the AWS CLI, the AWS CodeDeploy console, or the AWS APIs to launch the instance with the template. In addition to launching the instance, the template does the following:

- Instructs AWS CloudFormation to give the instance permission to participate in AWS CodeDeploy deployments.
- Tags the instance so AWS CodeDeploy can find it during a deployment.
- Installs and runs the AWS CodeDeploy agent on the instance.

You don't have to use our AWS CloudFormation to set up an Amazon EC2 instance. For alternatives, see [Working with Instances for AWS CodeDeploy \(p. 147\)](#).

We do not provide an AWS CloudFormation template for Amazon EC2 instances running Ubuntu Server or Red Hat Enterprise Linux (RHEL).

Important

If you use the AWS CloudFormation template to launch Amazon EC2 instances, the calling IAM user must have access to AWS CloudFormation and AWS services and actions on which AWS CloudFormation depends. If you have not followed the steps in [Step 1: Provision an IAM User \(p. 19\)](#) to provision the calling IAM user, you must at least attach the following policy:

```
{
```

```
"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": "*"
  }
]
```

Topics

- [Launch an Amazon EC2 Instance with the AWS CloudFormation Template \(Console\) \(p. 163\)](#)
- [Launch an Amazon EC2 Instance with the AWS CloudFormation Template \(AWS CLI\) \(p. 166\)](#)

Launch an Amazon EC2 Instance with the AWS CloudFormation Template (Console)

Before you begin, you must have an instance key pair to enable SSH access to the Amazon EC2 instance running Amazon Linux or RDP access to the instance running Windows Server. Type the key pair name only, not the key pair file extension.

To find a key pair name, open the Amazon EC2 console at <https://console.aws.amazon.com/ec2>. In the navigation pane, under **Network & Security**, choose **Key Pairs**, and note the key pair name in the list.

To generate a new key pair, see [Creating Your Key Pair Using Amazon EC2](#). Be sure the key pair is created in one of the regions listed in [Region and Endpoints](#) in *AWS General Reference*. Otherwise, you won't be able to use the 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

Sign in to the AWS Management Console with the same account you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#). On the navigation bar, in the region selector, choose one of the regions listed in [Region and Endpoints](#) in *AWS General Reference*. AWS CodeDeploy supports these regions only.

2. Choose **Create Stack**.
3. In **Choose a template**, choose **Specify an Amazon S3 template URL**. In the box, type the location of the AWS CloudFormation template for your region, and then choose **Next**.

Region	Location of AWS CloudFormation template
US East (Ohio) Region	http://s3-us-east-2.amazonaws.com/aws-codedeploy-us-

Region	Location of AWS CloudFormation template
	east-2/templates/latest/ CodeDeploy_SampleCF_Template.json
US East (N. Virginia) Region	http://s3.amazonaws.com/ aws-codedeploy-us- east-1/templates/latest/ CodeDeploy_SampleCF_Template.json
US West (N. California) Region	http://s3-us-west-1.amazonaws.com/ aws-codedeploy-us- west-1/templates/latest/ CodeDeploy_SampleCF_Template.json
US West (Oregon) Region	http://s3-us-west-2.amazonaws.com/ aws-codedeploy-us- west-2/templates/latest/ CodeDeploy_SampleCF_Template.json
Canada (Central) Region	http://s3-ca- central-1.amazonaws.com/ aws-codedeploy-ca- central-1/templates/latest/ CodeDeploy_SampleCF_Template.json
EU (Ireland) Region	http://s3-eu-west-1.amazonaws.com/ aws-codedeploy-eu- west-1/templates/latest/ CodeDeploy_SampleCF_Template.json
EU (London) Region	http://s3-eu-west-2.amazonaws.com/ aws-codedeploy-eu- west-2/templates/latest/ CodeDeploy_SampleCF_Template.json
EU (Paris) Region	http://s3-eu-west-3.amazonaws.com/ aws-codedeploy-eu- west-3/templates/latest/ CodeDeploy_SampleCF_Template.json
EU (Frankfurt) Region	http://s3-eu- central-1.amazonaws.com/ aws-codedeploy-eu- central-1/templates/latest/ CodeDeploy_SampleCF_Template.json
Asia Pacific (Tokyo) Region	http://s3-ap- northeast-1.amazonaws.com/ aws-codedeploy-ap- northeast-1/templates/latest/ CodeDeploy_SampleCF_Template.json
Asia Pacific (Seoul) Region	http://s3-ap- northeast-2.amazonaws.com/ aws-codedeploy-ap- northeast-2/templates/latest/ CodeDeploy_SampleCF_Template.json

Region	Location of AWS CloudFormation template
Asia Pacific (Singapore) Region	http://s3-ap-southeast-1.amazonaws.com/ aws-codedeploy-ap-southeast-1/templates/latest/ CodeDeploy_SampleCF_Template.json
Asia Pacific (Sydney) Region	http://s3-ap-southeast-2.amazonaws.com/ aws-codedeploy-ap-southeast-2/templates/latest/ CodeDeploy_SampleCF_Template.json
Asia Pacific (Mumbai) Region	http://s3-ap-south-1.amazonaws.com/ aws-codedeploy-ap-south-1/templates/latest/ CodeDeploy_SampleCF_Template.json
South America (São Paulo) Region	http://s3-sa-east-1.amazonaws.com/ aws-codedeploy-sa-east-1/templates/latest/ CodeDeploy_SampleCF_Template.json

4. In the **Stack name** box, type a name for the stack (for example, **CodeDeployDemoStack**).
5. In **Parameters**, type the following, and then choose **Next**.
 - For **InstanceCount**, type the number of instances you want to launch. (We recommend you leave the default of **1**.)
 - For **InstanceType**, type the instance type you want to launch (or leave the default of **t1.micro**).
 - For **KeyPairName**, type the instance key name.
 - For **OperatingSystem** box, type **windows** to launch instances running Windows Server (or leave the default of **Linux**).
 - For **SSHLocation**, type the IP address range to use for connecting to the instance with SSH or RDP (or leave the default of **0.0.0.0/0**).

Important

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

- For **TagKey**, type the instance tag key AWS CodeDeploy will use to identify the instances during deployment (or leave the default of **Name**).
 - For **TagValue**, type the instance tag value AWS CodeDeploy will use to identify the instances during deployment (or leave the default of **CodeDeployDemo**).
6. On the **Options** page, leave the option boxes blank, and choose **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 specified AWS CodeDeploy tags on the **Specify Parameters** page.

7. On the **Review** page, in **Capabilities**, select the **I acknowledge that AWS CloudFormation might create IAM resources** box, and then choose **Create**.

After AWS CloudFormation has created the stack and launched the Amazon EC2 instances, in the AWS CloudFormation console, **CREATE_COMPLETE** will be displayed in the **Status** column. This process can take several minutes.

To verify the AWS CodeDeploy agent is running on the Amazon EC2 instances, see [Managing AWS CodeDeploy Agent Operations](#) (p. 132), and then proceed to [Create an Application with AWS CodeDeploy](#) (p. 209).

Launch an Amazon EC2 Instance with the AWS CloudFormation Template (AWS CLI)

Follow the instructions in [Getting Started with AWS CodeDeploy](#) (p. 19) to install and configure the AWS CLI for use with AWS CodeDeploy.

Before you call the **create-stack** command, you must have an Amazon EC2 instance key pair to enable SSH access to the Amazon EC2 instance running Amazon Linux or RDP access to the Amazon EC2 instance running Windows Server. Type the key pair name only, not the key pair file extension.

To find a key pair name, open the Amazon EC2 console at <https://console.aws.amazon.com/ec2>. In the navigation pane, under **Network & Security**, choose **Key Pairs**, and note the key pair name in the list.

To generate a key pair, see [Creating Your Key Pair Using Amazon EC2](#). Be sure the key pair is created in one of the regions listed in [Region and Endpoints](#) in the *AWS General Reference*. Otherwise, you won't be able to use the instance key pair 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.

To launch an Amazon EC2 instance running Amazon Linux:

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

To launch an Amazon EC2 instance running Windows Server:

```
aws cloudformation create-stack --stack-name CodeDeployDemoStack --template-  
url template-url --parameters ParameterKey=InstanceCount,ParameterValue=1  
  ParameterKey=InstanceType,ParameterValue=t1.micro  
  ParameterKey=KeyPairName,ParameterValue=keyName  
  ParameterKey=OperatingSystem,ParameterValue=Windows  
  ParameterKey=SSHLocation,ParameterValue=0.0.0.0/0  
  ParameterKey=TagKey,ParameterValue=Name  
  ParameterKey=TagValue,ParameterValue=CodeDeployDemo --capabilities CAPABILITY_IAM
```

template-url is the location of the AWS CloudFormation template for your region:

Region	Location of AWS CloudFormation template
US East (Ohio) Region	http://s3-us-east-2.amazonaws.com/ aws-codedeploy-us- east-2/templates/latest/ CodeDeploy_SampleCF_Template.json
US East (N. Virginia) Region	http://s3.amazonaws.com/ aws-codedeploy-us- east-1/templates/latest/ CodeDeploy_SampleCF_Template.json
US West (N. California) Region	http://s3-us-west-1.amazonaws.com/ aws-codedeploy-us- west-1/templates/latest/ CodeDeploy_SampleCF_Template.json
US West (Oregon) Region	http://s3-us-west-2.amazonaws.com/ aws-codedeploy-us- west-2/templates/latest/ CodeDeploy_SampleCF_Template.json
Canada (Central) Region	http://s3-ca- central-1.amazonaws.com/ aws-codedeploy-ca- central-1/templates/latest/ CodeDeploy_SampleCF_Template.json
EU (Ireland) Region	http://s3-eu-west-1.amazonaws.com/ aws-codedeploy-eu- west-1/templates/latest/ CodeDeploy_SampleCF_Template.json
EU (London) Region	http://s3-eu-west-2.amazonaws.com/ aws-codedeploy-eu- west-2/templates/latest/ CodeDeploy_SampleCF_Template.json
EU (Paris) Region	http://s3-eu-west-3.amazonaws.com/ aws-codedeploy-eu- west-3/templates/latest/ CodeDeploy_SampleCF_Template.json
EU (Frankfurt) Region	http://s3-eu- central-1.amazonaws.com/ aws-codedeploy-eu- central-1/templates/latest/ CodeDeploy_SampleCF_Template.json
Asia Pacific (Tokyo) Region	http://s3-ap- northeast-1.amazonaws.com/ aws-codedeploy-ap- northeast-1/templates/latest/ CodeDeploy_SampleCF_Template.json

Region	Location of AWS CloudFormation template
Asia Pacific (Seoul) Region	<code>http://s3-ap-northeast-2.amazonaws.com/aws-codedeploy-ap-northeast-2/templates/latest/CodeDeploy_SampleCF_Template.json</code>
Asia Pacific (Singapore) Region	<code>http://s3-ap-southeast-1.amazonaws.com/aws-codedeploy-ap-southeast-1/templates/latest/CodeDeploy_SampleCF_Template.json</code>
Asia Pacific (Sydney) Region	<code>http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/templates/latest/CodeDeploy_SampleCF_Template.json</code>
Asia Pacific (Mumbai) Region	<code>http://s3-ap-south-1.amazonaws.com/aws-codedeploy-ap-south-1/templates/latest/CodeDeploy_SampleCF_Template.json</code>
South America (São Paulo) Region	<code>http://s3-sa-east-1.amazonaws.com/aws-codedeploy-sa-east-1/templates/latest/CodeDeploy_SampleCF_Template.json</code>

This command creates an AWS CloudFormation stack named **CodeDeployDemoStack**, using the AWS CloudFormation template in the specified Amazon S3 bucket. The Amazon EC2 instance is based on the t1.micro instance type, but you can use any type. It is tagged with the value **CodeDeployDemo**, but you can tag it with any value. It has the specified instance key pair applied.

2. Call the **describe-stacks** command to verify 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 the AWS CodeDeploy agent is running on the Amazon EC2 instance, see [Managing AWS CodeDeploy Agent Operations \(p. 132\)](#), and then proceed to [Create an Application with AWS CodeDeploy \(p. 209\)](#).

Configure an Amazon EC2 Instance to Work with AWS CodeDeploy

These instructions show you how to configure an Amazon EC2 instance running Amazon Linux, Ubuntu Server, Red Hat Enterprise Linux (RHEL), or Windows Server for use in AWS CodeDeploy deployments.

Note

If you do not have an Amazon EC2 instance, you can use the AWS CloudFormation template to launch one running Amazon Linux or Windows Server. We do not provide a template for Ubuntu Server or RHEL.

To perform the steps in this topic:

- An IAM instance profile with permissions to participate in AWS CodeDeploy deployments must be attached to your instance.

For information about how to attach an IAM instance profile when you create an Amazon EC2 instance, see [Create an Amazon EC2 Instance for AWS CodeDeploy \(AWS CLI or Amazon EC2 Console\)](#) (p. 156) and [Create an Amazon EC2 Instance for AWS CodeDeploy \(AWS CloudFormation Template\)](#) (p. 162).

For information about how to attach an IAM instance profile to an existing Amazon EC2 instance, see [Attaching an IAM Role to an Instance](#).

- Your Amazon EC2 instance must be tagged.
- The AWS CodeDeploy agent must be installed and running on the Amazon EC2 instance.

If the agent is not running, deployments will appear to be stalled in a pending state.

Step 1: Verify 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**, choose **Instances**.
3. Browse to and choose your Amazon EC2 instance in the list.
4. In the details pane, on the **Description** tab, note the value in the **IAM role** field, and then proceed to the next section.

If the field is empty, you can attach an IAM instance profile to the instance. For information, see [Attaching an IAM Role to an Instance](#).

Step 2: Verify 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, choose **Roles**.
3. Browse to and choose the IAM role name you noted in step 4 of the previous section.

Note

If you want to use the service role generated by the AWS CloudFormation template instead of one you created by following the instructions in [Step 3: Create a Service Role for AWS CodeDeploy](#) (p. 21), note the following:

In some versions of our AWS CloudFormation template, the display name of the IAM instance profile 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 might have a display name of `CodeDeploySampleStack-expnyi6-InstanceRoleInstanceProfile-IK8J8A9123EX`, while the IAM instance profile in the IAM console might have a display name of `CodeDeploySampleStack-expnyi6-InstanceRole-C5P33V1L64EX`.

To help you identify the instance profile in the IAM console, you'll see the prefix of `CodeDeploySampleStack-expny16-InstanceRole` is the same for both. For information about why these display names might be different, see [Instance Profiles](#).

4. Choose the **Trust Relationships** tab. If there is no entry in **Trusted Entities** that reads **The identity provider(s) ec2.amazonaws.com**, you cannot use this Amazon EC2 instance. Stop and create an Amazon EC2 instance using the information in [Working with Instances for AWS CodeDeploy \(p. 147\)](#).

If there is an entry that reads **The identity provider(s) ec2.amazonaws.com**, and you will be storing your applications in GitHub repositories only, then skip ahead to [Step 3: Tag the Amazon EC2 Instance \(p. 171\)](#).

If there is an entry that reads **The identity provider(s) ec2.amazonaws.com**, and you will be storing your applications in Amazon S3 buckets, choose the **Permissions** tab.

5. If there is a policy in the **Managed Policies** area, choose the policy's name, and then choose **Edit**. If there is a policy in **Inline Policies**, under **Actions**, choose **Edit Policy**.
6. If you will be storing your applications in Amazon S3 buckets, in the **Policy Document** box, make sure `"s3:Get*"` and `"s3:List*"` are in the list of specified actions.

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 in the list of specified actions, choose **Edit** to add them, and then choose **Save**. (If neither `"s3:Get*"` or `"s3:List*"` is the last action in the list, be sure to add a comma after the action, so the policy document will validate.)

Note

We recommend that you restrict this policy to only those Amazon S3 buckets your Amazon EC2 instances must access. Make sure to give access to the Amazon S3 buckets that contain the AWS CodeDeploy agent. Otherwise, an error may occur when the AWS CodeDeploy agent is installed or updated on the instances. To grant the IAM instance profile access to only some AWS CodeDeploy resource kit buckets in Amazon S3, use the following policy but remove the lines for buckets you want to prevent access to:

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

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:Get*",
      "s3:List*"
    ],
    "Resource": [
      "arn:aws:s3:::replace-with-your-s3-bucket-name/*",
      "arn:aws:s3:::aws-codedeploy-us-east-2/*",
      "arn:aws:s3:::aws-codedeploy-us-east-1/*",
      "arn:aws:s3:::aws-codedeploy-us-west-1/*",
      "arn:aws:s3:::aws-codedeploy-us-west-2/*",
      "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
      "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
      "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
      "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
      "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
      "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
      "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
      "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
      "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
      "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
      "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
    ]
  }
]
```

Step 3: Tag the Amazon EC2 Instance

For instructions about how to tag the Amazon EC2 instance so that AWS CodeDeploy can find it during a deployment, see [Working with Tags in the Console](#), and then return to this page.

Note

You can tag the Amazon EC2 instance with any key and value you like. Just make sure to specify this key and value when you deploy to it.

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

For instructions about how to install the AWS CodeDeploy agent on the Amazon EC2 instance and verify it is running, see [Managing AWS CodeDeploy Agent Operations \(p. 132\)](#), and then proceed to [Create an Application with AWS CodeDeploy \(p. 209\)](#).

Working with On-Premises Instances for AWS CodeDeploy

An on-premises instance is any physical device that is not an Amazon EC2 instance that can run the AWS CodeDeploy agent and connect to public AWS service endpoints.

Deploying an AWS CodeDeploy application revision to an on-premises instance involves two major steps:

- **Step 1** – Configure each on-premises instance, register it with AWS CodeDeploy, and then tag it.
- **Step 2** – Deploy application revisions to the on-premises instance.

Note

To experiment with creating and deploying a sample application revision to a correctly configured and registered on-premises instance, see [Tutorial: Deploy an Application to an On-Premises Instance with AWS CodeDeploy \(Windows Server, Ubuntu Server, or Red Hat Enterprise Linux\)](#) (p. 90). For information about on-premises instances and how they work with AWS CodeDeploy, see [Working with On-Premises Instances](#) (p. 171).

If you don't want an on-premises instance to be used in deployments anymore, you can simply remove the on-premises instance tags from the deployment groups. For a more robust approach, remove the on-premises instance tags from the instance. You can also explicitly deregister an on-premises instance so it can no longer be used in any deployments. For more information, see [Managing On-Premises Instances Operations in AWS CodeDeploy](#) (p. 193).

The instructions in this section show you how to configure an on-premises instance and then register and tag it with AWS CodeDeploy so it can be used in deployments. This section also describes how to use AWS CodeDeploy to get information about on-premises instances and deregister an on-premises instance after you're no longer planning to deploy to it.

Topics

- [Prerequisites for Configuring an On-Premises Instance](#) (p. 172)
- [Register an On-Premises Instance with AWS CodeDeploy](#) (p. 173)
- [Managing On-Premises Instances Operations in AWS CodeDeploy](#) (p. 193)

Prerequisites for Configuring an On-Premises Instance

The following prerequisites must be met before you can register an on-premises instance.

Important

If you are using the [register-on-premises-instance](#) command and periodically refreshed temporary credentials generated with the AWS Security Token Service (AWS STS), there are other prerequisites. For information, see [IAM Session ARN Registration Prerequisites](#) (p. 187).

Device requirements

The device you want to prepare, register, and tag as an on-premises instance with AWS CodeDeploy must be running a supported operating system. For a list, see [Operating Systems Supported by the AWS CodeDeploy Agent](#) (p. 125).

If your operating system is not supported, 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.

Outbound communication

The on-premises instance must be able to connect to public AWS service endpoints to communicate with AWS CodeDeploy.

The AWS CodeDeploy agent communicates outbound using HTTPS over port 443.

Administrative control

The local or network account 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).

IAM permissions

The IAM identity you use to register the on-premises instance must be granted permissions to complete the registration (and to deregister the on-premises instance, as needed).

In addition to the policy described in [Getting Started with AWS CodeDeploy \(p. 19\)](#), make sure the calling IAM identity 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": "*"
    }
  ]
}
```

Register an On-Premises Instance with AWS CodeDeploy

To register an on-premises instance, you must use an IAM identity to authenticate your requests. You can choose from the following options for the IAM identity and registration method you use:

- Use an IAM user ARN to authenticate requests.
 - Use the [register](#) command for the most automated registration process. Best for registering a single on-premises instance. For information, see [Use the register Command \(IAM User ARN\) to Register an On-Premises Instance \(p. 174\)](#).
 - Use the [register-on-premises-instance](#) command to manually configure most registration options. Suitable for registering a small number of on-premises instances. For information, see [Use the register-on-premises-instance Command \(IAM User ARN\) to Register an On-Premises Instance \(p. 178\)](#).
- Use an IAM role ARN to authenticate requests.
 - Use the [register-on-premises-instance](#) command and periodically refreshed temporary credentials generated with the AWS Security Token Service (AWS STS) to manually configure most registration options. Best for registering a large number of on-premises instances. For information, see [Use the register-on-premises-instance Command \(IAM Session ARN\) to Register an On-Premises Instance \(p. 186\)](#).

Topics

- [Use the register Command \(IAM User ARN\) to Register an On-Premises Instance \(p. 174\)](#)
- [Use the register-on-premises-instance Command \(IAM User ARN\) to Register an On-Premises Instance \(p. 178\)](#)
- [Use the register-on-premises-instance Command \(IAM Session ARN\) to Register an On-Premises Instance \(p. 186\)](#)

Use the register Command (IAM User ARN) to Register an On-Premises Instance

This section describes how to configure an on-premises instance and register and tag it with AWS CodeDeploy with the least amount of effort. The **register** command is most useful when you are working with single or small fleets of on-premises instances. You can use the **register** command only when you are using an IAM user ARN to authenticate an instance. You cannot use the **register** command with an IAM session ARN for authentication.

When you use the **register** command, you can let AWS CodeDeploy do the following:

- Create an IAM user in AWS Identity and Access Management for the on-premises instance, if you do not specify one with the command.
- Save the IAM user's credentials to an on-premises instance configuration file.
- Register the on-premises instance with AWS CodeDeploy.
- Add tags to the on-premises instance, if you specify them as part of the command.

Note

The [register-on-premises-instance](#) command is an alternative to the [register](#) command. You use the **register-on-premises-instance** command if you want to configure an on-premises instance and register and tag it with AWS CodeDeploy mostly on your own. The **register-on-premises-instance** command also gives you the option to use an IAM session ARN to register instances instead of an IAM user ARN. This approach provides a major advantage if you have large fleets of on-premises instances. Specifically, you can use a single IAM session ARN to authenticate multiple instances instead of having to create an IAM user for each on-premises instance one by one. For more information, see [Use the register-on-premises-instance Command \(IAM User ARN\) to Register an On-Premises Instance](#) (p. 178) and [Use the register-on-premises-instance Command \(IAM Session ARN\) to Register an On-Premises Instance](#) (p. 186).

Topics

- [Step 1: Install and Configure the AWS CLI on the On-Premises Instance](#) (p. 174)
- [Step 2: Call the register Command](#) (p. 175)
- [Step 3: Call the install Command](#) (p. 176)
- [Step 4: Deploy Application Revisions to the On-Premises Instance](#) (p. 177)
- [Step 5: Track Deployments to the On-Premises Instance](#) (p. 178)

Step 1: Install and Configure the AWS CLI on the On-Premises Instance

1. Install the AWS CLI on the on-premises instance. 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 are available in AWS CLI version 1.7.19 and later. If you have the AWS CLI already installed, call **aws --version** to check its version.

2. Configure the AWS CLI on the on-premises instance. Follow the instructions in [Configuring the AWS Command Line Interface](#) in *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 who has, at minimum, the following AWS access permissions in addition to the permissions specified in [Prerequisites for Configuring an On-Premises Instance](#) (p. 172). This makes it possible to download and

install the AWS CodeDeploy agent on the on-premises instance. The access permissions might 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",
        "tag:GetTags",
        "tag:GetResources"
      ],
      "Resource" : "*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource" : [
        "arn:aws:s3:::aws-codedeploy-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
      ]
    }
  ]
}
```

Step 2: Call the register Command

For this step, we assume 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 preceding 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 use a name that maps to some unique characteristic of the on-premises instance (for example, the serial number or some unique internal asset identifier, if applicable). If you specify a MAC address for a name, be aware that MAC addresses contain characters that AWS CodeDeploy does not allow, such as colon (:). For a list of allowed characters, see [AWS CodeDeploy Limits \(p. 340\)](#).

- 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 IAM user, call the [get-user](#) command, or choose the IAM user name in the **Users** section of the IAM console and then find the **User ARN** value in the **Summary** section. If this option is not specified, AWS CodeDeploy will create an IAM user on your behalf in 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 on-premises instance configuration file, as described in [Step 4: Add a Configuration File to the On-Premises Instance \(p. 182\)](#).

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 (with the `--tags` option) that AWS CodeDeploy will use to identify the set of Amazon EC2 instances to which to deploy. 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. 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 listed in [Region and Endpoints](#) in *AWS General Reference* (for example, `us-west-2`). If this option is not specified, the default AWS region 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 an IAM user, attaches the required 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 AWS CodeDeploy the tags that are specified with the `--tags` option with the registered on-premises instance name.
4. If an IAM user was created, also creates the required configuration file in the same directory from which the **register** command was called.

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 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 listed in [Region and Endpoints](#) in *AWS General Reference* (for example, `us-west-2`). If this option is not specified, the default AWS region associated with the calling IAM user will be used.
- Optionally, a custom location from which to install the AWS CodeDeploy agent (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).
 - 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 operating system on 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. Copies the on-premises instances configuration file from the specified location on the instance to the location where the AWS CodeDeploy agent expects to find it, provided that the file is not already in that location.

For Ubuntu Server and Red Hat Enterprise Linux (RHEL)), this is `/etc/codedeploy-agent/conf/codedeploy.onpremises.yml`.

For Windows Server, this is `C:\ProgramData\Amazon\CodeDeploy\conf.onpremises.yml`.

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 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 in a way that's similar to deploying application revisions to Amazon EC2 instances. For instructions, see [Create a Deployment with AWS CodeDeploy](#) (p. 245). 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 described in [Step 2: Create a Sample Application Revision](#) (p. 91) in the [Tutorial: Deploy an Application to an On-Premises Instance with AWS CodeDeploy](#) (Windows Server, Ubuntu Server, or Red Hat Enterprise Linux) (p. 90).

Important

If you 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 [Step 3: Create a Service Role for AWS CodeDeploy \(p. 21\)](#).

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 in a way that's similar to tracking deployments to Amazon EC2 instances. For instructions, see [View Deployment Details with AWS CodeDeploy \(p. 254\)](#).

For more options, see [Managing On-Premises Instances Operations in AWS CodeDeploy \(p. 193\)](#).

Use the register-on-premises-instance Command (IAM User ARN) to Register an On-Premises Instance

Follow these instructions to configure an on-premises instance and register and tag it with AWS CodeDeploy mostly on your own, using static IAM user credentials for authentication.

Topics

- [Step 1: Create an IAM User for the On-Premises Instance \(p. 178\)](#)
- [Step 2: Assign Permissions to the IAM User \(p. 179\)](#)
- [Step 3: Get the IAM User Credentials \(p. 181\)](#)
- [Step 4: Add a Configuration File to the On-Premises Instance \(p. 182\)](#)
- [Step 5: Install and Configure the AWS CLI \(p. 183\)](#)
- [Step 6: Set the AWS_REGION Environment Variable \(Ubuntu Server and RHEL Only\) \(p. 184\)](#)
- [Step 7: Install the AWS CodeDeploy Agent \(p. 184\)](#)
- [Step 8: Register the On-Premises Instance with AWS CodeDeploy \(p. 184\)](#)
- [Step 9: Tag the On-Premises Instance \(p. 185\)](#)
- [Step 10: Deploy Application Revisions to the On-Premises Instance \(p. 186\)](#)
- [Step 11: Track Deployments to the On-Premises Instance \(p. 186\)](#)

Step 1: Create an IAM User for the On-Premises Instance

Create an IAM user that the on-premises instance will use to authenticate and interact with AWS CodeDeploy.

Important

You must create a separate IAM user for each participating on-premises instance. If you try to reuse an individual IAM user for multiple on-premises instances, you might not be able to successfully register or tag those on-premises instances with AWS CodeDeploy. Deployments to those on-premises instances might be stuck in a perpetual pending state or fail altogether.

We recommend that you assign the IAM user a name that identifies its purpose, such as `CodeDeployUser-OnPrem`.

You can use the AWS CLI or the IAM console to create an IAM user. For information, see [Creating an IAM User in Your AWS Account](#).

Important

Whether you use the AWS CLI or the IAM console to create a new IAM user, make a note of the user ARN provided for the user. You will need this information later in [Step 4: Add a Configuration File to the On-Premises Instance](#) (p. 182) and [Step 8: Register the On-Premises Instance with AWS CodeDeploy](#) (p. 184).

Step 2: Assign 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 permissions to interact with those buckets. You can use the AWS CLI or the IAM console to assign permissions.

Note

If you will be deploying application revisions only from GitHub repositories, skip this step and go directly to [Step 3: Get the IAM User Credentials](#) (p. 181). (You will still need information about the IAM user you created in [Step 1: Create an IAM User for the On-Premises Instance](#) (p. 178). It will be used in later steps.)

To assign permissions (CLI)

1. Create a file with the following policy contents on the Amazon EC2 instance or device you are using to call the AWS CLI. Name the file something like **CodeDeploy-OnPrem-Permissions.json**, 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 those Amazon S3 buckets your on-premises instance needs to access. If you restrict this policy, make sure to also give access to the Amazon S3 buckets that contain the AWS CodeDeploy agent. Otherwise, an error might 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:::replace-with-your-s3-bucket-name/*",
        "arn:aws:s3:::aws-codedeploy-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*"
      ]
    }
  ]
}
```

```
"arn:aws:s3::aws-codedeploy-ca-central-1/*",
"arn:aws:s3::aws-codedeploy-eu-west-1/*",
"arn:aws:s3::aws-codedeploy-eu-west-2/*",
"arn:aws:s3::aws-codedeploy-eu-west-3/*",
"arn:aws:s3::aws-codedeploy-eu-central-1/*",
"arn:aws:s3::aws-codedeploy-ap-northeast-1/*",
"arn:aws:s3::aws-codedeploy-ap-northeast-2/*",
"arn:aws:s3::aws-codedeploy-ap-southeast-1/*",
"arn:aws:s3::aws-codedeploy-ap-southeast-2/*",
"arn:aws:s3::aws-codedeploy-ap-south-1/*",
"arn:aws:s3::aws-codedeploy-sa-east-1/*"
    ]
  }
}
```

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) from which you're calling this command:

Important

Be sure to include `file://` before the file name. It is required in this command.

```
aws iam put-user-policy --user-name CodeDeployUser-OnPrem --policy-name CodeDeploy-
OnPrem-Permissions --policy-document file://CodeDeploy-OnPrem-Permissions.json
```

To assign permissions (console)

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Policies**, and then choose **Create Policy**. (If a **Get Started** button appears, choose it, and then choose **Create Policy**.)
3. Next to **Create Your Own Policy**, choose **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 Amazon S3 bucket specified in the policy 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. Choose **Create Policy**.
7. In the navigation pane, choose **Users**.
8. In the list of users, browse to and choose the name of the IAM user you created in [Step 1: Create an IAM User for the On-Premises Instance \(p. 178\)](#).

9. On the **Permissions** tab, in **Managed Policies**, choose **Attach Policy**.
10. Select the policy named **CodeDeploy-OnPrem-Permissions**, and then choose **Attach Policy**.

Step 3: Get the IAM User Credentials

Get the secret key ID and the secret access key for the IAM user. You will need them for [Step 4: Add a Configuration File to the On-Premises Instance \(p. 182\)](#). You can use the AWS CLI or the IAM console to get the secret key ID and the secret access key.

Note

If you already have the secret key ID and the secret access key, skip this step and go directly to [Step 4: Add a Configuration File to the On-Premises Instance \(p. 182\)](#).

To get the credentials (CLI)

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  
"AccessKeyMetadata[*].AccessKeyId" --output text
```

2. If no keys appear in the output or information about only one key appears in the output, call the `create-access-key` command, specifying the name of the IAM user (with the `--user-name` option):

```
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. You will need this information in [Step 4: Add a Configuration File to the On-Premises Instance \(p. 182\)](#).

Important

This will be the only time you will have access to this secret access key. If you forget or lose access to this secret access key, you will need to generate a new one by following the steps in [Step 3: Get the IAM User Credentials \(p. 181\)](#).

3. If two access keys are already listed, you must delete 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: Add a Configuration File to the On-Premises Instance \(p. 182\)](#), you will need to follow the instructions in [Step 4: Add a Configuration File to the On-Premises Instance \(p. 182\)](#) again to specify a different access key ID and secret access key associated with this IAM user. Otherwise, any deployments to that on-premises instance might be stuck in a perpetual pending state or fail altogether.

To get the credentials (console)

1. 1. Open the IAM console at <https://console.aws.amazon.com/iam/>.

2. If the list of users is not displayed, in the navigation pane, choose **Users**.
3. In the list of users, browse to and choose the name of the IAM user you created in [Step 1: Create an IAM User for the On-Premises Instance \(p. 178\)](#).
2. On the **Security credentials** tab, if no keys or only one key is listed, choose **Create access key**.

If two access keys are listed, you must delete one of them. Choose **Delete** next to one of the access keys, and then choose **Create access key**.

Important

If you choose **Delete** next to one of these access keys, and an on-premises instance is already using this access key as described in [Step 4: Add a Configuration File to the On-Premises Instance \(p. 182\)](#), you will need to follow the instructions in [Step 4: Add a Configuration File to the On-Premises Instance \(p. 182\)](#) again to specify a different access key ID and secret access key associated with this IAM user. Otherwise, deployments to that on-premises instance might be stuck in a perpetual pending state or fail altogether.

3. Choose **Show** and note the access key ID and secret access key. You will need this information for the next step. Alternatively, you can choose **Download .csv file** to save a copy of the access key ID and the secret access key.

Important

Unless you make a note of or download the credentials, this will be the only time you will have access to this secret access key. If you forget or lose access to this secret access key, you will need to generate a new one by following the steps in [Step 3: Get the IAM User Credentials \(p. 181\)](#).

4. Choose **Close** to return to the **Users > 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. The file must follow a specific format.

1. Create a file named `codedeploy.onpremises.yml` (for an Ubuntu Server or RHEL 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 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 you noted in [Step 1: Create an IAM User for the On-Premises Instance \(p. 178\)](#) or [Step 3: Get the IAM User Credentials \(p. 181\)](#).
- *secret-access-key* is the corresponding IAM user's secret access key you noted in [Step 1: Create an IAM User for the On-Premises Instance \(p. 178\)](#) or [Step 3: Get the IAM User Credentials \(p. 181\)](#).

- `iam-user-arn` is the IAM user's ARN you noted earlier in [Step 1: Create an IAM User for the On-Premises Instance](#) (p. 178).
- `supported-region` is the identifier of a region supported by AWS CodeDeploy where your AWS CodeDeploy applications, deployment groups, and application revisions are located (for example, `us-west-2`). For a list of regions, see [Region and Endpoints](#) in the *AWS General Reference*.

Important

If you chose **Delete** next to one of the access keys in [Step 3: Get the IAM User Credentials](#) (p. 181), 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 [Step 4: Add a Configuration File to the On-Premises Instance](#) (p. 182) to specify a different access key ID and secret access key associated with this IAM user. Otherwise, any deployments to your on-premises instance might be stuck in a perpetual pending state or fail altogether.

Step 5: Install and Configure the AWS CLI

Install and configure the AWS CLI on the on-premises instance. (The AWS CLI will be used in [Step 7: Install the AWS CodeDeploy Agent](#) (p. 184) to download and install the AWS CodeDeploy agent on the on-premises instance.)

1. To install the AWS CLI on the on-premises instance, 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 in 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. To configure the AWS CLI on the on-premises instance, follow 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 specified in the [Prerequisites for Configuring an On-Premises Instance](#) (p. 172). This makes it possible 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-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
```

```
"arn:aws:s3::aws-codedeploy-ca-central-1/*",
"arn:aws:s3::aws-codedeploy-eu-west-1/*",
"arn:aws:s3::aws-codedeploy-eu-west-2/*",
"arn:aws:s3::aws-codedeploy-eu-west-3/*",
"arn:aws:s3::aws-codedeploy-eu-central-1/*",
"arn:aws:s3::aws-codedeploy-ap-northeast-1/*",
"arn:aws:s3::aws-codedeploy-ap-northeast-2/*",
"arn:aws:s3::aws-codedeploy-ap-southeast-1/*",
"arn:aws:s3::aws-codedeploy-ap-southeast-2/*",
"arn:aws:s3::aws-codedeploy-ap-south-1/*",
"arn:aws:s3::aws-codedeploy-sa-east-1/*"
    ]
  }
}
```

These access permissions can be assigned to either the IAM user you created in [Step 1: Create an IAM User for the On-Premises Instance](#) (p. 178) or to a different IAM user. To assign these permissions to an IAM user, follow the instructions in [Step 1: Create an IAM User for the On-Premises Instance](#) (p. 178), using these access permissions instead of the ones in that step.

Step 6: Set the AWS_REGION Environment Variable (Ubuntu Server and RHEL Only)

If you are not running Ubuntu Server or RHEL on your on-premises instance, skip this step and go directly to [Step 7: Install the AWS CodeDeploy Agent](#) (p. 184).

Install the AWS CodeDeploy agent on an Ubuntu Server or RHEL on-premises instance and enable the instance to update the AWS CodeDeploy agent whenever a new version becomes available. You do this by setting the AWS_REGION environment variable on the instance to the identifier of one of the regions supported by AWS CodeDeploy. We recommend that you set the value to the region where your AWS CodeDeploy applications, deployment groups, and application revisions are located (for example, us-west-2). For a list of regions, see [Region and Endpoints](#) in the *AWS General Reference*.

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:

- For an Ubuntu Server on-premises instance, follow the instructions in [Install or reinstall the AWS CodeDeploy agent for Ubuntu Server](#) (p. 136), and then return to this page.
- For a RHEL on-premises instance, follow the instructions in [Install or reinstall the AWS CodeDeploy agent for Amazon Linux or RHEL](#) (p. 135), and then return to this page.
- For a Windows Server on-premises instance, follow the instructions in [Install or reinstall the AWS CodeDeploy agent for Windows Server](#) (p. 137), and then return to this page.

Step 8: Register the On-Premises Instance with AWS CodeDeploy

The instructions in this step assume you are registering the on-premises instance from the on-premises instance itself. You can register an on-premises instance from a separate device or instance that has the AWS CLI installed and configured, as described in [Step 5: Install and Configure the AWS CLI](#) (p. 183).

Use the AWS CLI to register the on-premises instance with AWS CodeDeploy so that it can be used in deployments.

1. Before you can use the AWS CLI, you will need the user ARN of the IAM user you created in [Step 1: Create an IAM User for the On-Premises Instance \(p. 178\)](#). 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):

```
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 (with the `--instance-name` option).

Important

To help identify the on-premises instance, especially for debugging purposes, we strongly recommend that you specify a name that maps to some unique characteristic of the on-premises instance (for example, the serial number or an internal asset identifier, if applicable). If you specify a MAC address as a name, be aware that MAC addresses contain characters that AWS CodeDeploy does not allow, such as colon (:). For a list of allowed characters, see [AWS CodeDeploy Limits \(p. 340\)](#).

- The user ARN of the IAM user you created in [Step 1: Create an IAM User for the On-Premises Instance \(p. 178\)](#) (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 deployment targets during a deployment.)

To tag the on-premises instance (CLI)

- 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 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 values only.

For example:

```
aws deploy add-tags-to-on-premises-instances --instance-names AssetTag12010298EX --tags Key=Name,Value=CodeDeployDemo-OnPrem
```

To tag the on-premises instance (console)

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#).

2. From the AWS CodeDeploy menu, choose **On-premises instances**.
3. In the list of on-premises instances, choose the arrow next to the on-premises instance 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, choose the delete icon (✕).
5. After you have added tags, choose **Update Tags**.

Step 10: Deploy 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 in a way that's similar to deploying application revisions to Amazon EC2 instances. For instructions, see [Create a Deployment with AWS CodeDeploy \(p. 245\)](#). These instructions include a 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 described in [Step 2: Create a Sample Application Revision \(p. 91\)](#) in the [Tutorial: Deploy an Application to an On-Premises Instance with AWS CodeDeploy \(Windows Server, Ubuntu Server, or Red Hat Enterprise Linux\) \(p. 90\)](#).

Important

If you reuse an 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 [Step 3: Create a Service Role for AWS CodeDeploy \(p. 21\)](#).

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 in a way that's similar to tracking deployments to Amazon EC2 instances. For instructions, see [View Deployment Details with AWS CodeDeploy \(p. 254\)](#).

Use the register-on-premises-instance Command (IAM Session ARN) to Register an On-Premises Instance

For maximum control over the authentication and registration of your on-premises instances, you can use the [register-on-premises-instance](#) command and periodically refreshed temporary credentials generated with the AWS Security Token Service (AWS STS). A static IAM role for the instance assumes the role of these refreshed AWS STS credentials to perform AWS CodeDeploy deployment operations.

This method is most useful when you need to register a large number of instances. It allows you to automate the registration process with AWS CodeDeploy. You can use your own identity and authentication system to authenticate on-premises instances and distribute IAM session credentials from the service to the instances for use with AWS CodeDeploy.

Note

Alternatively, you can use a shared IAM user distributed to all on-premises instances to call the AWS STS [AssumeRole](#) API to retrieve session credentials for on-premises instances. This method is less secure and not recommended for production or mission-critical environments.

Use the information in the following topics to configure an on-premises instance using temporary security credentials generated with AWS STS.

Topics

- [IAM Session ARN Registration Prerequisites \(p. 187\)](#)
- [Step 1: Create the IAM Role that On-Premises Instances Will Assume \(p. 187\)](#)
- [Step 2: Generate Temporary Credentials for an Individual Instance Using AWS STS \(p. 188\)](#)
- [Step 3: Add a Configuration File to the On-Premises Instance \(p. 189\)](#)
- [Step 4: Prepare an On-Premises Instance for AWS CodeDeploy Deployments \(p. 190\)](#)
- [Step 5: Register the On-Premises Instance with AWS CodeDeploy \(p. 191\)](#)
- [Step 6: Tag the On-Premises Instance \(p. 191\)](#)
- [Step 7: Deploy Application Revisions to the On-Premises Instance \(p. 192\)](#)
- [Step 8: Track Deployments to the On-Premises Instance \(p. 192\)](#)

IAM Session ARN Registration Prerequisites

In addition to the prerequisites listed in [Prerequisites for Configuring an On-Premises Instance \(p. 172\)](#), the following additional requirements must be met:

IAM permissions

The IAM identity you use to register an on-premises instance must be granted permissions to perform AWS CodeDeploy operations. Make sure the **AWSCodeDeployFullAccess** managed policy is attached to the IAM identity. For information, see [AWS Managed Policies](#) in the *IAM User Guide*.

System to refresh temporary credentials

If you use an IAM session ARN to register on-premises instances, you must have a system in place to periodically refresh the temporary credentials. Temporary credentials expire after one hour or sooner if a shorter period is specified when the credentials are generated. There are two methods for refreshing the credentials:

- **Method 1:** Use the identity and authentication system in place in your corporate network with a CRON script that periodically polls the identity and authentication system and copies the latest session credentials to the instance. This enables you to integrate your authentication and identity structure with AWS without needing to make changes to the AWS CodeDeploy agent or service to support authentication types you use in your organization.
- **Method 2:** Periodically run a CRON job on the instance to call the AWS STS [AssumeRole](#) action and write the session credentials to a file that the AWS CodeDeploy agent can access. This method still requires using an IAM user and copying credentials to the on-premises instance, but you can re-use the same IAM user and credentials across your fleet of on-premises instances.

For information about creating and working with AWS STS credentials, see [AWS Security Token Service API Reference](#) and [Using Temporary Security Credentials to Request Access to AWS Resources](#).

Step 1: Create the IAM Role that On-Premises Instances Will Assume

You can use the AWS CLI or the IAM console to create an IAM role that will be used by your on-premises instances to authenticate and interact with AWS CodeDeploy.

You only need to create a single IAM role. Each one of your on-premises instances can assume this role to retrieve the temporary security credentials that provide the permissions granted to this role.

The role you create will require the following permissions to access the files required to install the AWS CodeDeploy agent:

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

We recommend that you restrict this policy to only those Amazon S3 buckets your on-premises instance needs to access. If you restrict this policy, make sure to give access to the Amazon S3 buckets that contain the AWS CodeDeploy agent. Otherwise, an error might occur whenever the AWS CodeDeploy agent is installed or updated on the on-premises instance. For information about controlling access to Amazon S3 buckets, see [Managing Access Permissions to Your Amazon S3 Resources](#).

To create the IAM role

1. Call the [create-role](#) command using the `--role-name` option to specify a name for the IAM role (for example, `CodeDeployInstanceRole`) and the `--assume-role-policy-document` option to provide the permissions.

When you create the IAM role for this instance, you might give it the name `CodeDeployInstanceRole` and include the required permissions in a file named `CodeDeployRolePolicy.json`:

```
aws iam create-role --role-name CodeDeployInstanceRole --assume-role-policy-document
file://CodeDeployRolePolicy.json
```

2. In the output of the call to the **create-role** command, note the value of the ARN field. For example:

```
arn:aws:iam::123456789012:role/CodeDeployInstanceRole
```

You will need the role ARN when you use the AWS STS [AssumeRole](#) API to generate short-term credentials for each instance.

For more information about creating IAM roles, see [Creating a Role to Delegate Permissions to an AWS Service](#) in *IAM User Guide*.

For information about assigning permissions to an existing role, see [put-role-policy](#) in [AWS CLI Command Reference](#).

Step 2: Generate Temporary Credentials for an Individual Instance Using AWS STS

Before you generate the temporary credentials that will be used for registering an on-premises instance, you must create or choose the IAM identity (user or role) that you will generate the temporary credentials for. The `sts:AssumeRole` permission must be included in the policy settings for this IAM identity.

For information about granting `sts:AssumeRole` permissions to an IAM identity, see [Creating a Role to Delegate Permissions to an AWS Service](#) and [AssumeRole](#).

There are two ways to generate the temporary credentials:

- Use the [assume-role](#) command with the AWS CLI. For example:

```
aws sts assume-role --role-arn arn:aws:iam::12345ACCOUNT:role/role-arn --role-session-name session-name
```

Where:

- **12345ACCOUNT** is the 12-digit account number for your organization.
- *role-arn* is the ARN of the role to be assumed, which you generated in [Step 1: Create the IAM Role that On-Premises Instances Will Assume \(p. 187\)](#).
- *session-name* is the name you want to give to the role session you are creating now.

Note

If you use a CRON script that periodically polls the identity and authentication system and copies the latest session credentials to the instance (method 1 for refreshing temporary credentials described in [IAM Session ARN Registration Prerequisites \(p. 187\)](#)), you can instead use any supported AWS SDK to call [AssumeRole](#).

- Use a tool provided by AWS.

The `aws-codedeploy-session-helper` tool generates AWS STS credentials and writes them to a file you place on the instance. This tool is best suited to method 2 for refreshing temporary credentials described in [IAM Session ARN Registration Prerequisites \(p. 187\)](#). In this method, the `aws-codedeploy-session-helper` tool is placed on each instance and executes the command using an IAM user's permissions. Each instance uses the same IAM user's credentials in conjunction with this tool.

For more information, see the [aws-codedeploy-session-helper](#) GitHub repository.

Note

After you have created the IAM session credentials, place them in any location on the on-premises instance. In the next step, you will configure the AWS CodeDeploy agent to access the credentials in this location.

Before continuing, make sure the system you will use to periodically refresh the temporary credentials is in place. If the temporary credentials are not refreshed, deployments to the on-premises instance will fail. For more information, see "System to refresh temporary credentials" in [IAM Session ARN Registration Prerequisites \(p. 187\)](#).

Step 3: 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 is used to declare the IAM 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 temporary session ARN, its secret key ID and secret access key, and the target AWS region.

To add a configuration file

1. Create a file named `codedeploy.onpremises.yml` (for an Ubuntu Server or RHEL 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 to add the following information to the newly created `codedeploy.onpremises.yml` or `conf.onpremises.yml` file:

```
---
```

```
iam_session_arn: iam-session-arn  
aws_credentials_file: credentials-file  
region: supported-region
```

Where:

- *iam-session-arn* is the IAM session ARN you noted in [Step 2: Generate Temporary Credentials for an Individual Instance Using AWS STS \(p. 188\)](#).
- *credentials-file* is the location of the credentials file for the temporary session ARN, as noted in [Step 2: Generate Temporary Credentials for an Individual Instance Using AWS STS \(p. 188\)](#).
- *supported-region* is one of the regions that AWS CodeDeploy supports, as listed in [Region and Endpoints](#) in *AWS General Reference*.

Step 4: Prepare an On-Premises Instance for AWS CodeDeploy Deployments

Install and configure the AWS CLI

Install and configure the AWS CLI on the on-premises instance. (The AWS CLI will be used to download and install the AWS CodeDeploy agent on the on-premises instance.)

1. To install the AWS CLI on the on-premises instance, 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 in 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. To configure the AWS CLI on the on-premises instance, follow 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 permissions described in [IAM Session ARN Registration Prerequisites \(p. 187\)](#).

Set the AWS_REGION Environment Variable (Ubuntu Server and RHEL Only)

If you are not running Ubuntu Server or RHEL on your on-premises instance, skip this step and go directly to "Install the AWS CodeDeploy Agent."

Install the AWS CodeDeploy agent on an Ubuntu Server or RHEL on-premises instance and enable instance to update the AWS CodeDeploy agent whenever a new version becomes available. You do this by setting the `AWS_REGION` environment variable on the instance to the identifier of one of the regions supported by AWS CodeDeploy. We recommend that you set the value to the region where your AWS CodeDeploy applications, deployment groups, and application revisions are located (for example, `us-west-2`). For a list of regions, see [Region and Endpoints](#) in the *AWS General Reference*.

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`).

Install the AWS CodeDeploy Agent

- For an Ubuntu Server on-premises instance, follow the instructions in [Install or reinstall the AWS CodeDeploy agent for Ubuntu Server \(p. 136\)](#), and then return to this page.

- For a RHEL on-premises instance, follow the instructions in [Install or reinstall the AWS CodeDeploy agent for Amazon Linux or RHEL \(p. 135\)](#), and then return to this page.
- For a Windows Server on-premises instance, follow the instructions in [Install or reinstall the AWS CodeDeploy agent for Windows Server \(p. 137\)](#), and then return to this page.

Step 5: Register the On-Premises Instance with AWS CodeDeploy

The instructions in this step assume you are registering the on-premises instance from the on-premises instance itself. You can register an on-premises instance from a separate device or instance that has the AWS CLI installed and configured.

Use the AWS CLI to register the on-premises instance with AWS CodeDeploy so that it can be used in deployments.

Before you can use the AWS CLI, you will need the ARN of the temporary session credentials you created in [Step 3: Add a Configuration File to the On-Premises Instance \(p. 189\)](#). For example, for an instance you identify as `AssetTag12010298EX`:

```
arn:sts:iam::123456789012:assumed-role/CodeDeployInstanceRole/AssetTag12010298EX
```

Call the [register-on-premises-instance](#) command, specifying:

- A name that uniquely identifies the on-premises instance (with the `--instance-name` option).

Important

To help identify the on-premises instance, especially for debugging purposes, we strongly recommend that you specify a name that maps to some unique characteristic of the on-premises instance (for example, the session-name of the STS credentials and the serial number or an internal asset identifier, if applicable). If you specify a MAC address as a name, be aware that MAC addresses contain characters that AWS CodeDeploy does not allow, such as colon (:). For a list of allowed characters, see [AWS CodeDeploy Limits \(p. 340\)](#).

- The IAM session ARN that you set up to authenticate multiple on-premises instances in [Step 1: Create the IAM Role that On-Premises Instances Will Assume \(p. 187\)](#).

For example:

```
aws deploy register-on-premises-instance --instance-name name-of-instance --iam-session-arn  
arn:aws:sts::account-id:assumed-role/role-to-assume/session-name
```

Where:

- *name-of-instance* is the name you use to identify the on-premises instance, such as `AssetTag12010298EX`.
- *account-id* is the 12-digit account ID for your organization, such as `111222333444`.
- *role-to-assume* is the name of the IAM role you created for the instance, such as `CodeDeployInstanceRole`.
- *session-name* is the name of the session role you specified in [Step 2: Generate Temporary Credentials for an Individual Instance Using AWS STS \(p. 188\)](#).

Step 6: 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 deployment targets during a deployment.)

To tag the on-premises instance (CLI)

- 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 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 values only.

For example:

```
aws deploy add-tags-to-on-premises-instances --instance-names AssetTag12010298EX --tags Key=Name,Value=CodeDeployDemo-OnPrem
```

To tag the on-premises instance (console)

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.
Note
Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#).
2. From the AWS CodeDeploy menu, choose **On-premises instances**.
3. In the list of on-premises instances, choose the arrow next to the on-premises instance 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, choose the delete icon (✕).
5. After you have added tags, choose **Update Tags**.

Step 7: Deploy 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 in a way that's similar to deploying application revisions to Amazon EC2 instances. For instructions, see [Create a Deployment with AWS CodeDeploy \(p. 245\)](#). These instructions include a 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 described in [Step 2: Create a Sample Application Revision \(p. 91\)](#) in the [Tutorial: Deploy an Application to an On-Premises Instance with AWS CodeDeploy \(Windows Server, Ubuntu Server, or Red Hat Enterprise Linux\) \(p. 90\)](#).

Important

If you reuse an 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 [Step 3: Create a Service Role for AWS CodeDeploy \(p. 21\)](#).

Step 8: 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 in a way that's similar to tracking deployments to Amazon EC2 instances. For instructions, see [View Deployment Details with AWS CodeDeploy \(p. 254\)](#).

Managing On-Premises Instances Operations in AWS CodeDeploy

Follow the instructions in this section to manage operations on your on-premises instances after you have registered them with AWS CodeDeploy, such as getting more information about, removing tags from, and uninstalling and deregistering on-premises instances.

Topics

- [Get Information About a Single On-Premises Instance \(p. 193\)](#)
- [Get Information About Multiple On-Premises Instances \(p. 193\)](#)
- [Manually Remove On-Premises Instance Tags from an On-Premises Instance \(p. 194\)](#)
- [Automatically Uninstall the AWS CodeDeploy Agent and Remove the Configuration File from an On-Premises Instance \(p. 195\)](#)
- [Automatically Deregister an On-Premises Instance \(p. 196\)](#)
- [Manually Deregister an On-Premises Instance \(p. 197\)](#)

Get Information About a Single On-Premises Instance

You can get information about a single on-premises instance by following the instructions in [View Deployment Details with AWS CodeDeploy \(p. 254\)](#). You can use the AWS CLI or the AWS CodeDeploy console to get more information about a single on-premises instance.

To get information about a single on-premises instance (CLI)

- Call the [get-on-premises-instance](#) command, specifying the name that uniquely identifies the on-premises instance (with the `--instance-name` option):

```
aws deploy get-on-premises-instance --instance-name AssetTag12010298EX
```

To get information about a single on-premises instance (console)

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#).

2. In the AWS CodeDeploy menu, choose **On-premises instances**.
3. In the list of on-premises instances, choose the arrow next to the on-premises instance. Details about the on-premises instance are displayed.

Get Information About Multiple On-Premises Instances

You can get information about on-premises instances by following the instructions in [View Deployment Details with AWS CodeDeploy \(p. 254\)](#). You can use the AWS CLI or the AWS CodeDeploy console to get more information about on-premises instances.

To get information about multiple on-premises instances (CLI)

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 and `Registered` or `Deregistered`, respectively). If you omit this, then both registered and deregistered on-premises instance names are returned.
- Whether to get information only about on-premises instances 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 names of the on-premises instances (with the `--instance-names` option):

```
aws deploy batch-get-on-premises-instances --instance-names AssetTag12010298EX
AssetTag09920444EX
```

To get information about multiple on-premises instances (console)

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. From the AWS CodeDeploy menu, choose **On-premises instances**. Information about the on-premises instances is displayed.

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 tag is no longer being used, or you want to remove the on-premises instance from any deployment groups that rely on that tag. You can use the AWS CLI or the AWS CodeDeploy console to remove on-premises instance tags from on-premises instances.

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 instance. It does not uninstall the AWS CodeDeploy agent from the instance. It does not remove the configuration file from the instance. It does not delete the IAM user associated with the instance.

To automatically deregister the on-premises instance, see [Automatically Deregister an On-Premises Instance](#) (p. 196).

To manually deregister the on-premises instance, see [Manually Deregister an On-Premises Instance](#) (p. 197).

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 \(p. 195\)](#).

To manually uninstall just the AWS CodeDeploy agent from the on-premises instance, see [Managing AWS CodeDeploy Agent Operations \(p. 132\)](#).

To manually delete the associated IAM user, see [Deleting an IAM User from Your AWS Account](#).

To remove on-premises instance tags from an on-premises instance (CLI)

- Call the [remove-tags-from-on-premises-instances](#), specifying:
 - The names that uniquely identify the on-premises instance (with the `--instance-names` option).
 - The names and values of the tags you want to remove (with the `--tags` option).

For example:

```
aws deploy remove-tags-from-on-premises-instances --instance-names AssetTag12010298EX
--tags Key=Name,Value=CodeDeployDemo-OnPrem
```

To remove on-premises instance tags from an on-premises instance (console)

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.
- Note**
Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#).
2. In the AWS CodeDeploy menu, choose **On-premises instances**.
 3. In the list of on-premises instances, choose the arrow next to the on-premises instance from which you want to remove tags.
 4. In the **Tags** area, choose the delete icon (✕) next to each tag you want to remove.
 5. After you have deleted the tags, choose **Update tags**.

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.

Note

Automatically uninstalling the AWS CodeDeploy agent and removing the configuration file from an on-premises instance does not deregister an on-premises instance. It does not disassociate any on-premises instance tags associated with the on-premises instance. It does not delete the IAM user associated with the on-premises instance.

To automatically deregister the on-premises instance, see [Automatically Deregister an On-Premises Instance \(p. 196\)](#).

To manually deregister the on-premises instance, see [Manually Deregister an On-Premises Instance \(p. 197\)](#).

To manually disassociate any associated on-premises instance tags, see [Manually Remove On-Premises Instance Tags from an On-Premises Instance \(p. 194\)](#).

To manually uninstall the AWS CodeDeploy agent from the on-premises instance, see [Managing AWS CodeDeploy Agent Operations \(p. 132\)](#).

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 and RHEL, this is `/etc/codedeploy-agent/conf/codedeploy.onpremises.yml`. For Windows Server, this is `C:\ProgramData\Amazon\CodeDeploy\conf.onpremises.yml`.)

Automatically Deregister an On-Premises Instance

Typically, you deregister an on-premises instance after you're no longer planning to deploy to it. When you deregister an on-premises instance, even though the on-premises instance might be part of a deployment group's on-premises instance tags, the on-premises instance will not be included in any deployments. You can use the AWS CLI to deregister on-premises instances.

Note

You cannot use the AWS CodeDeploy console to deregister an on-premises instance. Also, deregistering an on-premises instance does not disassociate any on-premises instance tags that are associated with the on-premises instance. It does not uninstall the AWS CodeDeploy agent from the on-premises instance. It does not remove 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. 197).

To manually disassociate any associated on-premises instance tags, see [Manually Remove On-Premises Instance Tags from an On-Premises Instance](#) (p. 194).

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. 195).

To manually uninstall only the AWS CodeDeploy agent from the on-premises instance, see [Managing AWS CodeDeploy Agent Operations](#) (p. 132).

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 associated with the on-premises instance (with the `--delete-iam-user` option, the default). If you do not want to delete the IAM user associated with the on-premises instance, 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 listed in [Region and Endpoints](#) in the *AWS General Reference* (for example, `us-west-2`). If this option is not specified, the default AWS region 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 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.

Manually Deregister an On-Premises Instance

Typically, you deregister an on-premises instance after you're no longer planning to deploy to it. You use the AWS CLI to manually deregister on-premises instances.

Manually deregistering an on-premises instance does not uninstall the AWS CodeDeploy agent. It does not remove the configuration file from the instance. It does not delete the IAM user associated with the instance. It does not remove any tags associated with the instance.

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. 195).

To manually uninstall only the AWS CodeDeploy agent, see [Managing AWS CodeDeploy Agent Operations](#) (p. 132).

To manually delete the associated IAM user, see [Deleting an IAM User from Your AWS Account](#).

To manually remove only the associated on-premises instance tags, see [Manually Remove On-Premises Instance Tags from an On-Premises Instance](#) (p. 194).

- Call the [deregister-on-premises-instance](#) command, specifying the name that uniquely identifies the on-premises instance (with the `--instance-name` option):

```
aws deploy deregister-on-premises-instance --instance-name AssetTag12010298EX
```

After you deregister an on-premises instance, you cannot create a replacement 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.

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 used in a deployment.

For information about using AWS CodeDeploy API actions to view instances, see [GetDeploymentInstance](#), [ListDeploymentInstances](#), and [ListOnPremisesInstances](#).

Topics

- [View Instance Details \(Console\)](#) (p. 198)
- [View Instance Details \(CLI\)](#) (p. 198)

View Instance Details (Console)

To view instance details:

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#).

2. On the AWS CodeDeploy menu, choose **Deployments**.

Note

If no entries are displayed, make sure the correct region is selected. On the navigation bar, in the region selector, choose one of the regions listed in [Region and Endpoints](#) in the *AWS General Reference*. AWS CodeDeploy is supported in these regions only.

3. To display deployment details, choose the arrow next to the deployment ID that corresponds to the instance.
4. In **Instances**, choose **View all instances**.
5. To see information about individual deployment lifecycle events for an instance, on the deployment details page, in the **Events** column, choose **View events**.

Note

If **Failed** is displayed for any of the lifecycle events, on the instance details page, choose **View logs**, **View in EC2**, or both. You can find troubleshooting tips in [Troubleshoot Instance Issues \(p. 354\)](#).

6. If you want to see more information about an Amazon EC2 instance, but **View in EC2** is not available on the instance details page, return to the deployment details page, and in the **Instance ID** column, choose the ID of the Amazon EC2 instance.

View Instance Details (CLI)

To use the AWS CLI to view instance details, call either the `get-deployment-instance` command or the `list-deployment-instances` command.

To view details about a single instance, call the `get-deployment-instance` command, specifying:

- The unique deployment ID. To get the deployment ID, call the `list-deployments` command.
- The unique instance ID. To get the instance ID, call the `list-deployment-instances` command.

To view a list of IDs for instances used 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.)

AWS CodeDeploy Instance Health

AWS CodeDeploy monitors the health status of the instances in a deployment group. It fails deployments if the number of healthy instances falls below the minimum number of healthy instances that have been specified for the deployment group during a deployment. For example, if 85% of instances must remain

healthy during a deployment, and the deployment group contains 10 instances, the overall deployment will fail if deployment to even a single instance fails. This is because when an instance is taken offline so the latest application revision can be installed, the available healthy instance counts already drops to 90%. A failed instance plus another offline instance would mean that only 80% of instances are healthy and available. AWS CodeDeploy will fail the overall deployment.

It is important to remember that for the overall deployment to succeed, the following must be true:

- AWS CodeDeploy is able to deploy to each instance in the deployment.
- Deployment to at least one instance must succeed. This means that even if the minimum healthy hosts value is 0, deployment to at least one instance must succeed (that is, at least one instance must be healthy) for the overall deployment to succeed.

The required minimum number of healthy instances is defined as part of a deployment configuration.

Important

During a blue/green deployment, the deployment configuration and minimum healthy hosts value apply to instances in the replacement environment, not those in the original environment. However, when instances in the original environment are deregistered from the load balancer, the overall deployment is marked as Failed if even a single original instance fails to be deregistered successfully.

AWS CodeDeploy provides three default deployment configurations that have commonly used minimum healthy host values:

Default deployment configuration name	Predefined minimum healthy hosts value
CodeDeployDefault.OneAtATime	99%
CodeDeployDefault.HalfAtATime	50%
CodeDeployDefault.AllAtOnce	0

You'll find more information about default deployment configurations in [Working with Deployment Configurations in AWS CodeDeploy \(p. 203\)](#).

You can create custom deployment configurations in AWS CodeDeploy to define your own minimum healthy host values. You can define these values as whole numbers or percentages when using the following operations:

- As `minimum-healthy-hosts` when you use the `create-deployment-config` command in the AWS CLI.
- As `Value` in the `MinimumHealthyHosts` data type in the AWS CodeDeploy API.
- As `MinimumHealthyHosts` when you use `AWS::CodeDeploy::DeploymentConfig` in an AWS CloudFormation template.

Topics

- [Health Status \(p. 199\)](#)
- [Minimum Healthy Instances and Deployments \(p. 200\)](#)

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 application revision currently installed on the instance. It has the following status values:

- **Current:** The revision installed on the instance matches the revision for the deployment group's last successful deployment.
- **Old:** The revision installed on the instance matches an older version of the application.
- **Unknown:** The application revision has not been installed successfully on the instance.

Instance health

Instance health is based on whether deployments to an instance have been successful. It has the following values:

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

Minimum Healthy Instances and Deployments

AWS CodeDeploy allows you to specify a minimum number of healthy instances for the deployment 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 proceed.

You can specify the minimum number of healthy instances for your deployment group as a number of instances or as a percentage of the total number of instances. 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.

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

Note

AWS CodeDeploy will attempt to deploy to all instances in a deployment group, even those that are currently in a Stopped state. In the minimum healthy host calculation, a stopped instance has the same impact as a failed instance. To resolve deployment failures due to too many stopped instances, either restart instances or change their tags to exclude them from the deployment group.

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 the specified minimum number of healthy instances.

- If the number of healthy instances is less than or equal to the specified minimum number of healthy instances, AWS CodeDeploy cancels the deployment to ensure the number of healthy instances doesn't decrease with more deployments.
- If the number of healthy instances is greater than the specified minimum number of healthy instances by at least one, AWS CodeDeploy deploys 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 the specified minimum number of healthy instances. If the number of healthy instances falls to the specified minimum number at any point in the deployment process, AWS CodeDeploy stops the deployment. This practice prevents the possibility the next deployment will fail, dropping the number of healthy instances below the specified minimum number.

Note

Make sure the minimum number of healthy instances you specify is less than the total number of instances in the deployment group. If you specify a percentage value, remember it will be rounded up. Otherwise, when the deployment starts, the number of healthy instances will already be less than or equal to the specified minimum number of healthy instances, and AWS CodeDeploy will immediately fail the overall deployment.

AWS CodeDeploy also uses the specified minimum number of healthy instances and the actual 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 the specified minimum number of healthy instances. For example:

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

Examples

The following examples assume a deployment group with 10 instances.

Minimum healthy instances: 95%

AWS CodeDeploy rounds the minimum healthy instances number 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 deployment to any of the instances fails, AWS CodeDeploy immediately fails the overall deployment because the number of healthy instances equals the minimum healthy instances number. The exception to this rule is that if the last instance fails, the deployment still succeeds.

AWS CodeDeploy continues the deployment, one instance at a time, until any deployment fails or the overall deployment is complete. If all 10 deployments succeed, the deployment group now has 10 healthy instances.

Minimum healthy instances: 8

AWS CodeDeploy deploys the revision to two instances at a time. If two of these deployments fail, AWS CodeDeploy immediately fails the overall deployment. The exception to this rule is that if the last instance is the second to fail, the deployment still succeeds.

Minimum healthy instances: 0

AWS CodeDeploy deploys the revision to the entire deployment group at once. At least one deployment to an instance must succeed for the overall deployment to succeed. If 0 instances are healthy, then the deployment fails. This is because of the requirement that in order to mark an overall deployment as successful, at least one instance must be healthy when the overall deployment is completed, even if the minimum healthy instances value is 0.

Working with Deployment Configurations in AWS CodeDeploy

A deployment configuration is a set of rules and success and failure conditions used by AWS CodeDeploy during a deployment. These rules and conditions are different, depending on whether you deploy to an EC2/On-Premises compute platform or an AWS Lambda compute platform.

Deployment Configurations on an EC2/On-Premises Compute Platform

When you deploy to an EC2/On-Premises compute platform, the deployment configuration specifies through the use of a minimum healthy hosts value, the number or percentage of instances that must remain available at any time during a deployment.

You can use one of the three predefined deployment configurations provided by AWS or create a custom deployment configuration. If you don't specify a deployment configuration, AWS CodeDeploy uses the CodeDeployDefault.OneAtATime deployment configuration.

For more information about how AWS CodeDeploy monitors and evaluates instance health during a deployment, see [Instance Health \(p. 198\)](#). To view a list of deployment configurations already registered to your AWS account, see [View Deployment Configuration Details \(p. 207\)](#).

Predefined Deployment Configurations for an EC2/On-Premises Compute Platform

The following table lists the predefined deployment configurations.

Deployment Configuration	Description
CodeDeployDefault.AllAtOnce	<p>In-place deployments:</p> <p>Attempts to deploy an application revision to as many instances as possible at once. The status of the overall deployment will be displayed as Succeeded if the application revision is deployed to one or more of the instances. The status of the overall deployment will be displayed as Failed if the application revision is not deployed to any of the instances. Using an example of nine instances, CodeDeployDefault.AllAtOnce will attempt to deploy to all nine instances at once. The overall deployment will succeed if deployment to even a single instance is successful; it will fail only if deployments to all nine instances fail.</p> <p>Blue/green deployments:</p> <ul style="list-style-type: none">Deployment to replacement environment: Follows same deployment rules as

Deployment Configuration	Description
	<p>CodeDeployDefault.AllAtOnce for in-place deployments.</p> <ul style="list-style-type: none">• Traffic rerouting: Routes traffic to all instances in the replacement environment at once. Succeeds if traffic is successfully rerouted to at least one instance. Fails after rerouting to all instances fails.
CodeDeployDefault.HalfAtATime	<p>In-place deployments:</p> <p>Deploys to up to half of the instances at a time (with fractions rounded down). The overall deployment succeeds if the application revision is deployed to at least half of the instances (with fractions rounded up); otherwise, the deployment fails. In the example of nine instances, it will deploy to up to four instances at a time. The overall deployment succeeds if deployment to five or more instances succeed; otherwise, the deployment fails.</p> <p>Blue/green deployments:</p> <ul style="list-style-type: none">• Deployment to replacement environment: Follows same deployment rules as CodeDeployDefault.HalfAtATime for in-place deployments.• Traffic rerouting: Routes traffic to up to half the instances in the replacement environment at a time. Succeeds if rerouting to at least half of the instances succeeds; fails otherwise.

Deployment Configuration	Description
CodeDeployDefault.OneAtATime	<p>In-place deployments:</p> <p>Deploys the application revision to only one instance at a time.</p> <p>For deployment groups that contain more than one instance:</p> <ul style="list-style-type: none">• The overall deployment succeeds if the application revision is deployed to all of the instances. The exception to this rule is if deployment to the last instance fails, the overall deployment still succeeds. This is because AWS CodeDeploy allows only one instance at a time to be taken offline with the CodeDeployDefault.OneAtATime configuration.• The overall deployment fails as soon as the application revision fails to be deployed to any but the last instance.• In an example using nine instances, it will deploy to one instance at a time. The overall deployment succeeds if deployment to the first eight instances is successful; the overall deployment fails if deployment to any of the first eight instances fails. <p>For deployment groups that contain only one instance, the overall deployment is successful only if deployment to the single instance is successful.</p> <p>Blue/green deployments:</p> <ul style="list-style-type: none">• Deployment to replacement environment: Follows same deployment rules as CodeDeployDefault.OneAtATime for in-place deployments.• Traffic rerouting: Routes traffic to one instance in the replacement environment at a time. Succeeds if traffic is successfully rerouted to all replacement instances. Fails after the very first rerouting failure. The exception to this rule is if the last instance fails to register, the overall deployment still succeeds.

Deployment Configurations on an AWS Lambda Compute Platform

When you deploy to an AWS Lambda compute platform, the deployment configuration specifies the way traffic is shifted to the new Lambda function versions in your application.

There are three ways traffic can shift during a deployment:

- **Canary:** Traffic is shifted in two increments. You can choose from predefined canary options that specify the percentage of traffic shifted to your updated Lambda function version in the first increment and the interval, in minutes, before the remaining traffic is shifted in the second increment.
- **Linear:** Traffic is shifted in equal increments with an equal number of minutes between each increment. You can choose from predefined linear options that specify the percentage of traffic shifted in each increment and the number of minutes between each increment.
- **All-at-once:** All traffic is shifted from the original Lambda function to the updated Lambda function version at once.

You can also create your own custom canary or linear deployment configuration.

Predefined Deployment Configurations for an AWS Lambda Compute Platform

The following table lists the predefined configurations available for AWS Lambda deployments.

Deployment Configuration	Description
<code>CodeDeployDefault.LambdaCanary10Percent5Minutes</code>	Shifts 10 percent of traffic in the first increment. The remaining 90 percent is deployed five minutes later.
<code>CodeDeployDefault.LambdaCanary10Percent10Minutes</code>	Shifts 10 percent of traffic in the first increment. The remaining 90 percent is deployed 10 minutes later.
<code>CodeDeployDefault.LambdaCanary10Percent15Minutes</code>	Shifts 10 percent of traffic in the first increment. The remaining 90 percent is deployed 15 minutes later.
<code>CodeDeployDefault.LambdaCanary10Percent30Minutes</code>	Shifts 10 percent of traffic in the first increment. The remaining 90 percent is deployed 30 minutes later.
<code>CodeDeployDefault.LambdaLinear10PercentEvery1Minute</code>	Shifts 10 percent of traffic every minute until all traffic is shifted.
<code>CodeDeployDefault.LambdaLinear10PercentEvery2Minutes</code>	Shifts 10 percent of traffic every two minutes until all traffic is shifted.
<code>CodeDeployDefault.LambdaLinear10PercentEvery3Minutes</code>	Shifts 10 percent of traffic every three minutes until all traffic is shifted.
<code>CodeDeployDefault.LambdaLinear10PercentEvery10Minutes</code>	Shifts 10 percent of traffic every 10 minutes until all traffic is shifted.
<code>CodeDeployDefault.LambdaAllAtOnce</code>	Shifts all traffic to the updated Lambda functions at once.

Topics

- [Create a Deployment Configuration \(p. 207\)](#)
- [View Deployment Configuration Details \(p. 207\)](#)

- [Delete a Deployment Configuration \(p. 208\)](#)

Create a Deployment Configuration with AWS CodeDeploy

You can use the AWS CLI, the AWS CodeDeploy APIs, or an AWS CloudFormation template to create custom deployment configurations.

Note

In the AWS CodeDeploy console, you can use the Sample deployment wizard for an in-place deployment to create a custom deployment configuration.

For information about using an AWS CloudFormation template to create a deployment configuration, see [AWS CloudFormation Templates for AWS CodeDeploy Reference \(p. 333\)](#).

To use the AWS CLI to create a deployment configuration, call the [create-deployment-config](#) command, specifying:

- A name that uniquely identifies the deployment configuration. This name must be unique across all of the deployment configurations 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 \(p. 198\)](#).

The following example creates an EC2/On-Premises deployment configuration named `ThreeQuartersHealthy` that requires 75% of target instances to remain healthy during a deployment:

```
aws deploy create-deployment-config --deployment-config-name ThreeQuartersHealthy --
minimum-healthy-hosts type=FLEET_PERCENT,value=75
```

The following example creates an AWS Lambda deployment configuration named `Canary25Percent45Minutes`. It uses canary traffic shifting to shift 25 percent of traffic in the first increment. The remaining 75 percent shifted 45 minutes later:

```
aws deploy create-deployment-config --deployment-config-
name Canary25Percent45Minutes --traffic-routing-config
"type="TimeBasedCanary",timeBasedCanary={canaryPercentage=25,canaryInterval=45}" --
compute-platform Lambda
```

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 deployment configurations associated with your AWS account. For descriptions of the predefined AWS CodeDeploy deployment configurations, see [Predefined Deployment Configurations for an EC2/On-Premises Compute Platform \(p. 203\)](#).

Topics

- [View Deployment Configuration Details \(Console\) \(p. 208\)](#)
- [View Deployment Configuration \(CLI\) \(p. 208\)](#)

View Deployment Configuration Details (Console)

To use the AWS CodeDeploy console to view a list of deployment configuration names:

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. On the AWS CodeDeploy menu, choose **Deployment configurations** to see a list of deployment configuration names and criteria for each deployment configuration.

Note

If no entries are displayed, make sure the correct region is selected. On the navigation bar, in the region selector, choose one of the regions listed in [Region and Endpoints](#) in the *AWS General Reference*. AWS CodeDeploy is supported in these regions only.

View Deployment Configuration (CLI)

To use the AWS CLI to view deployment configuration details, call either the `get-deployment-config` command or the `list-deployment-configs` command.

To view details about a single deployment configuration, call the `get-deployment-config` command, specifying the unique deployment configuration name.

To view details about multiple deployment configurations, call the `list-deployments` command

Delete a Deployment Configuration with AWS CodeDeploy

You can use the AWS CLI or the AWS CodeDeploy APIs to delete custom deployment configurations associated with your AWS account. You cannot delete built-in deployment configurations, such as `CodeDeployDefault.AllAtOnce`, `CodeDeployDefault.HalfAtATime`, and `CodeDeployDefault.OneAtATime`.

Warning

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 it with new deployments and new deployment groups. This action cannot be undone.

To use the AWS CLI to delete a deployment configuration, call the `delete-deployment-config` command, specifying the deployment configuration name. To view a list of deployment configuration names, call the `list-deployment-configs` command.

The following example deletes a deployment configuration named `ThreeQuartersHealthy`.

```
aws deploy delete-deployment-config --deployment-config-name ThreeQuartersHealthy
```


Working with Applications in AWS CodeDeploy

After you configure instances, but before you can deploy a revision, you must create an application in AWS CodeDeploy. An *application* is simply a name or container used by AWS CodeDeploy to ensure the correct revision, deployment configuration, and deployment group are referenced during a deployment.

Use the information in the following table for next steps:

I haven't created instances yet.	See Working with Instances for AWS CodeDeploy (p. 147), and then return to this page.
I have created instances, but I haven't finished tagging them.	See Tagging Instances for AWS CodeDeploy Deployments (p. 149), and then return to this page.
I haven't created an application yet.	See Create an Application with AWS CodeDeploy (p. 209)
I have already created an application, but I haven't created a deployment group.	See Create a Deployment Group with AWS CodeDeploy (p. 220).
I have already created an application and deployment group, but I haven't created an application revision.	See Working with Application Revisions for AWS CodeDeploy (p. 232).
I have already created an application and deployment group, and I have already uploaded my application revision. I'm ready to deploy.	See Create a Deployment with AWS CodeDeploy (p. 245).

Topics

- [Create an Application with AWS CodeDeploy](#) (p. 209)
- [View Application Details with AWS CodeDeploy](#) (p. 216)
- [Rename an AWS CodeDeploy Application](#) (p. 217)
- [Delete an Application in AWS CodeDeploy](#) (p. 217)

Create an Application with AWS CodeDeploy

An *application* is simply a name or container used by AWS CodeDeploy to ensure that the correct revision, deployment configuration, and deployment group are referenced during a deployment. You can use the AWS CodeDeploy console, the AWS CLI, the AWS CodeDeploy APIs, or an AWS CloudFormation template to create applications.

Your code, or application revision, is installed to instances through a process called a deployment. AWS CodeDeploy supports two types of deployments:

- **In-place deployment:** The application on each instance in the deployment group is stopped, the latest application revision is installed, and the new version of the application is started and validated.

You can use a load balancer so that each instance is deregistered during its deployment and then restored to service after the deployment is complete. Only deployments that use the EC2/On-Premises compute platform can use in-place deployments. For more information about in-place deployments, see [Overview of an In-Place Deployment \(p. 4\)](#).

- **Blue/green deployment:** The behavior of your deployment depends on which compute platform you use:
 - **Blue/green on an EC2/On-Premises compute platform:** The instances in a deployment group (the original environment) are replaced by a different set of instances (the replacement environment) using these steps:
 - Instances are provisioned for the replacement environment.
 - The latest application revision is installed on the replacement instances.
 - An optional wait time occurs for activities such as application testing and system verification.
 - Instances in the replacement environment are registered with an Elastic Load Balancing load balancer, causing traffic to be rerouted to them. Instances in the original environment are deregistered and can be terminated or kept running for other uses.

Note

When using an EC2/On-Premises compute platform, blue/green deployments work with Amazon EC2 instances only.

- **Blue/green on an AWS Lambda compute platform:** Traffic is shifted from your current serverless environment to one with your updated Lambda function versions. You can specify Lambda functions that perform validation tests and choose the way in which the traffic shift occurs. All AWS Lambda compute platform deployments are blue/green deployments. For this reason, you do not need to specify a deployment type.

For more information about blue/green deployments, see [Overview of a Blue/Green Deployment \(p. 5\)](#).

When you use the AWS CodeDeploy console to create an application, you configure its first deployment group at the same time. When you use the AWS CLI to create an application, you create its first deployment group in a separate step.

To view a list of applications already registered to your AWS account, see [View Application Details with AWS CodeDeploy \(p. 216\)](#). For information about using an AWS CloudFormation template to create an application, see [AWS CloudFormation Templates for AWS CodeDeploy Reference \(p. 333\)](#).

Both deployment types do not apply to all destinations. The following table lists which deployment types work with deployments to the three types of deployment destinations.

Deployment destination	In-place	Blue/green
Amazon EC2	Yes	Yes
On-premises	Yes	No
Serverless AWS Lambda functions	No	Yes

Topics

- [Create an Application for an In-Place Deployment \(Console\) \(p. 211\)](#)
- [Create an Application for a Blue/Green Deployment \(Console\) \(p. 212\)](#)
- [Create an Application for an AWS Lambda Function Deployment \(Console\) \(p. 215\)](#)
- [Create an Application \(CLI\) \(p. 216\)](#)

Create an Application for an In-Place Deployment (Console)

To use the AWS CodeDeploy console to create an application for an in-place deployment:

Warning

Do not follow these steps if:

- You have not prepared your instances to be used in AWS CodeDeploy deployments. To set up your instances, follow the instructions in [Working with Instances for AWS CodeDeploy \(p. 147\)](#), and then follow the steps in this topic.
- You want to create an application that uses a custom deployment configuration, but you have not yet created the deployment configuration. Follow the instructions in [Create a Deployment Configuration \(p. 207\)](#), and then follow the steps in this topic.
- You do not have a service role that trusts AWS CodeDeploy with the minimum required trust and permissions. To create and configure a service role with the required permissions, follow the instructions in [Step 3: Create a Service Role for AWS CodeDeploy \(p. 21\)](#), and then return to the steps in this topic.
- You want to select a Classic Load Balancer, Application Load Balancer, or Network Load Balancer in Elastic Load Balancing for the in-place deployment, but have not yet created it.

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#).

2. If the AWS CodeDeploy home page appears, choose **Get Started Now**.
3. Choose **Create application**.
4. In the **Application name** box, type a name for the application. (In an AWS account, an AWS CodeDeploy application name can be used only once per region. You can reuse an application name in different regions.)
5. From the **Compute platform** drop-down list, choose **EC2/On-Premises**.
6. In the **Deployment group name** box, type a name that describes the deployment group.

Note

If you want to use the same settings used in another deployment group (including the deployment group name; tags, Auto Scaling group names, or both; and the deployment configuration), specify those settings on this page. Although this new deployment group and the existing deployment group have the same name, AWS CodeDeploy treats them as separate deployment groups, because they are each associated with separate applications.

7. Choose **In-place deployment**.
8. In **Environment configuration**, choose from the following:
 - On the **Auto Scaling groups** tab: Choose the name of an Auto Scaling group to deploy your application revision to. When new Amazon EC2 instances are launched as part of an Auto Scaling group, AWS CodeDeploy can deploy your revisions to the new instances automatically. You can add up to 10 Auto Scaling groups to a deployment group.
 - On the **Amazon EC2 instances** tab or **On-premises instances** tab: In the **Key** and **Value** fields, type the values of the key-value pair you used to tag the instances. You can tag up to 10 key-value pairs in a single tag group.
 - You can use wildcards in the **Value** field to identify all instances tagged in certain patterns, such as similar Amazon EC2 instance, cost center, and group names, and so on. For example, if you

select **Name** in the **Key** field and type **GRP-*a** in the **Value** field, AWS CodeDeploy identifies all instances that fit that pattern, such as **GRP-1a**, **GRP-2a**, and **GRP-XYZ-a**.

- The **Value** field is case-sensitive.
- To remove a key-value pair from the list, choose the remove icon.

As AWS CodeDeploy finds instances that match each specified key-value pair or Auto Scaling group name, it displays the number of matching instances. To see more information about the instances, click the number.

If you want to refine the criteria for the deployed-to instances, choose **Add tag group** to create an tag group. You can create up to three tag groups with up to 10 key-value pairs each. When you use multiple tag groups in a deployment group, only instances that are identified by all the tag groups are included in the deployment group. That means an instance must match at least one of the tags in each of the groups to be included in the deployment group.

For information about using tag groups to refine your deployment group, see [Tagging Instances for AWS CodeDeploy Deployments \(p. 149\)](#).

9. (Optional) In **Load balancer**, choose **Enable load balancing**, and then choose an existing Classic Load Balancer, Application Load Balancer, or Network Load Balancer to manage traffic to the instances during the deployment processes.

Each instance is deregistered from the load balancer (Classic Load Balancers) or target group (Application Load Balancers and Network Load Balancers) to prevent traffic from being routed to it during the deployment. It is re-registered when the deployment is complete.

For more information about load balancers for AWS CodeDeploy deployments, see [Integrating AWS CodeDeploy with Elastic Load Balancing \(p. 46\)](#).

- 10 In the **Deployment configuration** list, choose a deployment configuration to control the rate at which instances are deployed to, such as one at a time or all at once. For more information about deployment configurations, see [Working with Deployment Configurations in AWS CodeDeploy \(p. 203\)](#).

- 11 (Optional) In **Advanced**, configure any options you want to include in the deployment, such as Amazon SNS notification triggers, Amazon CloudWatch alarms, or automatic rollbacks.

For more information, see [Configure Advanced Options for a Deployment Group \(p. 228\)](#).

- 12 In **Service role ARN**, choose an Amazon Resource Name (ARN) for a service role that trusts AWS CodeDeploy with, at minimum, the trust and permissions described in [Step 3: Create a Service Role for AWS CodeDeploy \(p. 21\)](#). To get the service role ARN, see [Get the Service Role ARN \(Console\) \(p. 24\)](#).

- 13 Choose **Create application**.

The next step is to prepare a revision to deploy to the application and deployment group. For instructions, see [Working with Application Revisions for AWS CodeDeploy \(p. 232\)](#).

Create an Application for a Blue/Green Deployment (Console)

To use the AWS CodeDeploy console to create an application for a blue/green deployment:

Note

A deployment to the AWS Lambda compute platform is always a blue/green deployment. You do not specify a deployment type option.

Warning

Do not follow these steps if:

- You do not have instances with the AWS CodeDeploy agent installed that you want to replace during the blue/green deployment process. To set up your instances, follow the instructions in [Working with Instances for AWS CodeDeploy \(p. 147\)](#), and then follow the steps in this topic.
- You want to create an application that uses a custom deployment configuration, but you have not yet created the deployment configuration. Follow the instructions in [Create a Deployment Configuration \(p. 207\)](#), and then follow the steps in this topic.
- You do not have a service role that trusts AWS CodeDeploy with, at minimum, the trust and permissions described in [Step 3: Create a Service Role for AWS CodeDeploy \(p. 21\)](#). To create and configure a service role, follow the instructions in [Step 3: Create a Service Role for AWS CodeDeploy \(p. 21\)](#), and then follow the steps in this topic.
- You have not created a Classic Load Balancer, Application Load Balancer, or Network Load Balancer in Elastic Load Balancing for the registration of the instances in your replacement environment. For more information, see [Set Up a Load Balancer in Elastic Load Balancing for AWS CodeDeploy Deployments \(p. 224\)](#).

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#).

2. If the AWS CodeDeploy home page appears, choose **Get Started Now**.
3. Choose **Create application**.
4. In **Application name**, type a name for the application. (In an AWS account, an AWS CodeDeploy application name can be used only once per region. You can reuse an application name in different regions.)
5. In the **Compute platform** drop-down list choose **EC2/On-Premises**.
6. In the **Deployment group name** box, type a name that describes the deployment group.

Note

If you want to use the same settings used in another deployment group, specify those settings on this page. The settings you might want to reuse include, for example, the deployment group name, tags, Auto Scaling group names, or deployment configuration. Although the new and existing deployment group have the same name, AWS CodeDeploy treats them as separate deployment groups, because they are associated with separate applications.

7. Choose **Blue/green deployment**.
8. In **Environment configuration**, choose the method to use to provide instances for your replacement environment:
 - **Automatically copy Auto Scaling group:** AWS CodeDeploy creates an Auto Scaling group by copying one you specify.
 - **Manually provision instances:** You won't specify the instances for your replacement environment until you create a deployment. You must create the instances before you start the deployment. Instead, here you specify the instances you want to replace.
9. Depending on your choice in step 7, do one of the following:
 - If you chose **Automatically copy Auto Scaling group:** In **Auto Scaling group**, choose the name of the Auto Scaling group you want to use as a template for the Auto Scaling group that will be created for the instances in your replacement environment. The number of currently healthy instances in the Auto Scaling group you select will be created in your replacement environment.
 - If you chose **Manually provision instances:** In **Choose the EC2 instances or Auto Scaling groups where the current application revision is deployed**, enter Amazon EC2 tag values or Auto Scaling group names to identify the instances in your original environment (that is, the instances you want to replace or that are running the current application revision).

10 In **Load balancer**, choose the Classic Load Balancer, Application Load Balancer, or Network Load Balancer to use in the registration of instances in your replacement environment during the deployment process.

Note

The instances in your original environment can be, but are not required to be, registered with the load balancer you choose.

For more information about load balancers for AWS CodeDeploy deployments, see [Integrating AWS CodeDeploy with Elastic Load Balancing \(p. 46\)](#).

11 In **Deployment settings**, review the default options for rerouting traffic to the replacement environment, which deployment configuration to use for the deployment, and how instances in the original environment are handled after the deployment.

If you want to change the settings, continue to step 11. Otherwise, skip to step 12.

12 To change the deployment settings for the blue/green deployment, choose **Edit deployment settings**, update any of the following settings, and then choose **Submit**.

Setting	Options
Traffic rerouting	<ul style="list-style-type: none">• Reroute traffic immediately: As soon as instances in the replacement environment are provisioned and the latest application revision is installed on them, they are registered with the load balancer automatically, causing traffic to be rerouted to them. Instances in the original environment are then deregistered.• I will choose whether to reroute traffic: Instances in the replacement environment are not registered with the load balancer unless you manually reroute traffic. If the wait time you specify passes without traffic being rerouted, the deployment status is changed to Stopped.
Deployment configuration	<p>Choose the rate at which instances in the replacement environment are registered with the load balancer, such as one at a time or all at once.</p> <p>Note After traffic is successfully routed to the replacement environment, instances in the original environment are deregistered all at once no matter which deployment configuration was selected.</p> <p>For more information, see Working with Deployment Configurations in AWS CodeDeploy (p. 203).</p>
Original instances	<ul style="list-style-type: none">• Terminate the original instances in the deployment group: After traffic is rerouted to the replacement environment, the instances that were deregistered from the load balancer

Setting	Options
	are terminated following the wait period you specify. <ul style="list-style-type: none">• Keep the original instances in the deployment group running: After traffic is rerouted to the replacement environment, the instances that were deregistered from the load balancer are kept running.

13(Optional) In **Advanced**, configure options you want to include in the deployment, such as Amazon SNS notification triggers, Amazon CloudWatch alarms, or automatic rollbacks.

For information about specifying advanced options in deployment groups, see [Configure Advanced Options for a Deployment Group](#) (p. 228).

14In the **Service role ARN** box, choose an Amazon Resource Name (ARN) for a service role that trusts AWS CodeDeploy with, at minimum, the trust and permissions described in [Step 3: Create a Service Role for AWS CodeDeploy](#) (p. 21). To get the service role ARN, see [Get the Service Role ARN \(Console\)](#) (p. 24).

15Choose **Create application**.

The next step is to prepare a revision to deploy to the application and deployment group. For instructions, see [Working with Application Revisions for AWS CodeDeploy](#) (p. 232).

Create an Application for an AWS Lambda Function Deployment (Console)

To use the AWS CodeDeploy console to create an application for a Lambda function deployment:

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. If the AWS CodeDeploy home page appears, choose **Get Started Now**.
3. Choose **Create application**.
4. In **Application name**, type a name for the application. (In an AWS account, an AWS CodeDeploy application name can be used only once per region. You can reuse an application name in different regions.)
5. From the **Compute platform** drop-down list, choose **AWS Lambda**.
6. In **Deployment group name**, type a name for the deployment group.

Note

If you want to use the same settings used in another deployment group, specify those settings on this page. You might want to reuse the deployment triggers, rollbacks, or deployment configuration. Although the new and existing deployment group have the same name, AWS CodeDeploy treats them as separate deployment groups, because they are associated with separate applications.

7. From the **Deployment configuration** drop-down list, choose one of the predefined deployment configurations, and then skip to step 9.

For more information about deployment configurations, see [Deployment Configurations on an AWS Lambda Compute Platform](#) (p. 205).

8. To create a custom configuration, choose **Create deployment configuration** and do the following:
- For **Deployment configuration name**, type a name for the configuration.
 - (Optional) For **Description**, type a description for the configuration.
 - From the **Type** drop-down list, choose a configuration type. If you choose **Canary**, traffic is shifted in two increments. If you choose **Linear**, traffic is shifted in equal increments, with an equal number of minutes between each increment.
 - For **Step**, enter a percentage of traffic, between 1 and 99, to be shifted. If your configuration type is **Canary**, this is the percentage of traffic that is shifted in the first increment. The remaining traffic is shifted after the selected interval in the second increment. If your configuration type is **Linear**, this is the percentage of traffic that is shifted at the start of each interval.
 - In the **Interval** dialog box, enter the number of minutes. If your configuration type is **Canary**, this is the number of minutes between the first and second traffic shift. If your configuration type is **Linear**, this is the number of minutes between each incremental shift.

Note

The maximum length of an AWS Lambda deployment is two days, or 2,880 minutes. Therefore, the maximum value specified for **Interval** for a canary configuration is 2,800 minutes. The maximum value for a linear configuration depends on the value for **Step**. For example, if the step percentage of a linear traffic shift is 25%, then there are four traffic shifts. The maximum interval value would be 2,880 divided by four, or 720 minutes.

- Choose **Submit**.
9. (Optional) In **Advanced**, configure any options you want to include in the deployment, such as Amazon SNS notification triggers, Amazon CloudWatch alarms, or automatic rollbacks.

For more information, see [Configure Advanced Options for a Deployment Group](#) (p. 228).

10. In **Service role ARN**, choose an Amazon Resource Name (ARN) for a service role that trusts AWS CodeDeploy with, at minimum, the trust and permissions described in [Step 3: Create a Service Role for AWS CodeDeploy](#) (p. 21). To get the service role ARN, see [Get the Service Role ARN \(Console\)](#) (p. 24).

11. Choose **Create application**.

Create an Application (CLI)

To use the AWS CLI to create an application, call the [create-application](#) command, specifying a name that uniquely represents the application. (In an AWS account, an AWS CodeDeploy application name can be used only once per region. You can reuse an application name in different regions.)

After you use the AWS CLI to create an application, the next step is to create a deployment group that specifies the instances to which to deploy revisions. For instructions, see [Create a Deployment Group with AWS CodeDeploy](#) (p. 220).

After you create the deployment group, the next step is to prepare a revision to deploy to the application and deployment group. For instructions, see [Working with Application Revisions for AWS CodeDeploy](#) (p. 232).

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 applications associated with your AWS account.

Topics

- [View Application Details \(Console\)](#) (p. 217)
- [View Application Details \(CLI\)](#) (p. 217)

View Application Details (Console)

To use the AWS CodeDeploy console to view application details:

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. On the AWS CodeDeploy menu, choose **Applications**.

Note

If no entries are displayed, make sure the correct region is selected. On the navigation bar, in the region selector, choose one of the regions listed in [Region and Endpoints](#) in the *AWS General Reference*. AWS CodeDeploy is supported in these regions only.

3. To view additional application details, choose the application name in the list.

View Application Details (CLI)

To use the AWS CLI to view application details, call the **get-application** command, the **batch-get-application** command, or the **list-applications** command.

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.

Rename an AWS CodeDeploy Application

You can use the AWS CLI or the AWS CodeDeploy APIs to change the name of an application.

To view a list of application names, use the AWS CLI to call the [list-applications](#) command.

For information about using the AWS CLI to change an application name, see [update-application](#).

For information about using the AWS CodeDeploy APIs to change an application name, see [API_UpdateApplication](#).

Delete an Application in AWS CodeDeploy

You can use the AWS CodeDeploy console, the AWS CLI, or an AWS CodeDeploy API action to delete applications. For information about using the AWS CodeDeploy API action, see [DeleteApplication](#).

Warning

Deleting an application removes information about the application from the AWS CodeDeploy system, including all related deployment group information and deployment details. Deleting an application created for an EC2/On-Premises deployment does not remove any application revisions from instances nor does it delete revisions from Amazon S3 buckets. Deleting an application created for an EC2/On-Premises deployment does not terminate any Amazon EC2 instances or deregister any on-premises instances. This action cannot be undone.

Topics

- [Delete an Application \(Console\)](#) (p. 218)
- [Delete an Application \(AWS CLI\)](#) (p. 218)

Delete an Application (Console)

To use the AWS CodeDeploy console to delete an application:

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. If the **Applications** page does not appear, on the AWS CodeDeploy menu, choose **Applications**.
3. In the list of applications, choose the name of the application you want to delete.
4. On the **Application details** page, in **Deployment groups**, choose the button next to the deployment group. On the **Actions** menu, choose **Delete**. When prompted, type the name of the deployment group to confirm you want to delete it, and then choose **Delete**. Repeat for any additional deployment groups.
5. At the bottom of the **Application details** page, choose **Delete application**.
6. When prompted, type the name of the application to confirm you want to delete it, and then choose **Delete**.

Delete an Application (AWS CLI)

To use the AWS CLI to delete an application, call the [delete-application](#) command, specifying the application name. To view a list of application names, call the [list-applications](#) command.

Working with Deployment Groups in AWS CodeDeploy

Deployment Groups in AWS Lambda Compute Platform Deployments

In an AWS Lambda deployment, a deployment group defines a set of AWS CodeDeploy configurations for future serverless Lambda deployment to the group. For example, the deployment group might specify alarms and rollbacks. A single deployment in an AWS Lambda deployment group can override one or more group configurations.

Deployment Groups in EC2/On-Premises Compute Platform Deployments

In an EC2/On-Premises deployment, a deployment group is a set of individual instances targeted for a deployment. A deployment group contains individually tagged instances, Amazon EC2 instances in Auto Scaling groups, or both.

In an in-place deployment, the instances in the deployment group are updated with the latest application revision.

In a blue/green deployment, traffic is rerouted from one set of instances to another by deregistering the original instances from a load balancer and registering a replacement set of instances that typically has the latest application revision already installed.

You can associate more than one deployment group with an application in AWS CodeDeploy. This makes it possible to deploy an application revision to different sets of instances at different times. For example, you might use one deployment group to deploy an application revision to a set of instances tagged `Test` where you ensure the quality of the code. Next, you deploy the same application revision to a deployment group with instances tagged `Staging` for additional verification. Finally, when you are ready to release the latest application to customers, you deploy to a deployment group that includes instances tagged `Production`.

You can also use multiple tag groups to further refine the criteria for the instances included in a deployment group. For information, see [Tagging Instances for AWS CodeDeploy Deployments \(p. 149\)](#).

When you use the AWS CodeDeploy console to create an application, you configure its first deployment group at the same time. When you use the AWS CLI to create an application, you create its first deployment group in a separate step.

To view a list of deployment groups already associated with your AWS account, see [View Deployment Group Details with AWS CodeDeploy \(p. 226\)](#).

For information about Amazon EC2 instance tags, see [Working with Tags Using the Console](#). For information about on-premises instances, see [Working with On-Premises Instances \(p. 171\)](#). For information about Auto Scaling, see [Integrating AWS CodeDeploy with Auto Scaling \(p. 44\)](#).

Topics

- [the section called “Create a Deployment Group” \(p. 220\)](#)
- [the section called “View Deployment Group Details” \(p. 226\)](#)
- [the section called “Change Deployment Group Settings” \(p. 227\)](#)
- [the section called “Configure Advanced Options for a Deployment Group” \(p. 228\)](#)
- [the section called “Delete a Deployment Group” \(p. 230\)](#)

Create a Deployment Group with AWS CodeDeploy

You can use the AWS CodeDeploy console, the AWS CLI, the AWS CodeDeploy APIs, or an AWS CloudFormation template to create deployment groups. For information about using an AWS CloudFormation template to create a deployment group, see [AWS CloudFormation Templates for AWS CodeDeploy Reference](#) (p. 333).

When you use the AWS CodeDeploy console to create an application, you configure its first deployment group at the same time. When you use the AWS CLI to create an application, you create its first deployment group in a separate step.

As part of creating a deployment group, you must specify a service role. For more information, see [Step 3: Create a Service Role for AWS CodeDeploy](#) (p. 21).

Topics

- [Create a Deployment Group for an In-Place Deployment \(Console\) \(p. 220\)](#)
- [Create a Deployment Group for a Blue/Green Deployment \(Console\) \(p. 222\)](#)
- [Set Up a Load Balancer in Elastic Load Balancing for AWS CodeDeploy Deployments \(p. 224\)](#)
- [Create a Deployment Group \(CLI\) \(p. 225\)](#)

Create a Deployment Group for an In-Place Deployment (Console)

To use the AWS CodeDeploy console to create a deployment group for an in-place deployment:

Warning

Do not follow these steps if:

- You have not prepared your instances to be used in the first AWS CodeDeploy deployment of an application. To set up your instances, follow the instructions in [Working with Instances for AWS CodeDeploy](#) (p. 147), and then follow the steps in this topic.
- You want to create a deployment group that uses a custom deployment configuration, but you have not yet created the deployment configuration. Follow the instructions in [Create a Deployment Configuration](#) (p. 207), and then follow the steps in this topic.
- You do not have a service role that trusts AWS CodeDeploy with, at minimum, the trust and permissions described in [Step 3: Create a Service Role for AWS CodeDeploy](#) (p. 21). To create and configure a service role, follow the instructions in [Step 3: Create a Service Role for AWS CodeDeploy](#) (p. 21), and then follow the steps in this topic.
- You want to select a Classic Load Balancer, Application Load Balancer, or Network Load Balancer in Elastic Load Balancing for the in-place deployment, but have not yet created it.

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#).

2. On the AWS CodeDeploy menu, choose **Applications**.
3. On the **Applications** page, choose the name of the application for which you want to create a deployment group.
4. Choose **Create deployment group**.
5. In the **Deployment group name** box, type a name that describes the deployment group.

Note

If you want to use the same settings used in another deployment group (including the deployment group name; tags, Auto Scaling group names, or both; and the deployment configuration), specify those settings on this page. Although this new deployment group and the existing deployment group have the same name, AWS CodeDeploy treats them as separate deployment groups, because they are each associated with separate applications.

6. Choose **In-place deployment**.
7. In **Environment configuration**, choose from the following:
 - On the **Auto Scaling groups** tab: Choose the name of an Auto Scaling group to deploy your application revision to. When new Amazon EC2 instances are launched as part of an Auto Scaling group, AWS CodeDeploy can deploy your revisions to the new instances automatically. You can add up to 10 Auto Scaling groups to a deployment group.
 - On the **Amazon EC2 instances** tab or **On-premises instances** tab: In the **Key** and **Value** fields, type the values of the key-value pair you used to tag the instances. You can tag up to 10 key-value pairs in a single tag group.
 - You can use wildcards in the **Value** field to identify all instances tagged in certain patterns, such as similar Amazon EC2 instance, cost center, and group names, and so on. For example, if you select **Name** in the **Key** field and type **GRP-*a** in the **Value** field, AWS CodeDeploy identifies all instances that fit that pattern, such as **GRP-1a**, **GRP-2a**, and **GRP-XYZ-a**.
 - The **Value** field is case-sensitive.
 - To remove a key-value pair from the list, choose the remove icon.

As AWS CodeDeploy finds instances that match each specified key-value pair or Auto Scaling group name, it displays the number of matching instances. To see more information about the instances, click the number.

If you want to refine the criteria for the deployed-to instances, choose **Add tag group** to create an tag group. You can create up to three tag groups with up to 10 key-value pairs each. When you use multiple tag groups in a deployment group, only instances that are identified by all the tag groups are included in the deployment group. That means an instance must match at least one of the tags in each of the groups to be included in the deployment group.

For information about using tag groups to refine your deployment group, see [Tagging Instances for AWS CodeDeploy Deployments \(p. 149\)](#).

8. (Optional) In **Load balancer**, choose **Enable load balancing**, and then choose an existing Classic Load Balancer, Application Load Balancer, or Network Load Balancer to manage traffic to the instances during the deployment processes.

Each instance is deregistered from the load balancer (Classic Load Balancers) or target group (Application Load Balancers and Network Load Balancers) to prevent traffic from being routed to it during the deployment. It is re-registered when the deployment is complete.

For more information about load balancers for AWS CodeDeploy deployments, see [Integrating AWS CodeDeploy with Elastic Load Balancing \(p. 46\)](#).

9. In the **Deployment configuration** list, choose a deployment configuration to control the rate at which instances are deployed to, such as one at a time or all at once. For more information about deployment configurations, see [Working with Deployment Configurations in AWS CodeDeploy](#) (p. 203).

10(Optional) In **Advanced**, configure any options you want to include in the deployment, such as Amazon SNS notification triggers, Amazon CloudWatch alarms, or automatic rollbacks.

For more information, see [Configure Advanced Options for a Deployment Group](#) (p. 228).

11In **Service role ARN**, choose an Amazon Resource Name (ARN) for a service role that trusts AWS CodeDeploy with, at minimum, the trust and permissions described in [Step 3: Create a Service Role for AWS CodeDeploy](#) (p. 21). To get the service role ARN, see [Get the Service Role ARN \(Console\)](#) (p. 24).

12Choose **Create deployment group**.

Create a Deployment Group for a Blue/Green Deployment (Console)

To use the AWS CodeDeploy console to create a deployment group for a blue/green deployment:

Warning

Do not follow these steps if:

- You do not have instances with the AWS CodeDeploy agent installed that you want to replace during the blue/green deployment process. To set up your instances, follow the instructions in [Working with Instances for AWS CodeDeploy](#) (p. 147), and then follow the steps in this topic.
- You want to create an application that uses a custom deployment configuration, but you have not yet created the deployment configuration. Follow the instructions in [Create a Deployment Configuration](#) (p. 207), and then follow the steps in this topic.
- You do not have a service role that trusts AWS CodeDeploy with, at minimum, the trust and permissions described in [Step 3: Create a Service Role for AWS CodeDeploy](#) (p. 21). To create and configure a service role, follow the instructions in [Step 3: Create a Service Role for AWS CodeDeploy](#) (p. 21), and then follow the steps in this topic.
- You have not created a Classic Load Balancer or an Application Load Balancer in Elastic Load Balancing for the registration of the instances in your replacement environment. For more information, see [Set Up a Load Balancer in Elastic Load Balancing for AWS CodeDeploy Deployments](#) (p. 224).

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. From the AWS CodeDeploy menu, choose **Applications**.

3. On the **Applications** page, choose the name of the application for which you want to create a deployment group.

4. Choose **Create deployment group**.

5. In the **Deployment group name** box, type a name that describes the deployment group.

Note

If you want to use the same settings used in another deployment group, specify those settings on this page. The settings you might want to reuse include, for example, the deployment group name, tags, Auto Scaling group names, or deployment configuration.

Although the new and existing deployment group have the same name, AWS CodeDeploy treats them as separate deployment groups, because they are associated with separate applications.

6. Choose **Blue/green deployment**.
7. In **Environment configuration**, choose the method to use to provide instances for your replacement environment:
 - **Automatically copy Auto Scaling group**: AWS CodeDeploy creates an Auto Scaling group by copying one you specify.
 - **Manually provision instances**: You won't specify the instances for your replacement environment until you create a deployment. You must create the instances before you start the deployment. Instead, here you specify the instances you want to replace.
8. Depending on your choice in step 7, do one of the following:
 - If you chose **Automatically copy Auto Scaling group**: In **Auto Scaling group**, choose the name of the Auto Scaling group you want to use as a template for the Auto Scaling group that will be created for the instances in your replacement environment. The number of currently healthy instances in the Auto Scaling group you select will be created in your replacement environment.
 - If you chose **Manually provision instances**: In **Choose the EC2 instances or Auto Scaling groups where the current application revision is deployed**, enter Amazon EC2 tag values or Auto Scaling group names to identify the instances in your original environment (that is, the instances you want to replace or that are running the current application revision).
9. In **Load balancer**, choose the Classic Load Balancer, Application Load Balancer, or Network Load Balancer to use in the registration of instances in your replacement environment during the deployment process.

Note

The instances in your original environment can be, but are not required to be, registered with the load balancer you choose.

For more information about load balancers for AWS CodeDeploy deployments, see [Integrating AWS CodeDeploy with Elastic Load Balancing \(p. 46\)](#).

10. In **Deployment settings**, review the default options for rerouting traffic to the replacement environment, which deployment configuration to use for the deployment, and how instances in the original environment are handled after the deployment.

If you want to change the settings, continue to step 11. Otherwise, skip to step 12.

11. To change the deployment settings for the blue/green deployment, choose **Edit deployment settings**, update any of the following settings, and then choose **Submit**.

Setting	Options
Traffic rerouting	<ul style="list-style-type: none">• Reroute traffic immediately: As soon as instances in the replacement environment are provisioned and the latest application revision is installed on them, they are registered with the load balancer automatically, causing traffic to be rerouted to them. Instances in the original environment are then deregistered.• I will choose whether to reroute traffic: Instances in the replacement environment are not registered with the load balancer unless you manually reroute traffic. If the wait time you specify passes without traffic being rerouted, the deployment status is changed to Stopped.

Setting	Options
Deployment configuration	<p>Choose the rate at which instances in the replacement environment are registered with the load balancer, such as one at a time or all at once.</p> <p>Note After traffic is successfully routed to the replacement environment, instances in the original environment are deregistered all at once no matter which deployment configuration was selected.</p> <p>For more information, see Working with Deployment Configurations in AWS CodeDeploy (p. 203).</p>
Original instances	<ul style="list-style-type: none">• Terminate the original instances in the deployment group: After traffic is rerouted to the replacement environment, the instances that were deregistered from the load balancer are terminated following the wait period you specify.• Keep the original instances in the deployment group running: After traffic is rerouted to the replacement environment, the instances that were deregistered from the load balancer are kept running.

12(Optional) In **Advanced**, configure options you want to include in the deployment, such as Amazon SNS notification triggers, Amazon CloudWatch alarms, or automatic rollbacks.

For information about specifying advanced options in deployment groups, see [Configure Advanced Options for a Deployment Group](#) (p. 228).

13In the **Service role ARN** box, choose an Amazon Resource Name (ARN) for a service role that trusts AWS CodeDeploy with, at minimum, the trust and permissions described in [Step 3: Create a Service Role for AWS CodeDeploy](#) (p. 21). To get the service role ARN, see [Get the Service Role ARN \(Console\)](#) (p. 24).

14Choose **Create deployment group**.

Set Up a Load Balancer in Elastic Load Balancing for AWS CodeDeploy Deployments

Before you run any blue/green deployment, or an in-place deployment for which you want to specify an optional load balancer in the deployment group, you must have created a Classic Load Balancer or Application Load Balancer in Elastic Load Balancing. For blue/green deployments, you use that load balancer to register the instances that make up your replacement environment. Instances in your original environment can optionally be registered with this same load balancer.

To configure a Classic Load Balancer, follow the instructions in [Tutorial: Create a Classic Load Balancer in User Guide for Classic Load Balancers](#). Note the following:

- In **Step 2: Define Load Balancer**, in **Create LB Inside**, choose the same VPC you selected when you created your instances.

- In **Step 5: Register EC2 Instances with Your Load Balancer**, select the instances currently in your deployment group (in-place deployments) or that you have designated to be in your original environment (blue/green deployments).
- In **Step 7: Create and Verify Your Load Balancer**, make a note of the DNS address of your load balancer.

For example, if you named your load balancer `my-load-balancer`, your DNS address would appear in a format such as `my-load-balancer-1234567890.us-east-2.elb.amazonaws.com`.

To configure an Application Load Balancer, follow the instructions in one of the following topics:

- [Create an Application Load Balancer](#)
- [Tutorial: Create an Application Load Balancer Using the AWS CLI](#)

Create a Deployment Group (CLI)

To use the AWS CLI to create a deployment group, call the `create-deployment-group` command, specifying:

- The application name. To view a list of application names, call the `list-applications` command.
- A name for the deployment group. This name must be unique for each application associated with the deployment group.
- Information about the tags, tag groups, or Auto Scaling group names that identify the instances to be included in the deployment group.
- The Amazon Resource Name (ARN) identifier of the service role that allows AWS CodeDeploy to act on behalf of your AWS account when interacting with other AWS services. To get the service role ARN, see [Get the Service Role ARN \(CLI\)](#) (p. 25). For more information about service roles, see [Roles Terms and Concepts](#) in *IAM User Guide*.
- Information about the type of deployment, either an in-place deployment or blue/green deployment, to associate with the deployment group.
- (Optional) The name of an existing deployment configuration. To view a list of deployment configurations, see [View Deployment Configuration Details](#) (p. 207). If not specified, AWS CodeDeploy uses a default deployment configuration.
- (Optional) Commands to create a trigger that pushes notifications about deployment and instance events to those who are subscribed to an Amazon Simple Notification Service topic. For more information, see [Monitoring Deployments with Amazon SNS Event Notifications](#) (p. 278).
- (Optional) Commands to add existing CloudWatch alarms to the deployment group that are activated if a metric specified in an alarm falls below or exceeds a defined threshold.
- (Optional) Commands for a deployment to roll back to the last known good revision when a deployment fails or a CloudWatch alarm is activated.
- For in-place deployments:
 - (Optional) The name of the Classic Load Balancer or Application Load Balancer in Elastic Load Balancing that manages traffic to the instances during the deployment processes.
- For blue/green deployments:
 - Configuration of the blue/green deployment process:
 - How new instances in the replacement environment are provisioned.
 - Whether to reroute traffic to the replacement environment immediately or wait a specified period for traffic to be rerouted manually.
 - Whether instances in the original environment should be terminated.
 - The name of the Classic Load Balancer or Application Load Balancer in Elastic Load Balancing to be used for instances registered in the replacement environment.

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 associated with an application.

Topics

- [View Deployment Group Details \(Console\)](#) (p. 226)
- [View Deployment Group Details \(CLI\)](#) (p. 226)

View Deployment Group Details (Console)

To use the AWS CodeDeploy console to view deployment group details:

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. If the **Applications** page does not appear, on the AWS CodeDeploy menu, choose **Applications**.
3. On the **Applications** page, choose the application name associated with the deployment group.

Note

If no entries are displayed, make sure the correct region is selected. On the navigation bar, in the region selector, choose one of the regions listed in [Region and Endpoints](#) in the *AWS General Reference*. AWS CodeDeploy is supported in these regions only.

4. To view details about an individual deployment group, in **Deployment groups**, choose the arrow next to the deployment group.

View Deployment Group Details (CLI)

To use the AWS CLI to view deployment group details, call either the `get-deployment-group` command or the `list-deployment-groups` command.

To view details about a single deployment group, call the `get-deployment-group` command, specifying:

- The application name associated with the deployment group. To get the application name, call the `list-applications` command.
- The deployment group name. To get the deployment group name, call the `list-deployment-groups` command.

To view a list of deployment group names, call the `list-deployment-groups` command, specifying the application name associated with the deployment groups. To get the application name, call the `list-applications` command.

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 a deployment group.

Warning

Do not use these steps if you want the deployment group to use a not-yet-created custom deployment group. Instead, follow the instructions in [Create a Deployment Configuration \(p. 207\)](#), and then return to this topic. Do not use these steps if you want the deployment group to use a different, not-yet-created service role. The service role must trust AWS CodeDeploy with, at minimum, the permissions described in [Step 3: Create a Service Role for AWS CodeDeploy \(p. 21\)](#). To create and configure a service role with the correct permissions, follow the instructions in [Step 3: Create a Service Role for AWS CodeDeploy \(p. 21\)](#), and then return to this topic.

Topics

- [Change Deployment Group Settings \(Console\) \(p. 227\)](#)
- [Change Deployment Group Settings \(CLI\) \(p. 228\)](#)

Change Deployment Group Settings (Console)

To use the AWS CodeDeploy console to change deployment group settings:

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#).

2. Choose **Applications**.
3. In the list of applications, choose the application that is associated with the deployment group you want to change.

Note

If no entries are displayed, make sure the correct region is selected. On the navigation bar, in the region selector, choose one of the regions listed in [Region and Endpoints](#) in the *AWS General Reference*. AWS CodeDeploy is supported in these regions only.

4. On the **Application details** page, in **Deployment groups**, choose the button next to the deployment group you want to change.
5. On the **Actions** menu, choose **Edit**.
6. Revise the deployment group options as needed.

For information about deployment group components, see [Create a Deployment Group with AWS CodeDeploy \(p. 220\)](#).

7. If you want to deploy the last successful revision to the deployment group, select **Deploy changes made to *deployment group name***, and then choose **Save**. When prompted, choose **Deploy**. AWS CodeDeploy updates the deployment group's information, starts a deployment of the last successful revision to the deployment group based on changes you specified, and displays the **Deployments** page.

Note

The **Deploy changes made to *deployment group name*** check box appears only if there was a successful deployment to this deployment group.

8. If you want to update the deployment group's information with your changes, but do not want to deploy any applications to the deployment group at this time, clear **Deploy changes made to *deployment group name***, and then choose **Save**. AWS CodeDeploy updates the deployment group's information, but does not deploy any applications to the deployment group.

Change Deployment Group Settings (CLI)

To use the AWS CLI to change deployment group settings, call the [update-deployment-group](#) command, specifying:

- For EC2/On-Premises and AWS Lambda deployments:
 - The application name. To view a list of application names, call the [list-applications](#) command.
 - The current deployment group name. To view a list of deployment group names, call the [list-deployment-groups](#) command.
 - (Optional) A different deployment group name.
 - (Optional) A different Amazon Resource Name (ARN) that corresponds to a service role that allows AWS CodeDeploy to act on your AWS account's behalf when interacting with other AWS services. To get the service role ARN, see [Get the Service Role ARN \(CLI\)](#) (p. 25). For more information about service roles, see [Roles Terms and Concepts](#) in *IAM User Guide*.
 - (Optional) The name of the deployment configuration. To view a list of deployment configurations, see [View Deployment Configuration Details](#) (p. 207). (If not specified, AWS CodeDeploy uses a default deployment configuration.)
 - (Optional) Commands to add one or more existing CloudWatch alarms to the deployment group that are activated if a metric specified in an alarm falls below or exceeds a defined threshold.
 - (Optional) Commands for a deployment to roll back to the last known good revision when a deployment fails or a CloudWatch alarm is activated.
 - (Optional) Commands to create or update a trigger that publishes to a topic in Amazon Simple Notification Service, so that subscribers to that topic receive notifications about deployment and instance events in this deployment group. For information, see [Monitoring Deployments with Amazon SNS Event Notifications](#) (p. 278).
- For EC2/On-Premises deployments only:
 - (Optional) Replacement tags or tag groups that uniquely identify the instances to be included in the deployment group.
 - (Optional) The names of replacement Auto Scaling groups to be added to the deployment group.

Configure Advanced Options for a Deployment Group

When you create or update a deployment group, you can configure a number of options to provide more control and oversight over the deployments for that deployment group.

Use the information on this page to help you configure advanced options when you work with deployment groups in the following topics:

- [Create an Application with AWS CodeDeploy](#) (p. 209)
- [Create a Deployment Group with AWS CodeDeploy](#) (p. 220)
- [Change Deployment Group Settings with AWS CodeDeploy](#) (p. 227)

Amazon SNS notification triggers: You can add triggers to an AWS CodeDeploy deployment group to receive notifications about events related to deployments in that deployment group. These notifications

are sent to recipients who are subscribed to an Amazon SNS topic you have made part of the trigger's action.

You must have already set up the Amazon SNS topic to which this trigger will point, and AWS CodeDeploy must have permission to publish to the topic from this deployment group. If you have not yet completed these setup steps, you can add triggers to the deployment group later.

If you want to create a trigger now to receive notifications about deployment events in the deployment group for this application, choose **Create trigger**.

If your deployment is to an Amazon EC2 instance, you can create notifications for and receive notifications about instances.

For more information, see [Monitoring Deployments with Amazon SNS Event Notifications \(p. 278\)](#).

Amazon CloudWatch alarms: You can create a CloudWatch alarm that watches a single metric over a time period you specify and performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods. For an Amazon EC2 deployment, you can create an alarm for an instance or Amazon EC2 Auto Scaling group that you are using in your AWS CodeDeploy operations. For an AWS Lambda deployment, you can create an alarm for errors in a Lambda function.

You can configure a deployment to stop when an Amazon CloudWatch alarm detects that a metric has fallen below or exceeded a defined threshold.

You must have already created the alarm in CloudWatch before you can add it to a deployment group.

1. To add alarm monitoring to the deployment group, choose **Add alarm**.
2. In **Alarm name**, type the name of a CloudWatch alarm you have already set up to monitor this deployment.

You must enter the CloudWatch alarm exactly as it was created in CloudWatch. To view a list of alarms, open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>, and then choose **ALARM**.

Additional options:

- If you want deployments to proceed without taking into account alarms you have added, choose **Ignore alarm configuration**.

This choice is useful when you want to temporarily deactivate alarm monitoring for a deployment group without having to add the same alarms again later.

- (Optional) If you want deployments to proceed in the event that AWS CodeDeploy is unable to retrieve alarm status from Amazon CloudWatch, choose **Continue deployments even if alarm status is unavailable**.

Note

This option corresponds to **ignorePollAlarmFailure** in the [AlarmConfiguration](#) object in the AWS CodeDeploy API.

For more information, see [Monitoring Deployments with CloudWatch Alarms in AWS CodeDeploy \(p. 273\)](#).

Automatic rollbacks: You can configure a deployment group or deployment to automatically roll back when a deployment fails or when a monitoring threshold you specify is met. In this case, the last known good version of an application revision is deployed. You can configure optional settings for a deployment group when you use the console to create an application, create a deployment group, or update a deployment group. When you create a new deployment, you can also choose to override the automatic rollback configuration that were specified for the deployment group.

- You can enable deployments to roll back to the most recent known good revision when something goes wrong by choosing one or both of the following:
 - **Roll back when a deployment fails.** AWS CodeDeploy will redeploy the last known good revision as a new deployment.
 - **Roll back when alarm thresholds are met.** If you added an alarm to this application in the previous step, AWS CodeDeploy will redeploy the last known good revision when one or more of the specified alarms is activated.

Note

To temporarily ignore a rollback configuration, choose **Disable rollbacks**. This choice is useful when you want to temporarily disable automatic rollbacks without having to set up the same configuration again later.

For more information, see [Redeploy and Roll Back a Deployment with AWS CodeDeploy \(p. 259\)](#).

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 associated with your AWS account.

Warning

If you delete a deployment group, all details associated with that deployment group will also be deleted from AWS CodeDeploy. The instances used in the deployment group will remain unchanged. This action cannot be undone.

Topics

- [Delete a Deployment Group \(Console\) \(p. 230\)](#)
- [Delete a Deployment Group \(CLI\) \(p. 230\)](#)

Delete a Deployment Group (Console)

To use the AWS CodeDeploy console to delete a deployment group:

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#).

2. On the AWS CodeDeploy menu, choose **Applications**.
3. In the list of applications, choose the name of the application associated with the deployment group.
4. On the **Application details** page, in **Deployment groups**, choose the button next to the deployment group you want to delete.
5. On the **Actions** menu, choose **Delete**.
6. When prompted, type the name of the deployment group to confirm you want to delete it, and then choose **Delete**.

Delete a Deployment Group (CLI)

To use the AWS CLI to delete a deployment group, call the `delete-deployment-group` command, specifying:

- The name of the application associated with the deployment group. To view a list of application names, call the [list-applications](#) command.
- The name of the deployment group associated with the application. To view a list of deployment group names, call the [list-deployment-groups](#) command.

Working with Application Revisions for AWS CodeDeploy

In AWS CodeDeploy, a revision contains a version of the source files AWS CodeDeploy will deploy to your instances or scripts AWS CodeDeploy will run on your instances.

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 \(p. 232\)](#)
- [Add an Application Specification File to a Revision for AWS CodeDeploy \(p. 233\)](#)
- [Choose an AWS CodeDeploy Repository Type \(p. 237\)](#)
- [Push a Revision for AWS CodeDeploy to Amazon S3 \(p. 238\)](#)
- [View Application Revision Details with AWS CodeDeploy \(p. 241\)](#)
- [Register an Application Revision in Amazon S3 with AWS CodeDeploy \(p. 242\)](#)

Plan a Revision for AWS CodeDeploy

Good planning makes deploying revisions to instances much easier.

For deployments to an AWS Lambda compute platform, a revision is the same as the AppSpec file. The following information does not apply. For more information, see [Add an Application Specification File to a Revision for AWS CodeDeploy \(p. 233\)](#)

For deployments to an EC2/On-Premises compute platform, start by creating an empty root directory (folder) on the development machine. This is where you will store the source files (such as text and binary files, executables, packages, and so on) to be deployed to the instances or scripts to be run on the instances.

For example, at the `/tmp/` root folder in Linux, macOS, 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
```

The root folder should also include an application specification file (AppSpec file), as shown here. For more information, see [Add an Application Specification File to a Revision for AWS CodeDeploy \(p. 233\)](#).

Add an Application Specification File to a Revision for AWS CodeDeploy

This topic shows how to add an AppSpec file to your deployment. It also includes templates to create an AppSpec file for an AWS Lambda and EC2/On-Premises deployment.

Add an AppSpec File for an AWS Lambda Deployment

For a deployment to an AWS Lambda compute platform:

- The AppSpec file contains instructions about the Lambda functions to be deployed and used for deployment validation.
- A revision is the same as an AppSpec file.
- An AppSpec file can be written using JSON or YAML.
- An AppSpec file can be saved as a text file or entered directly into a console AppSpec editor when creating a deployment. For more information, see [Create an AWS Lambda Compute Platform Deployment \(Console\)](#) (p. 247).

To create an AppSpec file:

1. Copy the JSON or YAML template into a text editor or into the AppSpec editor in the console.
2. Modify the template as needed.
3. Use a JSON or YAML validator to validate your AppSpec file. If you use the AppSpec editor, the file is validated when you choose **Deploy**.
4. If you use a text editor, save the file. If you use the AWS CLI to create your deployment, reference the AppSpec file if it's on your hard drive or in an Amazon S3 bucket. If you use the console, you must push your AppSpec file to Amazon S3.
5. (Optional) If you use the AppSpec editor, choose **Save as text file** to save the AppSpec file to your hard drive.

YAML AppSpec File Template with Instructions

For information about lifecycle events to use in the hooks section, see [AppSpec 'hooks' Section for an AWS Lambda Deployment](#) (p. 317).

```
# This is an appspec.yml template file for use with an AWS Lambda deployment in 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
# https://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html
version: 0.0
# In the Resources section specify the name, alias,
# target version, and (optional) the current version of your AWS Lambda function.
Resources:
  - MyFunction: # Replace "MyFunction" with the name of your Lambda function
    Type: AWS::Lambda::Function
    Properties:
      Name: "" # Specify the name of your Lambda function
```

```
Alias: "" # Specify the alias for your Lambda function
CurrentVersion: "" # Specify the current version of your Lambda function
TargetVersion: "" # Specify the version of your Lambda function to deploy
# (Optional) In the Hooks section, specify a validation Lambda function to run during
# a lifecycle event. Replace "LifeCycleEvent" with BeforeAllowTraffic
# or AfterAllowTraffic.
Hooks:
  - LifeCycleEvent: "" # Specify a Lambda validation function between double-quotes.
```

JSON AppSpec File Template

In the following template, replace "MyFunction" with the name of your AWS Lambda function. In the optional Hooks section, replace the lifecycle events with BeforeAllowTraffic or AfterAllowTraffic.

For information about lifecycle events to use in the Hooks section, see [AppSpec 'hooks' Section for an AWS Lambda Deployment \(p. 317\)](#).

```
{
  "version": 0.0,
  "Resources": [{
    "MyFunction": {
      "Type": "AWS::Lambda::Function",
      "Properties": {
        "Name": "",
        "Alias": "",
        "CurrentVersion": "",
        "TargetVersion": ""
      }
    }
  ]},
  "Hooks": [{
    "LifeCycleEvent": ""
  }
]
```

Add an AppSpec File for an EC2/On-Premises Deployment

Without an AppSpec file, AWS CodeDeploy cannot map the source files in your application revision to their destinations or run scripts for your deployment to an EC2/On-Premises compute platform, .

Each revision must contain only one AppSpec file.

To add an AppSpec file to a revision:

1. Copy the template into a text editor.
2. Modify the template as needed.
3. Use a YAML validator to check the validity of your AppSpec file.
4. Save the file as `appspec.yml` in the root directory of the revision.
5. Run one of the following commands to verify that you have placed your AppSpec file in the root directory:
 - For Linux, macOS, or Unix:

```
find /path/to/root/directory -name appspec.yml
```

There will be no output if the AppSpec file is not found there.

- For Windows:

```
dir path\to\root\directory\appspec.yml
```

A **File Not Found** error will be displayed if the AppSpec file is not stored there.

6. Push the revision to Amazon S3 or GitHub.

For instructions, see [Push a Revision for AWS CodeDeploy to Amazon S3 \(p. 238\)](#).

AppSpec File Template with Instructions

Note

Deployments to Windows Server instances do not support the `runas` element. If you are deploying to Windows Server instances, do not include it in your AppSpec file.

```
# This is an appspec.yml template file for use with an EC2/On-Premises deployment in 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
# https://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html
version: 0.0
# Specify "os: linux" if this revision targets Amazon Linux,
# Red Hat Enterprise Linux (RHEL), 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 Amazon EC2 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 Amazon EC2 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, Ubuntu Server, or RHEL 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 Amazon EC2 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 Amazon EC2 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 Amazon EC2 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, Ubuntu Server, or RHEL 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 ApplicationStart 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:
```

Choose an AWS CodeDeploy Repository Type

The storage location for files required by AWS CodeDeploy is called a *repository*. Use of a repository depends on which compute platform your deployment uses.

- **EC2/On-Premises:** To deploy your application code to one or more instances, your code must be bundled into an archive file and placed in a repository where AWS CodeDeploy can access it during the deployment process. You bundle your deployable content and an AppSpec file into an archive file, and then upload it to one of the repository types supported by AWS CodeDeploy.
- **AWS Lambda:** Deployments require an AppSpec file, which can be accessed during a deployment in one of the following ways:
 - From an Amazon S3 bucket.
 - From text typed directly into the AppSpec editor in the console. For more information, see [Create an AWS Lambda Compute Platform Deployment \(Console\)](#) (p. 247).
 - If you use the AWS CLI, you can reference an AppSpec file that is on your hard drive or on a network drive. For more information, see [Create an AWS Lambda Compute Platform Deployment \(CLI\)](#) (p. 251).

AWS CodeDeploy currently supports the following repository types:

Amazon S3	<p>Amazon Simple Storage Service (Amazon S3) is the AWS solution for secure, scalable object storage. Amazon S3 stores data as objects in <i>buckets</i>. An object consists of a file and, optionally, any metadata that describes that file.</p> <p>To store an object in Amazon S3, you upload the file you want to store to a bucket. When you upload a file, you can set permissions and metadata on the object.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Create a Bucket in Amazon S3• Push a Revision for AWS CodeDeploy to Amazon S3 (p. 238)• Automatically Deploy from Amazon S3 Using AWS CodeDeploy
------------------	---

GitHub	<p>(EC2/On-Premises deployment only) You can store your application revisions in GitHub repositories. You can trigger a deployment from a GitHub repository whenever the source code in that repository is changed.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Integrating AWS CodeDeploy with GitHub (p. 53)• Tutorial: Use AWS CodeDeploy to Deploy an Application from GitHub (p. 112)• Automatically Deploy from GitHub Using AWS CodeDeploy
Bitbucket	<p>(EC2/On-Premises deployment only) You can push code to Amazon EC2 instances directly from the Bitbucket UI to any of your deployment groups without having to sign in to your continuous integration (CI) platform or Amazon EC2 instances to run a manual deployment process. Bitbucket first pushes the code to an Amazon S3 bucket you have specified, and from there deploys the code. After the initial setup to support this process is complete, however, the code you push from Bitbucket is automatically deployed to your instances without any intermediate steps.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Atlassian Bitbucket Support for AWS CodeDeploy

Note

An AWS Lambda deployment works with an Amazon S3 repository only.

Push a Revision for AWS CodeDeploy to Amazon S3

After you plan your revision as described in [Plan a Revision for AWS CodeDeploy \(p. 232\)](#) and add an AppSpec file to the revision as described in [Add an Application Specification File to a Revision for AWS CodeDeploy \(p. 233\)](#), you are ready to bundle the component files and push the revision to Amazon S3. For deployments to Amazon EC2 instances, after you push the revision, you can use AWS CodeDeploy to deploy the revision from Amazon S3 to the instances.

Note

AWS CodeDeploy can also be used to deploy revisions that have been pushed to GitHub. For more information, see your GitHub documentation.

We assume you have already followed the instructions in [Getting Started with AWS CodeDeploy \(p. 19\)](#) to set up the AWS CLI. This is especially important for calling the **push** command described later.

Be sure you have an Amazon S3 bucket. Follow the instructions in [Create a Bucket](#).

If your deployment is to Amazon EC2 instances, then 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 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 two 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.

You must have permissions to upload to the Amazon S3 bucket. You can specify these permissions through an Amazon S3 bucket policy. For example, in the following Amazon S3 bucket policy, using the wildcard character (*) allows AWS account 111122223333 to upload files to any directory in the Amazon S3 bucket named `codedeploydemobucket`:

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

To view your AWS account ID, see [Finding Your AWS Account ID](#).

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 have, at minimum, permissions to upload the revision to each target Amazon S3 bucket. For example, the following policy allows the IAM user to upload revisions anywhere in 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 [Working with Policies](#).

Push a Revision Using the AWS CLI

Note

The `push` command bundles application artifacts and an AppSpec file into a revision. The file format of this revision is a compressed ZIP file. The command cannot be used with an AWS Lambda deployment because it expects a revision that is a JSON-formatted or YAML-formatted AppSpec file.

Call the **push** command to bundle and push the revision for a deployment. Its parameters are:

- **--application-name:** (string) Required. The name of the AWS CodeDeploy application to be associated with the application revision.
- **--s3-location:** (string) Required. Information about the location of the application revision to be uploaded to Amazon S3. You must specify an Amazon S3 bucket and a key. The key is the name of the revision. AWS CodeDeploy zips the content before it is uploaded. Use the format `s3://your-S3-bucket-name/your-key.zip`.
- **--ignore-hidden-files** or **--no-ignore-hidden-files:** (boolean) Optional. Use the `--no-ignore-hidden-files` flag (the default) to bundle and upload hidden files to Amazon S3. Use the `--ignore-hidden-files` flag to not bundle and upload hidden files to Amazon S3.
- **--source** (string) Optional. The location of the content to be deployed and the AppSpec file on the development machine to be zipped and uploaded to Amazon S3. The location is specified as a path relative to the current directory. If the relative path is not specified or if a single period is used for the path ("."), the current directory is used.
- **--description** (string) Optional. A comment that summarizes the application revision. If not specified, the default string "Uploaded by AWS CLI 'time' UTC" is used, where 'time' is the current system time in Coordinated Universal Time (UTC).

You can use the AWS CLI to push a revision for an Amazon EC2 deployment. An example push command looks like this:

In Linux, macOS, 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 revision for the  
  application WordPress_App" --ignore-hidden-files --s3-location s3://codedeploydemobucket/  
WordPressApp.zip --source .
```

This command does the following:

- Associates the bundled files with an application named `WordPress_App`.
- Attaches a description to the revision.
- Ignores hidden files.
- Names the revision `WordPressApp.zip` and pushes it to a bucket named `codedeploydemobucket`.
- Bundles all files in the root directory into the revision.

After the push is successful, you can use the AWS CLI or the AWS CodeDeploy console to deploy the revision from Amazon S3. To deploy this revision with the AWS CLI:

In Linux, macOS, or Unix:

```
aws deploy create-deployment \  
  --application-name WordPress_App \  
  --deployment-config-name your-deployment-config-name \  
  --deployment-group-name your-deployment-group-name \  
  --revision-id your-revision-id
```



```
--s3-location bucket=codedeploydemobucket,key=WordPressApp.zip,bundleType=zip
```

In Windows:

```
aws deploy create-deployment --application-name WordPress_App --deployment-config-  
name your-deployment-config-name --your-deployment-group-name your-deployment-group-name --  
s3-location bucket=codedeploydemobucket,key=WordPressApp.zip,bundleType=zip
```

For more information, see [Create a Deployment with AWS CodeDeploy \(p. 245\)](#).

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 specified application.

For information about registering a revision, see [Register an Application Revision in Amazon S3 with AWS CodeDeploy \(p. 242\)](#).

Topics

- [View Application Revision Details \(Console\) \(p. 241\)](#)
- [View Application Revision Details \(CLI\) \(p. 241\)](#)

View Application Revision Details (Console)

To view application revision details:

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#).

2. On the AWS CodeDeploy menu, choose **Applications**.
3. On the **Applications** page, choose the name of the application with revision details you want to view.

Note

If no entries are displayed, make sure the correct region is selected. On the navigation bar, in the region selector, choose one of the regions listed in [Region and Endpoints](#) in the *AWS General Reference*. AWS CodeDeploy is supported in these regions only.

4. On the **Application details** page, under **Revisions**, review the list of revisions that are registered for the application. Choose the arrow next to a revision for more details.

View Application Revision Details (CLI)

To use the AWS CLI to view an application revision, call either the **get-application-revision** command or the **list-application-revisions** command.

Note

References to GitHub apply to deployments to EC2/On-Premises deployments only. Revisions for AWS Lambda deployments do not work with GitHub.

To view details about a single application revision, call the [get-application-revision](#) command, specifying:

- The 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, ETag, or both were specified 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 application name. To get the application name, call the [list-applications](#) command.
- Optionally, to view details for Amazon S3 application revisions only, the Amazon S3 bucket name containing the revisions.
- Optionally, to view details for Amazon S3 application revisions only, a prefix string to limit the search to Amazon S3 application revisions. (If not specified, AWS CodeDeploy will list all matching Amazon S3 application 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, 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 only those revisions stored in Amazon S3. You cannot list only those revisions stored in GitHub.

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, you don't need to register the revision. However, if you upload a revision to Amazon S3 through other means and want the revision to appear in the AWS CodeDeploy console or through the AWS CLI, follow these steps to register the revision first.

If you've pushed an application revision to a GitHub repository and want the revision to appear in the AWS CodeDeploy console or through the AWS CLI, you must also follow these steps.

You can use only the AWS CLI or the AWS CodeDeploy APIs to register application revisions in Amazon S3 or GitHub.

Topics

- [Register a revision in Amazon S3 with AWS CodeDeploy \(CLI\)](#) (p. 242)
- [Register a revision in GitHub with AWS CodeDeploy \(CLI\)](#) (p. 243)

Register a revision in Amazon S3 with AWS CodeDeploy (CLI)

1. Upload the revision to Amazon S3.

2. Call the [register-application-revision](#) command, specifying:

- The application name. To view a list of application names, call the [list-applications](#) command.
- Information about the revision to be registered:
 - The name of the Amazon S3 bucket that contains the revision.
 - The name and file type of the uploaded revision. For AWS Lambda deployments, the revision is an AppSpec file written in JSON or YAML. For EC2/On-Premises deployments, the revision contains a version of the source files that AWS CodeDeploy will deploy to your instances or scripts that AWS CodeDeploy will run on your instances.

Note

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

- (Optional) The revision's Amazon S3 version identifier. (If the version identifier is not specified, AWS CodeDeploy will use the most recent version.)
- (Optional) The revision's ETag. (If the ETag is not specified, AWS CodeDeploy will skip object validation.)
- (Optional) Any description you want to associate with the revision.

Information about a revision in Amazon S3 can be specified on the command line, using this syntax as part of the **register-application-revision** call. (*version* and *eTag* are optional.)

For a revision file for an EC2/On-Premises deployment:

```
--s3-location bucket=string,key=string,bundleType=tar|tgz|zip,version=string,eTag=string
```

For a revision file for an AWS Lambda deployment:

```
--s3-location bucket=string,key=string,bundleType=JSON|YAML,version=string,eTag=string
```

Register a revision in GitHub with AWS CodeDeploy (CLI)

Note

AWS Lambda deployments do not work with GitHub.

1. Upload the revision to your GitHub repository.
2. Call the [register-application-revision](#) command, specifying:
 - The application name. To view a list of 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 (/), followed by the repository name.
 - The ID of the commit that references the revision in the repository.
 - (Optional) Any description you want to associate with the revision.

Information about a revision in GitHub can be specified on the command line, using this syntax as part of the **register-application-revision** call:

```
--github-location repository=string,commitId=string
```

Working with Deployments in AWS CodeDeploy

In AWS CodeDeploy, a deployment is the process, and the components involved in the process, of installing content on one or more instances. This content can consist of code, web and configuration files, executables, packages, scripts, and so on. AWS CodeDeploy deploys content that is stored in a source repository, according to the configuration rules you specify.

AWS CodeDeploy provides two deployment type options, in-place deployments and blue/green deployments.

- **In-place deployment:** The application on each instance in the deployment group is stopped, the latest application revision is installed, and the new version of the application is started and validated. You can use a load balancer so that each instance is deregistered during its deployment and then restored to service after the deployment is complete. Only deployments that use the EC2/On-Premises compute platform can use in-place deployments. For more information about in-place deployments, see [Overview of an In-Place Deployment \(p. 4\)](#).
- **Blue/green deployment:** The behavior of your deployment depends on which compute platform you use:
 - **Blue/green on an EC2/On-Premises compute platform:** The instances in a deployment group (the original environment) are replaced by a different set of instances (the replacement environment) using these steps:
 - Instances are provisioned for the replacement environment.
 - The latest application revision is installed on the replacement instances.
 - An optional wait time occurs for activities such as application testing and system verification.
 - Instances in the replacement environment are registered with an Elastic Load Balancing load balancer, causing traffic to be rerouted to them. Instances in the original environment are deregistered and can be terminated or kept running for other uses.

Note

When using an EC2/On-Premises compute platform, blue/green deployments work with Amazon EC2 instances only.

- **Blue/green on an AWS Lambda compute platform:** Traffic is shifted from your current serverless environment to one with your updated Lambda function versions. You can specify Lambda functions that perform validation tests and choose the way in which the traffic shift occurs. All AWS Lambda compute platform deployments are blue/green deployments. For this reason, you do not need to specify a deployment type.

For more information about blue/green deployments, see [Overview of a Blue/Green Deployment \(p. 5\)](#).

For information about automatically deploying from Amazon S3, see [Automatically Deploy from Amazon S3 Using AWS CodeDeploy](#).

Topics

- [Create a Deployment with AWS CodeDeploy \(p. 245\)](#)
- [View Deployment Details with AWS CodeDeploy \(p. 254\)](#)
- [View Log Data for AWS CodeDeploy Deployments \(p. 255\)](#)
- [Stop a Deployment with AWS CodeDeploy \(p. 258\)](#)

- [Redeploy and Roll Back a Deployment with AWS CodeDeploy \(p. 259\)](#)
- [Deploy an Application in a Different AWS Account \(p. 262\)](#)
- [Use the AWS CodeDeploy Agent to Validate a Deployment Package on a Local Machine \(p. 265\)](#)

Create a Deployment with AWS CodeDeploy

You can use the AWS CodeDeploy console, the AWS CLI, or the AWS CodeDeploy APIs to create a deployment that installs application revisions you have already pushed to Amazon S3 or, if your deployment is to an EC2/On-Premises compute platform, GitHub, on the instances in a deployment group.

The process for creating a deployment depends on the compute platform used by your deployment.

Topics

- [Deployment Prerequisites \(p. 245\)](#)
- [Create an AWS Lambda Compute Platform Deployment \(Console\) \(p. 247\)](#)
- [Create an EC2/On-Premises Compute Platform Deployment \(Console\) \(p. 248\)](#)
- [Create an AWS Lambda Compute Platform Deployment \(CLI\) \(p. 251\)](#)
- [Create an EC2/On-Premises Compute Platform Deployment \(CLI\) \(p. 252\)](#)

Deployment Prerequisites

Make sure the following steps are complete before you start a deployment.

Deployment Prerequisites on an AWS Lambda Compute Platform

- Create an application that includes at least one deployment group. For information, see [Create an Application with AWS CodeDeploy \(p. 209\)](#) and [Create a Deployment Group with AWS CodeDeploy \(p. 220\)](#).
- Prepare the application revision, also known as the AppSpec file, that specifies the Lambda function version you want to deploy. The AppSpec file can also specify Lambda functions to validate your deployment. For more information see [Working with Application Revisions for AWS CodeDeploy \(p. 232\)](#).
- If you want to use a custom deployment configuration for your deployment, create it before starting the deployment process. For information, see [Create a Deployment Configuration \(p. 207\)](#).

Deployment Prerequisites on an EC2/On-Premises Compute Platform

- For an in-place deployment, create or configure the instances you want to deploy to. For information, see [Working with Instances for AWS CodeDeploy \(p. 147\)](#). For a blue/green deployment, you either have an existing Auto Scaling group to use as a template for your replacement environment, or you have one or more instances or Auto Scaling groups that you specify as your original environment. For more information, see [Tutorial: Use AWS CodeDeploy to Deploy an Application to an Auto Scaling Group \(p. 96\)](#) and [Integrating AWS CodeDeploy with Auto Scaling \(p. 44\)](#).
- Create an application that includes at least one deployment group. For information, see [Create an Application with AWS CodeDeploy \(p. 209\)](#) and [Create a Deployment Group with AWS CodeDeploy \(p. 220\)](#).

- Prepare the application revision that you want to deploy to the instances in your deployment group. For information, see [Working with Application Revisions for AWS CodeDeploy \(p. 232\)](#).
- If you want to use a custom deployment configuration for your deployment, create it before starting the deployment process. For information, see [Create a Deployment Configuration \(p. 207\)](#).
- If you are deploying your application revision from an Amazon S3 bucket, the bucket is in the same AWS region as the instances in your deployment group.
- If you are deploying your application revision from an Amazon S3 bucket, an Amazon S3 bucket policy has been applied to the bucket. This policy grants your instances the permissions required to download the application revision.

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 in 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"
        ]
      }
    }
  ]
}
```

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 in 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](#).

- If you are creating a blue/green deployment, or you have specified an optional Classic Load Balancer, Application Load Balancer, or Network Load Balancer in the deployment group for an in-place deployment, you have created a VPC using Amazon VPC that contains at least two subnets. (AWS

CodeDeploy uses Elastic Load Balancing, which requires all instances in a load balancer group to be in a single VPC.)

If you have not created a VPC yet, see the [Amazon VPC Getting Started Guide](#).

- If you are creating a blue/green deployment, you have configured a Classic Load Balancer, Application Load Balancer, or Network Load Balancer in Elastic Load Balancing and used it to register the instances that make up your original environment.

Note

The instances in your replacement environment will be registered with the load balancer later.

To configure a Classic Load Balancer, complete the steps in [Tutorial: Create a Classic Load Balancer in User Guide for Classic Load Balancers](#). As you do, make note of the following:

- In **Step 2: Define Load Balancer**, in **Create LB Inside**, choose the same VPC you selected when you created your instances.
- In **Step 5: Register EC2 Instances with Your Load Balancer**, select the instances in your original environment.
- In **Step 7: Create and Verify Your Load Balancer**, make a note of the DNS address of your load balancer.

For example, if you named your load balancer `my-load-balancer`, your DNS address would appear in a format such as `my-load-balancer-1234567890.us-east-2.elb.amazonaws.com`.

When you paste the DNS name into the address field of an Internet-connected web browser, you should see the application you have deployed for your original environment.

To configure an Application Load Balancer, follow the instructions in one of the following topics:

- [Create an Application Load Balancer](#)
- [Tutorials: Create an Application Load Balancer Using the AWS CLI](#)

To configure a Network Load Balancer, follow the instructions in one of the following topics:

- [Create a Network Load Balancer](#)
- [Tutorials: Create a Network Load Balancer Using the AWS CLI](#)

Create an AWS Lambda Compute Platform Deployment (Console)

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. Do one of the following:
 - On the AWS CodeDeploy menu, choose **Deployments**, and then choose **Create deployment**.
 - On the AWS CodeDeploy menu, choose **Applications**, and then choose the name of the AWS Lambda application you want to deploy a revision to. You can look in the Compute platform column to identify AWS Lambda applications. On the **Application details**, page, select the button for the deployment group you want to deploy a revision to. On the **Actions** menu, choose **Deploy new revision**.
3. In the **Application** list, choose the name of the application you want to use for this deployment. After selecting your application, make sure the Compute platform under the **Application** list says AWS Lambda.

4. In the **Deployment group** list, choose the name of the deployment group associated with the application.
5. Notice the **Deployment type** label. AWS Lambda deployments are all blue/green.
6. Next to **Revision location** choose where your revision is located:
 - **My revision is stored in Amazon S3** — For information, see [Specify Information About a Revision Stored in an Amazon S3 Bucket \(p. 250\)](#), and then return to step 6.
 - **I will use the AppSpec editor** — Select either JSON or YAML, then type your AppSpec file into the provided editor. You may save the AppSpec file you type in by selecting **Save as text file**. When you click **Deploy** at the end of these steps you will receive an error if your JSON or YAML is not valid. For more information about creating an AppSpec file, see [Add an Application Specification File to a Revision for AWS CodeDeploy \(p. 233\)](#).
7. (Optional) In the **Deployment description** box, type a description for this deployment.
8. In the **Deployment configuration** list, choose a deployment configuration to control how traffic is shifted to the Lambda function version.

For more information, see [Deployment Configurations on an AWS Lambda Compute Platform \(p. 205\)](#)

9. (Optional) In **Rollback configuration overrides**, you can specify different automatic rollback options for this deployment than were specified for the deployment group, if any.

Note

For information about rollbacks in AWS CodeDeploy, see [Redeployments and Deployment Rollbacks \(p. 12\)](#) and [Redeploy and Roll Back a Deployment with AWS CodeDeploy \(p. 259\)](#).

Choose from the following:

- **Roll back when a deployment fails** — AWS CodeDeploy will redeploy the last known good revision as a new deployment.
 - **Roll back when alarm thresholds are met** — If alarms were added to the deployment group, AWS CodeDeploy will redeploy the last known good revision when one or more of the specified alarms is activated.
 - **Disable rollbacks** — Do not perform rollbacks for this deployment.
10. Choose **Deploy**.

To track the status of your deployment, see [View Deployment Details with AWS CodeDeploy \(p. 254\)](#).

Create an EC2/On-Premises Compute Platform Deployment (Console)

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy \(p. 19\)](#).

2. Do one of the following:
 - On the AWS CodeDeploy menu, choose **Deployments**, and then choose **Create deployment**.
 - On the AWS CodeDeploy menu, choose **Applications**, and then choose the name of the EC2/On-Premises application you want to deploy a revision to. You can look in the **Compute platform** column to identify EC2/On-Premises applications. On the **Application details**, page, select the

button for the deployment group you want to deploy a revision to. On the **Actions** menu, choose **Deploy new revision**.

3. In the **Application** list, choose the name of the application you want to use for this deployment. Make sure EC2/On-Premises is displayed for the **Compute platform**.
4. In the **Deployment group** list, choose the name of the deployment group associated with the application.
5. Next to **Repository type**, choose the repository type your revision is stored in:
 - **My application is stored in Amazon S3** — For information, see [Specify Information About a Revision Stored in an Amazon S3 Bucket \(p. 250\)](#), and then return to step 6.
 - **My application is stored in GitHub** — For information, see [Specify Information About a Revision Stored in a GitHub Repository \(p. 250\)](#) below, and then return to step 6.
6. (Optional) In the **Deployment description** box, type a description for this deployment.
7. In the **Deployment configuration** list, choose a deployment configuration to control the rate at which instances are updated with the new application revisions (in-place deployments) or the rate at which traffic is rerouted to a replacement environment (blue/green deployments).

For more information, see [Working with Deployment Configurations in AWS CodeDeploy \(p. 203\)](#).

8. In **Content options**, you can specify how AWS CodeDeploy handles files in a deployment target location that weren't part of the previous successful deployment.

Choose from the following:

- **Fail the deployment** — An error is reported and the deployment status is changed to Failed.
- **Overwrite the content** — If a file of the same name exists in the target location, the version from the application revision replaces it.
- **Retain the content** — If a file of the same name exists in the target location, it is kept and the version in the application revision is not copied to the instance.

For more information, see [Rollback Behavior with Existing Content \(p. 260\)](#).

9. (Optional) In **Rollback configuration overrides**, you can specify different automatic rollback options for this deployment than were specified for the deployment group, if any.

Note

For information about rollbacks in AWS CodeDeploy, see [Redeployments and Deployment Rollbacks \(p. 17\)](#) and [Redeploy and Roll Back a Deployment with AWS CodeDeploy \(p. 259\)](#).

Choose from the following:

- **Roll back when a deployment fails** — AWS CodeDeploy will redeploy the last known good revision as a new deployment.
 - **Roll back when alarm thresholds are met** — If alarms were added to the deployment group, AWS CodeDeploy will redeploy the last known good revision when one or more of the specified alarms is activated.
 - **Disable rollbacks** — Do not perform rollbacks for this deployment.
10. Choose **Deploy**.

To track the status of your deployment, see [View Deployment Details with AWS CodeDeploy \(p. 254\)](#).

Topics

- [Specify Information About a Revision Stored in an Amazon S3 Bucket \(p. 250\)](#)

- [Specify Information About a Revision Stored in a GitHub Repository \(p. 250\)](#)

Specify Information About a Revision Stored in an Amazon S3 Bucket

If you are following the steps in [Create an EC2/On-Premises Compute Platform Deployment \(Console\) \(p. 248\)](#), follow these steps to add details about an application revision 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 AWS Management Console and open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.

Browse to and choose your revision.
 2. If the **Properties** pane is not visible, choose 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.

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.
- If the **Link** field value does not specify a version ID, add `?etag=` and the ETag to the end of the **Link** field value.

Note

Although it's not as easy as copying the value of the **Link** field, you can also type the revision location in one of the following formats:

```
s3://bucket-name/folders/objectName
s3://bucket-name/folders/objectName?versionId=versionId
s3://bucket-name/folders/objectName?etag=etag
s3://bucket-name/folders/objectName?versionId=versionId&etag=etag
bucket-name.s3.amazonaws.com/folders/objectName
```

2. If a message appears in the **File type** list that says the file type could not be detected, choose the revision's file type. Otherwise, accept the detected file type.

Specify Information About a Revision Stored in a GitHub Repository

If you are follow the steps in [Create an EC2/On-Premises Compute Platform Deployment \(Console\) \(p. 248\)](#), follow these steps to add details about an application revision stored in a GitHub repository.

1. In **Connect to GitHub**, do one of the following:
 - To create a connection for AWS CodeDeploy applications to a GitHub account, sign out of GitHub in a separate web browser tab. In **GitHub account**, type a name to identify this connection, and then choose **Connect to GitHub**. The web page prompts you to authorize AWS CodeDeploy to interact with GitHub for your application. Continue to step 2.

- To use a connection you have already created, in **GitHub account**, select its name, and then choose **Connect to GitHub**. Continue to step 4.
 - To create a connection to a different GitHub account, sign out of GitHub in a separate web browser tab. Choose **Connect to a different GitHub account**, and then choose **Connect to GitHub**. Continue to step 2.
2. If you are prompted to sign in to GitHub, follow the instructions on the **Sign in** page. Sign in with your GitHub user name or email and password.
 3. If an **Authorize application** page appears, choose **Authorize application**.
 4. On the **Create deployment** page, in the **Repository name** box, type the GitHub user or organization name that contains the revision, followed by a forward slash (/), 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 **Your repositories**, hover your mouse pointer over the target repository name. A tooltip appears, displaying the GitHub user or organization name, followed by a forward slash (/), followed by the name of the repository. Type this displayed value into the **Repository name** box.
- Note**
If the target repository name is not visible in **Your repositories**, use the **Search GitHub** box to find the target repository name and 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 **Your repositories**, choose the repository name that contains the target commit.
 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.

Create an AWS Lambda Compute Platform Deployment (CLI)

After you have created the application and revision (in AWS Lambda deployments, this is the AppSpec file):

Call the [create-deployment](#) command, specifying:

- An application name. To view a list of application names, call the [list-applications](#) command.
- A deployment group name. To view a list of 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 that contains the revision.
- The name of the uploaded revision.
- (Optional) The Amazon S3 version identifier for the revision. (If the version identifier is not specified, AWS CodeDeploy uses the most recent version.)
- (Optional) The ETag for the revision. (If the ETag is not specified, AWS CodeDeploy skips object validation.)

For revisions stored in a file that is not in Amazon S3, you need the file name and its path. Your revision file is written using JSON or YAML, so it most likely has a .json or .yaml extension.

- (Optional) The name of a deployment configuration to use. To view a list of deployment configurations, call the [list-deployment-configs](#) command. (If not specified, AWS CodeDeploy uses a specific default deployment configuration.)
- (Optional) A description for the deployment.

The revision file can be specified as a file uploaded to an Amazon S3 bucket or as a string. The syntax for each when used as part of the **create-deployment** command is:

- Amazon S3 bucket:

The `version` and `eTag` are optional.

```
--s3-location bucket=string,key=string,bundleType=JSON|YAML,version=string,eTag=string
```

- String:

```
--revision '{"revisionType": "String", "string": {"content": "revision-as-string"}}'
```

Note

The **create-deployment** command can load a revision from a file. For more information, see [Loading Parameters from a File](#).

For AWS Lambda deployment revision templates, see [Add an AppSpec File for an AWS Lambda Deployment \(p. 233\)](#). For an example revision, see [AppSpec File Example for an AWS Lambda Deployment \(p. 326\)](#).

To track the status of your deployment, see [View Deployment Details with AWS CodeDeploy \(p. 254\)](#).

Create an EC2/On-Premises Compute Platform Deployment (CLI)

To use the AWS CLI to deploy a revision to the EC2/On-Premises compute platform:

1. After you have prepared the instances, created the application, and pushed the revision, do one of the following:
 - If you want to deploy a revision from an Amazon S3 bucket, continue to step 2 now.
 - If you want to deploy a revision from a GitHub repository, first complete the steps in [Connect an AWS CodeDeploy Application to a GitHub Repository \(p. 253\)](#), and then continue to step 2.
2. Call the [create-deployment](#) command, specifying:
 - An application name. To view a list of application names, call the [list-applications](#) command.
 - An Amazon EC2 deployment group name. To view a list of 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 that contains 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.

- (Optional) The Amazon S3 version identifier for the revision. (If the version identifier is not specified, AWS CodeDeploy uses the most recent version.)
- (Optional) The ETag for the revision. (If the ETag is not specified, AWS CodeDeploy skips 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 (/), followed by the repository name.
- The commit ID for the revision.
- (Optional) The name of a deployment configuration to use. To view a list of deployment configurations, call the [list-deployment-configs](#) command. (If not specified, AWS CodeDeploy uses a specific default deployment configuration.)
- (Optional) Whether you want the deployment to an instance to continue to the **BeforeInstall** deployment lifecycle event if the **ApplicationStop** deployment lifecycle event fails.
- (Optional) A description for the deployment.
- For blue/green deployments, information about the instances that belong to the replacement environment in a blue/green deployment, including the names of one or more Auto Scaling groups, or the tag filter key, type, and value used to identify Amazon EC2 instances.

Note

Use this syntax as part of the **create-deployment** call to specify information about a revision in Amazon S3 directly on the command line. (The `version` and `eTag` are optional.)

```
--s3-location bucket=string,key=string,bundleType=tar|tgz|  
zip,version=string,eTag=string
```

Use this syntax as part of the **create-deployment** call to specify information about a revision in GitHub directly on the command line:

```
--github-location repository=string,commitId=string
```

To get information about revisions that have been pushed already, call the [list-application-revisions](#) command.

To track the status of your deployment, see [View Deployment Details with AWS CodeDeploy \(p. 254\)](#).

Topics

- [Connect an AWS CodeDeploy Application to a GitHub Repository \(p. 253\)](#)

Connect an AWS CodeDeploy Application to a GitHub Repository

Before you can deploy an application from a GitHub repository for the first time using the AWS CLI, you must first give AWS CodeDeploy permission to interact with GitHub on behalf of your GitHub account. This step must be completed once for each application 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

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. On the AWS CodeDeploy menu, choose **Deployments**.
3. Choose **Create deployment**.

Note

You will not be creating a new deployment. This is currently the only way to give AWS CodeDeploy permission to interact with GitHub on behalf of your GitHub user account.

4. In the **Application** drop-down list, choose the application you want to link to your GitHub user account.
5. In the **Deployment group** drop-down list, choose any available deployment group.
6. Next to **Repository type**, choose **My application revision is stored in GitHub**.
7. Choose **Connect to GitHub**.

Note

If you see a **Connect to a different GitHub account** link:

You might have already authorized AWS CodeDeploy to interact with GitHub on behalf of a different GitHub account for the application.

You might have revoked authorization for AWS CodeDeploy to interact with GitHub on behalf of the signed-in GitHub account for all applications linked to in AWS CodeDeploy.

For more information, see [GitHub Authentication with Applications in AWS CodeDeploy](#) (p. 54).

8. If you are not already signed in to GitHub, follow the instructions on the **Sign in** page.
9. On the **Authorize application** page, choose **Authorize application**.
10. Now that AWS CodeDeploy has permission, choose **Cancel**, and continue with the steps in [Create an EC2/On-Premises Compute Platform Deployment \(CLI\)](#) (p. 252).

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 deployments associated with your AWS account.

Note

You can view deployment logs on your instances in the following locations:

- Amazon Linux, RHEL, and Ubuntu Server: `/opt/codedeploy-agent/deployment-root/deployment-logs/codedeploy-agent-deployments.log`
- Windows Server: `C:\ProgramData\Amazon\CodeDeploy<DEPLOYMENT-GROUP-ID>\logs\scripts.log`

For more information, see [Analyzing log files to investigate deployment failures on instances](#) (p. 355).

Topics

- [View Deployment Details \(Console\)](#) (p. 254)
- [View Deployment Details \(CLI\)](#) (p. 255)

View Deployment Details (Console)

To use the AWS CodeDeploy console to view deployment details:

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. On the AWS CodeDeploy menu, choose **Deployments** to view a list of deployments and their details.

Note

If no entries are displayed, make sure the correct region is selected. On the navigation bar, in the region selector, choose one of the regions listed in [Region and Endpoints](#) in the *AWS General Reference*. AWS CodeDeploy is supported in these regions only.

3. To see more details for a single deployment, in **Deployments**, choose the deployment ID.

View Deployment Details (CLI)

To use the AWS CLI to view deployment details, call the `get-deployment` command or the `batch-get-deployments` command. 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 deployment IDs, call the `list-deployments` command, specifying:

- The name of the application associated with the deployment. To view a list of application names, call the `list-applications` command.
- The name of the deployment group associated with the deployment. To view a list of deployment group names, call the `list-deployment-groups` command.
- Optionally, whether to include details about 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 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 Log Data for AWS CodeDeploy Deployments

You can view the log data created by an AWS CodeDeploy deployment by setting up the Amazon CloudWatch Logs agent to view aggregated data in the CloudWatch console or by logging into an individual instance to review the log file.

Topics

- [View Log File Data in the Amazon CloudWatch Console](#) (p. 256)
- [View Log Files on an Instance](#) (p. 256)

View Log File Data in the Amazon CloudWatch Console

When the Amazon CloudWatch Logs agent is installed on an instance, deployment log data for all deployments to that instance becomes available for viewing in the CloudWatch console. For simplicity, we recommend using Amazon CloudWatch Logs to centrally monitor log files instead of viewing them instance by instance. For information about setting up the Amazon CloudWatch Logs agent, see [View AWS CodeDeploy Logs in CloudWatch Logs Console](#).

View Log Files on an Instance

To view deployment log data for an individual instance, you can sign in to the instance and browse for information about errors or other deployment events.

Topics

- [To view deployment log files on Amazon Linux, RHEL, and Ubuntu Server instances \(p. 256\)](#)
- [To view deployment logs files on Windows Server instances \(p. 257\)](#)

To view deployment log files on Amazon Linux, RHEL, and Ubuntu Server instances

On Amazon Linux, RHEL, and Ubuntu Server instances, deployment logs are stored in the following location:

```
/opt/codedeploy-agent/deployment-root/deployment-logs/codedeploy-agent-deployments.log
```

To view or analyze deployment logs on Amazon Linux, RHEL, and Ubuntu Server instances, sign in to the instance, and then 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:

Command	Result
& ERROR	Show just the error messages in the log file. Use a single space before and after the word ERROR .
/ ERROR	Search for the next error message. ¹
? ERROR	Search for the previous error message. ² Use a single space before and after the word ERROR .
G	Go to the end of the log file.
g	Go to the start of the log file.
q	Exit the log file.
h	Learn about additional commands.
¹ After you type / ERROR , type n for the next error message. Type N for the previous error message.	

Command	Result
² After you type ? ERROR , type n for the next error message, or type N for the previous error message.	

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:

Command	Result
&stderr	Show just the error messages in the log file.
/stderr	Search for the next error message. ¹
?stderr	Search for the previous error message. ²
G	Go to the end of the log file.
g	Go to the start of the log file.
q	Exit the log file.
h	Learn about additional commands.
¹ After you type /stderr , type n for the next error message forward. Type N for the previous error message backward.	
² After you type ?stderr , type n for the next error message backward. Type N for the previous error message forward.	

To view deployment logs files on Windows Server instances

AWS CodeDeploy agent log file: On Windows Server instances, the AWS CodeDeploy agent log file is stored at the following location:

```
C:\ProgramData\Amazon\CodeDeploy\log\codedeploy-agent-log.txt
```

To view or analyze the AWS CodeDeploy agent log file on a Windows Server instance, sign in to the instance, and then type the following command to open the 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.

AWS CodeDeploy scripts log files: On Windows Server instances, deployment logs are stored at the following location:

```
C:\ProgramData\Amazon\CodeDeploy\deployment-group-id\ideployment-id\logs\scripts.log
```

Where:

- `deployment-group-id` is a string such as `examplebf3a9c7a-7c19-4657-8684-b0c68d0cd3c4`
- `deployment-id` is an identifier such as `d-12EXAMPLE`

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.

Stop a Deployment with AWS CodeDeploy

You can use the AWS CodeDeploy console, the AWS CLI, or the AWS CodeDeploy APIs to stop deployments associated with your AWS account.

Warning

Stopping an EC2/On-Premises deployment can 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. 16).

Topics

- [Stop a deployment \(console\)](#) (p. 258)
- [Stop a deployment \(CLI\)](#) (p. 258)

Stop a deployment (console)

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. On the AWS CodeDeploy menu, choose **Deployments**.

Note

If no entries are displayed, make sure the correct region is selected. On the navigation bar, in the region selector, choose one of the regions listed in [Region and Endpoints](#) in the *AWS General Reference*. AWS CodeDeploy is supported in these regions only.

3. In the **Actions** column for the deployment you want to stop, choose **Stop**.

Note

If a **Stop** button does not appear in the **Actions** column, the deployment has progressed to a point where it cannot be stopped.

Stop a deployment (CLI)

Call the `stop-deployment` command, specifying the deployment ID. To view a list of deployment IDs, call the `list-deployments` command.

Redeploy and Roll Back a Deployment with AWS CodeDeploy

AWS CodeDeploy rolls back deployments by redeploying a previously deployed revision of an application as a new deployment. These rolled-back deployments are technically new deployments, with new deployment IDs, rather than restored versions of a previous deployment.

Deployments can be rolled back automatically or manually.

Topics

- [Automatic Rollbacks \(p. 259\)](#)
- [Manual Rollbacks \(p. 259\)](#)
- [Rollback and Redeployment Workflow \(p. 259\)](#)
- [Rollback Behavior with Existing Content \(p. 260\)](#)

Automatic Rollbacks

You can configure a deployment group or deployment to automatically roll back when a deployment fails or when a monitoring threshold you specify is met. In this case, the last known good version of an application revision is deployed. You configure automatic rollbacks when you create an application or create or update a deployment group.

When you create a new deployment, you can also choose to override the automatic rollback configuration that were specified for the deployment group.

Note

You can use Amazon Simple Notification Service to receive a notification whenever a deployment is rolled back automatically. For information, see [Monitoring Deployments with Amazon SNS Event Notifications \(p. 278\)](#).

For more information about configuring automatic rollbacks, see [Configure Advanced Options for a Deployment Group \(p. 228\)](#).

Manual Rollbacks

If you have not set up automatic rollbacks, you can manually roll back a deployment by creating a new deployment that uses any previously deployed application revision and following the steps to redeploy a revision. You might do this if an application has gotten into an unknown state. Rather than spending a lot of time troubleshooting, you can redeploy the application to a known working state. For more information, see [Create a Deployment with AWS CodeDeploy \(p. 245\)](#).

Note

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

Rollback and Redeployment Workflow

When automatic rollback is initiated, or when you manually initiate a redeployment or manual 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 cleanup file:

```
/opt/codedeploy-agent/deployment-root/deployment-instructions/deployment-group-ID-cleanup file (for Amazon Linux, Ubuntu Server, and RHEL instances)
```

C:\ProgramData\Amazon\CodeDeploy\deployment-instructions**deployment-group-ID**-cleanup file (for Windows Server instances)

If it exists, AWS CodeDeploy uses the cleanup file to remove from the instance all listed files before starting the new deployment.

For example, the first two text files and two script files were already deployed to an Amazon EC2 instance running Windows Server, and the scripts created two more text files during 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)
```

The cleanup file will list only the first two text files and two script files:

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

Before the new deployment, AWS CodeDeploy will remove only the first two text files and the two script files, leaving the last two text files untouched:

```
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
```

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, whether manual or automatic rollbacks. For example, if the `c.bat` and `d.bat` files contain logic to not re-create the `e.txt` and `f.txt` files if they 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. You can add logic to `c.bat` and `d.bat` to always check for and delete old versions of `e.txt` and `f.txt` before creating new ones.

Rollback Behavior with Existing Content

As part of the deployment process, the AWS CodeDeploy agent removes from each instance all the files installed by the most recent deployment. If files that weren't part of a previous deployment appear in target deployment locations, you can choose what AWS CodeDeploy does with them during the next deployment:

- **Fail the deployment** — An error is reported and the deployment status is changed to Failed.
- **Overwrite the content** — The version of the file from the application revision replaces the version already on the instance.
- **Retain the content** — The file in the target location is kept and the version in the application revision is not copied to the instance.

You might choose to retain files that you want to be part of the next deployment without having to add them to the application revision package. For example, you might upload files directly to the instance

that are required for the deployment but weren't added to the application revision bundle. Or you might upload files to the instance if your applications are already in your production environment but you want to use AWS CodeDeploy for the first time to deploy them.

In the case of rollbacks, where the most recent successfully deployed application revision is redeployed due to a deployment failure, the content-handling option for that last successful deployment is applied to the rollback deployment.

However, if the deployment that failed was configured to overwrite, instead of retain files, an unexpected result can occur during the rollback. Specifically, the files you expected to be retained might be removed by the failed deployment. The files are not on the instance when the rollback deployment runs.

In the following example, there are three deployments. Any file that is overwritten (deleted) during the failed second deployment is no longer available (cannot be retained) when application revision 1 is deployed again during deployment 3:

Deployment	Application revision	Content overwrite option	Deployment status	Behavior and result
deployment 1	application revision 1	RETAIN	Succeeded	AWS CodeDeploy detects files in the target locations that were not deployed by the previous deployment. These files might be placed there intentionally to become part of the current deployment. They are kept and recorded as part of the current deployment package.
deployment 2	application revision 2	OVERWRITE	Failed	During the deployment process, AWS CodeDeploy deletes all the files that are part of the previous successful deployment. This includes the files that were retained during deployment 1. However, the deployment fails for unrelated reasons.

Deployment	Application revision	Content overwrite option	Deployment status	Behavior and result
deployment 3	application revision 1	RETAIN		<p>Because auto rollback is enabled for the deployment or deployment group, AWS CodeDeploy deploys the last known good application revision, application revision 1.</p> <p>However, the files that you wanted to retain in deployment 1 were deleted before deployment 2 failed and cannot be retrieved by AWS CodeDeploy. You can add them to the instance yourself if they are required for application revision 1, or you can create a new application revision.</p>

Deploy an Application in a Different AWS Account

Organizations commonly have multiple AWS accounts that they use for different purposes (for example, one for system administration tasks and another for development, test, and production tasks or one associated with development and test environments and another associated with the production environment).

Although you might perform related work in different accounts, AWS CodeDeploy deployment groups and the Amazon EC2 instances to which they deploy are strictly tied to the accounts under which they were created. You cannot, for example, add an instance that you launched in one account to a deployment group in another.

Assume you have two AWS accounts: your development account and your production account. You work primarily in the development account, but you want to be able kick off deployments in your production account without a full set of credentials there or without having to sign out of the development account and in to the production account.

After following the cross-account configuration steps, you can initiate deployments that belong to another of your organization's accounts without needing a full set of credentials for that other account.

You do this, in part, by using a capability provided by the AWS Security Token Service (AWS STS) that grants you temporary access to that account.

Step 1: Create an S3 Bucket in Either Account

In either the development account or the production account:

- If you have not already done so, create an Amazon S3 bucket where the application revisions for the production account will be stored. For information, see [Create a Bucket in Amazon S3](#). You can even use the same bucket and application revisions for both accounts, deploying the same files to your production environment that you tested and verified in your development account.

Step 2: Grant Amazon S3 Bucket Permissions to the Production Account's IAM Instance Profile

If the Amazon S3 bucket you created in step 1 is in your production account, this step is not required. The role you assume later will already have access to this bucket because it is also in the production account.

If you created the Amazon S3 bucket in the development account, do the following:

- In the production account, create an IAM instance profile. For information, see [Step 4: Create an IAM Instance Profile for Your Amazon EC2 Instances \(p. 25\)](#).

Note

Make note of the ARN for this IAM instance profile. You will need to add it to the cross-bucket policy you create next.

- In the development account, give access to the Amazon S3 bucket you created in the development account to the IAM instance profile you just created in your production account. For information, see [Example 2: Bucket Owner Granting Cross-Account Bucket Permissions](#).

Note the following as you complete the process of granting cross-account bucket permissions:

- In the sample walkthrough, Account A represents your development account and Account B represents your production account.
- When you [perform the Account A \(development account\) tasks](#), modify the following bucket policy to grant cross-account permissions instead of using the sample policy provided in the walkthrough.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Cross-account permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::account-id:role/role-name"
      },
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3::bucket-name/*"
      ]
    }
  ]
}
```

`account-id` represents the account number of the production account where you just created the IAM instance profile.

`role-name` represents the name of the IAM instance profile you just created.

`bucket-name` represents the name of the bucket you created in step 1. Be sure to include the `/*` after the name of your bucket to provide access to each of the files inside the bucket.

Step 3: Create Resources and a Cross-Account Role in the Production Account

In your production account:

- Create your AWS CodeDeploy resources — application, deployment group, deployment configuration, Amazon EC2 instances, Amazon EC2 instance profile, service role, and so on — using the instructions in this guide.
- Create an additional role, a cross-account IAM role, that a user in your development account can assume to perform AWS CodeDeploy operations in this production account.

Use the [Walkthrough: Delegate Access Across AWS Accounts Using IAM Roles](#) as a guide to help you create the cross-account role. Instead of adding the sample permissions in the walkthrough to your policy document, you should attach, at minimum, the following two AWS-supplied policies to the role:

- `AmazonS3FullAccess`: Required only if the S3 bucket is in the development account. Provides the assumed production account role with full access to the Amazon S3 services and resources in the development account, where the revision is stored.
- `AWSCodeDeployDeployerAccess`: Enables an IAM user to register and deploy revisions.

If you want to create and manage deployment groups and not just initiate deployments, add the `AWSCodeDeployFullAccess` policy instead of the `AWSCodeDeployDeployerAccess` policy. For more information about using IAM managed policies to grant permissions for AWS CodeDeploy tasks, see [AWS Managed \(Predefined\) Policies for AWS CodeDeploy \(p. 296\)](#).

You can attach additional policies if you want to perform tasks in other AWS services while using this cross-account role.

Important

As you create the cross-account IAM role, make a note of the details you will need to gain access to the production account.

To use the AWS Management Console to switch roles, you will need to supply either of the following:

- A URL for accessing the production account with the assumed role's credentials. You will find the URL on the **Review** page, which is displayed at the end of the cross-account role creation process.
- The name of the cross-account role and either the account ID number or alias.

To use the AWS CLI to switch roles, you will need to supply the following:

- The ARN of the cross-account role you will assume.

Step 4: Upload the Application Revision to Amazon S3 Bucket

In the account in which you created the Amazon S3 bucket:

- Upload your application revision to the Amazon S3 bucket. For information, see [Push a Revision for AWS CodeDeploy to Amazon S3](#) (p. 238).

Step 5: Assume the Cross-Account Role and Deploy Applications

In the development account, you can use the AWS CLI or the AWS Management Console to assume the cross-account role and initiate the deployment in the production account.

For instructions about how to use the AWS Management Console to switch roles and initiate deployments, see [Switching to a Role \(AWS Management Console\)](#) and [Create an EC2/On-Premises Compute Platform Deployment \(Console\)](#) (p. 248).

For instructions about how to use the AWS CLI to assume the cross-account role and initiate deployments, see [Switching to an IAM Role \(AWS Command Line Interface\)](#) and [Create an EC2/On-Premises Compute Platform Deployment \(CLI\)](#) (p. 252).

For more information about assuming a role through AWS STS, see [AssumeRole](#) in the [AWS Security Token Service User Guide](#) and [assume-role](#) in the [AWS CLI Command Reference](#).

Related topic:

- [AWS CodeDeploy: Deploying from a Development Account to a Production Account](#)

Use the AWS CodeDeploy Agent to Validate a Deployment Package on a Local Machine

Using the AWS CodeDeploy agent, you can deploy content on an instance you are logged in to. This allows you to test the integrity of an application specification file (AppSpec file) that you intend to use in a deployment and the content you intend to deploy.

You do not need to create an application and deployment group. If you want to deploy content stored on the local instance, you do not even need an AWS account. For the simplest testing, you can run the **codedeploy-local** command, without specifying any options, in a directory that contains the AppSpec file and the content to be deployed. There are options for other test cases in the tool.

By validating a deployment package on a local machine you can:

- Test the integrity of an application revision.
- Test the contents of an AppSpec file.
- Try out AWS CodeDeploy for the first time with your existing application code.
- Deploy content rapidly when you are already logged in to an instance.

You can use deploy content that is stored on the local instance or in a supported remote repository type (Amazon S3 buckets or public GitHub repositories).

Prerequisites

Before you start a local deployment, complete the following steps:

- Create or use an instance type supported by the AWS CodeDeploy agent. For information, see [Operating Systems Supported by the AWS CodeDeploy Agent \(p. 125\)](#).
- Install version 1.0.1.1352 or later of the AWS CodeDeploy agent. For information, see [Install or Reinstall the AWS CodeDeploy Agent \(p. 134\)](#).
- If you are deploying your content from an Amazon S3 bucket or GitHub repository, provision an IAM user to use with AWS CodeDeploy. For information, see [Step 1: Provision an IAM User \(p. 19\)](#).
- If you are deploying your application revision from an Amazon S3 bucket, create an Amazon S3 bucket in the region you are working in and apply an Amazon S3 bucket policy to the bucket. This policy grants your instances the permissions required to download the application revision.

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 in 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"
        ]
      }
    }
  ]
}
```

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 in 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"
        ]
      }
    }
  ]
}
```

For information about how to generate and attach an Amazon S3 bucket policy, see [Bucket Policy Examples](#).

- If you are deploying your application revision from an Amazon S3 bucket or GitHub repository, set up an IAM instance profile and attach it to the instance. For information, see [Step 4: Create an IAM Instance Profile for Your Amazon EC2 Instances](#) (p. 25), [Create an Amazon EC2 Instance for AWS CodeDeploy \(AWS CLI or Amazon EC2 Console\)](#) (p. 156), and [Create an Amazon EC2 Instance for AWS CodeDeploy \(AWS CloudFormation Template\)](#) (p. 162).
- If you are deploying your content from GitHub, create a GitHub account and a public repository. To create a GitHub account, see [Join GitHub](#). To create a GitHub repository, see [Create a Repo](#).

Note

Private repositories are not currently supported. If your content is stored in a private GitHub repository, you can download it to the instance and use the `--bundle-location` option to specify its local path.

- Prepare the content (including an AppSpec file) that you want to deploy to the instance and place it on the local instance, in your Amazon S3 bucket, or in your GitHub repository. For information, see [Working with Application Revisions for AWS CodeDeploy](#) (p. 232).
- If you want to use values other than the defaults for other configuration options, create the configuration file and place it on the instance (`/etc/codedeploy-agent/conf/codedeployagent.yml` for Amazon Linux, RHEL, or Ubuntu Server instances or `C:\ProgramData\Amazon\CodeDeploy\conf.yml` for Windows Server instances). For information, see [AWS CodeDeploy Agent Configuration Reference](#) (p. 330).

Note

If you use a configuration file on Amazon Linux, RHEL, or Ubuntu Server instances, you must either:

- Use the `:root_dir:` and `:log_dir:` variables to specify locations other than the defaults for the deployment root and log directory folders.
- Use `sudo` to run AWS CodeDeploy agent commands.

Create a Local Deployment

On the instance where you want to create the local deployment, open a terminal session (Amazon Linux, RHEL, or Ubuntu Server instances) or a command prompt (Windows Server) to run the tool commands.

Note

The `codedeploy-local` command is installed in the following locations:

- On Amazon Linux, RHEL, or Ubuntu Server: `/opt/codedeploy-agent/bin`.
- On Windows Server: `C:\ProgramData\Amazon\CodeDeploy\bin`.

Basic Command Syntax

```
codedeploy-local [options]
```

Synopsis

```
codedeploy-local
[--bundle-location <value>]
[--type <value>]
[--file-exists-behavior <value>]
[--deployment-group <value>]
[--events <comma-separated values>]
[--agent-configuration-file <value>]
```

Options

-l, --bundle-location

The location of the application revision bundle. If you do not specify a location, the tool uses the directory you are currently working in by default. If you specify a value for `--bundle-location`, you must also specify a value for `--type`.

Bundle location format examples:

- Local Amazon Linux, RHEL, or Ubuntu Server instance: `/path/to/local/bundle.tgz`
- Local Windows Server instance: `C:/path/to/local/bundle`
- Amazon S3 bucket: `s3://mybucket/bundle.tar`
- GitHub repository: `https://github.com/account-name/repository-name/`

-t, --type

The format of the application revision bundle. Supported types include `tgz`, `tar`, `zip`, and `directory`. If you do not specify a type, the tool uses `directory` by default. If you specify a value for `--type`, you must also specify a value for `--bundle-location`.

-b, --file-exists-behavior

Indicates how files are handled that already exist in a deployment target location but weren't part of a previous successful deployment. Options include `DISALLOW`, `OVERWRITE`, `RETAIN`. For more information, see [fileExistsBehavior](#) in *AWS CodeDeploy API Reference*.

-g, --deployment-group

The path to the folder that is the target location for the content to be deployed. If you do not specify a folder, the tool creates one named *default-local-deployment-group* inside your deployment root directory. For each local deployment you create, the tool creates a subdirectory inside this folder with names such as *d-98761234-local*.

-e, --events

A set of override lifecycle event hooks you want to run, in order, instead of the events you listed in the AppSpec file. Multiple hooks can be specified, separated by commas. You can use this option if:

- You want to run a different set of events without having to update the AppSpec file.
- You want to run a single event hook as an exception to what's in the AppSpec file, such as `ApplicationStop`.

If you don't specify **DownloadBundle** and **Install** events in the override list, they will run before all the event hooks you do specify. If you include **DownloadBundle** and **Install** in the list of `--events` options, they must be preceded only by events that normally run before them in AWS CodeDeploy deployments. For information, see [AppSpec 'hooks' Section \(p. 316\)](#).

-c, --agent-configuration-file

The location of a configuration file to use for the deployment, if you store it in a location other than the default. A configuration file specifies alternatives to other default values and behaviors for a deployment.

By default, configuration files are stored in `/etc/codedeploy-agent/conf/codedeployagent.yml` (Amazon Linux, RHEL, or Ubuntu Server instances) or `C:/ProgramData/Amazon/CodeDeploy/conf.yml` (Windows Server). For more information, see [AWS CodeDeploy Agent Configuration Reference \(p. 330\)](#).

-h, --help

Displays a summary of help content.

-v, --version

Displays the tool's version number.

Examples

The following are examples of valid command formats.

```
codedeploy-local
```

```
codedeploy-local --bundle-location /path/to/local/bundle/directory
```

```
codedeploy-local --bundle-location C:/path/to/local/bundle.zip --type zip --deployment-group my-deployment-group
```

```
codedeploy-local --bundle-location /path/to/local/directory --type directory --deployment-group my-deployment-group
```

Deploy a bundle from Amazon S3:

```
codedeploy-local --bundle-location s3://mybucket/bundle.tgz --type tgz
```

```
codedeploy-local --bundle-location s3://mybucket/bundle.zip?versionId=1234&etag=47e8 --type zip --deployment-group my-deployment-group
```

Deploy a bundle from a public GitHub repository:

```
codedeploy-local --bundle-location https://github.com/aws-labs/aws-codedeploy-sample-tomcat --type zip
```

```
codedeploy-local --bundle-location https://api.github.com/repos/aws-labs/aws-codedeploy-sample-tomcat/zipball/master --type zip
```

```
codedeploy-local --bundle-location https://api.github.com/repos/aws-labs/aws-codedeploy-sample-tomcat/zipball/HEAD --type zip
```

```
codedeploy-local --bundle-location https://api.github.com/repos/aws-labs/aws-codedeploy-sample-tomcat/zipball/1a2b3c4d --type zip
```

Deploy a bundle specifying multiple lifecycle events:

```
codedeploy-local --bundle-location /path/to/local/bundle.tar --type tar --application-folder my-deployment --events DownloadBundle,Install,ApplicationStart,HealthCheck
```

Stop a previously deployed application using the ApplicationStop lifecycle event:

```
codedeploy-local --bundle-location /path/to/local/bundle.tgz --type tgz --deployment-group  
--events ApplicationStop
```

Deploy using a specific deployment group ID:

```
codedeploy-local --bundle-location C:/path/to/local/bundle/directory --deployment-group  
1234abcd-5dd1-4774-89c6-30b107ac5dca
```

```
codedeploy-local --bundle-location C:/path/to/local/bundle.zip --type zip --deployment-  
group 1234abcd-5dd1-4774-89c6-30b107ac5dca
```

Monitoring Deployments in AWS CodeDeploy

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS CodeDeploy and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. Before you start monitoring AWS CodeDeploy, however, you should create a monitoring plan that includes answers to the following questions:

- What are your monitoring goals?
- What resources will you monitor?
- How often will you monitor these resources?
- What monitoring tools will you use?
- Who will perform the monitoring tasks?
- Who should be notified when something goes wrong?

The next step is to establish a baseline for normal AWS CodeDeploy performance in your environment, by measuring performance at various times and under different load conditions. As you monitor AWS CodeDeploy, store historical monitoring data so that you can compare it with current performance data, identify normal performance patterns and performance anomalies, and devise methods to address issues.

For example, if you're using AWS CodeDeploy, you can monitor the status of deployments and target instances. When deployments or instances fail, you might need to reconfigure an application specification file, reinstall or update the AWS CodeDeploy agent, update settings in an application or deployment group, or make changes to instance settings or an AppSpec file.

To establish a baseline, you should, at a minimum, monitor the following items:

- Deployment events and status
- Instance events and status

Automated Monitoring Tools

AWS provides various tools that you can use to monitor AWS CodeDeploy. You can configure some of these tools to do the monitoring for you, while some of the tools require manual intervention. We recommend that you automate monitoring tasks as much as possible.

You can use the following automated monitoring tools to watch AWS CodeDeploy and report when something is wrong:

- **Amazon CloudWatch Alarms** – Watch a single metric over a time period that you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon Simple Notification Service (Amazon SNS) topic or Amazon EC2 Auto Scaling policy. CloudWatch alarms do not invoke actions simply because

they are in a particular state; the state must have changed and been maintained for a specified number of periods. For more information, see [Monitoring Deployments with Amazon CloudWatch Tools](#) (p. 273).

For information about updating your service role to work with CloudWatch alarm monitoring, see [Grant CloudWatch Permissions to an AWS CodeDeploy Service Role](#) (p. 274). For information about adding CloudWatch alarm monitoring to your AWS CodeDeploy operations, see [Create an Application with AWS CodeDeploy](#) (p. 209), [Create a Deployment Group with AWS CodeDeploy](#) (p. 220), or [Change Deployment Group Settings with AWS CodeDeploy](#) (p. 227).

- **Amazon CloudWatch Logs** – Monitor, store, and access your log files from AWS CloudTrail or other sources. For more information, see [Monitoring Log Files](#) in the *Amazon CloudWatch User Guide*.

For information about using the CloudWatch console to view AWS CodeDeploy logs, see [View AWS CodeDeploy Logs in CloudWatch Logs Console](#).

- **Amazon CloudWatch Events** – Match events and route them to one or more target functions or streams to make changes, capture state information, and take corrective action. For more information, see [What is Amazon CloudWatch Events](#) in the *Amazon CloudWatch User Guide*.

For information about using CloudWatch Events in your AWS CodeDeploy operations, see [Monitoring Deployments with Amazon CloudWatch Events](#) (p. 274).

- **AWS CloudTrail Log Monitoring** – Share log files between accounts, monitor CloudTrail log files in real time by sending them to CloudWatch Logs, write log processing applications in Java, and validate that your log files have not changed after delivery by CloudTrail. For more information, see [Working with CloudTrail Log Files](#) in the *AWS CloudTrail User Guide*.

For information about using CloudTrail with AWS CodeDeploy, see [Monitoring Deployments with AWS CloudTrail](#) (p. 277).

- **Amazon Simple Notification Service** — Configure event-driven triggers to receive SMS or email notifications about deployment and instance events, such as success or failure. For more information, see [Create a Topic](#) and [What Is Amazon Simple Notification Service](#).

For information about setting up Amazon SNS notifications for AWS CodeDeploy, see [Monitoring Deployments with Amazon SNS Event Notifications](#) (p. 278).

Manual Monitoring Tools

Another important part of monitoring AWS CodeDeploy involves manually monitoring those items that the CloudWatch alarms don't cover. The AWS CodeDeploy, CloudWatch, and other AWS console dashboards provide an at-a-glance view of the state of your AWS environment. We recommend that you also check the log files on AWS CodeDeploy deployments.

- AWS CodeDeploy console shows:
 - The status of deployments
 - The date and time of each last attempted and last successful deployment of a revision
 - The number of instances that succeeded, failed, were skipped, or are in progress in a deployment
 - The status of on-premises instances
 - The date and time when on-premises instances were registered or deregistered
- CloudWatch home page shows:
 - Current alarms and status
 - Graphs of alarms and resources
 - Service health status

In addition, you can use CloudWatch to do the following:

- Create [customized dashboards](#) to monitor the services you care about
- Graph metric data to troubleshoot issues and discover trends
- Search and browse all your AWS resource metrics
- Create and edit alarms to be notified of problems

Topics

- [Monitoring Deployments with Amazon CloudWatch Tools \(p. 273\)](#)
- [Monitoring Deployments with AWS CloudTrail \(p. 277\)](#)
- [Monitoring Deployments with Amazon SNS Event Notifications \(p. 278\)](#)

Monitoring Deployments with Amazon CloudWatch Tools

You can monitor AWS CodeDeploy deployments using the following CloudWatch tools: Amazon CloudWatch Events, CloudWatch alarms, and Amazon CloudWatch Logs.

Reviewing the logs created by the AWS CodeDeploy agent and deployments can help you troubleshoot the causes of deployment failures. As an alternative to reviewing AWS CodeDeploy logs on one instance at a time, you can use CloudWatch Logs to monitor all logs in a central location.

For information about using the CloudWatch console to view AWS CodeDeploy logs, see [View AWS CodeDeploy Logs in CloudWatch Logs Console](#).

For information about using CloudWatch alarms and CloudWatch Events to monitor your AWS CodeDeploy deployments, see the following topics.

Topics

- [Monitoring Deployments with CloudWatch Alarms in AWS CodeDeploy \(p. 273\)](#)
- [Monitoring Deployments with Amazon CloudWatch Events \(p. 274\)](#)

Monitoring Deployments with CloudWatch Alarms in AWS CodeDeploy

You can create a CloudWatch alarm for an instance or Amazon EC2 Auto Scaling group you are using in your AWS CodeDeploy operations. An alarm watches a single metric over a time period you specify and performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods. CloudWatch alarms do not invoke actions simply because they are in a particular state; the state must have changed and been maintained for a specified number of periods.

Using native CloudWatch alarm functionality, you can specify any of the actions supported by CloudWatch when an instance you are using in a deployment fails, such as sending an Amazon SNS notification or stopping, terminating, rebooting, or recovering an instance. For your AWS CodeDeploy operations, you can configure a deployment group to stop a deployment whenever any CloudWatch alarm you associate with the deployment group is activated.

You can associate up to ten CloudWatch alarms with an AWS CodeDeploy deployment group. If any of the specified alarms are activated, the deployment stops, and the status is updated to Stopped. To use this option, you must grant CloudWatch permissions to your AWS CodeDeploy service role.

For information about setting up CloudWatch alarms in the CloudWatch console, see [Creating Amazon CloudWatch Alarms](#) in the *Amazon CloudWatch User Guide*.

For information about associating a CloudWatch alarm with a deployment group in AWS CodeDeploy, see [Create a Deployment Group with AWS CodeDeploy](#) (p. 220) and [Change Deployment Group Settings with AWS CodeDeploy](#) (p. 227).

Topics

- [Grant CloudWatch Permissions to an AWS CodeDeploy Service Role](#) (p. 274)

Grant CloudWatch Permissions to an AWS CodeDeploy Service Role

Before you can use CloudWatch alarm monitoring with your deployments, the service role you use in your AWS CodeDeploy operations must be granted permission to access the CloudWatch resources.

To grant CloudWatch permissions to a service role

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, choose **Roles**.
3. Choose the name of the service role you use in your AWS CodeDeploy operations.
4. On the **Permissions** tab, in the **Inline Policies** area, choose **Create Role Policy**.

–or–

If the **Create Role Policy** button is not available, expand the **Inline Policies** area, and then choose [click here](#).

5. On the **Set Permissions** page, choose **Custom Policy**, and then choose **Select**.
6. On the **Review Policy** page, in the **Policy Name** field, type a name to identify this policy, such as `CWAlarms`.
7. Paste the following into the **Policy Document** field:

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

8. Choose **Apply Policy**.

Monitoring Deployments with Amazon CloudWatch Events

You can use Amazon CloudWatch Events to detect and react to changes in the state of an instance or a deployment (an "event") in your AWS CodeDeploy operations. Then, based on rules you create, CloudWatch Events will invoke one or more target actions when a deployment or instance enters the state you specify in a rule. Depending on the type of state change, you might want to send notifications,

capture state information, take corrective action, initiate events, or take other actions. You can select the following types of targets when using CloudWatch Events as part of your AWS CodeDeploy operations:

- AWS Lambda functions
- Kinesis streams
- Amazon SQS queues
- Built-in targets (CloudWatch alarm actions)
- Amazon SNS topics

The following are some use cases:

- Use a Lambda function to pass a notification to a Slack channel whenever deployments fail.
- Push data about deployments or instances to a Kinesis stream to support comprehensive, real-time status monitoring.
- Use CloudWatch alarm actions to automatically stop, terminate, reboot, or recover Amazon EC2 instances when a deployment or instance event you specify occurs.

The remainder of this topic describes the basic procedure for creating a CloudWatch Events rule for AWS CodeDeploy. Before you create event rules for use in your AWS CodeDeploy operations, however, you should do the following:

- Complete the CloudWatch Events prerequisites. For information, see [Amazon CloudWatch Events Prerequisites](#).
- Familiarize yourself with events, rules, and targets in CloudWatch Events. For more information, see [What Is Amazon CloudWatch Events?](#) and [New CloudWatch Events – Track and Respond to Changes to Your AWS Resources](#).
- Create the target or targets you will use in your event rules.

To create a CloudWatch Events rule for AWS CodeDeploy:

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Events**.
3. Choose **Create rule**, and then under **Event selector**, choose **AWS CodeDeploy**.
4. Specify a detail type:
 - To make a rule that applies to all state changes of both instances and deployments, choose **Any detail type**, and then skip to step 6.
 - To make a rule that applies to instances only, choose **Specific detail type**, and then choose **CodeDeploy Instance State-change Notification**.
 - To make a rule that applies to deployments only, choose **Specific detail type**, and then choose **CodeDeploy Deployment State-change Notification**.
5. Specify the state changes the rule applies to:
 - To make a rule that applies to all state changes, choose **Any state**.
 - To make a rule that applies to some state changes only, choose **Specific state(s)**, and then choose one or more status values from the list. The following table lists the status values you can choose:

Deployment status values	Instance status values
FAILURE	FAILURE
START	START

Deployment status values	Instance status values
STOP	READY
QUEUED	SUCCESS
READY	
SUCCESS	

6. Specify which AWS CodeDeploy applications the rule applies to:
 - To make a rule that applies to all applications, choose **Any application**, and then skip to step 8.
 - To make a rule that applies to one application only, choose **Specific application**, and then choose the name of the application from the list.
7. Specify which deployment groups the rule applies to:
 - To make a rule that applies to all deployment groups associated with the selected application, choose **Any deployment group**.
 - To make a rule that applies to only one of the deployment groups associated with the selected application, choose **Specific deployment group(s)**, and then choose the name of the deployment group from the list.
8. Review your rule setup to make sure it meets your event-monitoring requirements.

The following shows the setup for an event rule that will be processed whenever a deployment fails to any instance in the `MyDeploymentFleet` deployment group for the application named `MyCodeDeployApp`:

The screenshot shows the 'Event selector' configuration window in AWS CloudWatch. It is titled 'Event selector' with the subtitle 'Build a pattern that selects events for processing by your targets.' The configuration is set for 'AWS CodeDeploy' and includes the following settings:

- Any detail type** (radio button) and **Specific detail type** (radio button, selected). The dropdown menu shows 'CodeDeploy Deployment State-change Notification'.
- Any state** (radio button) and **Specific state(s)** (radio button, selected). The dropdown menu shows 'FAILURE' with a close button (x).
- Any application** (radio button) and **Specific application** (radio button, selected). The dropdown menu shows 'MyCodeDeployApp'.
- Any deployment group** (radio button) and **Specific deployment group(s)** (radio button, selected). The dropdown menu shows 'MyDeploymentFleet' with a close button (x).

9. In the **Targets** area, choose **Add target***.

10. In the **Select target type** list, choose the type of target you have prepared to use with this rule, and then configure any additional options required by that type.
11. Choose **Configure details**.
12. On the **Configure rule details** page, type a name and description for the rule, and then choose the **State** box to enable the rule now.
13. If you're satisfied with the rule, choose **Create rule**.

Monitoring Deployments with 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 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 which 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 [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 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, Amazon S3 server-side encryption (SSE) is used to encrypt your log files.

You can have CloudTrail publish Amazon SNS notifications when new log files are delivered. For more information, see [Configuring Amazon SNS Notifications for CloudTrail](#).

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 [Receiving CloudTrail Log Files from Multiple Regions](#).

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 ...
]
```

Monitoring Deployments with Amazon SNS Event Notifications

You can add triggers to an AWS CodeDeploy deployment group to receive notifications about events related to deployments or instances in that deployment group. These notifications are sent to recipients who are subscribed to an Amazon SNS topic you have made part of the trigger's action.

You can receive notifications for AWS CodeDeploy events in SMS messages or email messages. You can also use the JSON data that is created when a specified event occurs in other ways, such as sending messages to Amazon SQS queues or invoking a function in AWS Lambda. For a look at the structure

of the JSON data provided for deployment and instance triggers, see [JSON Data Formats for AWS CodeDeploy Triggers](#) (p. 287).

You might choose to use triggers to receive notifications if:

- You are a developer who needs to know when a deployment fails or stops so you can troubleshoot it.
- You are a system administrator who needs to know how many instances fail in order to monitor the health of your Amazon EC2 fleet.
- You are a manager who wants an at-a-glance count of deployment and instance events, which you can get through filtering rules that route different types of notifications into folders in your desktop email client.

You can create up to 10 triggers for each AWS CodeDeploy deployment group, for any of the following event types.

Deployment events	Instance events
<ul style="list-style-type: none">• Success• Failure• Started• Stopped• Rollback• Ready¹• All deployment events	<ul style="list-style-type: none">• Success• Failure• Started• Ready¹• All instance events
¹ Applies to blue/green deployments only. Indicates that the latest application revision has been installed on instances in a replacement environment and traffic from the original environment can now be rerouted behind a load balancer. For more information see Working with Deployments in AWS CodeDeploy (p. 244).	

Topics

- [Grant Amazon SNS Permissions to an AWS CodeDeploy Service Role](#) (p. 279)
- [Create a Trigger for an AWS CodeDeploy Event](#) (p. 280)
- [Edit a Trigger in an AWS CodeDeploy Deployment Group](#) (p. 285)
- [Delete a Trigger from an AWS CodeDeploy Deployment Group](#) (p. 286)
- [JSON Data Formats for AWS CodeDeploy Triggers](#) (p. 287)

Grant Amazon SNS Permissions to an AWS CodeDeploy Service Role

Before your triggers can generate notifications, the service role you use in your AWS CodeDeploy operations must be granted permission to access the Amazon SNS resources.

To grant Amazon SNS permissions to a service role

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, choose **Roles**.
3. Choose the name of the service role you use in your AWS CodeDeploy operations.
4. On the **Permissions** tab, in the **Inline Policies** area, choose **Create Role Policy**.

–or–

If the **Create Role Policy** button is not available, expand the **Inline Policies** area, and then choose **click here**.

5. On the **Set Permissions** page, choose **Custom Policy**, and then choose **Select**.
6. On the **Review Policy** page, in the **Policy Name** field, type a name to identify this policy, such as `SNSPublish`.
7. Paste the following into the **Policy Document** field:

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

8. Choose **Apply Policy**.

Create a Trigger for an AWS CodeDeploy Event

You can create a trigger that publishes an Amazon Simple Notification Service (Amazon SNS) topic for an AWS CodeDeploy deployment or instance event. Then, when that event occurs, all subscribers to the associated topic will receive notifications through the endpoint specified in the topic, such as an SMS message or email message. Amazon SNS offers multiple methods for subscribing to topics.

Before you create a trigger, you must set up the Amazon SNS topic to which the trigger will point. For information, see [Create a Topic](#). When you create a topic, we recommend you give it a name that will identify its purpose, in formats such as `Topic-group-us-west-3-deploy-fail` or `Topic-group-project-2-instance-stop`.

You must also grant Amazon SNS permissions to an AWS CodeDeploy service role before notifications can be sent for your trigger. For information, see [Grant Amazon SNS Permissions to an AWS CodeDeploy Service Role](#) (p. 279).

After you have created the topic, you can add subscribers. For information about creating, managing, and subscribing to topics, see [What Is Amazon Simple Notification Service](#).

Create a Trigger to Send Notifications for AWS CodeDeploy Events (Console)

You can use the AWS CodeDeploy console to create triggers for an AWS CodeDeploy event. At the end of the setup process, a test notification message is sent to ensure that both permissions and trigger details are set up correctly.

To create a trigger for an AWS CodeDeploy event

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. On the **Applications** page, choose the name of the application for which triggers will be sent.
3. On the **Applications details** page, choose the arrow next to the deployment group for which triggers will be sent.
4. In the **Triggers** area, choose **Create trigger**.
5. In the **Create trigger** pane, do the following:
 - In **Trigger name**, type a name for the trigger that makes it easy to identify its purpose. We recommend formats such as `Trigger-group-us-west-3-deploy-fail` or `Trigger-group-eu-central-instance-stop`.
 - In **Events**, choose the event type or types that will trigger the Amazon SNS topic to send notifications.
 - In **Amazon SNS topic**, choose the name of topic you created for sending notifications for this trigger.
6. Choose **Create trigger**.

AWS CodeDeploy will send a test notification to confirm you have correctly configured access between AWS CodeDeploy and the Amazon SNS topic. Depending on the endpoint type you selected for the topic, and if you are subscribed to the topic, you will receive confirmation in an SMS message or email message.

Create a Trigger to Send Notifications for AWS CodeDeploy Events (CLI)

You can use the CLI to include triggers when you create a deployment group, or you can add triggers to an existing deployment group.

To create a trigger to send notifications for a new deployment group

Create a JSON file to configure the deployment group, and then run the [create-deployment-group](#) command using the `--cli-input-json` option.

The simplest way to create the JSON file is to use the `--generate-cli-skeleton` option to get a copy of the JSON format, and then provide the required values in a plain-text editor.

1. Run the following command, and then copy the results into a plain-text editor.

```
aws deploy create-deployment-group --generate-cli-skeleton
```

2. Add the name of an existing AWS CodeDeploy application to the output:

```
{
  "applicationName": "TestApp-us-east-2",
  "deploymentGroupName": "",
  "deploymentConfigName": "",
  "ec2TagFilters": [
    {
      "Key": "",
      "Value": "",
      "Type": ""
    }
  ],
  "onPremisesInstanceTagFilters": [
    {
      "Key": "",
      "Value": "",
      "Type": ""
    }
  ]
}
```

```
    }
  ],
  "autoScalingGroups": [
    ""
  ],
  "serviceRoleArn": "",
  "triggerConfigurations": [
    {
      "triggerName": "",
      "triggerTargetArn": "",
      "triggerEvents": [
        ""
      ]
    }
  ]
}
```

3. Provide values for the parameters you want to configure.

When you use the `create-deployment-group` command, you must provide, at a minimum, values for the following parameters:

- `applicationName`: The name of an application already created in your account.
- `deploymentGroupName`: A name for the deployment group you are creating.
- `serviceRoleArn`: The ARN of an existing service role set up for AWS CodeDeploy in your account. For information, see [Step 3: Create a Service Role for AWS CodeDeploy \(p. 21\)](#).

In the `triggerConfigurations` section, provide values for the following parameters:

- `triggerName`: The name you want to give the trigger so you can easily identify it. We recommend formats such as `Trigger-group-us-west-3-deploy-fail` or `Trigger-group-eu-central-instance-stop`.
- `triggerTargetArn`: The ARN of the Amazon SNS topic you created to associate with your trigger, in this format: `arn:aws:sns:us-east-2:80398EXAMPLE:NewTestTopic`.
- `triggerEvents`: The type of event or events for which you want to trigger notifications. You can specify one or more event types, separating multiple event type names with commas (for example, `"triggerEvents": ["DeploymentSuccess", "DeploymentFailure", "InstanceFailure"]`). When you add more than one event type, notifications for all those types are sent to the topic you specified, rather than to a different topic for each one. You can choose from the following event types:
 - `DeploymentStart`
 - `DeploymentSuccess`
 - `DeploymentFailure`
 - `DeploymentStop`
 - `DeploymentRollback`
 - `DeploymentReady` (Applies only to replacement instances in a blue/green deployment)
 - `InstanceStart`
 - `InstanceSuccess`
 - `InstanceFailure`
 - `InstanceReady` (Applies only to replacement instances in a blue/green deployment)

The following configuration example creates a deployment group named `dep-group-ghi-789-2` for an application named `TestApp-us-east-2` and a trigger that will prompt the sending of notifications whenever a deployment starts, succeeds, or fails:

```
{
  "applicationName": "TestApp-us-east-2",
  "deploymentConfigName": "CodeDeployDefault.OneAtATime",
  "deploymentGroupName": "dep-group-ghi-789-2",
  "ec2TagFilters": [
    {
      "Key": "Name",
      "Value": "Project-ABC",
      "Type": "KEY_AND_VALUE"
    }
  ],
  "serviceRoleArn": "arn:aws:iam::444455556666:role/AnyCompany-service-role",
  "triggerConfigurations": [
    {
      "triggerName": "Trigger-group-us-east-2",
      "triggerTargetArn": "arn:aws:sns:us-east-2:80398EXAMPLE:us-east-
deployments",
      "triggerEvents": [
        "DeploymentStart",
        "DeploymentSuccess",
        "DeploymentFailure"
      ]
    }
  ]
}
```

4. Save your updates as a JSON file, and then call that file using the `--cli-input-json` option when you run the **create-deployment-group** command:

Important

Be sure to include `file://` before the file name. It is required in this command.

```
aws deploy create-deployment-group --cli-input-json file://filename.json
```

At the end of the creation process, you will receive a test notification message that indicates both permissions and trigger details are set up correctly.

To create a trigger to send notifications for an existing deployment group

To use the AWS CLI to add triggers for AWS CodeDeploy events to an existing deployment group, create a JSON file to update the deployment group, and then run the **update-deployment-group** command using the `--cli-input-json` option.

The simplest way to create the JSON file is to run the **get-deployment-group** command to get a copy of the deployment group's configuration, in JSON format, and then update the parameter values in a plain-text editor.

1. Run the following command, and then copy the results into a plain-text editor.

```
aws deploy get-deployment-group --application-name application --deployment-group-
name deployment-group
```

2. Delete the following from the output:
 - At the beginning of the output, delete `{ "deploymentGroupInfo": {`.
 - At the end of the output, delete `}`.
 - Delete the row containing `deploymentGroupId`.
 - Delete the row containing `deploymentGroupName`.

The contents of your text file should now look similar to the following:

```
{
  "applicationName": "TestApp-us-east-2",
  "deploymentConfigName": "CodeDeployDefault.OneAtATime",
  "autoScalingGroups": [],
  "ec2TagFilters": [
    {
      "Type": "KEY_AND_VALUE",
      "Value": "Project-ABC",
      "Key": "Name"
    }
  ],
  "triggerConfigurations": [],
  "serviceRoleArn": "arn:aws:iam::444455556666:role/AnyCompany-service-role",
  "onPremisesInstanceTagFilters": []
}
```

3. In the `triggerConfigurations` section, add data for the `triggerEvents`, `triggerTargetArn`, and `triggerName` parameters. For information about trigger configuration parameters, see [TriggerConfig](#).

The contents of your text file should now look similar to the following. This code will prompt notifications to be sent whenever a deployment starts, succeeds, or fails.

```
{
  "applicationName": "TestApp-us-east-2",
  "deploymentConfigName": "CodeDeployDefault.OneAtATime",
  "autoScalingGroups": [],
  "ec2TagFilters": [
    {
      "Type": "KEY_AND_VALUE",
      "Value": "Project-ABC",
      "Key": "Name"
    }
  ],
  "triggerConfigurations": [
    {
      "triggerEvents": [
        "DeploymentStart",
        "DeploymentSuccess",
        "DeploymentFailure"
      ],
      "triggerTargetArn": "arn:aws:sns:us-east-2:80398EXAMPLE:us-east-
deployments",
      "triggerName": "Trigger-group-us-east-2"
    }
  ],
  "serviceRoleArn": "arn:aws:iam::444455556666:role/AnyCompany-service-role",
  "onPremisesInstanceTagFilters": []
}
```

4. Save your updates as a JSON file, and then run the `update-deployment-group` command using the `--cli-input-json` option. Be sure to include the `--current-deployment-group-name` option and substitute the name of your JSON file for *filename*:

Important

Be sure to include `file://` before the file name. It is required in this command.

```
aws deploy update-deployment-group --current-deployment-group-name deployment-group-name --cli-input-json file://filename.json
```

At the end of the creation process, you will receive a test notification message that indicates both permissions and trigger details are set up correctly.

Edit a Trigger in an AWS CodeDeploy Deployment Group

If your notification requirements change, you can modify your trigger rather than create a new one.

Modify an AWS CodeDeploy Trigger (Console)

1. Sign in to the AWS Management Console and open the AWS CodeDeploy console at <https://console.aws.amazon.com/codedeploy>.
Note
Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).
2. On the **Applications** page, choose the name of the application associated with the deployment group where you will modify a trigger.
3. On the **Application details** page, choose the arrow next to the deployment group where you will edit a trigger.
4. In the **Triggers** area, locate the name of the trigger you want to modify, and then choose the pencil icon at the end of its row.
5. Update the trigger name, selected events, or Amazon SNS topic, and then choose **Save**.

Modify an AWS CodeDeploy Trigger (CLI)

To use the AWS CLI to change trigger details for AWS CodeDeploy events when you update a deployment group, create a JSON file to define changes to the deployment group's properties, and then run the `update-deployment-group` command with the `--cli-input-json` option.

The simplest way to create the JSON file is to run the `get-deployment-group` command to get the current deployment group details in JSON format, and then edit the required values in a plain-text editor.

1. Run the following command, substituting the names of your application and deployment group for *application* and *deployment-group*:

```
aws deploy get-deployment-group --application-name application --deployment-group-name deployment-group
```

2. Copy the results of the command into a plain-text editor and then delete the following:
 - At the beginning of the output, delete { "deploymentGroupInfo":.
 - At the end of the output, delete }.
 - Delete the row containing deploymentGroupId.
 - Delete the row containing deploymentGroupName.

The contents of your text file should now look similar to the following:

```
{
  "applicationName": "TestApp-us-east-2",
  "deploymentConfigName": "CodeDeployDefault.OneAtATime",
  "autoScalingGroups": [],
  "ec2TagFilters": [
    {
      "Type": "KEY_AND_VALUE",
      "Value": "East-1-Instances",
      "Key": "Name"
    }
  ],
  "triggerConfigurations": [
    {
      "triggerEvents": [
        "DeploymentStart",
        "DeploymentSuccess",
        "DeploymentFailure",
        "DeploymentStop"
      ],
      "triggerTargetArn": "arn:aws:sns:us-east-2:111222333444:Trigger-group-us-east-2",
      "triggerName": "Trigger-group-us-east-2"
    }
  ],
  "serviceRoleArn": "arn:aws:iam::444455556666:role/AnyCompany-service-role",
  "onPremisesInstanceTagFilters": []
}
```

3. Change any parameters, as necessary. For information about trigger configuration parameters, see [TriggerConfig](#).
4. Save your updates as a JSON file, and then run the [update-deployment-group](#) command using the `--cli-input-json` option. Be sure to include the `--current-deployment-group-name` option and substitute the name of your JSON file for *filename*:

Important

Be sure to include `file://` before the file name. It is required in this command.

```
aws deploy update-deployment-group --current-deployment-group-name deployment-group-name --cli-input-json file://filename.json
```

At the end of the creation process, you will receive a test notification message that indicates both permissions and trigger details are set up correctly.

Delete a Trigger from an AWS CodeDeploy Deployment Group


Because there is a limit of 10 triggers per deployment group, you might want to delete triggers if they are no longer being used. You cannot undo the deletion of a trigger, but you can re-create one.

Delete a Trigger from a Deployment Group (Console)

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

Note

Sign in with the same account or IAM user information you used in [Getting Started with AWS CodeDeploy](#) (p. 19).

2. On the **Applications** page, choose the application associated with the deployment group from which you want to delete a trigger.
3. On the **Application details** page, choose the arrow next to the deployment group.
4. In the **Triggers** area, locate the name of the trigger to delete, choose the  button at the end of its row, and then choose **Delete**.

Delete a Trigger from a Deployment Group (CLI)

To use the CLI to delete a trigger, call the `update-deployment-group` command, with empty trigger configuration parameters, specifying:

- The name of the application associated with the deployment group. To view a list of application names, call the `list-applications` command.
- The name of the deployment group associated with the application. To view a list of deployment group names, call the `list-deployment-groups` command.

For example:

```
aws deploy update-deployment-group --application-name application-name --current-  
deployment-group-name deployment-group-name --trigger-configurations
```

JSON Data Formats for AWS CodeDeploy Triggers

You can use the JSON output that is created when a trigger for a deployment or instance is activated in a custom notification workflow, such as sending messages to Amazon SQS queues or invoking a function in AWS Lambda.

Note

This guide does not address how to configure notifications using JSON. For information about using Amazon SNS to send messages to Amazon SQS queues, see [Sending Amazon SNS Messages to Amazon SQS Queues](#). For information about using Amazon SNS to invoke a Lambda function, see [Invoking Lambda Functions Using Amazon SNS Notifications](#).

The following examples show the structure of the JSON output available with AWS CodeDeploy triggers.

Sample JSON Output for Instance-Based Triggers

```
{  
  "region": "us-east-2",  
  "accountId": "111222333444",  
  "eventTriggerName": "trigger-group-us-east-instance-succeeded",  
  "deploymentId": "d-75I7MBT7C",  
  "instanceId": "arn:aws:ec2:us-east-2:444455556666:instance/i-496589f7",  
  "lastUpdatedAt": "1446744207.564",  
  "instanceStatus": "Succeeded",  
  "lifecycleEvents": [  
    {  
      "LifecycleEvent": "ApplicationStop",  
      "LifecycleEventStatus": "Succeeded",  
      "StartTime": "1446744188.595",  
      "EndTime": "1446744188.711"  
    },  
    {  
      "LifecycleEvent": "BeforeInstall",  
      "LifecycleEventStatus": "Succeeded",  
      "StartTime": "1446744189.827",  
      "EndTime": "1446744189.827"  
    }  
  ]  
}
```

```
        "EndTime": "1446744190.402"
      }
    //More lifecycle events might be listed here
  ]
}
```

Sample JSON Output for Deployment-Based Triggers

```
{
  "region": "us-west-1",
  "accountId": "111222333444",
  "eventTriggerName": "Trigger-group-us-west-3-deploy-failed",
  "applicationName": "ProductionApp-us-west-3",
  "deploymentId": "d-75I7MBT7C",
  "deploymentGroupName": "dep-group-def-456",
  "createTime": "1446744188.595",
  "completeTime": "1446744190.402",
  "deploymentOverview": {
    "Failed": "10",
    "InProgress": "0",
    "Pending": "0",
    "Skipped": "0",
    "Succeeded": "0"
  },
  "status": "Failed",
  "errorInformation": {
    "ErrorCode": "IAM_ROLE_MISSING",
    "ErrorMessage": "IAM Role is missing for deployment group: dep-group-def-456"
  }
}
```


Authentication and Access Control for AWS CodeDeploy

Access to AWS CodeDeploy requires credentials. Those credentials must have permissions to access AWS resources, such as retrieving application revisions from Amazon S3 buckets and reading the tags on Amazon EC2 instances. The following sections provide details on how you can use [AWS Identity and Access Management \(IAM\)](#) and AWS CodeDeploy to help secure access to your resources:

- [Authentication](#) (p. 289)
- [Access Control](#) (p. 290)

Authentication

You can access AWS as any of the following types of identities:

- **AWS account root user** – When you sign up for AWS, you provide an email address and password that is associated with your AWS account. These are your *root credentials* and they provide complete access to all of your AWS resources.

Important

For security reasons, we recommend that you use the root credentials only to create an *administrator user*, which is an *IAM user* with full permissions to your AWS account. Then, you can use this administrator user to create other IAM users and roles with limited permissions. For more information, see [IAM Best Practices](#) and [Creating an Admin User and Group](#) in the *IAM User Guide*.

- **IAM user** – An [IAM user](#) is simply an identity within your AWS account that has specific custom permissions (for example, permissions to send event data to a target in AWS CodeDeploy). You can use an IAM user name and password to sign in to secure AWS webpages like the [AWS Management Console](#), [AWS Discussion Forums](#), or the [AWS Support Center](#).

In addition to a user name and password, you can also generate [access keys](#) for each user. You can use these keys when you access AWS services programmatically, either through [one of the several SDKs](#) or by using the [AWS Command Line Interface \(AWS CLI\)](#). The SDK and CLI tools use the access keys to cryptographically sign your request. If you don't use the AWS tools, you must sign the request yourself. AWS CodeDeploy supports *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see [Signature Version 4 Signing Process](#) in the *AWS General Reference*.

- **IAM role** – An [IAM role](#) is another IAM identity you can create in your account that has specific permissions. It is similar to an *IAM user*, but it is not associated with a specific person. An IAM role enables you to obtain temporary access keys that can be used to access AWS services and resources. IAM roles with temporary credentials are useful in the following situations:
- **Federated user access** – Instead of creating an IAM user, you can use preexisting user identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an [identity](#)

[provider](#). For more information about federated users, see [Federated Users and Roles](#) in the *IAM User Guide*.

- **Cross-account access** – You can use an IAM role in your account to grant another AWS account permissions to access your account's resources. For an example, see [Tutorial: Delegate Access Across AWS Accounts Using IAM Roles](#) in the *IAM User Guide*.
- **AWS service access** – You can use an IAM role in your account to grant an AWS service permissions to access your account's resources. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data stored in the bucket into an Amazon Redshift cluster. For more information, see [Creating a Role to Delegate Permissions to an AWS Service](#) in the *IAM User Guide*.
- **Applications running on Amazon EC2** – Instead of storing access keys within the EC2 instance for use by applications running on the instance and making AWS API requests, you can use an IAM role to manage temporary credentials for these applications. To assign an AWS role to an EC2 instance and make it available to all of its applications, you can create an instance profile that is attached to the instance. An instance profile contains the role and enables programs running on the EC2 instance to get temporary credentials. For more information, see [Using Roles for Applications on Amazon EC2](#) in the *IAM User Guide*.

Access Control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot create or access AWS CodeDeploy resources. For example, you must have permissions to create, view, or delete applications, deployments, deployment configurations, and deployment groups; to register, deregister, add tags to, or remove tags from on-premises instances; and so on.

The following sections describe how to manage permissions for AWS CodeDeploy. We recommend that you read the overview first.

- [Overview of Managing Access Permissions to Your AWS CodeDeploy Resources](#) (p. 290)
- [Using Identity-Based Policies \(IAM Policies\) for AWS CodeDeploy](#) (p. 295)
- [AWS CodeDeploy Permissions Reference](#) (p. 299)

Overview of Managing Access Permissions to Your AWS CodeDeploy Resources

Every AWS resource is owned by an AWS account, and permissions to create or access a resource are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles), and some services (such as AWS Lambda) also support attaching permissions policies to resources.

Note

An *account administrator* (or administrator user) is a user with administrator privileges. For more information, see [IAM Best Practices](#) in the *IAM User Guide*.

When granting permissions, you decide who is getting the permissions, the resources they get permissions for, and the specific actions that you want to allow on those resources.

Topics

- [AWS CodeDeploy Resources and Operations \(p. 291\)](#)
- [Understanding Resource Ownership \(p. 292\)](#)
- [Managing Access to Resources \(p. 292\)](#)
- [Specifying Policy Elements: Actions, Effects, and Principals \(p. 294\)](#)
- [Specifying Conditions in a Policy \(p. 295\)](#)

AWS CodeDeploy Resources and Operations

In AWS CodeDeploy, the primary resource is a deployment group. In a policy, you use an Amazon Resource Name (ARN) to identify the resource that the policy applies to. AWS CodeDeploy supports other resources that can be used with deployment groups, including applications, deployment configurations and instances. These are referred to as subresources. These resources and subresources have unique Amazon Resource Names (ARNs) associated with them. For more information about ARNs, see [Amazon Resource Names \(ARN\)](#) and [AWS Service Namespaces](#) in the *Amazon Web Services General Reference*.

Resource Type	ARN Format
Deployment group	arn:aws:codedeploy: <i>region</i> : <i>account-id</i> :deploymentgroup/ <i>deployment-group-name</i>
Application	arn:aws:codedeploy: <i>region</i> : <i>account-id</i> :application/ <i>application-name</i>
Deployment configuration	arn:aws:codedeploy: <i>region</i> : <i>account-id</i> :deploymentconfig/ <i>deployment-configuration-name</i>
Instance	arn:aws:codedeploy: <i>region</i> : <i>account-id</i> :instance/ <i>instance-ID</i>
All AWS CodeDeploy resources	arn:aws:codedeploy:*
All AWS CodeDeploy resources owned by the specified account in the specified region	arn:aws:codedeploy: <i>region</i> : <i>account-id</i> :*

Note

Most services in AWS treat a colon (:) or a forward slash (/) as the same character in ARNs. However, AWS CodeDeploy uses an exact match in resource patterns and rules. Be sure to use the correct ARN characters when creating event patterns so that they match the ARN syntax in the resource.

For example, you can indicate a specific deployment group (*myDeploymentGroup*) in your statement using its ARN as follows:

```
"Resource": "arn:aws:codedeploy:us-west-2:123456789012:deploymentgroup/myDeploymentGroup"
```

You can also specify all deployment groups that belong to a specific account by using the wildcard character (*) as follows:

```
"Resource": "arn:aws:codedeploy:us-west-2:123456789012:deploymentgroup/*"
```

To specify all resources, or if a specific API action does not support ARNs, use the wildcard character (*) in the Resource element as follows:

```
"Resource": "*"
```

Some AWS CodeDeploy API actions accept multiple resources (for example, BatchGetDeploymentGroups). To specify multiple resources in a single statement, separate their ARNs with commas, as follows:

```
"Resource": ["arn1", "arn2"]
```

AWS CodeDeploy provides a set of operations to work with the AWS CodeDeploy resources. For a list of available operations, see [AWS CodeDeploy Permissions Reference](#) (p. 299).

Understanding Resource Ownership

The AWS account owns the resources that are created in the account, regardless of who created the resources. Specifically, the resource owner is the AWS account of the [principal entity](#) (that is, the root account, an IAM user, or an IAM role) that authenticates the resource creation request. The following examples illustrate how this works:

- If you use the root account credentials of your AWS account to create a rule, your AWS account is the owner of the AWS CodeDeploy resource.
- If you create an IAM user in your AWS account and grant permissions to create AWS CodeDeploy resources to that user, the user can create AWS CodeDeploy resources. However, your AWS account, to which the user belongs, owns the AWS CodeDeploy resources.
- If you create an IAM role in your AWS account with permissions to create AWS CodeDeploy resources, anyone who can assume the role can create AWS CodeDeploy resources. Your AWS account, to which the role belongs, owns the AWS CodeDeploy resources.

Managing Access to Resources

A *permissions policy* describes who has access to what. The following section explains the available options for creating permissions policies.

Note

This section discusses using IAM in the context of AWS CodeDeploy. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see [What Is IAM?](#) in the *IAM User Guide*. For information about IAM policy syntax and descriptions, see [AWS IAM Policy Reference](#) in the *IAM User Guide*.

Policies attached to an IAM identity are referred to as identity-based policies (IAM policies) and policies attached to a resource are referred to as resource-based policies. AWS CodeDeploy supports only identity-based (IAM policies).

Topics

- [Identity-Based Policies \(IAM Policies\)](#) (p. 292)
- [Resource-Based Policies](#) (p. 294)

Identity-Based Policies (IAM Policies)

You can attach policies to IAM identities. For example, you can do the following:

- **Attach a permissions policy to a user or a group in your account** – To grant a user permissions to view applications, deployment groups, and other AWS CodeDeploy resources in the AWS CodeDeploy console, you can attach a permissions policy to a user or group that the user belongs to.
- **Attach a permissions policy to a role (grant cross-account permissions)** – You can attach an identity-based permissions policy to an IAM role to grant cross-account permissions. For example, the administrator in Account A can create a role to grant cross-account permissions to another AWS account (for example, Account B) or an AWS service as follows:
 1. Account A administrator creates an IAM role and attaches a permissions policy to the role that grants permissions on resources in Account A.
 2. Account A administrator attaches a trust policy to the role identifying Account B as the principal who can assume the role.
 3. Account B administrator can then delegate permissions to assume the role to any users in Account B. Doing this allows users in Account B to create or access resources in Account A. The principal in the trust policy can also be an AWS service principal if you want to grant an AWS service permissions to assume the role.

For more information about using IAM to delegate permissions, see [Access Management](#) in the *IAM User Guide*.

In AWS CodeDeploy, identity-based policies are used to manage permissions to the various resources related to the deployment process. You can control access to all the following resource types:

- Applications and application revisions
- Deployments
- Deployment configurations
- Instances and on-premises instances

The capabilities controlled by resource-based policies vary depending on the resource type, as outlined in the following table:

Resource types	Capabilities
All	View and list details about resources
Applications	Create resources
Deployment configurations	Delete resources
Deployment groups	
Deployments	Create deployments
	Stop deployments
Application revisions	Register application revisions

Resource types	Capabilities
Applications Deployment groups	Update resources
On-premises instances	Add tags to instances Remove tags from instances Register instances Deregister instances

You can create specific IAM policies to restrict the calls and resources that users in your account have access to, and then attach those policies to IAM users. For more information about how to create IAM roles and to explore example IAM policy statements for AWS CodeDeploy, see [Overview of Managing Access Permissions to Your AWS CodeDeploy Resources \(p. 290\)](#).

Resource-Based Policies

Other services, such as Amazon S3, also support resource-based permissions policies. For example, you can attach a policy to an S3 bucket to manage access permissions to that bucket. AWS CodeDeploy doesn't support resource-based policies.

Specifying Policy Elements: Actions, Effects, and Principals

For each AWS CodeDeploy resource, the service defines a set of API operations. To grant permissions for these API operations, AWS CodeDeploy defines a set of actions that you can specify in a policy. Some API operations can require permissions for more than one action in order to perform the API operation. For more information about resources and API operations, see [AWS CodeDeploy Resources and Operations \(p. 291\)](#) and [AWS CodeDeploy Permissions Reference \(p. 299\)](#).

The following are the basic policy elements:

- **Resource** – You use an Amazon Resource Name (ARN) to identify the resource that the policy applies to. For more information, see [AWS CodeDeploy Resources and Operations \(p. 291\)](#).
- **Action** – You use action keywords to identify resource operations that you want to allow or deny. For example, the `codedeploy:GetApplication` permission allows the user permissions to perform the `GetApplication` operation.
- **Effect** – You specify the effect, either allow or deny, when the user requests the specific action. If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource, which you might do to make sure that a user cannot access it, even if a different policy grants access.
- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions (applies to resource-based policies only).

To learn more about IAM policy syntax and descriptions, see [AWS IAM Policy Reference](#) in the *IAM User Guide*.

For a table showing all of the AWS CodeDeploy API actions and the resources that they apply to, see [AWS CodeDeploy Permissions Reference \(p. 299\)](#).

Specifying Conditions in a Policy

When you grant permissions, you can use the access policy language to specify the conditions when a policy should take effect. For example, you might want a policy to be applied only after a specific date. For more information about specifying conditions in a policy language, see [Condition](#) in the *IAM User Guide*.

To express conditions, you use predefined condition keys. There are no condition keys specific to AWS CodeDeploy. However, there are AWS-wide condition keys that you can use as appropriate. For a complete list of AWS-wide keys, see [Available Keys for Conditions](#) in the *IAM User Guide*.

Using Identity-Based Policies (IAM Policies) for AWS CodeDeploy

This topic provides examples of identity-based policies that demonstrate how an account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles) and thereby grant permissions to perform operations on AWS CodeDeploy resources. For information about the policy that must be attached to an IAM user in order to use AWS CodeDeploy, see [Step 1: Provision an IAM User](#) (p. 19).

Important

We recommend that you first review the introductory topics that explain the basic concepts and options available to manage access to your AWS CodeDeploy resources. For more information, see [Overview of Managing Access Permissions to Your AWS CodeDeploy Resources](#) (p. 290).

Topics

- [Permissions Required to Use the AWS CodeDeploy Console](#) (p. 296)
- [AWS Managed \(Predefined\) Policies for AWS CodeDeploy](#) (p. 296)
- [Customer Managed Policy Examples](#) (p. 297)

The following shows an example of a permissions policy that allows a user to delete the deployment group named **WordPress_DepGroup** associated with the application named **WordPress_App** in the **us-west-2** region.

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:DeleteDeploymentGroup"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:80398EXAMPLE:deploymentgroup:WordPress_App/WordPress_DepGroup"
      ]
    }
  ]
}
```

Permissions Required to Use the AWS CodeDeploy Console

For a user to work with the AWS CodeDeploy console, that user must have a minimum set of permissions that allows the user to describe other AWS resources for their AWS account. In order to use fully use AWS CodeDeploy in the AWS CodeDeploy console, you must have permissions from the following services:

- Amazon EC2 Auto Scaling
- AWS CodeDeploy
- Amazon Elastic Compute Cloud
- Elastic Load Balancing
- AWS Identity and Access Management
- Amazon Simple Storage Service
- Amazon Simple Notification Service
- Amazon CloudWatch

If you create an IAM policy that is more restrictive than the minimum required permissions, the console won't function as intended for users with that IAM policy. To ensure that those users can still use the AWS CodeDeploy console, also attach the `AWSCodeDeployReadOnlyAccess` managed policy to the user, as described in [AWS Managed \(Predefined\) Policies for AWS CodeDeploy \(p. 296\)](#).

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS CodeDeploy API.

AWS Managed (Predefined) Policies for AWS CodeDeploy

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. These AWS managed policies grant necessary permissions for common use cases so you can avoid having to investigate what permissions are needed. For more information, see [AWS Managed Policies](#) in the *IAM User Guide*.

The following AWS managed policies, which you can attach to users in your account, are specific to AWS CodeDeploy:

- **AWSCodeDeployFullAccess** – Grants full access to AWS CodeDeploy.

Note

`AWSCodeDeployFullAccess` does not provide permissions to operations in other services required to deploy your applications, such as Amazon EC2 and Amazon S3, only to operations specific to AWS CodeDeploy.

- **AWSCodeDeployDeployerAccess** – Grants access to an IAM user to register and deploy revisions.
- **AWSCodeDeployReadOnlyAccess** – Grants read-only access to AWS CodeDeploy.
- **AWSCodeDeployRole** – Allows AWS CodeDeploy to identify Amazon EC2 instances by their Amazon EC2 tags or Auto Scaling group names, and on-premises instances by their on-premises instance tags, and to deploy application revisions to them accordingly. Provides permissions needed to publish notification to an Amazon SNS topic and retrieve information about alarms from CloudWatch.
- **AWSCodeDeployRoleForLambda** – Grants AWS CodeDeploy permission to access to AWS Lambda.

Permissions for some aspects of the deployment process are granted to two other role types that act on behalf of AWS CodeDeploy, rather than to IAM users:

- **IAM instance profile:** An IAM role that you attach to your Amazon EC2 instances. This profile includes the permissions required to access the Amazon S3 buckets or GitHub repositories where the applications that will be deployed by AWS CodeDeploy are stored. For more information, see [Step 4: Create an IAM Instance Profile for Your Amazon EC2 Instances \(p. 25\)](#).
- **Service role:** An IAM role that grants permissions to an AWS service so it can access AWS resources. The policies you attach to the service role determine which AWS resources the service can access and the actions it can perform with those resources. For AWS CodeDeploy, a service role is used for the following:
 - To read either the tags applied to the instances or the Amazon EC2 Auto Scaling group names associated with the instances. This enables AWS CodeDeploy to identify instances to which it can deploy applications.
 - To perform operations on instances, Auto Scaling groups, and Elastic Load Balancing load balancers.
 - To publish information to Amazon SNS topics so that notifications can be sent when specified deployment or instance events occur.
 - To retrieve information about CloudWatch alarms in order to set up alarm monitoring for deployments.

For more information, see [Step 3: Create a Service Role for AWS CodeDeploy \(p. 21\)](#).

You can also create your own custom IAM policies to allow permissions for AWS CodeDeploy actions and resources. You can attach these custom policies to the IAM users or groups that require those permissions.

Customer Managed Policy Examples

In this section, you can find example user policies that grant permissions for various AWS CodeDeploy actions. These policies work when you are using the AWS CodeDeploy API, AWS SDKs, or the AWS CLI. When you are using the console, you need to grant additional permissions specific to the console, which is discussed in [Permissions Required to Use the AWS CodeDeploy Console \(p. 296\)](#).

You can use the following sample IAM policies listed to limit the AWS CodeDeploy access for your IAM users and roles.

Note

All examples use the US West (Oregon) Region (us-west-2) and contain fictitious account IDs.

Examples

- [Example 1: Allow a User to Perform AWS CodeDeploy Operations in a Single Region \(p. 297\)](#)
- [Example 2: Allow a User to Register Revisions for a Single Application \(p. 298\)](#)
- [Example 3: Allow a User to Create Deployments for a Single Deployment Group \(p. 298\)](#)

Example 1: Allow a User to Perform AWS CodeDeploy Operations in a Single Region

The following example grants permissions to perform AWS CodeDeploy operations in the **us-west-2** region only:

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

```
{
  "Effect" : "Allow",
  "Action" : [
    "codedeploy:*"
  ],
  "Resource" : [
    "arn:aws:codedeploy:us-west-2:80398EXAMPLE:*"
  ]
}
```

Example 2: Allow a User to Register Revisions for a Single Application

The following example grants permissions to register application revisions for all applications that begin with **Test** in the **us-west-2** region:

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:RegisterApplicationRevision"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:80398EXAMPLE:application:Test*"
      ]
    }
  ]
}
```

Example 3: Allow a User to Create Deployments for a Single Deployment Group

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 application revisions associated with the application named **WordPress_App**. All of these resources are associated with the **us-west-2** region.

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:CreateDeployment"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:80398EXAMPLE:deploymentgroup:WordPress_App/WordPress_DepGroup"
      ]
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetDeploymentConfig"
      ]
    }
  ]
}
```

```
    ],
    "Resource" : [
        "arn:aws:codedeploy:us-west-2:80398EXAMPLE:deploymentconfig:ThreeQuartersHealthy"
    ]
  },
  {
    "Effect" : "Allow",
    "Action" : [
        "codedeploy:GetApplicationRevision"
    ],
    "Resource" : [
        "arn:aws:codedeploy:us-west-2:80398EXAMPLE:application:WordPress_App"
    ]
  }
]
```

AWS CodeDeploy Permissions Reference

When you are setting up [Access Control \(p. 290\)](#) and writing permissions policies that you can attach to an IAM identity (identity-based policies), you can use the following table as a reference. The table lists each AWS CodeDeploy API operation, the corresponding actions for which you can grant permissions to perform the action, and the format of the resource ARN to use for granting permissions. You specify the actions in the policy's `Action` field, and you specify an ARN, with or without a wildcard character (*), as the resource value in the policy's `Resource` field.

You can use AWS-wide condition keys in your AWS CodeDeploy policies to express conditions. For a complete list of AWS-wide keys, see [Available Keys](#) in the *IAM User Guide*.

To specify an action, use the `codedeploy:` prefix followed by the API operation name (for example, `codedeploy:GetApplication` and `codedeploy:CreateApplication`). To specify multiple actions in a single statement, separate them with commas (for example, `"Action": ["codedeploy:action1", "codedeploy:action2"]`).

Using Wildcard Characters

You can use a wildcard character (*) in your ARN to specify multiple actions or resources. For example, `codedeploy:*` specifies all AWS CodeDeploy actions and `codedeploy:Get*` specifies all AWS CodeDeploy actions that begin with the word `Get`. The following example grants access to all deployment groups with names that begin with `west` and are associated with applications that have names beginning with `Test`.

```
arn:aws:codedeploy:us-west-2:80398EXAMPLE:deploymentgroup:Test*/West*
```

You can use wildcards with the following resources listed in the table:

- *application-name*
- *deployment-group-name*
- *deployment-configuration-name*
- *instance-ID*

Wildcards can't be used with *region* or *account-id*. For more information about wildcards, see [IAM Identifiers](#) in *IAM User Guide*.

The actions you can specify in an IAM policy for use with AWS CodeDeploy are listed below.

Note

In the ARN for each action a colon (:) follows the resource. You can also follow the resource with a forward slash (/). For more information, see [AWS CodeDeploy Example ARNs](#).

AWS CodeDeploy API Operations and Required Permissions for Actions**AddTagsToOnPremisesInstances**

Action(s): codedeploy:AddTagsToOnPremisesInstances

Required to add tags to one or more on-premises instances.

Resource: arn:aws:codedeploy:*region*:*account-id*:instance/*instance-ID*

BatchGetApplicationRevisions

Action(s): codedeploy:BatchGetApplicationRevisions

Required to get information about multiple application revisions associated with the IAM user.

Resource: arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

BatchGetApplications

Action(s): codedeploy:BatchGetApplications

Required to get information about multiple applications associated with the IAM user.

Resource: arn:aws:codedeploy:*region*:*account-id*:application:*

BatchGetDeploymentGroups

Action(s): codedeploy:BatchGetDeploymentGroups

Required to get information about multiple deployment groups associated with the IAM user.

Resource: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

BatchGetDeploymentInstances

Action(s): codedeploy:BatchGetDeploymentInstances

Required to get information about one or more instance that are part of a deployment group.

Resource: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

BatchGetDeployments

Action(s): codedeploy:BatchGetDeployments

Required to get information about multiple deployments associated with the IAM user.

Resource: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

BatchGetOnPremisesInstances

Action(s): codedeploy:BatchGetOnPremisesInstances

Required to get information about one or more on-premises instances.

Resource: arn:aws:codedeploy:*region*:*account-id*:*

ContinueDeployment

Action(s): codedeploy:ContinueDeployment

Required during a blue/green deployment to start registering instances in a replacement environment with Elastic Load Balancing load balancers.

Resource: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name/deployment-group-name*

CreateApplication

Action(s): codedeploy>CreateApplication

Required to create an application associated with the IAM user.

Resource: arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

CreateDeployment¹

Action(s): codedeploy>CreateDeployment

Required to create a deployment for an application associated with the IAM user.

Resource: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name/deployment-group-name*

CreateDeploymentConfig

Action(s): codedeploy>CreateDeploymentConfig

Required to create a custom deployment configuration associated with the IAM user.

Resource: arn:aws:codedeploy:*region*:*account-id*:deploymentconfig/*deployment-configuration-name*

CreateDeploymentGroup

Action(s): codedeploy>CreateDeploymentGroup

Required to create a deployment group for an application associated with the IAM user.

Resource: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name/deployment-group-name*

DeleteApplication

Action(s): codedeploy>DeleteApplication

Required to delete an application associated with the IAM user.

Resource: arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

DeleteDeploymentConfig

Action(s): codedeploy>DeleteDeploymentConfig

Required to delete a custom deployment configuration associated with the IAM user.

Resource: arn:aws:codedeploy:*region*:*account-id*:deploymentconfig/*deployment-configuration-name*

DeleteDeploymentGroup

Action(s): codedeploy>DeleteDeploymentGroup

Required to delete a deployment group for an application associated with the IAM user.

Resource: `arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/deployment-group-name`

DeregisterOnPremisesInstance

Action(s): `codedeploy:DeregisterOnPremisesInstance`

Required to deregister an on-premises instance.

Resource: `arn:aws:codedeploy:region:account-id:instance/instance-ID`

GetApplication

Action(s): `codedeploy:GetApplication`

Required to get information about a single application associated with the IAM user.

Resource: `arn:aws:codedeploy:region:account-id:application:application-name`

GetApplicationRevision

Action(s): `codedeploy:GetApplicationRevision`

Required to get information about a single application revision for an application associated with the IAM user.

Resource: `arn:aws:codedeploy:region:account-id:application:application-name`

GetDeployment

Action(s): `codedeploy:GetDeployment`

Required to get information about a single deployment to a deployment group for an application associated with the IAM user.

Resource: `arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/deployment-group-name`

GetDeploymentConfig

Action(s): `codedeploy:GetDeploymentConfig`

Required to get information about a single deployment configuration associated with the IAM user.

Resource: `arn:aws:codedeploy:region:account-id:deploymentconfig/deployment-configuration-name`

GetDeploymentGroup

Action(s): `codedeploy:GetDeploymentGroup`

Required to get information about a single deployment group for an application associated with the IAM user.

Resource: `arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/deployment-group-name`

GetDeploymentInstance

Action(s): `codedeploy:GetDeploymentInstance`

Required to get information about a single instance in a deployment associated with the IAM user.

Resource: `arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/deployment-group-name`

GetOnPremisesInstance

Action(s): codedeploy:GetOnPremisesInstance

Required to get information about a single on-premises instance.

Resource: arn:aws:codedeploy:*region*:*account-id*:instance/*instance-ID*

ListApplicationRevisions

Action(s): codedeploy:ListApplicationRevisions

Required to get information about all application revisions for an application associated with the IAM user.

Resource: arn:aws:codedeploy:*region*:*account-id*:application:*

ListApplications

Action(s): codedeploy:ListApplications

Required to get information about all applications associated with the IAM user.

Resource: arn:aws:codedeploy:*region*:*account-id*:application:*

ListDeploymentConfigs

Action(s): codedeploy:ListDeploymentConfigs

Required to get information about all deployment configurations associated with the IAM user.

Resource: arn:aws:codedeploy:*region*:*account-id*:deploymentconfig/*

ListDeploymentGroups

Action(s): codedeploy:ListDeploymentGroups

Required to get information about all deployment groups for an application associated with the IAM user.

Resource: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*

ListDeploymentInstances

Action(s): codedeploy:ListDeploymentInstances

Required to get information about all instances in a deployment associated with the IAM user or AWS account.

Resource: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

ListDeployments

Action(s): codedeploy:ListDeployments

Required 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 or AWS account.

Resource: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

ListGitHubAccountTokenNames

Action(s): codedeploy:ListGitHubAccountTokenNames

Required to get a list of the names of stored connections to GitHub accounts.

Resource: `arn:aws:codedeploy:region:account-id:*`

ListOnPremisesInstances

Action(s): `codedeploy:ListOnPremisesInstances`

Required to get a list of one or more on-premises instance names.

Resource: `arn:aws:codedeploy:region:account-id:*`

RegisterApplicationRevision

Action(s): `codedeploy:RegisterApplicationRevision`

Required to register information about an application revision for an application associated with the IAM user.

Resource: `arn:aws:codedeploy:region:account-id:application:application-name`

RegisterOnPremisesInstance

Action(s): `codedeploy:RegisterOnPremisesInstance`

Required to register an on-premises instance with AWS CodeDeploy.

Resource: `arn:aws:codedeploy:region:account-id:instance/instance-ID`

RemoveTagsFromOnPremisesInstances

Action(s): `codedeploy:RemoveTagsFromOnPremisesInstances`

Required to remove tags from one or more on-premises instances.

Resource: `arn:aws:codedeploy:region:account-id:instance/instance-ID`

SkipWaitTimeForInstanceTermination

Action(s): `codedeploy:SkipWaitTimeForInstanceTermination`

Required to override a specified wait time and begin terminating instances in the original environment immediately after the traffic routing is successfully completed in a blue/green deployment.

Resource: `arn:aws:codedeploy:region:account-id:instance/instance-ID`

StopDeployment

Action(s): `codedeploy:StopDeployment`

Required to stop an in-progress deployment to a deployment group for an application associated with the IAM user.

Resource: `arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/deployment-group-name`

UpdateApplication³

Action(s): `codedeploy:UpdateApplication`

Required to change information about an application associated with the IAM user.

Resource: `arn:aws:codedeploy:region:account-id:application:application-name`

UpdateDeploymentGroup³

Action(s): `codedeploy:UpdateDeploymentGroup`

Required to change information about a single deployment group for an application associated with the IAM user.

Resource: `arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/deployment-group-name`

¹ When you specify `CreateDeployment` permissions, you must also specify `GetDeploymentConfig` permissions for the deployment configuration and `GetApplicationRevision` or `RegisterApplicationRevision` permissions for the application revision.

² Valid for `ListDeployments` when providing a specific deployment group, but not when listing all of the deployments associated with the IAM user)

³ For `UpdateApplication`, you must have `UpdateApplication` permissions for both the old application name and the new application name. For `UpdateDeploymentGroup` actions that involve changing a deployment group's name, you must have `UpdateDeploymentGroup` permissions for both the old and new deployment group name.

AWS CodeDeploy AppSpec File Reference

This section is a reference only. For a conceptual overview of the AppSpec file, see [Application Specification Files](#) (p. 17).

The application specification file (AppSpec file) is a [YAML](#)-formatted or JSON-formatted file used by AWS CodeDeploy to manage a deployment.

Topics

- [AppSpec Files on an AWS Lambda Compute Platform](#) (p. 306)
- [AppSpec Files on an EC2/On-Premises Compute Platform](#) (p. 306)
- [AppSpec File Structure](#) (p. 307)
- [AppSpec File Example](#) (p. 325)
- [AppSpec File Spacing](#) (p. 327)
- [Validate Your AppSpec File and File Location](#) (p. 328)

AppSpec Files on an AWS Lambda Compute Platform

If your application uses the AWS Lambda compute platform, the AppSpec file is used by AWS CodeDeploy to determine:

- Which Lambda function version to deploy.
- Which Lambda functions to use as validation tests.

An AppSpec file can be YAML-formatted or JSON-formatted. You can also enter the contents of an AppSpec file directly into AWS CodeDeploy console when you create a deployment.

AppSpec Files on an EC2/On-Premises Compute Platform

If your application uses the EC2/On-Premises compute platform, the AppSpec file is used by AWS CodeDeploy to determine:

- What it should install onto your instances from your application revision in Amazon S3 or GitHub.
- Which lifecycle event hooks to run in response to deployment lifecycle events.

An AppSpec file must be a YAML-formatted file named `appspec.yml` and it must be placed in the root of the directory structure of an application's source code. Otherwise, deployments fail.

After you have a completed AppSpec file, you bundle it, along with the content to deploy, into an archive file (zip, tar, or compressed tar). For more information, see [Working with Application Revisions for AWS CodeDeploy \(p. 232\)](#).

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*), you upload it to an Amazon S3 bucket or Git repository. Then you use AWS CodeDeploy to deploy the revision. For instructions, see [Create a Deployment with AWS CodeDeploy \(p. 245\)](#).

The appspec.yml for an EC2/On-Premises compute platform deployment is saved in the root directory of your revision. For more information, see [Add an AppSpec File for an EC2/On-Premises Deployment \(p. 234\)](#) and [Plan a Revision for AWS CodeDeploy \(p. 232\)](#).

AppSpec File Structure

The following is the high-level structure for an AppSpec file used for deployments to AWS Lambda and EC2/On-Premises compute platforms.

A value in a YAML-formatted AppSpec file that is a string must not be wrapped in quotation marks (""), unless otherwise specified.

AppSpec File Structure for AWS LambdaDeployments

Note

This AppSpec file is written in YAML, but you can use the same structure to write an AppSpec file for a Lambda deployment in JSON. A string in a JSON-formatted AppSpec file is always wrapped in quotation marks ("").

```
version: 0.0
resources:
  lambda-function-specifications
hooks:
  deployment-lifecycle-event-mappings
```

In this structure:

version

This section specifies the version of the AppSpec file. Do not change this value. It is required. Currently, the only allowed value is **0.0**. It is reserved by AWS CodeDeploy for future use.

Specify **version** with a string.

resources

This section specifies information about the Lambda function to deploy.

For more information, see [AppSpec 'resources' Section \(AWS Lambda Deployments Only\) \(p. 312\)](#).

hooks

This section specifies Lambda functions to run at specific deployment lifecycle events to validate the deployment.

For more information, see [AppSpec 'hooks' Section \(p. 316\)](#).

AppSpec File Structure for EC2/On-Premises Deployments

```
version: 0.0
os: operating-system-name
files:
  source-destination-files-mappings
permissions:
  permissions-specifications
hooks:
  deployment-lifecycle-event-mappings
```

In this structure:

version

This section specifies the version of the AppSpec file. Do not change this value. It is required. Currently, the only allowed value is **0.0**. It is reserved by AWS CodeDeploy for future use.

Specify **version** with a string.

os

This section specifies the operating system value of the instance to which you deploy. It is required. The following values can be specified:

- **linux** – The instance is an Amazon Linux, Ubuntu Server, or RHEL instance.
- **windows** – The instance is a Windows Server instance.

Specify **os** with a string.

files

This section specifies the names of files that should be copied to the instance during the deployment's **Install** event.

For more information, see [AppSpec 'files' Section \(EC2/On-Premises Deployments Only\) \(p. 309\)](#).

permissions

This section specifies how special permissions, if any, should be applied to the files in the **files** section as they are being copied over to the instance. This section applies to Amazon Linux, Ubuntu Server, and Red Hat Enterprise Linux (RHEL) instances only.

For more information see, [AppSpec 'permissions' Section \(EC2/On-Premises Deployments Only\) \(p. 312\)](#).

hooks

This section specifies scripts to run at specific deployment lifecycle events during the deployment.

For more information, see [AppSpec 'hooks' Section \(p. 316\)](#).

Topics

- [AppSpec 'files' Section \(EC2/On-Premises Deployments Only\) \(p. 309\)](#)
- [AppSpec 'resources' Section \(AWS Lambda Deployments Only\) \(p. 312\)](#)
- [AppSpec 'permissions' Section \(EC2/On-Premises Deployments Only\) \(p. 312\)](#)
- [AppSpec 'hooks' Section \(p. 316\)](#)

AppSpec 'files' Section (EC2/On-Premises Deployments Only)

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 are copying files from your revision to 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 files are copied to the instance.
- If **source** refers to a directory, then all files in the directory are copied to the instance.
- If **source** is a single slash ("/" for Amazon Linux, RHEL, and Ubuntu Server instances, or "\" for Windows Server instances), then all of the files from your revision are 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 where the files should be copied. This must be a fully qualified path.

source and **destination** are each specified with a string.

Here's an example **files** section for an Amazon Linux, Ubuntu Server, or RHEL instance.

```
files:
  - source: Config/config.txt
    destination: /webapps/Config
  - source: source
    destination: /webapps/myApp
```

In this example, the following two operations are performed during the **Install** event:

1. Copy the Config/config.txt file in your revision to the /webapps/Config/config.txt path on the instance.
2. Recursively copy all of the files in your revision's source directory to the /webapps/myApp directory on the instance.

'files' Section Examples

The following examples show 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, Ubuntu Server, and RHEL instances.

Note

Only EC2/On-Premises deployments use the **files** section. It does not apply to AWS Lambda deployments.

For the following examples, we assume these files appear in the bundle in the root of source:

- 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
```

For the following examples, we assume the appspec.yml appears in the bundle in the root of source along with a folder named my-folder that contains three files:

- 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 destination 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 destination
#   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
```

AppSpec 'resources' Section (AWS Lambda Deployments Only)

The 'resources' section specifies the Lambda function to deploy and has the following structure:

YAML:

```
resources:
  - name-of-function-to-deploy:
      type: "AWS::Lambda::Function"
      properties:
        name: name-of-lambda-function-to-deploy
        alias: alias-of-lambda-function-to-deploy
        currentversion: version-of-the-lambda-function-traffic-currently-points-to
        targetversion: version-of-the-lambda-function-to-shift-traffic-to
```

JSON:

```
"resources": [{
  "name-of-function-to-deploy" {
    "type": "AWS::Lambda::Function"
    "properties": {
      "name": "name-of-lambda-function-to-deploy"
      "alias": "alias-of-lambda-function-to-deploy"
      "currentversion": "version-of-the-lambda-function-traffic-currently-points-to"
      "targetversion": "version-of-the-lambda-function-to-shift-traffic-to"
    }
  }
}]
```

Each property is specified with a string.

- **name** – Required. This is the name of the Lambda function to deploy.
- **alias** – Required. This is the name of the alias to the Lambda function.
- **currentversion** – Required. This is the version of the Lambda function traffic currently points to.
- **targetversion** – Required. This is the version of the Lambda function traffic is shifted to.

AppSpec 'permissions' Section (EC2/On-Premises Deployments Only)

The 'permissions' section specifies how special permissions, if any, should be applied to the files and directories/folders in the 'files' section after they are copied to the instance. You can specify multiple

object instructions. This section is optional. It applies to Amazon Linux, Ubuntu Server, and RHEL instances only.

Note

The 'permissions' section is used for EC2/On-Premises deployments only. It is not used for AWS Lambda deployments.

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 are applied to after the file system objects are copied to the instance.

Specify **object** with a string.

- **pattern** – Optional. Specifies a pattern to apply permissions. If not specified or specified with the special characters "***", the permissions are applied to all matching files or directories, depending on the **type**.

Specify **pattern** with a string with quotation marks ("").

- **except** – Optional. Specifies any files or directories that are exceptions to **pattern**.

Specify **except** with a comma-separated list of strings inside square brackets.

- **owner** – Optional. The name of the owner of **object**. If not specified, all existing owners applied to the original file or directory/folder structure remain unchanged after the copy operation.

Specify **owner** with a string.

- **group** – Optional. The name of the group for **object**. If not specified, all existing groups applied to the original file or directory/folder structure remain unchanged after the copy operation.

Specify **group** with a string.

- **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. **4755** represents the setuid attribute is set, full control permissions for the owner, read and execute permissions for the group, and read and execute permissions for all other users. (For more examples, see the Linux **chmod** command documentation.) If **mode** is not specified, all existing modes applied to the original file or directory/folder structure remain unchanged after the copy operation.

Specify **mode** with a string.

- **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 more examples, see ACL entry format examples in the Linux **setfacl** command documentation.) You

can specify multiple ACL entries. If **acls** is not specified, any existing ACLs applied to the original file or directory/folder structure remain unchanged after the copy operation. These replace any existing ACLs.

Specify an **acls** with a dash (-), followed by a space, and then a string (for example, - u:jane:rw). If you have more than one ACL, each is specified on a separate line.

Note

Setting unnamed users, unnamed groups, or other similar ACL entries causes 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.) Each key is entered with a string. If not specified, any existing labels applied to the original file or directory/folder structure 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) are enabled on the machine. If not enabled, **range** defaults to **s0**.

Specify **context** with a string. For example, user: unconfined_u. Each **context** is specified on a separate line.

- **type** – Optional. The types of objects to which to apply the specified permissions. **type** is a string that can be set to **file** or **directory**. If **file** is specified, the permissions are applied only to files that are immediately contained in **object** after the copy operation (and not to **object** itself). If **directory** is specified, the permissions are recursively applied to all directories/folders that are anywhere in **object** after the copy operation (but not to **object** itself).

Specify **type** with a dash (-), followed by a space, and then a string (for example, - file).

'permissions' Section Example

The following example shows how to specify the 'permissions' section with the **object**, **pattern**, **except**, **owner**, **mode**, and **type** instructions. This example applies to Amazon Linux, Ubuntu Server, and RHEL instances only. In this example, assume the following files and folders are copied 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 permissions on these files and folders after they are copied:

```
version: 0.0
os: linux
# Copy over all of the folders and files with the 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 of 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/my-folder-3
  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 to Amazon Linux, Ubuntu Server, and RHEL instances only.

```
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
```

AppSpec 'hooks' Section

The content of the 'hooks' section of the AppSpec file varies, depending on the compute platform for your deployment. The 'hooks' section for an EC2/On-Premises deployment contains mappings that link deployment lifecycle event hooks to one or more scripts. The 'hooks' section for a Lambda deployment specifies Lambda validation functions to run during a deployment lifecycle event. If an event hook is not present, then no operation is executed for that event. This section is required only if you are running scripts or Lambda validation functions as part of the deployment.

Topics

- [AppSpec 'hooks' Section for an AWS Lambda Deployment \(p. 317\)](#)
- [AppSpec 'hooks' Section for an EC2/On-Premises Deployment \(p. 319\)](#)

AppSpec 'hooks' Section for an AWS Lambda Deployment

Topics

- [List of Lifecycle Event Hooks for an AWS Lambda Deployment \(p. 317\)](#)
- [Run Order of Hooks in a Lambda Function Version Deployment \(p. 317\)](#)
- [Structure of 'hooks' Section \(p. 317\)](#)
- [Sample Lambda 'hooks' Function \(p. 318\)](#)

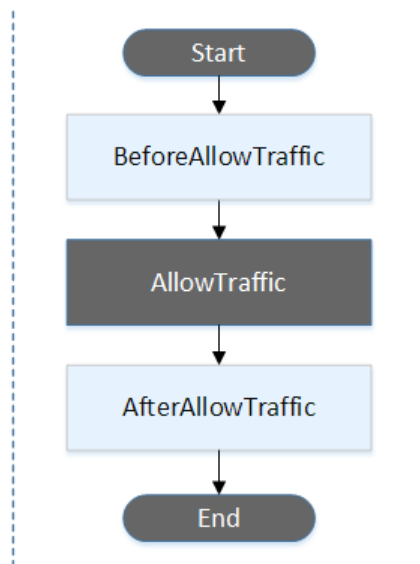
List of Lifecycle Event Hooks for an AWS Lambda Deployment

An AWS Lambda hook is one Lambda function specified with a string on a new line after the name of the lifecycle event. Each hook is executed once per deployment. Following are descriptions of the hooks that are available for use in your AppSpec file.

- **BeforeAllowTraffic** – Use to run tasks before traffic is shifted to the deployed Lambda function version.
- **AfterAllowTraffic** – Use to run tasks after all traffic is shifted to the deployed Lambda function version.

Run Order of Hooks in a Lambda Function Version Deployment

In a serverless Lambda function version deployment, event hooks run in the following order:



Note

The **Start**, **AllowTraffic**, and **End** events in the deployment cannot be scripted, which is why they appear in gray in this diagram.

Structure of 'hooks' Section

The following are examples of the structure of the 'hooks' section.

Using YAML:

```
hooks:
  - BeforeAllowTraffic: BeforeAllowTrafficHookFunctionName
```

```
- AfterAllowTraffic: AfterAllowTrafficHookFunctionName
```

Using JSON:

```
"hooks": [{
  "BeforeAllowTraffic": "BeforeAllowTrafficHookFunctionName"
},
{
  "AfterAllowTraffic": "AfterAllowTrafficHookFunctionName"
}]
```

Sample Lambda 'hooks' Function

Use the 'hooks' section to specify a Lambda function that AWS CodeDeploy can call to validate a Lambda deployment. You can use the same function or a different one for the **BeforeAllowTraffic** and **AfterAllowTraffic** deployment lifecycle events. Following completion of the validation tests, the Lambda validation function calls back AWS CodeDeploy and delivers a result of 'Succeeded' or 'Failed'.

Important

If AWS CodeDeploy is not notified by the Lambda validation function within one hour, then it assumes the deployment failed.

Before invoking a Lambda hook function, the server must be notified of the deployment ID and the lifecycle event hook execution ID:

```
aws deploy put-lifecycle-event-hook-execution-status --deployment-id <deployment-id> --
status Succeeded --lifecycle-event-hook-execution-id <execution-id> --region <region>
```

The following is a sample Lambda hook function written in Node.js.

```
'use strict';

const aws = require('aws-sdk');
const codedeploy = new aws.CodeDeploy({apiVersion: '2014-10-06'});

exports.handler = (event, context, callback) => {
  //Read the DeploymentId from the event payload.
  var deploymentId = event.DeploymentId;

  //Read the LifecycleEventHookExecutionId from the event payload
  var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;

  /*
   Enter validation tests here.
  */

  // Prepare the validation test results with the deploymentId and
  // the lifecycleEventHookExecutionId for AWS CodeDeploy.
  var params = {
    deploymentId: deploymentId,
    lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
    status: 'Succeeded' // status can be 'Succeeded' or 'Failed'
  };

  // Pass AWS CodeDeploy the prepared validation test results.
  codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data) {
    if (err) {
      // Validation failed.
      callback('Validation test failed');
    } else {
      // Validation succeeded.
      callback(null, 'Validation test succeeded');
    }
  });
}
```

```
}  
});  
};
```

AppSpec 'hooks' Section for an EC2/On-Premises Deployment

Topics

- [List of Lifecycle Event Hooks \(p. 319\)](#)
- [Lifecycle Event Hook Availability \(p. 320\)](#)
- [Run Order of Hooks in a Deployment \(p. 321\)](#)
- [Structure of 'hooks' Section \(p. 323\)](#)
- [Environment Variable Availability for Hooks \(p. 324\)](#)
- [Hooks Example \(p. 325\)](#)

List of Lifecycle Event Hooks

An EC2/On-Premises deployment hook is executed once per deployment to an instance. You can specify one or more scripts to run in a hook. Each hook for a lifecycle event is specified with a string on a separate line. Following are descriptions of the hooks that are available for use in your AppSpec file.

For information about which lifecycle event hooks are valid for which deployment and rollback types, see [Lifecycle Event Hook Availability \(p. 320\)](#).

- **ApplicationStop** – This deployment lifecycle event occurs even before the application revision is downloaded. You can specify scripts for this event 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 previous successfully deployed application revision.

Note

An AppSpec file does not exist on an instance before you deploy to it. For this reason, the **ApplicationStop** hook does not run the first time you deploy to the instance. You can use the **ApplicationStop** hook the second time you deploy to an instance.

To determine the location of the last successfully deployed application revision, the AWS CodeDeploy agent looks up the location listed in the `deployment-group-id_last_successful_install` file. This file is located in:

`/opt/codedeploy-agent/deployment-root/deployment-instructions` folder on Amazon Linux, Ubuntu Server, and RHEL Amazon EC2 instances.

`C:\ProgramData\Amazon\CodeDeploy\deployment-instructions` folder on Windows Server Amazon EC2 instances.

To troubleshoot a deployment that fails during the **ApplicationStop** deployment lifecycle event, see [Troubleshooting failed ApplicationStop, BeforeBlockTraffic, and AfterBlockTraffic deployment lifecycle events \(p. 350\)](#).

- **DownloadBundle** – During this deployment lifecycle event, the AWS CodeDeploy agent copies the application revision files to a temporary location:

`/opt/codedeploy-agent/deployment-root/deployment-group-id/deployment-id/deployment-archive` folder on Amazon Linux, Ubuntu Server, and RHEL Amazon EC2 instances.

`C:\ProgramData\Amazon\CodeDeploy\deployment-group-id\deployment-id\deployment-archive` folder on Windows Server Amazon EC2 instances.

This 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. 351).

- **BeforeInstall** – You can use this deployment lifecycle event for preinstall tasks, such as decrypting files and creating a backup of the current version.
- **Install** – During this deployment lifecycle event, the AWS CodeDeploy agent copies the revision files from the temporary location to the final destination folder. This event is reserved for the AWS CodeDeploy agent and cannot be used to run scripts.
- **AfterInstall** – You can use this deployment lifecycle event for tasks such as configuring your application or changing file permissions.
- **ApplicationStart** – You typically use this deployment lifecycle event to restart services that were stopped during **ApplicationStop**.
- **ValidateService** – This is the last deployment lifecycle event. It is used to verify the deployment was completed successfully.
- **BeforeBlockTraffic** – You can use this deployment lifecycle event to run tasks on instances before they are deregistered from a load balancer.

To troubleshoot a deployment that fails during the **BeforeBlockTraffic** deployment lifecycle event, see [Troubleshooting failed ApplicationStop, BeforeBlockTraffic, and AfterBlockTraffic deployment lifecycle events](#) (p. 350).

- **BlockTraffic** – During this deployment lifecycle event, internet traffic is blocked from accessing instances that are currently serving traffic. This event is reserved for the AWS CodeDeploy agent and cannot be used to run scripts.
- **AfterBlockTraffic** – You can use this deployment lifecycle event to run tasks on instances after they are deregistered from a load balancer.

To troubleshoot a deployment that fails during the **AfterBlockTraffic** deployment lifecycle event, see [Troubleshooting failed ApplicationStop, BeforeBlockTraffic, and AfterBlockTraffic deployment lifecycle events](#) (p. 350).

- **BeforeAllowTraffic** – You can use this deployment lifecycle event to run tasks on instances before they are registered with a load balancer.
- **AllowTraffic** – During this deployment lifecycle event, internet traffic is allowed to access instances after a deployment. This event is reserved for the AWS CodeDeploy agent and cannot be used to run scripts.
- **AfterAllowTraffic** – You can use this deployment lifecycle event to run tasks on instances after they are registered with a load balancer.

Lifecycle Event Hook Availability

The following table lists the lifecycle event hooks available for each deployment and rollback scenario.

Lifecycle event name	In-place deployment ¹	Blue/green deployment: Original instances	Blue/green deployment: Replacement instances	Blue/green deployment rollback: Original instances	Blue/green deployment rollback: Replacement instances
ApplicationStop	✓		✓		
DownloadBundle ²	✓		✓		
BeforeInstall	✓		✓		

Lifecycle event name	In-place deployment ¹	Blue/green deployment: Original instances	Blue/green deployment: Replacement instances	Blue/green deployment rollback: Original instances	Blue/green deployment rollback: Replacement instances
Install²	✓		✓		
AfterInstall	✓		✓		
ApplicationStart	✓		✓		
ValidateService	✓		✓		
BeforeBlockTraffic	✓	✓			✓
BlockTraffic²	✓	✓			✓
AfterBlockTraffic	✓	✓			✓
BeforeAllowTraffic	✓		✓	✓	
AllowTraffic²	✓		✓	✓	
AfterAllowTraffic	✓		✓	✓	
¹ Also applies to the rollback of an in-place deployment.					
² Reserved for AWS CodeDeploy operations. Cannot be used to run scripts.					

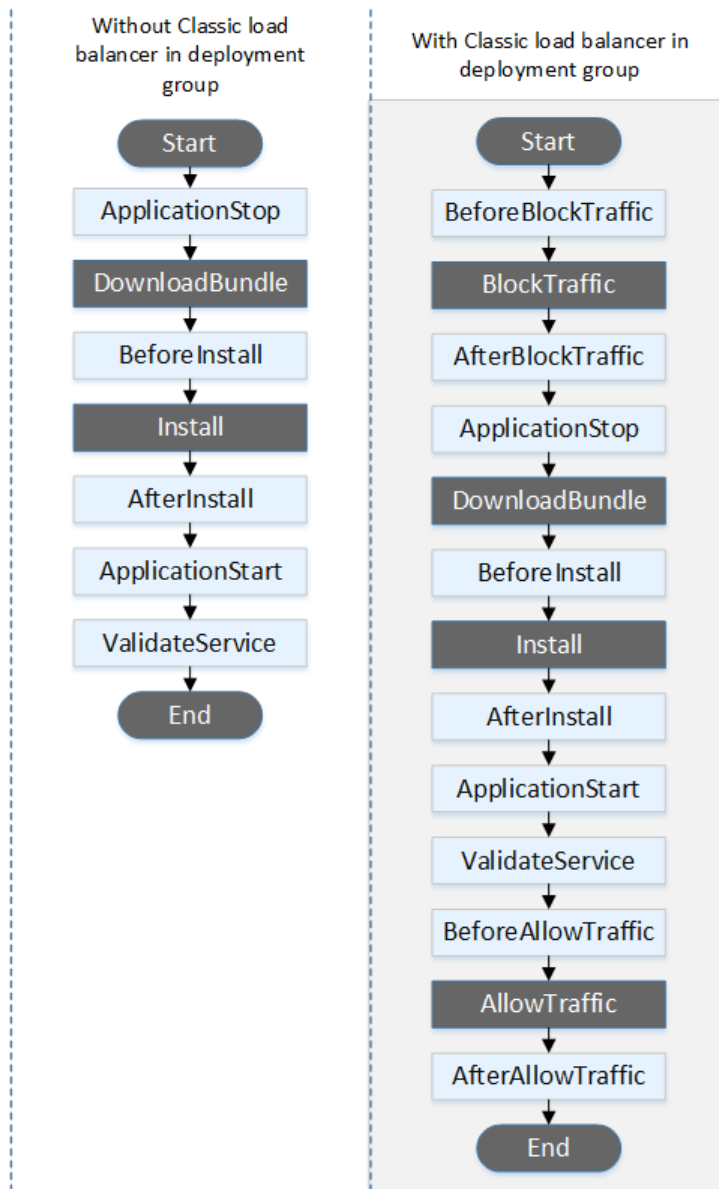
Run Order of Hooks in a Deployment

In-place deployments

In an in-place deployment, including the rollback of an in-place deployment, event hooks are run in the following order:

Note

For in-place deployments, the six hooks related to blocking and allowing traffic apply only if you specify a Classic Load Balancer, Application Load Balancer, or Network Load Balancer from Elastic Load Balancing in the deployment group.

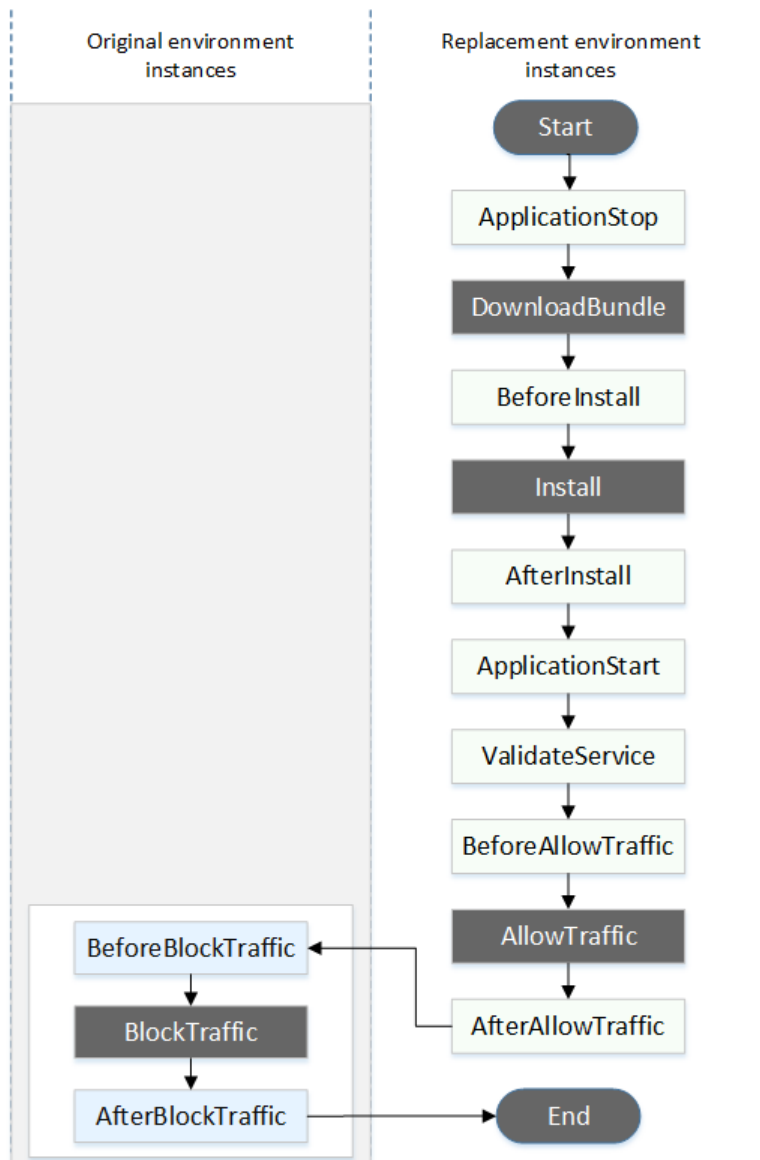


Note

The **Start**, **DownloadBundle**, **Install**, and **End** events in the deployment cannot be scripted, which is why they appear in gray in this diagram. However, you can edit the 'files' section of the AppSpec file to specify what's installed during the **Install** event.

Blue/green deployments

In a blue/green deployment, event hooks are run in the following order:



Note

The **Start**, **DownloadBundle**, **Install**, **BlockTraffic**, **AllowTraffic**, and **End** events in the deployment cannot be scripted, which is why they appear in gray in this diagram. However, you can edit the 'files' section of the AppSpec file to specify what's installed during the **Install** event.

Structure of 'hooks' Section

The 'hooks' 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 deployment lifecycle event name:

location

Required. The location in the bundle of the script file for the revision.

timeout

Optional. The number of seconds to allow the script to execute before it is considered to have failed. The default is 3600 seconds (1 hour).

Note

3600 seconds (1 hour) is the maximum amount of time allowed for script execution for each deployment lifecycle event. If scripts exceed this limit, the deployment stops and the deployment to the instance fails. Make sure the total number of seconds specified in **timeout** for all scripts in each deployment lifecycle event does not exceed this limit.

runas

Optional. The user to impersonate when running the script. By default, this is the AWS CodeDeploy agent running on the instance. AWS CodeDeploy does not store passwords, so the user cannot be impersonated if the **runas** user needs a password. This element applies to Amazon Linux and Ubuntu Server instances only.

Environment Variable Availability for Hooks

During each deployment lifecycle event, hook scripts can access the following environment variables:

APPLICATION_NAME

The name of the application in AWS CodeDeploy that is part of the current deployment (for example, `WordPress_App`).

DEPLOYMENT_ID

The ID AWS CodeDeploy has assigned to the current deployment (for example, `d-AB1CDEF23`).

DEPLOYMENT_GROUP_NAME

The name of the deployment group in AWS CodeDeploy that is part of the current deployment (for example, `WordPress_DepGroup`).

DEPLOYMENT_GROUP_ID

The ID of the deployment group in AWS CodeDeploy that is part of the current deployment (for example, `b1a2189b-dd90-4ef5-8f40-4c1c5EXAMPLE`).

LIFECYCLE_EVENT

The name of the current deployment lifecycle event (for example, `AfterInstall`).

These environment variables are local to each deployment lifecycle event.

The following script changes the listening port on an Apache HTTP server to 9090 instead of 80 if the value of **DEPLOYMENT_GROUP_NAME** is equal to `Staging`. This script must be invoked during the **BeforeInstall** deployment lifecycle event:

```
if [ "$DEPLOYMENT_GROUP_NAME" == "Staging" ]
then
    sed -i -e 's/Listen 80/Listen 9090/g' /etc/httpd/conf/httpd.conf
fi
```

The following script example changes the verbosity level of messages recorded in its error log from warning to debug if the value of the **DEPLOYMENT_GROUP_NAME** environment variable is equal to `Staging`. This script must be invoked during the **BeforeInstall** deployment lifecycle event:

```
if [ "$DEPLOYMENT_GROUP_NAME" == "Staging" ]
then
    sed -i -e 's/LogLevel warn/LogLevel debug/g' /etc/httpd/conf/httpd.conf
fi
```

The following script example replaces the text in the specified web page with text that displays the value of these environment variables. This script must be invoked during the **AfterInstall** deployment lifecycle event:

```
#!/usr/bin/python

import os

strToSearch=<h2>This application was deployed using AWS CodeDeploy.</h2>
strToReplace=<h2>This page for "+os.environ['APPLICATION_NAME']+"
    application and "+os.environ['DEPLOYMENT_GROUP_NAME']+" deployment group with
    "+os.environ['DEPLOYMENT_GROUP_ID']+" deployment group ID was generated by a
    "+os.environ['LIFECYCLE_EVENT']+" script during "+os.environ['DEPLOYMENT_ID']+"
    deployment.</h2>

fp=open("/var/www/html/index.html","r")
buffer=fp.read()
fp.close()

fp=open("/var/www/html/index.html","w")
fp.write(buffer.replace(strToSearch,strToReplace))
fp.close()
```

Hooks Example

Here is an example of a **hooks** entry that specifies two hooks for the **AfterInstall** lifecycle event:

```
hooks:
  AfterInstall:
    - location: Scripts/RunResourceTests.sh
      timeout: 180
    - location: Scripts/PostDeploy.sh
      timeout: 180
```

The `Scripts/RunResourceTests.sh` script runs during the **AfterInstall** stage of the deployment process. The deployment is unsuccessful if it takes the script more than 180 seconds (3 minutes) to run.

The location of scripts you specify in the 'hooks' section is relative to the root of the application revision bundle. In the preceding example, a file named `RunResourceTests.sh` is in a directory named `Scripts`. The `Scripts` directory is at the root level of the bundle. For more information, see [Plan a Revision for AWS CodeDeploy \(p. 232\)](#).

AppSpec File Example

This topic provides example AppSpec files for an AWS Lambda and an EC2/On-Premises deployment.

Topics

- [AppSpec File Example for an AWS Lambda Deployment \(p. 326\)](#)
- [AppSpec File Example for an EC2/On-Premises Deployment \(p. 327\)](#)

AppSpec File Example for an AWS Lambda Deployment

Here is an example of an AppSpec file written in YAML for deploying a Lambda function version.

```
version: 0.0
Resources:
  - myLambdaFunction:
      Type: AWS::Lambda::Function
      Properties:
        Name: "myLambdaFunction"
        Alias: "myLambdaFunctionAlias"
        CurrentVersion: "1"
        TargetVersion: "2"
Hooks:
  - BeforeAllowTraffic: "LambdaFunctionToValidateBeforeTrafficShift"
  - AfterAllowTraffic: "LambdaFunctionToValidateAfterTrafficShift"
```

Here is a version of the preceding example written in JSON.

```
{
  "version": 0.0,
  "Resources": [{
    "myLambdaFunction": {
      "Type": "AWS::Lambda::Function",
      "Properties": {
        "Name": "myLambdaFunction",
        "Alias": "myLambdaFunctionAlias",
        "CurrentVersion": "1",
        "TargetVersion": "2"
      }
    }
  ]},
  "Hooks": [{
    "BeforeAllowTraffic": "LambdaFunctionToValidateBeforeTrafficShift"
  },
  {
    "AfterAllowTraffic": "LambdaFunctionToValidateAfterTrafficShift"
  }
]
}
```

Here is the sequence of events during deployment:

1. Before shifting traffic from version 1 of a Lambda function called `myLambdaFunction` to version 2, run a Lambda function called `LambdaFunctionToValidateBeforeTrafficShift` that validates the deployment is ready to start traffic shifting.
2. If `LambdaFunctionToValidateBeforeTrafficShift` returned an exit code of 0 (success), begin shifting traffic to version 2 of `myLambdaFunction`. The deployment configuration for this deployment determines the rate at which traffic is shifted.
3. After the shifting of traffic from version 1 of a Lambda function called `myLambdaFunction` to version 2 is complete, run a Lambda function called `LambdaFunctionToValidateAfterTrafficShift` that validates the deployment was completed successfully.

AppSpec File Example for an EC2/On-Premises Deployment

Here is an example of an AppSpec file for an in-place deployment to an Amazon Linux, Ubuntu Server, or RHEL instance.

Note

Deployments to Windows Server instances do not support the `runas` element. If you are deploying to Windows Server instances, do not include it in your AppSpec file.

```
version: 0.0
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
```

For a Windows Server instance, change `os: linux` to `os: windows`. Also, you must fully qualify the destination paths (for example, `c:\temp\webapps\Config` and `c:\temp\webapps\myApp`). Do not include the `runas` element.

Here is the sequence of events during deployment:

1. Run the script located at `Scripts/UnzipResourceBundle.sh`.
2. If the previous script returned an exit code of 0 (success), run the script located at `Scripts/UnzipDataBundle.sh`.
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 `codedeployuser` with a timeout of 3600 seconds (1 hour).

AppSpec File Spacing

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. AWS CodeDeploy will raise an error that might be difficult to debug if the locations and number of spaces in an AppSpec file are not correct.

```
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 correctly spaced 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
    - location: scripts/create_test_db.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

For more information about spacing, see the [YAML](#) specification.

Validate Your AppSpec File and File Location

File syntax

AWS CodeDeploy does not provide a tool for validating the contents of an AppSpec File. Instead, consider using a browser-based tool such as [YAML Lint](#) or [Online YAML Parser](#) to help you check your YAML syntax.

File location

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:

On local Linux, macOS, or Unix instances:

```
ls path/to/root/directory/appspec.yml
```

If the AppSpec file is not located there, a "No such file or directory" error is displayed.

On local Windows instances:

```
dir path\to\root\directory\appspec.yml
```

If the AppSpec file is not located there, a "File Not Found" error is displayed.

AWS CodeDeploy Agent Configuration Reference

When the AWS CodeDeploy agent is installed, a configuration file is placed on the instance. This configuration file specifies directory paths and other settings for AWS CodeDeploy to use as it interacts with the instance. You can change some of the configuration options in the file.

For Amazon Linux, Ubuntu Server, and Red Hat Enterprise Linux (RHEL) instances, the configuration file is named `codedeployagent.yml`. It is placed in the `/etc/codedeploy-agent/conf` directory.

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

The configuration settings include:

:log_aws_wire:	<p>Set to <code>true</code> for the AWS CodeDeploy agent to capture wire logs from Amazon S3 and write them to a file named <code>codedeploy-agent.wire.log</code> in the location pointed to by the :log_dir: setting.</p> <p>Warning</p> <p>You should set :log_aws_wire: to <code>true</code> only for the amount of time required to capture wire logs. The <code>codedeploy-agent.wire.log</code> file can grow to a very large size quickly. The wire log output in this file might contain sensitive information, including the plain-text contents of files transferred into, or out of, Amazon S3 while this setting was set to <code>true</code>. The wire logs contain information about all Amazon S3 activity associated with the AWS account while this setting was set to <code>true</code>, not just activity related to AWS CodeDeploy deployments.</p> <p>The default setting is <code>false</code>.</p> <p>This setting applies to all instance types. You must add this configuration setting to Windows Server instances to be able to use it.</p>
:log_dir:	<p>The folder on the instance where log files related to AWS CodeDeploy agent operations are stored.</p> <p>The default setting is <code>' /var/log/aws/codedeploy-agent '</code> for Amazon Linux, Ubuntu Server, and RHEL instances and <code>C:\ProgramData\Amazon\CodeDeploy\log</code> for Windows Server instances.</p>
:pid_dir:	<p>The folder where <code>codedeploy-agent.pid</code> is stored.</p>

	<p>This file contains the process ID (PID) of the AWS CodeDeploy agent. The default setting is <code>'/opt/codedeploy-agent/state/.pid'</code>.</p> <p>This setting applies to Amazon Linux, Ubuntu Server, and RHEL instances only.</p>
:program_name:	<p>The AWS CodeDeploy agent program name.</p> <p>The default setting is <code>codedeploy-agent</code>.</p> <p>This setting applies to Amazon Linux, Ubuntu Server, and RHEL instances only.</p>
:root_dir:	<p>The folder where related revisions, deployment history, and deployment scripts on the instance are stored.</p> <p>The default setting is <code>'/opt/codedeploy-agent/deployment-root'</code> for Amazon Linux, Ubuntu Server, and RHEL instances and <code>C:\ProgramData\Amazon\CodeDeploy</code> for Windows Server instances.</p>
:verbose:	<p>Set to <code>true</code> for the AWS CodeDeploy agent to print debug messages log files on the instance.</p> <p>The default setting is <code>false</code> for Amazon Linux, Ubuntu Server, and RHEL instances and <code>true</code> for Windows Server instances.</p>
:wait_between_runs:	<p>The interval, in seconds, between AWS CodeDeploy agent polling of AWS CodeDeploy for pending deployments.</p> <p>The default setting is 1.</p>
:on_premises_config_file:	<p>For on-premises instances, the path to an alternate location for the configuration file named <code>codedeploy.onpremises.yml</code> (for Ubuntu Server and RHEL) or <code>conf.onpremises.yml</code> (for Windows Server).</p> <p>By default, these files are stored in <code>/etc/codedeploy-agent/conf/codedeploy.onpremises.yml</code> for Ubuntu Server and RHEL and <code>C:\ProgramData\Amazon\CodeDeploy\conf.onpremises.yml</code> for Windows Server.</p> <p>Available in version 1.0.1.686 and later versions of the AWS CodeDeploy agent.</p>

:proxy_uri:	<p>(Optional) The HTTP proxy through which you want the AWS CodeDeploy agent to connect to AWS for your AWS CodeDeploy operations. Use a format similar to <code>https://user:password@my.proxy:443/path?query</code>.</p> <p>Available in version 1.0.1.824 and later versions of the AWS CodeDeploy agent.</p>
:max_revisions:	<p>(Optional) The number of application revisions for a deployment group that you want the AWS CodeDeploy agent to archive. Any revisions that exceed the number specified are deleted.</p> <p>Enter any positive integer. If no value is specified, AWS CodeDeploy will retain the five most recent revisions in addition to the currently deployed revision.</p> <p>Supported in version 1.0.1.966 and later versions of the AWS CodeDeploy agent.</p>

Related Topics

[Working with the AWS CodeDeploy Agent \(p. 125\)](#)

[Managing AWS CodeDeploy Agent Operations \(p. 132\)](#)

AWS CloudFormation Templates for AWS CodeDeploy Reference

In addition to the other methods available to you in AWS CodeDeploy, you can use AWS CloudFormation templates to perform the following tasks:

- Create applications.
- Create deployment groups and specify a target revision.
- Create deployment configurations.
- Create Amazon EC2 instances.

AWS CloudFormation is a service that helps you model and set up your AWS resources using templates. An AWS CloudFormation template is a text file whose format complies with the JSON standard. You create a template that describes all of the AWS resources you want, and AWS CloudFormation takes care of provisioning and configuring those resources for you.

For more information, see [What Is AWS CloudFormation?](#) and [Working with AWS CloudFormation Templates](#) in *AWS CloudFormation User Guide*.

If you plan to use AWS CloudFormation templates that are compatible with AWS CodeDeploy in your organization, as an administrator, you must grant access to AWS CloudFormation and to the AWS services and actions on which AWS CloudFormation depends. To grant permissions to create applications, deployment groups, and deployment configurations, attach the following policy to the IAM users who will work with AWS CloudFormation:

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

For more information about managed policies, see the following topics:

- To view the policy that must be attached to IAM users who will create Amazon EC2 instances, see [Create an Amazon EC2 Instance for AWS CodeDeploy \(AWS CloudFormation Template\)](#) (p. 162).
- For information about attaching policies to IAM users, see [Working with Managed Policies](#) in *IAM User Guide*.
- To learn how to restrict users to a limited set of AWS CodeDeploy actions and resources, see [AWS Managed \(Predefined\) Policies for AWS CodeDeploy](#) (p. 296).

The following table shows the actions an AWS CloudFormation template can perform on your behalf and includes links to more information about the AWS resource types and their property types you can add to an AWS CloudFormation template.

Action	AWS CloudFormation Resource Type
Create an AWS CodeDeploy application.	AWS::CodeDeploy::Application
Create and specify the details for a deployment group to be used to deploy your application revisions. ¹	AWS::CodeDeploy::DeploymentGroup
Create a set of deployment rules, deployment success conditions, and deployment failure conditions that AWS CodeDeploy will use during a deployment.	AWS::CodeDeploy::DeploymentConfig
Create an Amazon EC2 instance. ²	AWS::EC2::Instance
<p>¹ If you specify the version of the application revision that you want to be deployed as part of the deployment group, your target revision will be deployed as soon as the provisioning process is complete. For more information about template configuration, see AWS CodeDeploy DeploymentGroup Deployment Revision S3Location and AWS CodeDeploy DeploymentGroup Deployment Revision GitHubLocation in the <i>AWS CloudFormation User Guide</i>.</p> <p>² We provide templates you can use to create Amazon EC2 instances in the regions in which AWS CodeDeploy is supported. For more information about using these templates, see Create an Amazon EC2 Instance for AWS CodeDeploy (AWS CloudFormation Template) (p. 162).</p>	

AWS CodeDeploy Resource Kit Reference

Many of the files AWS CodeDeploy relies on are stored in publicly available, AWS region-specific Amazon S3 buckets. These files include installation files for the AWS CodeDeploy agent, templates, and sample application files. We call this collection of files the AWS CodeDeploy Resource Kit.

Topics

- [Resource Kit Bucket Names by Region \(p. 335\)](#)
- [Resource Kit Contents \(p. 336\)](#)
- [Display a List of the Resource Kit Files \(p. 337\)](#)
- [Download the Resource Kit Files \(p. 338\)](#)

Resource Kit Bucket Names by Region

This table lists the names of *bucket-name* replacements required for some procedures in the guide. These are the names of the Amazon S3 buckets that contain the AWS CodeDeploy Resource Kit files.

Region name	<i>bucket-name</i> replacement	Region identifier
US East (Ohio)	aws-codedeploy-us-east-2	us-east-2
US East (N. Virginia)	aws-codedeploy-us-east-1	us-east-1
US West (N. California)	aws-codedeploy-us-west-1	us-west-1
US West (Oregon)	aws-codedeploy-us-west-2	us-west-2
Canada (Central)	aws-codedeploy-ca-central-1	ca-central-1
EU (Ireland)	aws-codedeploy-eu-west-1	eu-west-1
EU (London)	aws-codedeploy-eu-west-2	eu-west-2
EU (Paris)	aws-codedeploy-eu-west-3	eu-west-3
EU (Frankfurt)	aws-codedeploy-eu-central-1	eu-central-1
Asia Pacific (Tokyo)	aws-codedeploy-ap-northeast-1	ap-northeast-1
Asia Pacific (Seoul)	aws-codedeploy-ap-northeast-2	ap-northeast-2
Asia Pacific (Singapore)	aws-codedeploy-ap-southeast-1	ap-southeast-1
Asia Pacific (Sydney)	aws-codedeploy-ap-southeast-2	ap-southeast-2
Asia Pacific (Mumbai)	aws-codedeploy-ap-south-1	ap-south-1

Region name	<i>bucket-name</i> replacement	Region identifier
South America (São Paulo)	aws-codedeploy-sa-east-1	sa-east-1

Resource Kit Contents

The following table lists the files in the AWS CodeDeploy Resource Kit.

File	Description
VERSION	A file used by AWS CodeDeploy agents to update themselves as they are running on instances.
codedeploy-agent.noarch.rpm	The AWS CodeDeploy agent for Amazon Linux and Red Hat Enterprise Linux (RHEL). There may be several files with the same base file name, but different versions (such as <code>-1.0-0</code>).
codedeploy-agent_all.deb	The AWS CodeDeploy agent for Ubuntu Server. There may be several files with the same base file name, but different versions (such as <code>_1.0-0</code>).
codedeploy-agent.msi	The AWS CodeDeploy agent for Windows Server. There may be several files with the same base file name, but different versions (such as <code>-1.0-0</code>).
install	A file you can use to more easily install the AWS CodeDeploy agent.
CodeDeploy_SampleCF_Template.json	An AWS CloudFormation template you can use to launch from one to three Amazon EC2 instances running Amazon Linux or Windows Server. There may be several files with the same base file name, but different versions (such as <code>-1.0.0</code>).
CodeDeploy_SampleCF_ELB_Integration.json	An AWS CloudFormation template you can use to create a load-balanced sample Web site running on an Apache Web Server. The application is configured to span all Availability Zones in the region you create it in. This template creates three Amazon EC2 instances and IAM instance profile to grant the instances access to the resources in Amazon S3, Auto Scaling, AWS CloudFormation, and Elastic Load Balancing. It also creates the load balancer and an AWS CodeDeploy service role.
SampleApp_ELB_Integration.zip	A sample application revision you can deploy to an Amazon EC2 instance that is registered to an Elastic Load Balancing load balancer.
SampleApp_Linux.zip	A sample application revision you can deploy to an Amazon EC2 instance running Amazon Linux or to a Ubuntu Server or RHEL instance. There may be several files with the same base file name, but different versions (such as <code>-1.0</code>).

File	Description
SampleApp2_Linux.zip	A sample application revision that is deployed to a replacement fleet of instances when you run theSample deployment wizard.
SampleApp_Windows.zip	A sample application revision you can deploy to a Windows Server instance. There may be several files with the same base file name, but different versions (such as -1.0).

Display a List of the Resource Kit Files

To view a list of files, use the **aws s3 ls** command 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-2
```
- ```
aws s3 ls --recursive s3://aws-codedeploy-us-east-1
```
- ```
aws s3 ls --recursive s3://aws-codedeploy-us-west-1
```
- ```
aws s3 ls --recursive s3://aws-codedeploy-us-west-2
```
- ```
aws s3 ls --recursive s3://aws-codedeploy-ca-central-1
```
- ```
aws s3 ls --recursive s3://aws-codedeploy-eu-west-1
```
- ```
aws s3 ls --recursive s3://aws-codedeploy-eu-west-2
```
- ```
aws s3 ls --recursive s3://aws-codedeploy-eu-west-3
```
- ```
aws s3 ls --recursive s3://aws-codedeploy-eu-central-1
```
- ```
aws s3 ls --recursive s3://aws-codedeploy-ap-northeast-1
```
- ```
aws s3 ls --recursive s3://aws-codedeploy-ap-northeast-2
```
- ```
aws s3 ls --recursive s3://aws-codedeploy-ap-southeast-1
```
- ```
aws s3 ls --recursive s3://aws-codedeploy-ap-southeast-2
```
- ```
aws s3 ls --recursive s3://aws-codedeploy-ap-south-1
```
- ```
aws s3 ls --recursive s3://aws-codedeploy-sa-east-1
```

## Download the Resource Kit Files

To download a file, use the **aws s3 cp** command for your region.

### Note

Be sure to use the period (.) near the end. This downloads the file to your current directory.

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-2/samples/latest/SampleApp_Linux.zip . --region us-east-2
```
- ```
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-1/samples/latest/SampleApp_Linux.zip . --region us-west-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-ca-central-1/samples/latest/SampleApp_Linux.zip . --region ca-central-1
```
- ```
aws s3 cp s3://aws-codedeploy-eu-west-1/samples/latest/SampleApp_Linux.zip . --region eu-west-1
```
- ```
aws s3 cp s3://aws-codedeploy-eu-west-2/samples/latest/SampleApp_Linux.zip . --region eu-west-2
```
- ```
aws s3 cp s3://aws-codedeploy-eu-west-3/samples/latest/SampleApp_Linux.zip . --region eu-west-3
```
- ```
aws s3 cp s3://aws-codedeploy-eu-central-1/samples/latest/SampleApp_Linux.zip . --region eu-central-1
```
- ```
aws s3 cp s3://aws-codedeploy-ap-northeast-1/samples/latest/SampleApp_Linux.zip . --region ap-northeast-1
```
- ```
aws s3 cp s3://aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Linux.zip . --region ap-northeast-2
```
- ```
aws s3 cp s3://aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Linux.zip . --region ap-southeast-1
```
- ```
aws s3 cp s3://aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Linux.zip . --region ap-southeast-2
```
- ```
aws s3 cp s3://aws-codedeploy-ap-south-1/samples/latest/SampleApp_Linux.zip . --region ap-south-1
```
- ```
aws s3 cp s3://aws-codedeploy-sa-east-1/samples/latest/SampleApp_Linux.zip . --region sa-east-1
```

To download all of the files, use one of the following commands for your region:

- `aws s3 cp --recursive s3://aws-codedeploy-us-east-2 . --region us-east-2`
- `aws s3 cp --recursive s3://aws-codedeploy-us-east-1 . --region us-east-1`
- `aws s3 cp --recursive s3://aws-codedeploy-us-west-1 . --region us-west-1`
- `aws s3 cp --recursive s3://aws-codedeploy-us-west-2 . --region us-west-2`
- `aws s3 cp --recursive s3://aws-codedeploy-ca-central-1 . --region ca-central-1`
- `aws s3 cp --recursive s3://aws-codedeploy-eu-west-1 . --region eu-west-1`
- `aws s3 cp --recursive s3://aws-codedeploy-eu-west-2 . --region eu-west-2`
- `aws s3 cp --recursive s3://aws-codedeploy-eu-west-3 . --region eu-west-3`
- `aws s3 cp --recursive s3://aws-codedeploy-eu-central-1 . --region eu-central-1`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-northeast-1 . --region ap-northeast-1`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-northeast-2 . --region ap-northeast-2`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-southeast-1 . --region ap-southeast-1`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-southeast-2 . --region ap-southeast-2`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-south-1 . --region ap-south-1`
- `aws s3 cp --recursive s3://aws-codedeploy-sa-east-1 . --region sa-east-1`

AWS CodeDeploy Limits

The following tables describe limits in AWS CodeDeploy.

Note

You can [request a limit increase](#) for the AWS CodeDeploy limits listed in [AWS Service Limits](#) in the *Amazon Web Services General Reference*. You cannot increase the limit on the number of hours a deployment can run.

Topics

- [Applications](#) (p. 340)
- [Application Revisions](#) (p. 340)
- [Deployments](#) (p. 341)
- [Deployment Configurations](#) (p. 342)
- [Deployment Groups](#) (p. 342)
- [Instances](#) (p. 343)

Applications

Maximum number of applications associated with an AWS account in a single region	100
Maximum number of characters in an application name	100
Characters allowed in an application name	Letters (a-z, A-Z), numbers (0-9), periods (.), underscores (_), + (plus signs), = (equals signs), , (commas), @ (at signs), - (minus signs).
Maximum number of applications that can be passed to the BatchGetApplications API action	100
Maximum number of GitHub connection tokens for a single AWS account	25

Application Revisions

Maximum number of characters in an application revision name	100
Allowed file types for application revisions	<p>Archive files with the extension <code>.zip</code> or <code>.tar</code> and compressed archive files with the extension <code>.tar.gz</code>.</p> <p>An archive or compressed archive file that is compatible with AWS CodeDeploy must contain a</p>

single application specification file (AppSpec file) with the file name `appspec.yml`.

Deployments

Maximum number of concurrent deployments to a deployment group ¹	1
Maximum number of concurrent deployments associated with an AWS account ²	100
Maximum number of hours an EC2/On-Premises in-place deployment can run	8
Maximum number of hours between the deployment of a revision and the shifting of traffic to the replacement environment during an EC2/On-Premises blue/green deployment	48
Maximum number of hours between the completion of a deployment and the termination of the original environment during an EC2/On-Premises blue/green deployment	48
Maximum number of hours an EC2/On-Premises blue/green deployment can run	109 (48 for each of the above two limits) plus one hour for each of 13 possible lifecycle events
Maximum number of hours an AWS Lambda deployment can run ³	50 (48 hours for the maximum time between the first and last traffic shift plus one hour for each of two possible lifecycle hooks)
Maximum number of seconds until a deployment lifecycle event fails if not completed	3600
Maximum number of characters in a deployment description	256
Maximum number of deployments that can be passed to the BatchGetDeployments API action	100
Maximum number of minutes until a deployment fails if a lifecycle event doesn't start after: <ul style="list-style-type: none"> A deployment is triggered by using the console or the AWS CLI <code>create-deployment</code> command. The previous lifecycle event is completed. 	5
Maximum number of minutes a blue/green deployment can wait after a successful deployment before terminating instances from the original deployment	2800

¹ This limit is intended to prevent accidental, concurrent deployments of the same application to the same deployment group.

² Each deployment to a scaled-up Amazon EC2 instance in an Auto Scaling group counts as a single concurrent deployment. If the scaled-up Amazon EC2 instance is associated with multiple applications,

then additional concurrent deployment for each application would be generated. For example, an Auto Scaling group that scales up by five Amazon EC2 instances and is associated with a single application would generate five concurrent deployments. If the same five scaled-up Amazon EC2 instances are associated with two additional applications, this would generate ten additional concurrent deployments.

Deployment Configurations

Maximum 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 or 0 (zero). Zero (0) results in deployment to all instances at once.
Allowed values for a minimum healthy instances setting of FLEET_PERCENT	Any positive integer less than 100 or 0 (zero). Zero (0) results in deployment to all instances at once.
Maximum number of characters in a custom deployment configuration name	100
Characters allowed in a custom deployment configuration name	Letters (a-z, A-Z), numbers (0-9), periods (.), underscores (_), + (plus signs), = (equals signs), , (commas), @ (at signs), - (minus signs).
Disallowed prefixes in a custom deployment configuration name	CodeDeployDefault.
Maximum number of minutes between the first and last traffic shift during an AWS Lambda canary or linear deployment	2880
Maximum percentage of traffic that can be shifted in one increment during an AWS Lambda deployment	99

Deployment Groups

Maximum number of deployment groups associated with a single application	100
Maximum number of tags in a deployment group	10
Maximum number of Auto Scaling groups in a deployment group	10
Maximum number of characters in a deployment group name	100
Characters allowed in a deployment group name	Letters (a-z, A-Z), numbers (0-9), periods (.), underscores (_), + (plus signs), = (equals signs), , (commas), @ (at signs), - (minus signs).
Maximum number of event notification triggers in a deployment group	10

Instances

Maximum number of instances in a single deployment	500
Maximum number of characters in a tag key	128
Maximum number of characters in a tag value	256
Maximum number of instances that can be passed to the BatchGetOnPremisesInstances API action	100
Maximum number of instances that can be used by concurrent deployments that are in progress and associated with one account	500
Required version of AWS SDK for Ruby (aws-sdk-core)	2.1.2 or earlier for AWS CodeDeploy agent versions earlier than 1.0.1.880. 2.2 or earlier for AWS CodeDeploy agent version 1.0.1.880 and later.

Troubleshooting AWS CodeDeploy

Use the topics in this section to help solve problems and errors you might encounter when using AWS CodeDeploy.

Note

The causes of many deployment failures can be identified by reviewing the log files created during the deployment process. For simplicity, we recommend using Amazon CloudWatch Logs to centrally monitor log files instead of viewing them instance by instance. For information, see [View AWS CodeDeploy Logs in CloudWatch Logs Console](#).

Topics

- [General Troubleshooting Issues](#) (p. 344)
- [Troubleshoot EC2/On-Premises Deployment Issues](#) (p. 347)
- [Troubleshoot AWS Lambda Deployment Issues](#) (p. 353)
- [Troubleshoot Deployment Group Issues](#) (p. 354)
- [Troubleshoot Instance Issues](#) (p. 354)
- [Troubleshoot GitHub Token Issues](#) (p. 356)
- [Troubleshoot Auto Scaling Issues](#) (p. 357)
- [Error Codes for AWS CodeDeploy](#) (p. 361)

General Troubleshooting Issues

Topics

- [General Troubleshooting Checklist](#) (p. 344)
- [AWS CodeDeploy deployment resources are supported in certain regions only](#) (p. 345)
- [Required IAM roles are not available](#) (p. 346)
- [Avoid concurrent deployments to the same Amazon EC2 instance](#) (p. 346)
- [Using some text editors to create AppSpec files and shell scripts can cause deployments to fail](#) (p. 347)
- [Using Finder in macOS to bundle an application revision can cause deployments to fail](#) (p. 347)

General Troubleshooting Checklist

You can use the following checklist to troubleshoot a failed deployment.

1. See [View Deployment Details with AWS CodeDeploy](#) (p. 254) and [View Instance Details](#) (p. 197) to determine why the deployment failed. If you are unable to determine the cause, continue to the rest of the items in this checklist.
2. Check whether you have correctly configured the instances:
 - Was the instance launched with an Amazon EC2 key pair specified? For more information, see [Amazon EC2 Key Pairs](#) in *Amazon EC2 User Guide for Linux Instances*.
 - Is the correct IAM instance profile attached to the instance? For more information, see [Configure an Amazon EC2 Instance to Work with AWS CodeDeploy](#) (p. 168) and [Step 4: Create an IAM Instance Profile for Your Amazon EC2 Instances](#) (p. 25).

- Was the instance tagged? For more information, see [Working with Tags in the Console](#) in *Amazon EC2 User Guide for Linux Instances*.
 - Is the AWS CodeDeploy agent installed and running on the instance? For more information, see [Managing AWS CodeDeploy Agent Operations](#) (p. 132).
3. Check the application and deployment group settings:
 - To check your application settings, see [View Application Details with AWS CodeDeploy](#) (p. 216).
 - To check your deployment group settings, see [View Deployment Group Details with AWS CodeDeploy](#) (p. 226).
 4. Confirm the application revision is correctly configured:
 - Check the format of your AppSpec file. For more information, see [Add an Application Specification File to a Revision for AWS CodeDeploy](#) (p. 233) and [AWS CodeDeploy AppSpec File Reference](#) (p. 306).
 - Check your Amazon S3 bucket or GitHub repository to verify your application revision is in the expected location.
 - Review the details of your AWS CodeDeploy application revision to ensure that it is registered correctly. For information, see [View Application Revision Details with AWS CodeDeploy](#) (p. 241).
 - If you're deploying from Amazon S3, check your Amazon S3 bucket to verify AWS CodeDeploy has been granted permissions to download the application revision. For information about bucket policies, see [Deployment Prerequisites](#) (p. 245).
 - If you're deploying from GitHub, check your GitHub repository to verify AWS CodeDeploy has been granted permissions to download the application revision. For more information, see [Create a Deployment with AWS CodeDeploy](#) (p. 245) and [GitHub Authentication with Applications in AWS CodeDeploy](#) (p. 54).
 5. Check whether the service role is correctly configured. For information, see [Step 3: Create a Service Role for AWS CodeDeploy](#) (p. 21).
 6. Confirm you followed the steps in [Getting Started with AWS CodeDeploy](#) (p. 19) to:
 - Attach policies to the IAM user.
 - Install or upgrade and configure the AWS CLI.
 - Create an IAM instance profile and a service role.
- For more information, see [Authentication and Access Control for AWS CodeDeploy](#) (p. 289).
7. Confirm you are using AWS CLI version 1.6.1 or later. To check the version you have installed, call **aws --version**.

If you are still unable to troubleshoot your failed deployment, review the other issues in this topic.

AWS CodeDeploy deployment resources are supported in certain regions only

If you do not see or cannot access applications, deployment groups, instances, or other deployment resources from the AWS CLI or the AWS CodeDeploy console, make sure you're referencing one of the regions listed in [Region and Endpoints](#) in *AWS General Reference*.

Amazon EC2 instances and Auto Scaling groups that will be used in AWS CodeDeploy deployments must be launched and created in one of these regions.

If you're using the AWS CLI, run the `aws configure` command from the AWS CLI. Then you can view and set your default region.

If you're using the AWS CodeDeploy console, on the navigation bar, from the region selector, choose one of the supported regions.

Important

To use services in the China (Beijing) Region or China (Ningxia) Region, you must have an account and credentials for those regions. Accounts and credentials for other AWS regions do not work for the Beijing and Ningxia Regions, and vice versa.

Information about some resources for the China Regions, such as AWS CodeDeploy Resource Kit bucket names and AWS CodeDeploy agent installation procedures, are not included in this edition of the *AWS CodeDeploy User Guide*.

For more information:

- [AWS CodeDeploy](#) in *Getting Started with AWS in the China (Beijing) Region*
- *AWS CodeDeploy User Guide for the China Regions* ([English version](#) | [Chinese version](#))

Required IAM roles are not available

If you rely on an IAM instance profile or a service role that was created as part of an AWS CloudFormation stack, if you delete the stack, all IAM roles are deleted, too. This may be why the IAM role is no longer displayed in the IAM console and AWS CodeDeploy no longer works as expected. To fix this problem, you must manually re-create the deleted IAM role.

Avoid concurrent deployments to the same Amazon EC2 instance

As a best practice, you should avoid situations that would result in more than one attempted deployment to an Amazon EC2 instance at the same time. In cases where commands from different deployments compete to run on a single instance, the deployments can time out and fail for the following reasons:

- AWS CodeDeploy fails a deployment if its first lifecycle event doesn't start within five minutes of the triggering of the deployment. You can use the console or the AWS CLI [create-deployment](#) command to trigger a deployment.
- AWS CodeDeploy fails a deployment if a lifecycle event does not start within five minutes of the end of the previous lifecycle event.
- The AWS CodeDeploy agent can process only one deployment command at a time.
- It's not possible to control the order in which deployments occur if more than one deployment attempts to run at the same time.

Note

The default timeout for a script in a lifecycle event is 30 minutes. You can change the timeout to a different value in the AppSpec file. For more information, see [Add an AppSpec File for an EC2/On-Premises Deployment](#) (p. 234).

AWS CodeDeploy logic considers a deployment to have failed if its steps are not complete within five minutes, even if a deployment process is otherwise running as expected. The five-minute limit can be exceeded if commands from multiple deployments are being sent to the AWS CodeDeploy agent at the same time.

For information about other challenges you might face with concurrent deployments in Auto Scaling groups, see [Avoid associating multiple deployment groups with a single Auto Scaling group](#) (p. 359).

Using some text editors to create AppSpec files and shell scripts can cause deployments to fail

Some text editors introduce non-conforming, non-printing characters into files. If you use text editors to create or modify AppSpec files or shell script files to run on Amazon Linux, Ubuntu Server, or RHEL instances, then any deployments that rely on these files might fail. When AWS CodeDeploy uses these files during a deployment, the presence of these characters can lead to hard-to-troubleshoot AppSpec file validation failures and script execution failures.

In the AWS CodeDeploy console, on the event details page for the deployment, choose **View logs**. (Alternatively, you use the AWS CLI to call the [get-deployment-instance](#) command.) Look for errors like "invalid character," "command not found," or "file not found."

To address this issue, we recommend the following:

- Do not use text editors that automatically 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 in your AppSpec files and shell script files, so you can find 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, Ubuntu Server, or RHEL instances to create shell script files that run on Amazon Linux, Ubuntu Server, or RHEL instances. For examples of these types of text editors, search the Internet for "Linux shell script editor."
- If you must use a text editor in Windows or Mac OS to create shell script files to run on Amazon Linux, Ubuntu Server, or RHEL instances, use a program or utility that converts text in Windows or Mac OS format to Unix format. For examples of these programs and utilities, search the Internet for "DOS to UNIX" or "Mac to UNIX." Be sure to test the converted shell script files on the target operating systems.

Using Finder in macOS to bundle an application revision can cause deployments to fail

Deployments might fail if you use the Finder graphical user interface (GUI) application on a Mac to bundle (zip) an AppSpec file and related files and scripts into an application revision archive (.zip) file. This is because Finder creates an intermediate `__MACOSX` folder in the .zip file and places component files into it. AWS CodeDeploy cannot find the component files, so the deployment fails.

To address this issue, we recommend you use the AWS CLI to call the [push](#) command, which zips the component files into the expected structure. Alternatively, you can use Terminal instead of the GUI to zip the component files. Terminal does not create an intermediate `__MACOSX` folder.

Troubleshoot EC2/On-Premises Deployment Issues

Topics

- [Deployment fails with the message "Validation of PKCS7 signed message failed" \(p. 348\)](#)
- [Deployment or redeployment of the same files to the same instance locations fail with the error "The deployment failed because a specified file already exists at this location" \(p. 348\)](#)
- [Troubleshooting a failed AllowTraffic lifecycle event with no error reported in the deployment logs \(p. 350\)](#)
- [Troubleshooting failed ApplicationStop, BeforeBlockTraffic, and AfterBlockTraffic deployment lifecycle events \(p. 350\)](#)

- [Troubleshooting a failed DownloadBundle deployment lifecycle event with "UnknownError: not opened for reading" \(p. 351\)](#)
- [Windows PowerShell scripts fail to use the 64-bit version of Windows PowerShell by default \(p. 352\)](#)
- [Long-running processes can cause deployments to fail \(p. 352\)](#)

Note

The causes of many deployment failures can be identified by reviewing the log files created during the deployment process. For simplicity, we recommend using Amazon CloudWatch Logs to centrally monitor log files instead of viewing them instance by instance. For information, see [View AWS CodeDeploy Logs in CloudWatch Logs Console](#).

Deployment fails with the message "Validation of PKCS7 signed message failed"

This error message indicates the instance is running a version of the AWS CodeDeploy agent that supports only the SHA-1 hash algorithm. Support for the SHA-2 hash algorithm was introduced in version 1.0.1.854 of the AWS CodeDeploy agent, released in November 2015. Effective October 17, 2016, deployments will fail if a version of the AWS CodeDeploy agent earlier than 1.0.1.854 is installed. For more information, see [AWS to Switch to SHA256 Hash Algorithm for SSL Certificates](#), [NOTICE: Retiring AWS CodeDeploy host agents older than version 1.0.1.85](#), and [Update the AWS CodeDeploy Agent \(p. 141\)](#).

Deployment or redeployment of the same files to the same instance locations fail with the error "The deployment failed because a specified file already exists at this location"

When AWS CodeDeploy tries to deploy a file to an instance but a file with the same name already exists in the specified target location, the deployment to that instance may fail. You may receive the error message "The deployment failed because a specified file already exists at this location: *location-name*." This is because, during each deployment, AWS CodeDeploy first deletes all files from the previous deployment, which are listed in a cleanup log file. If there are files in the target installation folders that aren't listed in this cleanup file, the AWS CodeDeploy agent by default interprets this as an error and fails the deployment.

Note

On Amazon Linux, RHEL, and Ubuntu Server instances, the cleanup file is located in `/opt/codedeploy-agent/deployment-root/deployment-instructions/`. On Windows Server instances, the location is `C:\ProgramData\Amazon\CodeDeploy\deployment-instructions\`.

The easiest way to avoid this error is to specify an option other than the default behavior to fail the deployment. For each deployment, you can choose whether to fail the deployment, to overwrite the files not listed in the cleanup file, or to retain the files already on the instance.

The overwrite option is useful when, for example, you manually placed a file on an instance after the last deployment, but then added a file of the same name to the next application revision.

You might choose the retain option for files you place on the instance that you want to be part of the next deployment without having to add them to the application revision package. The retain option is also useful if your application files are already in your production environment and you want to

Deployment or redeployment of the same files to the same instance locations fail with the error "The deployment failed because a specified file already exists at this location"

deploy using AWS CodeDeploy for the first time. For more information, see [Create an EC2/On-Premises Compute Platform Deployment \(Console\)](#) (p. 248) and [Rollback Behavior with Existing Content](#) (p. 260).

Troubleshooting "The deployment failed because a specified file already exists at this location" deployment errors

If you choose not to specify an option to overwrite or retain content that AWS CodeDeploy detects in your target deployment locations (or if you do not specify any deployment option for handling existing content in a programmatic command), you can choose to troubleshoot the error.

The following information applies only if you choose not to retain or overwrite the content.

If you try to redeploy files with the same names and locations, the redeployment will have a better chance of succeeding if you specify the application name and the deployment group with the same underlying deployment group ID you used before. AWS CodeDeploy uses the underlying deployment group ID to identify files to remove before a redeployment.

Deploying new files or redeploying the same files to the same locations on instances can fail for these reasons:

- You specified a different application name for a redeployment of the same revision to the same instances. The redeployment will fail because even if the deployment group name is the same, the use of a different application name means a different underlying deployment group ID will be used.
- You deleted and re-created a deployment group for an application and then tried to redeploy the same revision to the deployment group. The redeployment will fail because even if the deployment group name is the same, AWS CodeDeploy will reference a different underlying deployment group ID.
- You deleted an application and deployment group in AWS CodeDeploy, then created a new application and deployment group with the same names as the ones you deleted. After that, you tried to redeploy a revision that had been deployed to the previous deployment group to the new one with the same name. The redeployment will fail because even though the application and deployment group names are the same, AWS CodeDeploy still references the ID of the deployment group you deleted.
- You deployed a revision to a deployment group and then deployed the same revision to another deployment group to the same instances. The second deployment will fail because AWS CodeDeploy will reference a different underlying deployment group ID.
- You deployed a revision to one deployment group and then deployed another revision to another deployment group to the same instances. There is at least one file with the same name and in the same location that the second deployment group tries to deploy. The second deployment will fail because AWS CodeDeploy will not remove the existing file before the second deployment starts. Both deployments will reference different deployment group IDs.
- You deployed a revision in AWS CodeDeploy, but there is at least one file with the same name and in the same location. The deployment will fail because, by default, AWS CodeDeploy will not remove the existing file before the deployment starts.

To address these situations, do one of the following:

- Remove the files from the locations and instances to which they were previously deployed, 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 delete files in any locations that match the files 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 those you are about to delete. (You can use the [get-deployment-group](#) command to retrieve the deployment group ID.) AWS CodeDeploy will use the underlying deployment group ID and AppSpec file to remove all of the files it installed in the previous successful deployment.

Troubleshooting a failed AllowTraffic lifecycle event with no error reported in the deployment logs

In some cases, a blue/green deployment fails during the AllowTraffic lifecycle event, but no cause for the failure is indicated in the deployment logs.

This failure is typically due to the health checks being configured incorrectly in Elastic Load Balancing for the Classic Load Balancer, Application Load Balancer, or Network Load Balancer used to manage traffic for the deployment group.

To resolve the issue, review and correct any errors in the the health check configuration for the load balancer.

For Classic Load Balancers, see [Configure Health Checks](#) in the *User Guide for Classic Load Balancers* and [ConfigureHealthCheck](#) in the *Elastic Load Balancing API Reference version 2012-06-01*.

For Application Load Balancers, see [Health Checks for Your Target Groups](#) in the *User Guide for Application Load Balancers*.

For Network Load Balancers, see [Health Checks for Your Target Groups](#) in the *Network Load Balancer User Guide*.

Troubleshooting failed ApplicationStop, BeforeBlockTraffic, and AfterBlockTraffic deployment lifecycle events

During a deployment, the AWS CodeDeploy agent runs the scripts specified for ApplicationStop, BeforeBlockTraffic, and AfterBlockTraffic in the AppSpec file from the *previous* successful deployment. (All other scripts are run from the AppSpec file in the current deployment.) If one of these scripts contains an error and does not run successfully, the deployment can fail.

Possible reasons for these failures include:

- The AWS CodeDeploy agent finds the `deployment-group-id_last_successful_install` file in the correct location, but the location listed in the `deployment-group-id_last_successful_install` file does not exist.

On Amazon Linux, Ubuntu Server, and RHEL instances, this file must exist in `/opt/codedeploy-agent/deployment-root/deployment-instructions`.

On Windows Server instances, this file must be stored in the `C:\ProgramData\Amazon\CodeDeploy\deployment-instructions` folder.

- In the location listed in the `deployment-group-id_last_successful_install` file, either the AppSpec file is invalid or the scripts do not run successfully.
- The script contains an error that cannot be corrected, so it will never run successfully.

Use the AWS CodeDeploy console to investigate why a deployment might have failed during any of these events. On the details page for the deployment, choose **View events**. On the details page for

the instance, in the **ApplicationStop**, **BeforeBlockTraffic**, or **AfterBlockTraffic** row, choose **View logs**. Alternatively, use the AWS CLI to call the [get-deployment-instance](#) command.

If the cause of the failure is a script from the last successful deployment that will never run successfully, create a new deployment and specify that the **ApplicationStop**, **BeforeBlockTraffic**, and **AfterBlockTraffic** failures should be ignored. You can do this in two ways:

- Use the AWS CodeDeploy console to create a deployment. On the **Create deployment** page, under **ApplicationStop lifecycle event failure**, choose **Don't fail the deployment to an instance if this lifecycle event on the instance fails**.
- Use the AWS CLI to call the [create-deployment](#) command and include the `--ignore-application-stop-failures` option.

When you deploy the application revision again, the deployment will continue even if any of these three lifecycle events fail. If the new revision includes fixed scripts for those lifecycle events, future deployments can succeed without applying this fix.

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 "UnknownError: not opened for reading" error:

- There was internal Amazon S3 service error. Deploy the application revision again.
- The IAM instance profile on your Amazon EC2 instance does not have permissions to access the application revision in Amazon S3. For information about Amazon S3 bucket policies, see [Push a Revision for AWS CodeDeploy to Amazon S3 \(p. 238\)](#) and [Deployment Prerequisites \(p. 245\)](#).
- The instances to which you will deploy are associated with one region (for example, US West (Oregon)), but the Amazon S3 bucket that contains the application revision is associated with another region (for example, US East (N. Virginia)). Make sure the application revision is in an Amazon S3 bucket associated with the same region as the instances.

On the event details page for the deployment, in the **Download bundle** row, choose **View logs**. Alternatively, use the AWS CLI to call the [get-deployment-instance](#) command. If this error occurred, there should be an error in the output with the error code "UnknownError" and the error message "not opened for reading."

To determine the reason for this error:

1. Enable wire logging on at least one of the instances, and then deploy the application revision again.
2. Examine the wire logging file to find the error. Common error messages for this issue include the phrase "access denied."
3. After you have examined the log files, we recommend that you disable wire logging to reduce log file size and the amount of sensitive information that may appear in the output in plain text on the instance in the future.

To learn how to find the wire logging file and enable and disable wire logging, see `:log_aws_wire:` in [Working with the AWS CodeDeploy Agent \(p. 125\)](#).

Windows PowerShell scripts fail to use the 64-bit version of Windows PowerShell by default

If a Windows PowerShell script running as part of a deployment relies on 64-bit functionality (for example, because it consumes more memory than a 32-bit application will allow or calls libraries that are offered only in a 64-bit version), the script may crash or otherwise not run as expected. This is because, by default, AWS CodeDeploy uses the 32-bit version of Windows PowerShell to run Windows PowerShell scripts that are part of an application revision.

Add code like the following to the beginning of any script that must 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 $LastExitCode
}

# 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 this code may seem counterintuitive, 32-bit Windows PowerShell uses a path like:

```
c:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
```

64-bit Windows PowerShell uses a path like:

```
c:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
```

Long-running processes can cause deployments to fail

For deployments to Amazon Linux, Ubuntu Server, and RHEL instances, if you have a deployment script that starts a long-running process, AWS CodeDeploy may spend a long time waiting in the deployment lifecycle event and then fail the deployment. This is because if the process runs longer than the foreground and background processes in that event are expected to take, AWS CodeDeploy stops and fails the deployment, even if the process is still running as expected.

For example, an application revision contains two files in its root, `after-install.sh` and `sleep.sh`. 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 `after-install.sh` file runs during the **AfterInstall** application lifecycle event. Here are its contents:

```
#!/bin/bash
/tmp/sleep.sh
```

The `sleep.sh` file contains the following, which suspends program execution for three minutes (180 seconds), simulating some long-running process:

```
#!/bin/bash
sleep 180
```

When `after-install.sh` calls `sleep.sh`, `sleep.sh` will start and keep running for three minutes (180 seconds), which is two minutes (120 seconds) past the time AWS CodeDeploy expects `sleep.sh` (and, by relation, `after-install.sh`) to stop running. After the timeout of one minute (60 seconds), AWS CodeDeploy stops and fails the deployment at the **AfterInstall** application lifecycle event, even though `sleep.sh` continues to run as expected. The following error is displayed:

```
Script at specified location: after-install.sh failed to complete in 60 seconds.
```

You cannot simply add an ampersand (&) in `after-install.sh` to run `sleep.sh` in the background.

```
#!/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.

In `after-install.sh`, call `sleep.sh` as follows, which enables AWS CodeDeploy to continue 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 you want to start running in the background, redirecting stdout, stderr, and stdin to `/dev/null`.

Troubleshoot AWS Lambda Deployment Issues

Topics

- [AWS Lambda deployments fail after manually stopping a Lambda deployment that does not have configured rollbacks \(p. 354\)](#)

AWS Lambda deployments fail after manually stopping a Lambda deployment that does not have configured rollbacks

In some cases, the alias of a Lambda function specified in a deployment might reference two different versions of the function. The result is that subsequent attempts to deploy the Lambda function fail. A Lambda deployment can get in this state when it does not have rollbacks configured and is manually stopped. To proceed, use the AWS Lambda console to make sure the function is not configured to traffic shift between two versions:

1. Sign in to the AWS Management Console and open the AWS Lambda console at <https://console.aws.amazon.com/lambda/>.
2. From the left pane, choose **Functions**.
3. Select the name of the Lambda function that is in your AWS CodeDeploy deployment.
4. From **Qualifiers**, choose the alias used in your AWS CodeDeploy deployment.
5. From **Additional Version**, choose **<none>**. This ensures the alias is not configured to shift a percentage, or weight, of traffic to more than one version. Make a note of the version selected in the **Version** drop-down menu.
6. Choose **Save and test**.
7. Open the AWS CodeDeploy console and attempt a deployment of the version displayed in the drop-down menu in step 5.

Troubleshoot Deployment Group Issues

Tagging an instance as part of a deployment group does not automatically deploy your application to the new instance

AWS CodeDeploy does not automatically deploy your application to a newly tagged instance. You must create a new deployment in the deployment group.

You can use AWS CodeDeploy to enable automatic deployments to new Amazon EC2 instances in Auto Scaling groups. For more information, see [Integrating AWS CodeDeploy with Auto Scaling \(p. 44\)](#).

Troubleshoot Instance Issues

Topics

- [Tags must be set correctly \(p. 355\)](#)
- [AWS CodeDeploy agent must be installed and running on instances \(p. 355\)](#)
- [Deployments do not fail for up to an hour when an instance is terminated during a deployment \(p. 355\)](#)
- [Analyzing log files to investigate deployment failures on instances \(p. 355\)](#)
- [Create a new AWS CodeDeploy log file if it was accidentally deleted \(p. 355\)](#)

- [Troubleshooting “InvalidSignatureException – Signature expired: \[time\] is now earlier than \[time\]” deployment errors \(p. 356\)](#)

Tags must be set correctly

Use the [list-deployment-instances](#) command to confirm the instances used for a deployment are tagged correctly. If an Amazon EC2 instance is missing in the output, use the Amazon EC2 console to confirm the tags have been set on the instance. For more information, see [Working with Tags in the Console](#) in the *Amazon EC2 User Guide for Linux Instances*.

Note

If you tag an instance and immediately use AWS CodeDeploy to deploy an application to it, the instance might not be included in the deployment. This is because it can take several minutes before AWS CodeDeploy can read the tags. We recommend that you wait at least five minutes between the time you tag an instance and attempt to deploy to it.

AWS CodeDeploy agent must be installed and running on instances

To verify the AWS CodeDeploy agent is installed and running on an instance, see [Verify the AWS CodeDeploy Agent Is Running \(p. 132\)](#).

To install, uninstall, or reinstall the AWS CodeDeploy agent, see [Install or Reinstall the AWS CodeDeploy Agent \(p. 134\)](#).

Deployments do not fail for up to an hour when an instance is terminated during a deployment

AWS CodeDeploy provides a one-hour window for each deployment lifecycle event to run to completion. This provides ample time for long-running scripts.

If anything occurs that prevents scripts from running to completion while a lifecycle event is in progress (for example, if an instance is terminated or the AWS CodeDeploy agent is shut down), it might take up to an hour for the status of the deployment to be displayed as **Failed**. This is true even if the timeout period specified in the script is shorter than an hour. This is because when the instance is terminated, the AWS CodeDeploy agent will shut down and will be unable to process any additional scripts.

If an instance is terminated between lifecycle events or before the first lifecycle event step starts, however, the timeout occurs after just five minutes.

Analyzing log files to investigate deployment failures on instances

If the status of an instance in the deployment is anything other than Succeeded, you can review the deployment log file data to help identify the problem. For information about accessing deployment log data, see [View Log Data for AWS CodeDeploy Deployments \(p. 255\)](#).

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 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, Ubuntu Server, or RHEL 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
```

Troubleshooting “InvalidSignatureException – Signature expired: [time] is now earlier than [time]” deployment errors

AWS CodeDeploy requires accurate time references in order to perform its operations. If your instance's date and time are not set correctly, they may not match the signature date of your deployment request, which AWS CodeDeploy will therefore reject.

To avoid deployment failures related to incorrect time settings, see the following topics:

- [Setting the Time for Your Linux Instance](#)
- [Setting the Time for a Windows Instance](#)

Troubleshoot GitHub Token Issues

Invalid GitHub OAuth token

AWS CodeDeploy applications created after June 2017 use GitHub OAuth tokens for each AWS region. The use of tokens tied to specific regions gives you more control over which AWS CodeDeploy applications have access to a GitHub repository.

If you receive a GitHub token error, you might have an older token that is now invalid.

To fix an invalid GitHub OAuth token:

1. Use [DeleteGitHubAccountToken](#) to remove the old token.
2. Add a new OAuth token. For more information, see [Integrating AWS CodeDeploy with GitHub \(p. 53\)](#).

Maximum Number of GitHub OAuth Tokens Exceeded

When you create an AWS CodeDeploy deployment, the maximum number of allowed GitHub tokens is 10. If you receive an error about GitHub OAuth tokens, make sure you have 10 or fewer tokens. If you have more than 10 tokens, the first tokens that were created are invalid. For example, if you have 11 tokens, the first token you created is invalid. If you have 12 tokens, the first two tokens you created are invalid. For information about using the AWS CodeDeploy API to remove old tokens, see [DeleteGitHubAccountToken](#).

Troubleshoot Auto Scaling Issues

Topics

- [General Auto Scaling troubleshooting \(p. 357\)](#)
- [Instances in an Auto Scaling group are continuously provisioned and terminated before a revision can be deployed \(p. 358\)](#)
- [Terminating or rebooting an Auto Scaling instance may cause deployments to fail \(p. 358\)](#)
- [Avoid associating multiple deployment groups with a single Auto Scaling group \(p. 359\)](#)
- [Amazon EC2 instances in an Auto Scaling group fail to launch and receive the error "Heartbeat Timeout" \(p. 359\)](#)
- [Mismatched Auto Scaling lifecycle hooks might cause automatic deployments to Auto Scaling groups to stop or fail \(p. 360\)](#)

General Auto Scaling troubleshooting

Deployments to Amazon EC2 instances in an Auto Scaling group can 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.

Disassociate the Auto Scaling group from the AWS CodeDeploy deployment group or change the configuration of your Auto Scaling group so that the desired number of instances matches the current number of instances (thus preventing Auto Scaling from launching any more Amazon EC2 instances). For more information, see [Change Deployment Group Settings with AWS CodeDeploy \(p. 227\)](#) or [Configuring Your Auto Scaling Groups](#).

- **The AWS CodeDeploy agent is unresponsive.** The AWS CodeDeploy agent may not be installed if initialization scripts (for example, cloud-init scripts) that run immediately after an Amazon EC2 instance is launched or started take more than one hour to run. AWS CodeDeploy has a one-hour timeout for the AWS CodeDeploy agent to respond to 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.** Your deployment can fail if an Amazon EC2 instance is rebooted during a deployment or the AWS CodeDeploy agent is shut down while processing a deployment command. For more information, see [Terminating or rebooting an Auto Scaling instance may cause deployments to fail \(p. 358\)](#).
- **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. 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, running the scripts in a deployment can prevent the launching of Amazon EC2 instances in the Auto Scaling group. (Other Amazon EC2 instances in the Auto Scaling group may appear to be running normally.) To address this issue, make sure that all other scripts are complete first:
 - **AWS CodeDeploy agent is not included in your AMI :** If you use the `cfn-init` command to install the AWS CodeDeploy agent while launching a new instance, place the agent installation script at the end of the `cfn-init` section of your AWS CloudFormation template.
 - **AWS CodeDeploy agent is included in your AMI :** If you include the AWS CodeDeploy agent in your AMI, configure it so that the agent is in a `Stopped` state when the instance is created, and then include a script for starting the agent as the final step in your `cfn-init` script library.

Instances in an Auto Scaling group are continuously provisioned and terminated before a revision can be deployed

In some cases, an error can prevent a successful deployment to newly provisioned instances in an Auto Scaling group. The results are no healthy instances and no successful deployments. Because the deployment can't run or be completed successfully, the instances are terminated soon after being created. The Auto Scaling group configuration then causes another batch of instances to be provisioned to try to meet the minimum healthy hosts requirement. This batch is also terminated, and the cycle continues.

Possible causes include:

- Failed Auto Scaling group health checks
- An error in the application revision

To work around this issue, follow these steps:

1. Manually create an Amazon EC2 instance that is not part of the Auto Scaling group. Tag the instance with a unique EC2 instance tag.
2. Add the new instance to the affected deployment group.
3. Deploy a new, error-free application revision to the deployment group.

This will prompt the Auto Scaling group to deploy the application revision to future instances in the Auto Scaling group.

Note

After you confirm that deployments are successful, you can delete the instance you created to avoid ongoing billing charges for it.

Terminating or rebooting an Auto Scaling instance may cause deployments to fail

If an Amazon EC2 instance is launched through Auto Scaling, and the instance is then terminated or rebooted, deployments to that instance may fail for 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 be completed, it fails.
- The instance is rebooted, but it takes more than five minutes for the instance to start. AWS CodeDeploy considers this to be a timeout. The service will fail all current and future deployments to the instance.

To address this issue:

- In general, make sure all deployments are complete before the instance is terminated or rebooted. Make sure all deployments start after the instance has started or been rebooted.

- 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 computer name of the instance, this behavior can cause deployments to fail. To disable this behavior, in the Windows Server base AMI, on the **General** tab of the **EC2 Service Properties** dialog box, clear the **Set Computer Name** box. After you clear this box, this behavior will be disabled for all new Windows Server Auto Scaling instances launched with that Windows Server base AMI. For Windows Server Auto Scaling instances on which this behavior enabled, you do not need to clear this box. Simply redeploy failed deployments to those instances after they have been rebooted.

Avoid associating multiple deployment groups with a single Auto Scaling group

As a best practice, you should associate only one deployment group with each Auto Scaling group.

This is because if Auto Scaling scales up an instance that has hooks associated with multiple deployment groups, it sends notifications for all of the hooks at once. This causes multiple deployments to each instance to start at the same time. When multiple deployments send commands to the AWS CodeDeploy agent at the same time, the five-minute timeout between a lifecycle event and either the start of the deployment or the end of the previous lifecycle event might be reached. If this happens, the deployment fails, even if a deployment process is otherwise running as expected.)

Note

The default timeout for a script in a lifecycle event is 30 minutes. You can change the timeout to a different value in the AppSpec file. For more information, see [Add an AppSpec File for an EC2/On-Premises Deployment](#) (p. 234).

It's not possible to control the order in which deployments occur if more than one deployment attempts to run at the same time.

Finally, if deployment to any instance fails, Auto Scaling immediately terminates the instance. When that first instance shuts down, the other deployments that were running will begin to fail. Because AWS CodeDeploy has a one-hour timeout for the AWS CodeDeploy agent to respond to pending deployments, it can take up to 60 minutes for each instance to time out.

For more information about problems with attempting multiple deployments to an instance at the same time, see [Avoid concurrent deployments to the same Amazon EC2 instance](#) (p. 346).

For more information about Auto Scaling, see [Under the Hood: AWS CodeDeploy and Auto Scaling Integration](#).

Amazon EC2 instances in an Auto Scaling group fail to launch and receive the error "Heartbeat Timeout"

An Auto Scaling group might fail to launch new Amazon EC2 instances, generating a message similar to the following:

```
Launching a new Amazon 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 message usually indicates one of the following:

- The maximum number of concurrent deployments associated with an AWS account was reached. For more information about deployment limits, see [AWS CodeDeploy Limits](#) (p. 340).
- An application in AWS CodeDeploy was deleted before its associated deployment groups were updated or deleted.

When you delete an application or deployment group, AWS CodeDeploy attempts to clean up any Auto Scaling hooks associated with it, but some hooks might remain. If you run a command to delete a deployment group, the leftover hooks will be returned in the output; however, if you run a command to delete an application, the leftover hooks will not appear in the output.

Therefore, as a best practice, you should delete all deployment groups associated with an application before you delete the application. You can use the command output to identify the lifecycle hooks that must be deleted manually.

If you are receiving a “Heartbeat Timeout” error message, you can determine whether leftover lifecycle hooks are the cause and resolve the problem by doing the following:

1. Run either the [update-deployment-group](#) command or [delete-deployment-group](#) command. Examine the output of the call. If the output contains a `hooksNotCleanedUp` structure with a list of Auto Scaling lifecycle hooks, leftover lifecycle hooks are most likely the cause of the error.
2. Call the [describe-lifecycle-hooks](#) command, specifying the name of the Auto Scaling group associated with the Amazon EC2 instances that fail to launch. In the output, look for any Auto Scaling lifecycle hook names that correspond to the `hooksNotCleanedUp` structure you identified in step 1. Alternatively, look for Auto Scaling lifecycle hook names that contain the name of the deployment group.
3. Call the [delete-lifecycle-hook](#) command for each Auto Scaling lifecycle hook. Specify the Auto Scaling group and lifecycle hook.

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 deploy to Amazon EC2 instances that are scaled up as part of that Auto Scaling group.

Mismatched Auto Scaling lifecycle hooks might 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 deployed to which Amazon EC2 instances after they are launched in Auto Scaling groups. Automatic deployments can stop or fail if lifecycle hooks and information about these hooks do not match exactly in Auto Scaling and AWS CodeDeploy.

If deployments to an Auto Scaling group are failing, see if the lifecycle hook names in Auto Scaling and AWS CodeDeploy match. If not, use these AWS CLI command calls.

First, get the list of lifecycle hook names for both the Auto Scaling group and the deployment group:

1. Call the [describe-lifecycle-hooks](#) command, specifying the name of the Auto Scaling group associated with the deployment group in AWS CodeDeploy. In the output, in the `LifecycleHooks` list, make a note of each `LifecycleHookName` value.
2. Call the [get-deployment-group](#) command, specifying the name of the deployment group associated with the Auto Scaling group. In the output, in 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. You may want to try other Auto Scaling troubleshooting steps described elsewhere in this section.

However, if the two sets of lifecycle hook names do not match exactly, character for character, do the following:

1. If there are lifecycle hook names in the **describe-lifecycle-hooks** command output that are not also in the **get-deployment-group** command output, then do the following:
 - a. For each lifecycle hook name in the **describe-lifecycle-hooks** command output, call the [delete-lifecycle-hook](#) command.
 - b. 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 associate the lifecycle hooks with the deployment group. Automatic deployments should now resume as new instances are added to the Auto Scaling group.
2. If there are lifecycle hook names in the **get-deployment-group** command output that are not also in the **describe-lifecycle-hooks** command output, then do the following:
 - a. Call the [update-deployment-group](#) command, but do not specify the name of the original Auto Scaling group.
 - b. Call the **update-deployment-group** command again, but this time specify the name of the original Auto Scaling group. AWS CodeDeploy will re-create the missing lifecycle hooks in the Auto Scaling group. Automatic deployments should now resume as new instances are added to the Auto Scaling group.

After you get the two sets of lifecycle hook names to match exactly, character for character, application revisions should be deployed again, but only to new instances as they are added to the Auto Scaling group. Deployments will not occur automatically to instances already in the Auto Scaling group.

Error Codes for AWS CodeDeploy

This topic provides reference information about AWS CodeDeploy errors.

Error Code	Description
AGENT_ISSUE	<p>The deployment failed because of a problem with the AWS CodeDeploy agent. Make sure the agent is installed and running on all instances in this deployment group.</p> <p>Learn more:</p> <ul style="list-style-type: none"> • Verify the AWS CodeDeploy Agent Is Running (p. 132) • Install or Reinstall the AWS CodeDeploy Agent (p. 134) • Working with the AWS CodeDeploy Agent (p. 125)
HEALTH_CONSTRAINTS	<p>The overall deployment failed because too many individual instances failed deployment, too few healthy instances are available for deployment, or some instances in your deployment group are experiencing problems.</p> <p>Learn more:</p> <ul style="list-style-type: none"> • Instance Health (p. 198) • Troubleshoot Instance Issues (p. 354)

Error Code	Description
	<ul style="list-style-type: none">• Troubleshoot EC2/On-Premises Deployment Issues (p. 347)
HEALTH_CONSTRAINTS_INVALID	<p>The deployment can't start because the minimum number of healthy instances, as defined by your deployment configuration, are not available. You can reduce the required number of healthy instances by updating your deployment configuration or increase the number of instances in this deployment group.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Instance Health (p. 198)• Working with Instances for AWS CodeDeploy (p. 147)
IAM_ROLE_MISSING	<p>The deployment failed because no service role exists with the service role name specified for the deployment group. Make sure you are using the correct service role name.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Step 3: Create a Service Role for AWS CodeDeploy (p. 21)• Change Deployment Group Settings with AWS CodeDeploy (p. 227)
IAM_ROLE_PERMISSIONS	<p>AWS CodeDeploy does not have the permissions required to assume a role, or the IAM role you're using doesn't give you permission to perform operations in an AWS service.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Step 1: Provision an IAM User (p. 19)• Step 3: Create a Service Role for AWS CodeDeploy (p. 21)• Step 4: Create an IAM Instance Profile for Your Amazon EC2 Instances (p. 25)
AUTO_SCALING_IAM_ROLE_PERMISSIONS	<p>The service role associated with your deployment group does not have the permission required to perform operations in the following AWS service.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Step 3: Create a Service Role for AWS CodeDeploy (p. 21)• Creating a Role to Delegate Permissions to an AWS Service

Error Code	Description
OVER_MAX_INSTANCES	<p>The deployment failed because more instances are targeted for deployment than are allowed for your account. To reduce the number of instances targeted for this deployment, update the tag settings for this deployment group or delete some of the targeted instances. Alternatively, you can contact AWS Support to request a limit increase.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Change Deployment Group Settings with AWS CodeDeploy (p. 227)• AWS CodeDeploy Limits (p. 340)• Request a Limit Increase
THROTTLED	<p>The deployment failed because more requests were made than are permitted for AWS CodeDeploy by an IAM role. Try reducing the number of requests.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Query API Request Rate
UNABLE_TO_SEND_ASG	<p>The deployment failed because the deployment group isn't configured correctly with its Auto Scaling group. In the AWS CodeDeploy console, delete the Auto Scaling group from the deployment group, and then add it again.</p> <p>Learn more:</p> <ul style="list-style-type: none">• Under the Hood: AWS CodeDeploy and Auto Scaling Integration

Related Topics

[Troubleshooting AWS CodeDeploy \(p. 344\)](#)

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 from customers 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 resources, such as forums, technical FAQs, service health status, and AWS Trusted Advisor.
- [AWS Support Plans](#) — The primary web page for information about AWS Support plans.
- [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 DevOps Blog](#) — Insights for developers, system administrators, and architects.

AWS Software Development Kits and Tools

The following AWS SDKs and tools support solution development with AWS CodeDeploy:

- [AWS SDK for .NET](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript](#)
- [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 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 major changes made to this user guide to support new and enhanced functionality since the last release of the *AWS CodeDeploy User Guide*.

- **API version:** 2014-10-06
- **Latest documentation update:** September 21, 2018

update-history-change	update-history-description	update-history-date
New minimum supported version of the AWS CodeDeploy agent (p. 366)	The minimum supported version of the AWS CodeDeploy agent is now 1.0.1.1458. For more information, see Version History of the AWS CodeDeploy Agent .	August 7, 2018

Earlier Updates

The following table describes important changes in each release of the *AWS CodeDeploy User Guide* before June 2018.

Change	Description	Date Changed
Topic updates	AWS CodeDeploy is now available in the EU (Paris) Region (eu-west-3). Several topics, including those containing instructions for setting up the AWS CodeDeploy agent, were updated to reflect the availability of this new region.	December 19, 2017
Updated topics	<p>AWS CodeDeploy is now available in the China (Ningxia) Region.</p> <p>To use services in the China (Beijing) Region or China (Ningxia) Region, you must have an account and credentials for those regions. Accounts and credentials for other AWS regions do not work for the Beijing and Ningxia Regions, and vice versa.</p> <p>Information about some resources for the China Regions, such as AWS CodeDeploy Resource Kit bucket names and AWS CodeDeploy agent installation procedures, are not included in this edition of the <i>AWS CodeDeploy User Guide</i>.</p> <p>For more information:</p> <ul style="list-style-type: none">• AWS CodeDeploy in <i>Getting Started with AWS in the China (Beijing) Region</i>• AWS CodeDeploy User Guide for the China Regions (English version Chinese version)	December 11, 2017
Updated topics	AWS CodeDeploy now supports the deployment of a Lambda function. An AWS Lambda deployment shifts	November 28, 2017

Change	Description	Date Changed
	incoming traffic from an existing Lambda function to an updated Lambda function version. You choose or create a deployment configuration to specify the number of traffic shifting intervals in the deployment and the percentage of traffic to shift in each interval. AWS Lambda deployments are supported by the AWS Serverless Application Model (AWS SAM) so that you can use an AWS SAM deployment preference to manage the way in which traffic is shifted during an AWS Lambda deployment. Several topics have been added and updated to reflect this change, including Overview of AWS CodeDeploy Compute Platforms (p. 2), Deployments on an AWS Lambda Compute Platform (p. 9), Create an AWS Lambda Compute Platform Deployment (Console) (p. 247), Create an Application for an AWS Lambda Function Deployment (Console) (p. 215), and Add an AppSpec File for an AWS Lambda Deployment (p. 233).	
New topic	AWS CodeDeploy now supports deployments directly to a local machine or instance where the AWS CodeDeploy Agent is installed. You can locally test a deployment and, if it has errors, use AWS CodeDeploy Agent error logs to debug them. You can also use a local deployment to test the integrity of an application revision, the contents of an AppSpec file, and more. For more information, see Use the AWS CodeDeploy Agent to Validate a Deployment Package on a Local Machine (p. 265).	November 16, 2017
Updated topics	AWS CodeDeploy support for Elastic Load Balancing load balancers in deployment groups has been expanded to include Network Load Balancers for both blue/green deployments and in-place deployments. You can now choose an Application Load Balancer, Classic Load Balancer, or Network Load Balancer for your deployment group. Load balancers are required for blue/green deployments and optional for in-place deployments. A number of topics have been updated to reflect this support, including Integrating AWS CodeDeploy with Elastic Load Balancing (p. 46), Create an Application for an In-Place Deployment (Console) (p. 211), Deployment Prerequisites (p. 245), Integrating AWS CodeDeploy with Elastic Load Balancing (p. 46), and Create an Application for an In-Place Deployment (Console) (p. 211).	September 12, 2017
Updated topics	AWS CodeDeploy support for Elastic Load Balancing load balancers in deployment groups has been expanded to include Application Load Balancers for both blue/green deployments and in-place deployments. You can now choose between an Application Load Balancer and a Classic Load Balancer for your deployment group. Load balancers are required for blue/green deployments, and optional for in-place deployments. Topics including Integrating AWS CodeDeploy with Elastic Load Balancing (p. 46), Create an Application with AWS CodeDeploy (p. 209), and Create a Deployment Group with AWS CodeDeploy (p. 220) have been updated to reflect this additional support.	August 10, 2017

Change	Description	Date Changed
New and updated topics	AWS CodeDeploy now supports using multiple tag groups to identify unions and intersections of instances to be included in a deployment group. If you use a single tag group, any instance identified by at least one tag in the group is included in the deployment group. If you use multiple tag groups, only instances that are identified by at least one tag in each of the tag groups are included. For information about the new method for adding instances to a deployment group, see Tagging Instances for AWS CodeDeploy Deployments (p. 149) . Other topics updated to reflect this support include Create an Application for an In-Place Deployment (Console) (p. 211) , Create an Application for a Blue/Green Deployment (Console) (p. 212) , Create a Deployment Group for an In-Place Deployment (Console) (p. 220) , Create a Deployment Group for a Blue/Green Deployment (Console) (p. 222) , Deployments (p. 9) , and Step 5: Create an Application and Deployment Group (p. 118) in Tutorial: Use AWS CodeDeploy to Deploy an Application from GitHub (p. 112) .	July 31, 2017
Updated topic	Two additional methods for installing the AWS CodeDeploy agent on Windows Server instances have been added to Install or reinstall the AWS CodeDeploy agent for Windows Server (p. 137) . In addition to Windows PowerShell commands, instructions are now available for downloading the installation file by using a direct HTTPS link and by using an Amazon S3 copy command. After the file is downloaded or copied to an instance, you can run the installation manually.	July 12, 2017
Updated topics	AWS CodeDeploy has improved how it manages connections to GitHub accounts and repositories. You can now create and store up to 25 connections to GitHub accounts in order to associate AWS CodeDeploy applications with GitHub repositories. Each connection can support multiple repositories. You can create connections to up to 25 different GitHub accounts, or create more than one connection to a single account. After you connect an application to a GitHub account, AWS CodeDeploy manages the required access permissions without requiring any further action from you. Updates to reflect this support have been made to Specify Information About a Revision Stored in a GitHub Repository (p. 250) , Integrating AWS CodeDeploy with GitHub (p. 53) , and Tutorial: Use AWS CodeDeploy to Deploy an Application from GitHub (p. 112) .	May 30, 2017

Change	Description	Date Changed
Updated topics	In the past, if the AWS CodeDeploy agent detected files in a target location that weren't part of the application revision from the most recent successful deployment, it would fail the current deployment by default. AWS CodeDeploy now provides options for how the agent handles these files: fail the deployment, retain the content, or overwrite the content. Create a Deployment with AWS CodeDeploy (p. 245) was updated to reflect this support, and the new section Rollback Behavior with Existing Content (p. 260) was added to Redeploy and Roll Back a Deployment with AWS CodeDeploy (p. 259) .	May 16, 2017
Updated topics	A Classic Load Balancer in Elastic Load Balancing can now be assigned to a deployment group using the AWS CodeDeploy console or AWS CLI. During an in-place deployment, a load balancer prevents internet traffic from being routed to an instance while it is being deployed to, and then makes the instance available for traffic again once the deployment to that instance completes. Several topics were updated to reflect this new support, including Integration with Other AWS Services (p. 40) , Integrating AWS CodeDeploy with Elastic Load Balancing (p. 46) , Create an Application for an In-Place Deployment (Console) (p. 211) , Create a Deployment Group for an In-Place Deployment (Console) (p. 220) , and AppSpec 'hooks' Section (p. 316) . A new section was added to the troubleshooting guide, Troubleshooting failed ApplicationStop, BeforeBlockTraffic, and AfterBlockTraffic deployment lifecycle events (p. 350) .	April 27, 2017
Updated topics	A Classic Load Balancer in Elastic Load Balancing can now be assigned to a deployment group using the AWS CodeDeploy console or AWS CLI. During an in-place deployment, a load balancer prevents internet traffic from being routed to an instance while it is being deployed to, and then makes the instance available for traffic again once the deployment to that instance completes. Several topics were updated to reflect this new support, including Integration with Other AWS Services (p. 40) , Integrating AWS CodeDeploy with Elastic Load Balancing (p. 46) , Create an Application for an In-Place Deployment (Console) (p. 211) , Create a Deployment Group for an In-Place Deployment (Console) (p. 220) , and AppSpec 'hooks' Section (p. 316) . A new section was added to the troubleshooting guide, Troubleshooting failed ApplicationStop, BeforeBlockTraffic, and AfterBlockTraffic deployment lifecycle events (p. 350) .	May 1, 2017

Change	Description	Date Changed
Updated topics	<p>AWS CodeDeploy is now available in the China (Beijing) Region.</p> <p>To use services in the China (Beijing) Region or China (Ningxia) Region, you must have an account and credentials for those regions. Accounts and credentials for other AWS regions do not work for the Beijing and Ningxia Regions, and vice versa.</p> <p>Information about some resources for the China Regions, such as AWS CodeDeploy Resource Kit bucket names and AWS CodeDeploy agent installation procedures, are not included in this edition of the <i>AWS CodeDeploy User Guide</i>.</p> <p>For more information:</p> <ul style="list-style-type: none"> • AWS CodeDeploy in <i>Getting Started with AWS in the China (Beijing) Region</i> • AWS CodeDeploy User Guide for the China Regions (English version Chinese version) 	March 29, 2017
New and updated topics	<p>Several new topics have been introduced to reflect new AWS CodeDeploy support for blue/green deployments, a deployment method in which the instances in a deployment group (the original environment) are replaced by a different set of instances (the replacement environment). Overview of a Blue/Green Deployment (p. 5) provides a high-level explanation of the blue/green methodology used by AWS CodeDeploy. Try a Sample Blue/Green Deployment in AWS CodeDeploy (p. 31) provides a guide for using the new Sample deployment wizard for blue/green deployments. Additional new topics include Create an Application for a Blue/Green Deployment (Console) (p. 212), Create a Deployment Group for a Blue/Green Deployment (Console) (p. 222), and Set Up a Load Balancer in Elastic Load Balancing for AWS CodeDeploy Deployments (p. 224).</p> <p>Numerous topics have been updated as well, including Create a Deployment with AWS CodeDeploy (p. 245), Working with Deployment Configurations in AWS CodeDeploy (p. 203), Create an Application with AWS CodeDeploy (p. 209), Working with Deployment Groups in AWS CodeDeploy (p. 219), Working with Deployments in AWS CodeDeploy (p. 244), and AppSpec 'hooks' Section (p. 316).</p>	January 25, 2017
New and updated topics	<p>A new topic, Use the register-on-premises-instance Command (IAM Session ARN) to Register an On-Premises Instance (p. 186), describes how to authenticate and register on-premises instances using periodically refreshed temporary credentials generated through AWS Security Token Service. This approach provides better support for supporting large fleets of on-premises instances than using only a static IAM user's credentials on each instance. Working with On-Premises Instances (p. 171) has also been updated to reflect this new support.</p>	December 28, 2016

Change	Description	Date Changed
Updated topics	AWS CodeDeploy is now available in the EU (London) Region (eu-west-2). Several topics, including those containing instructions for setting up the AWS CodeDeploy agent, were updated to reflect the availability of this new region.	December 13, 2016
Updated topics	AWS CodeDeploy is now available in the Canada (Central) Region (ca-central-1). Several topics, including those containing instructions for setting up the AWS CodeDeploy agent, were updated to reflect the availability of this new region.	December 8, 2016
Updated topics	AWS CodeDeploy is now available in the US East (Ohio) Region (us-east-2). Several topics, including those containing instructions for setting up the AWS CodeDeploy agent, were updated to reflect the availability of this new region.	October 17, 2016
New topics	A new section, Authentication and Access Control for AWS CodeDeploy (p. 289), provides comprehensive information about using AWS Identity and Access Management (IAM) and AWS CodeDeploy to help secure access to your resources through the use of credentials. These credentials provide the permissions required to access AWS resources, such as retrieving application revisions from Amazon S3 buckets and reading the tags on Amazon EC2 instances.	October 11, 2016
Updated topic	Update the AWS CodeDeploy Agent on Windows Server (p. 141) has been updated to reflect the availability of the new AWS CodeDeploy agent updater for Windows Server. When installed on a Windows Server instance, the updater will check periodically for new versions. When a new version is detected, the updater will uninstall the current version of the agent, if one is installed, before installing the newest version.	October 4, 2016
Updated topics	<p>AWS CodeDeploy now integrates with Amazon CloudWatch alarms, making it possible to stop a deployment if there is a change in the state of a specified alarm for a number of consecutive periods, as specified in the alarm threshold.</p> <p>AWS CodeDeploy also now supports automatically rolling back a deployment if certain conditions are met, such as a deployment failure or an activated alarm.</p> <p>A number of topics have been updated to reflect these changes, including Create an Application with AWS CodeDeploy (p. 209), Create a Deployment Group with AWS CodeDeploy (p. 220), Change Deployment Group Settings with AWS CodeDeploy (p. 227), Deployments (p. 9), Redeploy and Roll Back a Deployment with AWS CodeDeploy (p. 259), and Product and Service Integrations with AWS CodeDeploy (p. 40), along with a new topic, Monitoring Deployments with CloudWatch Alarms in AWS CodeDeploy (p. 273).</p>	September 15, 2016

Change	Description	Date Changed
New and updated topics	AWS CodeDeploy now provides integration with Amazon CloudWatch Events. You can now use CloudWatch Events to initiate one or more actions when changes are detected in the state of a deployment or the state of an instance that belongs to an AWS CodeDeploy deployment group. You can incorporate actions that invoke AWS Lambda functions; that publish to Kinesis streams or Amazon SNS topics; that push messages to Amazon SQS queues; or that, in turn, trigger CloudWatch alarm actions. For more information, see Monitoring Deployments with Amazon CloudWatch Events (p. 274).	September 9, 2016
Topic updates	The topics Integrating AWS CodeDeploy with Elastic Load Balancing (p. 46) and Integration with Other AWS Services (p. 40) have been updated to reflect an additional load balancing option. AWS CodeDeploy now supports the Classic Load Balancer and Application Load Balancer available in Elastic Load Balancing.	August 11, 2016
Topic updates	AWS CodeDeploy is now available in the Asia Pacific (Mumbai) Region (ap-south-1). Several topics, including those containing instructions for setting up the AWS CodeDeploy agent, were updated to reflect the availability of this new region.	June 27, 2016
Topic updates	<p>AWS CodeDeploy is now available in the Asia Pacific (Seoul) Region (ap-northeast-2). Several topics, including those containing instructions for setting up the AWS CodeDeploy agent, were updated to reflect the availability of this new region.</p> <p>The table of contents has been reorganized to include sections for instances, deployment configurations, applications, deployment groups, revisions, and deployments. A new section has been added for AWS CodeDeploy tutorials. For better usability, several long topics, including AWS CodeDeploy AppSpec File Reference (p. 306) and Troubleshooting AWS CodeDeploy (p. 344), have been divided into shorter topics. Configuration information for the AWS CodeDeploy agent has been moved to a new topic, AWS CodeDeploy Agent Configuration Reference (p. 330).</p>	June 15, 2016
New and updated topics	<p>Error Codes for AWS CodeDeploy (p. 361) provides information about some of the error messages that might be displayed when AWS CodeDeploy deployments fail. The following sections in Troubleshooting AWS CodeDeploy (p. 344) were updated to better assist with resolving deployment problems:</p> <ul style="list-style-type: none"> • Amazon EC2 instances in an Auto Scaling group fail to launch and receive the error "Heartbeat Timeout" (p. 359) • Avoid concurrent deployments to the same Amazon EC2 instance (p. 346) • Avoid associating multiple deployment groups with a single Auto Scaling group (p. 359) 	April 20, 2016

Change	Description	Date Changed
Topic updates	<p>AWS CodeDeploy is now available in the South America (São Paulo) Region (sa-east-1). Several topics, including those containing instructions for setting up the AWS CodeDeploy agent, were updated to reflect the availability of this new region.</p> <p>Working with the AWS CodeDeploy Agent (p. 125) was updated to reflect the new :max_revisions: configuration option, which you use to specify the number of application revisions for a deployment group that you want the AWS CodeDeploy agent to archive.</p>	March 10, 2016
New and updated topics	<p>AWS CodeDeploy now supports adding triggers to a deployment group to receive notifications about events related to deployments or instances in that deployment group. These notifications are sent to recipients who are subscribed to an Amazon Simple Notification Service topic you have made part of the trigger's action. You can also use JSON data that is created when a trigger is fired in your own customized notification workflow. For more information, see Monitoring Deployments with Amazon SNS Event Notifications (p. 278).</p> <p>Procedures were updated to reflect the redesign of the Application details page.</p> <p>The Deployments do not fail for up to an hour when an instance is terminated during a deployment (p. 355) section in Troubleshooting AWS CodeDeploy (p. 344) has been updated.</p> <p>AWS CodeDeploy Limits (p. 340) was updated to reflect revised limits for the number of deployment groups that can be associated with a single application; the allowed values for minimum healthy instances settings; and required versions of the AWS SDK for Ruby (aws-sdk-core).</p>	February 17, 2016

Change	Description	Date Changed
New and updated topics	<p>AWS CodeDeploy is now available in the US West (N. California) region (us-west-1). Several topics, including those containing instructions for setting up the AWS CodeDeploy agent, were updated to reflect the addition of this new region.</p> <p>Choose an AWS CodeDeploy Repository Type (p. 237) lists and describes the repository types now supported by AWS CodeDeploy. This new topic will be updated as support for other repository types is introduced.</p> <p>Managing AWS CodeDeploy Agent Operations (p. 132) was updated with information about the new <code>.version</code> file added to instances to report the current version of the AWS CodeDeploy agent, as well as information about supported versions of the agent.</p> <p>Syntax highlighting for code samples, including JSON and YAML examples, has been added to the user guide.</p> <p>Add an Application Specification File to a Revision for AWS CodeDeploy (p. 233) has been reorganized as step-by-step instructions.</p>	January 20, 2016
New topic	Deploy an Application in a Different AWS Account (p. 262) describes the setup requirements and process for initiating deployments that belong to another of your organization's accounts, without needing a full set of credentials for that other account. This is most useful for organizations that use multiple accounts for different purposes, such as one associated with development and test environments and another associated with the production environment.	December 30, 2015
Topic update	The Product and Service Integrations with AWS CodeDeploy (p. 40) topic has been redesigned. It now includes a section for integration examples from the community, with lists of blog posts and video examples related to AWS CodeDeploy integrations.	December 16, 2015
Topic updates	AWS CodeDeploy is now available in the Asia Pacific (Singapore) Region (ap-southeast-1). Several topics, including those containing instructions for setting up the AWS CodeDeploy agent, were updated to reflect the availability of this new region.	December 9, 2015
Topic updates	<p>Working with the AWS CodeDeploy Agent (p. 125) was updated to reflect the new <code>:proxy_uri:</code> option in the AWS CodeDeploy agent configuration file.</p> <p>AWS CodeDeploy AppSpec File Reference (p. 306) was updated with information about using a new environment variable, <code>DEPLOYMENT_GROUP_ID</code>, which hook scripts can access during a deployment lifecycle event.</p>	December 1, 2015
Topic update	Step 3: Create a Service Role for AWS CodeDeploy (p. 21) was updated to reflect a new procedure for creating a service role for AWS CodeDeploy and to incorporate other improvements.	November 13, 2015

Change	Description	Date Changed
Topic updates	<p>AWS CodeDeploy is now available in the EU (Frankfurt) Region (eu-central-1). Several topics, including those containing instructions for setting up the AWS CodeDeploy agent, were updated to reflect the availability of this new region.</p> <p>The Troubleshooting AWS CodeDeploy (p. 344) topic was updated with information about ensuring that time settings on instances are accurate.</p>	October 19, 2015
New topics	<p>AWS CloudFormation Templates for AWS CodeDeploy Reference (p. 333) was published to reflect new AWS CloudFormation support for AWS CodeDeploy actions.</p> <p>Created a Primary Components (p. 7) topic and introduced definition of a <i>target revision</i>.</p>	October 1, 2015
Topic updates	<p>Create a Deployment Group with AWS CodeDeploy (p. 220) was updated to reflect the ability to locate instances for a deployment group using wildcard searches.</p> <p>Instance Health (p. 198) was updated to clarify the concept of <i>minimum healthy instances</i>.</p>	August 31, 2015
Topic updates	<p>AWS CodeDeploy is now available in the Asia Pacific (Tokyo) Region (ap-northeast-1). Several topics, including those containing instructions for setting up the AWS CodeDeploy agent, were updated to reflect the availability of this new region.</p>	August 19, 2015
Topic updates	<p>AWS CodeDeploy now supports deployments to Red Hat Enterprise Linux (RHEL) on-premises instances and Amazon EC2 instances. For more information, see the following topics:</p> <ul style="list-style-type: none"> • Operating Systems Supported by the AWS CodeDeploy Agent (p. 125) • Working with Instances for AWS CodeDeploy (p. 147) • Tutorial: Deploy WordPress to an Amazon EC2 Instance (Amazon Linux or Red Hat Enterprise Linux and Linux, macOS, or Unix) (p. 57) • Tutorial: Deploy an Application to an On-Premises Instance with AWS CodeDeploy (Windows Server, Ubuntu Server, or Red Hat Enterprise Linux) (p. 90) 	June 23, 2015
Topic update	<p>AWS CodeDeploy now provides a set of environment variables your deployment scripts can use during deployments. These environment variables include information such as the name of the current AWS CodeDeploy application, deployment group, and deployment lifecycle event, as well as the current AWS CodeDeploy deployment identifier. For more information, see the end of the AppSpec 'hooks' Section (p. 316) section in the AWS CodeDeploy AppSpec File Reference (p. 306).</p>	May 29, 2015

Change	Description	Date Changed
Topic updates	<p>AWS CodeDeploy now provides a set of AWS managed policies in IAM that you can use instead of manually creating the equivalent policies on your own. These include:</p> <ul style="list-style-type: none"> • A policy for enabling an IAM user to register revisions with AWS CodeDeploy only and then deploy them through AWS CodeDeploy. • A policy for providing an IAM user with full access to AWS CodeDeploy resources. • A policy for providing an IAM user with read-only access to AWS CodeDeploy resources. • A policy to attach to a service role so that AWS CodeDeploy can identify Amazon EC2 instances by their Amazon EC2 tags, on-premises instance tags, or Auto Scaling group names and deploy application revisions to them accordingly. <p>For more information, see the Customer Managed Policy Examples (p. 297) section in Authentication and Access Control for AWS CodeDeploy (p. 289).</p>	May 29, 2015
Topic updates	<p>AWS CodeDeploy is now available in the EU (Ireland) Region (eu-west-1) and the Asia Pacific (Sydney) Region (ap-southeast-2). Several topics, including those containing instructions for setting up the AWS CodeDeploy agent, were updated to reflect the availability of these new regions.</p>	May 7, 2015
New topics	<p>AWS CodeDeploy now supports deployments to on-premises instance and Amazon EC2 instances. The following topics were added to describe this new support:</p> <ul style="list-style-type: none"> • Working with On-Premises Instances (p. 171) • Tutorial: Deploy an Application to an On-Premises Instance with AWS CodeDeploy (Windows Server, Ubuntu Server, or Red Hat Enterprise Linux) (p. 90) • Working with On-Premises Instances (p. 171) 	April 2, 2015
New topic	<p>AWS CodeDeploy Resources (p. 364) was added.</p>	April 2, 2015

Change	Description	Date Changed
Topic update	<p>Troubleshooting AWS CodeDeploy (p. 344) was updated:</p> <ul style="list-style-type: none">• A new Long-running processes can cause deployments to fail (p. 352) section describes steps you can take to identify and address deployment failures due to long-running processes.• The General Auto Scaling troubleshooting (p. 357) section was updated to show that AWS CodeDeploy has increased its Auto Scaling timeout logic for the AWS CodeDeploy agent from five minutes to one hour.• A new Mismatched Auto Scaling lifecycle hooks might cause automatic deployments to Auto Scaling groups to stop or fail (p. 360) section describes steps you can take to identify and address failed automatic deployments to Auto Scaling groups.	April 2, 2015
Topic updates	<p>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:</p> <ul style="list-style-type: none">• Configure an Amazon EC2 Instance to Work with AWS CodeDeploy (p. 168)• Step 4: Create an IAM Instance Profile for Your Amazon EC2 Instances (p. 25)• Step 3: Create a Service Role for AWS CodeDeploy (p. 21)• Authentication and Access Control for AWS CodeDeploy (p. 289) <p>Two sections were added to Troubleshooting AWS CodeDeploy (p. 344):</p> <ul style="list-style-type: none">• General Troubleshooting Checklist (p. 344)• Windows PowerShell scripts fail to use the 64-bit version of Windows PowerShell by default (p. 352) <p>The AppSpec 'hooks' Section (p. 316) section in the AWS CodeDeploy AppSpec File Reference (p. 306) was updated to more accurately describe the available deployment lifecycle events.</p>	February 12, 2015
Topic updates	<p>A new section was added to Troubleshooting AWS CodeDeploy (p. 344): Amazon EC2 instances in an Auto Scaling group fail to launch and receive the error "Heartbeat Timeout" (p. 359).</p> <p>A CloudBees section was added to Product and Service Integrations with AWS CodeDeploy (p. 40).</p>	January 28, 2015

Change	Description	Date Changed
Topic updates	<p>The following sections were added to Troubleshooting AWS CodeDeploy (p. 344):</p> <ul style="list-style-type: none"> • Using some text editors to create AppSpec files and shell scripts can cause deployments to fail (p. 347) • Using Finder in macOS to bundle an application revision can cause deployments to fail (p. 347) • Troubleshooting failed ApplicationStop, BeforeBlockTraffic, and AfterBlockTraffic deployment lifecycle events (p. 350) • Troubleshooting a failed DownloadBundle deployment lifecycle event with "UnknownError: not opened for reading" (p. 351) • General Auto Scaling troubleshooting (p. 357) <p>Information was added to the Step 5: Try the AWS CodeDeploy Sample Deployment Wizard (p. 29) to clarify that certain permissions are required for the calling IAM user, specifically:</p> <ul style="list-style-type: none"> • Step 3: Configure instances (p. 36) notes that certain permissions are required to use the walkthrough's AWS CloudFormation template. • Step 7: Select a service role (p. 37) notes that certain permissions are required to create a service role as part of the walkthrough. • Step 9: Review deployment details (p. 38) notes that certain permissions are required to create applications and deployment groups and to deploy applications. <p>For information about the required permissions, see Prerequisites (p. 30).</p>	January 20, 2015
New topics	<p>The Product and Service Integrations with AWS CodeDeploy (p. 40) section was updated to include the following topics:</p> <ul style="list-style-type: none"> • Integrating AWS CodeDeploy with Auto Scaling (p. 44) • Tutorial: Use AWS CodeDeploy to Deploy an Application to an Auto Scaling Group (p. 96) • Monitoring Deployments with AWS CloudTrail (p. 277) • Integrating AWS CodeDeploy with Elastic Load Balancing (p. 46) • Integrating AWS CodeDeploy with GitHub (p. 53) • Tutorial: Use AWS CodeDeploy to Deploy an Application from GitHub (p. 112) 	January 9, 2015

Change	Description	Date Changed
Topic updates	<ul style="list-style-type: none">The Automatically Deploy from GitHub with AWS CodeDeploy (p. 55) section was added to Integrating AWS CodeDeploy with GitHub (p. 53). You can now automatically trigger a deployment from a GitHub repository whenever the source code in that repository is changed.The Troubleshoot Auto Scaling Issues (p. 357) section was added to Troubleshooting AWS CodeDeploy (p. 344). This new section describes how to troubleshoot common issues with deploying to Auto Scaling groups.The new subsection "files Examples" was added to the AppSpec 'files' Section (EC2/On-Premises Deployments Only) (p. 309) section of AWS CodeDeploy AppSpec File Reference (p. 306). This new subsection includes several examples of how to use the <code>files</code> section of an AppSpec file to instruct AWS CodeDeploy to copy specific files or folders to specific locations on an Amazon EC2 instance during a deployment.	January 8, 2015
New topic	Monitoring Deployments with AWS CloudTrail (p. 277) 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 3: Configure instances (p. 36) section in Step 5: Try the AWS CodeDeploy Sample Deployment Wizard (p. 29) was updated.	December 3, 2014
Initial public release	This is the initial public release of the <i>AWS CodeDeploy User Guide</i> .	November 12, 2014

AWS Glossary

For the latest AWS terminology, see the [AWS Glossary](#) in the *AWS General Reference*.