User Guide

AWS CodeDeploy

API Version 2014-10-06

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS CodeDeploy: User Guide

Copyright © 2026 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 CodeDeploy? ... 1
Benefits of AWS CodeDeploy ... 2
Overview of CodeDeploy compute platforms ... 3
Overview of CodeDeploy deployment types ... 10
Overview of an in-place deployment ... 12
Overview of a blue/green deployment ... 13
We want to hear from you ... 17
Primary components ... 17
Application ... 18
Compute platform ... 18
Deployment configuration ... 19
Deployment group ... 20
Deployment type ... 20
Deployment type ... 21
Revision ... 21
Service role ... 22
Target revision ... 22
Other components ... 22
Deployments ... 23
Deployments on an AWS Lambda Compute Platform ... 23
Deployments on an Amazon ECS Compute Platform ... 26
Deployments on an EC2/On-Premises Compute Platform ... 38
Application specification files ... 45
AppSpec files on an Amazon ECS Compute Platform ... 45
AppSpec files on an AWS Lambda compute platform ... 46
AppSpec files on an EC2/on-premises compute platform ... 46
How the CodeDeploy agent uses the AppSpec file ... 46
Getting started ... 48
Step 1: Setting up ... 48
Sign up for an AWS account ... 48
Create a user with administrative access ... 49
Grant programmatic access ... 50
Step 2: Create a service role ... 53
Create a service role (console) ... 55

Create a service role (CLI) ... 57
Get the service role ARN (console) ... 60
Get the service role ARN (CLI) ... 61
Step 3: Limit the CodeDeploy user's permissions ... 61
Step 4: Create an IAM instance profile ... 65
Create an IAM instance profile for your Amazon EC2 instances (CLI) ... 66
Create an IAM instance profile for your Amazon EC2 instances (console) ... 70
Product and service integrations ... 73
Integration with other AWS services ... 73
Amazon EC2 Auto Scaling ... 81
Elastic Load Balancing ... 89
Integration with partner products and services ... 93
GitHub ... 99
Integration examples from the community ... 103
Blog posts ... 103
Tutorials ... 105
Tutorial: Deploy WordPress to a non-Windows instance ... 105
Step 1: Launch an Amazon EC2 instance ... 106
Step 2: Configure your source content ... 109
Step 3: Upload your application to Amazon S3 ... 114
Step 4: Deploy your application ... 119
Step 5: Update and redeploy your application ... 125
Step 6: Clean up ... 128
Tutorial: Deploy a Hello World application to a Windows Server instance ... 132
Step 1: Launch an Amazon EC2 instance ... 133
Step 2: Configure your source content ... 135
Step 3: Upload your application to Amazon S3 ... 138
Step 4: Deploy your application ... 143
Step 5: Update and redeploy your application ... 148
Step 6: Clean up ... 151
Tutorial: Deploy an application to an on-premises instance ... 155
Prerequisites ... 155
Step 1: Configure the on-premises instance ... 156
Step 2: Create a sample application revision ... 156
Step 3: Bundle and upload your application revision to Amazon S3 ... 161
Step 4: Deploy your application revision ... 161

Step 5: Verify your deployment ... 162
Step 6: Clean up resources ... 162
Tutorial: Deploy to an Auto Scaling group ... 164
Prerequisites ... 165
Step 1: Create and configure the Auto Scaling group ... 165
Step 2: Deploy the application to the Auto Scaling group ... 171
Step 3: Check your results ... 182
Step 4: Increase the number of Amazon EC2 instances in the Auto Scaling group ... 183
Step 5: Check your results again ... 185
Step 6: Clean up ... 187
Tutorial: Deploy an application from GitHub ... 189
Prerequisites ... 190
Step 1: Set up a GitHub account ... 190
Step 2: Create a GitHub repository ... 190
Step 3: Upload a sample application to your GitHub repository ... 193
Step 4: Provision an instance ... 198
Step 5: Create an application and deployment group ... 198
Step 6: Deploy the application to the instance ... 200
Step 7: Monitor and verify the deployment ... 204
Step 8: Clean up ... 206
Tutorial: Deploy an application into Amazon ECS ... 207
Prerequisites ... 209
Step 1: Update your Amazon ECS application ... 210
Step 2: Create the AppSpec file ... 211
Step 3: Use the CodeDeploy console to deploy your application ... 212
Step 4: Clean up ... 217
Tutorial: Deploy an Amazon ECS service with a validation test ... 217
Prerequisites ... 220
Step 1: Create a test listener ... 220
Step 2: Update your Amazon ECS application ... 220
Step 3: Create a lifecycle hook Lambda function ... 221
Step 4: Update your AppSpec file ... 223
Step 5: Use the CodeDeploy console to deploy your Amazon ECS service ... 225
Step 6: View your Lambda hook function output in CloudWatch Logs ... 227
Step 7: Clean up ... 228
Tutorial: Deploy a Lambda function using AWS SAM ... 229

Prerequisites ... 230
Step 1: Set up your infrastructure ... 230
Step 2: Update the Lambda function ... 245
Step 3: Deploy the updated Lambda function ... 248
Step 4: View your deployment results ... 250
Step 5: Clean up ... 253
Working with the CodeDeploy agent ... 254
Operating systems supported by the CodeDeploy agent ... 255
Supported Amazon EC2 AMI operating systems ... 255
Supported on-premises operating systems ... 255
Communication protocol and port for the CodeDeploy agent ... 255
Version history of the CodeDeploy agent ... 256
Managing the CodeDeploy process ... 270
Application revision and log file cleanup ... 270
Files installed by the CodeDeploy agent ... 270
Managing CodeDeploy agent operations ... 274
Verify the CodeDeploy agent is running ... 274
Determine the version of the CodeDeploy agent ... 276
Install the CodeDeploy agent ... 278
Update the CodeDeploy agent ... 289
Uninstall the CodeDeploy agent ... 293
Send CodeDeploy agent logs to CloudWatch ... 294
Working with instances ... 299
Comparing Amazon EC2 instances to on-premises instances ... 299
Instance tasks for CodeDeploy ... 301
Tagging instances for CodeDeploy deployments ... 302
Example 1: Single tag group, single tag ... 303
Example 2: Single tag group, multiple tags ... 304
Example 3: Multiple tag groups, single tags ... 306
Example 4: Multiple tag groups, multiple tags ... 308
Working with Amazon EC2 instances ... 312
Create an Amazon EC2 instance for CodeDeploy ... 312
Create an Amazon EC2 instance (CloudFormation template) ... 319
Configure an Amazon EC2 instance ... 329
Working with on-premises instances ... 333
Prerequisites for configuring an on-premises instance ... 334

Register an on-premises instance ... 336
Managing on-premises instances operations ... 371
View instance details ... 379
View instance details (console) ... 379
View instance details (CLI) ... 380
Instance health ... 380
Health status ... 381
About the minimum number of healthy instances ... 382
About the minimum number of healthy instances per Availability Zone ... 387
Working with deployment configurations ... 389
Deployment configurations on an EC2/on-premises compute platform ... 389
Predefined deployment configurations ... 389
Deployment configurations on an Amazon ECS compute platform ... 393
Predefined deployment configurations for Amazon ECS ... 393
Deployment configurations for CloudFormation blue/green deployments (Amazon ECS) ... 394
Deployment configurations on an AWS Lambda compute platform ... 395
Predefined deployment configurations for Lambda ... 395
Topics ... 396
Create a deployment configuration ... 396
Create a deployment configuration (console) ... 397
Create a deployment configuration (AWS CLI) ... 399
View deployment configuration details ... 400
View deployment configuration details (console) ... 400
View deployment configuration (CLI) ... 401
Delete a deployment configuration ... 401
Working with applications ... 403
Create an application ... 404
Create an application for an in-place deployment (console) ... 406
Create an application for a blue/green deployment (console) ... 409
Create an application for an Amazon ECS service deployment (console) ... 414
Create an application for an AWS Lambda function deployment (console) ... 416
Create an application (CLI) ... 418
View application details ... 418
View application details (console) ... 418
View application details (CLI) ... 419
Create a notification rule ... 419

Rename an application ... 422
Delete an application ... 422
Delete an application (console) ... 423
Delete an application (AWS CLI) ... 423
Working with deployment groups ... 424
Deployment groups in Amazon ECS compute platform deployments ... 424
Deployment groups in AWS Lambda compute platform deployments ... 424
Deployment groups in EC2/On-Premises Compute Platform deployments ... 424
Topics ... 425
Create a deployment group ... 426
Create a deployment group for an in-place deployment (console) ... 426
Create a deployment group for an EC2/On-Premises blue/green deployment (console) ... 430
Create a deployment group for an Amazon ECS deployment (console) ... 436
Set up a load balancer in Elastic Load Balancing for CodeDeploy Amazon EC2 deployments ... 437
Set up a load balancer, target groups, and listeners for CodeDeploy Amazon ECS deployments ... 439
Create a deployment group (CLI) ... 443
View deployment group details ... 445
View deployment group details (console) ... 445
View deployment group details (CLI) ... 446
Change deployment group settings ... 446
Change deployment group settings (console) ... 447
Change deployment group settings (CLI) ... 448
Configure advanced options for a deployment group ... 449
Delete a deployment group ... 452
Delete a deployment group (console) ... 452
Delete a deployment group (CLI) ... 453
Working with application revisions ... 454
Plan a revision ... 454
Add an AppSpec File ... 455
Add an AppSpec file for an Amazon ECS deployment ... 455
Add an AppSpec file for an AWS Lambda deployment ... 458
Add an AppSpec file for an EC2/On-Premises deployment ... 460
Choose a repository type ... 464
Push a revision ... 467

Push a revision using the AWS CLI ... 469
View application revision details ... 471
View application revision details (console) ... 472
View application revision details (CLI) ... 472
Register an application revision ... 473
Register a revision in Amazon S3 with CodeDeploy (CLI) ... 474
Register a revision in GitHub with CodeDeploy (CLI) ... 475
Working with deployments ... 476
Create a deployment ... 477
Deployment prerequisites ... 478
Create an Amazon ECS Compute Platform deployment (console) ... 481
Create an AWS Lambda Compute Platform deployment (console) ... 482
Create an EC2/On-Premises Compute Platform deployment (console) ... 484
Create an Amazon ECS Compute Platform deployment (CLI) ... 489
Create an AWS Lambda Compute Platform deployment (CLI) ... 490
Create an EC2/On-Premises Compute Platform deployment (CLI) ... 491
Create an Amazon ECS blue/green deployment through CloudFormation ... 495
View deployment details ... 499
View deployment details (console) ... 499
View deployment details (CLI) ... 500
View deployment log data ... 501
View log file data in the Amazon CloudWatch console ... 501
View log files on an instance ... 501
Stop a deployment ... 504
Stop a deployment (console) ... 505
Stop a deployment (CLI) ... 506
Redeploy and roll back a deployment ... 506
Automatic rollbacks ... 506
Manual rollbacks ... 507
Rollback and redeployment workflow ... 507
Rollback behavior with existing content ... 508
Deploy an application in a different AWS account ... 512
Step 1: Create an S3 bucket in either account ... 512
Step 2: Grant Amazon S3 bucket permissions to the production account's IAM instance profile ... 513
Step 3: Create resources and a cross-account role in the production account ... 514

Step 4: Upload the application revision to Amazon S3 bucket ... 515
Step 5: Assume the cross-account role and deploy applications ... 515
Validate a deployment package on a local machine ... 516
Prerequisites ... 517
Create a local deployment ... 519
Examples ... 522
Monitoring deployments ... 524
Automated tools ... 524
Manual tools ... 526
Monitoring deployments with Amazon CloudWatch tools ... 527
Monitoring deployments with CloudWatch alarms ... 527
Monitoring deployments with Amazon CloudWatch Events ... 529
Monitoring deployments with AWS CloudTrail ... 531
CodeDeploy information in CloudTrail ... 532
Understanding CodeDeploy log file entries ... 532
Monitoring deployments with Amazon SNS event notifications ... 534
Grant Amazon SNS permissions to a service role ... 535
Create a trigger for a CodeDeploy event ... 536
Edit a trigger in a deployment group ... 543
Delete a trigger from a deployment group ... 545
JSON data formats for triggers ... 546
Security ... 549
Data protection ... 549
Internetwork traffic privacy ... 550
Encryption at rest ... 551
Encryption in transit ... 551
Encryption key management ... 551
Identity and access management ... 552
Audience ... 552
Authenticating with identities ... 552
Managing access using policies ... 553
How AWS CodeDeploy works with IAM ... 555
AWS managed (predefined) policies for CodeDeploy ... 559
CodeDeploy updates to AWS managed policies ... 568
Identity-based policy examples ... 571
Troubleshooting ... 579

CodeDeploy permissions reference ... 580
Cross-service confused deputy prevention ... 589
Incident response ... 591
Auditing of all interactions with CodeDeploy ... 591
Alerting and incident management ... 592
Compliance validation ... 593
Resilience ... 593
Infrastructure security ... 593
Reference ... 595
AppSpec file reference ... 595
AppSpec files on an Amazon ECS compute platform ... 596
AppSpec files on an AWS Lambda compute platform ... 596
AppSpec files on an EC2/on-premises compute platform ... 596
AppSpec File structure ... 597
AppSpec File example ... 642
AppSpec File spacing ... 648
Validate your AppSpec File and file location ... 649
Agent configuration reference ... 650
Related topics ... 654
CloudFormation template reference ... 655
Use CodeDeploy with Amazon Virtual Private Cloud ... 658
Availability ... 659
Create VPC endpoints for CodeDeploy ... 662
Configure the CodeDeploy agent and IAM permissions ... 662
Resource kit reference ... 663
Resource kit bucket names by Region ... 663
Resource kit contents ... 665
Display a list of the resource kit files ... 667
Download the resource kit files ... 669
Quotas ... 671
Troubleshooting ... 678
General troubleshooting issues ... 678
General troubleshooting checklist ... 679
CodeDeploy deployment resources are supported in only in some AWS Regions ... 680
Procedures in this guide do not match the CodeDeploy console ... 681
Required IAM roles are not available ... 681

Using some text editors to create AppSpec files and shell scripts can cause deployments to fail ... 681
Using Finder in macOS to bundle an application revision can cause deployments to fail ... 682
Troubleshoot EC2/On-Premises deployment issues ... 682
CodeDeploy plugin CommandPoller missing credentials error ... 683
Deployment fails with the message “Validation of PKCS7 signed message failed” ... 684
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" ... 684
Long file paths cause "No such file or directory" errors ... 686
Long-running processes can cause deployments to fail ... 687
Troubleshooting a failed AllowTraffic lifecycle event with no error reported in the deployment logs ... 689
Troubleshooting a failed ApplicationStop, BeforeBlockTraffic, or AfterBlockTraffic deployment lifecycle event ... 689
Troubleshooting a failed DownloadBundle deployment lifecycle event with UnknownError: not opened for reading ... 691
Troubleshooting all lifecycle events skipped errors ... 692
Windows PowerShell scripts fail to use the 64-bit version of Windows PowerShell by default ... 693
Troubleshoot Amazon ECS deployment issues ... 694
A timeout occurs while waiting for replacement task set ... 695
A timeout occurs while waiting for a notification to continue ... 695
The IAM role does not have enough permissions ... 696
The deployment timed out while waiting for a status callback ... 697
The deployment failed because one or more of the lifecycle event validation functions failed ... 697
The ELB could not be updated due to the following error: Primary taskset target group must be behind listener ... 698
My deployment sometimes fails when using Auto Scaling ... 699
Only ALB supports gradual traffic routing, use AllAtOnce Traffic routing instead when you create/update Deployment group ... 699
Even though my deployment succeeded, the replacement task set fails the Elastic Load Balancing health checks, and my application is down ... 700
Can I attach multiple load balancers to a deployment group? ... 701
Can I perform CodeDeploy blue/green deployments without a load balancer? ... 701
How can I update my Amazon ECS service with new information during a deployment? ... 702

Troubleshoot AWS Lambda deployment issues ... 702
AWS Lambda deployments fail after manually stopping a Lambda deployment that does not have configured rollbacks ... 702
Troubleshoot deployment group issues ... 703
Tagging an instance as part of a deployment group does not automatically deploy your application to the new instance ... 703
Troubleshoot instance issues ... 703
Tags must be set correctly ... 704
AWS CodeDeploy agent must be installed and running on instances ... 704
Deployments do not fail for up to an hour when an instance is terminated during a deployment ... 704
Analyzing log files to investigate deployment failures on instances ... 705
Create a new CodeDeploy log file if it was accidentally deleted ... 705
Troubleshooting “InvalidSignatureException – Signature expired: [time] is now earlier than [time]” deployment errors ... 705
Troubleshoot GitHub token issues ... 706
Invalid GitHub OAuth token ... 706
Maximum number of GitHub OAuth tokens exceeded ... 706
Troubleshoot Amazon EC2 Auto Scaling issues ... 707
General Amazon EC2 Auto Scaling troubleshooting ... 707
"CodeDeployRole does not give you permission to perform operations in the following AWS service: AmazonAutoScaling" error ... 708
Instances in an Amazon EC2 Auto Scaling group are continuously provisioned and terminated before a revision can be deployed ... 709
Terminating or rebooting an Amazon EC2 Auto Scaling instance might cause deployments to fail ... 709
Avoid associating multiple deployment groups with a single Amazon EC2 Auto Scaling group ... 710
EC2 instances in an Amazon EC2 Auto Scaling group fail to launch and receive the error "Heartbeat Timeout" ... 711
Mismatched Amazon EC2 Auto Scaling lifecycle hooks might cause automatic deployments to Amazon EC2 Auto Scaling groups to stop or fail ... 714
"The deployment failed because no instances were found for your deployment group" error ... 715
Error codes ... 723
Related topics ... 728

Resources **729**
Reference guides and support resources 729
Samples 729
Blogs 729
AWS software development kits and tools 729
Document history **731**
Earlier updates 749
AWS Glossary **771**

What is CodeDeploy?

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

You can deploy a nearly unlimited variety of application content, including:

- Code
- Serverless AWS Lambda functions
- Web and configuration files
- Executables
- Packages
- Scripts
- Multimedia files

CodeDeploy can deploy application content that runs on a server and is stored in Amazon S3 buckets, GitHub repositories, or Bitbucket repositories. CodeDeploy can also deploy a serverless Lambda function. You do not need to make changes to your existing code before you can use CodeDeploy.

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.

CodeDeploy works with various systems for configuration management, source control, continuous integration, continuous delivery, and continuous deployment. For more information, see Product integrations.

The CodeDeploy console also provides a way to quickly search for your resources, such as repositories, build projects, deployment applications, and pipelines. Choose **Go to resource** or press

the / key, and then type the name of the resource. Any matches appear in the list. Searches are case insensitive. You only see resources that you have permissions to view. For more information, see Identity and access management for AWS CodeDeploy.

Topics

- Benefits of AWS CodeDeploy
- Overview of CodeDeploy compute platforms
- Overview of CodeDeploy deployment types
- We want to hear from you
- CodeDeploy primary components
- CodeDeploy deployments
- CodeDeploy application specification (AppSpec) files

Benefits of AWS CodeDeploy

CodeDeploy offers these benefits:

- **Server, serverless, and container applications**. CodeDeploy lets you deploy both traditional applications on servers and applications that deploy a serverless AWS Lambda function version or an Amazon ECS application.
- **Automated deployments**. CodeDeploy fully automates your application deployments across your development, test, and production environments. CodeDeploy scales with your infrastructure so that you can deploy to one instance or thousands.
- **Minimize downtime**. If your application uses the EC2/On-Premises compute platform, CodeDeploy helps maximize your application availability. During an in-place deployment, 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. 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, 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 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**. CodeDeploy is platform-agnostic and works with any application. You can easily reuse your setup code. CodeDeploy can also integrate with your software release process or continuous delivery toolchain.
- **Concurrent deployments**. If you have more than one application that uses the EC2/On-Premises compute platform, CodeDeploy can deploy them concurrently to the same set of instances.

Overview of CodeDeploy compute platforms

CodeDeploy is able to deploy applications to three 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 CodeDeploy deployment types.
- **AWS Lambda**: Used to deploy applications that consist of an updated version of a Lambda function. AWS Lambda manages the Lambda function 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.

  You can manage the way in which traffic is shifted to the updated Lambda function versions during a deployment by choosing a canary, linear, or all-at-once configuration.
- **Amazon ECS**: Used to deploy an Amazon ECS containerized application as a task set. CodeDeploy performs a blue/green deployment by installing an updated version of the application as a new replacement task set. CodeDeploy reroutes production traffic from the original application task set to the replacement task set. The original task set is terminated after a successful deployment. For more information about Amazon ECS, see Amazon Elastic Container Service.

  You can manage the way in which traffic is shifted to the updated task set during a deployment by choosing a canary, linear, or all-at-once configuration.

> **Note**
>
> Amazon ECS blue/green deployments are supported using both CodeDeploy and CloudFormation. Details for these deployments are described in subsequent sections.

The following table describes how CodeDeploy components are used with each compute platform. For more information, see:

- Working with deployment groups in CodeDeploy
- Working with deployments in CodeDeploy
- Working with deployment configurations in CodeDeploy
- Working with application revisions for CodeDeploy
- Working with applications in CodeDeploy

CodeDeploy component	EC2/On-Premises	AWS Lambda	Amazon ECS
Deployment group	Deploys a revision to a set of instances.	Deploys a new version of a serverless Lambda function on a high-avai lability compute infrastru cture.	Specifies the Amazon ECS service with the container ized applicati on to deploy as a task set, a productio n and optional test listener used to

CodeDeploy component	EC2/On-Premises	AWS Lambda	Amazon ECS
			serve traffic to the deployed application, when to reroute traffic and terminate the deployed application's original task set, and optional trigger, alarm, and rollback settings.

CodeDeploy component	EC2/On-Premises	AWS Lambda	Amazon ECS
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.	Shifts production traffic from one version of a Lambda function to a new version of the same function. The AppSpec file specifies which Lambda function version to deploy.	Deploys an updated version of an Amazon ECS containerized application as a new, replacement task set. CodeDeploy reroutes production traffic from the task set with the original version to the new replacement task set with the updated version. When the deployment completes, the original task set is

CodeDeploy component	EC2/On-Premises	AWS Lambda	Amazon ECS
			terminated.
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.	Settings that determine how traffic is shifted to the updated Amazon ECS task set.

CodeDeploy component	EC2/On-Premises	AWS Lambda	Amazon ECS
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 function to deploy and Lambda functions that can run validation tests during deployment lifecycle event hooks.	An AppSpec file that specifies: • The Amazon ECS task definition for the Amazon ECS service with the containerized application to deploy. • The container where your updated application is deployed. • A port for the container where production traffic

CodeDeploy component	EC2/On-Premises	AWS Lambda	Amazon ECS
			is rerouted. • Optional network configuration settings and Lambda functions that can run validation tests during deployment lifecycle event hooks.

CodeDeploy component	EC2/On-Premises	AWS Lambda	Amazon ECS
Application	A collection of deployment groups and revisions. An EC2/On-Premises application uses the EC2/On-Premises compute platform.	A collection of deployment groups and revisions. An application used for an AWS Lambda deployment uses the serverless AWS Lambda compute platform.	A collection of deployment groups and revisions. An application used for an Amazon ECS deployment uses the Amazon ECS compute platform.

Overview of CodeDeploy deployment types

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.

> **Note**
>
> AWS Lambda and Amazon ECS 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 one or more Elastic Load Balancing load balancers, 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**
    >
    > If you use an EC2/On-Premises compute platform, be aware that blue/green deployments work with Amazon EC2 instances only.

  - **Blue/green on an AWS Lambda or Amazon ECS compute platform**: Traffic is shifted in increments according to a **canary**, **linear**, or **all-at-once** deployment configuration.
  - **Blue/green deployments through CloudFormation**: Traffic is shifted from your current resources to your updated resources as part of an CloudFormation stack update. Currently, only ECS blue/green deployments are supported.

  For more information about blue/green deployments, see Overview of a blue/green deployment.

> **Note**
>
> Using the CodeDeploy agent, you can perform a deployment on an instance you are signed in to without the need for an application, deployment group, or even an AWS account. For information, see Use the CodeDeploy agent to validate a deployment package on a local machine.

Topics

- Overview of an in-place deployment

- Overview of a blue/green deployment

Overview of an in-place deployment

> **Note**
>
> AWS Lambda and Amazon ECS deployments cannot use an in-place deployment type.

Here's how an in-place deployment 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 CodeDeploy. It defines the deployment actions you want 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 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. CodeDeploy calls a set of Amazon EC2 instances a *deployment group*. A deployment group contains individually tagged Amazon EC2 instances, Amazon EC2 instances in Amazon EC2 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 is pulled for automatic deployments.
3. Next, the CodeDeploy agent on each instance polls CodeDeploy to determine what and when to pull from the specified Amazon S3 bucket or GitHub repository.
4. Finally, the CodeDeploy agent on each instance pulls the target revision from the Amazon S3 bucket or GitHub repository and, using the instructions in the AppSpec file, deploys the contents to the instance.

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 is used to update your applications while minimizing interruptions caused by the changes of a new application version. CodeDeploy provisions your new application version alongside the old version before rerouting your production traffic.

- **AWS Lambda**: Traffic is shifted from one version of a Lambda function to a new version of the same Lambda function.
- **Amazon ECS**: Traffic is shifted from a task set in your Amazon ECS service to an updated, replacement task set in the same Amazon ECS service.
- **EC2/On-Premises**: Traffic is shifted from one set of instances in the original environment to a replacement set of instances.

All AWS Lambda and Amazon ECS deployments are blue/green. An EC2/On-Premises deployment can be in-place or blue/green. A blue/green deployment offers a number of advantages over an in-place deployment:

- You can install and test an application in the new replacement environment and deploy it to production simply by rerouting traffic.
- If you're using the EC2/On-Premises compute platform, switching back to the most recent version of an application is faster and more reliable. That's 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 version to your new AWS Lambda function version.
- If you're using the Amazon ECS compute platform, you control how traffic is shifted from your original task set to your new task set.

A blue/green deployment with CloudFormation can use one of the following methods:

- **CloudFormation templates for deployments**: When you configure deployments with CloudFormation templates, your deployments are triggered by CloudFormation updates. When you change a resource and upload a template change, a stack update in CloudFormation initiates

the new deployment. For a list of resources you can use in CloudFormation templates, see CloudFormation templates for CodeDeploy reference.

- **Blue/green deployments through CloudFormation**: You can use CloudFormation to manage your blue/green deployments through stack updates. You define both your blue and green resources, in addition to specifying the traffic routing and stabilization settings, within the stack template. Then, if you update selected resources during a stack update, CloudFormation generates all the necessary green resources, shifts the traffic based on the specified traffic routing parameters, and deletes the blue resources. For more information, see Automate Amazon ECS blue/green deployments through CodeDeploy using CloudFormation in the *AWS CloudFormation User Guide*.

> **Note**
>
> Supported for Amazon ECS blue/green deployments only.

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

Blue/Green deployment on an AWS Lambda or Amazon ECS compute platform

If you're using the AWS Lambda or Amazon ECS compute platform, you must indicate how traffic is shifted from the original AWS Lambda function or Amazon ECS task set to the new function or task set. To indicate how traffic is shifted, you must specify one of the following deployment configurations:

- **canary**
- **linear**
- **all-at-once**

For information on how traffic is shifted in a canary, linear, or all-at-once deployment configurations, see Deployment configuration.

For details on the Lambda deployment configuration, see Deployment configurations on an AWS Lambda compute platform.

For details on the Amazon ECS deployment configuration, see Deployment configurations on an Amazon ECS compute platform.

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 Amazon EC2 Auto Scaling group. The instances must meet these additional requirements:

- Each Amazon EC2 instance must have the correct IAM instance profile attached.
- The 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 Amazon EC2 Auto Scaling group: During the blue/green deployment, CodeDeploy creates the instances for your replacement environment during the deployment. With this option, CodeDeploy uses the Amazon EC2 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, Amazon EC2 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 Amazon EC2 Auto Scaling group that serves as your original environment. The first time you run a blue/green deployment, you typically use instances that were already used in an in-place deployment.
2. In an existing 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 or load balancers that 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 Amazon EC2 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 starts being 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.

Blue/Green deployment through CloudFormation

You can manage CodeDeploy blue/green deployments by modeling your resources with an CloudFormation template.

When you model your blue/green resources using an CloudFormation template, you create a stack update in CloudFormation that updates your task set. Production traffic shifts from your service's original task set to a replacement task set either all at once, with linear deployments and bake

times, or with canary deployments. The stack update initiates a deployment in CodeDeploy. You can view the deployment status and history in CodeDeploy, but you do not otherwise create or manage CodeDeploy resources outside of the CloudFormation template.

> **Note**
>
> For blue/green deployments through CloudFormation, you don't create a CodeDeploy application or deployment group.

This method supports Amazon ECS blue/green deployments only. For more information about blue/green deployments through CloudFormation, see Create an Amazon ECS blue/green deployment through CloudFormation.

We want to hear from you

We welcome your feedback. To contact us, visit the CodeDeploy forum.

Topics

- Primary Components
- Deployments
- Application Specification Files

CodeDeploy primary components

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

Topics

- Application
- Compute platform
- Deployment configuration
- Deployment group
- Deployment type

- IAM instance profile
- Revision
- Service role
- Target revision
- Other components

Application

An *application* is a name that uniquely identifies the application you want to deploy. 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

A *compute platform* is a platform on which CodeDeploy deploys an application. There are three 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 CodeDeploy deployment types.
- **AWS Lambda**: Used to deploy applications that consist of an updated version of a Lambda function. AWS Lambda manages the Lambda function 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.

  You can manage the way in which traffic is shifted to the updated Lambda function versions during a deployment by choosing a canary, linear, or all-at-once configuration.
- **Amazon ECS**: Used to deploy an Amazon ECS containerized application as a task set. CodeDeploy performs a blue/green deployment by installing an updated version of the application as a new replacement task set. CodeDeploy reroutes production traffic from the original application task set to the replacement task set. The original task set is terminated after a successful deployment. For more information about Amazon ECS, see Amazon Elastic Container Service.

You can manage the way in which traffic is shifted to the updated task set during a deployment by choosing a canary, linear, or all-at-once configuration.

> **Note**
>
> Amazon ECS blue/green deployments are supported through both CodeDeploy and CloudFormation. Details for these deployments are described in subsequent sections.

Deployment configuration

A *deployment configuration* is a set of deployment rules and deployment success and failure conditions used by 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 or the Amazon ECS compute platform, you can specify how traffic is routed to your updated Lambda function or ECS task set.

For more information about specifying the minimum number of healthy hosts for a deployment that uses the EC2/On-Premises compute platform, see About the minimum number of healthy instances.

The following deployment configurations specify how traffic is routed during a deployment that uses the Lambda or the ECS 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 or ECS task set 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 or ECS task set to the updated function or task set all at once.

Deployment group

A *deployment group* is a set of individual instances. A deployment group contains individually tagged instances, Amazon EC2 instances in Amazon EC2 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. For information about Amazon EC2 Auto Scaling, see Integrating CodeDeploy with Amazon EC2 Auto Scaling.

Deployment type

A *deployment type* is a method used to make the latest application revision available on instances in a deployment group. There are two deployment types:

- **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.
- **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 one or more Elastic Load Balancing load balancers, 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**
>
> If you use an EC2/On-Premises compute platform, be aware that blue/green deployments work with Amazon EC2 instances only.

- **Blue/green on an AWS Lambda or Amazon ECS compute platform**: Traffic is shifted in increments according to a **canary**, **linear**, or **all-at-once** deployment configuration.
- **Blue/green deployments through CloudFormation**: Traffic is shifted from your current resources to your updated resources as part of an CloudFormation stack update. Currently, only ECS blue/green deployments are supported.

For more information about blue/green deployments, see Overview of a blue/green deployment.

> **Note**
>
> Amazon ECS blue/green deployments are supported using both CodeDeploy and CloudFormation. Details for these deployments are described in subsequent sections.

IAM instance profile

An *IAM instance profile* is 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 are stored. For more information, see Step 4: Create an IAM instance profile for your Amazon EC2 instances.

Revision

A *revision* is a version of your application. An AWS Lambda deployment revision is a YAML- 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, webpages, executable files, and deployment scripts) and 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

A *service role* is 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 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 CodeDeploy to identify instances to which it can deploy applications.
- To perform operations on instances, Amazon EC2 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 to set up alarm monitoring for deployments.

For more information, see Step 2: Create a service role for CodeDeploy.

Target revision

A *target revision* is 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. This is also the revision that is pulled for automatic deployments.

Other components

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

- Choose a CodeDeploy repository type
- Deployments
- Application Specification Files
- Instance Health
- Working with the CodeDeploy agent
- Working with On-Premises Instances

CodeDeploy deployments

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

Topics

- Deployments on an AWS Lambda Compute Platform
- Deployments on an Amazon ECS Compute Platform
- Deployments on an EC2/On-Premises Compute Platform

Deployments on an AWS Lambda Compute Platform

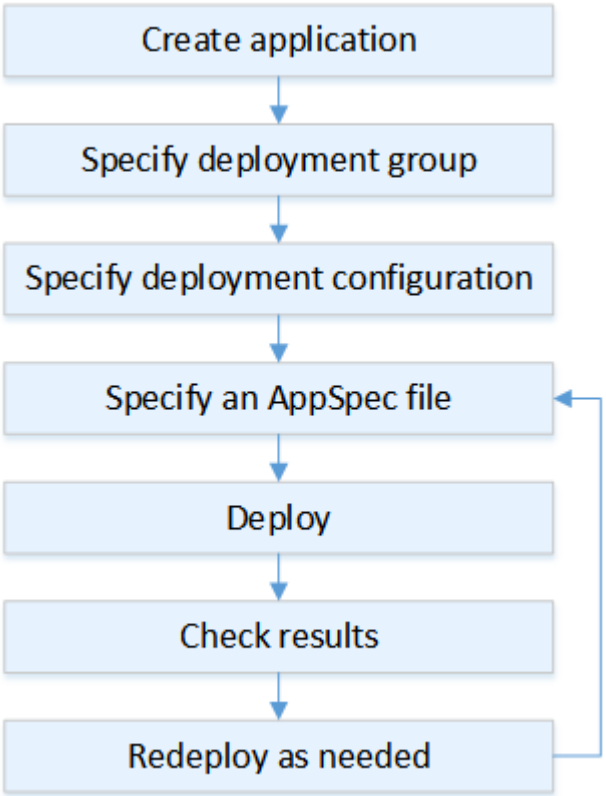
This topic provides information about the components and workflow of CodeDeploy deployments that use the AWS Lambda compute platform.

Topics

- Deployment workflow on an AWS Lambda compute platform
- Uploading your application revision
- Creating your application and deployment groups
- Deploying your application revision
- Updating your application
- Stopped and failed deployments
- Redeployments and deployment rollbacks

Deployment workflow on an AWS Lambda compute platform

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



These steps include:

1. Create an application and give it a name that uniquely identifies the application revisions you want to deploy. To deploy Lambda functions, choose AWS Lambda compute platform when you create your application. 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 CodeDeploy.
2. Set up a deployment group by specifying your deployment group's name.
3. Choose 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.
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 CodeDeploy.

5. Deploy 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 CodeDeploy.
6. Check the deployment results. For more information, see Monitoring deployments in CodeDeploy.

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.

Creating your application and deployment groups

A 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 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 CodeDeploy console or call the create-deployment command to push a revision.

Stopped and failed deployments

You can use the 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 CodeDeploy console or analyze the log file data from the failed deployment. For more information, see Application revision and log file cleanup and View log data for CodeDeploy EC2/On-Premises deployments.

Redeployments and deployment rollbacks

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 CodeDeploy console shows which ones are the result of an automatic deployment.

For more information, see Redeploy and roll back a deployment with CodeDeploy.

Deployments on an Amazon ECS Compute Platform

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

Topics

- Before you begin an Amazon ECS deployment

- Deployment workflow (high level) on an Amazon ECS compute platform
- What happens during an Amazon ECS deployment
- Uploading your application revision
- Creating your application and deployment groups
- Deploying your application revision
- Updating your application
- Stopped and failed deployments
- Redeployments and deployment rollbacks
- Amazon ECS blue/green deployments through AWS CloudFormation

Before you begin an Amazon ECS deployment

Before you begin an Amazon ECS application deployment, you must have the following ready. Some requirements are specified when you create your deployment group, and some are specified in the AppSpec file.

Requirement	Where specified
Amazon ECS cluster	Deployment group
Amazon ECS service	Deployment group
Application Load Balancer or Network Load Balancer	Deployment group
Production listener	Deployment group
Test listener (optional)	Deployment group
Two target groups	Deployment group
Amazon ECS task definition	AppSpec file
Container name	AppSpec file
Container port	AppSpec file

Amazon ECS cluster

An Amazon ECS *cluster* is a logical grouping of tasks or services. You specify the Amazon ECS cluster that contains your Amazon ECS service when you create your CodeDeploy application's deployment group. For more information, see Amazon ECS clusters in the *Amazon Elastic Container Service User Guide*.

Amazon ECS service

An Amazon ECS *service* maintains and runs specified instances of a task definition in an Amazon ECS cluster. Your Amazon ECS service must be enabled for CodeDeploy. By default, an Amazon ECS service is enabled for Amazon ECS deployments. When you create your deployment group, you choose to deploy an Amazon ECS service that is in your Amazon ECS cluster. For more information, see Amazon ECS services in the *Amazon Elastic Container Service User Guide*.

Application Load Balancer or Network Load Balancer

You must use Elastic Load Balancing with the Amazon ECS service you want to update with an Amazon ECS deployment. You can use an Application Load Balancer or a Network Load Balancer. We recommend an Application Load Balancer so you can take advantage of features such as dynamic port mapping and path-based routing and priority rules. You specify the load balancer when you create your CodeDeploy application's deployment group. For more information, see Set up a load balancer, target groups, and listeners for CodeDeploy Amazon ECS deployments and Creating a load balancer in the *Amazon Elastic Container Service User Guide*.

One or two listeners

A *listener* is used by your load balancer to direct traffic to your target groups. One production listener is required. You can specify an optional second test listener that directs traffic to your replacement task set while you run validation tests. You specify one or both listeners when you create your deployment group. If you use the Amazon ECS console to create your Amazon ECS service, your listeners are created for you. For more information, see Listeners for your application load balancers in the *Elastic Load Balancing User Guide* and Creating a service in the *Amazon Elastic Container Service User Guide*.

Two Amazon ECS target groups

A *target group* is used to route traffic to a registered target. An Amazon ECS deployment requires two target groups: one for your Amazon ECS application's original task set and one for its replacement task set. During deployment, CodeDeploy creates a replacement task set and

reroutes traffic from the original task set to the new one. You specify the target groups when you create your CodeDeploy application's deployment group.

During a deployment, CodeDeploy determines which target group is associated with the task set in your Amazon ECS service that has the status `PRIMARY` (this is the original task set) and associates one target group with it, and then associates the other target group with the replacement task set. If you do another deployment, the target group associated with the current deployment's original task set is associated with the next deployment's replacement task set. For more information, see Target groups for your application load balancers in the *Elastic Load Balancing User Guide*.

An Amazon ECS task definition

A *task definition* is required to run the Docker container that contains your Amazon ECS application. You specify the ARN of your task definition in your CodeDeploy application's AppSpec file. For more information, see Amazon ECS task definitions in the *Amazon Elastic Container Service User Guide* and AppSpec 'resources' section for Amazon ECS deployments.

A container for your Amazon ECS application

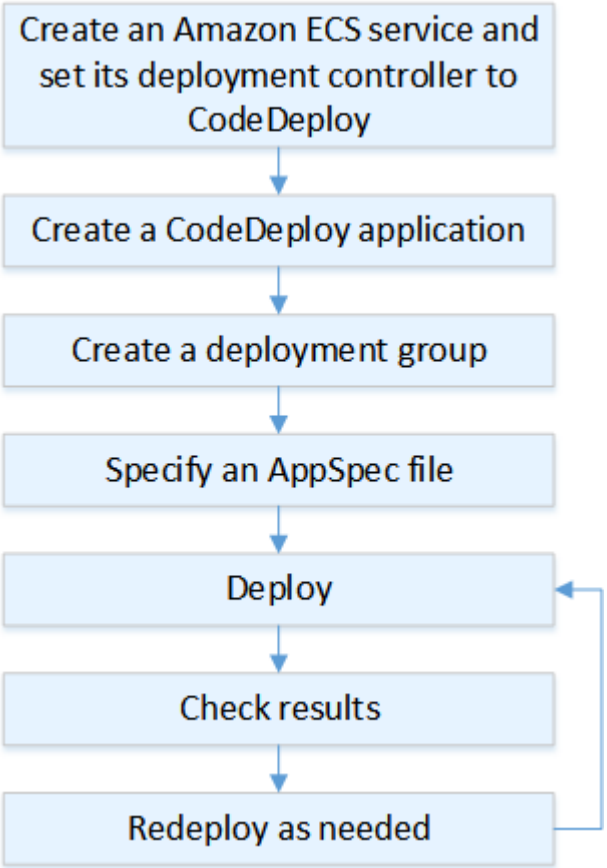
A Docker *container* is a unit of software that packages up code and its dependencies so your application can run. A container isolates your application so it runs in different computing environments. Your load balancer directs traffic to a container in your Amazon ECS application's task set. You specify the name of your container in your CodeDeploy application's AppSpec file. The container specified in your AppSpec file must be one of the containers specified in your Amazon ECS task definition. For more information, see What is Amazon Elastic Container Service? in the *Amazon Elastic Container Service User Guide* and AppSpec 'resources' section for Amazon ECS deployments.

A port for your replacement task set

During your Amazon ECS deployment, your load balancer directs traffic to this *port* on the container specified in your CodeDeploy application's AppSpec file. You specify the port in your CodeDeploy application's AppSpec file. For more information, see AppSpec 'resources' section for Amazon ECS deployments.

Deployment workflow (high level) on an Amazon ECS compute platform

The following diagram shows the primary steps in the deployment of updated Amazon ECS services.



These steps include:

1. Create an AWS CodeDeploy application by specifying a name that uniquely represents what you want to deploy. To deploy an Amazon ECS application, in your AWS CodeDeploy application, choose the Amazon ECS compute platform. CodeDeploy uses an application during a deployment to reference the correct deployment components, such as the deployment group, target groups, listeners, and traffic rerouting behavior, and application revision. For more information, see Create an application with CodeDeploy.
2. Set up a deployment group by specifying:
   - The deployment group name.
   - Your Amazon ECS cluster and service name. The Amazon ECS service's deployment controller must be set to CodeDeploy.
   - The production listener, an optional test listener, and target groups used during a deployment.
   - Deployment settings, such as when to reroute production traffic to the replacement Amazon ECS task set in your Amazon ECS service and when to terminate the original Amazon ECS task set in your Amazon ECS service.

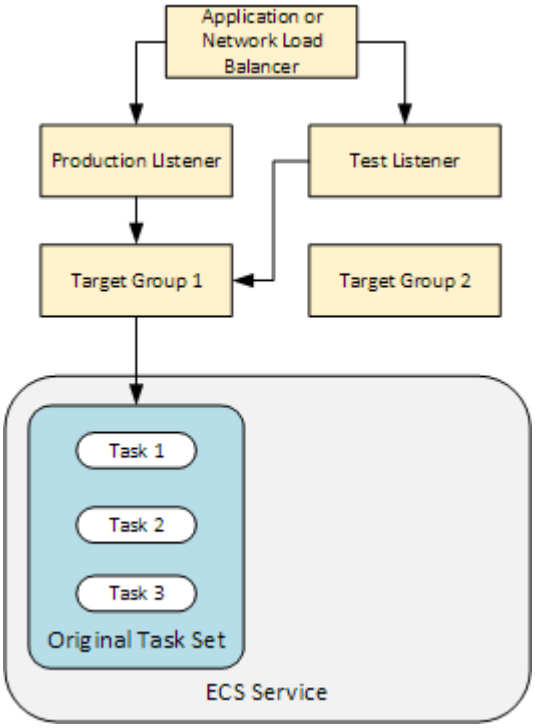
- Optional settings, such as triggers, alarms, and rollback behavior.

3. Specify an *application specification file* (AppSpec file). You can upload it to Amazon S3, enter it into the console in YAML or JSON format, or specify it with the AWS CLI or SDK. The AppSpec file specifies an Amazon ECS task definition for the deployment, a container name and port mapping used to route traffic, and Lambda functions run after deployment lifecycle hooks. The container name must be a container in your Amazon ECS task definition. For more information, see Working with application revisions for CodeDeploy.
4. Deploy your application revision. AWS CodeDeploy reroutes traffic from the original version of a task set in your Amazon ECS service to a new, replacement task set. Target groups specified in the deployment group are used to serve traffic to the original and replacement task sets. After the deployment is complete, the original task set is terminated. You can specify an optional test listener to serve test traffic to your replacement version before traffic is rerouted to it. For more information, see Create a deployment with CodeDeploy.
5. Check the deployment results. For more information, see Monitoring deployments in CodeDeploy.

What happens during an Amazon ECS deployment

Before an Amazon ECS deployment with a test listener starts, you must configure its components. For more information, see Before you begin an Amazon ECS deployment.

The following diagram shows the relationship between these components when an Amazon ECS deployment is ready to start.



When the deployment starts, the deployment lifecycle events start to execute one at a time. Some lifecycle events are hooks that only execute Lambda functions specified in the AppSpec file. The deployment lifecycle events in the following table are listed in the order they execute. For more information, see AppSpec 'hooks' section for an Amazon ECS deployment.

Lifecycle event	Lifecycle event action
`BeforeInstall` (a hook for Lambda functions)	Run Lambda functions.
Install	Set up the replacement task set.
`AfterInstall` (a hook for Lambda functions)	Run Lambda functions.
AllowTestTraffic	Route traffic from the test listener to target group 2.
`AfterAllowTestTraffic` (a hook for Lambda functions)	Run Lambda functions.

Lifecycle event	Lifecycle event action
`BeforeAllowTraffic` (a hook for Lambda functions)	Run Lambda functions.
AllowTraffic	Route traffic from the production listener to target group 2.
`AfterAllowTraffic`	Run Lambda functions.

> **Note**
>
> Lambda functions in a hook are optional.

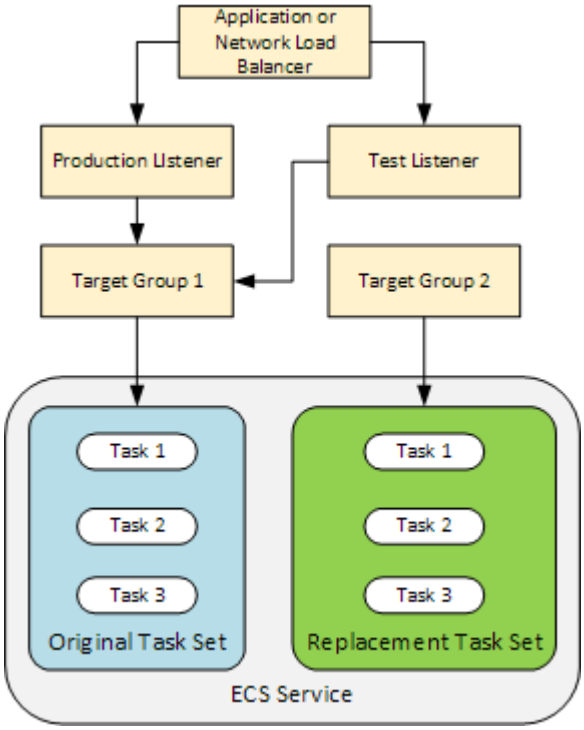
1. Execute any Lambda functions specified in the `BeforeInstall` hook in the AppSpec file.
2. During the `Install` lifecycle event:

   a. A replacement task set is created in your Amazon ECS service.

   b. The updated containerized application is installed into the replacement task set.

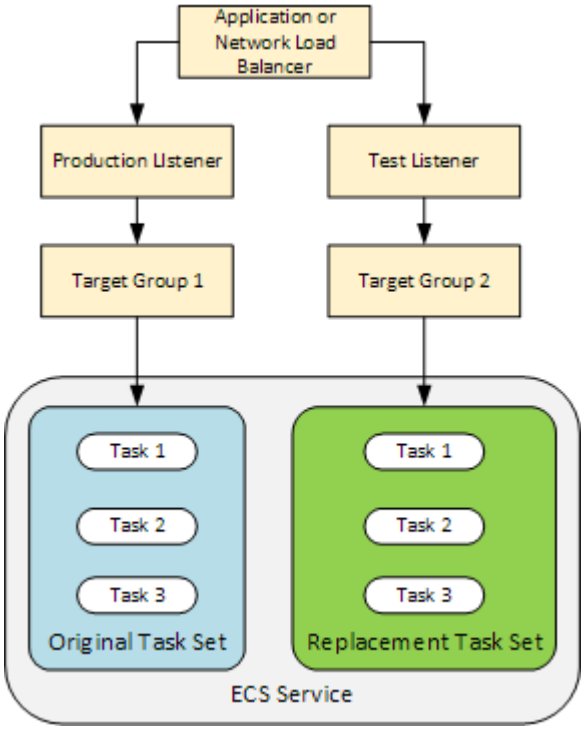
   c. The second target group is associated with the replacement task set.

   This diagram shows deployment components with the new replacement task set. The containerized application is inside this task set. The task set is composed of three tasks. (An application can have any number of tasks.) The second target group is now associated with the replacement task set.



3. Execute any Lambda functions specified in the `AfterInstall` hook in the AppSpec file.

4. The `AllowTestTraffic` event is invoked. During this lifecycle event, the test listener routes traffic to the updated containerized application.



5. Execute any Lambda functions specified in the `AfterAllowTestTraffic` hook in the AppSpec file. Lambda functions can validate the deployment using the test traffic. For example, a Lambda function can serve traffic to the test listener and track metrics from the replacement task set. If rollbacks are configured, you can configure a CloudWatch alarm that triggers a rollback when the validation test in your Lambda function fails.

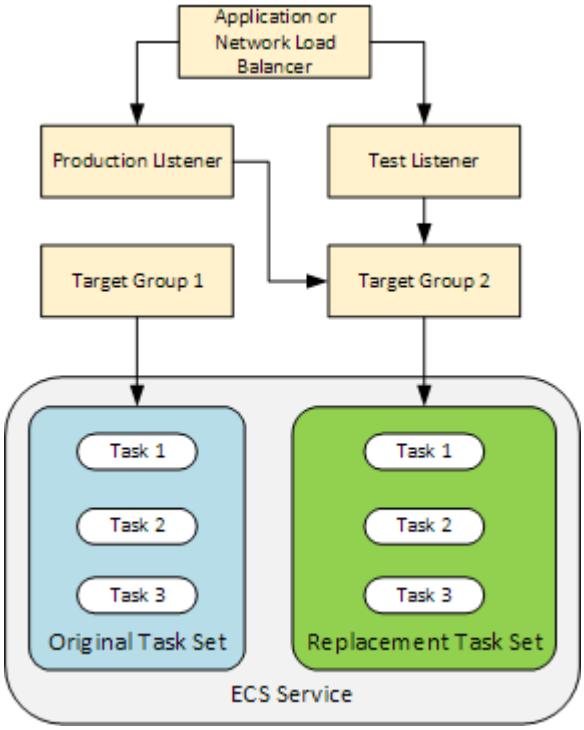
   After the validation tests are complete, one of the following occurs:

   - If validation fails and rollbacks are configured, the deployment status is marked `Failed` and components return to their state when the deployment started.
   - If validation fails and rollbacks are not configured, the deployment status is marked `Failed` and components remain in their current state.
   - If validation succeeds, the deployment continues to the `BeforeAllowTraffic` hook.

   For more information, see Monitoring deployments with CloudWatch alarms in CodeDeploy, Automatic rollbacks, and Configure advanced options for a deployment group.

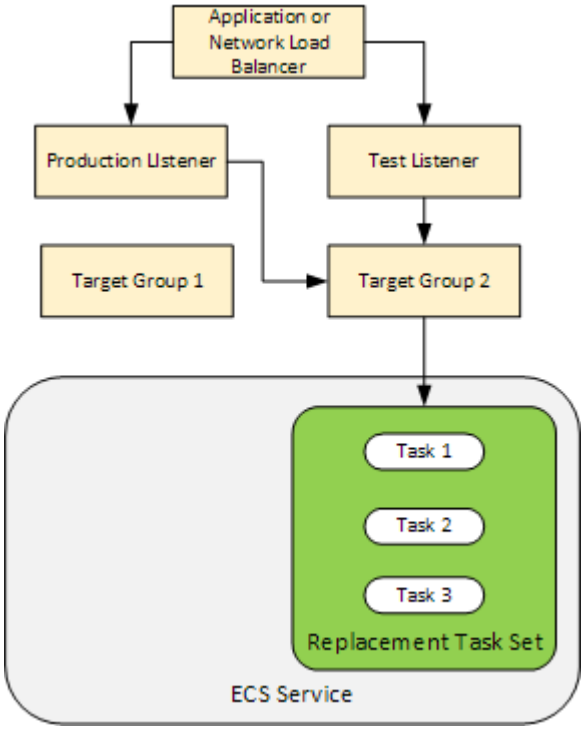
6. Execute any Lambda functions specified in the `BeforeAllowTraffic` hook in the AppSpec file.

7. The `AllowTraffic` event is invoked. Production traffic is rerouted from the original task set to the replacement task set. The following diagram shows the replacement task set receiving production traffic.



8. Execute any Lambda functions specified in the `AfterAllowTraffic` hook in the AppSpec file.

9. After all events succeed, the deployment status is set to `Succeeded` and the original task set is removed.



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.

Creating your application and deployment groups

A CodeDeploy deployment group on an Amazon ECS compute platform identifies listeners to serve traffic to your updated Amazon ECS application and two target groups used during your deployment. A deployment group also defines a set of configuration options, such as alarms and rollback configurations.

Deploying your application revision

Now you're ready to deploy the updated Amazon ECS service specified in your deployment group. You can use the CodeDeploy console or the create-deployment command. There are parameters you can specify to control your deployment, including the revision and deployment group.

Updating your application

You can make updates to your application and then use the CodeDeploy console or call the create-deployment command to push a revision.

Stopped and failed deployments

You can use the 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 Error information and Common errors in the AWS CodeDeploy API Reference.

Redeployments and deployment rollbacks

CodeDeploy implements rollbacks by rerouting traffic from the replacement task set to the original task set.

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 CodeDeploy console displays a list of deployments that are the result of an automatic deployment.

If you redeploy, the target group associated with the current deployment's original task set is associated with the redeployment's replacement task set.

For more information, see Redeploy and roll back a deployment with CodeDeploy.

Amazon ECS blue/green deployments through AWS CloudFormation

You can use AWS CloudFormation to manage Amazon ECS blue/green deployments through CodeDeploy. For more information, see Create an Amazon ECS blue/green deployment through CloudFormation.

> **Note**
>
> Managing Amazon ECS blue/green deployments with CloudFormation is not available in the Asia Pacific (Osaka) region.

Deployments on an EC2/On-Premises Compute Platform

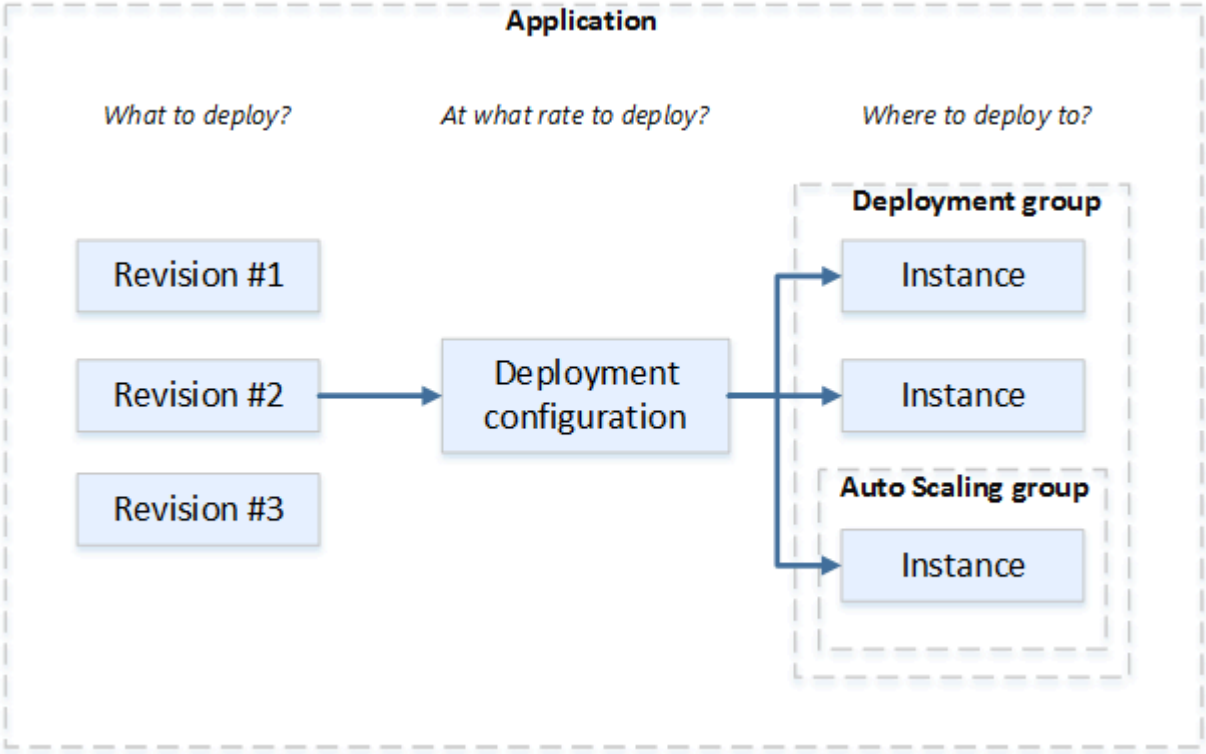
This topic provides information about the components and workflow of CodeDeploy deployments that use the EC2/On-Premises compute platform. For information about blue/green deployments, see Overview of a blue/green deployment.

Topics

- Deployment components on an EC2/on-premises compute platform
- Deployment workflow on an EC2/on-premises compute platform
- Setting up instances
- Uploading your application revision
- Creating your application and deployment groups
- Deploying your application revision
- Updating your application
- Stopped and failed deployments
- Redeployments and deployment rollbacks

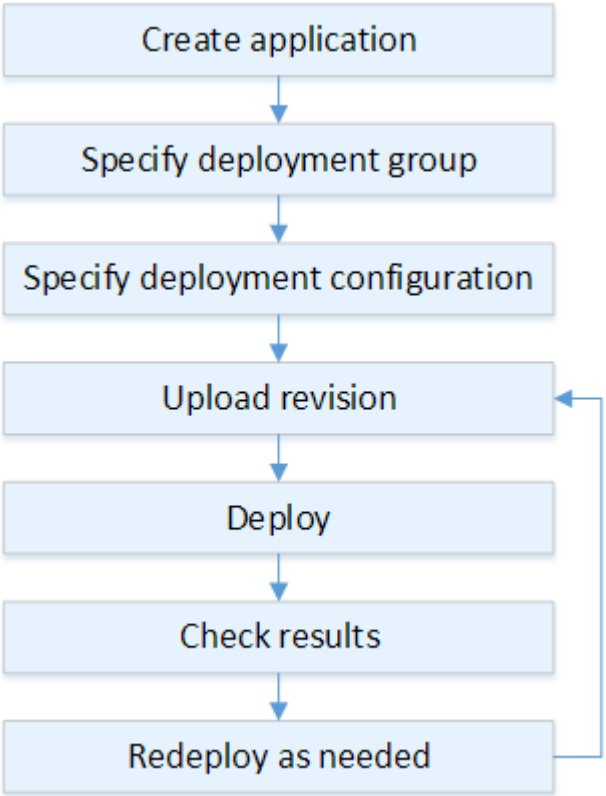
Deployment components on an EC2/on-premises compute platform

The following diagram shows the components in a 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. Create an application and give it a name that uniquely identifies the application revisions you want to deploy and the compute platform for your application. 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 CodeDeploy.
2. Set 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 Amazon EC2 Auto Scaling group names, or both.

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

In all cases, the instances must be configured to be used in a deployment (that is, they must be tagged or belong to an Amazon EC2 Auto Scaling group) and have the CodeDeploy agent installed and running.

We provide you with an 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 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 CodeDeploy.

You can also specify the following options:

- **Amazon SNS notifications**. Create triggers that 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.
- **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. Specify a deployment configuration to indicate to how many instances your application revisions should be simultaneously deployed and to describe the success and failure conditions for the deployment. For more information, see View Deployment Configuration Details.
4. Upload 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 CodeDeploy.
5. Deploy your application revision to the deployment group. The CodeDeploy agent on each instance in the deployment group copies your application revision from Amazon S3 or GitHub to the instance. The 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 CodeDeploy.
6. Check the deployment results. For more information, see Monitoring deployments in CodeDeploy.

7. Redeploy 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, 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. Then execute a new deployment to the same deployment group with the new revision. For more information, see Create a deployment with CodeDeploy.

Setting up instances

You must 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 launch or use five instances.

To manually provision instances:

1. Install the CodeDeploy agent on the instances. The 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. CodeDeploy relies on tags to identify and group instances into 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 for the CodeDeploy agent to verify the identity of the instance.
4. Create a service role. Provide service access so that CodeDeploy can expand the tags in your AWS account.

For an initial deployment, the 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 CodeDeploy agent already installed. For more information, see Working with instances for CodeDeploy.

> **Note**
>
> For a blue/green deployment, you can choose between using instances that you already have for the replacement environment or letting 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.

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

A CodeDeploy deployment group identifies a collection of instances based on their tags, Amazon EC2 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 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 Deployments.

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 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 CodeDeploy console or call the create-deployment command to push a revision.

Stopped and failed deployments

You can use the 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. 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 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 and View log data for CodeDeploy EC2/On-Premises deployments.

Redeployments and deployment rollbacks

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 CodeDeploy console shows which ones are the result of an automatic deployment.

For more information, see Redeploy and roll back a deployment with CodeDeploy.

CodeDeploy application specification (AppSpec) files

An application specification file (AppSpec file), which is unique to 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 CodeDeploy AppSpec file reference.

Topics

- AppSpec files on an Amazon ECS Compute Platform
- AppSpec files on an AWS Lambda compute platform
- AppSpec files on an EC2/on-premises compute platform
- How the CodeDeploy agent uses the AppSpec file

AppSpec files on an Amazon ECS Compute Platform

If your application uses the Amazon ECS 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 name of the Amazon ECS service and the container name and port used to direct traffic to the new task set.
- 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 Amazon ECS deployment, AppSpec file structure for Amazon ECS deployments, and AppSpec File example for an Amazon ECS deployment.

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.

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. 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.

How the CodeDeploy agent uses the AppSpec file

During deployment, the 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 CodeDeploy agent moves on to the next step. If the event is found, the 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 CodeDeploy agent log file on the instance.

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

> **Note**
>
> The CodeDeploy agent is not used in an AWS Lambda or an Amazon ECS deployment.

During the **Install** event, the 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 CodeDeploy agent installed on the operating system doesn't match what's listed in the AppSpec file, the deployment fails.

For information about CodeDeploy agent log files, see Working with the CodeDeploy agent.

Getting started with CodeDeploy

Topics

- Step 1: Setting up
- Step 2: Create a service role for CodeDeploy
- Step 3: Limit the CodeDeploy user's permissions
- Step 4: Create an IAM instance profile for your Amazon EC2 instances

Step 1: Setting up

Before you use AWS CodeDeploy for the first time, you must complete setup steps. The steps involve creating an AWS account (if you don't already have one), and an administrative user with programmatic access.

In this guide, the administrative user is called the **CodeDeploy administrative user**.

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call or text message and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

   For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.
2. Assign users to a group, and then assign single sign-on access to the group.

   For instructions, see Add groups in the *AWS IAM Identity Center User Guide*.

You have now created and signed in as the **CodeDeploy administrative user**.

Grant programmatic access

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.

To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
IAM	(Recommended) Use console credentials as temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Login for AWS local development in the *AWS Command Line Interface User Guide*. • For AWS SDKs, see Login for AWS local development in the *AWS SDKs and Tools Reference Guide*.
Workforce identity	Use temporary credentials to sign programmatic requests	Following the instructions for the interface that you want to use.

Which user needs programmatic access?	To	By
(Users managed in IAM Identity Center)	to the AWS CLI, AWS SDKs, or AWS APIs.	• For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*. • For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the *AWS SDKs and Tools Reference Guide*.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the *IAM User Guide*.

Which user needs programmatic access?	To	By
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Authenticating using IAM user credentials in the *AWS Command Line Interface User Guide*. • For AWS SDKs and tools, see Authenticate using long-term credentials in the *AWS SDKs and Tools Reference Guide*. • For AWS APIs, see Managing access keys for IAM users in the *IAM User Guide*.

Important

We strongly recommend you configure the CodeDeploy adminstrative user as a workforce identity (a user managed in IAM Identity Center) with the AWS CLI. Many of the procedures in this guide assume you're using the AWS CLI to perform configurations.

Important

If you configure the AWS CLI, you may be prompted to specify an AWS Region. Choose one of the supported Regions listed in Region and endpoints in the *AWS General Reference*.

Step 2: Create a service role for 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 resources the service can access and what it can do with those resources.

The service role you create for CodeDeploy must be granted the permissions required for your compute platform. If you deploy to more than one compute platform, create one service role for each. To add permissions, attach one or more of the following AWS supplied policies:

For EC2/On-Premises deployments, attach the **`AWSCodeDeployRole`** policy. It provides the permissions for your service role to:

- Read the tags on your instances or identify your Amazon EC2 instances by Amazon EC2 Auto Scaling group names.
- Read, create, update, and delete Amazon EC2 Auto Scaling groups, lifecycle hooks, and scaling policies.
- Publish information to Amazon SNS topics.
- Retrieve information about CloudWatch alarms.
- Read and update Elastic Load Balancing.

> **Note**
>
> If you create your Auto Scaling group with a launch template, you must add the following permissions:
>
> - `ec2:RunInstances`
> - `ec2:CreateTags`
> - `iam:PassRole`
>
> For more information, see Step 2: Create a service role, Creating a launch template for an Auto Scaling group, and Launch template support in the *Amazon EC2 Auto Scaling User Guide*.

For Amazon ECS deployments, if you want full access to support services, attach the **`AWSCodeDeployRoleForECS`** policy. It provides the permissions for your service role to:

- Read, update, and delete Amazon ECS task sets.

- Update Elastic Load Balancing target groups, listeners, and rules.
- Invoke AWS Lambda functions.
- Access revision files in Amazon S3 buckets.
- Retrieve information about CloudWatch alarms.
- Publish information to Amazon SNS topics.

For Amazon ECS deployments, if you want limited access to support services, attach the **`AWSCodeDeployRoleForECSLimited`** policy. It provides the permissions for your service role to:

- Read, update, and delete Amazon ECS task sets.
- Retrieve information about CloudWatch alarms.
- Publish information to Amazon SNS topics.

For AWS Lambda deployments, if you want to allow publishing to Amazon SNS, attach the **`AWSCodeDeployRoleForLambda`** policy. It provides the permissions for your service role to:

- Read, update, and invoke AWS Lambda functions and aliases.
- Access revision files in Amazon S3 buckets.
- Retrieve information about CloudWatch alarms.
- Publish information to Amazon SNS topics.

For AWS Lambda deployments, if you want to limit access to Amazon SNS, attach the **`AWSCodeDeployRoleForLambdaLimited`** policy. It provides the permissions for your service role to:

- Read, update, and invoke AWS Lambda functions and aliases.
- Access revision files in Amazon S3 buckets.
- Retrieve information about CloudWatch alarms.

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)
- Create a service role (CLI)
- Get the service role ARN (console)
- Get the service role ARN (CLI)

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. Choose **AWS service**, and under **Use case**, from the drop-down list, choose **CodeDeploy**.
4. Choose your use case:
   - For EC2/On-Premises deployments, choose **CodeDeploy**.
   - For AWS Lambda deployments, choose **CodeDeploy for Lambda**.
   - For Amazon ECS deployments, choose **CodeDeploy - ECS**.
5. Choose **Next**.
6. On the **Add permissions** page, the correct permissions policy for the use case is displayed. Choose **Next**.
7. On the **Name, review, and create** page, in **Role name**, enter a name for the service role (for example, **`CodeDeployServiceRole`**), and then choose **Create role**.

   You can also enter a description for this service role in **Role description**.
8. If you want this service role to have permission to access all currently supported endpoints, you are finished with this procedure.

   To restrict this service role from access to some endpoints, continue with the remaining steps in this procedure.
9. In the list of roles, search for and choose the role you just created (`CodeDeployServiceRole`).
10. Choose the **Trust relationships** tab.
11. Choose **Edit trust policy**.

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 trust policy text box with the following policy. Remove the lines for the endpoints you want to prevent access to, and then choose **Update policy**.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [

                    "codedeploy.us-east-1.amazonaws.com",
                    "codedeploy.us-east-2.amazonaws.com",
                    "codedeploy.us-west-1.amazonaws.com",
                    "codedeploy.us-west-2.amazonaws.com",
                    "codedeploy.ca-central-1.amazonaws.com",
                    "codedeploy.ap-east-1.amazonaws.com",
                    "codedeploy.ap-northeast-1.amazonaws.com",
                    "codedeploy.ap-northeast-2.amazonaws.com",
                    "codedeploy.ap-northeast-3.amazonaws.com",
```

```
                    "codedeploy.ap-southeast-1.amazonaws.com",
                    "codedeploy.ap-southeast-2.amazonaws.com",
                    "codedeploy.ap-southeast-3.amazonaws.com",
                    "codedeploy.ap-southeast-4.amazonaws.com",
                    "codedeploy.ap-south-1.amazonaws.com",
                    "codedeploy.ap-south-2.amazonaws.com",
                    "codedeploy.ca-central-1.amazonaws.com",
                    "codedeploy.eu-west-1.amazonaws.com",
                    "codedeploy.eu-west-2.amazonaws.com",
                    "codedeploy.eu-west-3.amazonaws.com",
                    "codedeploy.eu-central-1.amazonaws.com",
                    "codedeploy.eu-central-2.amazonaws.com",
                    "codedeploy.eu-north-1.amazonaws.com",
                    "codedeploy.eu-south-1.amazonaws.com",
                    "codedeploy.eu-south-2.amazonaws.com",
                    "codedeploy.il-central-1.amazonaws.com",
                    "codedeploy.me-central-1.amazonaws.com",
                    "codedeploy.me-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 CodeDeploy to work on your behalf.

   Do one of the following:

   - To grant access to all supported AWS 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-1.amazonaws.com",
                    "codedeploy.us-east-2.amazonaws.com",
                    "codedeploy.us-west-1.amazonaws.com",
                    "codedeploy.us-west-2.amazonaws.com",
                    "codedeploy.ca-central-1.amazonaws.com",
                    "codedeploy.ap-east-1.amazonaws.com",
                    "codedeploy.ap-northeast-1.amazonaws.com",
                    "codedeploy.ap-northeast-2.amazonaws.com",
                    "codedeploy.ap-northeast-3.amazonaws.com",
                    "codedeploy.ap-southeast-1.amazonaws.com",
                    "codedeploy.ap-southeast-2.amazonaws.com",
                    "codedeploy.ap-southeast-3.amazonaws.com",
                    "codedeploy.ap-southeast-4.amazonaws.com",
                    "codedeploy.ap-south-1.amazonaws.com",
                    "codedeploy.ap-south-2.amazonaws.com",
                    "codedeploy.ca-central-1.amazonaws.com",
                    "codedeploy.eu-west-1.amazonaws.com",
                    "codedeploy.eu-west-2.amazonaws.com",
                    "codedeploy.eu-west-3.amazonaws.com",
```

```
                    "codedeploy.eu-central-1.amazonaws.com",
                    "codedeploy.eu-central-2.amazonaws.com",
                    "codedeploy.eu-north-1.amazonaws.com",
                    "codedeploy.eu-south-1.amazonaws.com",
                    "codedeploy.eu-south-2.amazonaws.com",
                    "codedeploy.il-central-1.amazonaws.com",
                    "codedeploy.me-central-1.amazonaws.com",
                    "codedeploy.me-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).

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`** or **`AWSCodeDeployRoleForLambdaLimited`**. For example:

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

- If your deployment is to an Amazon ECS compute platform:

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

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

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 3: Limit the CodeDeploy user's permissions

For security reasons, we recommend that you limit the permissions of the administrative user that you created in Step 1: Setting up to just those required to create and manage deployments in CodeDeploy.

Use the following series of procedures to limit the CodeDeploy administrative user's permissions.

Before you begin

- Make sure you have created a CodeDeploy administrative user in IAM Identity Center following the instructions in Step 1: Setting up.

To create a permission set

You'll assign this permission set to the CodeDeploy administrative user later.

1. Sign in to the AWS Management Console and open the AWS IAM Identity Center console at https://console.aws.amazon.com/singlesignon/.
2. In the navigation pane, choose **Permission sets**, and then choose **Create permission set**.
3. Choose **Custom permission set**.
4. Choose **Next**.
5. Choose **Inline policy**.
6. Remove the sample code.
7. Add the following policy code:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CodeDeployAccessPolicy",
            "Effect": "Allow",
            "Action": [
                "autoscaling:*",
                "codedeploy:*",
                "ec2:*",
                "lambda:*",
                "ecs:*",
                "elasticloadbalancing:*",
                "iam:AddRoleToInstanceProfile",
                "iam:AttachRolePolicy",
                "iam:CreateInstanceProfile",
                "iam:CreateRole",
                "iam:DeleteInstanceProfile",
                "iam:DeleteRole",
                "iam:DeleteRolePolicy",
                "iam:GetInstanceProfile",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "iam:ListInstanceProfilesForRole",
                "iam:ListRolePolicies",
                "iam:ListRoles",
                "iam:PutRolePolicy",
                "iam:RemoveRoleFromInstanceProfile",
                "s3:*",
                "ssm:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CodeDeployRolePolicy",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
```

```
            "Resource": "arn:aws:iam::111122223333:role/
CodeDeployServiceRole"
        }
    ]
}
```

In this policy, replace `arn:aws:iam::account-ID:role/CodeDeployServiceRole` with the ARN value of the CodeDeploy service role that you created in Step 2: Create a service role for CodeDeploy. You can find the ARN value in the details page of the service role in the IAM console.

The preceding policy lets you deploy an application to an AWS Lambda compute platform, an EC2/On-Premises compute platform, and an Amazon ECS compute platform.

You can use the CloudFormation templates provided in this documentation to launch Amazon EC2 instances that are compatible with CodeDeploy. To use CloudFormation templates to create applications, deployment groups, or deployment configurations, you must provide access to CloudFormation—and AWS services and actions that CloudFormation depends on—by adding the `cloudformation:*` permission to the CodeDeploy administrative user's permission policy, like this:

JSON

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

8. Choose **Next**.
9. In **Permission set name**, enter:

```
CodeDeployUserPermissionSet
```

10. Choose **Next**.
11. On the **Review and create** page, review the information and choose **Create**.

To assign the permission set to the CodeDeploy administrative user

1. In the navigation pane, choose **AWS accounts**, and then select the check box next to the AWS account that you're currently signed in to.
2. Choose the **Assign users or groups** button.
3. Choose the **Users** tab.
4. Select the check box next to the CodeDeploy administrative user.
5. Choose **Next**.
6. Select the check box next to `CodeDeployUserPermissionSet`.
7. Choose **Next**.
8. Review the information and choose **Submit**.

   You have now assigned the CodeDeploy administrative user and `CodeDeployUserPermissionSet` to your AWS account, binding them together.

To sign out and sign back in as the CodeDeploy administrative user

1. Before you sign out, make sure you have the AWS access portal URL and the username and one-time password for the CodeDeploy adminstrative user.

   > **Note**
   >
   > If you do not have this information, go to the CodeDeploy adminstrative user details page in IAM Identity Center, choose **Reset password**, **Generate a one-time password [...]**, and **Reset password** again to display the information on the screen.

2. Sign out of AWS.
3. Paste the AWS access portal URL into your browser's address bar.
4. Sign in as the CodeDeploy adminstrative user.

An **AWS account** box appears on the screen.

5. Choose **AWS account**, and then choose the name of the AWS account to which you assigned the CodeDeploy adminstrative user and permission set.
6. Next to the `CodeDeployUserPermissionSet`, choose **Management console**.

   The AWS Management Console appears. You are now signed in as the CodeDeploy adminstrative user with the limited permissions. You can now perform CodeDeploy-related operations, and *only* CodeDeploy-related operations, as this user.

Step 4: Create an IAM instance profile for your Amazon EC2 instances

> **Note**
>
> If you are using the Amazon ECS or AWS Lambda compute platform , skip this step.

Your Amazon EC2 instances need permission to access the Amazon S3 buckets or GitHub repositories where the applications are stored. To launch Amazon EC2 instances that are compatible with 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 the CodeDeploy agent 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)
- Create an IAM instance profile for your Amazon EC2 instances (console)

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 CodeDeploy.

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:

   JSON

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

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ec2.cn-north-1.amazonaws.com",
                    "ec2.cn-northwest-1.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 CodeDeploy agent. Otherwise, an error might occur when the CodeDeploy agent is installed or updated on the instances. To grant the IAM instance profile access to only some 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:::amzn-s3-demo-bucket/*",
>  "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-eu-central-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-north-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-2/*",
        "arn:aws:s3:::aws-codedeploy-il-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-east-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-4/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-2/*",
        "arn:aws:s3:::aws-codedeploy-me-central-1/*",
        "arn:aws:s3:::aws-codedeploy-me-south-1/*",
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
      ]
    }
  ]
}
```

Note

If you want to use IAM authorization or Amazon Virtual Private Cloud (VPC) endpoints with CodeDeploy, you will need to add more permissions. See Use CodeDeploy with Amazon Virtual Private Cloud for more information.

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 **attach-role-policy** to give the role Amazon EC2 Systems Manager permissions so that SSM can install the CodeDeploy agent. This policy is not needed if you plan to install the agent from the public Amazon S3 bucket with the command line. Learn more about installing the CodeDeploy agent.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonSSMManagedInstanceCore --role-name CodeDeployDemo-EC2-Instance-Profile
```

6. 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**.
3. On the **Specify permissions** page, choose **JSON**.
4. Remove the example `JSON` code.
5. Paste the following code:

```
{
    "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 CodeDeploy agent. Otherwise, an error might occur when the CodeDeploy agent is installed or updated on the instances. To grant the IAM instance

profile access to only some 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:::amzn-s3-demo-bucket/*",
        "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-eu-central-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-north-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-2/*",
        "arn:aws:s3:::aws-codedeploy-il-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-east-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-4/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-2/*",
        "arn:aws:s3:::aws-codedeploy-me-central-1/*",
        "arn:aws:s3:::aws-codedeploy-me-south-1/*",
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
      ]
    }
```

```
    ]
}
```

> **Note**
>
> If you want to use IAM authorization or Amazon Virtual Private Cloud (VPC) endpoints with CodeDeploy, you will need to add more permissions. See Use CodeDeploy with Amazon Virtual Private Cloud for more information.

6. Choose **Next**.
7. On the **Review and create** page, in the **Policy name** box, type **`CodeDeployDemo-EC2-Permissions`**.
8. (Optional) For **Description**, type a description for the policy.
9. Choose **Create policy**.
10. In the navigation pane, choose **Roles**, and then choose **Create role**.
11. Under **Use case**, choose the **EC2** use case.
12. Choose **Next**.
13. In the list of policies, select the check box next to the policy you just created (**CodeDeployDemo-EC2-Permissions**). If necessary, use the search box to find the policy.
14. To use Systems Manager to install or configure the CodeDeploy agent, select the check box next to **AmazonSSMManagedInstanceCore**. This AWS managed policy enables an instance to use Systems Manager service core functionality. If necessary, use the search box to find the policy. This policy is not needed if you plan to install the agent from the public Amazon S3 bucket with the command line. Learn more about installing the CodeDeploy agent.
15. Choose **Next**.
16. On the **Name, review, and create** page, in **Role name**, enter a name for the service role (for example, **`CodeDeployDemo-EC2-Instance-Profile`**), and then choose **Create role**.

    You can also enter a description for this service role in **Role description**.

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*.

Product and service integrations with CodeDeploy

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

- Integration with other AWS services
- Integration with partner products and services
- Integration examples from the community

Integration with other AWS services

CodeDeploy is integrated with the following AWS services:

Amazon CloudWatch	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. CodeDeploy supports the following CloudWatch tools: • **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 CodeDeploy to the application or deployment group where deployments should stop when the alarm is activated. Learn more: • Creating CloudWatch Logs alarms

	• **Amazon CloudWatch Events** for detecting and reacting to changes in the state of an instance or a deployment in your 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. Learn more: • Monitoring deployments with Amazon CloudWatch Events • **Amazon CloudWatch Logs** for monitoring the three types of logs created by the CodeDeploy agent without having to sign in to instances one at a time. Learn more: • View CodeDeploy logs in CloudWatch Logs console

Amazon EC2 Auto Scaling	CodeDeploy supports Amazon EC2 Auto Scaling. This AWS service can automatically launch Amazon EC2 instances based on criteria you specify, for example: • Limits exceeded for specified CPU utilizati on. • Disk reads or writes. • Inbound or outbound network traffic over a specified time interval. You can scale out a group of Amazon EC2 instances whenever you need them and then use CodeDeploy to deploy application revisions to them automatically. Amazon EC2 Auto Scaling terminates those Amazon EC2 instances when they are no longer needed. Learn more: • Integrating CodeDeploy with Amazon EC2 Auto Scaling • Tutorial: Use CodeDeploy to deploy an application to an Auto Scaling group • Under the hood: CodeDeploy and Auto Scaling integration

Amazon Elastic Container Service	You can use CodeDeploy to deploy an Amazon ECS containerized application as a task set. CodeDeploy performs a blue/green deployment by installing an updated version of the application as a new replacement task set. CodeDeploy reroutes production traffic from the original application task set to the replacement task set. The original task set is terminated after a successful deployment. For more information about Amazon ECS, see Amazon Elastic Container Service. You can manage the way in which traffic is shifted to the updated task set during a deployment by choosing a canary, linear, or all-at-once configuration. For more informati on about Amazon ECS deployments, see Deployments on an Amazon ECS compute platform.

AWS CloudTrail	CodeDeploy is integrated with AWS CloudTrail. This service captures API calls made by or on behalf of CodeDeploy in your AWS account and delivers the log files to an Amazon S3 bucket you specify. CloudTrail captures API calls from the CodeDeploy console, from CodeDeploy commands through the AWS CLI, or from the CodeDeploy APIs directly. Using the information collected by CloudTrail, you can determine: • Which request was made to CodeDeploy. • The source IP address from which the request was made. • Who made the request. • When it was made. Learn more: • Monitoring Deployments

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 informati on 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 to create a deploymen t 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	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 applicati on: 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. You can configure AWS CodePipeline to use CodeDeploy to deploy: • Code to Amazon EC2 instances, on-premises instances, or both. • Serverless AWS Lambda function versions. You can create the 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. Learn more: • AWS for DevOps getting started guide — Learn how to use CodePipeline with CodeDeploy to continuously deliver and deploy source code in CodeCommit repositories to Amazon EC2 instances. • Simple pipeline walkthrough (Amazon S3 Bucket)

	• Simple pipeline walkthrough (CodeCommit Repository) • Four-stage pipeline tutorial
AWS Serverless Application Model	AWS Serverless Application Model (AWS SAM) is a model to define serverless applications. It extends CloudFormation to provide a simplifie d way of defining AWS Lambda functions , Amazon API Gateway APIs, and Amazon DynamoDB tables required by a serverles s application. If you already use AWS SAM, you can add deployment preferences to start using CodeDeploy to manage the way in which traffic is shifted during an AWS Lambda application deployment. For more information, see the AWS Serverless Application Model.
Elastic Load Balancing	CodeDeploy supports Elastic Load Balancing, a service that distributes incoming application traffic across multiple Amazon EC2 instances. For 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 CodeDeploy with Elastic Load Balancing

Topics

- Integrating CodeDeploy with Amazon EC2 Auto Scaling
- Integrating CodeDeploy with Elastic Load Balancing

Integrating CodeDeploy with Amazon EC2 Auto Scaling

CodeDeploy supports Amazon EC2 Auto Scaling, an AWS service that launches 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. Amazon EC2 Auto Scaling terminates the instances when they are no longer needed. For more information, see What is Amazon EC2 Auto Scaling? in the *Amazon EC2 Auto Scaling User Guide*.

When new Amazon EC2 instances are launched as part of an Amazon EC2 Auto Scaling group, CodeDeploy can deploy your revisions to the new instances automatically. You can also coordinate deployments in CodeDeploy with Amazon EC2 Auto Scaling instances registered with Elastic Load Balancing load balancers. For more information, see Integrating CodeDeploy with Elastic Load Balancing and Set up a load balancer in Elastic Load Balancing for CodeDeploy Amazon EC2 deployments.

> **Note**
>
> You might encounter issues if you associate multiple deployment groups with a single Amazon EC2 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 Amazon EC2 Auto Scaling group and Under the hood: CodeDeploy and Amazon EC2 Auto Scaling integration.

Topics

- Deploying CodeDeploy applications to Amazon EC2 Auto Scaling groups
- Enabling termination deployments during Auto Scaling scale-in events
- How Amazon EC2 Auto Scaling works with CodeDeploy
- Using a custom AMI with CodeDeploy and Amazon EC2 Auto Scaling

Deploying CodeDeploy applications to Amazon EC2 Auto Scaling groups

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

1. Create or locate an IAM instance profile that allows the Amazon EC2 Auto Scaling group to work with Amazon S3. For more information, see Step 4: Create an IAM instance profile for your Amazon EC2 instances.

   > **Note**
   >
   > You can also use CodeDeploy to deploy revisions from GitHub repositories to Amazon EC2 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.

2. Create or use an Amazon EC2 Auto Scaling group, specifying the IAM instance profile in your launch configuration or template. For more information, see IAM role for applications that run on Amazon EC2 instances.
3. Create or locate a service role that allows CodeDeploy to create a deployment group that contains the Amazon EC2 Auto Scaling group.
4. Create a deployment group with CodeDeploy, specifying the Amazon EC2 Auto Scaling group name, the service role, and a few other options. For more information, see Create a deployment group for an in-place deployment (console) or Create a deployment group for an in-place deployment (console).
5. Use CodeDeploy to deploy your revision to the deployment group that contains the Amazon EC2 Auto Scaling group.

For more information, see Tutorial: Use CodeDeploy to deploy an application to an Auto Scaling group.

Enabling termination deployments during Auto Scaling scale-in events

A *termination deployment* is a type of CodeDeploy deployment that is activated automatically when an Auto Scaling scale-in event occurs. CodeDeploy performs the termination deployment right before the Auto Scaling service terminates the instance. During a termination deployment, CodeDeploy doesn't deploy anything. Instead, it generates lifecycle events, which you can hook up to your own scripts to enable custom shutdown functionality. For example, you could hook up the `ApplicationStop` lifecycle event to a script that shuts down your application gracefully before the instance is terminated.

For a list of lifecycle events that CodeDeploy generates during a termination deployment, see Lifecycle event hook availability.

If the termination deployment fails for any reason, CodeDeploy will allow the instance termination to proceed. This means that the instance will be shut down even though CodeDeploy did not run the full set (or any) of the lifecycle events to completion.

If you don't enable termination deployments, the Auto Scaling service will still terminate Amazon EC2 instances when a scale-in event occurs, but CodeDeploy will not generate lifecycle events.

> **Note**
>
> Regardless of whether you enable termination deployments or not, if the Auto Scaling service terminates an Amazon EC2 instance while a CodeDeploy deployment is underway, then a race condition may occur between the lifecycle events generated by the Auto Scaling and CodeDeploy services. For example, the `Terminating` lifecycle event (generated by the Auto Scaling service) might override the `ApplicationStart` event (generated by the CodeDeploy deployment). In this scenario, you may experience a failure with either the Amazon EC2 instance termination or the CodeDeploy deployment.

To enable CodeDeploy to perform termination deployments

- Select the **Add a termination hook to Auto Scaling groups** check box when creating or updating your deployment group. For instructions, see Create a deployment group for an in-place deployment (console), or Create a deployment group for an EC2/On-Premises blue/green deployment (console).

  Enabling this check box causes CodeDeploy to install an Auto Scaling lifecycle hook into the Auto Scaling groups that you specify when you create or update your CodeDeploy deployment group. This hook is called the *termination hook* and enables termination deployments.

After the termination hook is installed, a scale-in (termination) event unfolds as follows:

1. The Auto Scaling service (or simply, Auto Scaling) determines that a scale-in event needs to occur, and contacts the EC2 service to terminate an EC2 instance.
2. The EC2 service starts terminating the EC2 instance. The instance moves into the `Terminating` state, and then into the `Terminating:Wait` state.

3. During `Terminating:Wait`, Auto Scaling runs all the lifecycle hooks attached to the Auto Scaling group, including the termination hook installed by CodeDeploy.
4. The termination hook sends a notification to the Amazon SQS queue that is polled by CodeDeploy.
5. Upon receiving the notification, CodeDeploy parses the message, performs some validation, and performs a termination deployment.
6. While the termination deployment is running, CodeDeploy sends heartbeats every five minutes to Auto Scaling to let it know that the instance is still being worked on.
7. So far, the EC2 instance is still in the `Terminating:Wait` state (or possibly the `Warmed:Pending:Wait` state, if you've enabled Auto Scaling group warm pools).
8. When the deployment completes, CodeDeploy indicates to Auto Scaling to `CONTINUE` the EC2 termination process, regardless of whether the termination deployment succeeded or failed.

How Amazon EC2 Auto Scaling works with CodeDeploy

When you create or update a CodeDeploy deployment group to include an Auto Scaling group, CodeDeploy accesses the Auto Scaling group using the CodeDeploy service role, and then installs Auto Scaling lifecycle hooks into your Auto Scaling groups.

> **Note**
>
> *Auto Scaling lifecycle hooks* are different from the *lifecycle events* (also called *lifecycle event hooks*) generated by CodeDeploy and described in the AppSpec 'hooks' section of this guide.

The Auto Scaling lifecycle hooks that CodeDeploy installs are:

- **A launch hook** — This hook notifies CodeDeploy that an Auto Scaling scale-out event is in progress, and that CodeDeploy needs to start a launch deployment.

  During a *launch deployment*, CodeDeploy:

  - Deploys a revision of your application to the scaled-out instance.
  - Generates lifecycle events to indicate the progress of the deployment. You can hook up these lifecycle events to your own scripts to enable custom startup functionality. For more information, see the table in Lifecycle event hook availability.

The launch hook and associated launch deployment are always enabled and cannot be turned off.

- **A termination hook** — This optional hook notifies CodeDeploy that an Auto Scaling scale-in event is in progress, and that CodeDeploy needs to start a termination deployment.

  During a *termination deployment*, CodeDeploy generates lifecycle events to indicate the progress of the instance shutdown. For more information, see Enabling termination deployments during Auto Scaling scale-in events.

Topics

- After CodeDeploy installs the lifecycle hooks, how are they used?
- How CodeDeploy names Amazon EC2 Auto Scaling groups
- Execution order of custom lifecycle hook events
- Scale-out events during a deployment
- Scale-in events during a deployment
- Order of events in AWS CloudFormation cfn-init scripts

After CodeDeploy installs the lifecycle hooks, how are they used?

After the launch and termination lifecycle hooks are installed, they are used by CodeDeploy during Auto Scaling group scale-out and scale-in events, respectively.

A scale-out (launch) event unfolds as follows:

1. The Auto Scaling service (or simply, Auto Scaling) determines that a scale-out event needs to occur, and contacts the EC2 service to launch a new EC2 instance.
2. The EC2 service launches a new EC2 instance. The instance moves into the `Pending` state, and then into the `Pending:Wait` state.
3. During `Pending:Wait`, Auto Scaling runs all the lifecycle hooks attached to the Auto Scaling group, including the launch hook installed by CodeDeploy.
4. The launch hook sends a notification to the Amazon SQS queue that is polled by CodeDeploy.
5. Upon receiving the notification, CodeDeploy parses the message, performs some validation, and starts a launch deployment.

6. While the launch deployment is running, CodeDeploy sends heartbeats every five minutes to Auto Scaling to let it know that the instance is still being worked on.
7. So far, the EC2 instance is still in the `Pending:Wait` state.
8. When the deployment completes, CodeDeploy indicates to Auto Scaling to either `CONTINUE` or `ABANDON` the EC2 launch process, depending on whether the deployment succeeded or failed.
   - If CodeDeploy indicates `CONTINUE`, Auto Scaling continues the launch process, either waiting for other hooks to complete, or putting the instance into the `Pending:Proceed` and then the `InService` state.
   - If CodeDeploy indicates `ABANDON`, Auto Scaling terminates the EC2 instance, and restarts the launch procedure if needed to meet the desired number of instances, as defined in the Auto Scaling **Desired Capacity** setting.

A scale-in (termination) event unfolds as follows:

See Enabling termination deployments during Auto Scaling scale-in events.

How CodeDeploy names Amazon EC2 Auto Scaling groups

During blue/green deployments on an EC2/On-Premises compute platform, you have two options for adding instances to your replacement (green) environment:

- Use instances that already exist or that you create manually.
- Use settings from an Amazon EC2 Auto Scaling group that you specify to define and create instances in a new Amazon EC2 Auto Scaling group.

If you choose the second option, CodeDeploy provisions a new Amazon EC2 Auto Scaling group for you. It uses the following convention to name the group:

```
CodeDeploy_deployment_group_name_deployment_id
```

For example, if a deployment with ID `10` deploys a deployment group named `alpha-deployments`, the provisioned Amazon EC2 Auto Scaling group is named `CodeDeploy_alpha-deployments_10`. For more information, see Create a deployment group for an EC2/On-Premises blue/green deployment (console) and GreenFleetProvisioningOption.

Execution order of custom lifecycle hook events

You can add your own lifecycle hooks to Amazon EC2 Auto Scaling groups to which CodeDeploy deploys. However, the order in which those custom lifecycle hook events are executed cannot be predetermined in relation to CodeDeploy default deployment lifecycle events. For example, if you add a custom lifecycle hook named `ReadyForSoftwareInstall` to an Amazon EC2 Auto Scaling group, you cannot know beforehand whether it will be executed before the first, or after the last, CodeDeploy default deployment lifecycle event.

To learn how to add custom lifecycle hooks to an Amazon EC2 Auto Scaling group, see Adding lifecycle hooks in the *Amazon EC2 Auto Scaling User Guide*.

Scale-out events during a deployment

If an Auto Scaling scale-out event occurs while a deployment is underway, the new instances will be updated with the application revision that was previously deployed, not the newest application revision. If the deployment succeeds, the old instances and the newly scaled-out instances will be hosting different application revisions. To bring the instances with the older revision up to date, CodeDeploy automatically starts a follow-on deployment (immediately after the first) to update any outdated instances. If you'd like to change this default behavior so that outdated EC2 instances are left at the older revision, see Automatic updates to outdated instances.

If you want to suspend Amazon EC2 Auto Scaling scale-out 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 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.

For more information about using `HANDLE_PROCS=true` to avoid deployment problems when using Amazon EC2 Auto Scaling, see Important notice about handling AutoScaling processes in aws-codedeploy-samples on GitHub.

Scale-in events during a deployment

If an Auto Scaling group starts scaling in while a CodeDeploy deployment is underway on that Auto Scaling group, a race condition could occur between the termination process (including the CodeDeploy termination deployment lifecycle events) and other CodeDeploy lifecycle events on the terminating instance. The deployment on that specific instance may fail if the instance is terminated before all CodeDeploy lifecycle events complete. Also, the overall CodeDeploy deployment may or may not fail, depending on how you've set your **Minimum healthy hosts** setting in your deployment configuration.

Order of events in 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 CodeDeploy agent starts.
4. The latest application revision is deployed to the instance.

If the order of events is not carefully controlled, the 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 CodeDeploy agent through a `cfn-init` script, placing it after all other scripts.
- Include the 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 the *AWS CloudFormation User Guide*.

Using a custom AMI with CodeDeploy and Amazon EC2 Auto Scaling

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

- You can specify a base custom AMI that already has the 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 CodeDeploy agent is out of date. If you choose this option, we recommend you regularly update the CodeDeploy agent in your base custom AMI.
- You can specify a base AMI that doesn't have the CodeDeploy agent installed and have the agent installed as each new instance is launched in an Amazon EC2 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 CodeDeploy agent.

Integrating CodeDeploy with Elastic Load Balancing

During 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.

Elastic Load Balancing types

Elastic Load Balancing provides three types of load balancers that can be used in 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 a VPC.

> **Note**
>
> Classic Load Balancers are not supported with Amazon ECS deployments.

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).

> **Note**
>
> The Application Load Balancer target groups must have a target type of `instance` for deployments on EC2 instances, and `IP` for Fargate deployments. For more information, see Target type.

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?

Blue/Green deployments

Rerouting instance traffic behind an Elastic Load Balancing load balancer is fundamental to 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 one or more load balancers, instances from the original environment are deregistered and, if you choose, terminated.

For a blue/green deployment, you can specify one or more Classic Load Balancers, Application Load Balancer target groups, or Network Load Balancer target groups in your deployment group. You use the CodeDeploy console or AWS CLI to add the load balancers 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 CodeDeploy Amazon EC2 deployments
- Create an application for a blue/green deployment (console)
- Create a deployment group for an EC2/On-Premises blue/green deployment (console)

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 the 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. CodeDeploy will wait for up to 1 hour for the instance to become healthy behind the load balancer. If the instance is not marked as healthy by the load balancer during the waiting period, CodeDeploy either moves onto the next instance or fails the deployment, based on the deployment configuration.

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

Specify in-place deployment load balancer using a deployment group

To add load balancers to a deployment group, you use the 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)
- Create a deployment group for an in-place deployment (console)
- Set up a load balancer in Elastic Load Balancing for CodeDeploy Amazon EC2 deployments

Specify in-place deployment load balancer using a script

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 CodeDeploy.

In the CodeDeploy Samples repository on GitHub, we provide instructions and samples you can adapt to use 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 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 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 Amazon EC2 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 balancer, wait for the connection to drain, and then re-register the Amazon EC2 instance with the load balancer after the deployment is complete.

Integration with partner products and services

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

Ansible	If you already have a set of Ansible playbooks , but just need somewhere to run them, the template for Ansible and CodeDeplo y demonstrates how a couple of simple

	deployment hooks can ensure Ansible is available on the local deployment instance and runs the playbooks. 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 CodeDeploy agent. Learn more: • Ansible and CodeDeploy
Atlassian – Bamboo and Bitbucket	The 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 CodeDeploy application. Atlassian Bitbucket support for 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 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. Learn more: • Using the CodeDeploy task for Bamboo • Announcing Atlassian Bitbucket support for CodeDeploy

Chef	AWS provides two template samples for integrating Chef and CodeDeploy. The first is a Chef cookbook that installs and starts the CodeDeploy agent. This allows you to continue managing your host infrastructure with Chef while using CodeDeploy. The second sample template demonstrates how to use CodeDeplo y to orchestrate the running of cookbooks and recipes with chef-solo on each node. Learn more: • Chef and CodeDeploy
CircleCI	CircleCI provides an automated testing and continuous integration and deploymen t 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 CodeDeploy to create application revisions, upload them to an Amazon S3 bucket, and then initiate and monitor your deployments. Learn more: • Using a CircleCI Orb to deploy applications to AWS CodeDeploy

CloudBees	You can use the 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. Learn more: • CodeDeploy Jenkins plugin now available on DEV@cloud
Codeship	You can use Codeship to deploy applicati on revisions through CodeDeploy. You can use the Codeship UI to add CodeDeploy to a deployment pipeline for a branch. Learn more: • Deploy to CodeDeploy • CodeDeploy integration on Codeship
GitHub	You can use CodeDeploy to deploy applicati on revisions from GitHub repositories. You can also trigger a deployment from a GitHub repository whenever the source code in that repository is changed. Learn more: • Integrating CodeDeploy with GitHub • Tutorial: Use CodeDeploy to deploy an application from GitHub • Automatically deploy from GitHub using CodeDeploy

HashiCorp Consul	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 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. Learn more: • CodeDeploy deployments with HashiCorp Consul
Jenkins	The CodeDeploy Jenkins plugin provides a post-build step for your Jenkins project. Upon a successful build, it zips the workspace, uploads to Amazon S3, and starts a new deployment. Learn more: • CodeDeploy Jenkins plugin • Setting up the Jenkins plugin for CodeDeploy

Puppet Labs	AWS provides sample templates for Puppet and CodeDeploy. The first is a Puppet module that installs and starts the CodeDeploy agent. This allows you to continue managing your host infrastructure with Puppet while using CodeDeploy. The second sample template demonstrates how to use CodeDeploy to orchestrate the running of modules and manifests with a masterless puppet on each node. Learn more: • Puppet and CodeDeploy
SaltStack	You can integrate SaltStack infrastructure with CodeDeploy. You can use the CodeDeplo y module to install and run the CodeDeplo y agent on your minions or, with a couple of simple deployment hooks, you can use CodeDeploy to orchestrate the running of your Salt States. Learn more: • SaltStack and CodeDeploy

TeamCity	You can use the 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 a CodeDeploy application, creates a CodeDeploy deployment and, if you choose, waits for the deployment to be completed. Learn more: • CodeDeploy Runner (Download) • CodeDeploy Runner plugin (Documentation)
Travis CI	You can configure Travis CI to trigger a deployment in CodeDeploy after a successful build. Learn more: • Travis CI and CodeDeploy deployments

Topics

- Integrating CodeDeploy with GitHub

Integrating CodeDeploy with GitHub

CodeDeploy supports GitHub, a web-based code hosting and sharing service. CodeDeploy can deploy application revisions stored in GitHub repositories or Amazon S3 buckets to instances. CodeDeploy supports GitHub for EC2/On-Premises deployments only.

Topics

- Deploying CodeDeploy revisions from GitHub
- GitHub behaviors with CodeDeploy

Deploying CodeDeploy revisions from GitHub

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

1. Create a revision that's compatible with 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 CodeDeploy and Add an application specification file to a revision for CodeDeploy.

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

   If you want to call the **create-deployment** command, you must first use the **Create deployment** page of the console to give 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 CodeDeploy.

   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).

   To learn how to prepare instances for use in CodeDeploy deployments, see Working with instances for CodeDeploy.

For more information, see Tutorial: Use CodeDeploy to deploy an application from GitHub.

GitHub behaviors with CodeDeploy

Topics

- GitHub authentication with applications in CodeDeploy
- CodeDeploy interaction with private and public GitHub repositories
- CodeDeploy interaction with organization-managed GitHub repositories

- Automatically deploy from CodePipeline with CodeDeploy

GitHub authentication with applications in CodeDeploy

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

To link a GitHub account to an application in CodeDeploy

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, then choose **Applications**.
3. Choose the application you want to link to a different GitHub account.
4. If your application does not have a deployment group, choose **Create deployment group** to create one. For more information, see Create a deployment group with CodeDeploy. A deployment group is required to choose **Create deployment** in the next step.
5. From **Deployments**, 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.

6. In **Deployment settings**, for **Revision type**, choose **My application is stored in GitHub**.
7. 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 token name**, type a name to identify this connection, and then choose **Connect to GitHub**. The web page prompts you to authorize CodeDeploy to interact with GitHub for your application. Continue to step 10.
   - To use a connection you have already created, in **GitHub token name**, select its name, and then choose **Connect to GitHub**. Continue to step 8.

- To create a connection to a different GitHub account, sign out of GitHub in a separate web browser tab. In **GitHub token name**, type a name to identify the connection, and then choose **Connect to GitHub**. The web page prompts you to authorize CodeDeploy to interact with GitHub for your application. Continue to step 10.

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

To revoke permission for 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 **CodeDeploy** in the list of authorized applications, and then follow the GitHub procedure for revoking authorization for an application.

CodeDeploy interaction with private and public GitHub repositories

CodeDeploy supports the deployment of applications from private and public GitHub repositories. When you give CodeDeploy permission to access GitHub on your behalf, CodeDeploy will have read-write access to all of the private GitHub repositories to which your GitHub account has access. However, CodeDeploy only reads from GitHub repositories. It will not write to any of your private GitHub repositories.

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 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 CodeDeploy. The steps for requesting this access depend on whether you have already authorized CodeDeploy for your individual account:

- If you have authorized access to CodeDeploy in your account, see Requesting organization approval for your authorized applications.
- If you have not yet authorized access to 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 CodePipeline with CodeDeploy

You can trigger a deployment from a CodePipeline whenever the source code changes. For more infomation, see CodePipeline.

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 CodeDeploy provisioning in CloudFormation

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

  Published January 2016
- AWS Toolkit for Eclipse Integration with CodeDeploy (Part 1)

  AWS Toolkit for Eclipse Integration with CodeDeploy (Part 2)

  AWS Toolkit for Eclipse Integration with CodeDeploy (Part 3)

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

Published February 2015

- Automatically deploy from GitHub using CodeDeploy

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

  Published December 2014

CodeDeploy tutorials

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

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, amzn-s3-demo-bucket, 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)
- Tutorial: Deploy a "hello, world!" application with CodeDeploy (Windows Server)
- Tutorial: Deploy an application to an on-premises instance with CodeDeploy (Windows Server, Ubuntu Server, or Red Hat Enterprise Linux)
- Tutorial: Use CodeDeploy to deploy an application to an Auto Scaling group
- Tutorial: Use CodeDeploy to deploy an application from GitHub
- Tutorial: Deploy an application into Amazon ECS
- Tutorial: Deploy an Amazon ECS service with a validation test
- Tutorial: Deploy an updated Lambda function with CodeDeploy and the AWS Serverless Application Model

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

In this tutorial, you 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 CodeDeploy (Windows Server).
- 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 CodeDeploy (Windows Server, Ubuntu Server, or Red Hat Enterprise Linux).

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 must 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 CodeDeploy. These include configuring a user, 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
- Step 2: Configure your source content to be deployed to the Amazon Linux or Red Hat Enterprise Linux Amazon EC2 instance
- Step 3: Upload your WordPress application to Amazon S3
- Step 4: Deploy your WordPress application
- Step 5: Update and redeploy your WordPress application
- Step 6: Clean up your WordPress application and related resources

Step 1: Launch and configure an Amazon Linux or Red Hat Enterprise Linux Amazon EC2 instance

To deploy the WordPress application with 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 Create an Amazon EC2 instance for CodeDeploy. 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 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 CodeDeploy as the 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. You can now sign out of the running Amazon EC2 instance.

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

Add an inbound rule that allows 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 then choose your instance.
3. On the **Description** tab, under **Security groups**, choose **view inbound rules**.

   You should see a list of rules in your security group like 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 the security group for your Amazon EC2 instance. It might be named **`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. If the security group for your instance is configured correctly, you should see a rule with the following values:

   - **Type**: HTTP

- **Protocol**: TCP
- **Port Range**: 80
- **Source**: 0.0.0.0/0

5. If you do not see a rule with these values, 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
- Create scripts to run your application
- Add an application specification file

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)
- To get a copy of the WordPress source code (Git)

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. 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 amazon-linux-extras install php7.4
sudo yum install -y httpd mariadb-server php
```

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
systemctl start mariadb.service
systemctl start httpd.service
systemctl start php-fpm.service
```

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
systemctl stop httpd.service
fi
isExistApp=pgrep mysqld
if [[ -n $isExistApp ]]; then
```

```
systemctl stop mariadb.service
fi
isExistApp=pgrep php-fpm
if [[ -n $isExistApp ]]; then
systemctl stop php-fpm.service

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 IF NOT EXISTS 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. 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 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
```

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, 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, CodeDeploy stops the deployment and marks the deployment as failed.

For more information about these settings, see the CodeDeploy AppSpec file reference.

> **Important**
>
> The locations and numbers of spaces between each of the items in this file are important. If the spacing is incorrect, CodeDeploy raises an error that might be difficult to debug. For more information, see AppSpec File spacing.

Step 3: Upload your WordPress application to Amazon S3

Now you will prepare and upload your source content to a location from which 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 CodeDeploy to deploy applications from GitHub repositories to instances. For more information, see Integrating CodeDeploy with GitHub.

Topics

- Provision an Amazon S3 bucket
- Prepare the application's files for the bucket
- Bundle the application's files into a single archive file and push the archive file

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 AWS account.

> **Note**
>
> Bucket names must be unique across Amazon S3 for all AWS accounts. If you aren't able to use **`amzn-s3-demo-bucket`**, try a different bucket name, such as **`amzn-s3-demo-bucket`** followed by a dash and your initials or some other unique identifier. Then be sure to substitute your bucket name for **`amzn-s3-demo-bucket`** 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)
- To create an Amazon S3 bucket (console)
- Give permissions to the Amazon S3 bucket and AWS account

To create an Amazon S3 bucket (CLI)

Call the **mb** command to create an Amazon S3 bucket named **`amzn-s3-demo-bucket`**:

```
aws s3 mb s3://amzn-s3-demo-bucket --region region
```

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 AWS account

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 `amzn-s3-demo-bucket`:

```
{
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "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::444455556666:role/CodeDeployDemo` to download files from any directory in the Amazon S3 bucket named `amzn-s3-demo-bucket`:

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

```
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/CodeDeployDemo"
                ]
            }
        }
    ]
}
```

For information about how to generate and attach an Amazon S3 bucket policy, see Bucket policy examples.

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

```
aws deploy push \
  --application-name WordPress_App \
  --s3-location s3://amzn-s3-demo-bucket/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 **`amzn-s3-demo-bucket`** bucket, and registers information with 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 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 CodeDeploy
- Monitor and troubleshoot your deployment
- Verify your deployment

Deploy your application revision with CodeDeploy

Use the AWS CLI or the console to deploy your application revision.

Topics

- To deploy your application revision (CLI)
- To deploy your application revision (console)

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 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) to create a service role. To get the ARN of the service role, see Get the service role ARN (CLI).

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. Before you create a deployment, the instances in your deployment group must have the CodeDeploy agent installed. You can install the agent from the command line with AWS Systems Manager with the following command:

```
aws ssm create-association \
  --name AWS-ConfigureAWSPackage \
  --targets Key=tag:Name,Values=CodeDeployDemo \
  --parameters action=Install,name=AWSCodeDeployAgent \
  --schedule-expression "cron(0 2 ? * SUN *)"
```

   This command creates an association in Systems Manager State Manager that will install the CodeDeploy agent and then attempt to update it at 2:00 every Sunday morning. For more information about the CodeDeploy agent, see Working with the CodeDeploy agent. For more information about Systems Manager, see What is AWS Systems Manager.

4. 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 **`amzn-s3-demo-bucket`**:

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

To deploy your application revision (console)

1. Before you use the 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 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) to create a service role. To get the ARN of the service role, see Get the service role ARN (console).

2. Now that you have the ARN, use the CodeDeploy console to deploy your application revision:

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

3. In the navigation pane, expand **Deploy**, then choose **Applications**.
4. In the list of applications, choose **WordPress_App**.
5. On the **Deployment groups** tab, choose **Create deployment group**.
6. In **Deployment group name**, enter **`WordPress_DepGroup`**.
7. Under **Deployment type**, choose **In-place deployment**.
8. In **Environment configuration**, select **Amazon EC2 instances**.
9. In **Agent configuration with AWS Systems Manager**, keep the defaults.
10. In **Key**, enter **`Name`**.
11. In **Value**, enter **`CodeDeployDemo`**.

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

12. In **Deployment configuration**, choose **CodeDeployDefault.OneAtATime**.
13. In **Service role ARN**, choose the service role ARN, and then choose **Create deployment group**.
14. Choose **Create deployment**.

15. In **Deployment group** choose **`WordPress_DepGroup`**.
16. Next to **Repository type**, choose **My application is stored in Amazon S3**. In **Revision location**, enter 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 **amzn-s3-demo-bucket** (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/amzn-s3-demo-bucket/WordPressApp.zip`
    e. Return to the 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. Expand **Deployment group overrides**, and from **Deployment configuration**, choose **CodeDeployDefault.OneAtATime**.
20. Choose **Start deployment**. Information about your newly created deployment appears on the **Deployments** page.

Monitor and troubleshoot your deployment

Use the AWS CLI or the console to monitor and troubleshoot your deployment.

Topics

- To monitor and troubleshoot your deployment (CLI)
- To monitor and troubleshoot your deployment (console)

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.

To monitor and troubleshoot your deployment (console)

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

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 name of the deployment. After a deployment fails, a message that describes the reason for the failure is displayed.
2. In **Instance activity**, 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.
3. If you want to do more troubleshooting, you can use a technique like the one described in View Instance Details. 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.

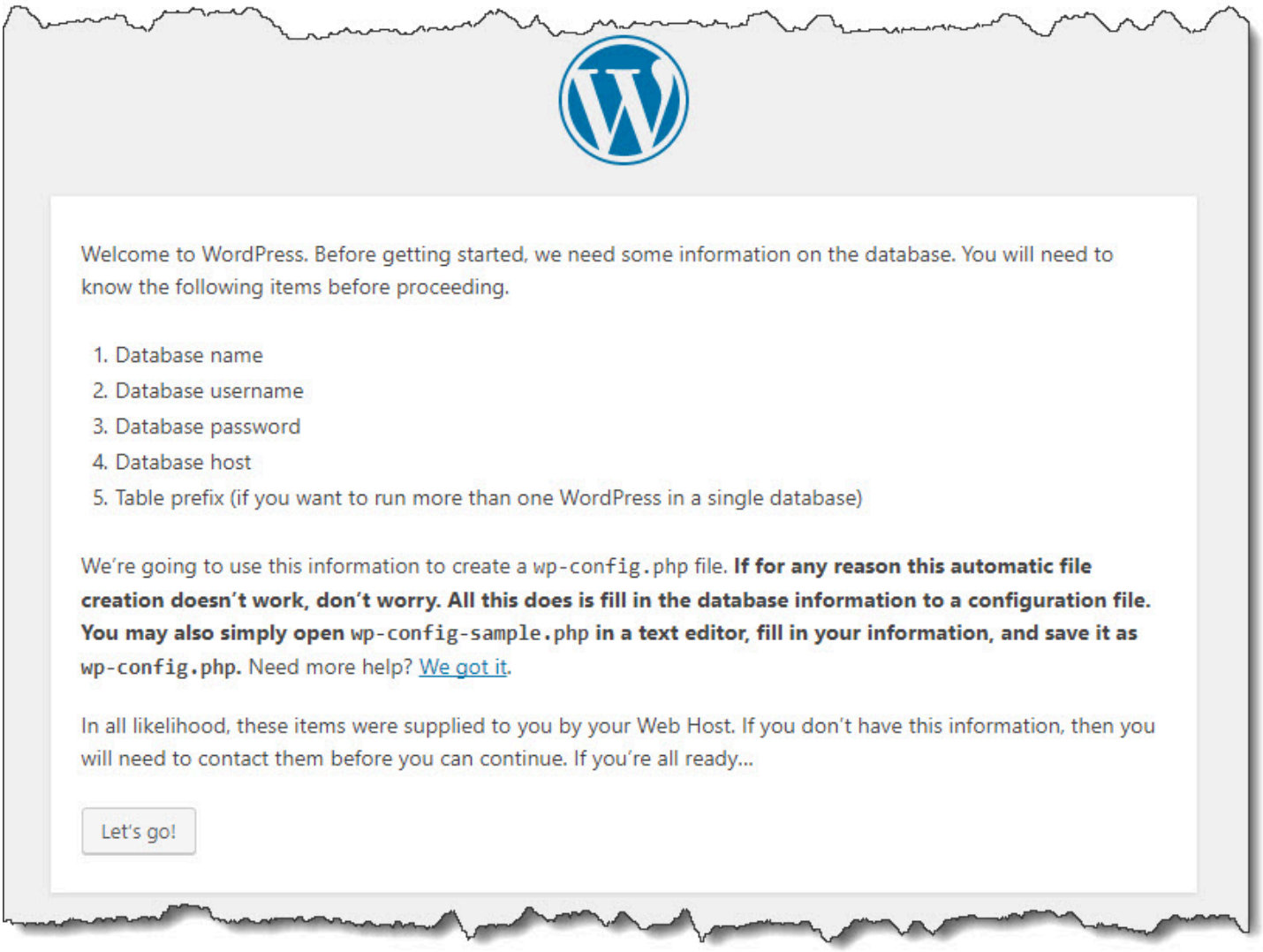
Verify your deployment

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 an inbound rule that allows HTTP traffic to your Amazon Linux or RHEL Amazon EC2 instance.

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 CodeDeploy to redeploy the site. Afterward, you should see the code changes on the Amazon EC2 instance.

Topics

- Set up the WordPress site
- Modify the site
- Redeploy the site

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 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. (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, **`amzn-s3-demo-bucket`**).

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

Topics

- To redeploy the site (CLI)
- To redeploy the site (console)

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 **`amzn-s3-demo-bucket`**:

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

You can check the status of the deployment, as described in Monitor and troubleshoot your deployment.

After 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 CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, then choose **Applications**.
3. In the list of applications, choose **WordPress_App**.
4. On the **Deployment groups** tab, choose **`WordPress_DepGroup`**.
5. Choose **Create deployment**.

6. On the **Create deployment** page:

   a. In **Deployment group**, choose **`WordPress_DepGroup`**.

   b. 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 **amzn-s3-demo-bucket**, 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 CodeDeploy console.

   c. If a message appears saying the file type could not be detected, choose **.zip**.

   d. Leave the **Deployment description** box blank.

   e. Expand **Deployment group overrides** and from **Deployment configuration**, choose **CodeDeployDefault.OneAtATime**.

   f. Choose **Start deployment**. Information about your newly created deployment appears on the **Deployments** page.

   g. You can check the status of the deployment, as described in Monitor and troubleshoot your deployment.

      After 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 CloudFormation stacks (or terminate any Amazon EC2 instances, if you created them outside of CloudFormation).

- Any Amazon S3 buckets.
- The `WordPress_App` application in CodeDeploy.
- The AWS Systems Manager State Manager association for the CodeDeploy agent.

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

Topics

- To clean up resources (CLI)
- To clean up resources (console)
- What's next?

To clean up resources (CLI)

1. If you used our 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 **`amzn-s3-demo-bucket`**. This will delete the bucket and all objects in the bucket:

```
aws s3 rm s3://amzn-s3-demo-bucket --recursive --region region
```

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
```

4. To delete the Systems Manager State Manager association, call the **delete-association** command.

```
aws ssm delete-association --assocation-id association-id
```

You can get the *association-id* by calling the **describe-association** command.

```
aws ssm describe-association --name AWS-ConfigureAWSPackage --targets
 Key=tag:Name,Values=CodeDeployDemo
```

If you did not use the 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 CloudFormation template for this tutorial, delete the associated CloudFormation stack.

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. In the **Filter** box, type the CloudFormation stack name you created earlier (for example, **`CodeDeployDemoStack`**).
3. Select the box beside stack name. In the **Actions** menu, choose **Delete Stack**.

   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 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, **`amzn-s3-demo-bucket`**).
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 CodeDeploy:

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, then choose **Applications**.
3. In the list of applications, choose **WordPress_App**.
4. On the **Application details** page, choose **Delete application**.
5. When prompted, enter the name of the application to confirm you want to delete it, and then choose **Delete**.

To delete the Systems Manager State Manager association:

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager.
2. In the navigation pane, choose **State Manager**.
3. Choose the association you created and choose **Delete**.

What's next?

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

Tutorial: Deploy a "hello, world!" application with CodeDeploy (Windows Server)

In this tutorial, you deploy a single webpage to a single Windows Server Amazon EC2 instance running Internet Information Services (IIS) as its web server. This webpage displays 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).
- To practice deploying to an on-premises instance instead, see Tutorial: Deploy an application to an on-premises instance with CodeDeploy (Windows Server, Ubuntu Server, or Red Hat Enterprise Linux).

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 must adapt those that cover Windows-based directory paths such as `c:\temp`. Also, if you want to connect to the Amazon EC2 instance, you need a client application that can connect 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 CodeDeploy, including configuring your 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
- Step 2: Configure your source content to deploy to the Windows Server Amazon EC2 instance
- Step 3: Upload your "hello, world!" application to Amazon S3

- Step 4: Deploy your Hello World application
- Step 5: Update and redeploy your "hello, world!" application
- Step 6: Clean up your "hello, world!" application and related resources

Step 1: Launch a Windows Server Amazon EC2 instance

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

Follow the instructions in Create an Amazon EC2 instance for CodeDeploy. 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 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 CodeDeploy 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**, and then choose **Choose File**.
6. 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 disconnect from the Amazon EC2 instance.

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

Add an inbound rule that allows HTTP traffic to your Windows Server Amazon EC2 instance

The next step confirms your Amazon EC2 instance has an open HTTP port so you can see the deployed webpage on your Windows Server Amazon EC2 instance 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 then choose your instance.
3. On the **Description** tab, under **Security groups**, choose **view inbound rules**.

   You should see a list of rules in your security group like 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 the security group for your Amazon EC2 instance. It might be named **`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. If the security group for your instance is configured correctly, you should see a rule with the following values:

   - **Type**: HTTP
   - **Protocol**: TCP
   - **Port Range**: 80
   - **Source**: 0.0.0.0/0
5. If you do not see a rule with these values, 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 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
- Create a script to run your application
- Add an application specification file

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
 CodeDeploy</h2></div>
  <div align="center">
    <p>What to do next? Take a look through the <a href="https://aws.amazon.com/
codedeploy">CodeDeploy Documentation</a>.</p>
  </div>
</body>
</html>
```

Create a script to run your application

Next, you will create a script that 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 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
```

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, 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, CodeDeploy will stop the deployment and mark the deployment to the Amazon EC2 instance as failed.

For more information about these settings, see the CodeDeploy AppSpec file reference.

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

Step 3: Upload your "hello, world!" application to Amazon S3

Now you will prepare and upload your source content to a location from which 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 CodeDeploy to deploy applications from GitHub repositories to instances. For more information, see Integrating CodeDeploy with GitHub.

Topics

- Provision an Amazon S3 bucket
- Prepare the application's files for the bucket
- Bundle the application's files into a single archive file and push the archive file

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 CodeDeploy user.

> **Note**
>
> Bucket names must be unique across Amazon S3 for all AWS accounts. If you aren't able to use **`amzn-s3-demo-bucket`**, try a different bucket name, such as **`amzn-s3-demo-bucket`** followed by a dash and your initials or some other unique identifier. Then be sure to substitute your bucket name for **`amzn-s3-demo-bucket`** 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)
- To create an Amazon S3 bucket (console)
- Give permissions to the Amazon S3 bucket and your AWS account

To create an Amazon S3 bucket (CLI)

Call the **mb** command to create an Amazon S3 bucket named **`amzn-s3-demo-bucket`**:

```
aws s3 mb s3://amzn-s3-demo-bucket --region region
```

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 AWS account

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 `amzn-s3-demo-bucket`:

```
{
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "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::444455556666:role/CodeDeployDemo` to download files from any directory in the Amazon S3 bucket named `amzn-s3-demo-bucket`:

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/CodeDeployDemo"
                ]
            }
        }
    ]
}
```

For information about how to generate and attach an Amazon S3 bucket policy, see Bucket policy examples.

The CodeDeploy adminstrative user that you created in Step 1: Setting up must also have permission to upload the revision to the Amazon S3 bucket. One way to specify this is through an IAM policy, which you add to the user's permission set, or to an IAM role (which you allow the user to assume). The following IAM policy allows the user to upload revisions anywhere in the Amazon S3 bucket named `amzn-s3-demo-bucket`:

JSON

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

```
}
```

For information about how to create an IAM policy, see Creating IAM policies in the *IAM User Guide*. For information on adding a policy to a permission set, see Create a permission set in the *AWS IAM Identity Center User Guide*.

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 CodeDeploy:

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

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

```
aws deploy push --application-name HelloWorld_App --s3-location s3://amzn-s3-demo-
bucket/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 **`amzn-s3-demo-bucket`** bucket, and registers information with CodeDeploy about the uploaded revision.

Step 4: Deploy your Hello World application

Now you deploy the sample Hello World application revision you uploaded to Amazon S3. You use the AWS CLI or the 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 CodeDeploy
- Monitor and troubleshoot your deployment
- Verify your deployment

Deploy your application revision with CodeDeploy

You can deploy your application using the CLI or the console.

Topics

- To deploy your application revision (CLI)
- To deploy your application revision (console)

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 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) to create a service role. To get the ARN of the service role, see Get the service role ARN (CLI).

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. Before you create a deployment, the instances in your deployment group must have the CodeDeploy agent installed. You can install the agent from the command line with AWS Systems Manager with the following command:

   ```
   aws ssm create-association --name AWS-ConfigureAWSPackage
    --targets Key=tag:Name,Values=CodeDeployDemo --parameters
    action=Install,name=AWSCodeDeployAgent --schedule-expression "cron(0 2 ? * SUN
    *)"
   ```

This command creates an association in Systems Manager State Manager that will install the CodeDeploy agent and then attempt to update it at 2:00 every Sunday morning. For more information about the CodeDeploy agent, see Working with the CodeDeploy agent. For more information about Systems Manager, see What is AWS Systems Manager.

4. 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 **`amzn-s3-demo-bucket`**:

```
aws deploy create-deployment --application-name HelloWorld_App --deployment-config-
name CodeDeployDefault.OneAtATime --deployment-group-name HelloWorld_DepGroup --s3-
location bucket=amzn-s3-demo-bucket,bundleType=zip,key=HelloWorld_App.zip
```

To deploy your application revision (console)

1. Before you use the 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 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) to create a service role. To get the ARN of the service role, see Get the service role ARN (console).

2. Now that you have the ARN, you can use the CodeDeploy console to deploy your application revision.

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

3. In the navigation pane, expand **Deploy**, then choose **Applications**.
4. Choose **HelloWorld_App**.
5. On the **Deployment groups** tab, choose **Create deployment group**.

6. In **Deployment group name**, enter **`HelloWorld_DepGroup`**.
7. In **Service Role**, choose the name of the service role.
8. In **Deployment type**, choose **In-place**.
9. In **Environment configuration**, select **Amazon EC2 instances**.
10. In **Agent configuration with AWS Systems Manager**, keep the defaults.
11. In **Key**, enter **`Name`**.
12. In **Value**, enter **`CodeDeployDemo`**.
13. In **Deployment configuration**, choose **CodeDeployDefault.OneAtATime**.
14. In **Load Balancer**, clear **Enable load balancing**.
15. Choose **Create deployment group**.
16. Choose **Create deployment**.
17. In **Deployment group**, choose **HelloWorld_DepGroup**
18. In **Revision type**, choose **My application is stored in Amazon S3**, and then in **Revision location**, enter the location of the sample Hello World 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 **amzn-s3-demo-bucket** (or the name of the bucket where you uploaded your application revision).
    c. In the list of objects, choose **HelloWorld_App.zip**.
    d. In the **Overview** tab, choose **Copy path**.
    e. Return to the CodeDeploy console, and in **Revision Location**, paste the **Link** field value.
19. For **Revision file type**, choose **.zip**.
20. (Optional) Enter a comment in **Deployment description**.
21. Choose **Create deployment**. Information about your newly created deployment appears on the **Deployments** page.

Monitor and troubleshoot your deployment

Use the AWS CLI or the console to monitor and troubleshoot your deployment.

Topics

- To monitor and troubleshoot your deployment (CLI)
- To monitor and troubleshoot your deployment (console)

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.

To monitor and troubleshoot your deployment (console)

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

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 your deployment ID. After a deployment fails, a message that describes the reason for the failure appears in the deployment's details page.
2. . More information about the deployment's instances is displayed. After a deployment fails, you might be able to determine on which Amazon EC2 instances and at which step the deployment failed.
3. If you want to do more troubleshooting, you can use a technique like View Instance Details. 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.

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
```

If successful, you should see a Hello World webpage.

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 webpage's code, and then use CodeDeploy to redeploy the site. After redeployment, you should be able to see the changes on the Amazon EC2 instance.

Topics

- Modify the webpage
- Redeploy the site

Modify the webpage

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 webpage, 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 CodeDeploy</h2></div>
  <div align="center">
    <p>What to do next? Take a look through the <a href="https://aws.amazon.com/
codedeploy">CodeDeploy Documentation</a>.</p>
  </div>
</body>
</html>
```

Redeploy the site

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

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. (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, **`amzn-s3-demo-bucket`**).

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

Topics

- To redeploy the site (CLI)
- To redeploy the site (console)

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 **`amzn-s3-demo-bucket`**:

```
 aws deploy create-deployment --application-name HelloWorld_App --deployment-config-
name CodeDeployDefault.OneAtATime --deployment-group-name HelloWorld_DepGroup --s3-
location bucket=amzn-s3-demo-bucket,bundleType=zip,key=HelloWorld_App.zip
```

You can check the status of the new deployment, as described in Monitor and troubleshoot your deployment.

When CodeDeploy has redeployed the site, revisit the site in your web browser to verify that the background color and text on the webpage 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 CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. On the navigation pane, choose **Applications**.
3. In the **Applications** list, choose **HelloWorld_App**.
4. In the **Deployments** tab, choose **Create deployment**.

   a. In the **Deployment group** list, choose **HelloWorld_DepGroup**.

   b. In **Revision location**, enter the Amazon S3 link for your revision.

      To find the link value:

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

Browse to and open **amzn-s3-demo-bucket**, and then choose your revision, **`HelloWorld_App.zip`**, in the Amazon S3 console.

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.

iv. Return to the CodeDeploy console, then paste the link into **Revision location**.

v.

c. In **Revision file type**, if a message appears stating that the file type could not be detected, choose **.zip**.

d. Leave **Deployment description** blank.

e. Expand **Deployment group overrides**In the **Deployment configuration** list, choose **CodeDeployDefault.OneAtATime**, and then choose **Create deployment**.

You can check the status of the deployment as described in Monitor and troubleshoot your deployment.

When CodeDeploy has redeployed the site, revisit the site in your web browser to verify that the background color and text on the webpage 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 CloudFormation stacks (or terminate any Amazon EC2 instances, if you created them outside of CloudFormation).
- Any Amazon S3 buckets.
- The `HelloWorld_App` application in CodeDeploy.
- The AWS Systems Manager State Manager association for the CodeDeploy agent.

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

Topics

- To use clean up resources (CLI)
- To clean up resources (console)
- What's next?

To use clean up resources (CLI)

1. If you used the CloudFormation stack for this tutorial, delete the stack by calling the **delete-stack** command against the stack named **`CodeDeployDemoStack`**. This terminates 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 **`amzn-s3-demo-bucket`**. This deletes the bucket and all objects in the bucket.

   ```
   aws s3 rm s3://amzn-s3-demo-bucket --recursive --region region
   ```

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

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

4. To delete the Systems Manager State Manager association, call the **delete-association** command.

   ```
   aws ssm delete-association --assocation-id association-id
   ```

   You can get the *`association-id`* by calling the **describe-association** command.

   ```
   aws ssm describe-association --name AWS-ConfigureAWSPackage --targets
    Key=tag:Name,Values=CodeDeployDemo
   ```

5. If you did not use the 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 CloudFormation template for this tutorial, delete the associated CloudFormation stack.

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. In the search box, type the CloudFormation stack name (for example, **`CodeDeployDemoStack`**).
3. Select the box beside the stack name.
4. In the **Actions** menu, choose **Delete Stack**. This deletes the stack, terminate all accompanying Amazon EC2 instances, and delete all accompanying IAM roles.

To terminate Amazon EC2 instances you created outside of an 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, **`amzn-s3-demo-bucket`**).

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 CodeDeploy:

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, then choose **Applications**.
3. Choose **`HelloWorld_App`**.
4. Choose **Delete application**.
5. When prompted, enter **`Delete`**, and then choose **Delete**.

To delete the Systems Manager State Manager association:

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager.
2. In the navigation pane, choose **State Manager**.
3. Choose the association you created and choose **Delete**.

What's next?

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

Tutorial: Deploy an application to an on-premises instance with CodeDeploy (Windows Server, Ubuntu Server, or Red Hat Enterprise Linux)

This tutorial helps you gain experience with 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 CodeDeploy, see Working with On-Premises Instances.

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).
- To practice deploying to an Amazon EC2 instance running Windows Server, see Tutorial: Deploy a "hello, world!" application with CodeDeploy (Windows Server).

Topics

- Prerequisites
- Step 1: Configure the on-premises instance
- Step 2: Create a sample application revision
- Step 3: Bundle and upload your application revision to Amazon S3
- Step 4: Deploy your application revision
- Step 5: Verify your deployment
- Step 6: Clean up resources

Prerequisites

Before you start this tutorial, you must complete the prerequisites in Getting started with CodeDeploy, which include configuring a 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 CodeDeploy agent.

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, and then return to this page.

Install the CodeDeploy agent

After you configure your on-premises instance, follow the steps for on-premises instances in Install the CodeDeploy agent and return to this page.

Step 2: Create a sample application revision

In this step, you 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 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 that starts the CodeDeploy agent on the on-premises instance does not have permission to execute scripts.
> - The user does not have permission to create or delete folders in the locations listed in the scripts.
> - The user 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 in the Tutorial: Deploy a "hello, world!" application with CodeDeploy (Windows Server) 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 in the Tutorial: Deploy WordPress to an Amazon EC2 instance (Amazon Linux or Red Hat Enterprise Linux and Linux, macOS, or Unix) 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 CodeDeploy and CodeDeploy AppSpec file reference.

`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 CodeDeploy and Push a revision for CodeDeploy to Amazon S3 (EC2/On-Premises deployments only). (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 CodeDeploy with GitHub.

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 CodeDeploy, and then return to this page.

Step 5: Verify your deployment

To verify the deployment was successful, follow the instructions in View CodeDeploy deployment details, 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 and Troubleshoot instance issues. 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 CodeDeploy and the on-premises instance.

You can use the AWS CLI or a combination of the 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, `amzn-s3-demo-bucket`). The bucket and all objects in the bucket will be deleted.

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

To delete the application and deployment group records in 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 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, `amzn-s3-demo-bucket`), 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 CodeDeploy

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

> **Note**
>
> Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane choose **Applications**.
3. Choose the name of the application you want to delete (for example, `CodeDeploy-OnPrem-App`) and then choose **Delete application**.
4. When prompted, enter 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 CodeDeploy agent. Follow the instructions in To deregister the on-premises instance and delete the IAM user.

Tutorial: Use CodeDeploy to deploy an application to an Auto Scaling group

In this tutorial, you'll use CodeDeploy to deploy an application revision to an Auto Scaling group. Amazon EC2 Auto Scaling launches Amazon EC2 instances using predefined conditions, and then terminates those instances when they are no longer needed. Amazon EC2 Auto Scaling can help CodeDeploy scale by ensuring it always has the correct number of Amazon EC2 instances available to handle the load for deployments. For information about Amazon EC2 Auto Scaling integration with CodeDeploy, see Integrating CodeDeploy with Amazon EC2 Auto Scaling.

Topics

- Prerequisites
- Step 1: Create and configure the Auto Scaling group
- Step 2: Deploy the application to the Auto Scaling group
- Step 3: Check your results
- Step 4: Increase the number of Amazon EC2 instances in the Auto Scaling group
- Step 5: Check your results again
- Step 6: Clean up

Prerequisites

To follow along in this tutorial:

- Complete all of the steps in Getting started with CodeDeploy, 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 create your Auto Scaling group with a launch template, you must add the following permissions:
  - `ec2:RunInstances`
  - `ec2:CreateTags`
  - `iam:PassRole`

  For more information, see Step 2: Create a service role, Creating a launch template for an Auto Scaling group, and Launch template support in the *Amazon EC2 Auto Scaling User Guide*.
- Create and use a revision that is compatible with an Ubuntu Server instance and CodeDeploy. For your revision, you can do one of the following:
  - Create and use the sample revision in Step 2: Create a sample application revision in the Tutorial: Deploy an application to an on-premises instance with CodeDeploy (Windows Server, Ubuntu Server, or Red Hat Enterprise Linux) tutorial.
  - Create a revision on your own, see Working with application revisions for CodeDeploy.
- Create a Security Group named **`CodeDeployDemo-AS-SG`** with the following **Inbound rule**:
  - Type: HTTP
  - Source: Anywhere

  This is required to view your application and verify deployment success. For information on how to create a Security Group, see Creating a security group in the *Amazon EC2 user guide*.

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 Amazon EC2 Auto Scaling to add one more Amazon EC2 instance, and CodeDeploy will deploy your revision to it.

Topics

- To create and configure the Auto Scaling group (CLI)
- To create and configure the Auto Scaling group (console)

To create and configure the Auto Scaling group (CLI)

1. Call the **create-launch-template** command to create an Amazon EC2 launch template.

   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*.

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

   a. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

   b. In the navigation pane, under **Instances**, choose **Instances**, and then choose **Launch Instance**.

   c. On the **Quick Start** tab of the **Choose an Amazon Machine Image** page, note the ID of the AMI next to **Amazon Linux 2 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 CodeDeploy, choose it here instead of browsing through the **Quick Start** tab. For information about using a custom AMI with CodeDeploy and Amazon EC2 Auto Scaling, see Using a custom AMI with CodeDeploy and Amazon EC2 Auto Scaling.

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

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

   On local Linux, macOS, or Unix machines:

```
aws ec2 create-launch-template \
  --launch-template-name CodeDeployDemo-AS-Launch-Template \
```

```
  --launch-template-data file://config.json
```

The contents of the `config.json` file:

```
{
  "InstanceType":"t1.micro",
  "ImageId":"image-id",
  "IamInstanceProfile":{
    "Name":"CodeDeployDemo-EC2-Instance-Profile"
  },
  "KeyName":"key-name"
}
```

On local Windows machines:

```
aws ec2 create-launch-template --launch-template-name CodeDeployDemo-AS-Launch-
Template --launch-template-data file://config.json
```

The contents of the `config.json` file:

```
{
  "InstanceType":"t1.micro",
  "ImageId":"image-id",
  "IamInstanceProfile":{
    "Name":"CodeDeployDemo-EC2-Instance-Profile"
  },
  "KeyName":"key-name"
}
```

These commands, along with the `config.json` file, create an Amazon EC2 launch template named CodeDeployDemo-AS-Launch-Template for your Auto Scaling group that will be created in a following step based on the t1.micro Amazon EC2 instance type. Based on your input for `ImageId`, `IamInstanceProfile`, and `KeyName`, the launch template also specifies the AMI ID, the name of the instance profile associated with the IAM role to pass to instances at launch, and the Amazon EC2 key pair to use when connecting to instances.

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.

On local Linux, macOS, or Unix machines:

```
aws autoscaling create-auto-scaling-group \
  --auto-scaling-group-name CodeDeployDemo-AS-Group \
  --launch-template CodeDeployDemo-AS-Launch-Template,Version='$Latest' \
  --min-size 1 \
  --max-size 1 \
  --desired-capacity 1 \
  --availability-zones availability-zone \
  --tags Key=Name,Value=CodeDeployDemo,PropagateAtLaunch=true
```

On local Windows machines:

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name
 CodeDeployDemo-AS-Group --launch-template LaunchTemplateName=CodeDeployDemo-
AS-Launch-Template,Version="$Latest" --min-size 1 --max-size 1 --
desired-capacity 1 --availability-zones availability-zone --tags
 Key=Name,Value=CodeDeployDemo,PropagateAtLaunch=true
```

These commands create an Auto Scaling group named **`CodeDeployDemo-AS-Group`** based on the Amazon EC2 launch template named **`CodeDeployDemo-AS-Launch-Template`**. This Auto Scaling group has only one Amazon EC2 instance, and it is created in the specified Availability Zone. Each instance in this Auto Scaling group will have the tag `Name=CodeDeployDemo`. The tag will be used when installing the CodeDeploy agent later.

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`.

4. The instances in your Auto Scaling group must have the CodeDeploy agent installed to be used in CodeDeploy deployments. Install the CodeDeploy agent by calling the **create-association** command from AWS Systems Manager with the tags that were added when the Auto Scaling group was created.

```
aws ssm create-association \
  --name AWS-ConfigureAWSPackage \
  --targets Key=tag:Name,Values=CodeDeployDemo \

 --parameters action=Install, name=AWSCodeDeployAgent \
  --schedule-expression "cron(0 2 ? * SUN *)"
```

This command creates an association in Systems Manager State Manager that will install the CodeDeploy agent on all instances in the Auto Scaling group and then attempt to update it at 2:00 every Sunday morning. For more information about the CodeDeploy agent, see Working with the CodeDeploy agent. For more information about Systems Manager, see What is AWS Systems Manager.

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. Amazon EC2 Auto Scaling resources are tied to the region you specify, and CodeDeploy is supported in select regions only.
3. In the navigation bar, under **Instances**, choose **Launch Templates**.
4. Choose **Create launch template**.
5. In the **Launch template name and description** dialog, for **Launch template name**, enter **`CodeDeployDemo-AS-Launch-Template`**. Leave the defaults for the other fields.
6. In the **Amazon machine image (AMI)** dialog, click the dropdown under **AMI**, choose an AMI that works with this tutorial:

- On the **Quick Start** tab of the **AMI** dropdown, choose one of the following: **Amazon Linux 2 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 CodeDeploy, choose it here instead of browsing through the **Quick Start** tab. For information about using a custom AMI with CodeDeploy and Amazon EC2 Auto Scaling, see Using a custom AMI with CodeDeploy and Amazon EC2 Auto Scaling.

7. In **Instance type**, select the dropdown and choose **t1.micro**. You can use the search bar to find it more quickly.
8. In the **Key pair (login)** 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.
9. In the **Network settings** dialog box, choose **Virtual Public Cloud (VPC)**.

   In the **Security groups** dropdown, choose the security group you created in the tutorial's prerequisites section (**`CodeDeployDemo-AS-SG`**).
10. Expand the **Advanced details** dialog box. In the **IAM instance profile** dropdown, select the IAM role you created earlier (**`CodeDeployDemo-EC2-Instance-Profile`**) under **IAM instance profile**.

    Leave the rest of the defaults.
11. Choose **Create launch template**.
12. In the **Next steps** dialog box, choose **Create Auto Scaling group**.
13. On the **Choose launch template or configuration** page, for **Auto Scaling group name**, type **`CodeDeployDemo-AS-Group`**.
14. In the **Launch template** dialog box, your launch template (**`CodeDeployDemo-AS-Launch-Template`**) should be filled in, if not, select it from the dropdown menu. Leave the defaults and choose **Next**.
15. On the **Choose instance launch options page** page, in the **Network** section, for **VPC**, choose the default VPC. Then for **Availability Zones and subnets**, choose a default subnet. You must

create a VPC if you cannot choose the default. For more information, see Getting started with Amazon VPC.

16. In the **Instance type requirements** section, use the default setting to simplify this step. (Do not override the launch template.) For this tutorial, you will launch only On-Demand Instances using the instance type specified in your launch template.
17. Choose **Next** to go to the **Configure advanced options** page.
18. Keep the default values and choose **Next**.
19. On the **Configure group size and scaling policies** page, keep the default **Group size** values of 1. Choose **Next**.
20. Skip the step for configuring notifications, and choose **Next**.
21. On the **Add tags** page, add a tag to be used when installing the CodeDeploy agent later. Choose **Add tag**.

    a. In **Key**, enter **`Name`**.

    b. In **Value**, enter **`CodeDeployDemo`**.

    Choose **Next**.
22. Review your Auto Scaling group information on the **Review** page, then choose **Create Auto Scaling group**.
23. In the navigation bar, with **Auto Scaling Groups** selected, choose **`CodeDeployDemo-AS-Group`**, and then choose the **Instance Management** 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.
24. Install the CodeDeploy agent by following the steps in Install the CodeDeploy agent and using the `Name=CodeDeployDemo` instance tags.

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)
- To create the deployment (console)

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 2: Create a service role for CodeDeploy. The service role will give 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).

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 bucket that contains the 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.

   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 bucket that contains the 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.

   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 bucket that contains the 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.

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 bucket that contains the 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.

> **Note**
>
> Currently, 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 CodeDeploy.

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 2: Create a service role for CodeDeploy. The service role will give CodeDeploy permission to access your instances to expand (read) their tags. Before you use the 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).
2. Now that you have the service role ARN, you can use the CodeDeploy console to deploy your application revision.

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

3. In the navigation pane, expand **Deploy**, then choose **Applications**.
4. Choose **Create application**.
5. Choose **Custom application**.
6. In **Application name**, enter **`SimpleDemoApp`**.
7. In **Compute platform**, choose **EC2/On-premises**.
8. Choose **Create application**.
9. On the **Deployment groups** tab, choose **Create deployment group**.
10. In **Deployment group name**, enter **`SimpleDemoDG`**.

11. In **Service Role**, choose the name of your service role.
12. In **Deployment type**, choose **In-place**.
13. In **Environment configuration** select **Auto Scaling groups**, and then choose **`CodeDeployDemo-AS-Group`**.
14. In **Deployment configuration**, choose **CodeDeployDefault.OneAtATime**.
15. Clear **Enable load balancing**.
16. Choose **Create deployment group**.
17. In the deployment group page, choose **Create deployment**.
18. In **Revision type**, choose **My application is stored in Amazon S3**.
19. In **Revision location**, enter 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	`http://s3-us-east-2.amazonaws.com/aws-codedeploy-us-east-2/samples/latest/SampleApp_Linux.zip`
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`

Region	Location of sample application
Canada (Central) Region	http://s3-ca-central-1.amazonaws.com/aws-codedeploy-ca-central-1/samples/latest/SampleApp_Linux.zip
Europe (Ireland) Region	http://s3-eu-west-1.amazonaws.com/aws-codedeploy-eu-west-1/samples/latest/SampleApp_Linux.zip
Europe (London) Region	http://s3-eu-west-2.amazonaws.com/aws-codedeploy-eu-west-2/samples/latest/SampleApp_Linux.zip
Europe (Paris) Region	http://s3-eu-west-3.amazonaws.com/aws-codedeploy-eu-west-3/samples/latest/SampleApp_Linux.zip
Europe (Frankfurt) Region	http://s3-eu-central-1.amazonaws.com/aws-codedeploy-eu-central-1/samples/latest/SampleApp_Linux.zip
Israel (Tel Aviv) Region	https://aws-codedeploy-il-central-1.s3.il-central-1.amazonaws.com/samples/latest/SampleApp_Linux.zip
Asia Pacific (Hong Kong) Region	https://aws-codedeploy-ap-east-1.s3.ap-east-1.amazonaws.com/samples/latest/SampleApp_Linux.zip

Region	Location of sample application
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 (Melbourne) Region	`https://aws-codedeploy-ap-southeast-4.s3.ap-southeast-4.amazonaws.com/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`
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`
Europe (Ireland) Region	`http://s3-eu-west-1.amazonaws.com/aws-codedeploy-eu-west-1/samples/latest/SampleApp_Windows.zip`

Region	Location of sample application
Europe (London) Region	http://s3-eu-west-2.amazonaws.com/aws-codedeploy-eu-west-2/samples/latest/SampleApp_Windows.zip
Europe (Paris) Region	http://s3-eu-west-3.amazonaws.com/aws-codedeploy-eu-west-3/samples/latest/SampleApp_Windows.zip
Europe (Frankfurt) Region	http://s3-eu-central-1.amazonaws.com/aws-codedeploy-eu-central-1/samples/latest/SampleApp_Windows.zip
Israel (Tel Aviv) Region	https://aws-codedeploy-il-central-1.s3.il-central-1.amazonaws.com/samples/latest/SampleApp_Windows.zip
Asia Pacific (Hong Kong) Region	https://aws-codedeploy-ap-east-1.s3.ap-east-1.amazonaws.com/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
Asia Pacific (Singapore) Region	http://s3-ap-southeast-1.amazonaws.com/aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Windows.zip

Region	Location of sample application
Asia Pacific (Sydney) Region	`http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Windows.zip`
Asia Pacific (Melbourne) Region	`https://aws-codedeploy-ap-southeast-4.s3.ap-southeast-4.amazonaws.com/samples/latest/SampleApp_Windows.zip`
Asia Pacific (Mumbai) Region	`http://s3-ap-south-1.amazonaws.com/aws-codedeploy-ap-south-1/samples/latest/SampleApp_Windows.zip`
South America (São Paulo) Region	`http://s3-sa-east-1.amazonaws.com/aws-codedeploy-sa-east-1/samples/latest/SampleApp_Windows.zip`

For Ubuntu Server Amazon EC2 instances

Type the location of your custom application revision stored in Amazon S3.

20. Leave **Deployment description** blank.
21. Expand **Advanced**.
22. Choose **Create deployment**.

> **Note**
>
> If **Failed** appears instead of **Succeeded**, you may want to try some of the techniques in Monitor and troubleshoot your deployment (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 CodeDeploy installed the **SimpleDemoApp** revision on the single Amazon EC2 instance in the Auto Scaling group.

Topics

- To check the results (CLI)
- To check the results (console)

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 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 Amazon EC2 Auto Scaling adds the Amazon EC2 instance, 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 CodeDeploy to deploy a revision to a single Amazon EC2 instance in an Auto Scaling group!

Next, you add an Amazon EC2 instance to the Auto Scaling group. After Amazon EC2 Auto Scaling adds the Amazon EC2 instance, 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 instruct the Auto Scaling group to create an additional Amazon EC2 instance. After Amazon EC2 Auto Scaling creates the instance, CodeDeploy deploys your revision to it.

Topics

- To scale out the number of Amazon EC2 instances in the Auto Scaling group (CLI)
- To scale out the number of Amazon EC2 instances in the deployment group (console)

To scale out 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 out 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 CodeDeploy installed the SimpleDemoApp revision on the new instance in the Auto Scaling group.

Topics

- To check automatic deployment results (CLI)
- To check automatic deployment results (console)

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 Amazon EC2 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 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 CodeDeploy console at https://console.aws.amazon.com/codedeploy.

> **Note**
>
> Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, and then choose **Deployments**.

3. Choose the deployment ID of the deployment that Amazon EC2 Auto Scaling created.

.

4. The **Deployment** page displays information about the deployment. Normally, you would create a deployment on your own, but Amazon EC2 Auto Scaling created one on your behalf to deploy your revision to the new Amazon EC2 instance.
5. After **Succeeded** is displayed at the top of the page, verify the results on the instance. You first need to get the public DNS of the instance:
6. In the Amazon EC2 navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**, and then choose the **`CodeDeployDemo-AS-Group`** entry.
7. On the **Instances** tab, choose the ID of the new Amazon EC2 instance.

8. 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 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 CodeDeploy deployment component records.

Topics

- To clean up resources (CLI)
- To clean up resources (console)

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 template by calling the **delete-launch-template** command against the launch configuration named **`CodeDeployDemo-AS-Launch-Template`**:

   ```
   aws ec2 delete-launch-template --launch-template-name CodeDeployDemo-AS-Launch-
   Template
   ```

3. Optionally, delete the application from 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
```

4. To delete the Systems Manager State Manager association, call the **delete-association** command.

```
aws ssm delete-association --assocation-id association-id
```

You can get the *association-id* by calling the **describe-association** command.

```
aws ssm describe-association --name AWS-ConfigureAWSPackage --targets
 Key=tag:Name,Values=CodeDeployDemo
```

To clean up resources (console)

To delete the Auto Scaling group, which also terminates the 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 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**.

(Optional) To delete the launch template:

1. In the navigation bar, under **Auto Scaling**, choose **Launch Configurations**, and then choose **`CodeDeployDemo-AS-Launch-Template`**.
2. Choose **Actions**, choose **Delete launch configuration**, and then choose **Yes, Delete**.

1. Optionally, delete the application from CodeDeploy. This will also delete all associated deployment, deployment group, and revision records. Open the CodeDeploy console at https://console.aws.amazon.com/codedeploy.
2. Sign in to the AWS Management Console and open the CodeDeploy console at https://console.aws.amazon.com/codedeploy.

> **Note**
>
> Sign in with the same user that you set up in Getting started with CodeDeploy.

In the navigation pane, expand **Deploy**, then choose **Applications**.

3. In the list of applications, choose **SimpleDemoApp**.
4. On the **Application details** page, choose **Delete application**.
5. When prompted, enter **`Delete`**, and then choose **Delete**.

To delete the Systems Manager State Manager association:

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager.
2. In the navigation pane, choose **State Manager**.
3. Choose the association you created and choose **Delete**.

Tutorial: Use CodeDeploy to deploy an application from GitHub

In this tutorial, you use 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 CodeDeploy, see Integrating CodeDeploy with GitHub.

> **Note**
>
> You can also use 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 in Tutorial: Deploy an application to an on-premises instance with CodeDeploy (Windows Server, Ubuntu Server, or Red Hat Enterprise Linux), or you can create a revision compatible with an Ubuntu Server instance and CodeDeploy. To create your own revision, see Plan a revision for CodeDeploy and Add an application specification file to a revision for CodeDeploy.

Topics

- Prerequisites
- Step 1: Set up a GitHub account
- Step 2: Create a GitHub repository
- Step 3: Upload a sample application to your GitHub repository
- Step 4: Provision an instance
- Step 5: Create an application and deployment group
- Step 6: Deploy the application to the instance
- Step 7: Monitor and verify the deployment
- Step 8: Clean up

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 CodeDeploy, 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.

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.

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, enter **`CodeDeployGitHubDemo`**.
   - Select **Public**.

     > **Note**
     >
     > Selecting the default **Public** option means that anyone can see this repository. You can select the **Private** option to limit who can see and commit to the repository.

   - 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 CodeDeploy and Add an application specification file to a revision for CodeDeploy.
> - 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.
> 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 CodeDeploy. For more information, see Plan a revision for CodeDeploy and Add an application specification file to a revision for CodeDeploy.

Topics

- Push a sample revision from a local Linux, macOS, or Unix machine
- Push a sample revision from a local Windows machine

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 Europe (Ireland) region
- `aws s3 cp s3://aws-codedeploy-eu-west-2/samples/latest/SampleApp_Linux.zip . --region eu-west-2` for the Europe (London) region
- `aws s3 cp s3://aws-codedeploy-eu-west-3/samples/latest/SampleApp_Linux.zip . --region eu-west-3` for the Europe (Paris) region
- `aws s3 cp s3://aws-codedeploy-eu-central-1/samples/latest/SampleApp_Linux.zip . --region eu-central-1` for the Europe (Frankfurt) Region
- `aws s3 cp s3://aws-codedeploy-il-central-1/samples/latest/SampleApp_Linux.zip . --region il-central-1` for the Israel (Tel Aviv) Region
- `aws s3 cp s3://aws-codedeploy-ap-east-1/samples/latest/SampleApp_Linux.zip . --region ap-east-1` for the Asia Pacific (Hong Kong) 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-southeast-4/samples/latest/SampleApp_Linux.zip . --region ap-southeast-4` for the Asia Pacific (Melbourne) 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 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 Europe (Ireland) region
- `aws s3 cp s3://aws-codedeploy-eu-west-2/samples/latest/SampleApp_Windows.zip . --region eu-west-2` for the Europe (London) region
- `aws s3 cp s3://aws-codedeploy-eu-west-3/samples/latest/SampleApp_Windows.zip . --region eu-west-3` for the Europe (Paris) region

- `aws s3 cp s3://aws-codedeploy-eu-central-1/samples/latest/SampleApp_Windows.zip . --region eu-central-1` for the Europe (Frankfurt) Region
- `aws s3 cp s3://aws-codedeploy-il-central-1/samples/latest/SampleApp_Windows.zip . --region il-central-1` for the Israel (Tel Aviv) Region
- `aws s3 cp s3://aws-codedeploy-ap-east-1/samples/latest/SampleApp_Windows.zip . --region ap-east-1` for the Asia Pacific (Hong Kong) 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-southeast-4/samples/latest/SampleApp_Windows.zip . --region ap-southeast-4` for the Asia Pacific (Melbourne) 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 CodeDeploy. For information see Operating systems supported by the CodeDeploy agent. (If you already have an instance configured for use in CodeDeploy deployments, skip to the next step.)

To provision an instance

1. Follow the instructions in Launch an Amazon EC2 instance (console) to provision an instance.
2. When launching the instance, remember to specify a tag on the **Add tags** page. For details on how to specify the tag, see Launch an Amazon EC2 instance (console).

To verify that the CodeDeploy agent is running on the instance

- Follow the instructions in Verify the CodeDeploy agent is running to verify that the agent is running.

After you have successfully provisioned the instance and verified the CodeDeploy agent is running, go to the next step.

Step 5: Create an application and deployment group

In this step, you will use the 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 CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, then choose **Applications**.

3. Choose **Create application**, and then select **Custom application**.
4. In **Application name**, enter **`CodeDeployGitHubDemo-App`**.
5. In **Compute Platform**, choose **EC2/On-premises**.
6. Choose **Create application**.
7. On the **Deployment groups** tab, choose **Create deployment group**.
8. In **Deployment group name**, enter **`CodeDeployGitHubDemo-DepGrp`**.
9. In **Service role**, choose the name of your CodeDeploy service role that you created in Create a service role for CodeDeploy.
10. In **Deployment type**, choose **In-place**.
11. In **Environment configuration**, depending on the type of instance you are using, choose **Amazon EC2 instances** or **On-premises instances**. For **Key** and **Value**, enter the instance tag key and value that was applied to your instance as part of Step 4: Provision an instance.
12. In **Deployment configuration**, choose **CodeDeployDefault.AllatOnce**.
13. In **Load Balancer**, clear **Enable load balancing**.
14. Expand **Advanced**.
15. In **Alarms**, select **Ignore alarm configuration**.
16. Choose **Create deployment group**, and continue to the next step.

Create an application and deployment group (CLI)

1. Call the **create-application** command to create an application in 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.
   - 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.

- 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.
- 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.
- *service-role-arn* is the service role ARN for the service role that you created in Create a service role for CodeDeploy. (Follow the instructions in Get the service role ARN (CLI) 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 use the 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 **Deployment group details** page, choose **Create deployment**.
2. In **Deployment group**, choose **`CodeDeployGitHubDemo-DepGrp`**.
3. In **Revision type**, choose **GitHub**.

4. In **Connect to GitHub**, do one of the following:

   - To create a connection for CodeDeploy applications to a GitHub account, sign out of GitHub in a separate web browser tab. In **GitHub account**, enter a name to identify this connection, and then choose **Connect to GitHub**. The webpage prompts you to authorize 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 CodeDeploy **Create deployment** page, in **Repository name**, enter 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 enter, or if you want to specify a different repository:

   a. In a separate web browser tab, go to your GitHub dashboard.

   b. 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. Enter this value into **Repository name**.

   > **Note**
   >
   > If the target repository name is not displayed in **Your repositories**, use the **Search GitHub** box to find the target repository and GitHub user or organization name.

8. In the **Commit ID** box, enter the ID of the commit associated with the push of your application revision to GitHub.

   If you are unsure of the value to enter:

   a. In a separate web browser tab, go to your GitHub dashboard.

b. In **Your repositories**, choose **CodeDeployGitHubDemo**.

c. 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.)

d. 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 CodeDeploy permission to use your GitHub user account to interact with GitHub for the `CodeDeployGitHubDemo-App` application. Currently, you must use the CodeDeploy console to do this.

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

> **Note**
>
> Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, then choose **Applications**.

3. Choose **CodeDeployGitHubDemo-App**.

4. On the **Deployments** tab, choose **Create deployment**.

> **Note**
>
> You will not be creating a new deployment. This is currently the only way to give CodeDeploy permission to interact with GitHub on behalf of your GitHub user account.

5. From **Deployment group**, choose **CodeDeployGitHubDemo-DepGrp**.

6. In **Revision type**, choose **GitHub**.

7. In **Connect to GitHub**, do one of the following:

- To create a connection for 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 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 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 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. In the navigation pane, expand **Deploy**, and then 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) 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.

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) 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 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 CloudFormation stack (if you used the CloudFormation template to create an Amazon EC2 instance)

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. In the **Stacks** column, choose the stack starting with `CodeDeploySampleStack`.
3. Choose **Delete**.
4. When prompted, choose **Delete stack**. The Amazon EC2 instance and the associated IAM instance profile and service role are 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 --no-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 CodeDeploy deployment component records

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, then choose **Applications**.
3. Choose **CodeDeployGitHubDemo-App**.
4. Choose **Delete application**.
5. When prompted, enter **`Delete`**, and then choose **Delete**.

To delete your GitHub repository

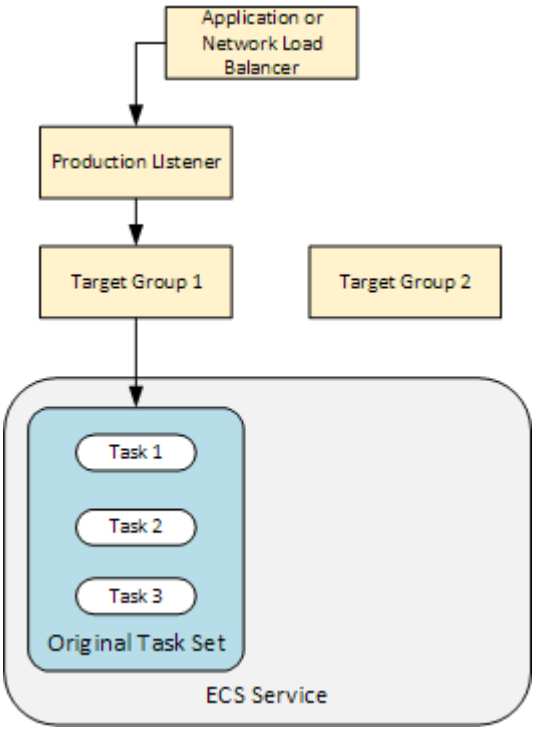
See Deleting a repository in GitHub help.

Tutorial: Deploy an application into Amazon ECS

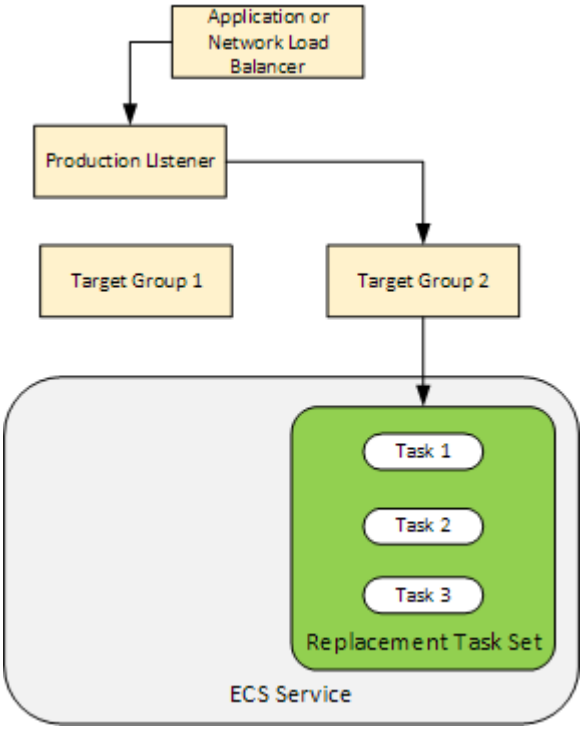
In this tutorial, you learn how to deploy an application into Amazon ECS using CodeDeploy. You start with an application you already created and deployed into Amazon ECS. The first step is to update your application by modifying its task definition file with a new tag. Next, you use CodeDeploy to deploy the update. During deployment, CodeDeploy installs your update into a new,

replacement task set. Then, it shifts production traffic from the original version of your Amazon ECS application, which is in its original task set, to the updated version in the replacement task set.

During an Amazon ECS deployment, CodeDeploy uses a load balancer that is configured with two target groups and one production traffic listener. The following diagram shows how the load balancer, production listener, target groups, and your Amazon ECS application are related before the deployment starts. This tutorial uses an Application Load Balancer. You can also use a Network Load Balancer.



After a successful deployment, the production traffic listener serves traffic to your new replacement task set and the original task set is terminated. The following diagram shows how your resources are related after a successful deployment. For more information, see [What happens during an Amazon ECS deployment](#).



For information about how to use the AWS CLI to deploy an application into Amazon ECS, see Tutorial: Creating a service using a blue/green deployment. For information about how to use CodePipeline to detect and automatically deploy changes to an Amazon ECS service with CodeDeploy, see Tutorial: Create a pipeline with an Amazon ECR source and ECS-to-CodeDeploy deployment.

After you complete this tutorial, you can use the CodeDeploy application and deployment group you created to add a deployment validation test in Tutorial: Deploy an Amazon ECS service with a validation test.

Topics

- Prerequisites
- Step 1: Update your Amazon ECS application
- Step 2: Create the AppSpec file
- Step 3: Use the CodeDeploy console to deploy your application
- Step 4: Clean up

Prerequisites

To complete this tutorial, you must first:

- Complete steps 2 and 3 in Getting started with CodeDeploy.
- Create an Application Load Balancer configured with two target groups and one listener. For information about creating a load balancer using the console, see Set up a load balancer, target groups, and listeners for CodeDeploy Amazon ECS deployments. For information about creating a load balancer using the AWS CLI, see Step 1: Create an Application Load Balancer in the *Amazon Elastic Container Service User Guide*. When you create your load balancer, make a note of the following for this tutorial:
  - The name of your load balancer.
  - The names of your target groups.
  - The port used by your load balancer's listener.
- Create an Amazon ECS cluster and service. For more information, see steps 2, 3, and 4 in Tutorial: Creating a service using a blue/green deployment in the *Amazon Elastic Container Service User Guide*. Make a note of the following for this tutorial:
  - The name of your Amazon ECS cluster.
  - The ARN of the the task definition used by your Amazon ECS service.
  - The name of the the container used by your Amazon ECS service.
- Create an Amazon S3 bucket for your AppSpec file.

Step 1: Update your Amazon ECS application

In this section, you update your Amazon ECS application with a new revision of its task definition. The updated revision adds a new key and tag pair. In Step 3: Use the CodeDeploy console to deploy your application, you deploy the updated version of your Amazon ECS application.

To update your task definition

1. Open the console at https://console.aws.amazon.com/ecs/v2.
2. In the navigation pane, choose **Task Definitions**.
3. Choose the task definition used by your Amazon ECS service.
4. Select the task definition revision, and then choose **Create new revision**, **Create new revision**.
5. For this tutorial, make a small update to the task definition by adding a tag. At the bottom of the page, under **Tags**, create a new tag by entering a new key and value pair.
6. Choose **Create**.

Your task definition's revision number is incremented by one.

7. Choose the **JSON** tab. Make a note of the following because you need this information in the next step.

   - The value for `taskDefinitionArn`. Its format is `arn:aws:ecs:`*`aws-region`*`:`*`account-id`*`:task-definition/`*`task-definition-family`*`:`*`task-definition-revision`*. This is the ARN of your updated task definition.
   - In the `containerDefinitions` element, the value for `name`. This is the name of your container.
   - In the `portMappings` element, the value for `containerPort`. This is the port for your container.

Step 2: Create the AppSpec file

In this section, you create your AppSpec file and upload it to the Amazon S3 bucket you created in the Prerequisites section. The AppSpec file for an Amazon ECS deployment specifies your task definition, container name, and container port. For more information, see AppSpec File example for an Amazon ECS deployment and AppSpec 'resources' section for Amazon ECS deployments.

To create your AppSpec file

1. If you want to create your AppSpec file using YAML, create a file named `appspec.yml`. If you want to create your AppSpec file using JSON, create a file named `appspec.json`.
2. Choose the appropriate tab, depending on if you use YAML or JSON for your AppSpec file, and copy its content into the AppSpec file you just created. For the `TaskDefinition` property, use the task definition ARN you noted in Step 1: Update your Amazon ECS application section.

   JSON AppSpec

```
{
  "version": 0.0,
  "Resources": [
    {
      "TargetService": {
        "Type": "AWS::ECS::Service",
        "Properties": {
          "TaskDefinition": "arn:aws:ecs:aws-region-id:aws-account-id:task-
definition/ecs-demo-task-definition:revision-number",
```

```
          "LoadBalancerInfo": {
            "ContainerName": "your-container-name",
            "ContainerPort": your-container-port
          }
        }
      }
    }
  ]
}
```

YAML AppSpec

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "arn:aws:ecs:aws-region-id:aws-account-id:task-
definition/ecs-demo-task-definition:revision-number"
        LoadBalancerInfo:
          ContainerName: "your-container-name"
          ContainerPort: your-container-port
```

> **Note**
>
> Your replacement task set inherits the subnet, security group, platform version, and assigned public IP values from your original task set. You can override these values for your replacement task set by setting their optional properties in your AppSpec file. For more information, see AppSpec 'resources' section for Amazon ECS deployments and AppSpec File example for an Amazon ECS deployment.

3. Upload your AppSpec file to the S3 bucket you created as a prerequisite for this tutorial.

Step 3: Use the CodeDeploy console to deploy your application

In this section, you create a CodeDeploy application and deployment group to deploy your updated application into Amazon ECS. During deployment, CodeDeploy shifts the production traffic for your application to its new version in a new, replacement task set. To complete this step, you need the following items:

- Your Amazon ECS cluster name.
- Your Amazon ECS service name.
- Your Application Load Balancer name.
- Your production listener port.
- Your target group names.
- The name of the S3 bucket you created.

To create a CodeDeploy application

1. Sign in to the AWS Management Console and open the CodeDeploy console at https://console.aws.amazon.com/codedeploy/.
2. Choose **Create application**.
3. In **Application name**, enter **`ecs-demo-codedeploy-app`**.
4. In **Compute platform**, choose **Amazon ECS**.
5. Choose **Create application**.

To create a CodeDeploy deployment group

1. On the **Deployment groups** tab of your application page, choose **Create deployment group**.
2. In **Deployment group name**, enter **`ecs-demo-dg`**.
3. In **Service role**, choose a service role that grants CodeDeploy access to Amazon ECS. For more information, see Identity and access management for AWS CodeDeploy.
4. In **Environment configuration**, choose your Amazon ECS cluster name and service name.
5. From **Load balancers**, choose the name of the load balancer that serves traffic to your Amazon ECS service.
6. From **Production listener port**, choose the port and protocol for the listener that serves production traffic to your Amazon ECS service (for example, **HTTP: 80**). This tutorial does not include an optional test listener, so do not choose a port from **Test listener port**.
7. From **Target group 1 name** and **Target group 2 name**, choose two different target groups to route traffic during your deployment. Make sure that these are the target groups you created for your load balancer. It does not matter which is used for target group 1 and which is used for target group 2.
8. Choose **Reroute traffic immediately**.

9. For **Original revision termination**, choose 0 days, 0 hours, and 5 minutes. This lets you see your deployment complete faster than if you use the default (1 hour).

Environment configuration

Choose an ECS cluster name

ecs-tutorial-cluster

Choose an ECS service name

ecs-demo-service

Load balancers

Choose a load balancer

ecs-demo-alb

Production listener port

HTTP: 80

Test listener port - *optional*

A test listener is required if you want to test your replacement version before traffic reroutes to it

Target group 1 name

ecs-demo-tg-1

Target group 2 name

ecs-demo-tg-2

Deployment settings

Traffic rerouting

Choose whether traffic reroutes to the replacement environment immediately or waits for you to start the rerouting process

- (●) Reroute traffic immediately
- ( ) Specify when to reroute traffic

Deployment Configuration

CodeDeployDefault.ECSALLAtOnce

Original revision termination

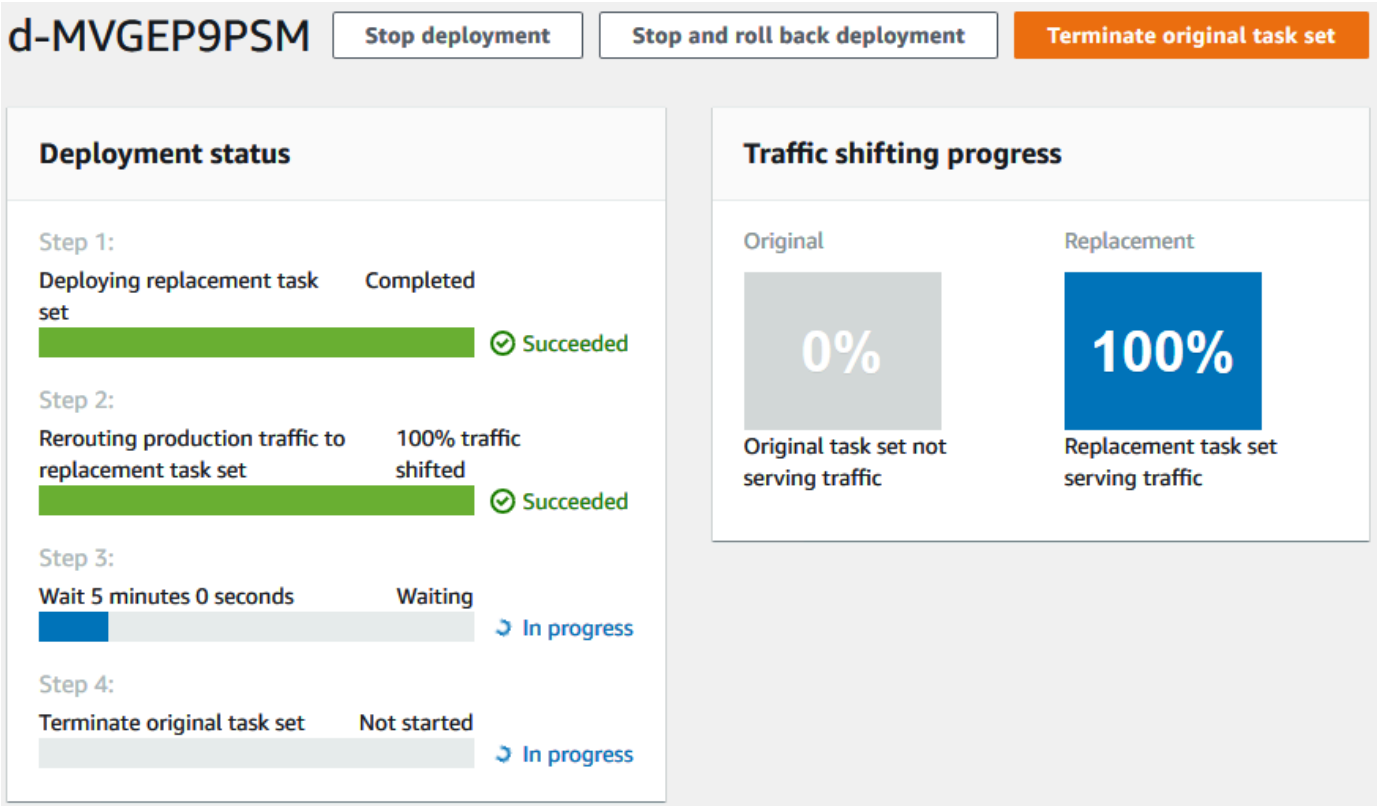
Specify how long CodeDeploy waits before it terminates the original task set. After termination starts, you cannot rollback manually or automatically

Days	Hours	Minutes
0	0	5

10. Choose **Create deployment group**.

To deploy your Amazon ECS application

1. From your deployment group console page, choose **Create deployment**.
2. For **Deployment group**, choose **ecs-demo-dg**.
3. For **Revision type**, choose **My application is stored in Amazon S3**. In **Revision location**, enter the name of your S3 bucket.
4. For **Revision file type**, choose **.json** or **.yaml**, as appropriate.
5. (Optional) In **Deployment description**, enter a description for your deployment.
6. Choose **Create deployment**.
7. In **Deployment status**, you can monitor your deployment. After 100% of production traffic is routed to the replacement task set and before the five-minute wait time expires, you can choose **Terminate original task set** to immediately terminate the original task set. If you do not choose **Terminate original task set**, the original task set terminates after the five-minute wait time you specified expires.



Step 4: Clean up

The next tutorial, Tutorial: Deploy an Amazon ECS service with a validation test, builds on this tutorial and uses the CodeDeploy application and deployment group you created. If you want to follow the steps in that tutorial, skip this step and do not delete the resources you created.

> **Note**
>
> Your AWS account does not incur charges for the CodeDeploy resources you created.

The resource names in these steps are the names suggested in this tutorial (for example, **`ecs-demo-codedeploy-app`** for the name of your CodeDeploy application). If you used different names, then be sure to use those during your cleanup.

1. Use the delete-deployment-group command to delete the CodeDeploy deployment group.

```
aws deploy delete-deployment-group --application-name ecs-demo-codedeploy-app --
deployment-group-name ecs-demo-dg --region aws-region-id
```

2. Use the delete-application command to delete the CodeDeploy application.

```
aws deploy delete-application --application-name ecs-demo-codedeploy-app --
region aws-region-id
```

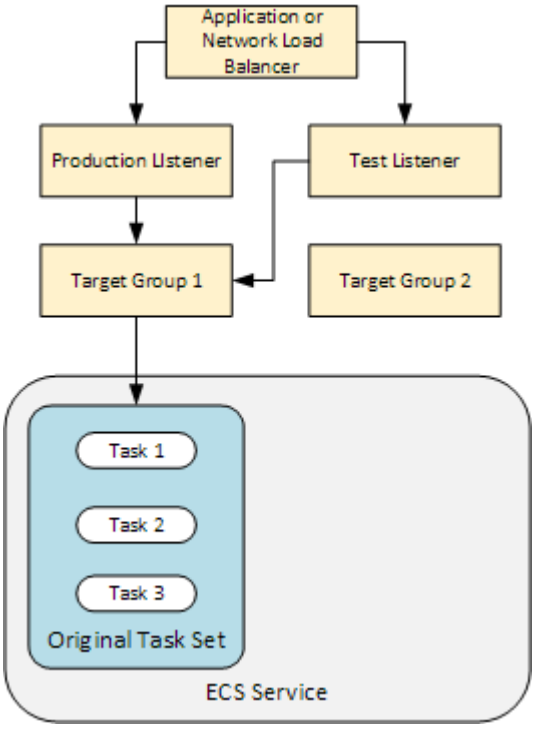
Tutorial: Deploy an Amazon ECS service with a validation test

In this tutorial, you learn how to use a Lambda function to validate part of the deployment of an updated Amazon ECS application. This tutorial uses the CodeDeploy application, CodeDeploy deployment group, and the Amazon ECS application you used in Tutorial: Deploy an application into Amazon ECS. Complete that tutorial before starting this one.

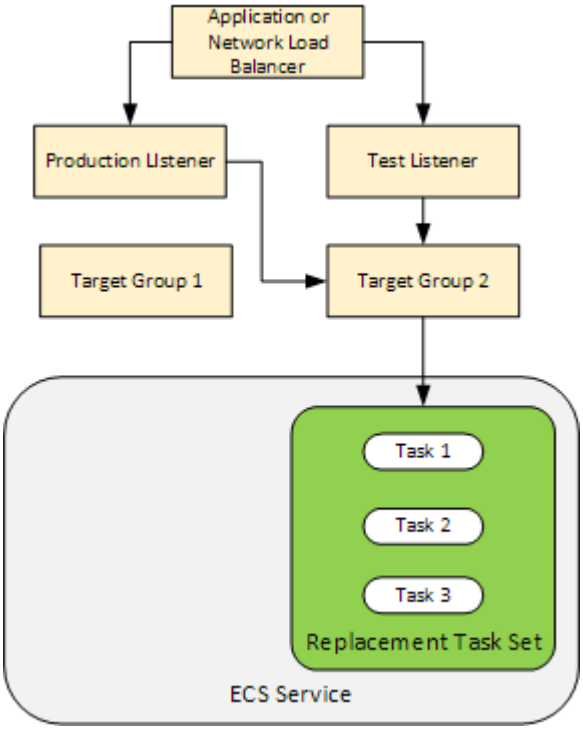
To add validation test, you first implement the test in a Lambda function. Next, in your deployment AppSpec file, you specify the Lambda function for the lifecycle hook you want to test. If a validation test fails, the deployment stops, is rolled back, and marked failed. If the test succeeds, the deployment continues to the next deployment lifecycle event or hook.

During an Amazon ECS deployment with validation tests, CodeDeploy uses a load balancer that is configured with two target groups: one production traffic listener and one test traffic listener.

The following diagram shows how the load balancer, production and test listeners, target groups, and your Amazon ECS application are related before the deployment starts. This tutorial uses an Application Load Balancer. You can also use a Network Load Balancer.



During an Amazon ECS deployment, there are five lifecycle hooks for testing. This tutorial implements one test during the third lifecycle deployment hook, `AfterAllowTestTraffic`. For more information, see List of lifecycle event hooks for an Amazon ECS deployment. After a successful deployment, the production traffic listener serves traffic to your new replacment task set and the original task set is terminated. The following diagram shows how your resources are related after a successful deployment. For more information, see What happens during an Amazon ECS deployment.



> **Note**
>
> Completing this tutorial might result in charges to your AWS account. These include possible charges for CodeDeploy, AWS Lambda, and CloudWatch. For more information, see AWS CodeDeploy pricing, AWS Lambda pricing, and Amazon CloudWatch pricing.

Topics

- Prerequisites
- Step 1: Create a test listener
- Step 2: Update your Amazon ECS application
- Step 3: Create a lifecycle hook Lambda function
- Step 4: Update your AppSpec file
- Step 5: Use the CodeDeploy console to deploy your Amazon ECS service
- Step 6: View your Lambda hook function output in CloudWatch Logs
- Step 7: Clean up

Prerequisites

To successfully complete this tutorial, you must first:

- Complete the prerequisites in Prerequisites for Tutorial: Deploy an application into Amazon ECS.
- Complete the steps in Tutorial: Deploy an application into Amazon ECS. Make a note of the following:
  - The name of your load balancer.
  - The names of your target groups.
  - The port used by your load balancer's listener.
  - The ARN of your load balancer. You use this to create a new listener.
  - The ARN of one of your target groups. You use this to create a new listener.
  - The CodeDeploy application and deployment group you create.
  - The AppSpec file you create that is used by your CodeDeploy deployment. You edit this file in this tutorial.

Step 1: Create a test listener

An Amazon ECS deployment with validation tests requires a second listener. This listener is used to serve test traffic to your updated Amazon ECS application in a replacement task set. Your validation tests run against the test traffic.

The listener for your test traffic can use either of your target groups. Use the create-listener AWS CLI command to create a second listener with a default rule that forwards test traffic to port 8080. Use the ARN of your load balancer and the ARN of one of your target groups.

```
aws elbv2 create-listener --load-balancer-arn your-load-balancer-arn \
--protocol HTTP --port 8080 \
--default-actions Type=forward,TargetGroupArn=your-target-group-arn --region your-aws-region
```

Step 2: Update your Amazon ECS application

In this section, you update your Amazon ECS application to use a new revision of its task definition. You create the new revision and add a minor update to it by adding a tag.

To update your task definition

1. Open the Amazon ECS classic console at https://console.aws.amazon.com/ecs/.
2. In the navigation pane, choose **Task Definitions**.
3. Select the check box for the task definition used by your Amazon ECS service.
4. Choose **Create new revision**.
5. Make a small update to the task definition by adding a tag. At the bottom of the page, in **Tags**, create a new tag by entering a new key and value pair.
6. Choose **Create**. You should see that your task definition's revision number has been incremented by one.
7. Choose the **JSON** tab. Make a note of the value for `taskDefinitionArn`. Its format is `arn:aws:ecs:`*`aws-region`*`:` *`account-id`*`:task-definition/`*`task-definition-family`*`:` *`task-definition-revision`*. This is the ARN of your updated task definition.

Step 3: Create a lifecycle hook Lambda function

In this section, you implement one Lambda function for your Amazon ECS deployment's `AfterAllowTestTraffic` hook. The Lambda function runs a validation test before the updated Amazon ECS application is installed. For this tutorial, the Lambda function returns `Succeeded`. During a real world deployment, validation tests return `Succeeded` or `Failed`, depending on the result of the validation test. Also during a real world deployment, you might implement a Lambda test function for one or more of the other Amazon ECS deployment lifecycle event hooks (`BeforeInstall`, `AfterInstall`, `BeforeAllowTraffic`, and `AfterAllowTraffic`). For more information, see List of lifecycle event hooks for an Amazon ECS deployment.

An IAM role is required to create your Lambda function. The role grants the Lambda function permission to write to CloudWatch Logs and set the status of a CodeDeploy lifecycle hook.

To create an IAM role

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. From the navigation pane, choose **Roles**, and then choose **Create role**.
3. Create a role with the following properties:
   - **Trusted entity**: **AWS Lambda**.

- **Permissions**: **AWSLambdaBasicExecutionRole**. This grants your Lambda function permission to write to CloudWatch Logs.
- **Role name**: **`lambda-cli-hook-role`**.

For more information, see Create an AWS Lambda execution role.

4. Attach the permission `codedeploy:PutLifecycleEventHookExecutionStatus` to the role you created. This grants your Lambda functions permission to set the status of a CodeDeploy lifecycle hook during a deployment. For more information, see Adding IAM identity permissions in the *AWS Identity and Access Management User Guide* and PutLifecycleEventHookExecutionStatus in the *CodeDeploy API Reference*.

To create an `AfterAllowTestTraffic` hook Lambda function

1. Create a file named `AfterAllowTestTraffic.js` with the following contents.

```
'use strict';

 const AWS = require('aws-sdk');
 const codedeploy = new AWS.CodeDeploy({apiVersion: '2014-10-06'});

 exports.handler = (event, context, callback) => {

  console.log("Entering AfterAllowTestTraffic hook.");

  // Read the DeploymentId and LifecycleEventHookExecutionId from the event payload
  var deploymentId = event.DeploymentId;
  var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;
  var validationTestResult = "Failed";

  // Perform AfterAllowTestTraffic validation tests here. Set the test result
  // to "Succeeded" for this tutorial.
  console.log("This is where AfterAllowTestTraffic validation tests happen.")
  validationTestResult = "Succeeded";

  // Complete the AfterAllowTestTraffic hook by sending CodeDeploy the validation
 status
  var params = {
   deploymentId: deploymentId,
   lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
   status: validationTestResult // status can be 'Succeeded' or 'Failed'
```

```
  };

  // Pass CodeDeploy the prepared validation test results.
  codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data) {
   if (err) {
    // Validation failed.
    console.log('AfterAllowTestTraffic validation tests failed');
    console.log(err, err.stack);
    callback("CodeDeploy Status update failed");
   } else {
    // Validation succeeded.
    console.log("AfterAllowTestTraffic validation tests succeeded");
    callback(null, "AfterAllowTestTraffic validation tests succeeded");
   }
  });
 }
```

2. Create a Lambda deployment package.

```
zip AfterAllowTestTraffic.zip AfterAllowTestTraffic.js
```

3. Use the `create-function` command to create a Lambda function for your `AfterAllowTestTraffic` hook.

```
aws lambda create-function --function-name AfterAllowTestTraffic \
       --zip-file fileb://AfterAllowTestTraffic.zip \
       --handler AfterAllowTestTraffic.handler \
       --runtime nodejs10.x \
       --role arn:aws:iam::aws-account-id:role/lambda-cli-hook-role
```

4. Make a note of your Lambda function ARN in the `create-function` response. You use this ARN when you update your CodeDeploy deployment's AppSpec file in the next step.

Step 4: Update your AppSpec file

In this section, you update your AppSpec file with a `Hooks` section. In the `Hooks` section, you specify a Lambda function for the `AfterAllowTestTraffic` lifecycle hook.

To update your AppSpec file

1. Open the AppSpec file file you created in Step 2: Create the AppSpec file of the Tutorial: Deploy an application into Amazon ECS.
2. Update the `TaskDefinition` property with the task definition ARN you noted in Step 2: Update your Amazon ECS application.
3. Copy and paste the `Hooks` section into your AppSpec file file. Update the ARN after `AfterAllowTestTraffic` with the ARN of the Lambda function that you noted in Step 3: Create a lifecycle hook Lambda function.

   JSON AppSpec

```
{
  "version": 0.0,
  "Resources": [
    {
      "TargetService": {
        "Type": "AWS::ECS::Service",
        "Properties": {
          "TaskDefinition": "arn:aws:ecs:aws-region-id:aws-account-id::task-
definition/ecs-demo-task-definition:revision-number",
          "LoadBalancerInfo": {
            "ContainerName": "sample-website",
            "ContainerPort": 80
          }
        }
      }
    }
  ],
  "Hooks": [
    {
      "AfterAllowTestTraffic": "arn:aws:lambda:aws-region-id:aws-account-
id:function:AfterAllowTestTraffic"
    }
  ]
}
```

   YAML AppSpec

```
version: 0.0
Resources:
```

```
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "arn:aws:ecs:aws-region-id:aws-account-id::task-
definition/ecs-demo-task-definition:revision-number"
        LoadBalancerInfo:
          ContainerName: "sample-website"
          ContainerPort: 80
Hooks:
  - AfterAllowTestTraffic: "arn:aws:lambda:aws-region-id:aws-account-
id:function:AfterAllowTestTraffic"
```

4. Save your AppSpec file and upload to its S3 bucket.

Step 5: Use the CodeDeploy console to deploy your Amazon ECS service

In this section, you update your deployment group by specifying the port for your test listener. This is the listener you created in Step 1: Create a test listener. During deployment, CodeDeploy runs your validation test during the `AfterAllowTestTraffic` deployment lifecycle hook using test traffic served to your replacement task set using the test listener. Your validation test returns the result `Succeeded`, so the deployment proceeds with the next deployment lifecycle event. In a real world scenario, your test function returns `Succeeded` or `Failed`.

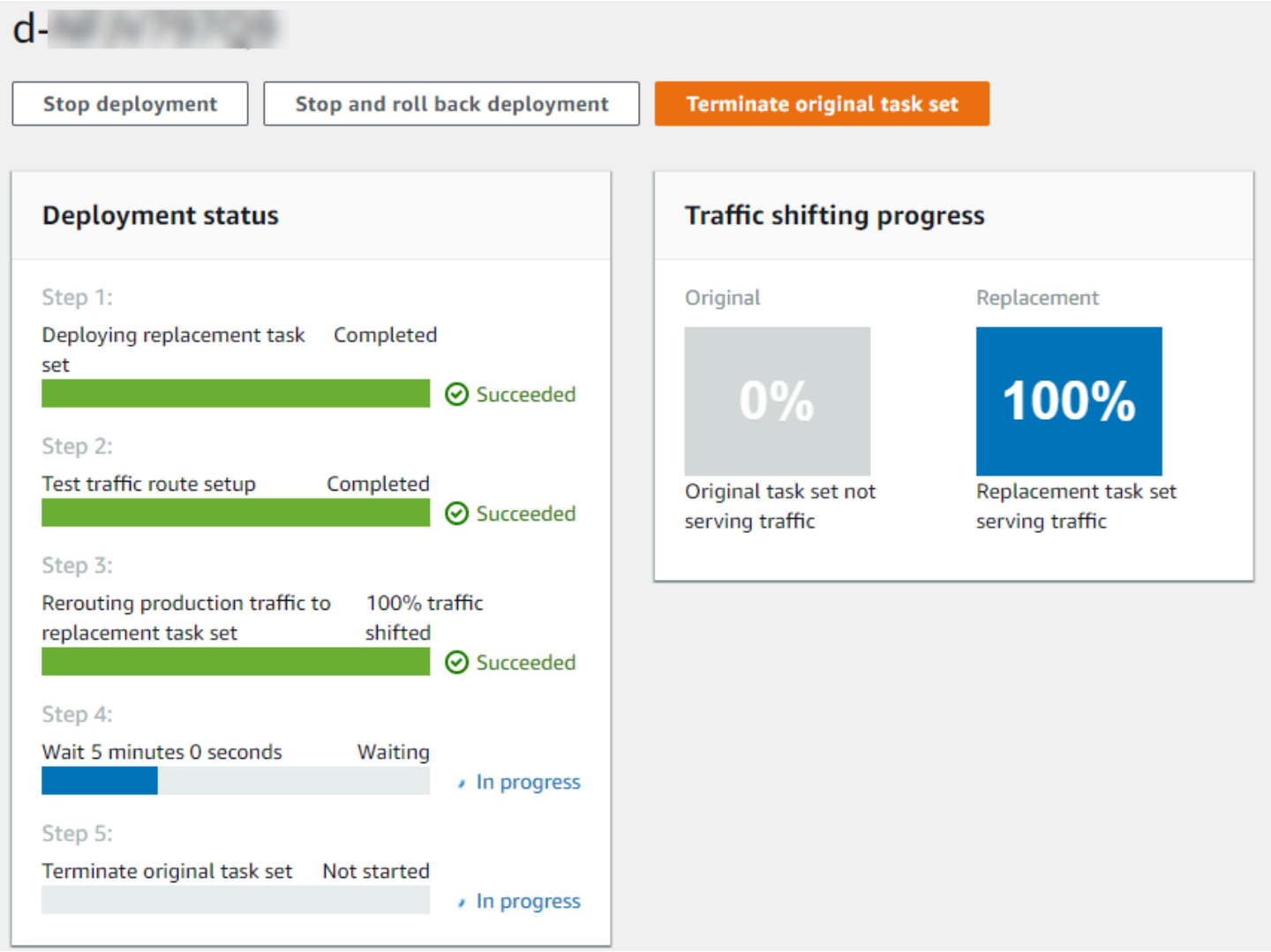
To add a test listener to your deployment group

1. Sign in to the AWS Management Console and open the CodeDeploy console at https://console.aws.amazon.com/codedeploy/.
2. From the navigation pane, choose **Applications**.
3. Choose the application you created in Tutorial: Deploy an application into Amazon ECS. If you used the suggested name, it is **ecs-demo-codedeploy-app**.
4. In **Deployment groups**, choose the deployment group you created in Tutorial: Deploy an application into Amazon ECS. If you used the suggested name, it is **ecs-demo-dg**.
5. Choose **Edit**.
6. From **Test listener port**, choose the port and protocol for the test listener you created earlier in this tutorial. This should be **HTTP: 8080**.
7. Choose **Save changes**.

To deploy your Amazon ECS application

1. From your deployment group console page, choose **Create deployment**.
2. For **Deployment group**, choose **ecs-demo-dg**.
3. For **Revision type**, choose **My application is stored in Amazon S3**. In **Revision location**, enter the name of your S3 bucket and AppSpec file (for example, **`s3://my-s3-bucket/appspec.json`**).
4. For **Revision file type**, choose **.json** or **.yaml** as appropriate.
5. (Optional) In **Deployment description**, enter a description for your deployment.
6. Choose **Create deployment**.

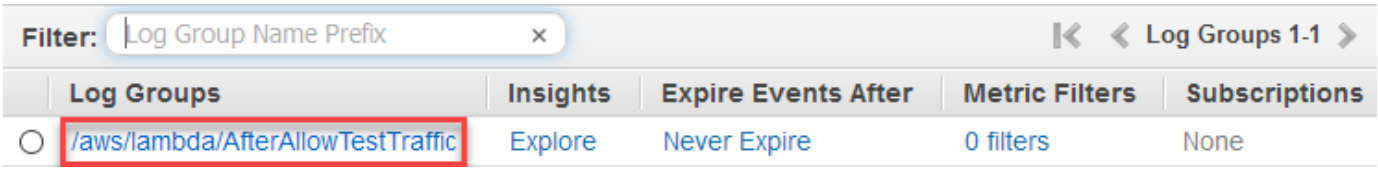
You can monitor your deployment in **Deployment status**. After 100% of production traffic is routed to the replacement task set, you can choose **Terminate original task set** to immediately terminate the original task set. If you do not choose **Terminate original task set**, the original task set terminates after the duration you specified when you created your deployment group.



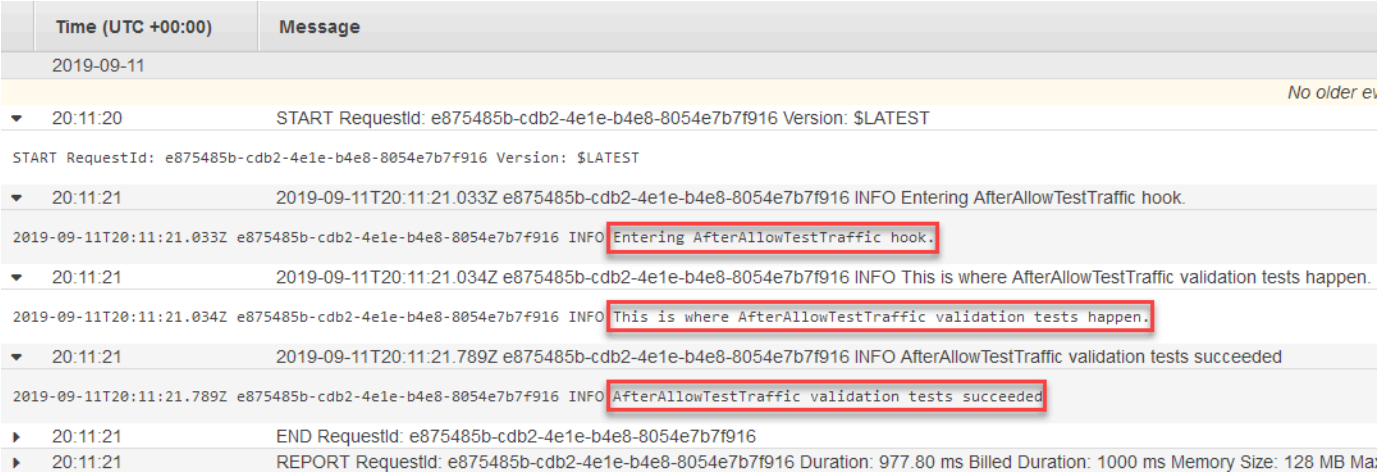
Step 6: View your Lambda hook function output in CloudWatch Logs

If your CodeDeploy deployment is successful, the validation tests in your Lambda hook fuctions are successful, too. You can confirm this by looking at the log for the hook function in CloudWatch Logs.

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. From the navigation pane, choose **Logs**. You should see one new log group for the Lambda hook function you specified in your AppSpec file.



3. Choose the new log group. This should be **/aws/lambda/AfterAllowTestTrafficHook**.
4. Choose the log stream. If you see more than one log stream, choose the one with the most recent date and time under **Last Event Time**.
5. Expand the log stream events to confirm your Lambda hook function wrote success messages to the log. The following shows the `AfterAllowTraffic` Lambda hook function was successful.



Step 7: Clean up

When you have finished this tutorial, clean up the resources associated with it to avoid incurring charges for resources that you aren't using. The resource names in this step are the names suggested in this tutorial (for example, **`ecs-demo-codedeploy-app`** for the name of your CodeDeploy application). If you used different names, then be sure to use those in your cleanup.

To clean up the tutorial resources

1. Use the delete-deployment-group command to delete the CodeDeploy deployment group.

```
aws deploy delete-deployment-group --application-name ecs-demo-deployment-group --
deployment-group-name ecs-demo-dg --region aws-region-id
```

2. Use the delete-application command to delete the CodeDeploy application.

```
aws deploy delete-application --application-name ecs-demo-deployment-group --
region aws-region-id
```

3. Use the delete-function command to delete your Lambda hook function.

```
aws lambda delete-function --function-name AfterAllowTestTraffic
```

4. Use the delete-log-group command to delete your CloudWatch log group.

```
aws logs delete-log-group --log-group-name /aws/lambda/AfterAllowTestTraffic
```

Tutorial: Deploy an updated Lambda function with CodeDeploy and the AWS Serverless Application Model

AWS SAM is an open-source framework for building serverless applications. It transforms and expands YAML syntax in a AWS SAM template into CloudFormation syntax to build serverless applications, such as a Lambda function. For more information, see What is the AWS Serverless Application Model?

In this tutorial, you use AWS SAM to create a solution that does the following:

- Creates your Lambda function.
- Creates your CodeDeploy application and deployment group.
- Creates two Lambda functions that execute deployment validation tests during CodeDeploy lifecycle hooks.
- Detects when your Lambda function is updated. The updating of the Lambda function triggers a deployment by CodeDeploy that incrementally shifts production traffic from the original version of your Lambda function to the updated version.

> **Note**
>
> This tutorial requires you to create resources that might result in charges to your AWS account. These include possible charges for CodeDeploy, Amazon CloudWatch, and AWS Lambda. For more information, see CodeDeploy pricing, Amazon CloudWatch pricing, and AWS Lambda pricing.

Topics

- Prerequisites

- Step 1: Set up your infrastructure
- Step 2: Update the Lambda function
- Step 3: Deploy the updated Lambda function
- Step 4: View your deployment results
- Step 5: Clean up

Prerequisites

To complete this tutorial, you must first:

- Complete the steps in Getting started with CodeDeploy.
- Install the AWS Serverless Application Model CLI. For information, see Install the AWS SAM CLI.
- Create an S3 bucket. AWS SAM uploads the artifacts that are referenced in your AWS SAM template into this bucket.

Step 1: Set up your infrastructure

This topic shows you how to use AWS SAM to create files for your AWS SAM template and your Lambda functions. Then, you use the AWS SAM `package` and `deploy` commands to generate the components in your infrastructure. When your infrastructure is ready, you have a CodeDeploy application and deployment group, the Lambda function to update and deploy, and two Lambda functions that contain validation tests that run when you deploy the Lambda function. When complete, you can use CloudFormation to view your components in the Lambda console or the AWS CLI to test your Lambda function.

Topics

- Create your files
- Package the AWS SAM application
- Deploy the AWS SAM application
- (Optional) inspect and test your infrastructure

Create your files

To create your infrastructure, you must create the following files:

- `template.yml`
- `myDateTimeFunction.js`
- `beforeAllowTraffic.js`
- `afterAllowTraffic.js`

Topics

- Create your AWS SAM template
- Create a file for your Lambda function
- Create a file for your BeforeAllowTraffic Lambda function
- Create a file for your AfterAllowTraffic Lambda function

Create your AWS SAM template

Create an AWS SAM template file that specifies the components in your infrastructure.

To create your AWS SAM template

1. Create a directory named `SAM-Tutorial`.
2. In your `SAM-Tutorial` directory, create a file named `template.yml`.
3. Copy the following YAML code into `template.yml`. This is your AWS SAM template.

```
AWSTemplateFormatVersion : '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: A sample SAM template for deploying Lambda functions.

Resources:
# Details about the myDateTimeFunction Lambda function
  myDateTimeFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: myDateTimeFunction.handler
      Runtime: nodejs18.x
# Instructs your myDateTimeFunction is published to an alias named "live".
      AutoPublishAlias: live
# Grants this function permission to call lambda:InvokeFunction
      Policies:
        - Version: "2012-10-17"
          Statement:
```

```
          - Effect: "Allow"
            Action:
              - "lambda:InvokeFunction"
            Resource: '*'
      DeploymentPreference:
# Specifies the deployment configuration
          Type: Linear10PercentEvery1Minute
# Specifies Lambda functions for deployment lifecycle hooks
          Hooks:
            PreTraffic: !Ref beforeAllowTraffic
            PostTraffic: !Ref afterAllowTraffic

# Specifies the BeforeAllowTraffic lifecycle hook Lambda function
  beforeAllowTraffic:
    Type: AWS::Serverless::Function
    Properties:
      Handler: beforeAllowTraffic.handler
      Policies:
        - Version: "2012-10-17"
# Grants this function permission to call
 codedeploy:PutLifecycleEventHookExecutionStatus
          Statement:
          - Effect: "Allow"
            Action:
              - "codedeploy:PutLifecycleEventHookExecutionStatus"
            Resource:
              !Sub 'arn:aws:codedeploy:${AWS::Region}:
${AWS::AccountId}:deploymentgroup:${ServerlessDeploymentApplication}/*'
        - Version: "2012-10-17"
# Grants this function permission to call lambda:InvokeFunction
          Statement:
          - Effect: "Allow"
            Action:
              - "lambda:InvokeFunction"
            Resource: !Ref myDateTimeFunction.Version
      Runtime: nodejs18.x
# Specifies the name of the Lambda hook function
      FunctionName: 'CodeDeployHook_beforeAllowTraffic'
      DeploymentPreference:
        Enabled: false
      Timeout: 5
      Environment:
        Variables:
          NewVersion: !Ref myDateTimeFunction.Version
```

```
# Specifies the AfterAllowTraffic lifecycle hook Lambda function
  afterAllowTraffic:
    Type: AWS::Serverless::Function
    Properties:
      Handler: afterAllowTraffic.handler
      Policies:
        - Version: "2012-10-17"
          Statement:
# Grants this function permission to call
 codedeploy:PutLifecycleEventHookExecutionStatus
          - Effect: "Allow"
            Action:
              - "codedeploy:PutLifecycleEventHookExecutionStatus"
            Resource:
              !Sub 'arn:aws:codedeploy:${AWS::Region}:
${AWS::AccountId}:deploymentgroup:${ServerlessDeploymentApplication}/*'
        - Version: "2012-10-17"
          Statement:
# Grants this function permission to call lambda:InvokeFunction
          - Effect: "Allow"
            Action:
              - "lambda:InvokeFunction"
            Resource: !Ref myDateTimeFunction.Version
      Runtime: nodejs18.x
# Specifies the name of the Lambda hook function
      FunctionName: 'CodeDeployHook_afterAllowTraffic'
      DeploymentPreference:
        Enabled: false
      Timeout: 5
      Environment:
        Variables:
          NewVersion: !Ref myDateTimeFunction.Version
```

This template specifies the following. For more information, see AWS SAM template concepts.

A Lambda function called `myDateTimeFunction`

When this Lambda function is published, the `AutoPublishAlias` line in the template links it to an alias named `live`. Later in this tutorial, an update to this function triggers a deployment by AWS CodeDeploy that incrementally shifts production traffic from the original version to the updated version.

Two Lambda deployment validation functions

The following Lambda functions are executed during CodeDeploy lifecycle hooks. The functions contain code that validate the deployment of the updated `myDateTimeFunction`. The result of the validation tests are passed to CodeDeploy using its `PutLifecycleEventHookExecutionStatus` API method. If a validation test fails, the deployment fails and is rolled back.

- `CodeDeployHook_beforeAllowTraffic` runs during the `BeforeAllowTraffic` hook.
- `CodeDeployHook_afterAllowTraffic` runs during the `AfterAllowTraffic` hook.

The name of both functions start with `CodeDeployHook_`. The `CodeDeployRoleForLambda` role allows calls to the Lambda `invoke` method only in Lambda functions with names that start with this prefix. For more information, see AppSpec 'hooks' section for an AWS Lambda deployment and PutLifecycleEventHookExecutionStatus in the *CodeDeploy API Reference*.

Automatic detection of an updated Lambda function

The `AutoPublishAlias` term tells the framework to detect when the `myDateTimeFunction` function changes, and then deploy it using the `live` alias.

A deployment configuration

The deployment configuration determines the rate at which your CodeDeploy application shifts traffic from the original version of the Lambda function to the new version. This template specifies the predefined deployment configuration `Linear10PercentEvery1Minute`.

> **Note**
>
> You cannot specify a custom deployment configuration in an AWS SAM template. For more information, see Create a Deployment Configuration.

Deployment lifecycle hook functions

The `Hooks` section specifies the functions to run during lifecycle event hooks. `PreTraffic` specifies the function that runs during the `BeforeAllowTraffic` hook. `PostTraffic` specifies the function that runs during the `AfterAllowTraffic` hook.

Permissions for Lambda to invoke another Lambda function

The specified `lambda:InvokeFunction` permission grants the role used by the AWS SAM application permission to invoke a Lambda function. This is required when the

`CodeDeployHook_beforeAllowTraffic` and `CodeDeployHook_afterAllowTraffic` functions invoke the deployed Lambda function during validation tests.

Create a file for your Lambda function

Create the file for the function you update and deploy later in this tutorial.

> **Note**
>
> A Lambda function can use any runtime supported by AWS Lambda. For more information, see AWS Lambda runtimes.

To create your Lambda function

1. Create a text file and save it as `myDateTimeFunction.js` in the `SAM-Tutorial` directory.
2. Copy the following Node.js code into `myDateTimeFunction.js`.

```
'use strict';

    exports.handler = function(event, context, callback) {

      if (event.body) {
        event = JSON.parse(event.body);
      }

      var sc; // Status code
      var result = ""; // Response payload

      switch(event.option) {
        case "date":
          switch(event.period) {
            case "yesterday":
              result = setDateResult("yesterday");
              sc = 200;
              break;
            case "today":
              result = setDateResult();
              sc = 200;
              break;
```

```
            case "tomorrow":
              result = setDateResult("tomorrow");
              sc = 200;
              break;
            default:
              result = {
                "error": "Must specify 'yesterday', 'today', or 'tomorrow'."
              };
              sc = 400;
              break;
          }
          break;

    /*      Later in this tutorial, you update this function by uncommenting
            this section. The framework created by AWS SAM detects the update
            and triggers a deployment by CodeDeploy. The deployment shifts
            production traffic to the updated version of this function.

            case "time":
            var d = new Date();
            var h = d.getHours();
            var mi = d.getMinutes();
            var s = d.getSeconds();

            result = {
              "hour": h,
              "minute": mi,
              "second": s
            };
            sc = 200;
            break;
    */
          default:
            result = {
              "error": "Must specify 'date' or 'time'."
            };
            sc = 400;
          break;
      }

      const response = {
        statusCode: sc,
        headers: { "Content-type": "application/json" },
        body: JSON.stringify( result )
```

```
    };

    callback(null, response);

    function setDateResult(option) {

      var d = new Date(); // Today
      var mo; // Month
      var da; // Day
      var y; // Year

      switch(option) {
        case "yesterday":
          d.setDate(d.getDate() - 1);
          break;
        case "tomorrow":
          d.setDate(d.getDate() + 1);
        default:
         break;
      }

      mo = d.getMonth() + 1; // Months are zero offset (0-11)
      da = d.getDate();
      y = d.getFullYear();

      result = {
        "month": mo,
        "day": da,
        "year": y
      };

      return result;
    }
  };
```

The Lambda function returns the day, month, and year for yesterday, today, or tomorrow. Later in this tutorial, you uncomment code that updates the function to return information about the day or time you specify (for example, the day, month, and year, or the current hour, minute, and second). The framework created by AWS SAM detects and deploys the updated version of the function.

> **Note**
>
> This Lambda function is also used in an AWS Cloud9 tutorial. AWS Cloud9 is a cloud-based integrated development environment. For information about how to create, execute, update, and debug this function in AWS Cloud9, see AWS Lambda tutorial for AWS Cloud9.

Create a file for your BeforeAllowTraffic Lambda function

Create the file for your `beforeAllowTraffic` hook Lambda function.

1. Create a text file and save it as `beforeAllowTraffic.js` in the `SAM-Tutorial` directory.
2. Copy the following Node.js code into `beforeAllowTraffic.js`. This function executes during your deployment's `BeforeAllowTraffic` hook.

```
'use strict';

    const AWS = require('aws-sdk');
    const codedeploy = new AWS.CodeDeploy({apiVersion: '2014-10-06'});
    var lambda = new AWS.Lambda();

    exports.handler = (event, context, callback) => {

     console.log("Entering PreTraffic Hook!");

     // Read the DeploymentId and LifecycleEventHookExecutionId from the event
 payload
      var deploymentId = event.DeploymentId;
     var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;

     var functionToTest = process.env.NewVersion;
     console.log("BeforeAllowTraffic hook tests started");
     console.log("Testing new function version: " + functionToTest);

     // Create parameters to pass to the updated Lambda function that
     // include the newly added "time" option. If the function did not
     // update, then the "time" option is invalid and function returns
     // a statusCode of 400 indicating it failed.
     var lambdaParams = {
      FunctionName: functionToTest,
      Payload: "{\"option\": \"time\"}",
      InvocationType: "RequestResponse"
```

```
    };

    var lambdaResult = "Failed";
    // Invoke the updated Lambda function.
    lambda.invoke(lambdaParams, function(err, data) {
     if (err){ // an error occurred
      console.log(err, err.stack);
      lambdaResult = "Failed";
     }
     else{ // successful response
      var result = JSON.parse(data.Payload);
      console.log("Result: " +  JSON.stringify(result));
         console.log("statusCode: " + result.statusCode);

         // Check if the status code returned by the updated
         // function is 400. If it is, then it failed. If
         // is not, then it succeeded.
      if (result.statusCode != "400"){
           console.log("Validation succeeded");
       lambdaResult = "Succeeded";
         }
         else {
           console.log("Validation failed");
         }

      // Complete the PreTraffic Hook by sending CodeDeploy the validation status
      var params = {
       deploymentId: deploymentId,
       lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
       status: lambdaResult // status can be 'Succeeded' or 'Failed'
      };

      // Pass CodeDeploy the prepared validation test results.
      codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data)
{
       if (err) {
        // Validation failed.
        console.log("CodeDeploy Status update failed");
        console.log(err, err.stack);
        callback("CodeDeploy Status update failed");
       } else {
        // Validation succeeded.
        console.log("CodeDeploy status updated successfully");
        callback(null, "CodeDeploy status updated successfully");
```

```
        }
       });
      }
     });
    }
```

Create a file for your AfterAllowTraffic Lambda function

Create the file for your `afterAllowTraffic` hook Lambda function.

1. Create a text file and save it as `afterAllowTraffic.js` in the `SAM-Tutorial` directory.
2. Copy the following Node.js code into `afterAllowTraffic.js`. This function executes during your deployment's `AfterAllowTraffic` hook.

```
'use strict';

    const AWS = require('aws-sdk');
    const codedeploy = new AWS.CodeDeploy({apiVersion: '2014-10-06'});
    var lambda = new AWS.Lambda();

    exports.handler = (event, context, callback) => {

     console.log("Entering PostTraffic Hook!");

     // Read the DeploymentId and LifecycleEventHookExecutionId from the event
 payload
      var deploymentId = event.DeploymentId;
     var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;

     var functionToTest = process.env.NewVersion;
     console.log("AfterAllowTraffic hook tests started");
     console.log("Testing new function version: " + functionToTest);

     // Create parameters to pass to the updated Lambda function that
     // include the original "date" parameter. If the function did not
     // update as expected, then the "date" option might be invalid. If
     // the parameter is invalid, the function returns
     // a statusCode of 400 indicating it failed.
     var lambdaParams = {
      FunctionName: functionToTest,
      Payload: "{\"option\": \"date\", \"period\": \"today\"}",
      InvocationType: "RequestResponse"
```

```
    };

    var lambdaResult = "Failed";
    // Invoke the updated Lambda function.
    lambda.invoke(lambdaParams, function(err, data) {
     if (err){ // an error occurred
      console.log(err, err.stack);
      lambdaResult = "Failed";
     }
     else{ // successful response
      var result = JSON.parse(data.Payload);
      console.log("Result: " +  JSON.stringify(result));
         console.log("statusCode: " + result.statusCode);

         // Check if the status code returned by the updated
         // function is 400. If it is, then it failed. If
         // is not, then it succeeded.
      if (result.statusCode != "400"){
           console.log("Validation of time parameter succeeded");
       lambdaResult = "Succeeded";
         }
         else {
           console.log("Validation failed");
         }

      // Complete the PostTraffic Hook by sending CodeDeploy the validation status
      var params = {
       deploymentId: deploymentId,
       lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
       status: lambdaResult // status can be 'Succeeded' or 'Failed'
      };

      // Pass CodeDeploy the prepared validation test results.
      codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data)
{
       if (err) {
        // Validation failed.
        console.log("CodeDeploy Status update failed");
        console.log(err, err.stack);
        callback("CodeDeploy Status update failed");
       } else {
        // Validation succeeded.
        console.log("CodeDeploy status updated successfully");
        callback(null, "CodeDeploy status updated successfully");
```

```
        }
       });
      }
     });
    }
```

Package the AWS SAM application

You should now have four files in your `SAM-Tutorial` directory:

- `beforeAllowTraffic.js`
- `afterAllowTraffic.js`
- `myDateTimeFunction.js`
- `template.yml`

You are now ready to use the AWS SAM **sam package** command to create and package artifacts for your Lambda functions and CodeDeploy application. The artifacts are uploaded to an S3 bucket. The output of the command is a new file called `package.yml`. This file is used by the AWS SAM **sam deploy** command in the next step.

> **Note**
>
> For more information on the **sam package** command, see AWS SAM CLI command reference in the *AWS Serverless Application Model Developer Guide*.

In the `SAM-Tutorial` directory, run the following.

```
sam package \
  --template-file template.yml \
  --output-template-file package.yml \
  --s3-bucket amzn-s3-demo-bucket
```

For the `s3-bucket` parameter, specify the Amazon S3 bucket you created as a prerequisite for this tutorial. The `output-template-file` specifies the name of the new file that is used by the AWS SAM **sam deploy** command.

Deploy the AWS SAM application

Use the AWS SAM **sam deploy** command with the `package.yml` file to create your Lambda functions and CodeDeploy application and deployment group using CloudFormation.

> **Note**
>
> For more information on the **sam deploy** command, see AWS SAM CLI command reference in the *AWS Serverless Application Model Developer Guide*.

In the `SAM-Tutorial` directory, run the following command.

```
sam deploy \
  --template-file package.yml \
  --stack-name my-date-time-app \
  --capabilities CAPABILITY_IAM
```

The `--capabilities CAPABILITY_IAM` parameter is required to authorize CloudFormation to create IAM roles.

(Optional) inspect and test your infrastructure

This topic shows how to view your infrastructure components and test your Lambda function.

To see the result of your stack after you run `sam deploy`

1. Open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. In the navigation pane, choose **Stacks**. The `my-date-time-app` stack appears at the top.
3. Choose the **Events** tab to see which events are complete. You can view the events while the stack creation is in progress. When creation of the stack is complete, you can see all stack creation events.
4. With your stack selected, choose **Resources**. In the **Type** column, you can see your Lambda functions, `myDateTimeFunction`, `CodeDeployHook_beforeAllowTraffic`, and `CodeDeployHook_afterAllowTraffic`. The **Physical ID** column of each of your Lambda functions contains a link to view the functions in the Lambda console.

> **Note**
>
> The name of the `myDateTimeFunction` Lambda function is prepended with the name of the CloudFormation stack and has an identifier added to it, so it looks like `my-date-time-app-myDateTimeFunction-123456ABCDEF`.

5. Open the CodeDeploy console at https://console.aws.amazon.com/codedeploy/.
6. In the navigation pane, expand **Deploy**, and then choose **Applications**.
7. You should see a new CodeDeploy application created by CloudFormation with a name that starts with `my-date-time-app-ServerlessDeploymentApplication`. Choose this application.
8. You should see a deployment group with a name that starts with `my-date-time-app-myDateTimeFunctionDeploymentGroup`. Choose this deployment group.

   Under **Deployment configuration**, you should see **CodeDeployDefault.LambdaLinear10PercentEvery1Minute**.

(Optional) to test your function (console)

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. From the navigation pane, choose your `my-date-time-app-myDateTimeFunction` function. In the console, its name contains an identifier, so it looks like `my-date-time-app-myDateTimeFunction-123456ABCDEF`.
3. Choose **Test**.
4. In **Event name**, enter a name for your test event.
5. Enter the following for your test event, and then choose **Create**.

```
{
  "option": "date",
  "period": "today"
}
```

6. Choose **Test**. You should see only your test event in the list of test events.

   For **Execution result**, you should see **succeeded**.

7. Under **Execution result**, expand **Details** to see the results. You should see the current month, day, and year.

(Optional) to test your function (AWS CLI)

1. Locate the ARN of your Lambda function. It appears at the top of the Lambda console when you are viewing your function.
2. Run the following command. Replace *your-function-arn* with the function ARN.

```
aws lambda invoke \
--function your-function-arn \
--cli-binary-format raw-in-base64-out \
--payload "{\"option\": \"date\", \"period\": \"today\"}" out.txt
```

3. Open `out.txt` to confirm the result contains the current month, day, and year.

Step 2: Update the Lambda function

In this topic, you update your `myDateTimeFunction.js` file. In the next step, you use the file to deploy the updated function. This triggers CodeDeploy to deploy it by shifting production traffic from the current version of the Lambda function to the updated version.

To update your Lambda function

1. Open `myDateTimeFunction.js`.
2. Remove the two comment markers ("`/*`" and "`*/`") and the explanatory text at the start and end of the `case` named `time` in the `switch` block.

   The uncommented code lets you pass a new parameter, `time`, to the function. If you pass `time` to the updated function, it returns the current `hour`, `minute`, and `second`.
3. Save `myDateTimeFunction.js`. It should look like the following:

```
'use strict';

exports.handler = function(event, context, callback) {

  if (event.body) {
    event = JSON.parse(event.body);
  }
```

```
var sc; // Status code
var result = ""; // Response payload

switch(event.option) {
  case "date":
    switch(event.period) {
      case "yesterday":
        result = setDateResult("yesterday");
        sc = 200;
        break;
      case "today":
        result = setDateResult();
        sc = 200;
        break;
      case "tomorrow":
        result = setDateResult("tomorrow");
        sc = 200;
        break;
      default:
        result = {
          "error": "Must specify 'yesterday', 'today', or 'tomorrow'."
        };
        sc = 400;
        break;
    }
    break;
    case "time":
      var d = new Date();
      var h = d.getHours();
      var mi = d.getMinutes();
      var s = d.getSeconds();

      result = {
        "hour": h,
        "minute": mi,
        "second": s
      };
      sc = 200;
      break;

    default:
      result = {
        "error": "Must specify 'date' or 'time'."
```

```
        };
        sc = 400;
      break;
  }

  const response = {
    statusCode: sc,
    headers: { "Content-type": "application/json" },
    body: JSON.stringify( result )
  };

  callback(null, response);

  function setDateResult(option) {

    var d = new Date(); // Today
    var mo; // Month
    var da; // Day
    var y; // Year

    switch(option) {
      case "yesterday":
        d.setDate(d.getDate() - 1);
        break;
      case "tomorrow":
        d.setDate(d.getDate() + 1);
      default:
       break;
    }

    mo = d.getMonth() + 1; // Months are zero offset (0-11)
    da = d.getDate();
    y = d.getFullYear();

    result = {
      "month": mo,
      "day": da,
      "year": y
    };

    return result;
  }
};
```

Step 3: Deploy the updated Lambda function

In this step, you use your updated `myDateTimeFunction.js` to update and initiate the deployment of your Lambda function. You can monitor the deployment progress in the CodeDeploy or AWS Lambda console.

The `AutoPublishAlias: live` line in your AWS SAM template causes your infrastructure to detect updates to functions that use the `live` alias. An update to your function triggers a deployment by CodeDeploy that shifts production traffic from the original version of the function to the updated version.

The **sam package** and **sam deploy** commands are used to update and trigger the deployment of your Lambda function. You executed these commands in Package the AWS SAM application and Deploy the AWS SAM application.

To deploy your updated Lambda function

1. In the `SAM-Tutorial` directory, run the following command.

```
sam package \
  --template-file template.yml \
  --output-template-file package.yml  \
  --s3-bucket amzn-s3-demo-bucket
```

   This creates a new set of artifacts that reference your updated Lambda function in your S3 bucket.

2. In the `SAM-Tutorial` directory, run the following command.

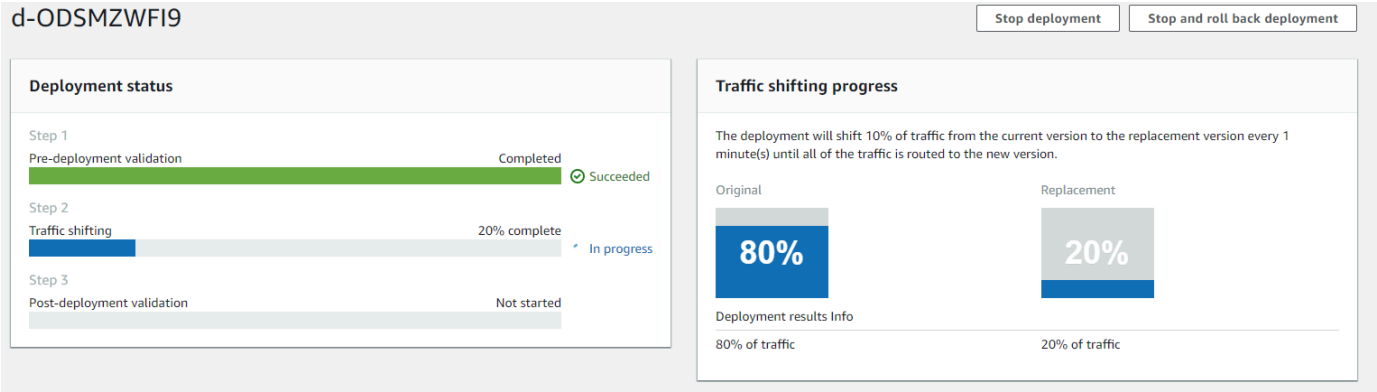
```
sam deploy \
  --template-file package.yml \
  --stack-name my-date-time-app \
  --capabilities CAPABILITY_IAM
```

   Because the stack name is still `my-date-time-app`, CloudFormation recognizes that this is a stack update. To view your updated stack, return the CloudFormation console, and from the navigation pane, choose **Stacks**.

(Optional) to view traffic during a deployment (CodeDeploy console)

1. Open the CodeDeploy console at https://console.aws.amazon.com/codedeploy/.
2. In the navigation pane, expand **Applications**, and then choose your **my-date-time-app-ServerlessDeploymentApplication** application.
3. In **Deployment groups**, choose your application's deployment group. Its status should be **In progress**.
4. In **Deployment group history**, choose the deployment that is in progress.

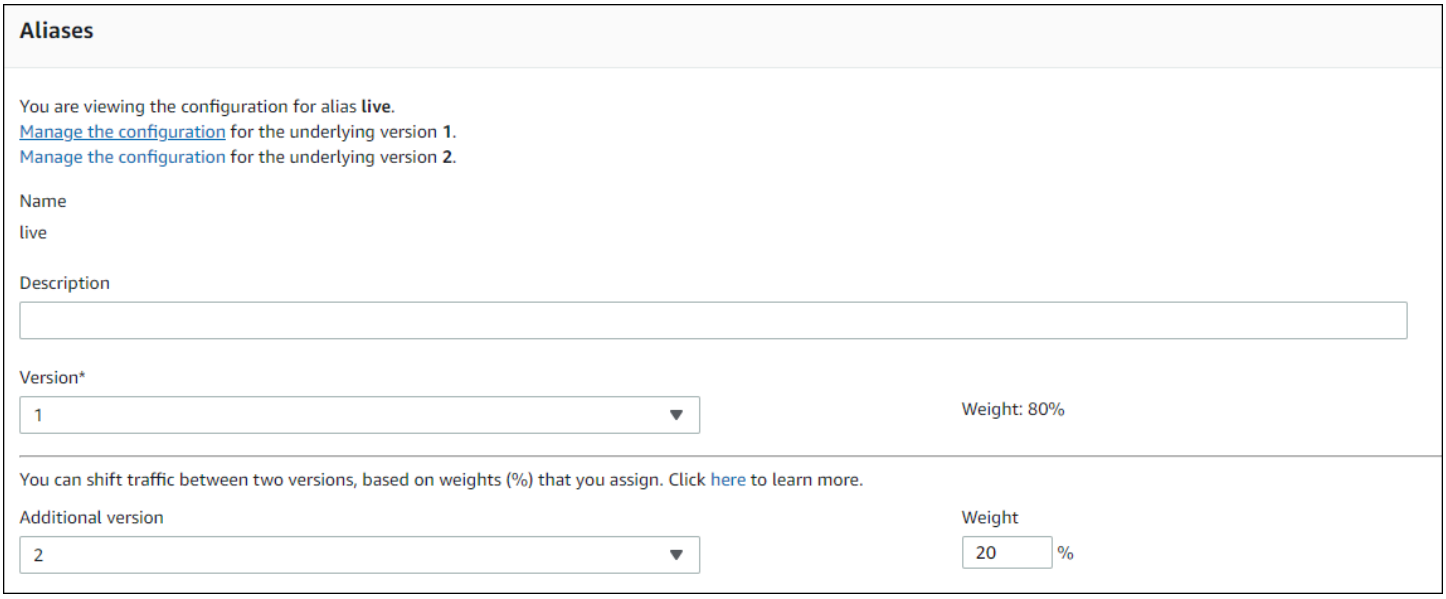
   The **Traffic shifting** progress bar and the percentages in the **Original** and **Replacement** boxes on this page display its progress.



(Optional) to view traffic during a deployment (Lambda console)

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. From the navigation pane, choose your `my-date-time-app-myDateTimeFunction` function. In the console, its name contains an identifier, so it looks like `my-date-time-app-myDateTimeFunction-123456ABCDEF`.
3. Choose **Aliases**, and then choose **live**.

The weights next to your original function version (version 1) and your updated function version (version 2) show how much traffic is served to each version at the time this AWS Lambda console page was loaded. The page does not update the weights over time. If you refresh the page once a minute, the weight for version 1 decreases by 10 percent and the weight for version 2 increases by 10 percent until the weight for version 2 is 100.



Step 4: View your deployment results

In this step, you view the results of your deployment. If your deployment succeeds, you can confirm that your updated Lambda function receives production traffic. If your deployment fails, you can use CloudWatch Logs to view the output of the validation tests in the Lambda function that run during your deployment's lifecycle hoooks.

Topics

- Test your deployed function
- View hook events in CloudWatch Logs

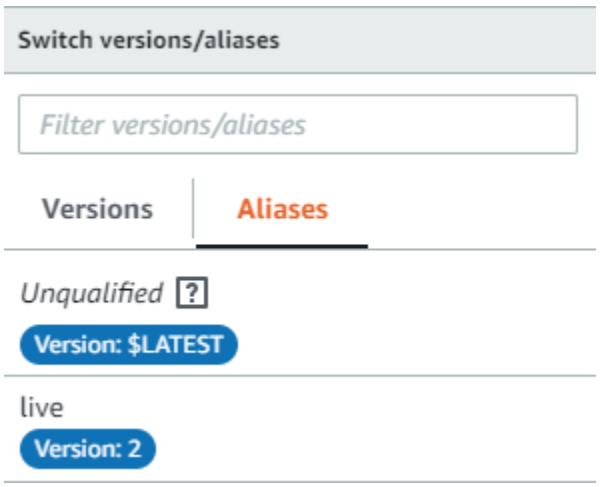
Test your deployed function

The **sam deploy** command updates the `my-date-time-app-myDateTimeFunction` Lambda function. The function version is updated to 2 and added to the `live` alias.

To see the update in the Lambda console

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. From the navigation pane, choose the `my-date-time-app-myDateTimeFunction` function. In the console, its name contains an identifier, so it looks like `my-date-time-app-myDateTimeFunction-123456ABCDEF`.

3. Choose **Qualifiers**, and then choose **Aliases**. After the deployment is complete (approximately 10 minutes), for the `live` alias, you should see **Version: 2**.



4. In **Function code**, view the source code for your function. Your changes should appear.
5. (Optional) You can use the test instructions in Step 2: Update the Lambda function to test your updated function. Create a new test event with the following payload, and then confirm the result contains the current hour, minute, and second.

```
{
    "option": "time"
  }
```

To use the AWS CLI to test your updated function, run the following command, and then open `out.txt` to confirm the result contains the current hour, minute, and second.

```
aws lambda invoke --function your-function-arn --payload "{\"option\": \"time\"}"
 out.txt
```

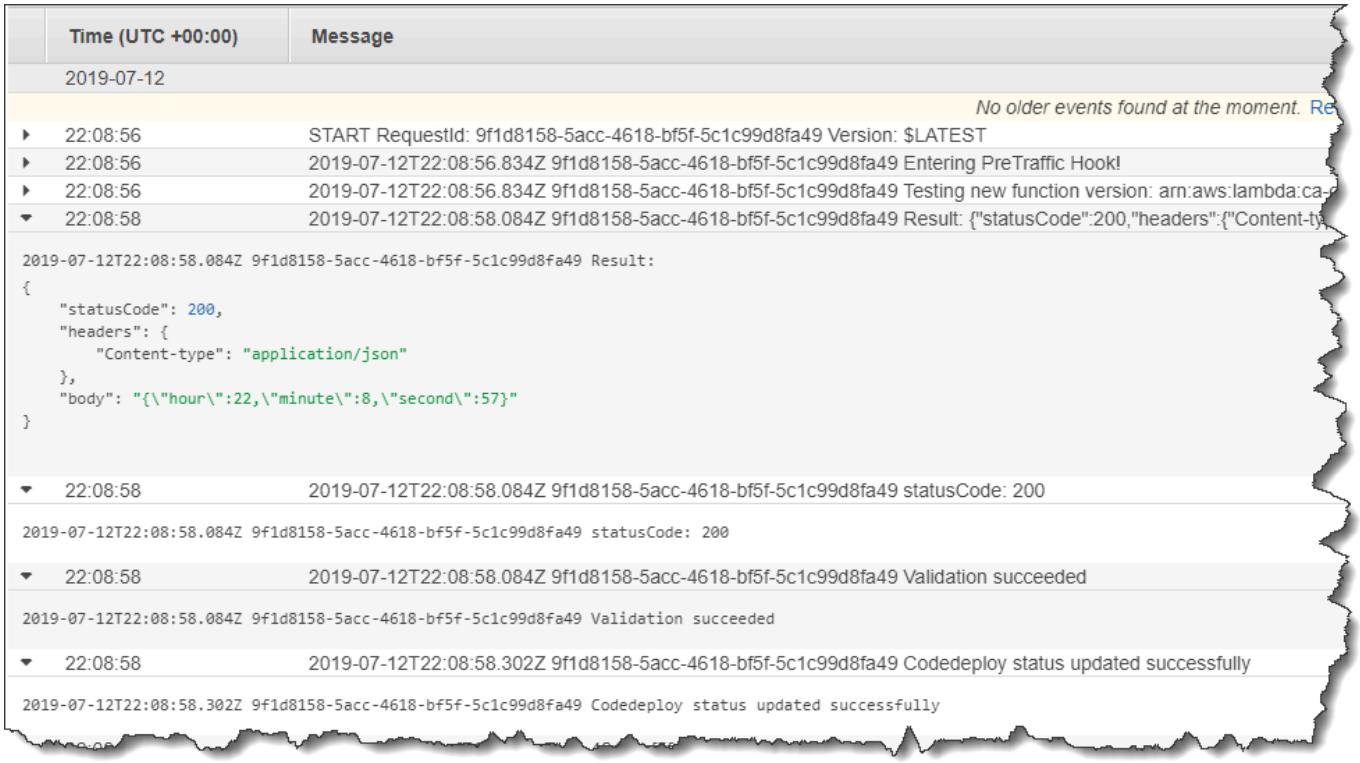
> **Note**
>
> If you use the AWS CLI to test your function before the deployment is complete, you might receive unexpected results. This is because CodeDeploy incrementally shifts 10 percent of traffic to the updated version every minute. During the deployment, some traffic still points to the original version, so `aws lambda invoke` might use the original version. After 10 minutes, the deployment is complete and all traffic points to the new version of the function.

View hook events in CloudWatch Logs

During the `BeforeAllowTraffic` hook, CodeDeploy executes your `CodeDeployHook_beforeAllowTraffic` Lambda function. During the `AfterAllowTraffic` hook, CodeDeploy executes your `CodeDeployHook_afterAllowTraffic` Lambda function. Each function runs a validation test that invokes the updated version of your function using the new `time` parameter. If the update of your Lambda function is successful, the `time` option does not cause an error and validation is successful. If the function was not updated, the unrecognized parameter causes an error and validation fails. These validation tests are for demonstration purposes only. You write your own tests to validate your deployment. You can use the CloudWatch Logs console to view your validation tests.

To view your CodeDeploy hook events

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. From the navigation pane, choose **Logs**.
3. From the list of log groups, choose **/aws/lambda/CodeDeployHook_beforeAllowTraffic** or **/aws/lambda/CodeDeployHook_afterAllowTraffic**.
4. Choose the log stream. You should see only one.
5. Expand the events to see their details.



Step 5: Clean up

To avoid further charges for resources you used during this tutorial, delete the resources created by your AWS SAM template and the CloudWatch logs created by your Lambda validation functions.

To delete your CloudFormation stack

1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. In the **Stacks** column, choose your `my-date-time-app` stack, and then choose **Delete**.
3. When prompted, choose **Delete stack**. The Lambda functions, CodeDeploy application and deployment group, and IAM roles created by AWS SAM are deleted.

To delete your logs in CloudWatch Logs

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. From the navigation pane, choose **Logs**.
3. From the list of log groups, choose the button next to **/aws/lambda/CodeDeployHook_beforeAllowTraffic**.
4. From **Actions**, choose **Delete log group**, and then choose **Yes, Delete**.
5. From the list of log groups, choose the button next to **/aws/lambda/CodeDeployHook_afterAllowTraffic**.
6. From **Actions**, choose **Delete log group**, and then choose **Yes, Delete**.

Working with the CodeDeploy agent

The AWS CodeDeploy agent is a software package that, when installed and configured on an instance, makes it possible for that instance to be used in CodeDeploy deployments.

AWS supports the latest minor version of the CodeDeploy agent. Currently the latest minor version is 1.8.x.

> **Note**
>
> The 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 Amazon ECS or 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 CodeDeploy agent configuration file, see CodeDeploy agent configuration reference.

For more information about working with the CodeDeploy agent, such as steps for installing, updating, and verifying versions, see Managing CodeDeploy agent operations.

Topics

- Operating systems supported by the CodeDeploy agent
- Communication protocol and port for the CodeDeploy agent
- Version history of the CodeDeploy agent
- Managing the CodeDeploy process
- Application revision and log file cleanup
- Files installed by the CodeDeploy agent
- Managing CodeDeploy agent operations

Operating systems supported by the CodeDeploy agent

Supported Amazon EC2 AMI operating systems

The CodeDeploy agent has been tested on the following Amazon EC2 AMI operating systems:

- Amazon Linux 2023 (ARM, x86)
- Amazon Linux 2 (ARM, x86)
- Microsoft Windows Server 2022, 2019
- Red Hat Enterprise Linux (RHEL) 9.x, 8.x, 7.x
- Ubuntu Server 22.04 LTS, 20.04 LTS, 18.04 LTS, 16.04 LTS

The 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 CodeDeploy agent repository in GitHub.

Supported on-premises operating systems

The CodeDeploy agent has been tested on the following on-premises operating systems:

- Microsoft Windows Server 2022, 2019
- Red Hat Enterprise Linux (RHEL) 9.x, 8.x, 7.x
- Ubuntu Server 22.04 LTS, 20.04 LTS

The 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 CodeDeploy agent repository in GitHub.

Communication protocol and port for the CodeDeploy agent

The CodeDeploy agent communicates outbound using HTTPS over port 443.

When the CodeDeploy agent runs on an EC2 instance, it will use the EC2 metadata endpoint to retrieve instance related information. Find out more about limiting and granting instance metadata service access.

Version history of the CodeDeploy agent

Your instances must be running a supported version of the CodeDeploy agent. The current minimum supported version is 1.8.x.

> **Note**
>
> We recommend using the latest version of the CodeDeploy agent. If you're having issues, update to the latest version before contacting AWS Support. For upgrade information, see Update the CodeDeploy agent.

The following table lists all releases of the CodeDeploy agent and the features and enhancements included with each version.

Version	Release date	Details
1.8.1	February 3, 2026	**Fixed**: S3 endpoint bugfix. **Added**: Sectigo CA certificate to Windows CA list.
1.8.0	July 31, 2025	**Changed**: Upgraded the bundled Ruby to 3.2 in the CodeDeploy agent for Windows.
1.7.1	November 14, 2024	**Changed**: Updated dependencies for security patches.
1.7.0	March 6, 2024	**Added**: A `:disable_imds_v1:` configuration setting to the CodeDeploy agent configuration file. Use this setting to disable the fallback to IMDSv1 when IMDSv2 errors occur. Defaults to `false` (enable the fallback). For more informati on, see CodeDeploy agent configuration reference. **Added**: Support for the Red Hat Enterprise Linux 9 (RHEL 9) operating system. **Added**: Support for Ruby versions 3.1 and 3.2 on Ubuntu Server.

Version	Release date	Details
		Fixed: The CodeDeploy agent now generates a user-friendly error if the CodeDeploy agent configuration file fails to load. **Changed**: Upgraded Ruby to 2.7.8-1 in the CodeDeploy agent for Windows.
1.6.0	March 30, 2023	**Added**: Support for Ruby 3.1, 3.2. **Added**: Support for Amazon Linux 2023. **Added**: Support for Windows Server 2022. **Changed**: The default setting of `verbose` is now `false` for Windows Server instances. To continue to print debug messages in log files on Windows, you must set `verbose` to `true`. **Removed**: Support for Windows Server 2016 and Windows Server 2012 R2. **Removed**: Support for Amazon Linux 2018.03.x.

Version	Release date	Details
1.5.0	March 3, 2023	**Added**: Support for Ruby 3. **Added**: Support for Ubuntu 22.04. **Fixed**: An issue where restarting the CodeDeploy agent soon after startup would lead to the agent hanging. **Changed**: The CodeDeploy agent now fails a host deployment on agent startup if the agent service restarts unexpectedly while running a hook script. This fix lets you avoid waiting for the 70-minute timeout period before retrying a deployment. **Deprecation notice**: CodeDeploy agent 1.5.0 is the last release to support Windows Server 2016 and Windows Server 2012 R2. **Removed:** Support for the CodeDeploy agent on Ubuntu 14.04 LTS, Windows Server 2008 R2, and Windows Server 2008 R2 32-bit.
1.4.1	December 6, 2022	**Fixed**: Security vulnerability related to logging. **Enhancement**: Improved logging when polling for the host command.

Version	Release date	Details
1.4.0	August 31, 2022	**Added**: Support for Red Hat Enterprise Linux 8. **Added**: Support for long file paths on the CodeDeploy agent for Windows. To enable long file paths, you'll need to set the appropriate Windows registry key and then restart your agent. For more information, see Long file paths cause "No such file or directory" errors. **Fixed**: An issue with the unzip operation when the disk was full. The CodeDeploy agent now detects the unzip's exit code 50 indicating a full disk, removes partially extracted files, and raises an exception to post a failure to the CodeDeploy server. The error message is visible as a lifecycle event error message, and the host-level deployment will stop without being stuck or timing-out. **Fixed**: An issue that would cause the agent to fail. **Fixed**: An issue where hooks would time out during an edge-case race condition. Hooks with no scripts will now continue and no longer cause failures or timeouts. **Changed**: The `update` script from the CodeDeploy agent's `bin` directory was removed because it is no longer used. **Changed**: The CodeDeploy agent for Windows Server now bundles Ruby 2.7. **Changed**: New environment variables were added, to be used by hook scripts depending on the source of the deployment bundle (Amazon S3 or GitHub). For more information, see Environment variable availability for hooks.

Version	Release date	Details
		Important **Deprecation notice**: CodeDeploy agent 1.4.0 is the last release that will include installers for 32-bit Windows Server. **Deprecation notice**: CodeDeploy agent 1.4.0 is the last release that will support Windows Server 2008 R2. **Removed**: Support for the CodeDeploy agent on the following Amazon EC2 AMIs: Amazon Linux 2014.09, 2016.03, 2016.09, and 2017.03.

Version	Release date	Details
1.3.2	May 6, 2021	**Important** CodeDeploy agent 1.3.2 addresses CVE-2018-1000201 which affects Windows hosts running the agent. The CVE cites ruby-ffi, which is a dependency of the CodeDeploy agent. If your agent was installed with Amazon EC2 Systems Manager (SSM) and is set to update automatically, no action is required. Otherwise, action is required to manually update the agent. To upgrade the agent follow the instructions in Update the CodeDeploy agent on Windows Server. **Fixed**: An issue when installing the CodeDeploy agent on Ubuntu 20.04 and later. **Fixed**: An intermittent issue that occurred when extractin g compressed files because relative paths weren't being handled correctly. **Added**: Support for AWS PrivateLink and VPC endpoints for Windows instances. **Added**: AppSpec file improvements, as described below. • You can now specify a custom filename for the AppSpec file when creating a local deployment. For more informati on, see Create a local deployment. • The AppSpec file can now have a `.yaml` file extension. • You can now overwrite deployed files using a new, optional `file_exists_behavior` setting in the AppSpec file. For more information, see AppSpec 'files' section (EC2/On-Premises deployments only).

Version	Release date	Details
		Upgraded: CodeDeploy now uses the AWS SDK for Ruby 3.0.
1.3.1	December 22, 2020	**Fixed**: 1.3.0 issue that prevented on-premises instances from starting.
1.3.0	November 10, 2020	**Important** This version is deprecated. **Fixed**: Removed an expired certificate that was no longer used. **Fixed**: Removed the prompt message from the agent uninstall script used by AWS Systems Manager, making it easier to downgrade a host or fleet to a previous version of the agent.
1.2.1	September 23, 2020	**Changed**: Upgraded AWS SDK for Ruby dependency from v2 to v3. **Added**: Support for IMDSv2. Includes a silent fallback to IMDSv1 if IMDSv2 http requests fail. **Changed**: Updated Rake and Rubyzip dependencies for security patches. **Fixed**: Ensure that an empty PID file will return a status of `No CodeDeploy Agent Running` and clean up the PID file on agent start.

Version	Release date	Details
1.1.2	August 4, 2020	**Added**: Support for Ubuntu Server 19.10 and 20.04. **Note**: : Version 19.10 reached its end-of-life date and is no longer supported by Ubuntu or CodeDeploy. **Added**: Memory efficiency improvements for Linux and Ubuntu to release reserved memory more timely. **Added**: Compatibility with Windows Server "silent-cleanup" which was causing the agent to be unresponsive in some cases. **Added**: Ignore non-empty directories during cleanup to avoid failures on deployment. **Added**: Support for AWS Local Zone in Los Angeles (LA). **Added**: Extract AZ from instance metadata to provide compatibility for AWS Local Zones. **Added**: Users can now provide their archive in subdirectories and aren't required to store it in the root directory. **Added**: Detected an issue with Rubyzip that could result in memory leaks. Updated the unzip command to first attempt to use a system-installed unzip utility before using Rubyzip. **Added**: `:enable_auth_policy:` as an agent configura tion setting. **Changed**: Unzip warnings are now ignored so deployments will continue.

Version	Release date	Details
1.1.0	June 30, 2020	**Changed**: Versioning of the CodeDeploy agent now follows the Ruby standard versioning convention. **Added**: New parameter to the install and update command to allow installation of specific agent version from the command line. **Removed**: Removed the CodeDeploy agent Auto Updater for Linux and Ubuntu. To configure automatic updates of the CodeDeploy agent, see Install the CodeDeploy agent using AWS Systems Manager.
1.0.1.1597	November 15, 2018	**Enhancement**: CodeDeploy supports Ubuntu 18.04. **Enhancement**: CodeDeploy supports Ruby 2.5. **Enhancement**: CodeDeploy supports FIPS endpoints. For more information about FIPS endpoints, see FIPS 140-2 overview. For endpoints that can be used with CodeBuild, see CodeDeploy regions and endpoints.
1.0.1.1518	June 12, 2018	**Enhancement**: Fixed an issue that caused an error when the CodeDeploy agent is closed while it is accepting poll requests. **Enhancement**: Added a deployment tracking feature that prevents the CodeDeploy agent from being closed when a deployment is in progress. **Enhancement**: Improved performance when deleting files.

Version	Release date	Details
1.0.1.1458	March 6, 2018	**Note**: This version is no longer supported. If you use this version, your deployments might fail. **Enhancement**: Improved certificate validations to support more trusted authorities. **Enhancement**: Fixed an issue that caused the local CLI to fail during a deployment that includes a BeforeInstall lifecycle event. **Enhancement**: Fixed an issue that might cause an active deployment to fail when the CodeDeploy agent is updated.
1.0.1.1352	November 16, 2017	**Note**: This version is no longer supported. If you use this version, your deployments might fail. **Feature**: Introduced a new feature for testing and debugging an EC2/On-Premises deployment on a local machine or instance where the CodeDeploy agent is installed.
1.0.1.1106	May 16, 2017	**Note**: This version is no longer supported. If you use this version, your deployments might fail. **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. Deploymen ts options for existing content now include retaining the content, overwriting the content, or failing the deployment. **Enhancement**: Made the CodeDeploy agent compatible with version 2.9.2 of the AWS SDK for Ruby (aws-sdk-core 2.9.2).

Version	Release date	Details
1.0.1.1095	March 29, 2017	**Note**: This version is no longer supported. If you use this version, your deployments might fail. **Enhancement**: Introduced support for the CodeDeploy agent in the China (Beijing) Region. **Enhancement**: Enabled Puppet to run on Windows Server instances when invoked by a lifecycle event hook. **Enhancement**: Improved the handling of `untar` operations.
1.0.1.1067	January 6, 2017	**Note**: This version is no longer supported. If you use this version, your deployments might fail. **Enhancement**: Revised many error messages to include more specific causes for deployment failures. **Enhancement**: Fixed an issue that prevented the CodeDeplo y agent from identifying the correct application revision to deploy during some deployments. **Enhancement**: Reverted the usage of `pushd` and popd before and after the `untar` operation.
1.0.1.1045	November 21, 2016	**Note**: This version is no longer supported. If you use this version, your deployments might fail. **Enhancement**: Made the CodeDeploy agent compatible with version 2.6.11 of the AWS SDK for Ruby (aws-sdk-core 2.6.11).

Version	Release date	Details
1.0.1.1037	October 19, 2016	**Note**: This version is no longer supported. If you use this version, your deployments might fail. The 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. **Enhancement**: The agent can now determine which version of Ruby is installed on an instance so it can invoke the `codedeploy-agent` script using that version.
1.0.1.1011.1	August 17, 2016	**Note**: This version is no longer supported. If you use this version, your deployments might fail. **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	**Note**: This version is no longer supported. If you use this version, your deployments might fail. The 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. **Feature**: Added support for invoking the CodeDeploy agent using the bash shell on operating systems where the systemd init system is in use. **Enhancement**: Enabled support for all versions of Ruby 2.x in the CodeDeploy agent and the CodeDeploy agent updater. Updated CodeDeploy agents are no longer dependent on Ruby 2.0 only. (Ruby 2.0 is still required for deb and rpm versions of the CodeDeploy agent installer.)

Version	Release date	Details
1.0.1.998	July 11, 2016	**Note**: This version is no longer supported. If you use this version, your deployments might fail. **Enhancement**: Fixed support for running the CodeDeploy agent with user profiles other than *root*. The variable named `USER` is replaced by `CODEDEPLOY_USER` to avoid conflicts with environmental variables.
1.0.1.966	June 16, 2016	**Note**: This version is no longer supported. If you use this version, your deployments might fail. **Feature**: Introduced support for running the CodeDeploy agent with user profiles other than *root*. **Enhancement**: Fixed support for specifying the number of application revisions you want the CodeDeploy agent to archive for a deployment group. **Enhancement**: Made the CodeDeploy agent compatible with version 2.3 of the AWS SDK for Ruby (aws-sdk-core 2.3). **Enhancement**: Fixed issues with UTF-8 encoding during deployments. **Enhancement**: Improved accuracy when identifying process names.
1.0.1.950	March 24, 2016	**Note**: This version is no longer supported. If you use this version, your deployments might fail. **Feature**: Added installation proxy support. **Enhancement**: Updated the installation script to not download the CodeDeploy agent if the latest version is already installed.

Version	Release date	Details
1.0.1.934	February 11, 2016	**Note**: This version is no longer supported. If you use this version, your deployments might fail. **Feature**: Introduced support for specifying the number of application revisions you want the CodeDeploy agent to archive for a deployment group.
1.0.1.880	January 11, 2016	**Note**: This version is no longer supported and might cause deployments to fail. **Enhancement**: Made the 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.
1.0.1.854	November 17, 2015	**Note**: This version is no longer supported. If you use this version, your deployments might fail. **Feature**: Introduced support for the SHA-256 hash algorithm . **Feature**: Introduced version tracking support in `.version` files. **Feature**: Made the deployment group ID available through the use of an environment variable. **Enhancement**: Added support for monitoring CodeDeploy agent logs using Amazon CloudWatch Logs.

For related information, see the following:

- Determine the version of the CodeDeploy agent
- Install the CodeDeploy agent

For a history of CodeDeploy agent versions, see the Release repository on GitHub.

Managing the CodeDeploy process

All Linux distributions of the CodeDeploy agent (rpm and deb) use systemd by default to manage the agent process.

However, both the rpm and deb distributions ship with startup scripts that reside at `/etc/init.d/codedeploy-agent`. Depending on which distribution you are using, when using a command such as `sudo service codedeploy-agent restart`, the scripts at `/etc/init.d` may be run to launch the agent process instead of allowing `systemd` to manage the process. Running scripts at `/etc/init.d` is undesirable.

To prevent this issue, for systems that support `systemd` we recommend using the `systemctl` utility for any agent operations instead of using the `service` command.

For example, to restart the CodeDeploy agent use `sudo systemctl restart codedeploy-agent` instead of the equivalent command with the `service` utility.

Application revision and log file cleanup

The CodeDeploy agent archives revisions and log files on instances. The 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. 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 is always retained, even if the number of failed deployments exceeds the number of retained revisions. If no value is specified, CodeDeploy retains the five most recent revisions in addition to the currently deployed revision.

CodeDeploy logs: For Amazon Linux, Ubuntu Server, and RHEL instances, the 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 CodeDeploy agent

The 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 CodeDeploy agent configuration file to configure the directory's name and location. For more information, see CodeDeploy agent configuration reference.

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 CodeDeploy deployment details.

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 CodeDeploy agent configuration file to change the default. For more information, see CodeDeploy agent configuration reference.

> **Note**
>
> If you want to recover hard disk space used by archived deployments, update the **max_revisions** setting to a low number, such as 1 or 2. 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 verison 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 CodeDeploy agent has copied all files in the deployment application to the instance. It is used by the 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 CodeDeploy agent continues to write to it while deleting old content.

Managing CodeDeploy agent operations

The instructions in this section show you how to install, uninstall, reinstall, or update the CodeDeploy agent and how to verify the CodeDeploy agent is running.

Topics

- Verify the CodeDeploy agent is running
- Determine the version of the CodeDeploy agent
- Install the CodeDeploy agent
- Update the CodeDeploy agent
- Uninstall the CodeDeploy agent
- Send CodeDeploy agent logs to CloudWatch

Verify the CodeDeploy agent is running

This section describes commands to run if you suspect the CodeDeploy agent has stopped running on an instance.

Topics

- Verify the CodeDeploy agent for Amazon Linux or RHEL is running
- Verify the CodeDeploy agent for Ubuntu Server is running
- Verify the CodeDeploy agent for Windows Server is running

Verify the CodeDeploy agent for Amazon Linux or RHEL is running

To see if the CodeDeploy agent is installed and running, sign in to the instance, and run the following command:

```
systemctl status codedeploy-agent
```

If the command returns an error, the CodeDeploy agent is not installed. Install it as described in Install the CodeDeploy agent for Amazon Linux or RHEL.

If the 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:

```
systemctl start codedeploy-agent
```

```
systemctl status codedeploy-agent
```

Verify the CodeDeploy agent for Ubuntu Server is running

To see if the CodeDeploy agent is installed and running, sign in to the instance, and run the following command:

```
systemctl status codedeploy-agent
```

If the command returns an error, the CodeDeploy agent is not installed. Install it as described in Install the CodeDeploy agent for Ubuntu Server.

If the 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:

```
systemctl start codedeploy-agent
```

```
systemctl status codedeploy-agent
```

Verify the CodeDeploy agent for Windows Server is running

To see if the 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 CodeDeploy agent is not installed. Install it as described in Install the CodeDeploy agent for Windows Server.

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 CodeDeploy agent

You can determine the version of the CodeDeploy agent running on your instance in two ways.

First, starting with version 1.0.1.854 of the 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 CodeDeploy agent.

Topics

- Determine the version on Amazon Linux or RHEL
- Determine the version on Ubuntu Server
- Determine the version on Windows Server

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 the CodeDeploy agent

To use CodeDeploy on EC2 instances or on-premises servers, the CodeDeploy agent must be installed first. We recommend installing and updating the CodeDeploy agent with AWS Systems Manager. For more information about Systems Manager, see What is AWS Systems Manager. You can set up installation and scheduled updates of the CodeDeploy agent with Systems Manager in the console when you create your deployment groups.

You can also install the CodeDeploy agent directly from an S3 bucket with the command line.

For the recommended version to install, see Version history of the CodeDeploy agent.

Topics

- Install the CodeDeploy agent using AWS Systems Manager
- Install the CodeDeploy agent using the command line

Install the CodeDeploy agent using AWS Systems Manager

You can use the AWS Management Console or the AWS CLI to install the CodeDeploy agent to your Amazon EC2 or on-premises instances by using AWS Systems Manager. You can choose to install a specific version or choose to always install the latest version of the agent. For more information about AWS Systems Manager, see What is AWS Systems Manager.

Using AWS Systems Manager is the recommended method for installing and updating the CodeDeploy agent. You can also install the CodeDeploy agent from an Amazon S3 bucket. For information about using an Amazon S3 download link, see Install the CodeDeploy agent using the command line.

Topics

- Prerequisites
- Install the CodeDeploy agent

Prerequisites

Follow the steps in Getting started with CodeDeploy to set up IAM permissions and the AWS CLI.

If installing the CodeDeploy agent on an on-premises server with Systems Manager, you must register your on-premises server with Amazon EC2 Systems Manager. For more information, see Setting up Systems Manager in hybrid environments in the *AWS Systems Manager User Guide*.

Install the CodeDeploy agent

Before you can use Systems Manager to install the CodeDeploy agent, you must make sure that the instance is configured correctly for Systems Manager.

Installing or updating the SSM agent

On an Amazon EC2 instance, the CodeDeploy agent requires that the instance is running version 2.3.274.0 or later. Before you install the CodeDeploy agent, update or install SSM agent on the instance if you haven't already done so.

The SSM agent comes preinstalled on some Amazon EC2 AMIs provided by AWS. For more information, see Amazon Machine Images (AMIs) with SSM agent preinstalled.

> **Note**
>
> Make sure that the instance's operating system is also supported by the CodeDeploy agent. For more information, see Operating systems supported by the CodeDeploy agent.

For information about installing or updating SSM agent on an instance running Linux, see Installing and configuring the SSM agent on Linux instances in the *AWS Systems Manager User Guide*.

For information about installing or updating SSM agent on an instance running Windows Server, see Installing and configuring SSM agent on Windows instances in the *AWS Systems Manager User Guide*.

(Optional) Verify Systems Manager prerequisites

Before you use Systems Manager Run Command to install the CodeDeploy agent, verify that your instances meet the minimum Systems Manager requirements. For more information, see Setting up AWS Systems Manager in the *AWS Systems Manager User Guide*.

Install the CodeDeploy agent

With SSM, you can install the CodeDeploy once or set up a schedule to install new versions.

To install the CodeDeploy agent, choose the `AWSCodeDeployAgent` package while you follow the steps in Install or update packages with AWS Systems Manager distributor.

Install the CodeDeploy agent using the command line

> **Note**
>
> We recommend installing the CodeDeploy agent with AWS Systems Manager to be able to configure scheduled updates of the agent. For more information, see Install the CodeDeploy agent using AWS Systems Manager.

Use the following topics to install and run the CodeDeploy agent using the command line.

Topics

- Install the CodeDeploy agent for Amazon Linux or RHEL
- Install the CodeDeploy agent for Ubuntu Server
- Install the CodeDeploy agent for Windows Server

Install the CodeDeploy agent for Amazon Linux or RHEL

Sign in to the instance, and run the following commands, one at a time. Running the command `sudo yum update` first is considered best practice when using `yum` to install packages, but you can skip it if you do not wish to update all of your packages.

```
sudo yum update
```

```
sudo yum install ruby
```

```
sudo yum install wget
```

(Optional) To clean the AMI of any previous agent caching information, run the following script:

```
#!/bin/bash
CODEDEPLOY_BIN="/opt/codedeploy-agent/bin/codedeploy-agent"
$CODEDEPLOY_BIN stop
yum erase codedeploy-agent -y
```

Change to your home directory:

```
cd /home/ec2-user
```

> **Note**
>
> In the previous 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.

Download the CodeDeploy agent installer:

```
wget https://bucket-name.s3.region-identifier.amazonaws.com/latest/install
```

`bucket-name` is the name of the Amazon S3 bucket that contains the CodeDeploy Resource Kit files for your region, and *`region-identifier`* is the identifier for your region.

For example:

`https://aws-codedeploy-us-east-2.s3.us-east-2.amazonaws.com/latest/install`

For a list of bucket names and region identifiers, see Resource kit bucket names by Region.

Set execute permissions on the `install` file:

```
chmod +x ./install
```

To install the latest version of the CodeDeploy agent:

- ```
  sudo ./install auto
  ```

To install a specific version of the CodeDeploy agent:

- List the available versions in your region:

  ```
  aws s3 ls s3://aws-codedeploy-region-identifier/releases/ --region region-identifier
   | grep '\.rpm$'
  ```
  ```

- Install one of the versions:

```
sudo ./install auto -v releases/codedeploy-agent-version.noarch.rpm
```

> **Note**
>
> AWS supports the latest minor version of the CodeDeploy agent. Currently the latest minor version is 1.7.x.

To check that the service is running, run the following command:

```
systemctl status codedeploy-agent
```

If the 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:

```
systemctl start codedeploy-agent
```

```
systemctl status codedeploy-agent
```

Install the CodeDeploy agent for Ubuntu Server

> **Note**
>
> We recommend installing the CodeDeploy agent with AWS Systems Manager to be able to configure scheduled updates of the agent. For more information, see Install the CodeDeploy agent using AWS Systems Manager.

To install the CodeDeploy agent on Ubuntu Server

1. Sign in to the instance.
2. Enter the following commands, one after the other:

```
sudo apt update
```

```
sudo apt install ruby-full
```

```
sudo apt install wget
```

3. Enter the following command:

```
cd /home/ubuntu
```

/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.

4. Enter the following command:

```
wget https://bucket-name.s3.region-identifier.amazonaws.com/latest/install
```

bucket-name is the name of the Amazon S3 bucket that contains the CodeDeploy Resource Kit files for your region, and *region-identifier* is the identifier for your region.

For example:

`https://aws-codedeploy-us-east-2.s3.us-east-2.amazonaws.com/latest/install`

For a list of bucket names and region identifiers, see Resource kit bucket names by Region.

5. Enter the following command:

```
chmod +x ./install
```

6. Do one of the following:

- To install the latest version of the CodeDeploy agent on any supported version of Ubuntu Server *except* 20.04:

```
sudo ./install auto
```

- To install the latest version of the CodeDeploy agent on Ubuntu Server 20.04:

> **Note**
>
> Writing the output to a temporary log file is a workaround that should be used while we address a known bug with the `install` script on Ubuntu Server 20.04.

```
sudo ./install auto > /tmp/logfile
```

- To install a specific version of the CodeDeploy agent on any supported version of Ubuntu Server *except* 20.04:
  - List the available versions in your region:

```
aws s3 ls s3://aws-codedeploy-region-identifier/releases/ --region region-
identifier | grep '\.deb$'
```

  - Install one of the versions:

```
sudo ./install auto -v releases/codedeploy-agent-###.deb
```

> **Note**
>
> AWS supports the latest minor version of the CodeDeploy agent. Currently the latest minor version is 1.7.x.

- To install a specific version of the CodeDeploy agent on Ubuntu Server 20.04:
  - List the available versions in your region:

```
aws s3 ls s3://aws-codedeploy-region-identifier/releases/ --region region-
identifier | grep '\.deb$'
```

  - Install one of the versions:

```
sudo ./install auto -v releases/codedeploy-agent-###.deb > /tmp/logfile
```

> **Note**
>
> Writing the output to a temporary log file is a workaround that should be used while we address a known bug with the `install` script on Ubuntu Server 20.04.

> **Note**
>
> AWS supports the latest minor version of the CodeDeploy agent. Currently the latest minor version is 1.7.x.

To check that the service is running

1. Enter the following command:

   ```
   systemctl status codedeploy-agent
   ```

   If the CodeDeploy agent is installed and running, you should see a message like `The AWS CodeDeploy agent is running`.

2. 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:

   ```
   systemctl start codedeploy-agent
   ```

   ```
   systemctl status codedeploy-agent
   ```

Install the CodeDeploy agent for Windows Server

On Windows Server instances, you can use one of these methods to download and install the CodeDeploy agent:

- Use AWS Systems Manager (recommended)
- Run a series of Windows PowerShell commands.
- Choose a direct download link.

- Run an Amazon S3 copy command.

> **Note**
>
> The folder that the CodeDeploy agent is installed to is `C:\Program Data\Amazon\CodeDeploy`. Make sure there are no directory junctions or symlinks on this path.

Topics

- Use Systems Manager
- Use Windows PowerShell
- Use a direct link
- Use an Amazon S3 copy command

Use Systems Manager

Follow the instructions in Install the CodeDeploy agent using AWS Systems Manager to install the CodeDeploy agent.

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 CodeDeploy agent installation file is downloaded.

   ```
   New-Item -Path "c:\temp" -ItemType "directory" -Force
   ```

4. Configure AWS credentials using the `Set-AWSCredential` and `Initialize-AWSDefaultConfiguration` commands. For more information, see Using AWS credentials in the *AWS tools for PowerShell User Guide*.
5. Download the CodeDeploy agent installation file.

> **Note**
>
> AWS supports the latest minor version of the CodeDeploy agent. Currently the latest minor version is 1.7.x.

To install the latest version of the CodeDeploy agent:

- ```
  powershell.exe -Command Read-S3Object -BucketName bucket-name -Key latest/codedeploy-agent.msi -File c:\temp\codedeploy-agent.msi
  ```

To install a specific version of the CodeDeploy agent:

- ```
  powershell.exe -Command Read-S3Object -BucketName bucket-name -Key releases/codedeploy-agent-###.msi -File c:\temp\codedeploy-agent.msi
  ```

`bucket-name` is the name of the Amazon S3 bucket that contains the 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.

6. Run the CodeDeploy agent installation file.

```
c:\temp\codedeploy-agent.msi /quiet /l c:\temp\host-agent-install-log.txt
```

To check that the service is running, run the following command:

```
powershell.exe -Command Get-Service -Name codedeployagent
```

If the 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 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 `https://s3.*.amazonaws.com`), you can use a direct link for your Region to download the CodeDeploy agent and then run the installer manually.

The link is:

```
https://s3.region.amazonaws.com/aws-codedeploy-region/latest/codedeploy-agent.msi
```

...where *`region`* is the AWS Region where you're deploying your application.

For example:

```
https://s3.af-south-1.amazonaws.com/aws-codedeploy-af-south-1/latest/codedeploy-
agent.msi
```

> **Important**
>
> Obtain the `.msi` file from the same Region as your CodeDeploy application. Choosing a different Region may cause `inconsistent region` failures in the `codedeploy-agent-log` file when you run the `.msi` file.

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 CodeDeploy agent and then run the installer manually. For information, see Install the AWS Command Line Interface on Microsoft Windows.

The Amazon S3 command is:

```
aws s3 cp s3://aws-codedeploy-region/latest/codedeploy-agent.msi codedeploy-agent.msi
 --region region
```

...where *region* is the AWS Region where you're deploying your application.

For example:

```
aws s3 cp s3://aws-codedeploy-af-south-1/latest/codedeploy-agent.msi codedeploy-
agent.msi --region af-south-1
```

Update the CodeDeploy agent

You can configure automatic, scheduled updates of the CodeDeploy agent on all supported operating systems using AWS Systems Manager. You can also force updates on all supported operating systems by running a command on an instance.

Topics

- Update the CodeDeploy agent on Amazon Linux or RHEL
- Update the CodeDeploy agent on Ubuntu Server
- Update the CodeDeploy agent on Windows Server

Update the CodeDeploy agent on Amazon Linux or RHEL

To configure automatic, scheduled updates of the CodeDeploy agent using AWS Systems Manager, follow the steps in Install the CodeDeploy agent with AWS Systems Manager.

If you want to force an update of the CodeDeploy agent, sign in to the instance, and run the following command:

```
sudo /opt/codedeploy-agent/bin/install auto
```

Update the CodeDeploy agent on Ubuntu Server

To configure automatic, scheduled updates of the CodeDeploy agent using AWS Systems Manager, follow the steps in Install the CodeDeploy agent with AWS Systems Manager.

If you want to force an update of the CodeDeploy agent, sign in to the instance, and run the following command:

```
sudo /opt/codedeploy-agent/bin/install auto
```

Update the CodeDeploy agent on Windows Server

You can enable automatic updates of the CodeDeploy agent with AWS Systems Manager. With Systems Manager, you can configure an update schedule for your Amazon EC2 or on-premises instances by creating an association with Systems Manager State Manager. You can also manually update the CodeDeploy agent by uninstalling the current version and installing a newer one.

Topics

- Set up automatic CodeDeploy agent update with AWS Systems Manager
- Update the CodeDeploy agent manually
- (Deprecated) Update the CodeDeploy agent with the Windows Server Updater

Set up automatic CodeDeploy agent update with AWS Systems Manager

To configure Systems Manager and enable automatic updates of the CodeDeploy agent, follow the instructions in Install the CodeDeploy agent using AWS Systems Manager.

Update the CodeDeploy agent manually

To update the CodeDeploy agent manually, you can install the latest version from the CLI or using Systems Manager. Follow the instructions in Install the CodeDeploy agent. It is recommended that you uninstall older versions of the CodeDeploy agent by following the instructions in Uninstall the CodeDeploy agent.

(Deprecated) Update the CodeDeploy agent with the Windows Server Updater

> **Note**
>
> The CodeDeploy agent updater for Windows Server is deprecated and will not update to any version after 1.0.1.1597.

To enable automatic updates of the CodeDeploy agent, install the 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 CodeDeploy agent, follow the instructions in Install the CodeDeploy agent for Windows Server.

On Windows Server instances, you can download and install the CodeDeploy agent updater by running Windows PowerShell commands, using a direct download link, or running an Amazon S3 copy command.

Topics

- Use Windows PowerShell
- Use a direct link
- Use an Amazon S3 copy command

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 bucket that contains the 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.

If you need to troubleshoot an update process error, type the following command to open the 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 required permissions (for example, to `http://s3.*.amazonaws.com`), you can use a direct link to download the CodeDeploy agent updater.

The link is:

```
https://s3.region.amazonaws.com/aws-codedeploy-region/latest/codedeploy-agent-
updater.msi
```

...where *`region`* is the AWS Region where you're updating your application.

For example:

```
https://s3.af-south-1.amazonaws.com/aws-codedeploy-af-south-1/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 CodeDeploy agent updater and then run the installer manually. For information, see Install the AWS Command Line Interface on Microsoft Windows.

The Amazon S3 command is:

```
aws s3 cp s3://aws-codedeploy-region/latest/codedeploy-agent-updater.msi codedeploy-
agent-updater.msi --region region
```

...where *region* is the AWS Region where you're updating your application.

For example:

```
aws s3 cp s3://aws-codedeploy-af-south-1/latest/codedeploy-agent-updater.msi
 codedeploy-agent-updater.msi --region af-south-1
```

Uninstall the CodeDeploy agent

You can remove the CodeDeploy agent from instances when it is no longer needed or when you want to perform a fresh installation.

Uninstall the CodeDeploy agent from Amazon Linux or RHEL

To uninstall the CodeDeploy agent, sign in to the instance and run the following command:

```
sudo yum erase codedeploy-agent
```

Uninstall the CodeDeploy agent from Ubuntu Server

To uninstall the CodeDeploy agent, sign in to the instance and run the following command:

```
sudo dpkg --purge codedeploy-agent
```

Uninstall the CodeDeploy agent from Windows Server

To uninstall the 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
```

You can also sign in to the instance, and in **Control Panel**, open **Programs and Features**, choose **CodeDeploy Host Agent**, and then choose **Uninstall**.

Send CodeDeploy agent logs to CloudWatch

You can send CodeDeploy agent metric and log data to CloudWatch using the unified CloudWatch agent, or more simply, the CloudWatch agent.

Use the following instructions to install the CloudWatch agent and configure it for use with CodeDeploy agents.

Prerequisites

Before you begin, complete the following tasks:

- Install the CodeDeploy agent and make sure it's running. For more information, see Install the CodeDeploy agent and Verify the CodeDeploy agent is running.
- Install the CloudWatch agent. For more information, see Installing the CloudWatch agent.
- Add the following permissions to the CodeDeploy IAM instance profile:
  - CloudWatchLogsFullAccess
  - CloudWatchAgentServerPolicy

  For more information on the CodeDeploy instance profile, see Step 4: Create an IAM instance profile for your Amazon EC2 instances of Getting started with CodeDeploy.

Configure the CloudWatch agent to collect CodeDeploy logs

You can configure the CloudWatch agent by stepping through a wizard or by manually creating or editing a configuration file.

To configure the CloudWatch agent using the wizard (Linux)

1. Run the wizard, as described in Run the CloudWatch agent configuration wizard.

2. In the wizard, when asked `Do you want to monitor any log files?` enter **1**.
3. Specify the CodeDeploy agent log file, as follows:
   1. For `Log file path` enter the path for the CodeDeploy log file, for example: **`/var/log/aws/codedeploy-agent/codedeploy-agent.log`**.
   2. For `Log group name` enter a log group name, for example: **`codedeploy-agent-log`**.
   3. For `Log stream name` enter a log stream name, for example: **`{instance_id}-codedeploy-agent-log`**.
4. When asked `Do you want to specify any additional log files?`, enter **1**.
5. Specify the CodeDeploy agent deployment logs, as follows:
   1. For `Log file path` enter the path for the CodeDeploy deployment log file, for example: **`/opt/codedeploy-agent/deployment-root/deployment-logs/codedeploy-agent-deployments.log`**.
   2. For `Log group name` enter a log group name, for example: **`codedeploy-agent-deployment-log`**.
   3. For `Log stream name` enter a log stream name, for example: **`{instance_id}-codedeploy-agent-deployment-log`**.
6. When asked `Do you want to specify any additional log files?`, enter **1**.
7. Specify the CodeDeploy agent updater logs, as follows:
   1. For `Log file path` enter the path for the CodeDeploy updater log file, for example: **`/tmp/codedeploy-agent.update.log`**.
   2. For `Log group name` enter a log group name, for example: **`codedeploy-agent-updater-log`**.
   3. For `Log stream name` enter a log stream name, for example: **`{instance_id}-codedeploy-agent-updater-log`**.

To configure the CloudWatch agent using the wizard (Windows)

1. Run the wizard, as described in Run the CloudWatch agent configuration wizard.
2. In the wizard, when asked `Do you want to monitor any customized log files?` enter **1**.
3. Specify the CodeDeploy log file, as follows:

   1. For `Log file path` enter the path r the CodeDeploy agent log file, for example: **`C:\ProgramData\Amazon\CodeDeploy\log\codedeploy-agent-log.txt`**.
   2. For `Log group name` enter a log group name, for example: **`codedeploy-agent-log`**.
   3. For `Log stream name` enter a log stream name, for example: **`{instance_id}-codedeploy-agent-log`**.
4. When asked `Do you want to specify any additional log files?`, enter **1**.
5. Specify the CodeDeploy agent deployment logs, as follows:
   1. For `Log file path` enter the path the CodeDeploy deployment log file, for example: **`C:\ProgramData\Amazon\CodeDeploy\deployment-logs\codedeploy-agent-deployments.log`**.
   2. For `Log group name` enter a log group name, for example: **`codedeploy-agent-deployment-log`**.
   3. For `Log stream name` enter a log stream name, for example: **`{instance_id}-codedeploy-agent-deployment-log`**.

To configure the CloudWatch agent by manually creating or editing a configuration file (Linux)

1. Create or edit the CloudWatch agent configuration file as described in Manually create or edit the CloudWatch agent configuration file.
2. Make sure that the file is called `/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json` and that it contains the following code:

```
...
"logs": {
    "logs_collected": {
        "files": {
            "collect_list": [
                {
                    "file_path": "/var/log/aws/codedeploy-agent/codedeploy-
agent.log",
                    "log_group_name": "codedeploy-agent-log",
                    "log_stream_name": "{instance_id}-agent-log"
                },
                {
                    "file_path": "/opt/codedeploy-agent/deployment-root/deployment-
logs/codedeploy-agent-deployments.log",
```

```
                    "log_group_name": "codedeploy-agent-deployment-log",
                    "log_stream_name": "{instance_id}-codedeploy-agent-deployment-
log"
                },
                {
                    "file_path": "/tmp/codedeploy-agent.update.log",
                    "log_group_name": "codedeploy-agent-updater-log",
                    "log_stream_name": "{instance_id}-codedeploy-agent-updater-log"
                }
            ]
        }
    }
}
...
```

To configure the CloudWatch agent by manually creating or editing a configuration file (Windows)

1. Create or edit the CloudWatch agent configuration file as described in Manually create or edit the CloudWatch agent configuration file.
2. Make sure that the file is called `C:\ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json` and that it contains the following code:

```
...
"logs": {
        "logs_collected": {
            "files": {
                "collect_list": [
                    {
                        "file_path": "C:\\ProgramData\\Amazon\\CodeDeploy\\log\
\codedeploy-agent-log.txt",
                        "log_group_name": "codedeploy-agent-log",
                        "log_stream_name": "{instance_id}-codedeploy-agent-log"
                    },
                    {
                        "file_path": "C:\\ProgramData\\Amazon\\CodeDeploy\
\deployment-logs\\codedeploy-agent-deployments.log",
                        "log_group_name": "codedeploy-agent-deployment-log",
                        "log_stream_name": "{instance_id}-codedeploy-agent-
deployment-log"
                    }
```

```
                ]
            },
            ...
        }
    },
...
```

Restart the CloudWatch agent

After making your changes, restart the CloudWatch agent as described in Start the CloudWatch agent.

Working with instances for CodeDeploy

CodeDeploy supports deployments to instances running Amazon Linux, Ubuntu Server, Red Hat Enterprise Linux (RHEL), and Windows Server.

You can use 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 that can run the CodeDeploy agent and connect to public AWS service endpoints. You can use CodeDeploy to simultaneously deploy an application to Amazon EC2 instances in the cloud and to desktop PCs 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 CodeDeploy agent that's compatible with the operating system running on the instance.	Yes	Yes
Requires the instance to be able to connect to CodeDeploy.	Yes	Yes
Requires an IAM instance profile to be attached to the instance. The IAM instance profile must have permissions to participate in CodeDeploy deployments. For information, see Step 4: Create an IAM instance profile for your Amazon EC2 instances.	Yes	No

Subject	Amazon EC2 instances	On-premises instances
Requires you to do one of the following to authenticate and register instances: • 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. • Create an IAM user for each instance and store the IAM user's account credentials in plain text on the instance.	No	Yes
Requires you to register each instance with CodeDeploy before you can deploy to it.	No	Yes
Requires you to tag each instance before CodeDeploy can deploy to it.	Yes	Yes
Can participate in Amazon EC2 Auto Scaling and Elastic Load Balancing scenarios as part of CodeDeploy deployments.	Yes	No
Can be deployed from Amazon S3 buckets and GitHub repositories.	Yes	Yes

Subject	Amazon EC2 instances	On-premises instances
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 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.	To launch the Amazon EC2 instance with the least amount of effort, see Create an Amazon EC2 instance for CodeDeploy (CloudFormation template). To launch the Amazon EC2 instance mostly on your own, see Create an Amazon EC2 instance for CodeDeploy (AWS CLI or Amazon EC2 console).
I want to launch a new Ubuntu Server or RHEL Amazon EC2 instance.	See Create an Amazon EC2 instance for CodeDeploy (AWS CLI or Amazon EC2 console).
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 CodeDeploy.
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.

I want CodeDeploy to provision a replacement fleet of instances during a blue/green deployment.	See Working with deployments in CodeDeploy.

To prepare Amazon EC2 instances in Amazon EC2 Auto Scaling groups, you must follow some additional steps. For more information, see Integrating CodeDeploy with Amazon EC2 Auto Scaling.

Topics

- Tagging Instances for Deployments
- Working with Amazon EC2 Instances
- Working with On-Premises Instances
- View Instance Details
- Instance Health

Tagging instances for deployment groups in 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 a 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 Amazon EC2 Auto Scaling groups in deployment groups, but they are identified by their names rather than by tags applied to instances. For information, see Integrating CodeDeploy with Amazon EC2 Auto Scaling.

The criteria for instances in a deployment group can be as simple as a single tag in a single tag group. It can be 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
- Example 2: Single tag group, multiple tags
- Example 3: Multiple tag groups, single tags
- Example 4: Multiple tag groups, multiple tags

Example 1: Single tag group, single tag

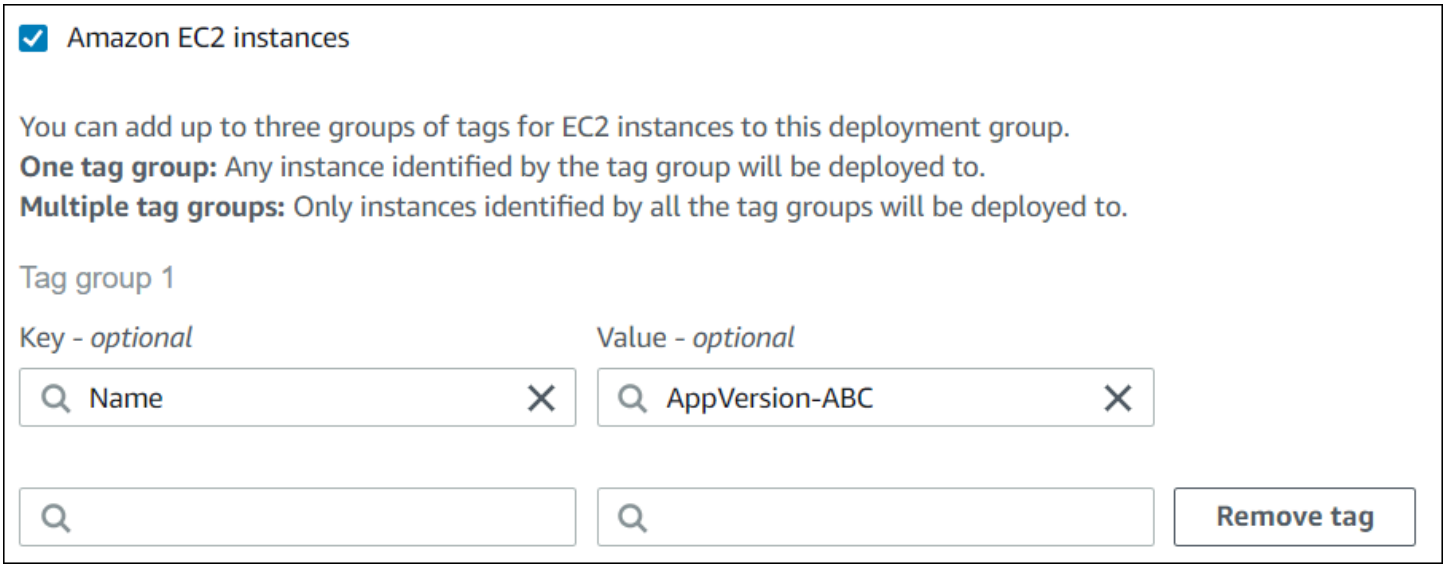
You can specify a single tag in a single tag group:

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.

CodeDeploy console setup view:



JSON structure:

```
"ec2TagSet": {
         "ec2TagSetList": [
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Name",
                  "Value": "AppVersion-ABC"
               }
            ]
         ]
      },
```

Example 2: Single tag group, multiple tags

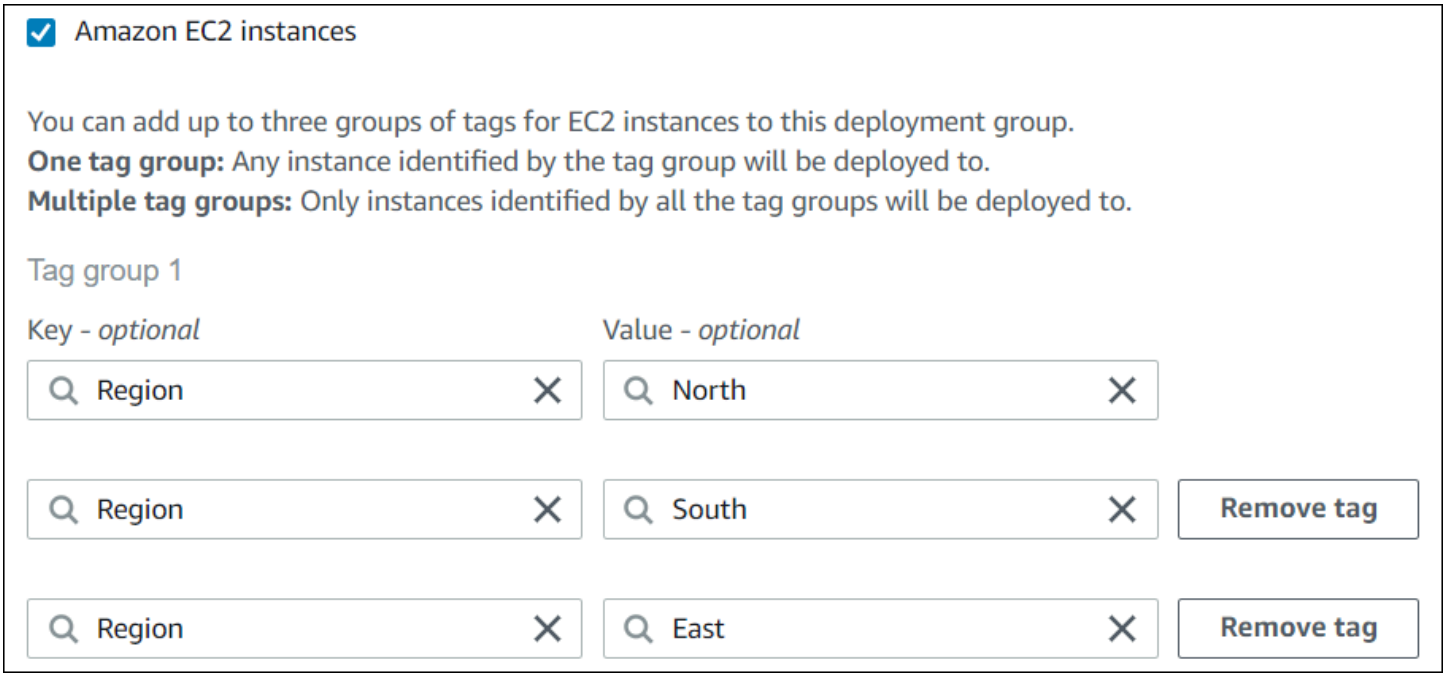
You can also specify multiple tags in a single tag group:

Tag group 1

Key	Value
Region	North
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.

CodeDeploy console setup view:



JSON structure:

```
"ec2TagSet": {
         "ec2TagSetList": [
            [
               {
                  "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

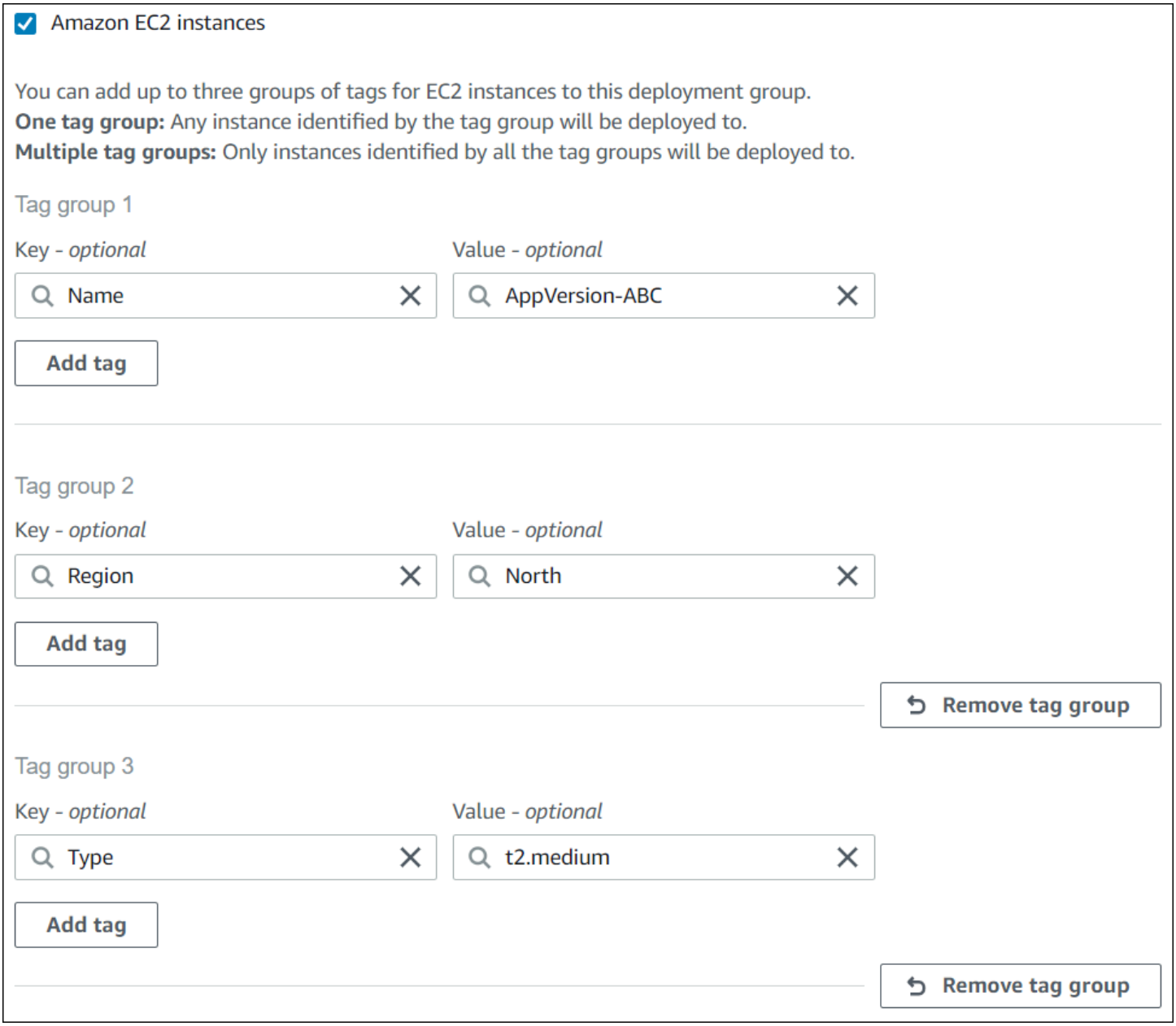
Key	Value
Region	North

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`. In this example, only the instances also tagged with `Region=North` and `Type=t2.medium` are part of the deployment group.

CodeDeploy console setup view:



JSON structure:

```
"ec2TagSet": {
         "ec2TagSetList": [
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Name",
                  "Value": "AppVersion-ABC"
               }
            ],
```

```
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "North"
               }
            ],
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Type",
                  "Value": "t2.medium"
               }
            ],
         ]
      },
```

Example 4: Multiple tag groups, multiple tags

When you use multiple tag groups with multiple tags in one or more group, an instance must match at least one of the tags in each of the groups.

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`

CodeDeploy console setup view:

Amazon EC2 instances

You can add up to three groups of tags for EC2 instances to this deployment group.
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 - *optional*	Value - *optional*	
Environment	Staging	
Environment	Beta	Remove tag

Add tag

Tag group 2

Key - *optional*	Value - *optional*	
Region	South	
Region	North	Remove tag
Region	East	Remove tag

Add tag

Remove tag group

Tag group 3

Key - *optional*	Value - *optional*	
Type	t2.medium	
Type	t2.large	Remove tag

Add tag

Remove tag group

JSON structure:

```
"ec2TagSet": {
         "ec2TagSetList": [
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Environment",
                  "Value": "Beta"
               },
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Environment",
                  "Value": "Staging"
               }
            ],
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "North"
               },
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "South"
               },
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "East"
               }
            ],
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Type",
                  "Value": "t2.medium"
               },
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Type",
                  "Value": "t2.large"
               }
```

```
            ],
        ]
    },
```

Working with Amazon EC2 instances for CodeDeploy

An Amazon EC2 instance is a virtual computing environment that you create and configure using Amazon Elastic Compute Cloud. Amazon EC2 provides scalable computing capacity in the AWS Cloud. You can use Amazon EC2 to launch as many or as few virtual servers as you need for your 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 CodeDeploy deployments.

Topics

- Create an Amazon EC2 instance for CodeDeploy (AWS CLI or Amazon EC2 console)
- Create an Amazon EC2 instance for CodeDeploy (CloudFormation template)
- Configure an Amazon EC2 instance to work with CodeDeploy

Create an Amazon EC2 instance for 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 CodeDeploy deployments.

You can use our CloudFormation template to launch an Amazon EC2 instance running Amazon Linux or Windows Server that is already configured for use in CodeDeploy deployments. We do not provide an 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 CodeDeploy.

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)

Prerequisites

If you have not done so already, follow the instructions in Getting started with CodeDeploy to set up and configure the AWS CLI and create an IAM instance profile.

Launch 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, 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**. You must choose an Amazon EC2 AMI operating systems supported by CodeDeploy. For more information, see Operating systems supported by the CodeDeploy agent.
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. If you used the suggested role name, then choose **`CodeDeployDemo-EC2-Instance-Profile`**. If you created your own role name, choose that.

   > **Note**
   >
   > If a default virtual private cloud (VPC) is not displayed in the **Network** list, you must choose or create an Amazon VPC and subnet. Choose **Create new VPC** or **Create new subnet** or both. For more information, see Your VPC and subnets.

6. Choose **Next: Add Storage**.
7. Leave the **Step 4: Add Storage** page unchanged, and choose **Next: Add Tags**.
8. On the **Step 5: Add Tags** page, choose **Add Tag**.
9. In the **Key** box, type **`Name`**. In the **Value** box type **`CodeDeployDemo`**.

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

10. Choose **Next: Configure Security Group**.
11. On the **Step 6: Configure Security Group** page, leave the **Create a new security group** option selected.

    A default SSH role is configured for Amazon EC2 instances running Amazon Linux, Ubuntu Server, or RHEL. A default RDP role is configured for Amazon EC2 instances running Windows Server.
12. 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 **Custom 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**. 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**.
13. Leave the **Step 7: Review Instance Launch** page unchanged, and choose **Launch**.
14. 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.
15. Choose **Launch Instances**.

16. Choose the ID for your Amazon EC2 instance. Do not continue until the instance has been launched and passed all checks.

Install the CodeDeploy agent

The CodeDeploy agent must be installed on your Amazon EC2 instance before using it in CodeDeploy deployments. For more information, see Install the CodeDeploy agent.

> **Note**
>
> You can configure automatic installation and updates of the CodeDeploy agent when you create your deployment group in the console.

Launch an Amazon EC2 instance (CLI)

Prerequisites

If you have not done so already, follow the instructions in Getting started with CodeDeploy to set up and configure the AWS CLI and create an IAM instance profile.

Launch an Amazon EC2 instance

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 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. 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. Call the **run-instances** command to create and launch the Amazon EC2 instance.

   Before you call this command, you need to collect the following:

   - The ID of an Amazon Machine Image (AMI) (*`ami-id`*) you 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 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 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.
   - 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 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. For example:

```
aws ec2 run-instances \
  --image-id ami-id \
  --key-name key-name \
  --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. 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, and specify the name of the security group you created in Step 1. For example:

```
aws ec2 run-instances --image-id ami-id --key-name key-name --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.

3. 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 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` appears in the output.

Install the CodeDeploy agent

The CodeDeploy agent must be installed on your Amazon EC2 instance before using it in CodeDeploy deployments. For more information, see Install the CodeDeploy agent.

> **Note**
>
> You can configure automatic installation and updates of the CodeDeploy agent when you create your deployment group in the console.

Create an Amazon EC2 instance for CodeDeploy (CloudFormation template)

You can use our CloudFormation template to quickly launch an Amazon EC2 instance running Amazon Linux or Windows Server. You can use the AWS CLI, the 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 CloudFormation to give the instance permission to participate in CodeDeploy deployments.
- Tags the instance so CodeDeploy can find it during a deployment.
- Installs and runs the CodeDeploy agent on the instance.

You don't have to use our CloudFormation to set up an Amazon EC2 instance. For alternatives, see Working with instances for CodeDeploy.

We do not provide an CloudFormation template for Amazon EC2 instances running Ubuntu Server or Red Hat Enterprise Linux (RHEL).

Topics

- Before you begin
- Launch an Amazon EC2 instance with the CloudFormation template (console)
- Launch an Amazon EC2 instance with the CloudFormation template (AWS CLI)

Before you begin

Before you can use the CloudFormation template to launch Amazon EC2 instances, make sure you complete the following steps.

1. Make sure you have created an administrative user, as described in Step 1: Setting up. Double-check that the user has the following minimum permissions and add any that are not present:

- cloudformation:*
- codedeploy:*
- ec2:*
- iam:AddRoleToInstanceProfile
- iam:CreateInstanceProfile
- iam:CreateRole
- iam:DeleteInstanceProfile
- iam:DeleteRole
- iam:DeleteRolePolicy
- iam:GetRole
- iam:DeleteRolePolicy
- iam:PutRolePolicy
- iam:RemoveRoleFromInstanceProfile

2. Make sure you 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.

   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 can't use the instance key pair with CodeDeploy.

Launch an Amazon EC2 instance with the CloudFormation template (console)

1. Sign in to the AWS Management Console and open the 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 CodeDeploy. On the navigation bar, in the region selector, choose one

of the regions listed in Region and endpoints in *AWS General Reference*. 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 CloudFormation template for your region, and then choose **Next**.

Region	Location of 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`
Europe (Ireland) Region	`http://s3-eu-west-1.amazonaws.com/aws-codedeploy-eu-we`

Region	Location of CloudFormation template
	st-1/templates/latest/CodeDeploy_SampleCF_Template.json
Europe (London) Region	http://s3-eu-west-2.amazonaws.com/aws-codedeploy-eu-west-2/templates/latest/CodeDeploy_SampleCF_Template.json
Europe (Paris) Region	http://s3-eu-west-3.amazonaws.com/aws-codedeploy-eu-west-3/templates/latest/CodeDeploy_SampleCF_Template.json
Europe (Frankfurt) Region	http://s3-eu-central-1.amazonaws.com/aws-codedeploy-eu-central-1/templates/latest/CodeDeploy_SampleCF_Template.json
Israel (Tel Aviv) Region	http://s3-il-central-1.amazonaws.com/aws-codedeploy-il-central-1/templates/latest/CodeDeploy_SampleCF_Template.json
Asia Pacific (Hong Kong) Region	http://s3-ap-east-1.amazonaws.com/aws-codedeploy-ap-east-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 CloudFormation template
Asia Pacific (Seoul) Region	`http://s3-ap-northeast-2.amazonaws.com/aws-codedeploy-ap-northeast-2/templates/latest/CodeDeploy_SampleCF_Template.json`
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 (Melbourne) Region	`https://aws-codedeploy-ap-southeast-4.s3.ap-southeast-4.amazonaws.com/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	`aws-codedeploy-ap-northeast-1.s3.sa-east-1.amazonaws.com/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 pair name. Type the key pair name only, not the key pair file extension.
   - 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. 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 CodeDeploy will use to identify the instances during deployment (or leave the default of **Name**).
   - For **TagValue**, type the instance tag value 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**
   >
   > CloudFormation tags are different from CodeDeploy tags. CloudFormation uses tags to simplify administration of your infrastructure. CodeDeploy uses tags to identify Amazon EC2 instances. You specified CodeDeploy tags on the **Specify Parameters** page.

7. On the **Review** page, in **Capabilities**, select the **I acknowledge that CloudFormation might create IAM resources** box, and then choose **Create**.

After CloudFormation has created the stack and launched the Amazon EC2 instances, in the CloudFormation console, **CREATE_COMPLETE** will be displayed in the **Status** column. This process can take several minutes.

To verify the CodeDeploy agent is running on the Amazon EC2 instances, see Managing CodeDeploy agent operations, and then proceed to Create an application with CodeDeploy.

Launch an Amazon EC2 instance with the CloudFormation template (AWS CLI)

1. Use our CloudFormation template in a call to the **create-stack** command. This stack will launch a new Amazon EC2 instance with the 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
```

   keyName is the instance key pair name. Type the key pair name only, not the key pair file extension.

`template-url` is the location of the CloudFormation template for your region:

Region	Location of 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`
Europe (Ireland) Region	`http://s3-eu-west-1.amazonaws.com/aws-codedeploy-eu-west-1/templates/latest/CodeDeploy_SampleCF_Template.json`

Region	Location of CloudFormation template
Europe (London) Region	`http://s3-eu-west-2.amazonaws.com/aws-codedeploy-eu-west-2/templates/latest/CodeDeploy_SampleCF_Template.json`
Europe (Paris) Region	`http://s3-eu-west-3.amazonaws.com/aws-codedeploy-eu-west-3/templates/latest/CodeDeploy_SampleCF_Template.json`
Europe (Frankfurt) Region	`http://s3-eu-central-1.amazonaws.com/aws-codedeploy-eu-central-1/templates/latest/CodeDeploy_SampleCF_Template.json`
Israel (Tel Aviv) Region	`http://s3-il-central-1.amazonaws.com/aws-codedeploy-il-central-1/templates/latest/CodeDeploy_SampleCF_Template.json`
Asia Pacific (Hong Kong) Region	`http://s3-ap-east-1.amazonaws.com/aws-codedeploy-ap-east-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 CloudFormation template
Asia Pacific (Seoul) Region	http://s3-ap-northeast-2.amazonaws.com/aws-codedeploy-ap-northeast-2/templates/latest/CodeDeploy_SampleCF_Template.json
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 (Melbourne) Region	https://aws-codedeploy-ap-southeast-4.s3.ap-southeast-4.amazonaws.com/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	aws-codedeploy-ap-northeast-1.s3.sa-east-1.amazonaws.com/templates/latest/CodeDeploy_SampleCF_Template.json

This command creates an CloudFormation stack named **`CodeDeployDemoStack`**, using the 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 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 CodeDeploy agent is running on the Amazon EC2 instance, see Managing CodeDeploy agent operations, and then proceed to Create an application with CodeDeploy.

Configure an Amazon EC2 instance to work with 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 CodeDeploy deployments.

> **Note**
>
> If you do not have an Amazon EC2 instance, you can use the CloudFormation template to launch one running Amazon Linux or Windows Server. We do not provide a template for Ubuntu Server or RHEL.

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 CloudFormation template instead of one you created by following the instructions in Step 2: Create a service role for CodeDeploy, note the following:
   > In some versions of our 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-expnyi6-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 CodeDeploy.

   If there is an entry that reads **The identity provider(s) ec2.amazonaws.com**, and you are storing your applications in GitHub repositories only, then skip ahead to Step 3: Tag the Amazon EC2 instance.

If there is an entry that reads **The identity provider(s) ec2.amazonaws.com**, and you are storing your applications in Amazon S3 buckets, choose the **Permissions** tab.

5. If there is a policy in the **Permissions policies** area, expand the policy, then choose **Edit policy**.
6. Choose the **JSON** tab. If you are storing your applications in Amazon S3 buckets, 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:

   JSON

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

   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 validates.)

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 CodeDeploy agent. Otherwise, an error might occur when the CodeDeploy agent is installed or updated on the instances. To grant the IAM instance profile access to only some 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:::amzn-s3-demo-bucket/*",
        "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-eu-central-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-north-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-2/*",
        "arn:aws:s3:::aws-codedeploy-il-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-east-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-3/*",
```

```
        "arn:aws:s3:::aws-codedeploy-ap-southeast-4/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-2/*",
        "arn:aws:s3:::aws-codedeploy-me-central-1/*",
        "arn:aws:s3:::aws-codedeploy-me-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 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 CodeDeploy agent on the Amazon EC2 instance and verify it is running, see Managing CodeDeploy agent operations, and then proceed to Create an application with CodeDeploy.

Working with on-premises instances for CodeDeploy

An on-premises instance is any physical device that is not an Amazon EC2 instance that can run the CodeDeploy agent and connect to public AWS service endpoints.

Deploying a CodeDeploy application revision to an on-premises instance involves two major steps:

- **Step 1** – Configure each on-premises instance, register it with 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 CodeDeploy (Windows Server, Ubuntu Server, or Red Hat Enterprise Linux). For information about on-premises instances and how they work with CodeDeploy, see Working with On-Premises Instances.

If you don't want an on-premises instance to be used in deployments anymore, you can 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 CodeDeploy.

The instructions in this section show you how to configure an on-premises instance and then register and tag it with CodeDeploy so it can be used in deployments. This section also describes how to use 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
- Register an on-premises instance with CodeDeploy
- Managing on-premises instances operations in CodeDeploy

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.

Device requirements

The device you want to prepare, register, and tag as an on-premises instance with CodeDeploy must be running a supported operating system. For a list, see Operating systems supported by the CodeDeploy agent.

If your operating system is not supported, the CodeDeploy agent is available as open source for you to adapt to your needs. For more information, see the CodeDeploy agent repository in GitHub.

Outbound communication

The on-premises instance must be able to connect to public AWS service endpoints to communicate with CodeDeploy.

The 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 Step 3: Limit the CodeDeploy user's permissions, make sure the calling IAM identity has the following additional policy attached.

JSON

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

For information on how to attach IAM policies, see Managing IAM policies.

Register an on-premises instance with 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 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. This option offers the highest level of security, because authentication occurs using a temporary token that times out and must be refreshed periodically. This option is recommended for production deployments of any size. For information, see Use the register-on-premises-instance command (IAM Session ARN) to register an on-premises instance.
- (Not recommended) Use an IAM user ARN to authenticate requests.
  - Use the register command for the most automated registration process. This option should only be used for non-production deployments where security is less of a concern. This option is less secure because it uses static (permanent) credentials for authentication. This option works well for registering a single on-premises instance. For information, see Use the register command (IAM user ARN) to register an on-premises instance.
  - 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.

Topics

- Use the register-on-premises-instance command (IAM Session ARN) to register an on-premises instance
- Use the register command (IAM user ARN) to register an on-premises instance
- Use the register-on-premises-instance command (IAM user ARN) to register an on-premises instance

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 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 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 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
- Step 1: Create the IAM role that on-premises instances will assume
- Step 2: Generate temporary credentials for an individual instance using AWS STS
- Step 3: Add a configuration file to the on-premises instance

- Step 4: Prepare an on-premises instance for CodeDeploy deployments
- Step 5: Register the on-premises instance with CodeDeploy
- Step 6: Tag the on-premises instance
- Step 7: Deploy application revisions to the on-premises instance
- Step 8: Track deployments to the on-premises instance

IAM session ARN registration prerequisites

In addition to the prerequisites listed in Prerequisites for configuring an on-premises instance, 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 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 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 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.

> **Note**
>
> Regardless of whether you're using method 1 or 2, you must set up a process to restart the CodeDeploy agent after the temporary session credentials are updated so that the new credentials take effect.

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 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 CodeDeploy agent:

JSON

```
{
    "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 CodeDeploy agent. Otherwise, an error might occur whenever the 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.
  - *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), 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. 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 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.

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 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` file (Linux) or `conf.onpremises.yml` file (Windows):

   ```
   ---
   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.
- *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.
- *supported-region* is one of the regions that CodeDeploy supports, as listed in Region and endpoints in *AWS General Reference*.

Step 4: Prepare an on-premises instance for 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 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 CLI in the *AWS Command Line Interface User Guide*.

   > **Note**
   >
   > 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 CLI 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.

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 CodeDeploy agent ."

Install the CodeDeploy agent on an Ubuntu Server or RHEL on-premises instance and enable instance to update the 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 CodeDeploy. We recommend that you set the value to the region where your 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 CodeDeploy agent

- For an Ubuntu Server on-premises instance, follow the instructions in Install the CodeDeploy agent for Ubuntu Server, and then return to this page.
- For a RHEL on-premises instance, follow the instructions in Install the CodeDeploy agent for Amazon Linux or RHEL, and then return to this page.
- For a Windows Server on-premises instance, follow the instructions in Install the CodeDeploy agent for Windows Server, and then return to this page.

Step 5: Register the on-premises instance with 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 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. 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 CodeDeploy does not allow, such as colon (:). For a list of allowed characters, see CodeDeploy quotas.

- 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.

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.

Step 6: Tag the on-premises instance

You can use either the AWS CLI or the CodeDeploy console to tag the on-premises instance. (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. 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 CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, and choose **On-premises instances**.
3. In the list of on-premises instances, choose name of the on-premises instance you want to tag.
4. In the list of tags, select or enter the desired tag key and tag value. After you enter the tag key and tag value, another row appears. You can repeat this for up to 10 tags. To remove a tag, choose **Remove**.
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

CodeDeploy. 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 in the Tutorial: Deploy an application to an on-premises instance with CodeDeploy (Windows Server, Ubuntu Server, or Red Hat Enterprise Linux).

> **Important**
>
> If you reuse a 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 2: Create a service role for CodeDeploy.

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 CodeDeploy deployment details.

Use the register command (IAM user ARN) to register an on-premises instance

> **Important**
>
> Registering an instance using an IAM user is not recommended because it uses static (permanent) credentials for authentication. For improved security, we recommend registering an instance using temporary credentials for authentication. For more information, see Use the register-on-premises-instance command (IAM Session ARN) to register an on-premises instance.

> **Important**
>
> Make sure you have a plan in place to rotate the IAM user's access keys (permanent credentials). For more information, see Rotating access keys.

This section describes how to configure an on-premises instance and register and tag it with 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 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 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 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 and Use the register-on-premises-instance command (IAM Session ARN) to register an on-premises instance.

Topics

- Step 1: Install and configure the AWS CLI on the on-premises instance
- Step 2: Call the register command
- Step 3: Call the install command
- Step 4: Deploy application revisions to the on-premises instance
- Step 5: Track deployments to the on-premises instance

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 CLI in the *AWS Command Line Interface User Guide*.

> **Note**
>
> 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 CLI 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. This makes it possible to download and install the CodeDeploy agent on the on-premises instance. The access permissions might look similar to this:
>
> JSON

```
{
  "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:getTagKeys",
        "tag:getTagValues",
        "tag:GetResources"
      ],
      "Resource" : "*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource" : [
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    }
  ]
}
```

Note

If you see access denied errors when trying to access one of the Amazon S3 buckets shown previously, try omitting the `/*` portion of the bucket's resource ARN, for example, `arn:aws:s3:::aws-codedeploy-sa-east-1`.

JSON

```
{
  "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:GetResources"
      ],
      "Resource" : "*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource" : [
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-bucket2/*"
      ]
    }
  ]
}
```

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 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 CodeDeploy does not allow, such as colon (`:`). For a list of allowed characters, see CodeDeploy quotas.

- 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, 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.
  >
  > 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 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 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::444455556666: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 CodeDeploy.
2. Registers the on-premises instance with CodeDeploy.
3. If specified, associates in 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 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 CodeDeploy agent (with the `--agent-installer` option). This option is useful for installing a custom version of the CodeDeploy agent that CodeDeploy does not officially support (such as a custom version based on the CodeDeploy agent repository in GitHub). The value must be the path to an Amazon S3 bucket that contains either:
  - A CodeDeploy agent installation script (for Linux- or Unix-based operating systems, similar to the install file in the CodeDeploy agent repository in GitHub).
  - A CodeDeploy agent installer package (.msi) file (for Windows-based operating systems).

If this option is not specified, CodeDeploy will make its best attempt to install from its own location an officially supported version of the 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 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 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 CodeDeploy. 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 in the Tutorial: Deploy an application to an on-premises instance with CodeDeploy (Windows Server, Ubuntu Server, or Red Hat Enterprise Linux).

> **Important**
>
> If you reuse an existing 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 2: Create a service role for CodeDeploy.

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 CodeDeploy deployment details.

For more options, see Managing on-premises instances operations in CodeDeploy.

Use the register-on-premises-instance command (IAM user ARN) to register an on-premises instance

> **Important**
>
> Registering an instance using an IAM user is not recommended because it uses static (permanent) credentials for authentication. For improved security, we recommend registering an instance using temporary credentials for authentication. For more information, see Use the register-on-premises-instance command (IAM Session ARN) to register an on-premises instance.

> **Important**
>
> Make sure you have a plan in place to rotate the IAM user's access keys (permanent credentials). For more information, see Rotating access keys.

Follow these instructions to configure an on-premises instance and register and tag it with CodeDeploy mostly on your own, using static IAM user credentials for authentication.

Topics

- Step 1: Create an IAM user for the on-premises instance
- Step 2: Assign permissions to the IAM user
- Step 3: Get the IAM user credentials
- Step 4: Add a configuration file to the on-premises instance
- Step 5: Install and configure the AWS CLI
- Step 6: Set the AWS_REGION environment variable (Ubuntu Server and RHEL only)
- Step 7: Install the CodeDeploy agent
- Step 8: Register the on-premises instance with CodeDeploy
- Step 9: Tag the on-premises instance
- Step 10: Deploy application revisions to the on-premises instance
- Step 11: Track deployments to the on-premises instance

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 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 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 and Step 8: Register the on-premises instance with CodeDeploy.

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. (You will still need information about the IAM user you created in Step 1: Create an IAM user for the on-premises instance. 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 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:::amzn-s3-demo-bucket/*",
        "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-eu-central-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-north-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-2/*",
        "arn:aws:s3:::aws-codedeploy-il-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-east-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
```

```
        "arn:aws:s3:::aws-codedeploy-ap-southeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-4/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-2/*",
        "arn:aws:s3:::aws-codedeploy-me-central-1/*",
        "arn:aws:s3:::aws-codedeploy-me-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:

```
{
    "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.
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. 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.
>
> Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.
>
> To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
IAM	(Recommended) Use console credentials as temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Login for AWS local development in the *AWS Command Line Interface User Guide*. • For AWS SDKs, see Login for AWS local development in the *AWS SDKs and Tools Reference Guide*.
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*. • For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the *AWS SDKs and Tools Reference Guide*.

Which user needs programmatic access?	To	By
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the *IAM User Guide*.
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Authenticating using IAM user credentials in the *AWS Command Line Interface User Guide*. • For AWS SDKs and tools, see Authenticate using long-term credentials in the *AWS SDKs and Tools Reference Guide*. • For AWS APIs, see Managing access keys for IAM users in the *IAM User Guide*.

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.

> **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.

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, you will need to follow the instructions in Step 4: Add a configuration file to the on-premises instance 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. a. Open the IAM console at https://console.aws.amazon.com/iam/.
   b. If the list of users is not displayed, in the navigation pane, choose **Users**.
   c. 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.
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, you will need to follow the instructions in Step 4: Add a configuration file to the on-premises instance 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.

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 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 or Step 3: Get the IAM user credentials.
   - *`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 or Step 3: Get the IAM user credentials.
   - *`iam-user-arn`* is the IAM user's ARN you noted earlier in Step 1: Create an IAM user for the on-premises instance.
   - *`supported-region`* is the identifier of a region supported by CodeDeploy where your 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, 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 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 CodeDeploy agent to download and install the 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 CLI in the *AWS Command Line Interface User Guide*.

   > **Note**
   >
   > 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 CLI 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. This makes it possible for you to download and install the CodeDeploy agent on the on-premises instance:
   >
   > JSON
   >
   > ```
   > {
   > "Version":"2012-10-17",
   > "Statement" : [
   > {
   > "Effect" : "Allow",
   > ```

```
      "Action" : [
        "codedeploy:*"
      ],
      "Resource" : "*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource" : [
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    }
  ]
}
```

JSON

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

```
  ]
}
```

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 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, 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 CodeDeploy agent.

Install the CodeDeploy agent on an Ubuntu Server or RHEL on-premises instance and enable the instance to update the 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 CodeDeploy. We recommend that you set the value to the region where your 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 CodeDeploy agent

Install the CodeDeploy agent on the on-premises instance:

- For an Ubuntu Server on-premises instance, follow the instructions in Install the CodeDeploy agent for Ubuntu Server, and then return to this page.
- For a RHEL on-premises instance, follow the instructions in Install the CodeDeploy agent for Amazon Linux or RHEL, and then return to this page.
- For a Windows Server on-premises instance, follow the instructions in Install the CodeDeploy agent for Windows Server, and then return to this page.

Step 8: Register the on-premises instance with 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.

Use the AWS CLI to register the on-premises instance with 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. 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 CodeDeploy does not allow, such as colon (`:`). For a list of allowed characters, see CodeDeploy quotas.

   - The user ARN of the IAM user you created in Step 1: Create an IAM user for the on-premises instance (with the `--iam-user-arn` option).

     For example:

     ```
     aws deploy register-on-premises-instance --instance-name AssetTag12010298EX --
     iam-user-arn arn:aws:iam::444455556666:user/CodeDeployUser-OnPrem
     ```

Step 9: Tag the on-premises instance

You can use either the AWS CLI or the CodeDeploy console to tag the on-premises instance. (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. 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 CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. From the 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 enter the desired tag key and tag value.
   After you enter 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 CodeDeploy. 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 in the Tutorial: Deploy an application to an on-premises instance with CodeDeploy (Windows Server, Ubuntu Server, or Red Hat Enterprise Linux).

> **Important**
>
> If you reuse a 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 2: Create a service role for CodeDeploy.

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 CodeDeploy deployment details.

Managing on-premises instances operations in CodeDeploy

Follow the instructions in this section to manage operations on your on-premises instances after you have registered them with 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
- Get information about multiple on-premises instances
- Manually remove on-premises instance tags from an on-premises instance
- Automatically uninstall the CodeDeploy agent and remove the configuration file from an on-premises instance
- Automatically deregister an on-premises instance
- Manually deregister an on-premises instance

Get information about a single on-premises instance

You can get information about a single on-premises instance by following the instructions in View CodeDeploy deployment details. You can use the AWS CLI or the 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 CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, and choose **On-premises instances**.
3. In the list of on-premises instances, choose the name of an on-premises instance to view its detail.

Get information about multiple on-premises instances

You can get information about on-premises instances by following the instructions in View CodeDeploy deployment details. You can use the AWS CLI or the 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 CodeDeploy console at https://console.aws.amazon.com/codedeploy.

> **Note**
>
> Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, and 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 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.

To manually deregister the on-premises instance, see Manually deregister an on-premises instance.

To automatically uninstall the CodeDeploy agent and remove the configuration file from the on-premises instance, see Automatically uninstall the CodeDeploy agent and remove the configuration file from an on-premises instance.

To manually uninstall just the CodeDeploy agent from the on-premises instance, see Managing CodeDeploy agent operations.

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 CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, and choose **On-premises instances**.
3. In the list of on-premises instances, choose the name of the on-premises instance from which you want to remove tags.
4. In **Tags**, choose **Remove** next to each tag you want to remove.
5. After you have deleted the tags, choose **Update tags**.

Automatically uninstall the CodeDeploy agent and remove the configuration file from an on-premises instance

Typically, you uninstall the 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 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.
To manually deregister the on-premises instance, see Manually deregister an on-premises instance.
To manually disassociate any associated on-premises instance tags, see Manually remove on-premises instance tags from an on-premises instance.
To manually uninstall the CodeDeploy agent from the on-premises instance, see Managing CodeDeploy agent operations.
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 CodeDeploy agent on the on-premises instance.
2. Uninstalls the 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 CodeDeploy console to deregister an on-premises instance. Also, deregistering an on-premises instance removes any on-premises instance tags that are

associated with the on-premises instance. It does not uninstall the 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 CodeDeploy console to perform some (but not all) of the activities in this section, see the CodeDeploy console section of Manually deregister an on-premises instance.
To manually disassociate any associated on-premises instance tags, see Manually remove on-premises instance tags from an on-premises instance.
To automatically uninstall the CodeDeploy agent and remove the configuration file from the on-premises instance, see Automatically uninstall the CodeDeploy agent and remove the configuration file from an on-premises instance.
To manually uninstall only the CodeDeploy agent from the on-premises instance, see Managing CodeDeploy agent operations.

Use the AWS CLI to call the deregister command, specifying:

- The name that uniquely identifies the on-premises instance to CodeDeploy (with the `--instance-name` option).
- Optionally, whether to delete the IAM user associated with the on-premises instance. The default behaviour is to delete the IAM user. If you do not want to delete the IAM user associated with the on-premises instance, specify the `--no-delete-iam-user` option in the command.
- Optionally, the AWS region where the on-premises instance was registered with 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.

An example that degisters an instance and deletes the user:

```
aws deploy deregister --instance-name AssetTag12010298EX --region us-west-2
```

An example that degisters an instance and does not delete the user:

```
aws deploy deregister --instance-name AssetTag12010298EX --no-delete-iam-user --region
 us-west-2
```

The **deregister** command does the following:

1. Deregisters the on-premises instance with CodeDeploy.
2. If specified, deletes the IAM user associated with the on-premises instance.

After you deregister an on-premises instance:

- It stops appearing in the console immediately.
- You can create another instance with the same name immediately.

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 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 CodeDeploy agent and remove the configuration file from the on-premises instance, see Automatically uninstall the CodeDeploy agent and remove the configuration file from an on-premises instance.

To manually uninstall only the CodeDeploy agent, see Managing CodeDeploy agent operations.

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.

- 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:

- It stops appearing in the console immediately.
- You can create another instance with the same name immediately.

View instance details with CodeDeploy

You can use the CodeDeploy console, the AWS CLI, or the CodeDeploy APIs to view details about instances used in a deployment.

For information about using CodeDeploy API actions to view instances, see GetDeploymentInstance, ListDeploymentInstances, and ListOnPremisesInstances.

Topics

- View instance details (console)
- View instance details (CLI)

View instance details (console)

To view instance details:

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, and then 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*. CodeDeploy is supported in these regions only.

3. To display deployment details, choose the deployment ID for the instance.
4. You can view all instances in the **Instance activity** section of the deployment's page.

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.

6. If you want to see more information about an Amazon EC2 instance, choose the ID of the instance in the **Instance ID** column.

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.)

CodeDeploy instance health

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. CodeDeploy will fail the overall deployment.

It is important to remember that for the overall deployment to succeed, the following must be true:

- 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.

Topics

- Health status
- About the minimum number of healthy instances
- About the minimum number of healthy instances per Availability Zone

Health status

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.

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 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 CodeDeploy did not attempt to deploy the application revision retains its current instance health value.
- The deployment group's revision remains the same.

About the minimum number of healthy instances

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.

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	1
CodeDeployDefault.HalfAtATime	50%
CodeDeployDefault.AllAtOnce	0

You'll find more information about default deployment configurations in Working with deployment configurations in CodeDeploy.

You can create custom deployment configurations in 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 CodeDeploy API.
- As `MinimumHealthyHosts` when you use AWS::CodeDeploy::DeploymentConfig in an CloudFormation template.

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, CodeDeploy converts the percentage to the equivalent number of instances, rounding up any fractional instances.

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**
>
> 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.

CodeDeploy starts the deployment process by attempting to deploy the application revision to the deployment group's unhealthy instances. For each successful deployment, CodeDeploy changes the instance's health status to healthy and adds it to the deployment group's healthy instances. 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, 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, CodeDeploy deploys the application revision to the original set of healthy instances.

If a deployment to a healthy instance fails, CodeDeploy changes that instance's health status to unhealthy. As the deployment progresses, 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, 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 CodeDeploy will immediately fail the overall deployment.

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, 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.

To determine the number of instances that should be deployed to at once, CodeDeploy uses the following calculation:

```
[total-hosts] - [minimum-healthy-hosts] =
        [number-of-hosts-to-deploy-to-at-once]
```

For example:

- If your deployment group has 10 instances and you set the minimum healthy instances number to 9, CodeDeploy deploys to 1 instance at a time.

- If your deployment group has 10 instances and you set the minimum healthy instances number to 3, CodeDeploy deploys to 7 instances at the same time in the first batch, and then to the remaining 3 in the second batch.
- If your deployment group has 10 instances and you set the minimum healthy instances number to 0, CodeDeploy deploys to 10 instances at the same time.

Examples

The following examples assume a deployment group with 10 instances.

Minimum healthy instances: 95%

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

CodeDeploy deploys the revision to one instance at a time. If deployment to any of the instances fails, 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.

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

CodeDeploy deploys the revision to two instances at a time. If two of these deployments fail, 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

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.

About the minimum number of healthy instances per Availability Zone

> **Note**
>
> This section uses the terms *instance* and *host* interchangeably to refer to Amazon EC2 instances.

If you're deploying to instances in several Availability Zones, you can optionally enable the zonal configuration feature, which allows CodeDeploy to deploy to one Availability Zone at a time.

When this feature is enabled, CodeDeploy will make sure that the number of healthy hosts stays above the 'minimum healthy hosts per zone' *and* the 'minimum healthy hosts' values. If the number of healthy hosts falls below either value, CodeDeploy fails the deployment across all Availability Zones.

To calculate the number of hosts to deploy to at once, CodeDeploy uses both the 'minimum healthy hosts per zone' and the 'minimum healthy hosts' values. CodeDeploy will use the *lesser* of calculations `[A]` and `[B]`, where `[A]` and `[B]` are:

```
[A] = [total-hosts] - [min-healthy-hosts] =
        [number-of-hosts-to-deploy-to-at-once]
```

```
[B] = [total-hosts-per-AZ] - [min-healthy-hosts-per-AZ] =
        [number-of-hosts-to-deploy-to-at-once-per-AZ]
```

After determining the number of hosts to deploy to at once, CodeDeploy deploys to hosts in batches of that number, one Availability Zone at a time, with an optional pause (or 'bake time') in between zones.

Example

If your deployment is configured like this:

- `[total-hosts]` is `200`
- `[minimum-healthy-hosts]` is `160`
- `[total-hosts-per-AZ]` is `100`
- `[minimum-healthy-hosts-per-AZ]` is `50`

Then...

- `[A] = 200 - 160 = 40`
- `[B] = 100 - 50 = 50`
- `40` is less than `50`

Therefore, CodeDeploy will deploy to `40` hosts at once.

In this scenario, the deployment unfolds as follows:

1. CodeDeploy deploys to the first Availability Zone:
   a. CodeDeploy deploys to the first `40` hosts.
   b. CodeDeploy deploys to the next `40` hosts.
   c. CodeDeploy deploys to the remaining `20` hosts.

   The deployment to the first Availability Zone is now complete.
2. (Optional) CodeDeploy waits while the deployment to the first zone 'bakes', as defined by the **Monitor duration** or **Add a monitor duration for the first zone** setting. If there are no problems, CodeDeploy continues.
3. CodeDeploy deploys to the second Availability Zone:
   a. CodeDeploy deploys to the first `40` hosts.
   b. CodeDeploy deploys to the next `40` hosts.
   c. CodeDeploy deploys to the remaining `20` hosts.

   The deployment to the second and final Availability Zone is now complete.

To learn about the zonal configuration feature, and how to specify the minimum number of healthy instances per Availability Zone, see zonal configuration.

Working with deployment configurations in CodeDeploy

A deployment configuration is a set of rules and success and failure conditions used by CodeDeploy during a deployment. These rules and conditions are different, depending on whether you deploy to an EC2/On-Premises compute platform, AWS Lambda compute platform, or Amazon ECS 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 and an optional 'minimum healthy hosts per zone' 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. For more information about creating custom deployment configurations, see Create a Deployment Configuration. If you don't specify a deployment configuration, CodeDeploy uses the CodeDeployDefault.OneAtATime deployment configuration.

For more information about how CodeDeploy monitors and evaluates instance health during a deployment, see Instance Health. To view a list of deployment configurations already registered to your AWS account, see View Deployment Configuration Details.

Predefined deployment configurations for an EC2/on-premises compute platform

The following table lists the predefined deployment configurations.

> **Note**
>
> There are no predefined deployment configurations that support the zonal configuration feature (which is the feature that lets you specify the number of healthy hosts per Availability Zone). If you want to use this feature, you must create your own deployment configuration.

Deployment configuration	Description
CodeDeployDefault.AllAtOnce	**In-place deployments**: Attempts to deploy an application revision to as many instances as possible at once. The status of the overall deployment is displayed as **Succeeded** if the application revision is deployed to one or more of the instances . The status of the overall deployment is displayed as **Failed** if the application revision is not deployed to any of the instances. Using an example of nine instances, CodeDeployDefault.AllAtOnce attempts to deploy to all nine instances at once. The overall deployment succeeds if deployment to even a single instance is successful. It fails only if deployments to all nine instances fail. **Blue/green deployments**: • Deployment to replacement environment: Follows the same deployment rules as CodeDeployDefault.AllAtOnce for in-place deployments. • 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	**In-place deployments**: 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).

Deployment configuration	Description
	Otherwise, the deployment fails. In the example of nine instances, it deploys to up to four instances at a time. The overall deployment succeeds if deployment to five or more instances succeed. Otherwise, the deployment fails. **Note** If you're deploying to instances in multiple Auto Scaling groups, CodeDeploy will deploy to up to half of the instances at a time *regardless of the Auto Scaling group they're in*. For example, let's assume you have two Auto Scaling groups, ASG1 and ASG2, each with 10 instances. In this scenario, CodeDeploy might deploy to 10 instances in just ASG1 and consider this a success because it has deployed to at least half of the instances. **Blue/green deployments**: • Deployment to replacement environment: Follows the 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. Otherwise, fails.

Deployment configuration	Description
CodeDeployDefault.OneAtATime	**In-place deployments**: Deploys the application revision to only one instance at a time. For deployment groups that contain more than one instance: • The overall deployment succeeds if the application revision is deployed to all of the instances. The exception to this rule is that if deployment to the last instance fails, the overall deployment still succeeds. This is because 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 deploys 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. For deployment groups that contain only one instance, the overall deployment is successfu l only if deployment to the single instance is successful. **Blue/green deployments**:

Deployment configuration	Description
	• 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 that if the last instance fails to register, the overall deployment still succeeds.

Deployment configurations on an Amazon ECS compute platform

When you deploy to an Amazon ECS compute platform, the deployment configuration specifies how traffic is shifted to the updated Amazon ECS task set. You can shift traffic using a **canary**, **linear**, or **all-at-once** deployment configuration. For more information, see Deployment configuration.

You can also create your own custom canary or linear deployment configuration. For more information, see Create a Deployment Configuration.

Predefined deployment configurations for an Amazon ECS compute platform

The following table lists the predefined configurations available for Amazon ECS deployments.

> **Note**
>
> If you're using a Network Load Balancer, only the `CodeDeployDefault.ECSAllAtOnce` predefined deployment configuration is supported.

Deployment configuration	Description
CodeDeployDefault.ECSLinear10PercentEvery1Minutes	Shifts 10 percent of traffic every minute until all traffic is shifted.
CodeDeployDefault.ECSLinear10PercentEvery3Minutes	Shifts 10 percent of traffic every three minutes until all traffic is shifted.
CodeDeployDefault.ECSCanary10Percent5Minutes	Shifts 10 percent of traffic in the first increment. The remaining 90 percent is deployed five minutes later.
CodeDeployDefault.ECSCanary10Percent15Minutes	Shifts 10 percent of traffic in the first increment. The remaining 90 percent is deployed 15 minutes later.
CodeDeployDefault.ECSAllAtOnce	Shifts all traffic to the updated Amazon ECS container at once.

Deployment configurations for CloudFormation blue/green deployments (Amazon ECS)

When you deploy to an Amazon ECS compute platform through CloudFormation blue/green deployments, the deployment configuration specifies how traffic is shifted to the updated Amazon ECS container. You can shift traffic using a **canary**, **linear**, or **all-at-once** deployment configuration. For more information, see Deployment configuration.

With CloudFormation blue/green deployments, you cannot create your own custom canary or linear deployment configuration. For step-by-step instructions on using CloudFormation to manage your Amazon ECS blue/green deployments, see Automate ECS blue/green deployments through CodeDeploy using CloudFormation in the *CloudFormation User Guide*.

> **Note**
>
> Managing Amazon ECS blue/green deployments with CloudFormation is not available in the Europe (Milan), Africa (Cape Town), and Asia Pacific (Osaka) regions.

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. You can shift traffic using a **canary**, **linear**, or **all-at-once** deployment configuration. For more information, see Deployment configuration.

You can also create your own custom canary or linear deployment configuration. For more information, see Create a 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
CodeDeployDefault.LambdaCanary10Percent5Minutes	Shifts 10 percent of traffic in the first increment. The remaining 90 percent is deployed five minutes later.
CodeDeployDefault.LambdaCanary10Percent10Minutes	Shifts 10 percent of traffic in the first increment. The remaining 90 percent is deployed 10 minutes later.
CodeDeployDefault.LambdaCanary10Percent15Minutes	Shifts 10 percent of traffic in the first increment. The remaining 90 percent is deployed 15 minutes later.
CodeDeployDefault.LambdaCanary10Percent30Minutes	Shifts 10 percent of traffic in the first increment. The remaining 90 percent is deployed 30 minutes later.
CodeDeployDefault.LambdaLinear10PercentEvery1Minute	Shifts 10 percent of traffic every minute until all traffic is shifted.

Deployment configuration	Description
CodeDeployDefault.LambdaLinear10PercentEvery2Minutes	Shifts 10 percent of traffic every two minutes until all traffic is shifted.
CodeDeployDefault.LambdaLinear10PercentEvery3Minutes	Shifts 10 percent of traffic every three minutes until all traffic is shifted.
CodeDeployDefault.LambdaLinear10PercentEvery10Minutes	Shifts 10 percent of traffic every 10 minutes until all traffic is shifted.
CodeDeployDefault.LambdaAllAtOnce	Shifts all traffic to the updated Lambda functions at once.

Topics

- Create a Deployment Configuration
- View Deployment Configuration Details
- Delete a Deployment Configuration

Create a deployment configuration with CodeDeploy

If you don't want to use one of the default deployment configurations provided with CodeDeploy, you can create your own using the following instructions.

You can use the CodeDeploy console, AWS CLI, the CodeDeploy APIs, or an CloudFormation template to create custom deployment configurations.

For information about using an CloudFormation template to create a deployment configuration, see CloudFormation templates for CodeDeploy reference.

Topics

- Creating a deployment configuration (console)
- Creating a deployment configuration with CodeDeploy (AWS CLI)

Creating a deployment configuration (console)

Use the following instructions to create a deployment configuration using the AWS console.

To create a deployment configuration in CodeDeploy using the console

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, choose **Deployment configurations**.

   A list of built-in deployment configurations appears.

3. Choose **Create deployment configuration**.
4. In **Deployment configuration name**, enter a name for the deployment configuration. For example, **`my-deployment-config`**.
5. Under **Compute platform**, choose one of the following:

   - **EC2/On-premises**
   - **AWS Lambda**
   - **Amazon ECS**

6. Do one of the following:

   - If you chose **EC2/On-premises**:

     1. Under **Minimum healthy hosts**, specify the number or percentage of instances that must remain available at any time during a deployment. For more information about how CodeDeploy monitors and evaluates instance health during a deployment, see Instance Health.
     2. (Optional) Under **Zonal configuration**, select **Enable zonal configuration** to have CodeDeploy deploy your application to one Availability Zone at a time, within an AWS Region. By deploying to one Availability Zone at a time, you can expose your deployment to a progressively larger audience as confidence in the deployment's performance and viability grows. If you don't enable a zonal configuration, CodeDeploy deploys your application to a random selection of hosts across a Region.

If you enable the zonal configuration feature, note the following:

- The zonal configuration feature is only supported with in-place deployments to Amazon EC2 instances. (Blue/green deployments and on-premises instances are not supported.) For more information about in-place deployments, see Deployment type.
- The zonal configuration feature is not supported with predefined deployment configurations. To use a zonal configuration, you must create a custom deployment configuration, as described here.
- If CodeDeploy needs to roll back a deployment, CodeDeploy will perform the rollback operations on random hosts. (CodeDeploy will not roll back one zone at a time, as you might expect.) This rollback behavior was chosen for performance reasons. For more information about rollbacks, see Redeploy and roll back a deployment with CodeDeploy.

3. If you selected the **Enable zonal configuration** check box, optionally specify the following options:

   - (Optional) In **Monitor duration**, specify the period of time, in seconds, that CodeDeploy must wait after completing a deployment to an Availability Zone. CodeDeploy will wait this amount of time before starting a deployment to the next Availability Zone. Consider adding a monitor duration to give the deployment some time to prove itself (or 'bake') in one Availability Zone before it is released in the next zone. If you don't specify a monitor duration, then CodeDeploy starts deploying to the next Availability Zone immediately. For more information about how the **Monitor duration** setting works, see About the minimum number of healthy instances per Availability Zone.
   - (Optional) Select **Add a monitor duration for the first zone** to set a monitor duration that only applies to the first Availability Zone. You might set this option if you want to allow extra bake time for the first Availability Zone. If you don't specify a value in **Add a first zone monitor duration**, then CodeDeploy uses the **Monitor duration** value for the first Availability Zone.
   - (Optional) Under **Minimum healthy hosts per zone**, specify the number or percentage of instances that must remain available per Availability Zone during a deployment. Choose **FLEET_PERCENT** to specify a percentage, or **HOST_COUNT** to specify a number. This field works in conjunction with the **Minimum healthy hosts** field. For more information, see About the minimum number of healthy instances per Availability Zone.

If you don't specify a value under **Minimum healthy hosts per zone**, then CodeDeploy uses a default value of `0` percent.

- If you chose **AWS Lambda** or **Amazon ECS**:
  1. For **Type**, choose **Linear** or **Canary**.
  2. In the **Step** and **Interval** fields, do one of the following:
     - If you chose **Canary**, for **Step**, enter a percentage of traffic, between 1 and 99, to be shifted. 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.

       For **Interval**, enter the number of minutes between the first and second traffic shift.
     - If you chose **Linear**, for **Step**, enter a percentage of traffic, between 1 and 99, to be shifted. This is the percentage of traffic that is shifted at the start of each interval.

       For **Interval**, enter the number of minutes between each incremental shift.

7. Choose **Create deployment configuration**.

   You now have a deployment configuration that you can associate with a deployment group.

Creating a deployment configuration with CodeDeploy (AWS CLI)

To use the AWS CLI to create a deployment configuration, call the create-deployment-config command.

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 EC2/On-Premises deployment configuration named `300Total50PerAZ` that requires 300 target instances to remain healthy in total per deployment, and 50 to remain healthy per Availability Zone. It also sets a monitor duration of 1 hour.

```
aws deploy create-deployment-config --deployment-config-name 300Total50PerAZ
 --minimum-healthy-hosts type=HOST_COUNT,value=300 --zonal-config
```

```
 '{"monitorDurationInSeconds":3600,"minimumHealthyHostsPerZone":
{"type":"HOST_COUNT","value":50}}'
```

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
```

The following example creates an Amazon ECS 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 ECS
```

View deployment configuration details with CodeDeploy

You can use the CodeDeploy console, the AWS CLI, or the CodeDeploy APIs to view details about deployment configurations associated with your AWS account. For descriptions of the predefined CodeDeploy deployment configurations, see Predefined deployment configurations for an EC2/on-premises compute platform.

Topics

- View deployment configuration details (console)
- View deployment configuration (CLI)

View deployment configuration details (console)

To use the CodeDeploy console to view a list of deployment configuration names:

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, and choose **Deployment configurations**.

   Here you can see the 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*. 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 CodeDeploy

You can use the AWS CLI or the 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 CodeDeploy

After you configure instances, but before you can deploy a revision, you must create an application in CodeDeploy. An *application* is simply a name or container used by 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:

Compute platform	Scenario	Information for next step
EC2/On-Premises	I haven't created instances yet.	See Working with instances for CodeDeploy, and then return to this page.
EC2/On-Premises	I have created instances, but I haven't finished tagging them.	See Tagging Instances for Deployments, and then return to this page.
EC2/On-Premises, **AWS Lambda**, and **Amazon ECS**	I haven't created an applicati on yet.	See Create an application with CodeDeploy
EC2/On-Premises, **AWS Lambda**, and **Amazon ECS**	I have already created an application, but I haven't created a deployment group.	See Create a deployment group with CodeDeploy.
EC2/On-Premises, **AWS Lambda**, and **Amazon ECS**	I have already created an application and deployment group, but I haven't created an application revision.	See Working with application revisions for CodeDeploy.
EC2/On-Premises, **AWS Lambda**, and **Amazon ECS**	I have already created an application and deploymen t group, and I have already uploaded my application revision. I'm ready to deploy.	See Create a deployment with CodeDeploy.

Topics

- Create an application with CodeDeploy
- View application details with CodeDeploy
- Create a notification rule
- Rename a CodeDeploy application
- Delete an application in CodeDeploy

Create an application with CodeDeploy

An *application* is simply a name or container used by CodeDeploy to ensure that the correct revision, deployment configuration, and deployment group are referenced during a deployment. You can use the CodeDeploy console, the AWS CLI, the CodeDeploy APIs, or an CloudFormation template to create applications.

Your code, or application revision, is installed to instances through a process called a deployment. 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.
- **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 one or more Elastic Load Balancing load balancers, 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**
>
> If you use an EC2/On-Premises compute platform, be aware that blue/green deployments work with Amazon EC2 instances only.

- **Blue/green on an AWS Lambda or Amazon ECS compute platform**: Traffic is shifted in increments according to a **canary**, **linear**, or **all-at-once** deployment configuration.
- **Blue/green deployments through CloudFormation**: Traffic is shifted from your current resources to your updated resources as part of an CloudFormation stack update. Currently, only ECS blue/green deployments are supported.

For more information about blue/green deployments, see Overview of a blue/green deployment.

When you use the 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 CodeDeploy. For information about using an CloudFormation template to create an application, see CloudFormation templates for CodeDeploy reference.

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
Amazon ECS applications	No	Yes

Topics

- Create an application for an in-place deployment (console)
- Create an application for a blue/green deployment (console)
- Create an application for an Amazon ECS service deployment (console)
- Create an application for an AWS Lambda function deployment (console)
- Create an application (CLI)

Create an application for an in-place deployment (console)

To use the 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 CodeDeploy deployments. To set up your instances, follow the instructions in Working with instances for CodeDeploy, 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, and then follow the steps in this topic.
> - You do not have a service role that trusts CodeDeploy with the minimum required trust and permissions. To create and configure a service role with the required permissions, follow the instructions in Step 2: Create a service role for CodeDeploy, 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.

To create an application for an in-place deployment using the CodeDeploy console:

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

> **Note**
>
> Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, and then choose **Getting started**.
3. Choose **Create application**.
4. In **Application name**, enter the name of your application .
5. From **Compute Platform**, choose **EC2/On-premises**.
6. Choose **Create application**.
7. On your application page, from the **Deployment groups** tab, choose **Create deployment group**.
8. In **Deployment group name**, enter 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, Amazon EC2 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, CodeDeploy treats them as separate deployment groups, because they are each associated with separate applications.

9. In **Service role**, choose a service role that grants CodeDeploy access to your target instance.
10. In **Deployment type**, choose **In-place**.
11. In **Environment configuration**, select any of the following:

    a. **Amazon EC2 Auto Scaling groups**: Enter or choose the name of an Amazon EC2 Auto Scaling group to deploy your application revision to. When new Amazon EC2 instances are launched as part of an Amazon EC2 Auto Scaling group, CodeDeploy can deploy your revisions to the new instances automatically. You can add up to 10 Amazon EC2 Auto Scaling groups to a deployment group.

    b. **Amazon EC2 instances** or **On-premises instances**: In the **Key** and **Value** fields, enter 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.

i. 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 choose **Name** in the **Key** field and enter **`GRP-*a`** in the **Value** field, CodeDeploy identifies all instances that fit that pattern, such as **`GRP-1a`**, **`GRP-2a`**, and **`GRP-XYZ-a`**.

ii. The **Value** field is case sensitive.

iii. To remove a key-value pair from the list, choose **Remove tag**.

As CodeDeploy finds instances that match each specified key-value pair or Amazon EC2 Auto Scaling group name, it displays the number of matching instances. Choose the number to see more information about the instances.

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 ten 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 Deployments.

12. In **Deployment settings**, choose a deployment configuration to control the rate your application is deployed to instances, such as one at a time or all at once. For more information about deployment configurations, see Working with deployment configurations in CodeDeploy.

13. (Optional) In **Load balancer**, select **Enable load balancing**, and then from the lists, select the Classic Load Balancers, Application Load Balancer target groups, and Network Load Balancer target groups to manage traffic to the instances during the CodeDeploy deployment. You can select up to 10 Classic Load Balancers and 10 target groups, for a total of 20 items. Make sure that the Amazon EC2 instances that you want to deploy to are registered with the selected load balancers (Classic Load Balancers) or target groups (Application Load Balancers and Network Load Balancers).

    During a deployment, the original instances are deregistered from the selected load balancers and target groups to prevent traffic from being routed to these instances during the

deployment. When the deployment is complete, each instance is re-registered with *all* the selected Classic Load Balancers and target groups.

For more information about load balancers for CodeDeploy deployments, see Integrating CodeDeploy with Elastic Load Balancing.

14. (Optional) Expand **Advanced**, and 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.

15. Choose **Create 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 CodeDeploy.

Create an application for a blue/green deployment (console)

To use the 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 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 CodeDeploy, 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, and then follow the steps in this topic.
> - You do not have a service role that trusts CodeDeploy with, at minimum, the trust and permissions described in Step 2: Create a service role for CodeDeploy. To create

and configure a service role, follow the instructions in Step 2: Create a service role for CodeDeploy, 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 CodeDeploy Amazon EC2 deployments.

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

   Note

   Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, and then choose **Getting started**.
3. In **Application name**, enter the name of your application .
4. From **Compute platform**, choose **EC2/On-Premises**.
5. Choose **Create application**.
6. On your application page, from the **Deployment groups** tab, choose **Create deployment group**.
7. In **Deployment group name**, enter 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, Amazon EC2 Auto Scaling group names, and the deployment configuration), choose those settings on this page. Although this new deployment group and the existing deployment group have the same name, CodeDeploy treats them as separate deployment groups, because each is associated with a separate application.

8. In **Service role**, choose a service role that grants CodeDeploy access to your target instance.
9. In **Deployment type** choose **Blue/green**.
10. In **Environment configuration**, choose the method to use to provide instances for your replacement environment:

a. **Automatically copy Amazon EC2 Auto Scaling group**: CodeDeploy creates an Amazon EC2 Auto Scaling group by copying one you specify.

b. **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.

11. Depending on your choice in step 10, do one of the following:

    - If you chose **Automatically copy Amazon EC2 Auto Scaling group**: In **Amazon EC2 Auto Scaling group**, choose or enter the name of the Amazon EC2 Auto Scaling group you want to use as a template for the Amazon EC2 Auto Scaling group for the instances in your replacement environment. The number of currently healthy instances in the Amazon EC2 Auto Scaling group you choose is created in your replacement environment.
    - If you chose **Manually provision instances**: Enable **Amazon EC2 Auto Scaling groups**, **Amazon EC2 instances**, or both to specify instances to add to this deployment group. Enter Amazon EC2 tag values or Amazon EC2 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).

12. In **Load balancer**, select **Enable load balancing**, and then from the lists, select the Classic Load Balancers, Application Load Balancer target groups, and Network Load Balancer target groups that you want to register your replacement Amazon EC2 instances with. Each replacement instance will be registered with *all* the selected Classic Load Balancers and target groups. You can select up to 10 Classic Load Balancers and 10 target groups, for a total of 20 items.

    Traffic will be rerouted from the original to the replacement instances according to your chosen **Traffic rerouting** and **Deployment configuration** settings.

    For more information about load balancers for CodeDeploy deployments, see Integrating CodeDeploy with Elastic Load Balancing.

13. 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 the next step. Otherwise, skip to step 15.

14. To change the deployment settings for the blue/green deployment, change any of the following settings.

Setting	Options
Traffic rerouting	• **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 specified load balancers and target groups 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 specified load balancers and target groups 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	Choose the rate at which instances in the replacement environment are registered with the load balancers and target groups, such as one at a time or all at once. **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. For more information, see Working with deployment configurations in CodeDeploy.
Original instances	• **Terminate the original instances in the deployment group**: After traffic is rerouted to the replacement environment, the instances that were deregistered from the load balancers and target groups 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 balancers and target groups are kept running.

15. (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.

16. Choose **Create 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 CodeDeploy.

Create an application for an Amazon ECS service deployment (console)

You can use the CodeDeploy console to create an application for an Amazon ECS service deployment.

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, and choose **Getting started**.
3. On the **Create application** page, choose **Use CodeDeploy**.
4. In **Application name**, enter the name of your application.
5. From **Compute platform**, choose **Amazon ECS**.
6. Choose **Create application**.
7. On your application page, from the **Deployment groups** tab, choose **Create deployment group**. For more information about what you need to create a deployment group for an Amazon ECS deployment, see Before you begin an Amazon ECS deployment.
8. In **Deployment group name**, enter 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 and the deployment configuration), choose those settings on this page. Although this new group and the existing group might have the same name,

CodeDeploy treats them as separate deployment groups, because each is associated with a separate application.

9. In **Service role**, choose a service role that grants CodeDeploy access to Amazon ECS. For more information, see Step 2: Create a service role for CodeDeploy.
10. From **Load balancer name**, choose the name of the load balancer that serves traffic to your Amazon ECS service.
11. From **Production listener port**, choose the port and protocol for the listener that serves production traffic to your Amazon ECS service.
12. (Optional) From **Test listener port**, choose the port and protocol of a test listener that serves traffic to the replacement task set in your Amazon ECS service during deployment. You can specify one or more Lambda funtions in the AppSpec file that run during the `AfterAllowTestTraffic` hook. The functions can run validation tests. If a validation test fails, a deployment rollback is triggered. If the validation tests succeed, the next hook in the deployment lifecycle, `BeforeAllowTraffic`, is triggered. If a test listener port is not specified, nothing happens during the `AfterAllowTestTraffic` hook. For more information, see AppSpec 'hooks' section for an Amazon ECS deployment.
13. From **Target group 1 name** and **Target group 2 name**, choose the target groups used to route traffic during your deployment. CodeDeploy binds one target group to your Amazon ECS service's original task set and the other to its replacement task set. For more information, see Target Groups for Your Application Load Balancers.
14. Choose **Reroute traffic immediately** or **Specify when to reroute traffic** to determine when to reroute traffic to your updated Amazon ECS service.

    If you choose **Reroute traffic immediately**, then the deployment automatically reroutes traffic after the replacement task set is provisioned.

    If you choose **Specify when to reroute traffic**, then choose the number of days, hours, and minutes to wait after the replacement task set is successfully provisioned. During this wait time, validation tests in Lambda functions specified in the AppSpec file are executed. If the wait time expires before traffic is rerouted, then the deployment status changes to `Stopped`.
15. For **Original revision termination**, choose the number of days, hours, and minutes to wait after a successful deployment before the original task set in your Amazon ECS service is terminated.
16. (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.

Create an application for an AWS Lambda function deployment (console)

You can use the CodeDeploy console to create an application for an AWS Lambda function deployment.

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, and choose **Getting started**.
3. On the **Create application** page, choose **Use CodeDeploy**.
4. Enter the name of your application in **Application name**.
5. From **Compute platform**, choose **AWS Lambda**.
6. Choose **Create application**.
7. On your application page, from the **Deployment groups** tab, choose **Create deployment group**.
8. In **Deployment group name**, enter 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 and the deployment configuration), choose those settings on this page. Although this new deployment group and the existing deployment group might have the same name, CodeDeploy treats them as separate deployment groups, because each is associated with a separate application.

9. In **Service role**, choose a service role that grants CodeDeploy access to AWS Lambda. For more information, see Step 2: Create a service role for CodeDeploy.

10. If you want to use a predefined deployment configuration, choose one from **Deployment configuration**, and then skip to step 12. To create a custom configuration, continue to the next step.

    For more information about deployment configurations, see Deployment configurations on an AWS Lambda compute platform.

11. To create a custom configuration, choose **Create deployment configuration**, and then do the following:

    a. For **Deployment configuration name**, enter a name for the configuration.

    b. From **Type**, 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.

    c. 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.

    d. In **Interval**, 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 is 2,880 divided by four, or 720 minutes.

    e. Choose **Create deployment configuration**.

12. (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.

13. Choose **Create deployment group**.

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, a 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 CodeDeploy.

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 CodeDeploy.

View application details with CodeDeploy

You can use the CodeDeploy console, the AWS CLI, or the CodeDeploy APIs to view details about all applications associated with your AWS account.

Topics

- View application details (console)
- View application details (CLI)

View application details (console)

To use the CodeDeploy console to view application details:

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, and choose **Getting started**.
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.

Create a notification rule

You can use notification rules to notify users when there are changes to deployment applications, such as deployment successes and failures. Notification rules specify both the events and the Amazon SNS topic that is used to send notifications. For more information, see What are notifications?

You can use the console or the AWS CLI to create notification rules for AWS CodeDeploy.

To create a notification rule (console)

1. Sign in to the AWS Management Console and open the CodeDeploy console at https://console.aws.amazon.com/codedeploy/.
2. Choose **Application**, and then choose an application where you want to add notifications.
3. On the application page, choose **Notify**, and then choose **Create notification rule**. You can also go to the **Settings** page for the application and choose **Create notification rule**.
4. In **Notification name**, enter a name for the rule.
5. In **Detail type**, choose **Basic** if you want only the information provided to Amazon EventBridge included in the notification. Choose **Full** if you want to include information provided to Amazon EventBridge and information that might be supplied by the CodeDeploy or the notification manager.

   For more information, see Understanding notification contents and security.
6. In **Events that trigger notifications**, select the events for which you want to send notifications.

Category	Events
Deployment	Failed Succeeded Started

7. In **Targets**, choose **Create SNS topic**.

> **Note**
>
> When you create the topic, the policy that allows CodeDeploy to publish events to the topic is applied for you. Using a topic created specifically for CodeDeploy notifications also helps ensure that you only add users to the subscription list for that topic that you want to see notifications about this deployment application.

After the **codestar-notifications-** prefix,enter a name for the topic, and then choose **Submit**.

> **Note**
>
> If you want to use an existing Amazon SNS topic instead of creating a new one, in **Targets**, choose its ARN. Make sure the topic has the appropriate access policy and that the subscriber list contains only those users who are allowed to see information about the deployment application. For more information, see Configure existing Amazon SNS topics for notifications and Understanding notification contents and security.

8. To finish creating the rule, choose **Submit**.
9. You must subscribe users to the Amazon SNS topic for the rule before they can receive notifications. For more information, see Subscribe users to Amazon SNS topics that are targets. You can also set up integration between notifications and Amazon Q Developer in chat applications to send notifications to Amazon Chime chatrooms or Slack channels. For more information, see Configure integration between notifications and Amazon Q Developer in chat applications.

To create a notification rule (AWS CLI)

1. At a terminal or command prompt, run the **create-notification rule** command to generate the JSON skeleton:

```
aws codestar-notifications create-notification-rule --generate-cli-skeleton
 > rule.json
```

   You can name the file anything you want. In this example, the file is named *rule.json*.

2. Open the JSON file in a plain-text editor and edit it to include the resource, event types, and Amazon SNS target you want for the rule. The following example shows a notification rule named **MyNotificationRule** for an application named *MyDeploymentApplication* in an AWS acccount with the ID *123456789012*. Notifications are sent with the full detail type to an Amazon SNS topic named *codestar-notifications-MyNotificationTopic* when deployments are successful:

```
{
    "Name": "MyNotificationRule",
    "EventTypeIds": [
        "codedeploy-application-deployment-succeeded"
    ],
    "Resource": "arn:aws:codebuild:us-east-2:123456789012:MyDeploymentApplication",
    "Targets": [
        {
            "TargetType": "SNS",
            "TargetAddress": "arn:aws:sns:us-east-2:123456789012:codestar-
notifications-MyNotificationTopic"
        }
    ],
    "Status": "ENABLED",
    "DetailType": "FULL"
}
```

   Save the file.

3. Using the file you just edited, at the terminal or command line, run the **create-notification-rule** command again to create the notification rule:

```
aws codestar-notifications create-notification-rule --cli-input-json
 file://rule.json
```

4. If successful, the command returns the ARN of the notification rule, similar to the following:

```
{
    "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

Rename a CodeDeploy application

You can use the AWS CLI or the 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 CodeDeploy APIs to change an application name, see API_UpdateApplication.

Delete an application in CodeDeploy

You can use the CodeDeploy console, the AWS CLI, or a CodeDeploy API action to delete applications. For information about using the CodeDeploy API action, see DeleteApplication.

> **Warning**
>
> Deleting an application removes information about the application from the 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)
- Delete an application (AWS CLI)

Delete an application (console)

To use the CodeDeploy console to delete an application:

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, then choose **Applications**.
3. In the list of applications, choose the the application you want to delete.

   A page appears containing details about the application.
4. Choose **Delete application** on the top-right.
5. When prompted, enter **`delete`** to confirm you want to delete the application, 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 CodeDeploy

You can specify one or more deployment groups for a CodeDeploy application. Each application deployment uses one of its deployment groups. The deployment group contains settings and configurations used during the deployment. Most deployment group settings depend on the compute platform used by your application. Some settings, such as rollbacks, triggers, and alarms can be configured for deployment groups for any compute platform.

Deployment groups in Amazon ECS compute platform deployments

In an Amazon ECS deployment, a deployment group specifies the Amazon ECS service, load balancer, optional test listener, and two target groups. It also specifies when to reroute traffic to the replacement task set and when to terminate the original task set and Amazon ECS application after a successful deployment.

Deployment groups in AWS Lambda compute platform deployments

In an AWS Lambda deployment, a deployment group defines a set of CodeDeploy configurations for future deployments of an AWS Lambda function. For example, the deployment group specifies how to route traffic to a new version of a Lambda function. It also 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 Amazon EC2 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 one or more load balancers 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 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 Deployments.

When you use the 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 CodeDeploy.

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. For information about Amazon EC2 Auto Scaling, see Integrating CodeDeploy with Amazon EC2 Auto Scaling.

Topics

- the section called “Create a deployment group”
- the section called “View deployment group details”
- the section called “Change deployment group settings”
- the section called “Configure advanced options for a deployment group”
- the section called “Delete a deployment group”

Create a deployment group with CodeDeploy

You can use the CodeDeploy console, the AWS CLI, the CodeDeploy APIs, or an CloudFormation template to create deployment groups. For information about using an CloudFormation template to create a deployment group, see CloudFormation templates for CodeDeploy reference.

When you use the 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 2: Create a service role for CodeDeploy.

Topics

- Create a deployment group for an in-place deployment (console)
- Create a deployment group for an EC2/On-Premises blue/green deployment (console)
- Create a deployment group for an Amazon ECS deployment (console)
- Set up a load balancer in Elastic Load Balancing for CodeDeploy Amazon EC2 deployments
- Set up a load balancer, target groups, and listeners for CodeDeploy Amazon ECS deployments
- Create a deployment group (CLI)

Create a deployment group for an in-place deployment (console)

To use the 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 CodeDeploy deployment of an application. To set up your instances, follow the instructions in Working with instances for CodeDeploy, 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, and then follow the steps in this topic.
- You do not have a service role that trusts CodeDeploy with, at minimum, the trust and permissions described in Step 2: Create a service role for CodeDeploy. To create

and configure a service role, follow the instructions in Step 2: Create a service role for CodeDeploy, 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 CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, then choose **Applications**.
3. On the **Applications** page, choose the name of the application for which you want to create a deployment group.
4. On your application page, from the **Deployment groups** tab, choose **Create deployment group**.
5. In **Deployment group name**, enter 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, Amazon EC2 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, CodeDeploy treats them as separate deployment groups, because they are each associated with separate applications.

6. In **Service role**, choose a service role that grants CodeDeploy access to your target instance.
7. In **Deployment type**, choose **In-place**.
8. In **Environment configuration**, do the following:

   a. If you want to deploy your application to an Amazon EC2 Auto Scaling group, select **Amazon EC2 Auto Scaling groups**, and then choose the name of an Amazon EC2 Auto Scaling group to deploy your application revision to. When new Amazon EC2 instances

are launched as part of an Amazon EC2 Auto Scaling group, CodeDeploy can deploy your revisions to the new instances automatically. You can add up to 10 Amazon EC2 Auto Scaling groups to a deployment group. For more information, see Integrating CodeDeploy with Amazon EC2 Auto Scaling.

b. If you selected **Amazon EC2 Auto Scaling groups**, optionally select **Add a termination hook to Auto Scaling groups** to have CodeDeploy install a termination hook into your Auto Scaling group when you create or update the deployment group. When this hook is installed, CodeDeploy will perform termination deployments. For more information, see Enabling termination deployments during Auto Scaling scale-in events.

c. If you want to tag your instances, select **Amazon EC2 instances** or **On-premises instances**. In the **Key** and **Value** fields, enter 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.

   i. 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 choose **Name** in the **Key** field and enter **`GRP-*a`** in the **Value** field, CodeDeploy identifies all instances that fit that pattern, such as **`GRP-1a`**, **`GRP-2a`**, and **`GRP-XYZ-a`**.

   ii. The **Value** field is case sensitive.

   iii. To remove a key-value pair from the list, choose the remove icon.

   As CodeDeploy finds instances that match each specified key-value pair or Amazon EC2 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 Deployments.

9. In **Agent configuration with Systems Manager**, specify how you would like to install and update the CodeDeploy agent on the instances in your deployment group. For more information on the CodeDeploy agent, see Working with the CodeDeploy agent. For more information about Systems Manager, see What is Systems Manager?

   a. **Never**: Skip configuring the CodeDeploy installation with Systems Manager. Instances must have the agent installed to be used in deployments, so only choose this option if you will install the CodeDeploy agent another way.

   b. **Only once**: Systems Manager will install the CodeDeploy agent once on every instance in your deployment group.

   c. **Now and schedule updates**: Systems Manager will create an association with State Manager that installs the CodeDeploy agent on the schedule you configure. For more information about State Manager and associations, see About State Manager.

10. In **Deployment configuration**, 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 CodeDeploy.

11. (Optional) In **Load balancer**, select **Enable load balancing**, and then from the lists, select the Classic Load Balancers, Application Load Balancer target groups, and Network Load Balancer target groups to manage traffic to the instances during the CodeDeploy deployment. You can select up to 10 Classic Load Balancers and 10 target groups, for a total of 20 items. Make sure that the Amazon EC2 instances that you want to deploy to are registered with the selected load balancers (Classic Load Balancers) or target groups (Application Load Balancers and Network Load Balancers).

    During a deployment, the original instances are deregistered from the selected load balancers and target groups to prevent traffic from being routed to these instances during the deployment. When the deployment is complete, each instance is re-registered with *all* the selected Classic Load Balancers and target groups.

    For more information about load balancers for CodeDeploy deployments, see Integrating CodeDeploy with Elastic Load Balancing.

    > **Warning**
    >
    > If you are configuring both Auto Scaling groups and an Elastic Load Balancing load balancers in this deployment group, and you want to attach load balancers to the Auto Scaling groups, we recommend completing this attachment *before* creating the

CodeDeploy deployment from this deployment group. Attempting to complete the attachment after creating the deployment may cause all the instances to become deregistered from the load balancers unexpectedly.

12. (Optional) Expand **Advanced** and configure any options you want to include in the deployment, such as Amazon SNS notification triggers, Amazon CloudWatch alarms, Auto Scaling options, or automatic rollbacks.

    For more information, see Configure advanced options for a deployment group.

13. Choose **Create deployment group**.

Create a deployment group for an EC2/On-Premises blue/green deployment (console)

To use the 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 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 CodeDeploy, 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, and then follow the steps in this topic.
- You do not have a service role that trusts CodeDeploy with, at minimum, the trust and permissions described in Step 2: Create a service role for CodeDeploy. To create and configure a service role, follow the instructions in Step 2: Create a service role for CodeDeploy, 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 CodeDeploy Amazon EC2 deployments.

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, then choose **Applications**.
3. On the **Applications** page, choose the name of the application for which you want to create a deployment group.
4. On your application page, from the **Deployment groups** tab, choose **Create deployment group**.
5. In **Deployment group name**, enter 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, Amazon EC2 Auto Scaling group names, and the deployment configuration), choose those settings on this page. Although this new deployment group and the existing deployment group have the same name, CodeDeploy treats them as separate deployment groups, because they are associated with separate applications.

6. In **Service role**, choose a service role that grants CodeDeploy access to your target instance.
7. In **Deployment type** choose **Blue/green**.
8. In **Environment configuration**, do the following:

   - Select the method to use to provide instances for your replacement environment. You have the following options:
     - **Automatically copy Amazon EC2 Auto Scaling group**: CodeDeploy creates an Amazon EC2 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.
   - If you selected **Automatically copy Amazon EC2 Auto Scaling group**, optionally select **Add a termination hook to Auto Scaling groups** to have CodeDeploy install a termination hook

into your Auto Scaling group when you create or update the deployment group. When this hook is installed, CodeDeploy will perform termination deployments. For more information, see Enabling termination deployments during Auto Scaling scale-in events.

9. In **Agent configuration with Systems Manager**, specify how you would like to install and update the CodeDeploy agent on the instances in your deployment group. For more information on the CodeDeploy agent, see Working with the CodeDeploy agent. For more information about Systems Manager, see What is Systems Manager?

   a. **Never**: Skip configuring the CodeDeploy installation with Systems Manager. Instances must have the agent installed to be used in deployments, so only choose this option if you will install the CodeDeploy agent another way.

   b. **Only once**: Systems Manager will install the CodeDeploy agent once on every instance in your deployment group.

   c. **Now and schedule updates**: Systems Manager will create an association with State Manager that installs the CodeDeploy agent on the schedule you configure. For more information about State Manager and associations, see About State Manager.

10. Depending on your choice in step 8, do one of the following:

    - If you chose **Automatically copy Amazon EC2 Auto Scaling group**: In **Amazon EC2 Auto Scaling group**, choose or enter the name of the Amazon EC2 Auto Scaling group you want to use as a template for the Amazon EC2 Auto Scaling group that is created for the instances in your replacement environment. The number of currently healthy instances in the Amazon EC2 Auto Scaling group you select is created in your replacement environment.
    - If you chose **Manually provision instances**: Select **Amazon EC2 Auto Scaling groups**, **Amazon EC2 Auto Scaling intances**, or both to specify instances to add to this deployment group. Enter Amazon EC2 Auto Scaling tag values or Amazon EC2 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).

11. In **Load balancer**, select **Enable load balancing**, and then from the lists, select the Classic Load Balancers, Application Load Balancer target groups, and Network Load Balancer target groups that you want to register your replacement Amazon EC2 instances with. Each replacement instance will be registered with *all* the selected Classic Load Balancers and target groups. You can select up to 10 Classic Load Balancers and 10 target groups, for a total of 20 items.

Traffic will be rerouted from the original to the replacement instances according to your chosen **Traffic rerouting** and **Deployment configuration** settings.

For more information about load balancers for CodeDeploy deployments, see Integrating CodeDeploy with Elastic Load Balancing.

> **Warning**
>
> If you are configuring both Auto Scaling groups and Elastic Load Balancing load balancers in this deployment group, and you want to attach the load balancers to Auto Scaling groups, we recommend completing this attachment *before* creating the CodeDeploy deployment from this deployment group. Attempting to complete the attachment after creating the deployment may cause all the instances to become deregistered from the load balancers unexpectedly.

12. 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 the next step. Otherwise, skip to step 14.

13. To change the deployment settings for the blue/green deployment, choose any of the following settings.

Setting	Options
Traffic rerouting	• **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 specified load balancers and target groups 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 environme

Setting	Options
	nt are not registered with the specified load balancers and target groups 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	Choose the rate at which instances in the replacement environment are registered with the load balancers and target groups, such as one at a time or all at once. **Note** After traffic is successfully routed to the replacement environment, instances in the original environme nt are deregistered all at once no matter which deployment configura tion was selected. For more information, see Working with deployment configurations in CodeDeploy.

Setting	Options
Original instances	• **Terminate the original instances in the deployment group**: After traffic is rerouted to the replacement environment, the instances that were deregistered from the load balancers and target groups are terminated following the wait period you specify. Note the following: • If your Auto Scaling group has Deletion Protection enabled (`prevent-force-deletion` or `prevent-all-deletion`), CodeDeploy blue/green deployment will disable this protection on the original (blue) Auto Scaling group before deleting it during deployment cleanup. The replacement (green) Auto Scaling group will maintain the same Deletion Protection settings as the original group. • **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 balancers and target groups are kept running.

14. (Optional) In **Advanced**, configure options you want to include in the deployment, such as Amazon SNS notification triggers, Amazon CloudWatch alarms, Auto Scaling options, or automatic rollbacks.

    For information about specifying advanced options in deployment groups, see Configure advanced options for a deployment group.

15. Choose **Create deployment group**.

Create a deployment group for an Amazon ECS deployment (console)

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, then choose **Applications**.
3. From the **Applications table**, choose the name of the application associated with the deployment group you want to edit.
4. On your application page, from **Deployment groups**, choose the name of the deployment group you want to edit.
5. On your application page, from the **Deployment groups** tab, choose **Create deployment group**. For more information about what you need to create a deployment group for an Amazon ECS deployment, see Before you begin an Amazon ECS deployment.
6. In **Deployment group name**, enter 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 and the deployment configuration), choose those settings on this page. Although this new group and the existing group might have the same name, CodeDeploy treats them as separate deployment groups, because each is associated with a separate application.

7. In **Service role**, choose a service role that grants CodeDeploy access to Amazon ECS. For more information, see Step 2: Create a service role for CodeDeploy.
8. From **Load balancer name**, choose the name of the load balancer that serves traffic to your Amazon ECS service.
9. From **Production listener port**, choose the port and protocol for the listener that serves production traffic to your Amazon ECS service.
10. (Optional) From **Test listener port**, choose the port and protocol of a test listener that serves traffic to the replacement task set in your Amazon ECS service during deployment. You can specify one or more Lambda funtions in the AppSpec file that run during the

`AfterAllowTestTraffic` hook. The functions can run validation tests. If a validation test fails, a deployment rollback is triggered. If the validation tests succeed, the next hook in the deployment lifecycle, `BeforeAllowTraffic`, is triggered. If a test listener port is not specified, nothing happens during the `AfterAllowTestTraffic` hook. For more information, see AppSpec 'hooks' section for an Amazon ECS deployment.

11. From **Target group 1 name** and **Target group 2 name**, choose the target groups used to route traffic during your deployment. CodeDeploy binds one target group to your Amazon ECS service's original task set and the other to its replacement task set. For more information, see Target Groups for Your Application Load Balancers.
12. Choose **Reroute traffic immediately** or **Specify when to reroute traffic** to determine when to reroute traffic to your updated Amazon ECS service.

    If you choose **Reroute traffic immediately**, then the deployment automatically reroutes traffic after the replacement task set is provisioned.

    If you choose **Specify when to reroute traffic**, then choose the number of days, hours, and minutes to wait after the replacement task set is successfully provisioned. During this wait time, validation tests in Lambda functions specified in the AppSpec file are executed. If the wait time expires before traffic is rerouted, then the deployment status changes to `Stopped`.
13. For **Original revision termination**, choose the number of days, hours, and minutes to wait after a successful deployment before the original task set in your Amazon ECS service is terminated.
14. (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.

Set up a load balancer in Elastic Load Balancing for CodeDeploy Amazon EC2 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 at least one Classic Load Balancer, Application Load Balancer, or Network 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. For in-place deployments, the load balancer is used to

deregister instances that are being worked on by CodeDeploy, and reregister them when the work is complete.

CodeDeploy supports blue/green and in-place deployment to Amazon EC2 instances behind muliple load balancers. For example, assume you have 200 Amazon EC2 instances, where 100 of them are registered with 2 Classic Load Balancers, and another 100 of them are registered with 4 target groups in 2 Application Load Balancers. In this scenario, CodeDeploy will allow you to do blue/green and in-place deployments to all 200 instances, even though they're spread across 2 Classic Load Balancers, 2 Application Load Balancers, and 4 target groups.

CodeDeploy supports up to 10 Classic Load Balancers and 10 target groups, for a total of 20 items.

To configure one or more Classic Load Balancers, 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 appears in a format such as `my-load-balancer-1234567890.us-east-2.elb.amazonaws.com`.

To configure one or more Application Load Balancers, follow the instructions in one of the following topics:

- Create an Application Load Balancer
- Tutorial: Create an Application Load Balancer using the AWS CLI

To configure one or more Network Load Balancers, follow the instructions in one of the following topics:

- Create a Network Load Balancer
- Tutorial: Create a Network Load Balancer using the AWS CLI

Set up a load balancer, target groups, and listeners for CodeDeploy Amazon ECS deployments

Before you run a deployment using the Amazon ECS compute platform, you must create an Application Load Balancer or a Network Load Balancer, two target groups, and one or two listeners. This topic shows you how to create an Application Load Balancer. For more information, see Before you begin an Amazon ECS deployment.

One of the target groups directs traffic to your Amazon ECS application's original task set. The other target group directs traffic to its replacement task set. During deployment, CodeDeploy creates a replacement task set and reroutes traffic from the original task set to the new one. CodeDeploy determines which target group is used for each task set.

A listener is used by your load balancer to direct traffic to your target groups. One production listener is required. You can specify an optional test listener that directs traffic to your replacement task set while you run validation tests.

The load balancer must use a VPC with two public subnets in different Availability Zones. The following steps show you how to confirm your default VPC, create an Amazon EC2 Application Load Balancer, and then create two target groups for your load balancer. For more information, see Target groups for your network load balancers.

Verify your default VPC, public subnets, and security group

This topic shows how to create an Amazon EC2 Application Load Balancer, two target groups, and two ports that can be used during an Amazon ECS deloyment. One of the ports is optional and needed only if you direct traffic to a test port for validation tests during your deployment.

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Verify the default VPC to use. In the navigation pane, choose **Your VPCs**. Note which VPC shows **Yes** in the **Default VPC** column. This is your default VPC. It contains default subnets that you use.
3. Choose **Subnets**. Make a note of the subnet IDs of two subnets that show **Yes** in the **Default subnet** column. You use these IDs when you create your load balancer.
4. Choose each subnet, and then choose the **Description** tab. Verify that the subnets you want to use are in different Availability Zones.

5. Choose the subnets, and then choose the **Route Table** tab. To verify that each subnet you want to use is a public subnet, confirm that a row with a link to an internet gateway is included in the route table.
6. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
7. From the navigation pane, choose **Security Groups**.
8. Verify the security group you want to use is available and make a note of its group ID (for example, **sg-abcd1234**). You use this when you create your load balancer.

Create an Amazon EC2 Application Load Balancer, two target groups, and listeners (console)

To use the Amazon EC2 console to create an Amazon EC2 Application Load Balancer:

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 **Load Balancers**.
3. Choose **Create Load Balancer**.
4. Choose **Application Load Balancer**, and then choose **Create**.
5. In **Name**, enter the name of your load balancer.
6. In **Scheme**, choose **internet-facing**.
7. In **IP address type**, choose **ipv4**.
8. (Optional) Configure a second listener port for your load balancer. You can run deployment validation tests using test traffic that is served to this port.
   a. Under **Load Balancer Protocol**, choose **Add listener**.
   b. Under **Load Balancer Protocol** for the second listener, choose **HTTP**.
   c. Under **Load Balancer Port**, enter **8080**.
9. Under **Availability Zones**, in **VPC**, choose the default VPC, and then select the two default subnets you want to use.
10. Choose **Next: Configure Security Settings**.
11. Choose **Next: Configure Security Groups**.
12. Choose **Select an existing security group**, choose the default security group, and then make a note of its ID.

13. Choose **Next: Configure Routing**.
14. In **Target group**, choose **New target group**, and configure your first target group:
    a. In **Name**, enter a target group name (for example, **`target-group-1`**).
    b. In **Target type**, choose **IP**.
    c. In **Protocol** choose **HTTP**. In **Port**, enter **`80`**.
    d. Choose **Next: Register Targets**.
15. Choose **Next: Review**, and then choose **Create**.

To create a second target group for your load balancer

1. After your load balancer is provisioned, open the Amazon EC2 console. In the navigation pane, choose **Target Groups**.
2. Choose **Create target group**.
3. In **Name**, enter a target group name (for example, **`target-group-2`**).
4. In **Target type**, choose **IP**.
5. In **Protocol** choose **HTTP**. In **Port**, enter **`80`**.
6. In **VPC**, choose the default VPC.
7. Choose **Create**.

> **Note**
>
> You must have two target groups created for your load balancer in order for your Amazon ECS deployment to run. You use the ARN of one of your target groups when you create your Amazon ECS service. For more information, see Step 4: Create an Amazon ECS service in the *Amazon ECS User Guide*.

Create an Amazon EC2 Application Load Balancer, two target groups, and listeners (CLI)

To create an Application Load Balancer using the AWS CLI:

1. Use the create-load-balancer command to create an Application Load Balancer. Specify two subnets that aren't in the same Availability Zone and a security group.

```
aws elbv2 create-load-balancer --name bluegreen-alb \
--subnets subnet-abcd1234 subnet-abcd5678 --security-groups sg-abcd1234 --
region us-east-1
```

The output includes the Amazon Resource Name (ARN) of the load balancer, in the following format:

```
arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642
```

2. Use the create-target-group command to create your first target group. CodeDeploy routes this target group's traffic to the original or the replacement task set in your service.

```
aws elbv2 create-target-group --name bluegreentarget1 --protocol HTTP --port 80 \
--target-type ip --vpc-id vpc-abcd1234 --region us-east-1
```

The output includes the ARN of the first target group, in the following format:

```
arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget1/209a844cd01825a4
```

3. Use the create-target-group command to create your second target group. CodeDeploy routes target group's traffic to the task set that is not served by your first target group. For example, if your first target group routes traffic to the original task set, this target group routes traffic to the replacement task set.

```
aws elbv2 create-target-group --name bluegreentarget2 --protocol HTTP --port 80 \
--target-type ip --vpc-id vpc-abcd1234 --region us-east-1
```

The output includes the ARN of the second target group, in the following format:

```
arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget2/209a844cd01825a4
```

4. Use the create-listener command to create a listener with a default rule that forwards production traffic to port 80.

```
aws elbv2 create-listener --load-balancer-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642 \
--protocol HTTP --port 80 \
--default-actions
 Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup
bluegreentarget1/209a844cd01825a4 --region us-east-1
```

The output includes the ARN of the listener, in the following format:

```
arn:aws:elasticloadbalancing:region:aws_account_id:listener/app/bluegreen-alb/
e5ba62739c16e642/665750bec1b03bd4
```

5. (Optional) Use the create-listener command to create a second listener with a default rule that forwards test traffic to port 8080. You can run deployment validation tests using test traffic that is served this port.

```
aws elbv2 create-listener --load-balancer-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642 \
--protocol HTTP --port 8080 \
--default-actions
 Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup
bluegreentarget2/209a844cd01825a4 --region us-east-1
```

The output includes the ARN of the listener, in the following format:

```
arn:aws:elasticloadbalancing:region:aws_account_id:listener/app/bluegreen-alb/
e5ba62739c16e642/665750bec1b03bd4
```

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. A deployment group with this name is created for the specified application. A deployment group can only be associated with one application.

- Information about the tags, tag groups, or Amazon EC2 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 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). For more information about service roles, see Roles terms and concepts in *IAM User Guide*.
- Information about the type of deployment, either in-place or blue/green, 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. If not specified, 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.
- (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.
- (Optional) Commands for a deployment to generate lifecycle event hooks during an Auto Scaling scale-in event. For more information, see How Amazon EC2 Auto Scaling works with CodeDeploy.
- For in-place deployments:
  - (Optional) The names of the Classic Load Balancers, Application Load Balancers, or Network Load Balancers in Elastic Load Balancing that manage 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 names of the Classic Load Balancers, Application Load Balancers, or Network Load Balancers in Elastic Load Balancing to be used for instances registered in the replacement environment.

> **Warning**
>
> If you are configuring both an Auto Scaling group and an Elastic Load Balancing load balancer in your deployment group, and you want to attach the load balancer to the Auto Scaling group, we recommend completing this attachment *before* creating the CodeDeploy deployment from this deployment group. Attempting to complete the attachment after creating the deployment may cause all the instances to become deregistered from the load balancer unexpectedly.

View deployment group details with CodeDeploy

You can use the CodeDeploy console, the AWS CLI, or the CodeDeploy APIs to view details about all deployment groups associated with an application.

Topics

- View deployment group details (console)
- View deployment group details (CLI)

View deployment group details (console)

To use the CodeDeploy console to view deployment group details:

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, then 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*. CodeDeploy is supported in these regions only.

4. To view details about an individual deployment group, on the **Deployment groups** tab, choose the name of 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 obtain 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 CodeDeploy

You can use the CodeDeploy console, the AWS CLI, or the 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, 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 CodeDeploy with,

at minimum, the permissions described in Step 2: Create a service role for CodeDeploy. To create and configure a service role with the correct permissions, follow the instructions in Step 2: Create a service role for CodeDeploy, and then return to this topic.

Topics

- Change deployment group settings (console)
- Change deployment group settings (CLI)

Change deployment group settings (console)

To use the CodeDeploy console to change deployment group settings:

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, then choose **Applications**.
3. In the list of applications, choose the name of 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*. CodeDeploy is supported in these regions only.

4. Choose the **Deployment groups** tab, and then choose the name of the deployment group you want to change.
5. On the **Depoyment group** page, choose **Edit**.
6. Edit the deployment group options as needed.

   For information about deployment group components, see Create a deployment group with CodeDeploy.

7. Choose **Save changes**.

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 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). 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. (If not specified, 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 for a deployment to generate lifecycle event hooks during an Auto Scaling scale-in event. For more information, see How Amazon EC2 Auto Scaling works with CodeDeploy.
  - (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.
- 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 Amazon EC2 Auto Scaling groups to be added to the deployment group.
- For Amazon ECS deployments only:
  - The Amazon ECS service to deploy.
  - Load balancer information, including the Application Load Balancer or Network Load Balancer, the target groups required for an Amazon ECS deployment, and production and optional test listener information.

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 CodeDeploy
- Create a deployment group with CodeDeploy
- Change deployment group settings with CodeDeploy

Amazon SNS notification triggers: You can add triggers to a 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 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.

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 CodeDeploy operations. For an AWS Lambda and an Amazon ECS 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, in **Alarms**, choose **Add alarm**.
2. Enter 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 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 CodeDeploy API.

For more information, see Monitoring deployments with CloudWatch alarms in CodeDeploy.

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**. 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, 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 CodeDeploy.

Automatic updates to outdated instances: Under certain circumstances, CodeDeploy may deploy an outdated revision of your application to your Amazon EC2 instances. For example, if your EC2 instances are launched into an Auto Scaling group (ASG) while a CodeDeploy deployment is underway, those instances receive the older revision of your application instead of the latest one. To bring those instances up to date, CodeDeploy automatically starts a follow-on deployment (immediatedly after the first) to update any outdated instances. If you'd like to change this default behavior so that outdated EC2 instances are left at the older revision, you can do so through the CodeDeploy API or the AWS Command Line Interface (CLI).

To configure automatic updates of outdated instances through the API, include the `outdatedInstancesStrategy` request parameter in the `UpdateDeploymentGroup` or `CreateDeploymentGroup` action. For details, see the *AWS CodeDeploy API Reference*.

To configure the automatic updates through the AWS CLI, use one of the following commands:

```
aws deploy update-deployment-group arguments --outdated-instances-strategy
UPDATE|IGNORE
```

Or...

```
aws deploy create-deployment-group arguments --outdated-instances-strategy
UPDATE|IGNORE
```

...where *arguments* is replaced with the arguments required for your deployment, and *UPDATE|IGNORE* is replaced with either `UPDATE` to enable auto-updates, or `IGNORE` to disable them.

Example:

```
aws deploy update-deployment-group --application-name "MyApp" --current-
deployment-group-name "MyDG" --region us-east-1 --outdated-instances-
strategy IGNORE
```

For details on these AWS CLI commands, see the *AWS CLI Command Reference*.

Delete a deployment group with CodeDeploy

You can use the CodeDeploy console, the AWS CLI, or the 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 CodeDeploy. The instances used in the deployment group will remain unchanged. This action cannot be undone.

Topics

- Delete a deployment group (console)
- Delete a deployment group (CLI)

Delete a deployment group (console)

To use the CodeDeploy console to delete a deployment group:

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, then 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, on the **Deployment groups** tab, choose the name of the deployment group you want to delete.
5. On the **Deployment details** page, 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 CodeDeploy

In CodeDeploy, a revision contains a version of the source files CodeDeploy will deploy to your instances or scripts 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 CodeDeploy
- Add an application specification file to a revision for CodeDeploy
- Choose a CodeDeploy repository type
- Push a revision for CodeDeploy to Amazon S3 (EC2/On-Premises deployments only)
- View application revision details with CodeDeploy
- Register an application revision in Amazon S3 with CodeDeploy

Plan a revision for CodeDeploy

Good planning makes deploying revisions much easier.

For deployments to an AWS Lambda or an Amazon ECS 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 CodeDeploy

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 CodeDeploy.

Add an application specification file to a revision for 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.

Topics

- Add an AppSpec file for an Amazon ECS deployment
- Add an AppSpec file for an AWS Lambda deployment
- Add an AppSpec file for an EC2/On-Premises deployment

Add an AppSpec file for an Amazon ECS deployment

For a deployment to an Amazon ECS compute platform:

- The AppSpec file specifies the Amazon ECS task definition used for the deployment, a container name and port mapping used to route traffic, and optional Lambda functions run after deployment lifecycle events.
- 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 when you create a deployment. For more information, see Create an Amazon ECS Compute Platform deployment (console).

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 **Create deployment**.
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.

YAML AppSpec file template for an Amazon ECS deployment with instructions

The following is a YAML template of an AppSpec file for an Amazon ECS deployment with all available options. For information about lifecycle events to use in the hooks section, see AppSpec 'hooks' section for an Amazon ECS deployment.

```
# This is an appspec.yml template file for use with an Amazon ECS deployment in
 CodeDeploy.
# The lines in this template that start with the hashtag are
#   comments that can be safely left in the file or
#   ignored.
# For help completing this file, see the "AppSpec File Reference" in the
#   "CodeDeploy User Guide" at
#   https://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html
version: 0.0
# In the Resources section, you must specify the following: the Amazon ECS service,
 task definition name,
# and the name and port of the load balancer to route traffic,
# target version, and (optional) the current version of your AWS Lambda function.
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "" # Specify the ARN of your task definition
 (arn:aws:ecs:region:account-id:task-definition/task-definition-family-name:task-
definition-revision-number)
        LoadBalancerInfo:
          ContainerName: "" # Specify the name of your Amazon ECS application's
 container
          ContainerPort: "" # Specify the port for your container where traffic
 reroutes
# Optional properties
```

```
        PlatformVersion: "" # Specify the version of your Amazon ECS Service
        NetworkConfiguration:
          AwsvpcConfiguration:
            Subnets: ["",""] # Specify one or more comma-separated subnets in your
 Amazon ECS service
            SecurityGroups: ["",""] # Specify one or more comma-separated security
 groups in your Amazon ECS service
            AssignPublicIp: "" # Specify "ENABLED" or "DISABLED"
# (Optional) In the Hooks section, specify a validation Lambda function to run during
# a lifecycle event.
Hooks:
# Hooks for Amazon ECS deployments are:
    - BeforeInstall: "" # Specify a Lambda function name or ARN
    - AfterInstall: "" # Specify a Lambda function name or ARN
    - AfterAllowTestTraffic: "" # Specify a Lambda function name or ARN
    - BeforeAllowTraffic: "" # Specify a Lambda function name or ARN
    - AfterAllowTraffic: "" # Specify a Lambda function name or ARN
```

JSON AppSpec file for an Amazon ECS deployment template

The following is a JSON template for an AppSpec file for an Amazon ECS deployment with all available options. For template instructions, refer to comments in the YAML version in the previous section. For information about lifecycle events to use in the hooks section, see AppSpec 'hooks' section for an Amazon ECS deployment.

```
{
 "version": 0.0,
 "Resources": [
  {
   "TargetService": {
    "Type": "AWS::ECS::Service",
    "Properties": {
       "TaskDefinition": "",
       "LoadBalancerInfo": {
        "ContainerName": "",
        "ContainerPort":
       },
       "PlatformVersion": "",
       "NetworkConfiguration": {
        "AwsvpcConfiguration": {
         "Subnets": [
          "",
          ""
```

```
          ],
          "SecurityGroups": [
           "",
           ""
          ],
          "AssignPublicIp": ""
         }
        }
      }
    }
   }
  ],
  "Hooks": [
   {
    "BeforeInstall": ""
   },
   {
    "AfterInstall": ""
   },
   {
    "AfterAllowTestTraffic": ""
   },
   {
    "BeforeAllowTraffic": ""
   },
   {
    "AfterAllowTraffic": ""
   }
  ]
 }
```

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).

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 **Create deployment**.
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.

YAML AppSpec file template for an AWS Lambda deployment with instructions

For information about lifecycle events to use in the hooks section, see AppSpec 'hooks' section for an AWS Lambda deployment.

```
# This is an appspec.yml template file for use with an AWS Lambda deployment in
 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
#   "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 for an AWS Lambda deployment 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.

```
{
  "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, 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 CodeDeploy to Amazon S3 (EC2/On-Premises deployments only).

AppSpec file template for an EC2/On-Premises deployment 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
 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
#   "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 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 a CodeDeploy repository type

The storage location for files required by 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 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 CodeDeploy.

- **AWS Lambda** and **Amazon ECS**: 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) and Create an Amazon ECS Compute Platform deployment (console).
  - 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) and Create an Amazon ECS Compute Platform deployment (CLI).

CodeDeploy currently supports the following repository types:

Repository Type	Repository Details	Supported Compute Platform
Amazon S3	Amazon Simple Storage Service (Amazon S3) is the AWS solution for secure, scalable object storage. Amazon S3 stores data as objects in *buckets*. An object consists of a file and, optionally, any metadata that describes that file. To store an object in Amazon S3, you upload the file to a bucket. When you upload a file, you can set permissions and metadata on the object. Learn more: • Create a bucket in Amazon S3 • Push a revision for CodeDeploy to Amazon	Deployments that use the following compute platforms can store the revision in an **Amazon S3** bucket. • EC2/On-Premises • AWS Lambda • Amazon ECS

	S3 (EC2/On-Premises deployments only) • Automatically deploy from Amazon S3 using CodeDeploy	
GitHub	You can store your applicati on revisions in GitHub repositories. You can trigger a deployment from a GitHub repository whenever the source code in that repository is changed. Learn more: • Integrating CodeDeploy with GitHub • Tutorial: Use CodeDeploy to deploy an application from GitHub	Only EC2/On-Premises deployments can store the revision in a GitHub repositor y.

Bitbucket	You can deploy code to deployment groups of EC2 instances by using the CodeDeploy pipe in Bitbucket Pipelines. Bitbucket Pipelines offers continuous integration and continuous deploymen t (CI/CD) features, including Bitbucket Deployments. The CodeDeploy pipe first pushes the artifact to an S3 bucket that you have specified, and then deploys the code artifact from the bucket. Learn more: • See the CodeDeploy pipe for Bitbucket	Only EC2/On-Premises deployments can store the revision in a BitBucket repository.

> **Note**
>
> An AWS Lambda deployment works with an Amazon S3 repository only.

Push a revision for CodeDeploy to Amazon S3 (EC2/On-Premises deployments only)

After you plan your revision as described in Plan a revision for CodeDeploy and add an AppSpec file to the revision as described in Add an application specification file to a revision for CodeDeploy, 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 CodeDeploy to deploy the revision from Amazon S3 to the instances.

> **Note**
>
> 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 CodeDeploy 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 `amzn-s3-demo-bucket`:

```
{
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "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 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 user to upload revisions anywhere in the Amazon S3 bucket named `amzn-s3-demo-bucket`:

JSON

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

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 or an Amazon ECS deployment because each 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 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. CodeDeploy zips the content before it is uploaded. Use the format `s3://amzn-s3-demo-bucket/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://amzn-s3-demo-bucket/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://amzn-s3-demo-
bucket/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 `amzn-s3-demo-bucket`.
- Bundles all files in the root directory into the revision.

After the push is successful, you can use the AWS CLI or the 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 \
  --s3-location bucket=amzn-s3-demo-bucket,key=WordPressApp.zip,bundleType=zip
```

In Windows:

```
aws deploy create-deployment --application-name WordPress_App --deployment-config-
name your-deployment-config-name --deployment-group-name your-deployment-group-name --
s3-location bucket=amzn-s3-demo-bucket,key=WordPressApp.zip,bundleType=zip
```

For more information, see Create a deployment with CodeDeploy.

View application revision details with CodeDeploy

You can use the CodeDeploy console, the AWS CLI, or the 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 CodeDeploy.

Topics

- View application revision details (console)
- View application revision details (CLI)

View application revision details (console)

To view application revision details:

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy** and 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*. CodeDeploy is supported in these regions only.

3. Choose the name of the application that has the revision you want to view.
4. On the **Application details** page, choose the **Revisions** tab, and review the list of revisions that are registered for the application. Choose a revision, then choose **View 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, 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, CodeDeploy will list all matching revisions.)
- Optionally, the column name and order by which to sort the list of revision details. (If not specified, 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 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 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 CodeDeploy console or through the AWS CLI, you must also follow these steps.

You can use only the AWS CLI or the CodeDeploy APIs to register application revisions in Amazon S3 or GitHub.

Topics

- Register a revision in Amazon S3 with CodeDeploy (CLI)
- Register a revision in GitHub with CodeDeploy (CLI)

Register a revision in Amazon S3 with 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 CodeDeploy will deploy to your instances or scripts that 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, CodeDeploy will use the most recent version.)
     - (Optional) The revision's ETag. (If the ETag is not specified, 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 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 CodeDeploy

In 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. CodeDeploy deploys content that is stored in a source repository, according to the configuration rules you specify.

If you use the EC2/On-Premises compute platform, then two deployments to the same set of instances can run concurrently.

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.
- **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 one or more Elastic Load Balancing load balancers, 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**
>
> If you use an EC2/On-Premises compute platform, be aware that blue/green deployments work with Amazon EC2 instances only.

- **Blue/green on an AWS Lambda or Amazon ECS compute platform**: Traffic is shifted in increments according to a **canary**, **linear**, or **all-at-once** deployment configuration.
- **Blue/green deployments through CloudFormation**: Traffic is shifted from your current resources to your updated resources as part of an CloudFormation stack update. Currently, only ECS blue/green deployments are supported.

For more information about blue/green deployments, see Overview of a blue/green deployment.

For information about automatically deploying from Amazon S3, see Automatically deploy from Amazon S3 using CodeDeploy.

Topics

- Create a deployment with CodeDeploy
- View CodeDeploy deployment details
- View log data for CodeDeploy EC2/On-Premises deployments
- Stop a deployment with CodeDeploy
- Redeploy and roll back a deployment with CodeDeploy
- Deploy an application in a different AWS account
- Use the CodeDeploy agent to validate a deployment package on a local machine

Create a deployment with CodeDeploy

You can use the CodeDeploy console, the AWS CLI, or the 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
- Create an Amazon ECS Compute Platform deployment (console)
- Create an AWS Lambda Compute Platform deployment (console)
- Create an EC2/On-Premises Compute Platform deployment (console)

- Create an Amazon ECS Compute Platform deployment (CLI)
- Create an AWS Lambda Compute Platform deployment (CLI)
- Create an EC2/On-Premises Compute Platform deployment (CLI)
- Create an Amazon ECS blue/green deployment through CloudFormation

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 CodeDeploy and Create a deployment group with CodeDeploy.
- 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 CodeDeploy.
- If you want to use a custom deployment configuration for your deployment, create it before you start the deployment process. For information, see Create a Deployment Configuration.

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 CodeDeploy. For a blue/green deployment, you either have an existing Amazon EC2 Auto Scaling group to use as a template for your replacement environment, or you have one or more instances or Amazon EC2 Auto Scaling groups that you specify as your original environment. For more information, see Tutorial: Use CodeDeploy to deploy an application to an Auto Scaling group and Integrating CodeDeploy with Amazon EC2 Auto Scaling.
- Create an application that includes at least one deployment group. For information, see Create an application with CodeDeploy and Create a deployment group with CodeDeploy.
- Prepare the application revision that you want to deploy to the instances in your deployment group. For information, see Working with application revisions for CodeDeploy.
- If you want to use a custom deployment configuration for your deployment, create it before you start the deployment process. For information, see Create a Deployment Configuration.

- 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::444455556666:role/CodeDeployDemo` to download from anywhere in the Amazon S3 bucket named `amzn-s3-demo-bucket`:

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/CodeDeployDemo"
                ]
            }
        }
    ]
}
```

  The following Amazon S3 bucket policy allows any on-premises instance with an associated IAM user containing the ARN `arn:aws:iam::444455556666:user/CodeDeployUser` to download from anywhere in the Amazon S3 bucket named `amzn-s3-demo-bucket`:

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
```

```
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:user/CodeDeployUser"
                ]
            }
        }
    ]
}
```

For information about 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. (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 at least one 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.

  For more information about configuring a load balancer, see Set up a load balancer in Elastic Load Balancing for CodeDeploy Amazon EC2 deployments, and Set up a load balancer, target groups, and listeners for CodeDeploy Amazon ECS deployments.

Deployment prerequisites for a blue/green deployment through CloudFormation

- Your template does not need to model resources for a CodeDeploy application or deployment group.

- Your template must include resources for a VPC using Amazon VPC that contains at least two subnets.
- Your template must include resources for one or more Classic Load Balancers, Application Load Balancers, or Network Load Balancers in Elastic Load Balancing that are used to direct traffic to your target groups.

Create an Amazon ECS Compute Platform deployment (console)

This topic shows you how to deploy an Amazon ECS service using the console. For more information, see Tutorial: Deploy an application into Amazon ECS and Tutorial: Deploy an Amazon ECS service with a validation test.

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. Do one of the following:

   - If you want to deploy an application, in the navigation pane, expand **Deploy**, and then choose **Applications**. Choose the name of the application you want to deploy. Make sure the **Compute platform** column for your application is **Amazon ECS**.
   - If you want to redeploy a deployment, in the navigation pane, expand **Deploy**, and then choose **Deployments**. Choose the deployment you want to redeploy, and in the **Application** column, choose the name of its application. Make sure the **Compute platform** column for your deployment is **Amazon ECS**.

3. On the **Deployments** tab, choose **Create deployment**.

   > **Note**
   >
   > Your application must have a deployment group before it can be deployed. If your application does not have a deployment group, on the **Deployment groups** tab, choose **Create deployment group**. For more information, see Create a deployment group with CodeDeploy.

4. In **Deployment group**, choose a deployment group to use for this deployment.
5. Next to **Revision location**, choose where your revision is located:
   - **My application is stored in Amazon S3** — For information, see Specify information about a revision stored in an Amazon S3 bucket, and then return to step 6.
   - **Use AppSpec editor** — Select either JSON or YAML, and then enter your AppSpec file into the editor. You can save the AppSpec file by choosing **Save as text file**. When you choose **Deploy** at the end of these steps, you 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 CodeDeploy.
6. (Optional) In **Deployment description**, enter a description for this deployment.
7. (Optional) In **Rollback configuration overrides**, you can specify different automatic rollback options for this deployment than were specified for the deployment group, if any.

   For information about rollbacks in CodeDeploy, see Redeployments and deployment rollbacks and Redeploy and roll back a deployment with CodeDeploy.

   Choose from the following:

   - **Roll back when a deployment fails** — CodeDeploy redeploys the last known good revision as a new deployment.
   - **Roll back when alarm thresholds are met** — If alarms were added to the deployment group, CodeDeploy redeploys the last known good revision when one or more of the specified alarms is activated.
   - **Disable rollbacks** — Do not perform rollbacks for this deployment.
8. Choose **Create deployment**.

   To track the status of your deployment, see View CodeDeploy deployment details.

Create an AWS Lambda Compute Platform deployment (console)

This topic shows you how to deploy a Lambda function using the console.

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

> **Note**
>
> Sign in with the same user that you set up in Getting started with CodeDeploy.

2. Do one of the following:

   - If you want to deploy an application, in the navigation pane, expand **Deploy**, and then choose **Applications**. Choose the name of the application you want to deploy. Make sure the **Compute platform** column for your application is **AWS Lambda**.
   - If you want to redeploy a deployment, in the navigation pane, expand **Deploy**, and then choose **Deployments**. Choose the deployment you want to redeploy, and in the **Application** column, choose the name of its application. Make sure the **Compute platform** column for your deployment is **AWS Lambda**.

3. On the **Deployments** tab, choose **Create deployment**.

> **Note**
>
> Your application must have a deployment group before it can be deployed. If your application does not have a deployment group, on the **Deployment groups** tab, choose **Create deployment group**. For more information, see Create a deployment group with CodeDeploy.

4. In **Deployment group**, choose a deployment group to use for this deployment.
5. Next to **Revision location**, choose where your revision is located:

   - **My application is stored in Amazon S3** — For information, see Specify information about a revision stored in an Amazon S3 bucket, and then return to step 6.
   - **Use AppSpec editor** — Select either JSON or YAML, and then enter your AppSpec file into the editor. You can save the AppSpec file by choosing **Save as text file**. When you choose **Deploy** at the end of these steps, you 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 CodeDeploy.

6. (Optional) In **Deployment description**, enter a description for this deployment.
7. (Optional) Expand **Deployment group overrides** to choose a deployment configuration to control how traffic is shifted to the Lambda function version that is different from the one specified in the deployment group.

For more information, see Deployment configurations on an AWS Lambda compute platform.

8. (Optional) In **Rollback configuration overrides**, you can specify different automatic rollback options for this deployment than were specified for the deployment group, if any.

   For information about rollbacks in CodeDeploy, see Redeployments and deployment rollbacks and Redeploy and roll back a deployment with CodeDeploy.

   Choose from the following:

   - **Roll back when a deployment fails** — CodeDeploy redeploys the last known good revision as a new deployment.
   - **Roll back when alarm thresholds are met** — If alarms were added to the deployment group, CodeDeploy redeploys the last known good revision when one or more of the specified alarms is activated.
   - **Disable rollbacks** — Do not perform rollbacks for this deployment.

9. Choose **Create deployment**.

   To track the status of your deployment, see View CodeDeploy deployment details.

Create an EC2/On-Premises Compute Platform deployment (console)

This topic shows you how to deploy an application to an Amazon EC2 or on-premises server using the console.

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. Do one of the following:

   - If you want to deploy an application, in the navigation pane, expand **Deploy**, and then choose **Applications**. Choose the name of the application you want to deploy. Make sure the **Compute platform** column for your application is **EC2/On-Premises**.

- If you want to redeploy a deployment, in the navigation pane, expand **Deploy**, and then choose **Deployments**. Locate the deployment you want to redeploy, and then choose the name of its application in the **Application** column. Make sure the **Compute platform** column for your deployment is **EC2/On-Premises**.

3. On the **Deployments** tab, choose **Create deployment**.

> **Note**
>
> Your application must have a deployment group before it can be deployed. If your application does not have a deployment group, on the **Deployment groups** tab, choose **Create deployment group**. For more information, see Create a deployment group with CodeDeploy.

4. In **Deployment group**, choose a deployment group to use for this deployment.
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, 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, and then return to step 6.

6. (Optional) In **Deployment description**, enter a description for this deployment.
7. (Optional) Expand **Override deployment configuration** to choose a deployment configuration to control how traffic is shifted to the Amazon EC2 or on-premises server that is different from the one specified in the deployment group.

   For more information, see Working with deployment configurations in CodeDeploy.

8. a. Select **Don't fail the deployment if the ApplicationStop lifecycle event fails** if you want a deployment to an instance to succeed if the `ApplicationStop` lifecycle event fails.

   b. Expand **Additional deployment behavior settings** to specify how 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, the file is kept and the version in the application revision is not copied to the instance.

For more information, see Rollback behavior with existing content.

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.

   For information about rollbacks in CodeDeploy, see Redeployments and deployment rollbacks and Redeploy and roll back a deployment with CodeDeploy.

   Choose from the following:

   - **Roll back when a deployment fails** — CodeDeploy redeploys the last known good revision as a new deployment.
   - **Roll back when alarm thresholds are met** — If alarms were added to the deployment group, CodeDeploy deploys 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 **Start deployment**.

    To track the status of your deployment, see View CodeDeploy deployment details.

Topics

- Specify information about a revision stored in an Amazon S3 bucket
- Specify information about a revision stored in a GitHub repository

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), 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 **Revision location**. To find the link value:

a. 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.

b. If the **Properties** pane is not visible, choose the **Properties** button.

c. In the **Properties** pane, copy the value of the **Link** field into the **Revision location** box in the 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 following the steps in Create an EC2/On-Premises Compute Platform deployment (console), 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 CodeDeploy applications to a GitHub account, in a different web browser tab, sign out of GitHub. In **GitHub account**, enter a name to identify this connection, and then choose **Connect to GitHub**. The webpage prompts you to authorize 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, in a different web browser tab, sign out of GitHub. 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, enter 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:

   a. In a different web browser tab, go to your GitHub dashboard.

   b. 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. Enter 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 **Commit ID**, enter the ID of the commit that refers to the revision in the repository. If you are unsure of the value to type:

   a. In a different web browser tab, go to your GitHub dashboard.

   b. In **Your repositories**, choose the repository name that contains the target commit.

c. 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.)

d. Paste the commit ID into the **Commit ID** box.

Create an Amazon ECS Compute Platform deployment (CLI)

After you have created the application and revision (in Amazon ECS 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, CodeDeploy uses the most recent version.)
  - (Optional) The ETag for the revision. (If the ETag is not specified, 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) 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. For an example revision, see AppSpec File example for an AWS Lambda deployment.

To track the status of your deployment, see View CodeDeploy deployment details.

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, CodeDeploy uses the most recent version.)

- (Optional) The ETag for the revision. (If the ETag is not specified, 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, 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. For an example revision, see AppSpec File example for an AWS Lambda deployment.

To track the status of your deployment, see View CodeDeploy deployment details.

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 a CodeDeploy application to a GitHub repository, and then continue to step 2.
2. Call the create-deployment command, specifying:
   - `--application-name`: An application name. To view a list of application names, call the list-applications command.
   - `--deployment-group-name`: An Amazon EC2 deployment group name. To view a list of deployment group names, call the list-deployment-groups command.
   - `--revision`: Information about the revision to be deployed:

     For revisions stored in Amazon S3:

     - `s3Location`: The Amazon S3 bucket name that contains the revision.
     - `s3Location --> key`: The name of the uploaded revision.
     - `s3Location --> bundleType`: The 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.

     - `s3Location --> version`: (Optional) The Amazon S3 version identifier for the revision. (If the version identifier is not specified, CodeDeploy uses the most recent version.)
     - `s3Location --> eTag`: (Optional) The ETag for the revision. (If the ETag is not specified, CodeDeploy skips object validation.)

     For revisions stored in GitHub:

     - `gitHubLocation --> repository`: The GitHub user or group name assigned to the repository that contains the revision, followed by a forward slash (/), followed by the repository name.
     - `gitHubLocation --> commitId`: The commit ID for the revision.
   - `--deployment-config-name`: (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, CodeDeploy uses a specific default deployment configuration.)

- `--ignore-application-stop-failures | --no-ignore-application-stop-failures`: (Optional) Whether you want the deployment to an instance to continue to the `BeforeInstall` deployment lifecycle event if the `ApplicationStop` deployment lifecycle event fails.
- `--description`: (Optional) A description for the deployment.
- `--file-exists-behavior`: (Optional) As part of the deployment process, the CodeDeploy agent removes from each instance all the files installed by the most recent deployment. Choose what happens when files that weren't part of a previous deployment appear in target deployment locations.
- `--target-instances`: 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 Amazon EC2 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 CodeDeploy deployment details.

create-deployment command reference

Below is the command structure and options for the `create-deployment` command. For more information, see the create-deployment reference in the *AWS CLI Command Reference*.

```
create-deployment
--application-name <value>
[--deployment-group-name <value>]
[--revision <value>]
[--deployment-config-name <value>]
[--description <value>]
[--ignore-application-stop-failures | --no-ignore-application-stop-failures]
[--target-instances <value>]
[--auto-rollback-configuration <value>]
[--update-outdated-instances-only | --no-update-outdated-instances-only]
[--file-exists-behavior <value>]
[--s3-location <value>]
[--github-location <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton <value>]
```

Connect a 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 CodeDeploy permission to interact with GitHub on behalf of your GitHub account. This step must be completed once for each application using the CodeDeploy console.

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. Choose **Applications**.
3. From **Applications**, choose the application you want to link to your GitHub user account and choose **Deploy application**.

   > **Note**
   >
   > You are not creating a deployment. This is currently the only way to give CodeDeploy permission to interact with GitHub on behalf of your GitHub user account.

4. Next to **Repository type**, choose **My application revision is stored in GitHub**.

5. Choose **Connect to GitHub**.

> **Note**
>
> If you see a **Connect to a different GitHub account** link:
> You might have already authorized CodeDeploy to interact with GitHub on behalf of a different GitHub account for the application.
> You might have revoked authorization for CodeDeploy to interact with GitHub on behalf of the signed-in GitHub account for all applications linked to in CodeDeploy.
> For more information, see GitHub authentication with applications in CodeDeploy.

6. If you are not already signed in to GitHub, follow the instructions on the **Sign in** page.
7. On the **Authorize application** page, choose **Authorize application**.
8. Now that CodeDeploy has permission, choose **Cancel**, and continue with the steps in Create an EC2/On-Premises Compute Platform deployment (CLI).

Create an Amazon ECS blue/green deployment through CloudFormation

You can use AWS CloudFormation to manage Amazon ECS blue/green deployments through CodeDeploy. You generate your deployment by defining your green and blue resources and specifying the traffic routing and stabilization settings to use in CloudFormation. This topic covers the differences between Amazon ECS blue/green deployments that are managed by CodeDeploy and deployments that are managed by CloudFormation.

For step-by-step instructions on using CloudFormation to manage your Amazon ECS blue/green deployments, see Automate ECS blue/green deployments through CodeDeploy using AWS CloudFormation in the *AWS CloudFormation User Guide*.

> **Note**
>
> Managing Amazon ECS blue/green deployments with CloudFormation is not available in the Asia Pacific (Osaka) region.

Differences between Amazon ECS blue/green deployments through CodeDeploy and CloudFormation

The CloudFormation stack template models Amazon ECS task-related resources and infrastructure, and also the configuration options for deployments. So there are differences between the standard Amazon ECS blue/green deployments and blue/green deployments that are created through CloudFormation.

Unlike standard Amazon ECS blue/green deployments, you don't model or manually create the following:

- You don't create an AWS CodeDeploy application by specifying a name that uniquely represents what you want to deploy.
- You don't create an AWS CodeDeploy deployment group.
- You don't specify an *application specification file* (AppSpec file). The information normally managed with the AppSpec file, such as the weighted configuration options or lifecycle events, is managed by the `AWS::CodeDeploy::BlueGreen` hook.

This table summarizes the differences in the high-level workflow between deployment types.

Function	Standard blue/green deployments	Blue/green deployments through CloudFormation
Specify the Amazon ECS cluster, Amazon ECS service, Application Load Balancer or Network Load Balancer, Production listener, test listener, and two target groups.	Create a CodeDeploy deployment group that specifies these resources.	Create an CloudFormation template to model these resources.
Specify the change to be deployed.	Create a CodeDeploy applicati on.	Create an CloudFormation template that specifies the container image.

Function	Standard blue/green deployments	Blue/green deployments through CloudFormation
Specify the Amazon ECS task definition, container name, and container port.	Create an AppSpec file that specifies these resources.	Create an CloudFormation template to model these resources.
Specify the deployment traffic shifting options and lifecycle event hooks.	Create an AppSpec file that specifies these options.	Create an CloudFormation template that uses the `AWS::CodeDeploy::BlueGreen` hook parameters to specify these options.
CloudWatch alarms.	Create a CloudWatch alarm that triggers a rollback.	Configure a CloudWatch alarm at the CloudFormation stack level that triggers a rollback.
Rollback/redeployment.	Specify rollback and redeployment options.	Cancel the stack update in CloudFormation.

Monitoring Amazon ECS blue/green deployments through CloudFormation

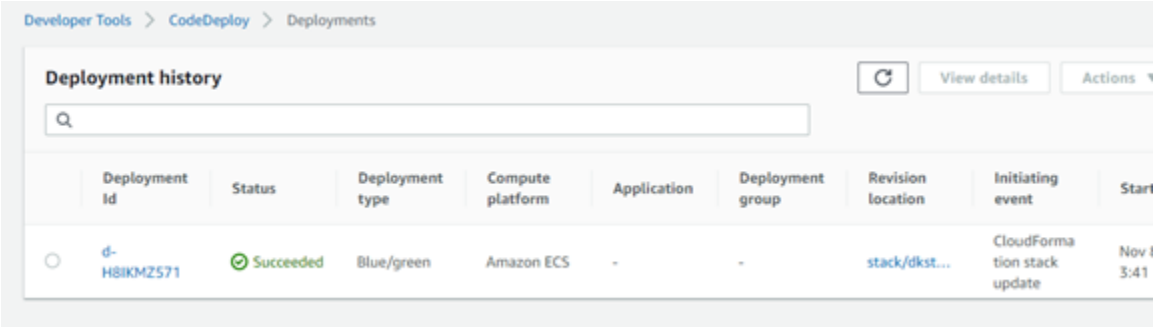
You can monitor blue/green deployments through CloudFormation and CodeDeploy. For information about monitoring through CloudFormation, see Monitoring blue/green events in CloudFormation in the *AWS CloudFormation User Guide*.

To view deployment status of blue/green deployments in CodeDeploy

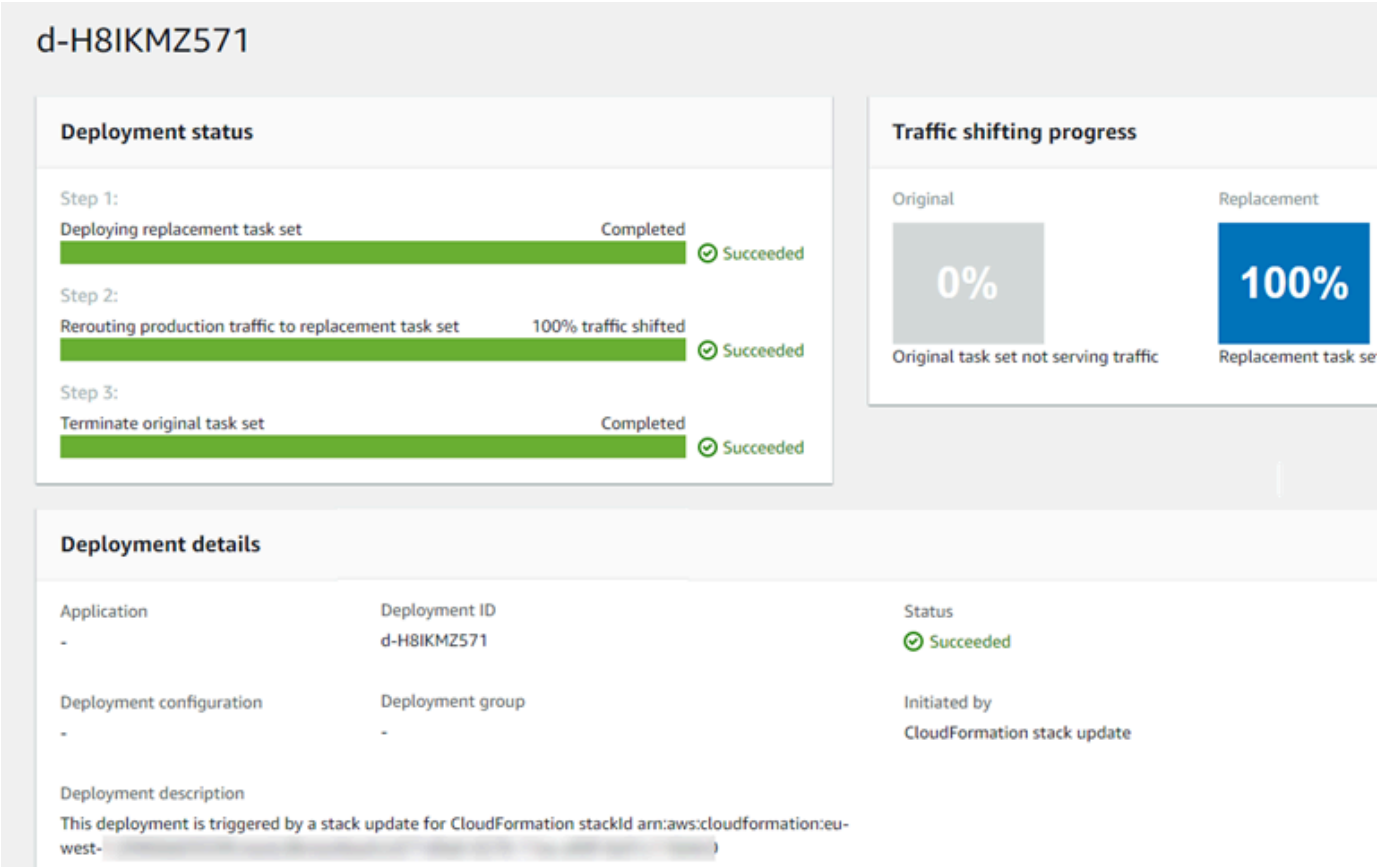
1. Sign in to the AWS Management Console and open the CodeDeploy console at https://console.aws.amazon.com/codedeploy.

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In **Deployments**, the deployment that was triggered by the CloudFormation stack update appears. Choose the deployment to view the **Deployment history**.



3. Choose the deployment to view the traffic shifting status. Note that the application and deployment group are not created.



4. The following apply for rolling back or stopping the deployment:

   - The successful deployment appears in CodeDeploy and shows that the deployment was initiated by CloudFormation.
   - If you want to stop and roll back the deployment, you must cancel the stack update in CloudFormation.

View CodeDeploy deployment details

You can use the CodeDeploy console, the AWS CLI, or the CodeDeploy APIs to view details about deployments associated with your AWS account.

> **Note**
>
> You can view EC2/On-Premises 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><DEPLOYMENT-ID>\logs\scripts.log
>
> For more information, see Analyzing log files to investigate deployment failures on instances.

Topics

- View deployment details (console)
- View deployment details (CLI)

View deployment details (console)

To use the CodeDeploy console to view deployment details:

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, and then 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*. CodeDeploy is supported in these regions only.

3. To see more details for a single deployment, in **Deployment history**, choose the deployment ID or choose the button next to the deployment ID, and then choose **View**.

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 CodeDeploy EC2/On-Premises deployments

You can view the log data created by a CodeDeploy deployment by setting up the Amazon CloudWatch agent to view aggregated data in the CloudWatch console or by logging into an individual instance to review the log file.

> **Note**
>
> Logs are not supported for AWS Lambda or Amazon ECS deployments. They can be created for EC2/On-Premises deployments only.

Topics

- View log file data in the Amazon CloudWatch console
- View log files on an instance

View log file data in the Amazon CloudWatch console

When the Amazon CloudWatch agent is installed on an instance, deployment data for all deployments to that instance becomes available for viewing in the CloudWatch console. For simplicity, we recommend using CloudWatch to centrally monitor log files instead of viewing them instance by instance. For more information, see Send CodeDeploy agent logs to CloudWatch.

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
- To view deployment log files on Windows Server instances

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 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.

[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 a 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 log files on Windows Server instances

CodeDeploy agent log file: On Windows Server instances, the CodeDeploy agent log file is stored at the following location:

`C:\ProgramData\Amazon\CodeDeploy\log\codedeploy-agent-log.txt`

To view or analyze the 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.

CodeDeploy scripts log files: On Windows Server instances, deployment logs are stored at the following location:

`C:\ProgramData\Amazon\CodeDeploy\`*`deployment-group-id`*`\`*`deployment-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 a 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 CodeDeploy

You can use the CodeDeploy console, the AWS CLI, or the 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.

You can either stop a deployment or stop and roll back a deployment.

- Stop a deployment (console)
- Stop a deployment (CLI)

> **Note**
>
> If your deployment is a blue/green deployment through CloudFormation, you cannot perform this task in the CodeDeploy console. Go to the CloudFormation console to perform this task.

Stop a deployment (console)

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, and then 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*. CodeDeploy is supported in these regions only.

3. Choose the deployment you want to stop do one of the following:

   1. Choose **Stop deployment** to stop the deployment without a rollback.
   2. Choose **Stop and roll back deployment** to stop and roll back the deployment

   For more information, see Redeploy and roll back a deployment with CodeDeploy.

   > **Note**
   >
   > If **Stop deployment** and **Stop and roll back deployment** are unavailable, 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 CodeDeploy

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
- Manual rollbacks
- Rollback and redeployment workflow
- Rollback behavior with existing content

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.

For more information about configuring automatic rollbacks, see Configure advanced options for a deployment group.

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 CodeDeploy.

> **Note**
>
> If you remove an instance from a deployment group, 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, CodeDeploy first tries to remove from each participating instance all files that were last successfully installed. 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, 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 CodeDeploy)
c:\temp\b.txt (previously deployed by CodeDeploy)
c:\temp\c.bat (previously deployed by CodeDeploy)
c:\temp\d.bat (previously deployed by 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, 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, 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 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 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 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 can choose this behavior when you create a deployment. If creating a deployment in the console, see Create an EC2/On-Premises Compute Platform deployment (console). If creating a deployment with the AWS CLI, see Create an EC2/On-Premises Compute Platform deployment (CLI).

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 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	CodeDeploy detects files in the target locations that were not deployed by the previous deploymen t. These files might be

Deployment	Application revision	Content overwrite option	Deployment status	Behavior and result
				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, 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		Because auto rollback is enabled for the deployment or deployment group, CodeDeploy deploys the last known good application revision, application revision 1. 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

Deployment	Application revision	Content overwrite option	Deployment status	Behavior and result
				new application revision.

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, 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.

  > **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.

    JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Cross-account permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333: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 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 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 a 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 CodeDeploy tasks, see AWS managed (predefined) policies for CodeDeploy.

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 CodeDeploy to Amazon S3 (EC2/On-Premises deployments only).

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).

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).

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:

- CodeDeploy: Deploying from a development account to a production account

Use the CodeDeploy agent to validate a deployment package on a local machine

Using the 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 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 CodeDeploy agent. For information, see Operating systems supported by the CodeDeploy agent.
- Install version 1.0.1.1352 or later of the CodeDeploy agent. For information, see Install the CodeDeploy agent.
- If you are deploying your content from an Amazon S3 bucket or GitHub repository, provision a user to use with CodeDeploy. For information, see Step 1: Setting up.
- 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::444455556666:role/CodeDeployDemo` to download from anywhere in the Amazon S3 bucket named `amzn-s3-demo-bucket`:

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/CodeDeployDemo"
                ]
            }
        }
    ]
}
```

The following Amazon S3 bucket policy allows any on-premises instance with an associated IAM user containing the ARN `arn:aws:iam::444455556666:user/CodeDeployUser` to download from anywhere in the Amazon S3 bucket named `amzn-s3-demo-bucket`:

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666: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, Create an Amazon EC2 instance for CodeDeploy (AWS CLI or Amazon EC2 console), and Create an Amazon EC2 instance for CodeDeploy (CloudFormation template).
- 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 CodeDeploy.
- 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 CodeDeploy agent configuration reference.

> **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 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>]
[--appspec-filename <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://amzn-s3-demo-bucket/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 CodeDeploy deployments. For information, see AppSpec 'hooks' section.

-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 CodeDeploy agent configuration reference.

-A, --appspec-filename

The name of the AppSpec file. For local deployments, accepted values are `appspec.yml` and `appspec.yaml`. By default, the AppSpec file is called `appspec.yml`.

-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://amzn-s3-demo-bucket/bundle.tgz --type tgz
```

```
codedeploy-local --bundle-location s3://amzn-s3-demo-bucket/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/awslabs/aws-codedeploy-sample-
tomcat --type zip
```

```
codedeploy-local --bundle-location https://api.github.com/repos/awslabs/aws-codedeploy-
sample-tomcat/zipball/master --type zip
```

```
codedeploy-local --bundle-location https://api.github.com/repos/awslabs/aws-codedeploy-
sample-tomcat/zipball/HEAD --type zip
```

```
codedeploy-local --bundle-location https://api.github.com/repos/awslabs/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 CodeDeploy

Monitoring is an important part of maintaining the reliability, availability, and performance of 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 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 CodeDeploy performance in your environment, by measuring performance at various times and under different load conditions. As you monitor 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 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 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 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 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.

  For information about updating your service role to work with CloudWatch alarm monitoring, see Grant CloudWatch permissions to a CodeDeploy service role. For information about adding CloudWatch alarm monitoring to your CodeDeploy operations, see Create an application with CodeDeploy, Create a deployment group with CodeDeploy, or Change deployment group settings with CodeDeploy.

- **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 CodeDeploy logs, see View 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 CodeDeploy operations, see Monitoring deployments with Amazon CloudWatch Events.

- **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 CodeDeploy, see Monitoring Deployments.

- **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 CodeDeploy, see Monitoring Deployments with Amazon SNS Event Notifications.

Manual monitoring tools

Another important part of monitoring CodeDeploy involves manually monitoring those items that the CloudWatch alarms don't cover. The 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 CodeDeploy deployments.

- 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
- Monitoring Deployments
- Monitoring Deployments with Amazon SNS Event Notifications

Monitoring deployments with Amazon CloudWatch tools

You can monitor CodeDeploy deployments using the following CloudWatch tools: Amazon CloudWatch Events, CloudWatch alarms, and Amazon CloudWatch Logs.

Reviewing the logs created by the CodeDeploy agent and deployments can help you troubleshoot the causes of deployment failures. As an alternative to reviewing 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 CloudWatch alarms and CloudWatch Events to monitor your CodeDeploy deployments, see the following topics.

Topics

- Monitoring deployments with CloudWatch alarms in CodeDeploy
- Monitoring deployments with Amazon CloudWatch Events

Monitoring deployments with CloudWatch alarms in CodeDeploy

You can create a CloudWatch alarm for an instance or Amazon EC2 Auto Scaling group you are using in your 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 invoke actions when their state changes (for example, from `OK` to `ALARM`).

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 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 a 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 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 CodeDeploy, see Create a deployment group with CodeDeploy and Change deployment group settings with CodeDeploy.

Topics

- Grant CloudWatch permissions to a CodeDeploy service role

Grant CloudWatch permissions to a CodeDeploy service role

Before you can use CloudWatch alarm monitoring with your deployments, the service role you use in your 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:

   JSON

```
{
    "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 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 CodeDeploy operations:

- AWS Lambda functions
- Kinesis streams
- Amazon SQS queues
- Built-in targets (`EC2 CreateSnapshot API call`, `EC2 RebootInstances API call`, `EC2 StopInstances API call` , and `EC2 TerminateInstances API call`)
- 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 CodeDeploy. Before you create event rules for use in your 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 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
STOP	READY
QUEUED	SUCCESS

Deployment status values	Instance status values
READY SUCCESS	

6. Specify which 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.
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 to rule now.
13. If you're satisfied with the rule, choose **Create rule**.

Monitoring deployments with AWS CloudTrail

CodeDeploy is integrated with CloudTrail, a service that captures API calls made by or on behalf of CodeDeploy in your AWS account and delivers the log files to an Amazon S3 bucket you specify. CloudTrail captures API calls from the CodeDeploy console, from CodeDeploy commands through the AWS CLI, or from the CodeDeploy APIs directly. Using the information collected by CloudTrail, you can determine which request was made to 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.

CodeDeploy information in CloudTrail

When CloudTrail logging is enabled in your AWS account, API calls made to CodeDeploy actions are tracked in log files. 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 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 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 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 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 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 a 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 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 CodeDeploy triggers.

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 CodeDeploy deployment group, for any of the following event types.

Deployment events	Instance events
• Success • Failure	• Success • Failure

Deployment events	Instance events
• Started • Stopped • Rollback • Ready[1] • All deployment events	• Started • Ready[1] • 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 CodeDeploy.

Topics

- Grant Amazon SNS permissions to a CodeDeploy service role
- Create a trigger for a CodeDeploy event
- Edit a trigger in a CodeDeploy deployment group
- Delete a trigger from a CodeDeploy deployment group
- JSON data formats for CodeDeploy triggers

Grant Amazon SNS permissions to a CodeDeploy service role

Before your triggers can generate notifications, the service role you use in your 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, enter a name to identify this policy, such as `SNSPublish`.
7. Paste the following into the **Policy Document** field:

   JSON

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

8. Choose **Apply Policy**.

Create a trigger for a CodeDeploy event

You can create a trigger that publishes an Amazon Simple Notification Service (Amazon SNS) topic for a AWS CodeDeploy deployment or instance event. Then, when that event occurs, all subscribers to the associated topic 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 for the trigger to point to. For information, see Create a topic. When you create a topic, we recommend you give it a name that identifies 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 a CodeDeploy service role before notifications can be sent for your trigger. For information, see Grant Amazon SNS permissions to a CodeDeploy service role.

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 CodeDeploy events (console)

You can use the CodeDeploy console to create triggers for a 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 a CodeDeploy event

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

   > **Note**
   >
   > Sign in with the same user that you set up in Getting started with CodeDeploy.

3. In the navigation pane, expand **Deploy**, then choose **Applications**.
4. On the **Applications** page, choose the name of the application associated with the deployment group where you want to add a trigger.
5. On the **Application details** page, choose the deployment group where you want to add a trigger.
6. Choose **Edit**.
7. Expand **Advanced - optional**.
8. In the **Triggers** area, choose **Create trigger**.
9. In **Create deployment trigger** pane, do the following:

   a. In **Trigger name**, enter 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`.

   b. In **Events**, choose the event type or types to trigger the Amazon SNS topic to send notifications.

   c. In **Amazon SNS topics**, choose the name of topic you created for sending notifications for this trigger.

d. Choose **Create trigger**. CodeDeploy sends a test notification to confirm you have correctly configured access between 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 receive confirmation in an SMS message or email message.

10. Choose **Save changes**.

Create a trigger to send notifications for 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 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 CodeDeploy in your account. For information, see Step 2: Create a service role for CodeDeploy.

   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-2:444455556666: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 prompts 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-2:444455556666: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 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 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 prompts 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-2:444455556666: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 receive a test notification message that indicates both permissions and trigger details are set up correctly.

Edit a trigger in a CodeDeploy deployment group

If your notification requirements change, you can modify your trigger rather than create a new one.

Modify a CodeDeploy trigger (CLI)

To use the AWS CLI to change trigger details for 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 receive a test notification message that indicates both permissions and trigger details are set up correctly.

Delete a trigger from a 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 CodeDeploy console at https://console.aws.amazon.com/codedeploy.

> **Note**
>
> Sign in with the same user that you set up in Getting started with CodeDeploy.

2. In the navigation pane, expand **Deploy**, then choose **Applications**.
3. On the **Applications** page, choose the name of the application associated with the deployment group where you want to delete a trigger.

4. On the **Application details** page, choose the deployment group where you want to delete a trigger.
5. Choose **Edit**.
6. Expand **Advanced - optional**.
7. In the **Triggers** area, choose the trigger you want to delete, then choose **Delete trigger**.
8. Choose **Save changes**.

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 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 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": "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"
    }
}
```

Security in AWS CodeDeploy

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS compliance programs. To learn about the compliance programs that apply to AWS CodeDeploy, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors, including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using CodeDeploy. The following topics show you how to configure CodeDeploy to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your CodeDeploy resources.

Topics

- Data protection in AWS CodeDeploy
- Identity and access management for AWS CodeDeploy
- Logging and monitoring in CodeDeploy
- Compliance validation for AWS CodeDeploy
- Resilience in AWS CodeDeploy
- Infrastructure security in AWS CodeDeploy

Data protection in AWS CodeDeploy

The AWS shared responsibility model applies to data protection in AWS CodeDeploy. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the

AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with CodeDeploy or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Internetwork traffic privacy

CodeDeploy is a fully managed deployment service that supports EC2 instances, Lambda functions, Amazon ECS, and on-premises servers. For EC2 instances and on-premises servers, a host-based agent communicates with CodeDeploy using TLS.

Currently, communication from the agent to the service requires an outbound internet connection so the agent can communicate with the public CodeDeploy and Amazon S3 service endpoints. In a virtual private cloud, this can be accomplished with an internet gateway, site to site VPN connection to your corporate network, or direct connect.

The CodeDeploy agent supports HTTP proxies.

Amazon VPC endpoints, powered by AWS PrivateLink, are available for CodeDeploy in certain regions. For details, see Use CodeDeploy with Amazon Virtual Private Cloud.

> **Note**
>
> The CodeDeploy agent is required only if you deploy to an Amazon EC2/On-premises compute platform. The agent is not required for deployments that use the Amazon ECS or AWS Lambda compute platform.

Encryption at rest

Customer code is not stored in CodeDeploy. As a deployment service, CodeDeploy is dispatching commands to the CodeDeploy agent running on EC2 instances or on-premises servers. The CodeDeploy agent then executes the commands using TLS. Service model data for deployments, deployment configuration, deployment groups, applications, and application revisions are stored in Amazon DynamoDB and encrypted at rest using an AWS owned key, owned and managed by CodeDeploy. For more information, see AWS owned keys.

Encryption in transit

The CodeDeploy agent initiates all communication with CodeDeploy over port 443. The agent polls CodeDeploy and listens for a command. The CodeDeploy agent is open source. All service-to-service and client-to-service communication is encrypted in transit using TLS. This protects customer data in transit between CodeDeploy and other services like Amazon S3.

Encryption key management

There are no encryption keys for you to manage. The CodeDeploy service model data is encrypted using an AWS owned key, owned and managed by CodeDeploy. For more information, see AWS owned keys.

Identity and access management for AWS CodeDeploy

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use CodeDeploy resources. IAM is an AWS service that you can use with no additional charge.

Topics

- Audience
- Authenticating with identities
- Managing access using policies
- How AWS CodeDeploy works with IAM
- AWS managed (predefined) policies for CodeDeploy
- CodeDeploy updates to AWS managed policies
- AWS CodeDeploy identity-based policy examples
- Troubleshooting AWS CodeDeploy identity and access
- CodeDeploy permissions reference
- Cross-service confused deputy prevention

Audience

How you use AWS Identity and Access Management (IAM) differs based on your role:

- **Service user** - request permissions from your administrator if you cannot access features (see Troubleshooting AWS CodeDeploy identity and access)
- **Service administrator** - determine user access and submit permission requests (see How AWS CodeDeploy works with IAM)
- **IAM administrator** - write policies to manage access (see AWS CodeDeploy identity-based policy examples)

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

Users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access AWS using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an AWS CLI or AWS API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For

more information about JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see Choose between managed policies and inline policies in the *IAM User Guide*.

Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How AWS CodeDeploy works with IAM

Before you use IAM to manage access to CodeDeploy, you should understand which IAM features are available to use with CodeDeploy. For more information, see AWS services that work with IAM in the *IAM User Guide*.

Topics

- CodeDeploy identity-based policies
- CodeDeploy resource-based policies
- Authorization based on CodeDeploy tags
- CodeDeploy IAM roles

CodeDeploy identity-based policies

With IAM identity-based policies, you can specify allowed or denied actions and resources and the conditions under which actions are allowed or denied. CodeDeploy supports actions, resources, and condition keys. For information about the elements that you use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Actions

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Include actions in a policy to grant permissions to perform the associated operation.

Policy actions in CodeDeploy use the `codedeploy:` prefix before the action. For example, the `codedeploy:GetApplication` permission grants the user permissions to perform the `GetApplication` operation. Policy statements must include either an `Action` or `NotAction` element. CodeDeploy defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows:

```
"Action": [
      "codedeploy:action1",
      "codedeploy:action2"
```

You can specify multiple actions using wildcards (*). For example, include the following action to specify all actions that begin with the word `Describe`:

```
"Action": "ec2:Describe*"
```

For a list of CodeDeploy actions, see Actions Defined by AWS CodeDeploy in the *IAM User Guide*.

For a table that lists all of the CodeDeploy API actions and the resources they apply to, see CodeDeploy permissions reference.

Resources

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. As a best practice, specify a resource using its Amazon Resource Name (ARN). For actions that don't support resource-level permissions, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

For example, you can indicate a deployment group (*`myDeploymentGroup`*) in your statement using its ARN as follows:

```
"Resource": "arn:aws:codedeploy:us-
west-2:123456789012:deploymentgroup:myApplication/myDeploymentGroup"
```

You can also specify all deployment groups that belong to an account by using the wildcard character (*) as follows:

```
"Resource": "arn:aws:codedeploy:us-west-2:123456789012:deploymentgroup:*"
```

To specify all resources, or if an API action does not support ARNs, use the wildcard character (*) in the `Resource` element as follows:

```
"Resource": "*"
```

Some 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"]
```

CodeDeploy provides a set of operations to work with the CodeDeploy resources. For a list of available operations, see CodeDeploy permissions reference.

For a list of CodeDeploy resource types and their ARNs, see Resources Defined by AWS CodeDeploy in the *IAM User Guide*. For information about the actions in which you can specify the ARN of each resource, see Actions Defined by AWS CodeDeploy.

CodeDeploy resources and operations

In 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. 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 ARNs associated with them. For more information, see Amazon resource names (ARNs) in the *Amazon Web Services General Reference*.

Resource type	ARN format
Deployment group	`arn:aws:codedeploy:` *`region`*`:`*`account-id`*` :deploymentgroup:` *`application-name`*` /`*`deployment-group-name`*
Application	`arn:aws:codedeploy:` *`region`*`:`*`account-id`*` :application:` *`application-name`*
Deployment configuration	`arn:aws:codedeploy:` *`region`*`:`*`account-id`*` :deploymentconfig:` *`deployment-configuration-name`*

Resource type	ARN format
Instance	`arn:aws:codedeploy:` *`region`*`:`*`account-id`*` :instance / `*`instance-ID`*
All CodeDeploy resources	`arn:aws:codedeploy:*`
All CodeDeploy resources owned by the specified account in the specified Region	`arn:aws:codedeploy:` *`region`*`:`*`account-id`*` :*`

> **Note**
>
> Most services in AWS treat a colon (:) or a forward slash (/) as the same character in ARNs. However, CodeDeploy uses an exact match in resource patterns and rules. Be sure to use the correct ARN characters when you create event patterns so that they match the ARN syntax in the resource.

Condition keys

CodeDeploy does not provide any service-specific condition keys, but it does support the use of some global condition keys. For more information, see AWS global condition context keys in the *IAM User Guide*.

Examples

To view examples of CodeDeploy identity-based policies, see AWS CodeDeploy identity-based policy examples.

CodeDeploy resource-based policies

CodeDeploy does not support resource-based policies. To view an example of a detailed resource-based policy page, see Using resource-based policies for AWS Lambda.

Authorization based on CodeDeploy tags

CodeDeploy does not support tagging resources or controlling access based on tags.

CodeDeploy IAM roles

An IAM role is an entity in your AWS account that has specific permissions.

Using temporary credentials with CodeDeploy

You can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as AssumeRole or GetFederationToken.

CodeDeploy supports the use of temporary credentials.

Service-linked roles

CodeDeploy does not support service-linked roles.

Service roles

This feature allows a service to assume a service role on your behalf. This role allows the service to access resources in other services to complete an action on your behalf. Service roles appear in your AWS account and are owned by the account. This means that a user can change the permissions for this role. However, doing so might break the functionality of the service.

CodeDeploy supports service roles.

Choosing an IAM role in CodeDeploy

When you create a deployment group resource in CodeDeploy, you must choose a role to allow CodeDeploy to access Amazon EC2 on your behalf. If you have previously created a service role or service-linked role, CodeDeploy provides you with a list of roles to choose from. It's important to choose a role that allows access to start and stop EC2 instances.

AWS managed (predefined) policies for CodeDeploy

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. These AWS-managed policies grant permissions for common use cases so

you can avoid having to investigate which permissions are required. For more information, see AWS managed policies in the *IAM User Guide*.

Topics

- List of AWS managed policies for CodeDeploy
- CodeDeploy managed policies and notifications

List of AWS managed policies for CodeDeploy

The following AWS managed policies, which you can attach to users in your account, are specific to CodeDeploy:

- `AWSCodeDeployFullAccess`: Grants full access to 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 CodeDeploy.

- `AWSCodeDeployDeployerAccess`: Grants permission to register and deploy revisions.

- `AWSCodeDeployReadOnlyAccess`: Grants read-only access to CodeDeploy.

- `AWSCodeDeployRole`: Allows CodeDeploy to:
  - read the tags on your instances or identify your Amazon EC2 instances by Amazon EC2 Auto Scaling group names
  - read, create, update, and delete Amazon EC2 Auto Scaling groups, lifecycle hooks, scaling policies, and warm pool features
  - publish information to Amazon SNS topics
  - retrieve information about Amazon CloudWatch alarms
  - read and update resources in the Elastic Load Balancing service

  The policy contains the following code:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CompleteLifecycleAction",
        "autoscaling:DeleteLifecycleHook",
        "autoscaling:DescribeAutoScalingGroups",
        "autoscaling:DescribeLifecycleHooks",
        "autoscaling:PutLifecycleHook",
        "autoscaling:RecordLifecycleActionHeartbeat",
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:CreateOrUpdateTags",
        "autoscaling:UpdateAutoScalingGroup",
        "autoscaling:EnableMetricsCollection",
        "autoscaling:DescribePolicies",
        "autoscaling:DescribeScheduledActions",
        "autoscaling:DescribeNotificationConfigurations",
        "autoscaling:SuspendProcesses",
        "autoscaling:ResumeProcesses",
        "autoscaling:AttachLoadBalancers",
        "autoscaling:AttachLoadBalancerTargetGroups",
        "autoscaling:PutScalingPolicy",
        "autoscaling:PutScheduledUpdateGroupAction",
        "autoscaling:PutNotificationConfiguration",
        "autoscaling:DescribeScalingActivities",
        "autoscaling:DeleteAutoScalingGroup",
        "autoscaling:PutWarmPool",
        "ec2:DescribeInstances",
        "ec2:DescribeInstanceStatus",
        "ec2:TerminateInstances",
        "tag:GetResources",
        "sns:Publish",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:PutMetricAlarm",
        "elasticloadbalancing:DescribeLoadBalancers",
        "elasticloadbalancing:DescribeLoadBalancerAttributes",
        "elasticloadbalancing:DescribeInstanceHealth",
        "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
        "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
```

```
        "elasticloadbalancing:DescribeTargetGroups",
        "elasticloadbalancing:DescribeTargetGroupAttributes",
        "elasticloadbalancing:DescribeTargetHealth",
        "elasticloadbalancing:RegisterTargets",
        "elasticloadbalancing:DeregisterTargets"
      ],
      "Resource": "*"
    }
  ]
}
```

- `AWSCodeDeployRoleForLambda`: Grants CodeDeploy permission to access AWS Lambda and any other resource required for a deployment.

- `AWSCodeDeployRoleForECS`: Grants CodeDeploy permission to access Amazon ECS and any other resource required for a deployment.

- `AWSCodeDeployRoleForECSLimited`: Grants CodeDeploy permission to access Amazon ECS and any other resource required for a deployment with the following exceptions:
  - In the `hooks` section of the AppSpec file, only Lambda functions with names that begin with `CodeDeployHook_` can be used. For more information, see AppSpec 'hooks' section for an Amazon ECS deployment.
  - S3 bucket access is limited to S3 buckets with a registration tag, `UseWithCodeDeploy`, that has a value of `true`. For more information, see Object tagging.
- `AmazonEC2RoleforAWSCodeDeployLimited`: Grants CodeDeploy permission to get and list objects in a CodeDeploy Amazon S3 bucket. The policy contains the following code:

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
```

```
                "s3:GetObjectVersion",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::*/CodeDeploy/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
                }
            }
        }
    ]
}
```

Permissions for some aspects of the deployment process are granted to two other role types that act on behalf of CodeDeploy:

- An *IAM instance profile* is 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 are stored. For more information, see Step 4: Create an IAM instance profile for your Amazon EC2 instances.
- A *service role* is 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 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 CodeDeploy to identify instances to which it can deploy applications.
  - To perform operations on instances, Amazon EC2 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 to set up alarm monitoring for deployments.

For more information, see Step 2: Create a service role for CodeDeploy.

You can also create custom IAM policies to grant permissions for CodeDeploy actions and resources. You attach these custom policies to IAM roles, and then you assign the roles to users or groups who require the permissions.

CodeDeploy managed policies and notifications

CodeDeploy supports notifications to make users aware of important changes to deployments. Managed policies for CodeDeploy include policy statements for notification functionality. For more information, see What are notifications?.

Permissions for notifications in full access managed policies

The `AWSCodeDeployFullAccess` managed policy includes the following statements to allow full access to notifications. Users with this managed policy applied can also create and manage Amazon SNS topics for notifications, subscribe and unsubscribe users to topics, and list topics to choose as targets for notification rules.

```
    {
        "Sid": "CodeStarNotificationsReadWriteAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:CreateNotificationRule",
            "codestar-notifications:DescribeNotificationRule",
            "codestar-notifications:UpdateNotificationRule",
            "codestar-notifications:DeleteNotificationRule",
            "codestar-notifications:Subscribe",
            "codestar-notifications:Unsubscribe"
        ],
        "Resource": "*",
        "Condition" : {
            "ArnLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codedeploy:*:*:application:*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
```

```
        "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListTargets",
            "codestar-notifications:ListTagsforResource"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
        "Effect": "Allow",
        "Action": [
            "sns:CreateTopic",
            "sns:SetTopicAttributes"
        ],
        "Resource": "arn:aws:sns:*:*:codestar-notifications*"
    },
    {
      "Sid" : "CodeStarNotificationsChatbotAccess",
      "Effect" : "Allow",
      "Action" : [
        "chatbot:DescribeSlackChannelConfigurations"
      ],
      "Resource" : "*"
    },
    {
        "Sid": "SNSTopicListAccess",
        "Effect": "Allow",
        "Action": [
            "sns:ListTopics"
        ],
        "Resource": "*"
    }
```

Permissions for notifications in read-only managed policies

The `AWSCodeDeployReadOnlyAccess` managed policy includes the following statements to allow read-only access to notifications. Users with this managed policy applied can view notifications for resources, but cannot create, manage, or subscribe to them.

```
    {
        "Sid": "CodeStarNotificationsPowerUserAccess",
        "Effect": "Allow",
        "Action": [
```

```
            "codestar-notifications:DescribeNotificationRule"
        ],
        "Resource": "*",
        "Condition" : {
            "ArnLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codedeploy:*:*:application:*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules"
        ],
        "Resource": "*"
    }
```

Permissions for notifications in other managed policies

The `AWSCodeDeployDeployerAccess` managed policy includes the following statements to allow users to create, update, subscribe, and view notifications for resources, but cannot delete them. Users with this managed policy applied can also create and manage Amazon SNS topics for notifications.

This policy includes permissions to do the following:

- `codestar-notifications:CreateNotificationRule` – Allows principals to create notifications.
- `codestar-notifications:DescribeNotificationRule` – Allows principals to retrieve information about notifications.
- `codestar-notifications:UpdateNotificationRule` – Allows principals to update notifications.
- `codestar-notifications:Subscribe` – Allows principals to subscribe to notification updates.
- `codestar-notifications:Unsubscribe` – Allows principals to unsubscribe to notification updates.
- `codestar-notifications:ListNotificationRules` – Allows principals to retrieve the list of notification rules.
- `codestar-notifications:ListTargets` – Allows principals to retrieve the list of targets.

- `codestar-notifications:ListTagsforResource` – Allows principals to retrieve the list of tags.
- `codestar-notifications:ListEventTypes` – Allows principals to retrieve the list of event types.
- `chatbot:DescribeSlackChannelConfiguration` – Allows principals to retrieve information about Slack channel configurations.
- `sns:ListTopics` – Allows principals to retrieve the list of Amazon SNS topics for notifications.

```
  {
     "Sid" : "CodeStarNotificationsReadWriteAccess",
     "Effect" : "Allow",
     "Action" : [
       "codestar-notifications:CreateNotificationRule",
       "codestar-notifications:DescribeNotificationRule",
       "codestar-notifications:UpdateNotificationRule",
       "codestar-notifications:Subscribe",
       "codestar-notifications:Unsubscribe"
     ],
     "Resource" : "*",
     "Condition" : {
       "ArnLike" : {
         "codestar-notifications:NotificationsForResource" :
"arn:aws:codedeploy:*:*:application:*"
       }
     }
   },
   {
     "Sid" : "CodeStarNotificationsListAccess",
     "Effect" : "Allow",
     "Action" : [
       "codestar-notifications:ListNotificationRules",
       "codestar-notifications:ListTargets",
       "codestar-notifications:ListTagsforResource",
       "codestar-notifications:ListEventTypes"
     ],
     "Resource" : "*"
   },
   {
     "Sid" : "CodeStarNotificationsChatbotAccess",
     "Effect" : "Allow",
     "Action" : [
```

```
        "chatbot:DescribeSlackChannelConfigurations"
      ],
      "Resource" : "*"
    },
    {
      "Sid" : "SNSTopicListAccess",
      "Effect" : "Allow",
      "Action" : [
        "sns:ListTopics"
      ],
      "Resource" : "*"
    }
```

For more information, see Identity and access management for AWS CodeStar Notifications.

CodeDeploy updates to AWS managed policies

View details about updates to AWS managed policies for CodeDeploy since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the CodeDeploy Document history.

Change	Description	Date
`AWSCodeDeployDeployerAccess` managed policy – Updates to existing policy	Updated the `codestar-notifications:NotificationsForResource` action to support IAM policy validation changes. The original resource `arn:aws:codedeploy:*` has been updated to `arn:aws:codedeploy:*:*:application:*`. For more information on this policy, see Permissions for notifications in other managed policies.	December 16, 2024

Change	Description	Date
`AWSCodeDeployReadOnlyAccess` managed policy – Updates to existing policy	Updated the `codestar-notifications:NotificationsForResource` action to support IAM policy validation changes. The original resource `arn:aws:codedeploy:*` has been updated to `arn:aws:codedeploy:*:*:application:*`. For more information on this policy, see Permissions for notifications in read-only managed policies.	December 16, 2024
`AWSCodeDeployFullAccess` managed policy – Updates to existing policy	Updated the `codestar-notifications:NotificationsForResource` action to support IAM policy validation changes. The original resource `arn:aws:codedeploy:*` has been updated to `arn:aws:codedeploy:*:*:application:*`. For more information on this policy, see Permissions for notifications in full access managed policies.	December 16, 2024

Change	Description	Date
`AWSCodeDeployRole` managed policy – Updates to existing policy	Added the `elasticloadbalancing:DescribeLoadBalancerAttributes` and `elasticloadbalancing:DescribeTargetGroupAttributes` actions to the policy statement to support Elastic Load Balancing changes. For more information on this policy, see AWSCodeDeployRole.	August 16, 2023
`AWSCodeDeployFullAccess` managed policy – Updates to existing policy	Added the `chatbot:ListMicrosoftTeamsChannelConfigurations` action to the policy statement to support notification changes. For more information on this policy, see AWSCodeDeployRole.	May 11, 2023
`AWSCodeDeployRole` managed policy – Updates to existing policy	Added the `autoscaling:CreateOrUpdateTags` action to the policy statement to support Amazon EC2 Auto Scaling authorization changes. For more information on this policy, see AWSCodeDeployRole.	February 3, 2023

Change	Description	Date
`AmazonEC2RoleforAWSCodeDeployLimited` managed policy – Updates to existing policy	Removed the `s3:ListBucket` action from the policy statement that includes the `s3:ExistingObjectTag/UseWithCodeDeploy` condition. For more information on this policy, see AmazonEC2RoleforAWSCodeDeployLimited.	November 22, 2021
`AWSCodeDeployRole` managed policy – Updates to existing policy	Added the `autoscaling:PutWarmPool` action to support adding warm pools to Amazon EC2 Auto Scaling groups for blue/green deployments. Removed needless duplicate actions.	May 18, 2021
CodeDeploy started tracking changes	CodeDeploy started tracking changes for its AWS managed policies.	May 18, 2021

AWS CodeDeploy identity-based policy examples

By default, users don't have permission to create or modify CodeDeploy resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. You must create IAM policies that grant IAM roles permission to perform API operations on the specified resources they need. You must then attach those IAM roles to users or groups who require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating policies on the JSON tab in the *IAM User Guide*.

In CodeDeploy, identity-based policies are used to manage permissions to the various resources related to the deployment process. You can control access to the following resource types:

- Applications and application revisions.
- Deployments.
- Deployment configurations.
- Instances and on-premises instances.

The capabilities controlled by resource policies vary depending on the resource type, as outlined in the following table:

Resource types	Capabilities
All	View and list details about resources
Applications Deployment configurations Deployment groups	Create resources Delete resources
Deployments	Create deployments Stop deployments
Application revisions	Register application revisions
Applications Deployment groups	Update resources
On-premises instances	Add tags to instances Remove tags from instances Register instances Deregister instances

The following example shows 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.

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:DeleteDeploymentGroup"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:111122223333:deploymentgroup:WordPress_App/
WordPress_DepGroup"
      ]
    }
  ]
}
```

Topics

- Customer-managed policy examples
- Policy best practices
- Using the CodeDeploy console
- Allow users to view their own permissions

Customer-managed policy examples

In this section, you can find example policies that grant permissions for various CodeDeploy actions. These policies work when you are using the CodeDeploy API, AWS SDKs, or the AWS CLI. You must grant additional permissions for actions you perform in the console. To learn more about granting console permissions, see Using the CodeDeploy console .

> **Note**
>
> All examples use the US West (Oregon) Region (`us-west-2`) and contain fictitious account IDs.

Examples

- Example 1: Allow permission to perform CodeDeploy operations in a single Region
- Example 2: Allow permission to register revisions for a single application
- Example 3: Allow permission to create deployments for a single deployment group

Example 1: Allow permission to perform CodeDeploy operations in a single Region

The following example grants permissions to perform CodeDeploy operations in the **`us-west-2`** Region only:

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:*"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:111122223333:*"
      ]
    }
  ]
}
```

Example 2: Allow permission 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:

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:RegisterApplicationRevision"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:111122223333:application:Test*"
      ]
    }
  ]
}
```

Example 3: Allow permission to create deployments for a single deployment group

The following example allows permission 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 in the **`us-west-2`** Region.

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:CreateDeployment"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:111122223333:deploymentgroup:WordPress_App/
WordPress_DepGroup"
      ]
    },
```

```
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetDeploymentConfig"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-
west-2:111122223333:deploymentconfig:ThreeQuartersHealthy"
      ]
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetApplicationRevision"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:111122223333:application:WordPress_App"
      ]
    }
  ]
}
```

Policy best practices

Identity-based policies determine whether someone can create, access, or delete CodeDeploy resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.

- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the CodeDeploy console

If you use the CodeDeploy console, you must have a minimum set of permissions that allows you to describe other AWS resources for your AWS account. To use CodeDeploy in the 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 who have a role with that IAM policy. To ensure that those users can still use the CodeDeploy console, also attach the `AWSCodeDeployReadOnlyAccess` managed policy to the role assigned to the user, as described in AWS managed (predefined) policies for CodeDeploy.

You don't need to allow minimum console permissions for users who are making calls only to the AWS CLI or the CodeDeploy API.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
```

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

Troubleshooting AWS CodeDeploy identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with CodeDeploy and IAM.

Topics

- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my CodeDeploy resources

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to CodeDeploy.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in CodeDeploy. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my CodeDeploy resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether CodeDeploy supports these features, see How AWS CodeDeploy works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

CodeDeploy permissions reference

Use the following table when you are setting up access and writing permissions policies that you can attach to an IAM identity (identity-based policies). The table lists each CodeDeploy API operation, the 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. 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 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 CodeDeploy actions and `codedeploy:Get*` specifies all 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-2:444455556666: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*.

> **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 CodeDeploy example ARNs.

CodeDeploy API operations and required permissions for actions

AddTagsToOnPremisesInstances

Action: `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: `codedeploy:BatchGetApplicationRevisions`

Required to get information about multiple application revisions associated with the user.

Resource: `arn:aws:codedeploy:region:account-id:application:application-name`

BatchGetApplications

Action: `codedeploy:BatchGetApplications`

Required to get information about multiple applications associated with the user.

Resource: `arn:aws:codedeploy:region:account-id:application:*`

BatchGetDeploymentGroups

Action: `codedeploy:BatchGetDeploymentGroups`

Required to get information about multiple deployment groups associated with the user.

Resource: `arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/deployment-group-name`

BatchGetDeploymentInstances

Action: `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: `codedeploy:BatchGetDeployments`

Required to get information about multiple deployments associated with the user.

Resource: `arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/deployment-group-name`

BatchGetOnPremisesInstances

Action: `codedeploy:BatchGetOnPremisesInstances`

Required to get information about one or more on-premises instances.

Resource: `arn:aws:codedeploy:region:account-id:*`

ContinueDeployment

Action: `codedeploy:ContinueDeployment`

Required during a blue/green deployment to start registering instances in a replacement environment with an Elastic Load Balancing load balancer.

Resource: `arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/deployment-group-name`

CreateApplication

Action: `codedeploy:CreateApplication`

Required to create an application associated with the user.

Resource: `arn:aws:codedeploy:region:account-id:application:application-name`

CreateDeployment [1]

Action: `codedeploy:CreateDeployment`

Required to create a deployment for an application associated with the user.

Resource: `arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/deployment-group-name`

CreateDeploymentConfig

Action: `codedeploy:CreateDeploymentConfig`

Required to create a custom deployment configuration associated with the user.

Resource: `arn:aws:codedeploy:region:account-id:deploymentconfig/deployment-configuration-name`

CreateDeploymentGroup

Action: `codedeploy:CreateDeploymentGroup`

Required to create a deployment group for an application associated with the user.

Resource: `arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/deployment-group-name`

DeleteApplication

Action: `codedeploy:DeleteApplication`

Required to delete an application associated with the user.

Resource: `arn:aws:codedeploy:region:account-id:application:application-name`

DeleteDeploymentConfig

Action: `codedeploy:DeleteDeploymentConfig`

Required to delete a custom deployment configuration associated with the user.

Resource: `arn:aws:codedeploy:region:account-id:deploymentconfig/deployment-configuration-name`

DeleteDeploymentGroup

Action: `codedeploy:DeleteDeploymentGroup`

Required to delete a deployment group for an application associated with the user.

Resource: `arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/deployment-group-name`

DeregisterOnPremisesInstance

Action: `codedeploy:DeregisterOnPremisesInstance`

Required to deregister an on-premises instance.

Resource: `arn:aws:codedeploy:region:account-id:instance/instance-ID`

GetApplication

Action: `codedeploy:GetApplication`

Required to get information about a single application associated with the user.

Resource: `arn:aws:codedeploy:region:account-id:application:application-name`

GetApplicationRevision

Action: `codedeploy:GetApplicationRevision`

Required to get information about a single application revision for an application associated with the user.

Resource: `arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:application:`*`application-name`*

GetDeployment

Action: `codedeploy:GetDeployment`

Required to get information about a single deployment to a deployment group for an application associated with the user.

Resource: `arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:deploymentgroup:`*`application-name`*`/`*`deployment-group-name`*

GetDeploymentConfig

Action: `codedeploy:GetDeploymentConfig`

Required to get information about a single deployment configuration associated with the user.

Resource: `arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:deploymentconfig/`*`deployment-configuration-name`*

GetDeploymentGroup

Action: `codedeploy:GetDeploymentGroup`

Required to get information about a single deployment group for an application associated with the user.

Resource: `arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:deploymentgroup:`*`application-name`*`/`*`deployment-group-name`*

GetDeploymentInstance

Action: `codedeploy:GetDeploymentInstance`

Required to get information about a single instance in a deployment associated with the user.

Resource: `arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/deployment-group-name`

GetOnPremisesInstance

Action: `codedeploy:GetOnPremisesInstance`

Required to get information about a single on-premises instance.

Resource: `arn:aws:codedeploy:region:account-id:instance/instance-ID`

ListApplicationRevisions

Action: `codedeploy:ListApplicationRevisions`

Required to get information about all application revisions for an application associated with the user.

Resource: `arn:aws:codedeploy:region:account-id:application:*`

ListApplications

Action: `codedeploy:ListApplications`

Required to get information about all applications associated with the user.

Resource: `arn:aws:codedeploy:region:account-id:application:*`

ListDeploymentConfigs

Action: `codedeploy:ListDeploymentConfigs`

Required to get information about all deployment configurations associated with the user.

Resource: `arn:aws:codedeploy:region:account-id:deploymentconfig/*`

ListDeploymentGroups

Action: `codedeploy:ListDeploymentGroups`

Required to get information about all deployment groups for an application associated with the user.

Resource: `arn:aws:codedeploy:region:account-id:deploymentgroup:application-name/*`

ListDeploymentInstances

Action: `codedeploy:ListDeploymentInstances`

Required to get information about all instances in a deployment associated with the user.

Resource: `arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:deploymentgroup:`*`application-name`*`/`*`deployment-group-name`*

ListDeployments

Action: `codedeploy:ListDeployments`

Required to get information about all deployments to a deployment group associated with the user, or to get all deployments associated with the user.

Resource: `arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:deploymentgroup:`*`application-name`*`/`*`deployment-group-name`*

ListGitHubAccountTokenNames

Action: `codedeploy:ListGitHubAccountTokenNames`

Required to get a list ofthe names of stored connections to GitHub accounts.

Resource: `arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:*`

ListOnPremisesInstances

Action: `codedeploy:ListOnPremisesInstances`

Required to get a list of one or more on-premises instance names.

Resource: `arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:*`

RegisterApplicationRevision

Action: `codedeploy:RegisterApplicationRevision`

Required to register information about an application revision for an application associated with the user.

Resource: `arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:application:`*`application-name`*

RegisterOnPremisesInstance

Action: `codedeploy:RegisterOnPremisesInstance`

Required to register an on-premises instance with CodeDeploy.

Resource: `arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:instance/`*`instance-ID`*

RemoveTagsFromOnPremisesInstances

Action: `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: `codedeploy:SkipWaitTimeForInstanceTermination`

Required to override a specified wait time and start terminating instances in the original environment immediately after the traffic is successfully routed in a blue/green deployment.

Resource: `arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:instance/`*`instance-ID`*

StopDeployment

Action: `codedeploy:StopDeployment`

Required to stop an in-progress deployment to a deployment group for an application associated with the user.

Resource: `arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:deploymentgroup:`*`application-name`*`/`*`deployment-group-name`*

UpdateApplication [3]

Action: `codedeploy:UpdateApplication`

Required to change information about an application associated with the user.

Resource: `arn:aws:codedeploy:`*`region`*`:`*`account-id`*`:application:`*`application-name`*

UpdateDeploymentGroup [3]

Action: `codedeploy:UpdateDeploymentGroup`

Required to change information about a single deployment group for an application associated with the 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 you provide a deployment group, but not when you list all of the deployments associated with the user.

[3] For `UpdateApplication`, you must have `UpdateApplication` permissions for both the old and 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.

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the calling service) calls another service (the called service). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the aws:SourceArn and aws:SourceAccount global condition context keys in resource policies to limit the permissions that CodeDeploy gives another service to the resource. If you use both global condition context keys and the `aws:SourceArn` value contains the account ID, the `aws:SourceAccount` value and the account in the `aws:SourceArn` value must use the same account ID when used in the same policy statement. Use `aws:SourceArn` if you want only

one resource to be associated with the cross-service access. Use `aws:SourceAccount` if you want any resource in that account to be associated with the cross-service use.

For EC2/On-Premises, AWS Lambda, and regular Amazon ECS deployments, the value of `aws:SourceArn` should include the CodeDeploy deployment group ARN with which CodeDeploy is allowed to assume the IAM role.

For Amazon ECS blue/green deployments created through CloudFormation, the value of `aws:SourceArn` should include the CloudFormation stack ARN with which CodeDeploy is allowed to assume the IAM role.

The most effective way to protect against the confused deputy problem is to use the `aws:SourceArn` key with the full ARN of the resource. If you don't know the full ARN or if you're specifying multiple resources, use wildcard characters (*) for the unknown portions.

For example, you might use the following trust policy with a EC2/On-Premises, AWS Lambda, or regular Amazon ECS deployment:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "codedeploy.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:codedeploy:us-
east-1:111122223333:deploymentgroup:myApplication/*"
                }
            }
        }
    ]
```

```
}
```

For an Amazon ECS blue/green deployment created through CloudFormation, you might use:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "codedeploy.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cloudformation:us-
east-1:111122223333:stack/MyCloudFormationStackName/*"
                }
            }
        }
    ]
}
```

Logging and monitoring in CodeDeploy

This section provides an overview of monitoring, logging, and incident response in CodeDeploy.

Auditing of all interactions with CodeDeploy

CodeDeploy is integrated with AWS CloudTrail, a service that captures API calls made by or on behalf of CodeDeploy in your AWS account and delivers the log files to an S3 bucket you specify. CloudTrail captures API calls from the CodeDeploy console, from CodeDeploy commands through

the AWS CLI, or from the CodeDeploy APIs directly. Using the information collected by CloudTrail, you can determine which request was made to 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, see Working with CloudTrail log files in the *AWS CloudTrail User Guide*.

You can view the log data created by a CodeDeploy deployment by setting up the Amazon CloudWatch agent to view aggregated data in the CloudWatch console or by signing in to an instance to review the log file. For more information, see Send CodeDeploy agent logs to CloudWatch.

Alerting and incident management

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 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. 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 you use CloudWatch Events as part of your CodeDeploy operations:

- AWS Lambda functions
- Kinesis streams
- Amazon SQS 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 EC2 instances when a deployment or instance event you specify occurs.

For more information, see What is Amazon CloudWatch Events in the *Amazon CloudWatch User Guide*.

Compliance validation for AWS CodeDeploy

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. For more information about your compliance responsibility when using AWS services, see AWS Security Documentation.

Resilience in AWS CodeDeploy

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS global infrastructure.

Infrastructure security in AWS CodeDeploy

As a managed service, AWS CodeDeploy is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of security processes whitepaper.

You use AWS published API calls to access CodeDeploy through the network. Clients must support Transport Layer Security (TLS) 1.2 or later. We recommend TLS 1.3 or later. Clients must also support cipher suites with perfect forward secrecy (PFS), such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Reference

Reference.

Topics

- CodeDeploy AppSpec file reference
- CodeDeploy agent configuration reference
- CloudFormation templates for CodeDeploy reference
- Use CodeDeploy with Amazon Virtual Private Cloud
- CodeDeploy resource kit reference
- CodeDeploy quotas

CodeDeploy AppSpec file reference

This section is a reference only. For a conceptual overview of the AppSpec file, see Application Specification Files.

The application specification file (AppSpec file) is a YAML-formatted or JSON-formatted file used by CodeDeploy to manage a deployment.

> **Note**
>
> The AppSpec file for an EC2/On-Premises deployment must be named `appspec.yml`, unless you are performing a local deployment. For more information, see Create a local deployment.

Topics

- AppSpec files on an Amazon ECS compute platform
- AppSpec files on an AWS Lambda compute platform
- AppSpec files on an EC2/on-premises compute platform
- AppSpec File structure
- AppSpec File example

- AppSpec File spacing
- Validate your AppSpec File and file location

AppSpec files on an Amazon ECS compute platform

For Amazon ECS compute platform applications, the AppSpec file is used by CodeDeploy to determine:

- Your Amazon ECS task definition file. This is specified with its ARN in the `TaskDefinition` instruction in the AppSpec file.
- The container and port in your replacement task set where your Application Load Balancer or Network Load Balancer reroutes traffic during a deployment. This is specified with the `LoadBalancerInfo` instruction in the AppSpec file.
- Optional information about your Amazon ECS service, such the platform version on which it runs, its subnets, and its security groups.
- Optional Lambda functions to run during hooks that correspond with lifecycle events during an Amazon ECS deployment. For more information, see AppSpec 'hooks' section for an Amazon ECS deployment.

AppSpec files on an AWS Lambda compute platform

For AWS Lambda compute platform applications, the AppSpec file is used by 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 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 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. It is used by 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.

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 CodeDeploy.

> **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 CodeDeploy as a *revision*), you upload it to an Amazon S3 bucket or Git repository. Then you use CodeDeploy to deploy the revision. For instructions, see Create a deployment with CodeDeploy.

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 and Plan a revision for CodeDeploy.

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 Amazon ECS deployments

> **Note**
>
> This AppSpec file is written in YAML, but you can use the same structure to write one in JSON. A string in a JSON-formatted AppSpec file is always wrapped in quotation marks ("").

```
version: 0.0
```

```
resources:
  ecs-service-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 CodeDeploy for future use.

Specify **version** with a string.

resources

This section specifies information about Amazon ECS application to deploy.

For more information, see AppSpec 'resources' section for Amazon ECS deployments.

hooks

This section specifies Lambda functions to run at specific deployment lifecycle event hooks to validate the deployment.

For more information, see List of lifecycle event hooks for an Amazon ECS deployment.

AppSpec file structure for AWS Lambda deployments

> **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 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 (Amazon ECS and AWS Lambda deployments only).

hooks

This section specifies Lambda functions to run at specific deployment lifecycle events to validate the deployment.

For more information, see AppSpec 'hooks' section.

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 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).

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).

hooks

This section specifies scripts to run at specific deployment lifecycle events during the deployment.

For more information, see AppSpec 'hooks' section.

Topics

- AppSpec 'files' section (EC2/On-Premises deployments only)
- AppSpec 'resources' section (Amazon ECS and AWS Lambda deployments only)
- AppSpec 'permissions' section (EC2/On-Premises deployments only)
- AppSpec 'hooks' section

AppSpec 'files' section (EC2/On-Premises deployments only)

Provides information to 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-1
    destination: destination-file-location-1
file_exists_behavior: DISALLOW|OVERWRITE|RETAIN
```

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 to the `appspec.yml` file, which should be at the root of your revision. For details on the file structure of a revision, see Plan a revision for CodeDeploy.

The `destination` instruction identifies the location on the instance where the files should be copied. This must be a fully qualified path such as `/root/destination/directory` (on Linux, RHEL, and Ubuntu) or `c:\destination\folder` (on Windows).

`source` and `destination` are each specified with a string.

The `file_exists_behavior` instruction is optional, and specifies how CodeDeploy handles files that already exist in a deployment target location but weren't part of the previous successful deployment. This setting can take any of the following values:

- DISALLOW: The deployment fails. This is also the default behavior if no option is specified.
- OVERWRITE: The version of the file from the application revision currently being deployed replaces the version already on the instance.

- RETAIN: The version of the file already on the instance is kept and used as part of the new deployment.

When using the `file_exists_behavior` setting, understand that this setting:

- can only be specified once, and applies to all files and directories listed under `files:`.
- takes precedence over the `--file-exists-behavior` AWS CLI option and the `fileExistsBehavior` API option (both of which are also optional).

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. If any of the files already exist on the instance,
 overwrite them.
#
```

```
files:
  - source: \
    destination: c:\temp
file_exists_behavior: OVERWRITE
#
# 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 (Amazon ECS and AWS Lambda deployments only)

The content in the `'resources'` section of the AppSpec file varies, depending on the compute platform of your deployment. The `'resources'` section for an Amazon ECS deployment contains your Amazon ECS task definition, container and port for routing traffic to your updated Amazon ECS task set, and other optional information. The `'resources'` section for an AWS Lambda deployment contains the name, alias, current version, and target version of a Lambda function.

Topics

- AppSpec 'resources' section for AWS Lambda deployments
- AppSpec 'resources' section for Amazon ECS deployments

AppSpec 'resources' section for AWS Lambda deployments

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. This value must be a valid positive integer.
- `targetversion` – Required. This is the version of the Lambda function traffic is shifted to. This value must be a valid positive integer.

AppSpec 'resources' section for Amazon ECS deployments

The `'resources'` section specifies the Amazon ECS service to deploy and has the following structure:

YAML:

```
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "task-definition-arn"
        LoadBalancerInfo:
          ContainerName: "ecs-container-name"
          ContainerPort: "ecs-application-port"
# Optional properties
```

```
        PlatformVersion: "ecs-service-platform-version"
        NetworkConfiguration:
          AwsvpcConfiguration:
            Subnets: ["ecs-subnet-1","ecs-subnet-n"]
            SecurityGroups: ["ecs-security-group-1","ecs-security-group-n"]
            AssignPublicIp: "ENABLED | DISABLED"
        CapacityProviderStrategy:
          - Base: integer
            CapacityProvider: "capacityProviderA"
            Weight: integer
          - Base: integer
            CapacityProvider: "capacityProviderB"
            Weight: integer
```

JSON:

```
"Resources": [
    {
        "TargetService": {
            "Type": "AWS::ECS::Service",
            "Properties": {
                "TaskDefinition": "task-definition-arn",
                "LoadBalancerInfo": {
                    "ContainerName": "ecs-container-name",
                    "ContainerPort": "ecs-application-port"
                },
                "PlatformVersion": "ecs-service-platform-version",
                "NetworkConfiguration": {
                    "AwsvpcConfiguration": {
                        "Subnets": [
                            "ecs-subnet-1",
                            "ecs-subnet-n"
                        ],
                        "SecurityGroups": [
                            "ecs-security-group-1",
                            "ecs-security-group-n"
                        ],
                        "AssignPublicIp": "ENABLED | DISABLED"
                    }
                },
                "CapacityProviderStrategy": [
                    {
                        "Base": integer,
```

```
                        "CapacityProvider": "capacityProviderA",
                        "Weight": integer
                    },
                    {
                        "Base": integer,
                        "CapacityProvider": "capacityProviderB",
                        "Weight": integer
                    }
                ]
            }
        }
    }
]
```

Each property is specified with a string except for `ContainerPort`, which is a number.

- `TaskDefinition` – Required. This is the task definition for the Amazon ECS service to deploy. It is specified with the ARN of the task definition. The ARN format is `arn:aws:ecs:`*`aws-region`*`:`*`account-id`*`:task-definition/`*`task-definition-family`*`:`*`task-definition-revision`*. For more information, see Amazon Resource Names (ARNs) and AWS service namespaces.

  > **Note**
  >
  > The `:`*`task-definition-revision`* portion of the ARN is optional. If it is omitted, Amazon ECS uses the latest ACTIVE revision of the task definition.

- `ContainerName` – Required. This is the name of the Amazon ECS container that contains your Amazon ECS application. It must be a container specified in your Amazon ECS task definition.
- `ContainerPort` – Required. This is the port on the container where traffic will be routed to.
- `PlatformVersion`: Optional. The platform version of the Fargate tasks in the deployed Amazon ECS service. For more information, see AWS Fargate platform versions. If not specified, `LATEST` is used by default.
- `NetworkConfiguration`: Optional. Under `AwsvpcConfiguration`, you can specify the following. For more information, see AwsVpcConfiguration in the *Amazon ECS Container Service API Reference*.
  - `Subnets`: Optional. A comma-separated list of one or more subnets in your Amazon ECS service.

- `SecurityGroups`: Optional. A comma-separated list of one or more security groups in your Amazon Elastic Container Service.
- `AssignPublicIp`: Optional. A string that specifies whether your Amazon ECS service's elastic network interface receives a public IP address. The valid values are `ENABLED` and `DISABLED`.

> **Note**
>
> All or none of the settings under `NetworkConfiguration` must be specified. For example, if you want to specify `Subnets`, you must also specify `SecurityGroups` and `AssignPublicIp`. If none is specified, CodeDeploy uses the current network Amazon ECS settings.

- `CapacityProviderStrategy`: Optional. A list of Amazon ECS capacity providers you want to use for your deployment. For more information, see Amazon ECS capacity providers in the *Amazon Elastic Container Service Developer Guide*. For each capacity provider, you can specify the following settings. For details on these settings, see AWS::ECS::ServiceCapacityProviderStrategyItem in the *AWS CloudFormation User Guide*
  - `Base`: Optional. The base value designates how many tasks, at a minimum, to run on the specified capacity provider. Only one capacity provider in a capacity provider strategy can have a base defined. If no value is specified, the default value of 0 is used.
  - `CapacityProvider`: Optional. The short name of the capacity provider. Example: *capacityProviderA*
  - `Weight`: Optional.

    The *weight* value designates the relative percentage of the total number of tasks launched that should use the specified capacity provider. The `weight` value is taken into consideration after the `base` value, if defined, is satisfied.

    If no `weight` value is specified, the default value of `0` is used. When multiple capacity providers are specified within a capacity provider strategy, at least one of the capacity providers must have a weight value greater than zero and any capacity providers with a weight of `0` will not be used to place tasks. If you specify multiple capacity providers in a strategy that all have a weight of `0`, any `RunTask` or `CreateService` actions using the capacity provider strategy will fail.

    An example scenario for using weights is defining a strategy that contains two capacity providers and both have a weight of `1`, then when the `base` is satisfied, the tasks will be split

evenly across the two capacity providers. Using that same logic, if you specify a weight of 1 for *capacityProviderA* and a weight of 4 for *capacityProviderB*, then for every one task that is run using *capacityProviderA*, four tasks would use *capacityProviderB*.

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 or Amazon ECS 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. A numeric value specifying the permissions to be applied to `object`. The mode setting follows the Linux chmod command syntax.

> **Important**
>
> If the value includes a leading zero, you must surround it with double-quotes, or remove the leading zero so that only three digits remain.

> **Note**
>
> Symbolic notation such as **u+x** is not supported for the `mode` setting.

Examples:

- `mode: "0644"` gives read and write permissions to the owner of the object (6), read-only permissions to the group (4), and read-only permissions to all other users (4).
- `mode: 644` grants the same permissions as `mode: "0644"`.

- `mode: 4755` sets the setuid attribute (4), gives full control permissions to the owner (7), gives read and execute permissions to the group (5), and gives read and execute permissions to all other users (5).

  For more examples, see the Linux chmod command documentation.

  If mode is not specified, all existing modes applied to the original file or folder structure remain unchanged after the copy operation.

- `acls` – Optional. A list of character strings representing one or more access control list (ACL) entries applied to `object`. For example, **`u:bob:rw`** represents read and write permissions for user **bob**. (For 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 seperate 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 in 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 or an Amazon ECS deployment specifies Lambda validation functions to run during a deployment lifecycle event. If an event hook is not present, 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 Amazon ECS deployment
- AppSpec 'hooks' section for an AWS Lambda deployment
- AppSpec 'hooks' section for an EC2/On-Premises deployment

AppSpec 'hooks' section for an Amazon ECS deployment

Topics

- List of lifecycle event hooks for an Amazon ECS deployment
- Run order of hooks in an Amazon ECS deployment.
- Structure of 'hooks' section
- Sample Lambda 'hooks' function

List of lifecycle event hooks for an Amazon ECS deployment

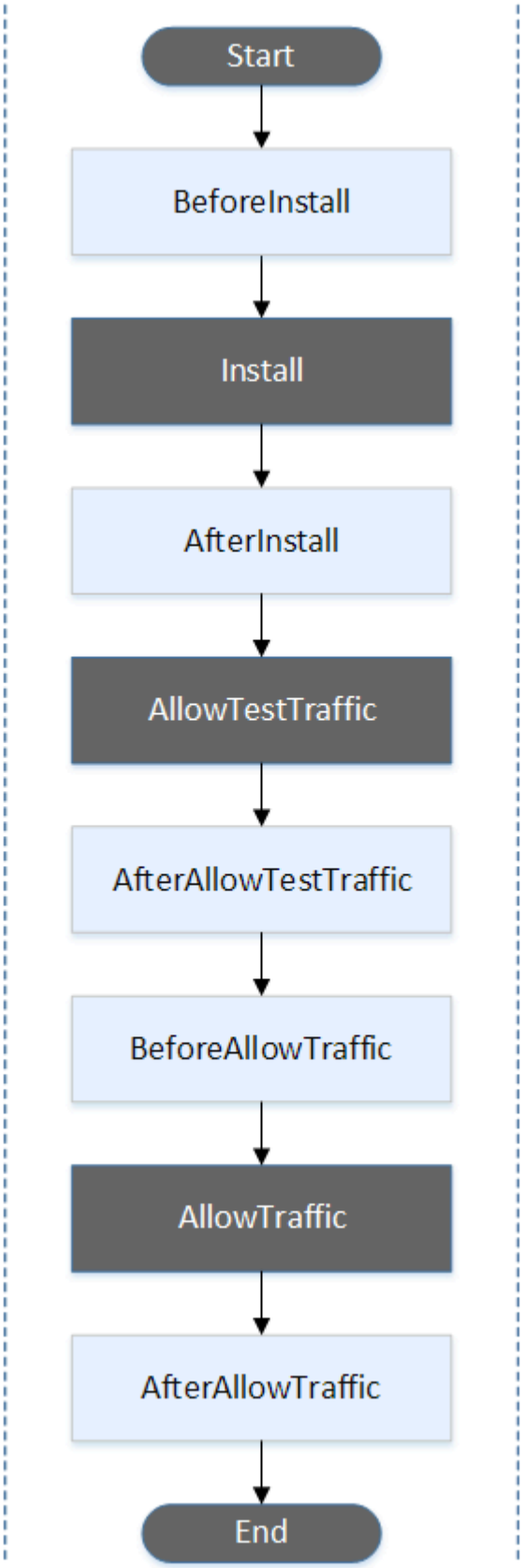
An AWS Lambda hook is one Lambda function specified with a string on a new line after the name of the lifecycle event. Each hook is executed once per deployment. Following are descriptions of the lifecycle events where you can run a hook during an Amazon ECS deployment.

- `BeforeInstall` – Use to run tasks before the replacement task set is created. One target group is associated with the original task set. If an optional test listener is specified, it is associated with the original task set. A rollback is not possible at this point.
- `AfterInstall` – Use to run tasks after the replacement task set is created and one of the target groups is associated with it. If an optional test listener is specified, it is associated with the original task set. The results of a hook function at this lifecycle event can trigger a rollback.
- `AfterAllowTestTraffic` – Use to run tasks after the test listener serves traffic to the replacement task set. The results of a hook function at this point can trigger a rollback.
- `BeforeAllowTraffic` – Use to run tasks after the second target group is associated with the replacement task set, but before traffic is shifted to the replacement task set. The results of a hook function at this lifecycle event can trigger a rollback.
- `AfterAllowTraffic` – Use to run tasks after the second target group serves traffic to the replacement task set. The results of a hook function at this lifecycle event can trigger a rollback.

For more information, see What happens during an Amazon ECS deployment and Tutorial: Deploy an Amazon ECS service with a validation test.

Run order of hooks in an Amazon ECS deployment.

In an Amazon ECS deployment, event hooks run in the following order:


Start
BeforeInstall
Install
AfterInstall
AllowTestTraffic
AfterAllowTestTraffic
BeforeAllowTraffic
AllowTraffic
AfterAllowTraffic
End

> **Note**
>
> The **Start**, **Install**, **TestTraffic**, **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:
  - BeforeInstall: "BeforeInstallHookFunctionName"
  - AfterInstall: "AfterInstallHookFunctionName"
  - AfterAllowTestTraffic: "AfterAllowTestTrafficHookFunctionName"
  - BeforeAllowTraffic: "BeforeAllowTrafficHookFunctionName"
  - AfterAllowTraffic: "AfterAllowTrafficHookFunctionName"
```

Using JSON:

```
"Hooks": [
  {
   "BeforeInstall": "BeforeInstallHookFunctionName"
  },
  {
   "AfterInstall": "AfterInstallHookFunctionName"
  },
  {
   "AfterAllowTestTraffic": "AfterAllowTestTrafficHookFunctionName"
  },
  {
   "BeforeAllowTraffic": "BeforeAllowTrafficHookFunctionName"
  },
  {
   "AfterAllowTraffic": "AfterAllowTrafficHookFunctionName"
  }
 ]
}
```

Sample Lambda 'hooks' function

Use the `'hooks'` section to specify a Lambda function that CodeDeploy can call to validate an Amazon ECS deployment. You can use the same function or a different one for the `BeforeInstall`, `AfterInstall`, `AfterAllowTestTraffic`, `BeforeAllowTraffic`, and `AfterAllowTraffic` deployment lifecycle events. Following completion of the validation tests, the Lambda `AfterAllowTraffic` function calls back CodeDeploy and delivers a result of `Succeeded` or `Failed`.

> **Important**
>
> The deployment is considered to have failed if CodeDeploy is not notified by the Lambda validation function within one hour.

Before invoking a Lambda hook function, the server must be notified of the deployment ID and the lifecycle event hook execution ID using the `putLifecycleEventHookExecutionStatus` command.

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 CodeDeploy.
    var params = {
        deploymentId: deploymentId,
```

```
        lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
        status: 'Succeeded' // status can be 'Succeeded' or 'Failed'
    };

    // Pass 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 AWS Lambda deployment

Topics

- List of lifecycle event hooks for an AWS Lambda deployment
- Run order of hooks in a Lambda function version deployment
- Structure of 'hooks' section
- Sample Lambda 'hooks' function

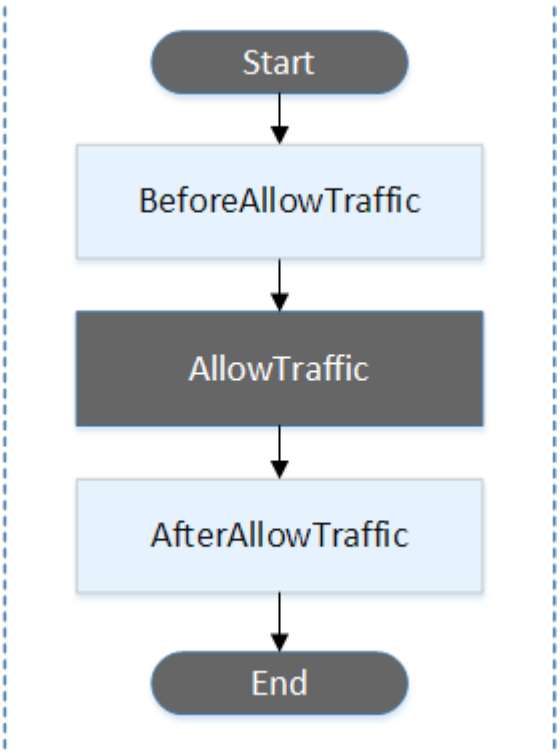
List of lifecycle event hooks for an AWS Lambda deployment

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. Here are descriptions of the hooks 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 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 CodeDeploy and delivers a result of `Succeeded` or `Failed`.

> **Important**
>
> The deployment is considered to have failed if CodeDeploy is not notified by the Lambda validation function within one hour.

Before invoking a Lambda hook function, the server must be notified of the deployment ID and the lifecycle event hook execution ID using the `putLifecycleEventHookExecutionStatus` command.

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 CodeDeploy.
    var params = {
```

```
        deploymentId: deploymentId,
        lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
        status: 'Succeeded' // status can be 'Succeeded' or 'Failed'
    };

    // Pass 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
- Lifecycle event hook availability
- Run order of hooks in a deployment
- Structure of 'hooks' section
- Referencing files in your hook scripts
- Environment variable availability for hooks
- Hooks example

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. Here are descriptions of the hooks 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.

- `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 for 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 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 a failed ApplicationStop, BeforeBlockTraffic, or AfterBlockTraffic deployment lifecycle event.

- `DownloadBundle` – During this deployment lifecycle event, the 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 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.

- `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 CodeDeploy agent copies the revision files from the temporary location to the final destination folder. This event is reserved for the 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 a failed ApplicationStop, BeforeBlockTraffic, or AfterBlockTraffic deployment lifecycle event.
- `BlockTraffic` – During this deployment lifecycle event, internet traffic is blocked from accessing instances that are currently serving traffic. This event is reserved for the 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 their respective load balancer.

  To troubleshoot a deployment that fails during the `AfterBlockTraffic` deployment lifecycle event, see Troubleshooting a failed ApplicationStop, BeforeBlockTraffic, or AfterBlockTraffic deployment lifecycle event.
- `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 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	Auto Scaling launch deployment[1]	Auto Scaling termination deployment[1]	In-place deployment[2]	Blue/green deployment: Original instances	Blue/green deployment: Replacement instances	Blue/green deployment rollback: Original instances	Blue/green deployment rollback: Replacement instances
ApplicationStop	✓	✓	✓		✓		
DownloadBundle[3]	✓		✓		✓		
BeforeInstall	✓		✓		✓		
Install[3]	✓		✓		✓		
AfterInstall	✓		✓		✓		
ApplicationStart	✓		✓		✓		
ValidateService	✓		✓		✓		
BeforeBlockTraffic		✓	✓	✓			✓
BlockTraffic[3]		✓	✓	✓			✓

Lifecycle event name	Auto Scaling launch deployment[1]	Auto Scaling termination deployment[1]	In-place deployment[2]	Blue/green deployment: Original instances	Blue/green deployment: Replacement instances	Blue/green deployment rollback: Original instances	Blue/green deployment rollback: Replacement instances
AfterBlockTraffic		✓	✓	✓			✓
BeforeAllowTraffic	✓		✓		✓	✓	
AllowTraffic[3]	✓		✓		✓	✓	
AfterAllowTraffic	✓		✓		✓	✓	

[1] For information about Amazon EC2 Auto Scaling deployments, see How Amazon EC2 Auto Scaling works with CodeDeploy.

[2] Also applies to the rollback of an in-place deployment.

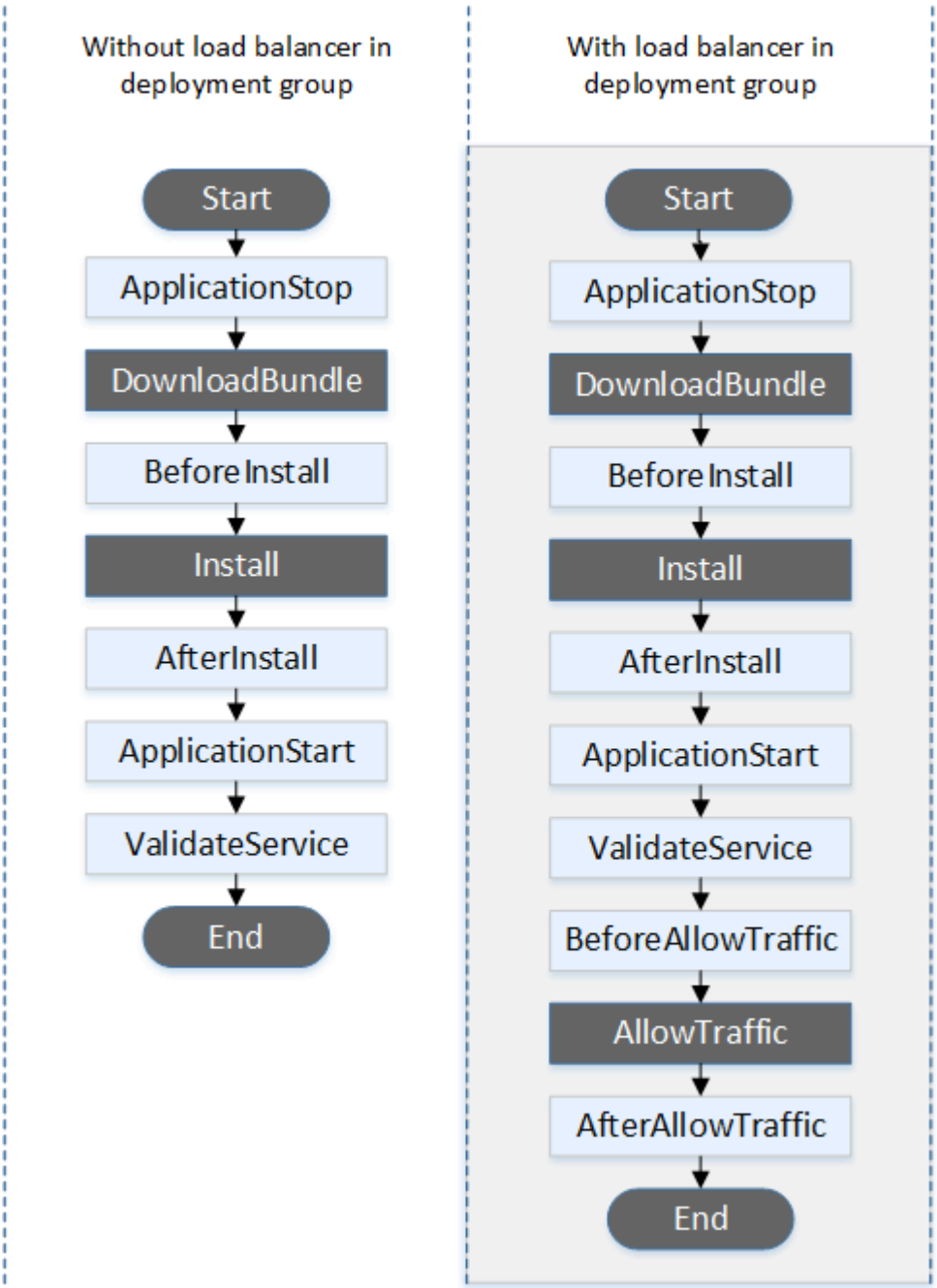
[3] Reserved for CodeDeploy operations. Cannot be used to run scripts.

Run order of hooks in a deployment

Auto Scaling launch deployments

During an Auto Scaling launch deployment, CodeDeploy runs event hooks in the following order.

For more information about Auto Scaling launch deployments, see How Amazon EC2 Auto Scaling works with CodeDeploy.



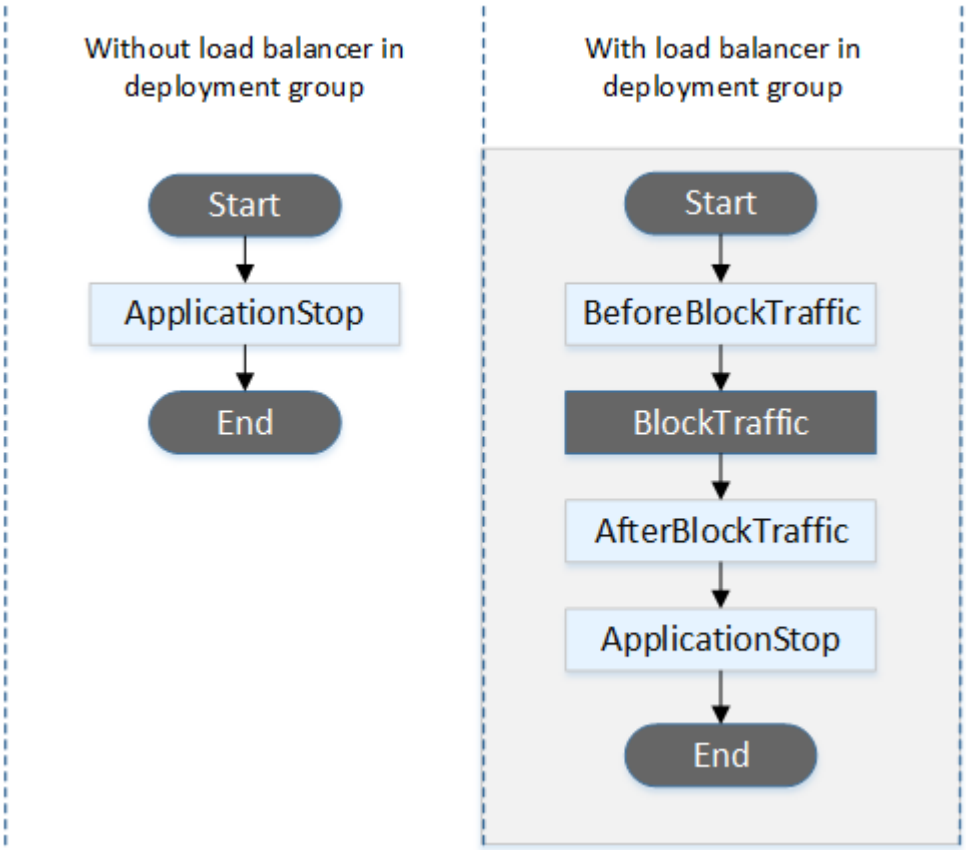
Note

The **Start**, **DownloadBundle**, **Install**, **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.

Auto Scaling termination deployments

During an Auto Scaling termination deployment, CodeDeploy runs event hooks in the following order.

For more information about Auto Scaling termination deployments, see Enabling termination deployments during Auto Scaling scale-in events.

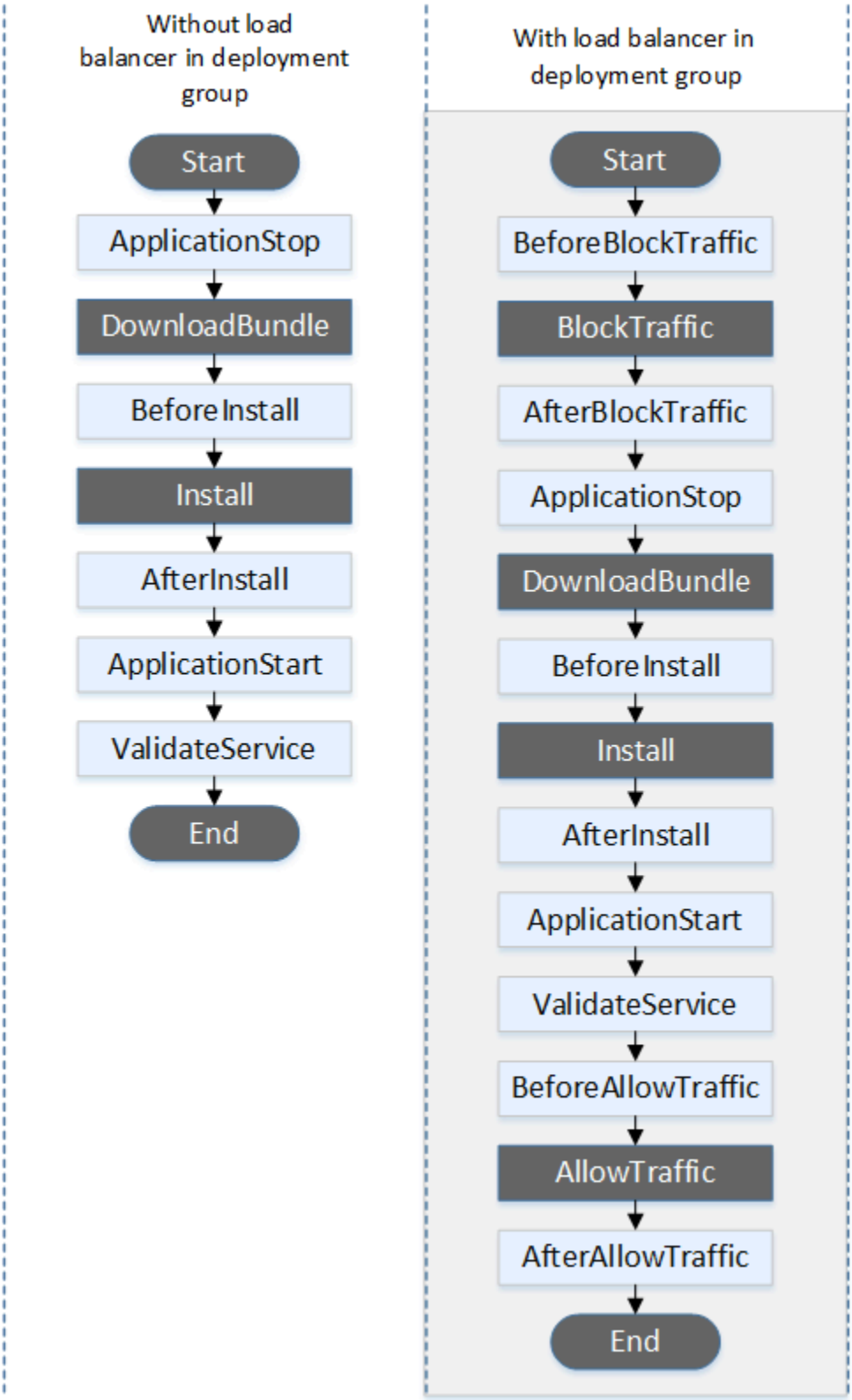


> **Note**
>
> The **Start**, **BlockTraffic**, and **End** events in the deployment cannot be scripted, which is why they appear in gray in this diagram.

In-place deployments

In an in-place deployment, including the rollback of an in-place deployment, event hooks are run in the following order:

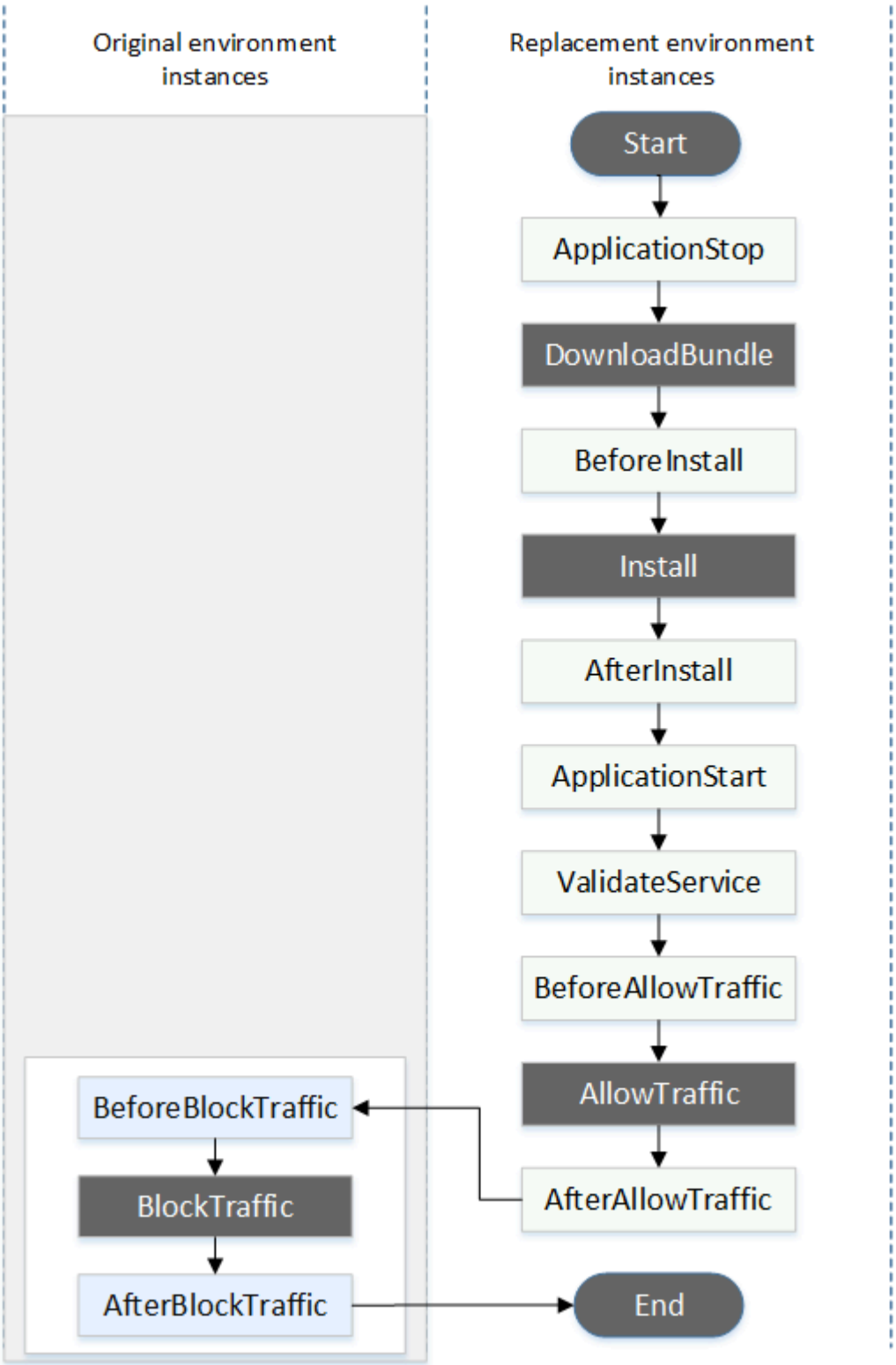
> **Note**
>
> For in-place deployments, the six hooks related to blocking and allowing traffic apply only if you specify a Classic Load Balancer, Application Load Balancer, or Network Load Balancer from Elastic Load Balancing in the deployment group.


Without load balancer in deployment group
Start
ApplicationStop
DownloadBundle
BeforeInstall
Install
AfterInstall
ApplicationStart
ValidateService
End
With load balancer in deployment group
Start
BeforeBlockTraffic
BlockTraffic
AfterBlockTraffic
ApplicationStop
DownloadBundle
BeforeInstall
Install
AfterInstall
ApplicationStart
ValidateService
BeforeAllowTraffic
AllowTraffic
AfterAllowTraffic
End

> **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:


Original environment instances
Replacement environment instances
Start
ApplicationStop
DownloadBundle
BeforeInstall
Install
AfterInstall
ApplicationStart
ValidateService
BeforeAllowTraffic
AllowTraffic
AfterAllowTraffic
End
BeforeBlockTraffic
BlockTraffic
AfterBlockTraffic

> **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. The location of scripts you specify in the `hooks` section is relative to the root of the application revision bundle. For more information, see Plan a revision for CodeDeploy.

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 CodeDeploy agent running on the instance. 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.

Referencing files in your hook scripts

If you are hooking up a script to a CodeDeploy lifecycle event as described in AppSpec 'hooks' section, and you want to reference a file (for example, `helper.sh`) in your script, then you will need to specify `helper.sh` using:

- (Recommended) An absolute path. See Using absolute paths.
- A relative path. See Using relative paths.

Using absolute paths

To reference a file using its *absolute* path, you can either:

- Specify the absolute path in the `files` section of the AppSpec file, in the `destination` property. Then, specify the same absolute path in your hook script. For more information, see AppSpec 'files' section (EC2/On-Premises deployments only).
- Specify a dynamic absolute path in your hook script. For more information, see Deployment archive location.

Deployment archive location

During the DownloadBundle lifecycle event, the CodeDeploy agent extracts the revision for the deployment to a directory that has the following format:

`root-directory``/`*`deployment-group-id`*`/`*`deployment-id`*`/deployment-archive`

The *`root-directory`* portion of the path is always set to either the default shown in the following table, or is controlled by the `:root_dir` configuration setting. For more information about configuration settings, see CodeDeploy agent configuration reference.

Agent platform	Default root directory
Linux – all rpm distributions	`/opt/codedeploy-agent/deployment-root`
Ubuntu Server – all deb distributions	`/opt/codedeploy-agent/deployment-root`
Windows Server	`%ProgramData%\Amazon\CodeDeploy`

From your hook scripts, you could access the current deployment archive using the the root directory path and the `DEPLOYMENT_ID` and `DEPLOYMENT_GROUP_ID` environment variables. For more information about variables you can use, see Environment variable availability for hooks.

For example, here is how you could access a `data.json` file that resides at the root of your revision on Linux:

```
#!/bin/bash

rootDirectory="/opt/codedeploy-agent/deployment-root" # note: this will be different if
 you
                                                       # customize the :root_dir
 configuration
dataFile="$rootDirectory/$DEPLOYMENT_GROUP_ID/$DEPLOYMENT_ID/deployment-archive/
data.json"
data=$(cat dataFile)
```

As another example, here is how you could access a `data.json` file that resides at the root of your revision using Powershell on Windows:

```
$rootDirectory="$env:ProgramData\Amazon\CodeDeploy" # note: this will be different if
 you
                                                     # customize the :root_dir
 configuration
$dataFile="$rootDirectory\$env:DEPLOYMENT_GROUP_ID\$env:DEPLOYMENT_ID\deployment-
archive\data.json"
$data=(Get-Content $dataFile)
```

Using relative paths

To reference a file using its *relative* path, you'll need to know the CodeDeploy agent's working directory. File paths are relative to this directory.

The following table shows the working directory for each supported platform of the CodeDeploy agent.

Agent platform	Process management method	Working directory for lifecycle event scripts
Linux – all rpm distributions	systemd (default)	`/`
	init.d – Learn more	`/opt/codedeploy-agent`
Ubuntu Server – all debian distributions	all	`/opt/codedeploy-agent`
Windows Server	not applicable	`C:\Windows\System32`

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 CodeDeploy that is part of the current deployment (for example, `WordPress_App`).

DEPLOYMENT_ID

The ID CodeDeploy has assigned to the current deployment (for example, `d-AB1CDEF23`).

DEPLOYMENT_GROUP_NAME

The name of the deployment group in CodeDeploy that is part of the current deployment (for example, `WordPress_DepGroup`).

DEPLOYMENT_GROUP_ID

The ID of the deployment group in 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.

There are additional environment variables available to hook scripts depending on the source of the deployment bundle:

Bundle from Amazon S3

- **BUNDLE_BUCKET**

  The name of the Amazon S3 bucket from which the deployment bundle was downloaded (for example, `my-s3-bucket`).

- **BUNDLE_KEY**

  The object key for the downloaded bundle within the Amazon S3 bucket (for example, `WordPress_App.zip`).

- **BUNDLE_VERSION**

  The object version for the bundle (for example, `3sL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo`). This variable is only set if the Amazon S3 bucket has object versioning enabled.

- **BUNDLE_ETAG**

  The object etag for the bundle (for example, `b10a8db164e0754105b7a99be72e3fe5-4`).

Bundle from GitHub

- **BUNDLE_COMMIT**

  The SHA256 commit hash of the bundle generated by Git (for example, `d2a84f4b8b650937ec8f73cd8be2c74add5a911ba64df27458ed8229da804a26`).

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 webpage 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 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 CodeDeploy.

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 Amazon ECS deployment
- AppSpec File example for an AWS Lambda deployment
- AppSpec File example for an EC2/On-Premises deployment

AppSpec File example for an Amazon ECS deployment

Here is an example of an AppSpec file written in YAML for deploying an Amazon ECS service.

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
```

```
      Properties:
        TaskDefinition: "arn:aws:ecs:us-east-1:111222333444:task-definition/my-task-
definition-family-name:1"
        LoadBalancerInfo:
          ContainerName: "SampleApplicationName"
          ContainerPort: 80
# Optional properties
        PlatformVersion: "LATEST"
        NetworkConfiguration:
          AwsvpcConfiguration:
            Subnets: ["subnet-1234abcd","subnet-5678abcd"]
            SecurityGroups: ["sg-12345678"]
            AssignPublicIp: "ENABLED"
        CapacityProviderStrategy:
          - Base: 1
            CapacityProvider: "FARGATE_SPOT"
            Weight: 2
          - Base: 0
            CapacityProvider: "FARGATE"
            Weight: 1
Hooks:
  - BeforeInstall: "LambdaFunctionToValidateBeforeInstall"
  - AfterInstall: "LambdaFunctionToValidateAfterInstall"
  - AfterAllowTestTraffic: "LambdaFunctionToValidateAfterTestTrafficStarts"
  - BeforeAllowTraffic: "LambdaFunctionToValidateBeforeAllowingProductionTraffic"
  - AfterAllowTraffic: "LambdaFunctionToValidateAfterAllowingProductionTraffic"
```

Here is a version of the preceding example written in JSON.

```
{
    "version": 0.0,
    "Resources": [
        {
            "TargetService": {
                "Type": "AWS::ECS::Service",
                "Properties": {
                    "TaskDefinition": "arn:aws:ecs:us-east-1:111222333444:task-
definition/my-task-definition-family-name:1",
                    "LoadBalancerInfo": {
                        "ContainerName": "SampleApplicationName",
                        "ContainerPort": 80
                    },
                    "PlatformVersion": "LATEST",
```

```
                    "NetworkConfiguration": {
                        "AwsvpcConfiguration": {
                            "Subnets": [
                                "subnet-1234abcd",
                                "subnet-5678abcd"
                            ],
                            "SecurityGroups": [
                                "sg-12345678"
                            ],
                            "AssignPublicIp": "ENABLED"
                        }
                    },
                    "CapacityProviderStrategy": [
                        {
                            "Base" : 1,
                            "CapacityProvider" : "FARGATE_SPOT",
                            "Weight" : 2
                        },
                        {
                            "Base" : 0,
                            "CapacityProvider" : "FARGATE",
                            "Weight" : 1
                        }
                    ]
                }
            }
        }
    ],
    "Hooks": [
        {
            "BeforeInstall": "LambdaFunctionToValidateBeforeInstall"
        },
        {
            "AfterInstall": "LambdaFunctionToValidateAfterInstall"
        },
        {
            "AfterAllowTestTraffic": "LambdaFunctionToValidateAfterTestTrafficStarts"
        },
        {
            "BeforeAllowTraffic":
 "LambdaFunctionToValidateBeforeAllowingProductionTraffic"
        },
        {
```

```
            "AfterAllowTraffic":
 "LambdaFunctionToValidateAfterAllowingProductionTraffic"
        }
    ]
}
```

Here is the sequence of events during deployment:

1. Before the updated Amazon ECS application is installed on the replacement task set, the Lambda function called `LambdaFunctionToValidateBeforeInstall` runs.
2. After the updated Amazon ECS application is installed on the replacement task set, but before it receives any traffic, the Lambda function called `LambdaFunctionToValidateAfterInstall` runs.
3. After the Amazon ECS application on the replacement task set starts receiving traffic from the test listener, the Lambda function called `LambdaFunctionToValidateAfterTestTrafficStarts` runs. This function likely runs validation tests to determine if the deployment continues. If you do not specify a test listener in your deployment group, this hook is ignored.
4. After any validation tests in the `AfterAllowTestTraffic` hook are completed, and before production traffic is served to the updated Amazon ECS application, the Lambda function called `LambdaFunctionToValidateBeforeAllowingProductionTraffic` runs.
5. After production traffic is served to the updated Amazon ECS application on the replacement task set, the Lambda function called `LambdaFunctionToValidateAfterAllowingProductionTraffic` runs.

The Lambda functions that run during any hook can perform validation tests or gather traffic metrics.

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 `codedeploy` 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. CodeDeploy raises 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

You can use the AWS-provided AppSpec Assistant script to validate the contents of an AppSpec file. You can find the script along with AppSpec file templates on GitHub.

You can also use 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.

CodeDeploy agent configuration reference

When the CodeDeploy agent is installed, a configuration file is placed on the instance. This configuration file specifies directory paths and other settings for 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:	Set to `true` for the CodeDeploy agent to capture wire logs from Amazon S3 and write

them to a file named `codedeploy-agent.wire.log` in the location pointed to by the **:log_dir:** setting.

> **Warning**
>
> You should set **:log_aws_wire:** to `true` only for the amount of time required to capture wire logs. The `codedeploy-agent.wire.log` 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 `true`. The wire logs contain information about all Amazon S3 activity associated with the AWS account while this setting was set to `true`, not just activity related to CodeDeploy deployments.

The default setting is `false`.

This setting applies to all instance types. You must add this configuration setting to Windows Server instances to be able to use it.

:log_dir:	The folder on the instance where log files related to CodeDeploy agent operations are stored. The default setting is `'/var/log/aws/codedeploy-agent'` for Amazon Linux, Ubuntu Server, and RHEL instances and `C:\ProgramData\Amazon\CodeDeploy\log` for Windows Server instances.
:pid_dir:	The folder where `codedeploy-agent.pid` is stored. This file contains the process ID (PID) of the CodeDeploy agent. The default setting is `'/opt/codedeploy-agent/state/.pid'` . This setting applies to Amazon Linux, Ubuntu Server, and RHEL instances only.
:program_name:	The CodeDeploy agent program name. The default setting is `codedeploy-agent` . This setting applies to Amazon Linux, Ubuntu Server, and RHEL instances only.
:root_dir:	The folder where related revisions, deployment history, and deployment scripts on the instance are stored. The default setting is `/opt/codedeploy-agent/deployment-root` for Amazon Linux, Ubuntu Server, and RHEL instances and `C:\ProgramData\Amazon\CodeDeploy` for Windows Server instances.

:verbose:	Set to `true` for the CodeDeploy agent to print debug messages log files on the instance. The default setting is `false`.
:wait_between_runs:	The interval, in seconds, between CodeDeploy agent polling of CodeDeploy for pending deployments. The default setting is 1.
:on_premises_config_file:	For on-premises instances, the path to an alternate location for the configuration file named `codedeploy.onpremises.yml` (for Ubuntu Server and RHEL) or `conf.onpremises.yml` (for Windows Server). By default, these files are stored in `/etc/codedeploy-agent/conf/codedeploy.onpremises.yml` for Ubuntu Server and RHEL and `C:\ProgramData\Amazon\CodeDeploy\conf.onpremises.yml` for Windows Server. Available in version 1.0.1.686 and later versions of the CodeDeploy agent.
:proxy_uri:	(Optional) The HTTP proxy through which you want the CodeDeploy agent to connect to AWS for your CodeDeploy operations. Use a format similar to `https://user:password@my.proxy:443/path?query`. Available in version 1.0.1.824 and later versions of the CodeDeploy agent.

:max_revisions:	(Optional) The number of application revisions for a deployment group that you want the CodeDeploy agent to archive. Any revisions that exceed the number specified are deleted. Enter any positive integer. If no value is specified, CodeDeploy will retain the five most recent revisions in addition to the currently deployed revision. Supported in version 1.0.1.966 and later versions of the CodeDeploy agent.
:enable_auth_policy:	(Optional) Set to `true` if you want to use IAM authorization to configure access control and limit permission of the IAM role or user that the CodeDeploy Agent is using. To Use CodeDeploy with Amazon Virtual Private Cloud, this value must be `true`. The default setting is `false`.
:disable_imds_v1:	This setting is available with CodeDeploy agent 1.7.0 and later. Set to `true` to disable the fallback to IMDSv1 when IMDSv2 errors occur. Defaults to `false` (enable the fallback).

Related topics

Working with the CodeDeploy agent

Managing CodeDeploy agent operations

CloudFormation templates for CodeDeploy reference

This section introduces the CloudFormation resources, transform, and hook designed to work with CodeDeploy deployments. For a walkthrough of creating a stack update managed by the CloudFormation hook for CodeDeploy, see Create an Amazon ECS blue/green deployment through CloudFormation

> **Note**
>
> CloudFormation hooks are part of the CloudFormation components for AWS and are different from CodeDeploy lifecycle event hooks.

In addition to the other methods available to you in CodeDeploy, you can use CloudFormation templates to perform the following tasks:

- Create applications.
- Create deployment groups and specify a target revision.
- Create deployment configurations.
- Create Amazon EC2 instances.

CloudFormation is a service that helps you model and set up your AWS resources using templates. An 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 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 CloudFormation templates that are compatible with CodeDeploy in your organization, as an administrator, you must grant access to CloudFormation and to the AWS services and actions on which CloudFormation depends. To grant permissions to create applications, deployment groups, and deployment configurations, add the following policy to the permission set of the users who will work with CloudFormation:

JSON

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

For more information about policies, see the following topics:

- To view the policy that must be added to the permission set of users who will create Amazon EC2 instances, see Create an Amazon EC2 instance for CodeDeploy (CloudFormation template).
- For information about adding policies to permission sets, see Create a permission set in the *IAM User Guide*.
- To learn how to restrict users to a limited set of CodeDeploy actions and resources, see AWS managed (predefined) policies for CodeDeploy.

The following table shows the actions an 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 CloudFormation template.

Action	CloudFormation reference	Reference type
Create a CodeDeploy application.	AWS::CodeDeploy::application	CloudFormation resource
Create and specify the details for a deployment group to be used to deploy your application revisions. [1]	AWS::CodeDeploy::DeploymentGroup	CloudFormation resource

Action	CloudFormation reference	Reference type
Create a set of deployment rules, deployment success conditions, and deployment failure conditions that CodeDeploy will use during a deployment.	AWS::CodeDeploy::DeploymentConfig	CloudFormation resource
Create an Amazon EC2 instance. [2]	AWS::EC2::instance	CloudFormation resource
Use the CloudFormation `AWS::CodeDeployBlueGreen` transform and `AWS::CodeDeploy::BlueGreen` hook to manage stack updates, create resources, and shift traffic for CodeDeploy blue/green deployments.[3]	AWS::CodeDeployBlueGreen	The `AWS::CodeDeployBlueGreen` transform is a macro hosted by CloudFormation
	AWS::CodeDeploy::BlueGreen	The `AWS::CodeDeploy::BlueGreen` hook is structured as a `Hook` resource in CloudFormation. The hook includes parameters that take the place of your CodeDeploy AppSpec file by pointing to designated CodeDeploy lifecycle event hooks.

Action	CloudFormation reference	Reference type
[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 CodeDeploy DeploymentGroup deployment revision S3Location and CodeDeploy DeploymentGroup deployment revision GitHubLocation in the *AWS CloudFormation User Guide*. [2] We provide templates you can use to create Amazon EC2 instances in the regions in which CodeDeploy is supported. For more information about using these templates, see Create an Amazon EC2 instance for CodeDeploy (CloudFormation template). [3]Only Amazon ECS blue/green deployments are supported by this deployment configuration. For more information about deployment configurations for Amazon ECS blue/green deployments through CloudFormation, see Deployment configurations for CloudFormation blue/green deployments (Amazon ECS). For more information about Amazon ECS blue/green deployments through CloudFormation and how to view your deployment in CodeDeploy, see Create an Amazon ECS blue/green deployment through CloudFormation.		

Use CodeDeploy with Amazon Virtual Private Cloud

If you use Amazon Virtual Private Cloud (Amazon VPC) to host your AWS resources, you can establish a private connection between your VPC and CodeDeploy. You can use this connection to enable CodeDeploy to communicate with your resources on your VPC without going through the public internet.

Amazon VPC is an AWS service that you can use to launch AWS resources in a virtual network that you define. With a VPC, you have control over your network settings, such the IP address range, subnets, route tables, and network gateways. With VPC endpoints, the routing between the VPC

and AWS services is handled by the AWS network, and you can use IAM policies to control access to service resources.

To connect your VPC to CodeDeploy, you define an *interface VPC endpoint* for CodeDeploy. An interface endpoint is an elastic network interface with a private IP address that serves as an entry point for traffic destined to a supported AWS service. The endpoint provides reliable, scalable connectivity to CodeDeploy without requiring an internet gateway, network address translation (NAT) instance, or VPN connection. For more information, see What Is Amazon VPC in the *Amazon VPC User Guide*.

Interface VPC endpoints are powered by AWS PrivateLink, an AWS technology that enables private communication between AWS services using an elastic network interface with private IP addresses. For more information, see AWS PrivateLink.

The following steps are for users of Amazon VPC. For more information, see Getting Started in the *Amazon VPC User Guide*.

Availability

CodeDeploy has two VPC endpoints: one for CodeDeploy agent operations, and one for CodeDeploy API operations. The table below shows the supported AWS Regions for each endpoint.

For more information and for FIPS endpoints, see AWS CodeDeploy endpoints and quotas.

Region name	Region code	Agent endpoint	API endpoint	FIPS compliant Region?
US East (N. Virginia)	us-east-1	Yes	Yes	Yes
US East (Ohio)	us-east-2	Yes	Yes	Yes
US West (N. California)	us-west-1	Yes	Yes	Yes
US West (Oregon)	us-west-2	Yes	Yes	Yes
Africa (Cape Town)	af-south-1	Yes	No	

Region name	Region code	Agent endpoint	API endpoint	FIPS compliant Region?
Asia Pacific (Hong Kong)	ap-east-1	Yes	Yes	
Asia Pacific (Hyderabad)	ap-south-2	Yes	No	
Asia Pacific (Jakarta)	ap-southeast-3	Yes	No	
Asia Pacific (Melbourne)	ap-southeast-4	Yes	No	
Asia Pacific (Mumbai)	ap-south-1	Yes	Yes	
Asia Pacific (Osaka)	ap-northeast-3	Yes	No	
Asia Pacific (Seoul)	ap-northeast-2	Yes	Yes	
Asia Pacific (Singapore)	ap-southeast-1	Yes	Yes	
Asia Pacific (Sydney)	ap-southeast-2	Yes	Yes	
Asia Pacific (Tokyo)	ap-northeast-1	Yes	Yes	
Canada (Central)	ca-central-1	Yes	Yes	
China (Beijing)	cn-north-1	Yes	No	
China (Ningxia)	cn-northwest-1	No	No	

Region name	Region code	Agent endpoint	API endpoint	FIPS compliant Region?
Europe (Frankfurt)	eu-central-1	Yes	Yes	
Europe (Ireland)	eu-west-1	Yes	Yes	
Europe (London)	eu-west-2	Yes	Yes	
Europe (Milan)	eu-south-1	Yes	No	
Europe (Paris)	eu-west-3	Yes	Yes	
Europe (Spain)	eu-south-2	Yes	No	
Europe (Stockholm)	eu-north-1	Yes	Yes	
Europe (Zurich)	eu-central-2	Yes	No	
Israel (Tel Aviv)	il-central-1	Yes	Yes	
Middle East (Bahrain)	me-south-1	Yes	Yes	
Middle East (UAE)	me-central-1	Yes	No	
South America (São Paulo)	sa-east-1	Yes	Yes	
AWS GovCloud (US-East)	us-gov-east-1	No	No	Yes
AWS GovCloud (US-West)	us-gov-west-1	No	No	Yes

Create VPC endpoints for CodeDeploy

To start using CodeDeploy with your VPC, create an interface VPC endpoint for CodeDeploy. CodeDeploy requires separate endpoints for agent Git operations and for CodeDeploy API operations. Depending on your business needs, you might need to create more than one VPC endpoint. When you create a VPC endpoint for CodeDeploy, choose **AWS Services**, and in **Service Name**, choose from the following options:

- **com.amazonaws.*region*.codedeploy**: Choose this option if you want to create a VPC endpoint for CodeDeploy API operations. For example, choose this option if your users use the AWS CLI, the CodeDeploy API, or the AWS SDKs to interact with CodeDeploy for operations such as `CreateApplication`, `GetDeployment`, and `ListDeploymentGroups`.
- **com.amazonaws.*region*.codedeploy-fips**: Choose this option if you want to create a VPC endpoint for CodeDeploy API operations for the FIPS endpoint.
- **com.amazonaws.*region*.codedeploy-commands-secure**: Choose this option if you want to create a VPC endpoint for CodeDeploy agent operations. You will also need to set `:enable_auth_policy:` to `true` in your agent configuration file and attach the required permissions. For more information, see Configure the CodeDeploy agent and IAM permissions.

If you are using Lambda or ECS deployments, you only need to create a VPC endpoint for **com.amazonaws.*region*.codedeploy**. Customers using Amazon EC2 deployments will need VPC endpoints for both **com.amazonaws.*region*.codedeploy** and **com.amazonaws.*region*.codedeploy-commands-secure**.

Configure the CodeDeploy agent and IAM permissions

To use Amazon VPC endpoints with CodeDeploy, you must set the value of `:enable_auth_policy:` to `true` in the agent configuration file located on your EC2 or on-premises instances. For more information on the agent configuration file, see CodeDeploy agent configuration reference.

You must also add the following IAM permissions to your Amazon EC2 instance profile (if you're using Amazon EC2 instances) or IAM user or role (if you are using on-premises instances).

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

```
      "Action": [
        "codedeploy-commands-secure:GetDeploymentSpecification",
        "codedeploy-commands-secure:PollHostCommand",
        "codedeploy-commands-secure:PutHostCommandAcknowledgement",
        "codedeploy-commands-secure:PutHostCommandComplete"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

For more information, see Creating an Interface Endpoint in the *Amazon VPC User Guide*.

CodeDeploy resource kit reference

Many of the files CodeDeploy relies on are stored in publicly available, AWS region-specific Amazon S3 buckets. These files include installation files for the CodeDeploy agent, templates, and sample application files. We call this collection of files the CodeDeploy Resource Kit.

Topics

- Resource kit bucket names by Region
- Resource kit contents
- Display a list of the resource kit files
- Download the resource kit files

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 CodeDeploy Resource Kit files.

> **Note**
>
> To access the Amazon S3 bucket in the Asia Pacific (Hong Kong) Region, you must enable the region in your AWS account. For more information, see Managing AWS Regions.

Region name	*Bucket-name* replacement	Region identifier
US East (N. Virginia)	aws-codedeploy-us-east-1	us-east-1
US East (Ohio)	aws-codedeploy-us-east-2	us-east-2
US West (N. California)	aws-codedeploy-us-west-1	us-west-1
US West (Oregon)	aws-codedeploy-us-west-2	us-west-2
Africa (Cape Town)	aws-codedeploy-af-south-1	af-south-1
Asia Pacific (Hong Kong)	aws-codedeploy-ap-east-1	ap-east-1
Asia Pacific (Hyderabad)	aws-codedeploy-ap-south-2	ap-south-2
Asia Pacific (Jakarta)	aws-codedeploy-ap-southeast-3	ap-southeast-3
Asia Pacific (Melbourne)	aws-codedeploy-ap-southeast-4	ap-southeast-4
Asia Pacific (Mumbai)	aws-codedeploy-ap-south-1	ap-south-1
Asia Pacific (Osaka)	aws-codedeploy-ap-northeast-3	ap-northeast-3
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 (Tokyo)	aws-codedeploy-ap-northeast-1	ap-northeast-1
Canada (Central)	aws-codedeploy-ca-central-1	ca-central-1

Region name	*Bucket-name* replacement	Region identifier
Europe (Frankfurt)	aws-codedeploy-eu-central-1	eu-central-1
Europe (Ireland)	aws-codedeploy-eu-west-1	eu-west-1
Europe (London)	aws-codedeploy-eu-west-2	eu-west-2
Europe (Milan)	aws-codedeploy-eu-south-1	eu-south-1
Europe (Paris)	aws-codedeploy-eu-west-3	eu-west-3
Europe (Spain)	aws-codedeploy-eu-south-2	eu-south-2
Europe (Stockholm)	aws-codedeploy-eu-north-1	eu-north-1
Europe (Zurich)	aws-codedeploy-eu-central-2	eu-central-2
Israel (Tel Aviv)	aws-codedeploy-il-central-1	il-central-1
Middle East (Bahrain)	aws-codedeploy-me-south-1	me-south-1
Middle East (UAE)	aws-codedeploy-me-central-1	me-central-1
South America (São Paulo)	aws-codedeploy-sa-east-1	sa-east-1
AWS GovCloud (US-East)	aws-codedeploy-us-gov-east-1	us-gov-east-1
AWS GovCloud (US-West)	aws-codedeploy-us-gov-west-1	us-gov-west-1

Resource kit contents

The following table lists the files in the CodeDeploy Resource Kit.

File	Description
`LATEST_VERSION`	A file used by update mechanisms like Amazon EC2 Systems Manager to determine the latest version of the CodeDeploy agent.
`VERSION`	**The auto-update mechanism was removed in CodeDeploy agent version 1.1.0 and this file is no longer used.** A file used by CodeDeplo y agents to update themselves as they are running on instances.
`codedeploy-agent.noarch.rpm`	The 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 `-1.0-0`).
`codedeploy-agent_all.deb`	The CodeDeploy agent for Ubuntu Server. There may be several files with the same base file name, but different versions (such as `_1.0-0`).
`codedeploy-agent.msi`	The CodeDeploy agent for Windows Server. There may be several files with the same base file name, but different versions (such as `-1.0-0`).
`install`	A file you can use to more easily install the CodeDeploy agent.
`CodeDeploy_SampleCF_Templat e.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 `-1.0.0`).

File	Description
`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, Amazon EC2 Auto Scaling, CloudFormation, and Elastic Load Balancing. It also creates a load balancer and a 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 `-1.0`).
`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 --region us-east-2
  ```
- ```
  aws s3 ls --recursive s3://aws-codedeploy-us-east-1 --region us-east-1
  ```
- ```
  aws s3 ls --recursive s3://aws-codedeploy-us-west-1 --region us-west-1
  ```
- ```
  aws s3 ls --recursive s3://aws-codedeploy-us-west-2 --region us-west-2
  ```
- ```
  aws s3 ls --recursive s3://aws-codedeploy-ca-central-1 --region ca-central-1
  ```
- ```
  aws s3 ls --recursive s3://aws-codedeploy-eu-west-1 --region eu-west-1
  ```
- ```
  aws s3 ls --recursive s3://aws-codedeploy-eu-west-2 --region eu-west-2
  ```
- ```
  aws s3 ls --recursive s3://aws-codedeploy-eu-west-3 --region eu-west-3
  ```
- ```
  aws s3 ls --recursive s3://aws-codedeploy-eu-central-1 --region eu-central-1
  ```
- ```
  aws s3 ls --recursive s3://aws-codedeploy-il-central-1 --region il-central-1
  ```
- ```
  aws s3 ls --recursive s3://aws-codedeploy-ap-east-1 --region ap-east-1
  ```
- ```
  aws s3 ls --recursive s3://aws-codedeploy-ap-northeast-1 --region ap-northeast-1
  ```
- ```
  aws s3 ls --recursive s3://aws-codedeploy-ap-northeast-2 --region ap-northeast-2
  ```
- ```
  aws s3 ls --recursive s3://aws-codedeploy-ap-southeast-1 --region ap-southeast-1
  ```
- ```
  aws s3 ls --recursive s3://aws-codedeploy-ap-southeast-2 --region ap-southeast-2
  ```
- ```
  aws s3 ls --recursive s3://aws-codedeploy-ap-southeast-4 --region ap-southeast-4
  ```
- ```
  aws s3 ls --recursive s3://aws-codedeploy-ap-south-1 --region ap-south-1
  ```
- ```
  aws s3 ls --recursive s3://aws-codedeploy-sa-east-1 --region 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-il-central-1/samples/latest/SampleApp_Linux.zip . --region il-central-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-ap-east-1/samples/latest/SampleApp_Linux.zip . --region ap-east-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-southeast-4/samples/latest/SampleApp_Linux.zip . --region ap-southeast-4
  ```
- ```
  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-il-central-1 . --region il-central-1
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-ap-east-1 . --region ap-east-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-southeast-4 . --region ap-southeast-4
  ```
- ```
  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
  ```

CodeDeploy quotas

The following tables describe quotas in CodeDeploy.

> **Note**
>
> The *EC2/On-Premises in-place deployment run in hours* limit varies. For custom deployment configurations created before June 2023, the limit is 8 hours. For custom deployment

configurations created in June 2023 or later, the limit is 12 hours. For pre-defined deployment configurations, the limit is 12 hours.

Name	Default	Adjus e	Description
AWS Lambda deployment run in hours	Each supported Region: 50	No	Maximum number of hours an AWS Lambda deployment can run (48 hours for the maximum time between the first and last traffic shift plus one hour for each of two possible lifecycle hooks)
Applications associated per account per region	Each supported Region: 1,000	Yes	The maximum number of applications associated with an AWS account in a single region
Associated alarms per deployment group	Each supported Region: 50	Yes	The maximum number of alarms associated with a deployment group
Auto Scaling groups in a deployment group	Each supported Region: 10	Yes	Maximum number of Amazon EC2 Auto Scaling groups in a deployment group
Concurrent deployments per account	Each supported Region: 1,300	Yes	Maximum number of concurrent deploymen ts associated with an AWS account. Each deployment to a scaled-up Amazon EC2 instance in an Amazon EC2 Auto

Name	Default	Adjuse	Description
			Scaling group counts as a single concurrent deployment for each application that the EC2 instance is associated with.
Concurrent deployments per deployment group	Each supported Region: 1	No	Maximum number of concurrent deployments to a deployment group. This limit prevents concurrent deployments of the same application to the same deployment group.
Custom deployment configurations per account	Each supported Region: 50	No	Maximum number of custom deployment configurations associated with an AWS account
Deployment groups associated with a single application	Each supported Region: 1,000	Yes	Maximum number of deployment groups associated with a single application
EC2/On-Premises blue/green deployment run in hours	Each supported Region: 109	No	Maximum number of hours an EC2/On-Premises blue/green deployment can run (48 hours for each of the above two limits plus one hour for each of 13 possible lifecycle events)

Name	Default	Adjuse	Description
EC2/On-Premises in-place deployment run in hours	Each supported Region: 12	No	Maximum number of hours an EC2/On-Premises in-place deployment can run
Event notification triggers in a deployment group	Each supported Region: 10	Yes	Maximum number of event notification triggers in a deployment group
GitHub connection tokens per account	Each supported Region: 25	No	Maximum number of GitHub connection tokens for a single AWS account
Hours between the completion of a deployment and the termination of the original instances during an EC2/On-Premises blue/green deployment	Each supported Region: 48	No	Maximum number of hours between the completion of a deployment and the termination of the original instances during an EC2/On-Premises blue/green deployment
Hours between the deployment of a revision and when traffic shifts to the replacement instances during an EC2/On-Premises blue/green deployment	Each supported Region: 48	No	Maximum number of hours between the deployment of a revision and when traffic shifts to the replacement instances during an EC2/On-Premises blue/green deployment

Name	Default	Adjusse	Description
Instances count per deployment	us-east-1: 2,000 Each of the other supported Regions: 1,000	Yes	Maximum number of instances in a single deployment
Minutes a blue/green deployment can wait after a successful deployment before terminating instances from the original deployment	Each supported Region: 2,800	No	Maximum number of minutes a blue/green deployment can wait after a successful deployment before terminating instances from the original deployment
Minutes between the first and last traffic shift during an AWS Lambda canary or linear deployment	Each supported Region: 2,880	No	Maximum number of minutes between the first and last traffic shift during an AWS Lambda canary or linear deployment
Minutes until a deployment fails if a lifecycle event doesnt start	Each supported Region: 5	No	Maximum number of minutes until a deployment fails if a lifecycle event doesnt start after (1) a deployment is triggered by using the console or the AWS CLI create-deployment command, or (2) the previous lifecycle event is completed.

Name	Default	Adjuse	Description
Number of deployment groups that can be associated with an Amazon ECS service	Each supported Region: 1	No	Maximum number of deployment groups that can be associated with an Amazon ECS service
Number of instances that can be passed to the BatchGetOnPremises Instances API action	Each supported Region: 100	No	Maximum number of instances that can be passed to the BatchGetOnPremisesInstances API action
Number of instances used by concurrent deployments that are in progress per account	us-east-1: 3,000 Each of the other supported Regions: 1,000	Yes	Maximum number of instances that can be used by concurrent deployments that are in progress and associated with one account
Number of listeners for a traffic route during an Amazon ECS deployment	Each supported Region: 1	No	Maximum number of listeners for a traffic route during an Amazon ECS deployment
Seconds until a deployment lifecycle event fails if not completed	Each supported Region: 3,600 Seconds	No	Maximum number of seconds until a deployment lifecycle event fails if not completed
Size of deployment group name	Each supported Region: 100	No	Maximum number of characters in a deployment group name
Size of tag key	Each supported Region: 128	No	Maximum number of characters in a tag key

Name	Default	Adjuse	Description
Size of tag value	Each supported Region: 256	No	Maximum number of characters in a tag value
Tags in a deployment group	Each supported Region: 10	No	Maximum number of tags in a deployment group
Traffic that can be shifted in one increment during an AWS Lambda deployment	Each supported Region: 99	No	Maximum percentage of traffic that can be shifted in one increment during an AWS Lambda deployment

Troubleshooting CodeDeploy

Use the topics in this section to help solve problems and errors you might encounter when using CodeDeploy.

> **Note**
>
> You can identify the causes of many deployment failures 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 Monitoring Deployments with Amazon CloudWatch Tools.

Topics

- General troubleshooting issues
- Troubleshoot EC2/On-Premises deployment issues
- Troubleshoot Amazon ECS deployment issues
- Troubleshoot AWS Lambda deployment issues
- Troubleshoot deployment group issues
- Troubleshoot instance issues
- Troubleshoot GitHub token issues
- Troubleshoot Amazon EC2 Auto Scaling issues
- Error codes for AWS CodeDeploy

General troubleshooting issues

Topics

- General troubleshooting checklist
- CodeDeploy deployment resources are supported in only in some AWS Regions
- Procedures in this guide do not match the CodeDeploy console
- Required IAM roles are not available
- Using some text editors to create AppSpec files and shell scripts can cause deployments to fail
- Using Finder in macOS to bundle an application revision can cause deployments to fail

General troubleshooting checklist

You can use the following checklist to troubleshoot a failed deployment.

1. See View CodeDeploy deployment details and View Instance Details to determine why the deployment failed. If you cannot determine the cause, review the items in this checklist.
2. Check that you have correctly configured the instances:
   - Was the instance launched with an EC2 key pair specified? For more information, see EC2 Key Pairs in *Amazon EC2 User Guide*.
   - Is the correct IAM instance profile attached to the instance? For more information, see Configure an Amazon EC2 instance to work with CodeDeploy and Step 4: Create an IAM instance profile for your Amazon EC2 instances.
   - Was the instance tagged? For more information, see Working with tags in the console in *Amazon EC2 User Guide*.
   - Is the CodeDeploy agent installed, updated, and running on the instance? For more information, see Managing CodeDeploy agent operations. To check which version of the agent is installed, see Determine the version of the CodeDeploy agent.
3. Check the application and deployment group settings:
   - To check your application settings, see View application details with CodeDeploy.
   - To check your deployment group settings, see View deployment group details with CodeDeploy.
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 CodeDeploy and CodeDeploy AppSpec file reference.
   - Check your Amazon S3 bucket or GitHub repository to verify your application revision is in the expected location.
   - Review the details of your CodeDeploy application revision to ensure that it is registered correctly. For information, see View application revision details with CodeDeploy.
   - If you're deploying from Amazon S3, check your Amazon S3 bucket to verify CodeDeploy has been granted permissions to download the application revision. For information about bucket policies, see Deployment prerequisites.
   - If you're deploying from GitHub, check your GitHub repository to verify CodeDeploy has been granted permissions to download the application revision. For more information, see Create a deployment with CodeDeploy and GitHub authentication with applications in CodeDeploy.

5. Check that the service role is correctly configured. For information, see Step 2: Create a service role for CodeDeploy.
6. Confirm you followed the steps in Getting started with CodeDeploy to:
   - Provisioned a user with the appropriate permissions.
   - Install or upgrade and configure the AWS CLI.
   - Create an IAM instance profile and a service role.

   For more information, see Identity and access management for AWS CodeDeploy.
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.

CodeDeploy deployment resources are supported in only in some AWS Regions

If you do not see or cannot access applications, deployment groups, instances, or other deployment resources from the AWS CLI or the CodeDeploy console, make sure you're referencing one of the AWS Regions listed in Region and endpoints in *AWS General Reference*.

EC2 instances and Amazon EC2 Auto Scaling groups that are used in CodeDeploy deployments must be launched and created in one of these AWS 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 AWS Region.

If you're using the CodeDeploy console, on the navigation bar, from the region selector, choose one of the supported AWS 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 CodeDeploy Resource Kit bucket names and CodeDeploy agent installation procedures, are not included in this edition of the *CodeDeploy User Guide*.

For more information:

- CodeDeploy in *Getting Started with AWS in the China (Beijing) Region*
- *CodeDeploy User Guide for the China Regions* (English version | Chinese version)

Procedures in this guide do not match the CodeDeploy console

The procedures in this guide are written to reflect the new console design. If you are using the older version of the console, many of the concepts and basic procedures in this guide still apply. To access help in the new console, choose the information icon.

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 CodeDeploy no longer works as expected. To fix this problem, you must manually re-create the deleted IAM role.

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 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 CodeDeploy console, on the event details page for the deployment, choose **View logs**. (Or 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 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 might be introduced. For

examples of these types of text editors, search the internet for text editors that 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 editors.
- If you must use a text editor in Windows or macOS 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 macOS 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. 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

- CodeDeploy plugin CommandPoller missing credentials error
- Deployment fails with the message “Validation of PKCS7 signed message failed”
- 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"
- Long file paths cause "No such file or directory" errors
- Long-running processes can cause deployments to fail
- Troubleshooting a failed AllowTraffic lifecycle event with no error reported in the deployment logs

- Troubleshooting a failed ApplicationStop, BeforeBlockTraffic, or AfterBlockTraffic deployment lifecycle event
- Troubleshooting a failed DownloadBundle deployment lifecycle event with UnknownError: not opened for reading
- Troubleshooting all lifecycle events skipped errors
- Windows PowerShell scripts fail to use the 64-bit version of Windows PowerShell by default

> **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 CodeDeploy Logs in CloudWatch Logs Console.

> **Tip**
>
> For a runbook that automates many troubleshooting tasks related to EC2/On-Premises deployments, see AWSSupport-TroubleshootCodeDeploy in the *AWS Systems Manager Automation runbook reference*.

CodeDeploy plugin CommandPoller missing credentials error

If you receive an error similar to `InstanceAgent::Plugins::CodeDeployPlugin::CommandPoller: Missing credentials - please check if this instance was started with an IAM instance profile`, it might be caused by one of the following:

- The instance you are deploying to does not have an IAM instance profile associated with it.
- Your IAM instance profile does not have the correct permissions configured.

An IAM instance profile grants the CodeDeploy agent permission to communicate with CodeDeploy and to download your revision from Amazon S3. For EC2 instances, see Identity and access management for AWS CodeDeploy. For on-premises instances, see Working with On-Premises Instances.

Deployment fails with the message “Validation of PKCS7 signed message failed”

This error message indicates the instance is running a version of the 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 CodeDeploy agent, released in November 2015. Effective October 17, 2016, deployments fail if a version of the 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 CodeDeploy host agents older than version 1.0.1.85, and Update the CodeDeploy agent.

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 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, 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 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 deploy using CodeDeploy for the first time. For more information, see Create an EC2/On-Premises Compute Platform deployment (console) and Rollback behavior with existing content.

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 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 is more likely to succeed if you specify the application name and the deployment group with the same underlying deployment group ID you used before. 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 fails because even if the deployment group name is the same, the use of a different application name means a different underlying deployment group ID is being 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 fails because even if the deployment group name is the same, CodeDeploy references a different underlying deployment group ID.
- You deleted an application and deployment group in CodeDeploy, and 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 fails because even though the application and deployment group names are the same, 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 fails because CodeDeploy references 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 fails because CodeDeploy does not remove the existing file before the second deployment starts. Both deployments >reference different deployment group IDs.
- You deployed a revision in CodeDeploy, but there is at least one file with the same name and in the same location. The deployment fails because, by default, CodeDeploy does 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.) CodeDeploy uses the underlying deployment group ID and AppSpec file to remove all of the files it installed in the previous successful deployment.

Long file paths cause "No such file or directory" errors

For deployments to Windows instances, if you have a file path greater than 260 characters in the files section of your appspec.yml file, you may see deployments fail with an error similar to the following:

```
No such file or directory @ dir_s_mkdir - C:\your-long-file-path
```

This error occurs because Windows by default does not allow file paths greater than 260 characters, as detailed in Microsoft's documentation.

For CodeDeploy agent versions 1.4.0 or later, you can enable long file paths in two ways, depending on the agent installation process:

If the CodeDeploy agent has not yet been installed:

1. On the machine where you plan to install the CodeDeploy agent, enable the `LongPathsEnabled` Windows registry key using this command:

```
New-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\FileSystem"
          -Name "LongPathsEnabled" -Value 1 -PropertyType DWORD -Force
```

2. Install the CodeDeploy agent. For more information, see Install the CodeDeploy agent.

If the CodeDeploy agent has already been installed:

1. On the CodeDeploy agent machine, enable the `LongPathsEnabled` Windows registry key using this command:

```
New-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\FileSystem"
-Name "LongPathsEnabled" -Value 1 -PropertyType DWORD -Force
```

2. Restart the CodeDeploy agent for the registry key change to take effect. To restart the agent, use this command:

```
powershell.exe -Command Restart-Service -Name codedeployagent
```

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, CodeDeploy might 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, 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` starts and runs for three minutes (180 seconds), which is two minutes (120 seconds) past the time CodeDeploy expects `sleep.sh` (and, by relation, `after-install.sh`) to stop running. After the timeout of one minute (60 seconds), 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 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 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`.

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 the deployment logs do not indicate the cause for the failure.

This failure is typically due to incorrectly configured health checks 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 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 a failed ApplicationStop, BeforeBlockTraffic, or AfterBlockTraffic deployment lifecycle event

During a deployment, the 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 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 never runs successfully.

Use the 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**. Or 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 never runs successfully, create a deployment and specify that the ApplicationStop, BeforeBlockTraffic, and AfterBlockTraffic failures should be ignored. There are two ways to do this:

- Use the 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 continues 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 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 CodeDeploy to Amazon S3 (EC2/On-Premises deployments only) and Deployment prerequisites.
- The instances you deploy to are associated with one AWS Region (for example, US West (Oregon)), but the Amazon S3 bucket that contains the application revision is associated with another AWS Region (for example, US East (N. Virginia)). Make sure the application revision is in an Amazon S3 bucket associated with the same AWS Region as the instances.

On the event details page for the deployment, in the **Download bundle** row, choose **View logs**. Or 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 might appear in the output in plain text on the instance in the future.

For information about how to find the wire logging file and enable and disable wire logging, see `:log_aws_wire:` in CodeDeploy agent configuration reference.

Troubleshooting all lifecycle events skipped errors

If all lifecycle events of an EC2 or on-premises deployment are skipped, you might receive an error similar to `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. (Error code: HEALTH_CONSTRAINTS)`. Here are some possible causes and solutions:

- The CodeDeploy agent might not be installed or running on the instance. To determine if the CodeDeploy agent is running:
  - For Amazon Linux RHEL or Ubuntu server, run the following:

    ```
    systemctl status codedeploy-agent
    ```

  - For Windows, run the following:

    ```
    powershell.exe -Command Get-Service -Name CodeDeployagent
    ```

  If the CodeDeploy agent is not installed or running, see Verify the CodeDeploy agent is running.

  Your instance might not be able to reach the CodeDeploy or Amazon S3 public endpoint using port 443. Try one of the following:

  - Assign a public IP address to the instance and use its route table to allow internet access. Make sure the security group associated with the instance allows outbound access over port 443 (HTTPS). For more information, see Communication protocol and port for the CodeDeploy agent.
  - If an instance is provisioned in a private subnet, use a NAT gateway instead of an internet gateway in the route table. For more information, see NAT Gateways.
- The service role for CodeDeploy might not have required permissions. To configure a CodeDeploy service role, see Step 2: Create a service role for CodeDeploy.
- If you use an HTTP proxy, make sure it is specified in the `:proxy_uri:` setting in the CodeDeploy agent configuration file. For more information, see CodeDeploy agent configuration reference.
- The date and time signature of your deployment instance might not match the date and time signature of your deployment request. Look for an error similar to `Cannot reach InstanceService: Aws::CodeDeployCommand::Errors::InvalidSignatureException - Signature`

`expired` in your CodeDeploy agent log file. If you see this error, follow the steps in Troubleshooting "InvalidSignatureException – Signature expired: [time] is now earlier than [time]" deployment errors. For more information, see View log data for CodeDeploy EC2/On-Premises deployments.

- The CodeDeploy agent might stop running because an instance is running low on memory or hard disk space. Try to lower the number of archived deployments on your instance by updating the `max_revisions` setting in the CodeDeploy agent configuration. If you do this for an EC2 instance and the issue persists, consider using a larger instance. For example, if your instance type is `t2.small`, try using a `t2.medium`. For more information, see Files installed by the CodeDeploy agent, CodeDeploy agent configuration reference, and Instance types.
- The instance you're deploying to might not have an IAM instance profile attached, or it might have an IAM instance profile attached that does not have the required permissions.
  - If an IAM instance profile is not attached to your instance, create one with the required permissions and then attach it.
  - If an IAM instance profile is already attached to your instance, make sure it has the required permissions.

  After you confirm your attached instance profile is configured with the required permissions, restart your instance. For more information, see Step 4: Create an IAM instance profile for your Amazon EC2 instances and IAM Roles for Amazon EC2 in the *Amazon EC2 User Guide*.

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 allows or calls libraries that are offered only in a 64-bit version), the script might crash or not run as expected. This is because, by default, 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 might 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`

Troubleshoot Amazon ECS deployment issues

Topics

- A timeout occurs while waiting for replacement task set
- A timeout occurs while waiting for a notification to continue
- The IAM role does not have enough permissions
- The deployment timed out while waiting for a status callback
- The deployment failed because one or more of the lifecycle event validation functions failed

- The ELB could not be updated due to the following error: Primary taskset target group must be behind listener
- My deployment sometimes fails when using Auto Scaling
- Only ALB supports gradual traffic routing, use AllAtOnce Traffic routing instead when you create/update Deployment group
- Even though my deployment succeeded, the replacement task set fails the Elastic Load Balancing health checks, and my application is down
- Can I attach multiple load balancers to a deployment group?
- Can I perform CodeDeploy blue/green deployments without a load balancer?
- How can I update my Amazon ECS service with new information during a deployment?

A timeout occurs while waiting for replacement task set

Problem: You see the following error message while deploying your Amazon ECS application using CodeDeploy:

```
The deployment timed out while waiting for the replacement task set to
become healthy. This time out period is 60 minutes.
```

Possible cause: This error might occur if there is a mistake in your task definition file or other deployment-related files. For example, if there is a typo in the `image` field in your task definition file, Amazon ECS will try to pull the wrong container image and continuously fail, causing this error.

Possible fixes and next steps:

- Fix typographical errors and configuration problems in your task definition file and other files.
- Check the related Amazon ECS service event and find out why replacement tasks are not becoming healthy. For more information on Amazon ECS events, see Amazon ECS events in the *Amazon Elastic Container Service Developer Guide*.
- Check the Amazon ECS troubleshooting section in the *Amazon Elastic Container Service Developer Guide* for errors related to the messages in the event.

A timeout occurs while waiting for a notification to continue

Problem: You see the following error message while deploying your Amazon ECS application using CodeDeploy:

```
The deployment timed out while waiting for a notification to continue. This
time out period is n minutes.
```

Possible cause: This error might occur if you specified a wait time in the **Specify when to reroute traffic** field when you created your deployment group, but the deployment couldn't finish before the wait time expired.

Possible fixes and next steps:

- In your deployment group, set the **Specify when to reroute traffic** to a larger amount of time and redeploy. For more information, see Create a deployment group for an Amazon ECS deployment (console).
- In your deployment group, change **Specify when to reroute traffic** to **Reroute traffic immediately** and redeploy. For more information, see Create a deployment group for an Amazon ECS deployment (console).
- Redeploy and then run the `aws deploy continue-deployment` AWS CLI command with the `--deployment-wait-type` option set to `READY_WAIT`. Make sure to run this command *before* the time specified in **Specify when to reroute traffic** expires.

The IAM role does not have enough permissions

Problem: You see the following error message while deploying your Amazon ECS application using CodeDeploy:

```
The IAM role role-arn does not give you permission to perform operations in
the following AWS service: AWSLambda.
```

Possible cause: This error might occur if you specified a Lambda function in the AppSpec file's Hooks section, but you did not give CodeDeploy permission to the Lambda service.

Possible fix: Add the `lambda:InvokeFunction` permission to the CodeDeploy service role. To add this permission, add one of the following AWS-managed policies to the role: **`AWSCodeDeployRoleForECS`** or **`AWSCodeDeployRoleForECSLimited`**. For information about these policies and how to add them to the CodeDeploy service role, see Step 2: Create a service role for CodeDeploy.

The deployment timed out while waiting for a status callback

Problem: You see the following error message while deploying your Amazon ECS application using CodeDeploy:

```
The deployment timed out while waiting for a status callback. CodeDeploy
expects a status callback within one hour after a deployment hook is
invoked.
```

Possible cause: This error might occur if you specified a Lambda function in the AppSpec file's Hooks section, but Lambda function could not call the necessary `PutLifecycleEventHookExecutionStatus` API to return a `Succeeded` or `Failed` status to CodeDeploy.

Possible fixes and next steps:

- Add the `codedeploy:putlifecycleEventHookExecutionStatus` permission to the Lambda execution role used by the Lambda function that you specified in the AppSpec file. This permission grants the Lambda function the ability to return a status of `Succeeded` or `Failed` to CodeDeploy. For more information about the Lambda execution role, see Lambda execution role in the *AWS Lambda User Guide*.
- Check your Lambda function code and execution logs to make sure your Lambda function is calling CodeDeploy's `PutLifecycleEventHookExecutionStatus` API to inform CodeDeploy about whether the lifecycle validation test `Succeeded` or `Failed`. For information about the `putlifecycleEventHookExecutionStatus` API, see PutLifecycleEventHookExecutionStatus in the *AWS CodeDeploy API Reference*. For information about Lambda execution logs, see Accessing Amazon CloudWatch logs for AWS Lambda.

The deployment failed because one or more of the lifecycle event validation functions failed

Problem: You see the following error message while deploying your Amazon ECS application using CodeDeploy:

```
The deployment failed because one or more of the lifecycle event validation
functions failed.
```

Possible cause: This error might occur if you specified a Lambda function in the AppSpec file's Hooks section, but the Lambda function returned `Failed` to CodeDeploy when it called `PutLifecycleEventHookExecutionStatus`. This failure indicates to CodeDeploy that the lifecycle validation test failed.

Possible next step: Check your Lambda execution logs to see why the validation test code is failing. For information about Lambda execution logs, see Accessing Amazon CloudWatch logs for AWS Lambda.

The ELB could not be updated due to the following error: Primary taskset target group must be behind listener

Problem: You see the following error message while deploying your Amazon ECS application using CodeDeploy:

```
The ELB could not be updated due to the following error: Primary taskset
target group must be behind listener
```

Possible cause: This error might occur if you have configured an optional test listener, and it is configured with wrong target group. For more information about the test listener in CodeDeploy, see Before you begin an Amazon ECS deployment and What happens during an Amazon ECS deployment. For more information about task sets, see TaskSet in the *Amazon Elastic Container Service API Reference* and describe-task-set in the Amazon ECS section of the *AWS CLI Command Reference*.

Possible fix: Make sure that the Elastic Load Balancing's production listener and test listener are both pointing to the target group that's currently serving your workloads. There are three places to check:

- In Amazon EC2, in your load balancer's **Listeners and rules** settings. For more information, see Listeners for your Application Load Balancers in the *User Guide for Application Load Balancers*, or Listeners for your Network Load Balancers in the *User Guide for Network Load Balancers*.
- In Amazon ECS, in your cluster, under your service's **Networking** configuration. For more information, see Application Load Balancer and Network Load Balancer considerations in the *Amazon Elastic Container Service Developer Guide*.
- In CodeDeploy, in your deployment group settings. For more information, see Create a deployment group for an Amazon ECS deployment (console).

My deployment sometimes fails when using Auto Scaling

Problem: You are using Auto Scaling with CodeDeploy and you notice that your deployments occasionally fail. For more information about the symptoms of this problem, see the topic that reads For services configured to use service auto scaling and the blue/green deployment type, auto scaling is not blocked during a deployment but the deployment may fail under some circumstances in the *Amazon Elastic Container Service Developer Guide*.

Possible cause: This problem might occur if CodeDeploy and Auto Scaling processes conflict.

Possible fix: Suspend and resume Auto Scaling processes during the CodeDeploy deployment using the `RegisterScalableTarget` API (or the corresponding `register-scalable-target` AWS CLI command). For more information, see Suspend and resume scaling for Application Auto Scaling in the *Application Auto Scaling User Guide*.

> **Note**
>
> CodeDeploy can't call `RegisterScaleableTarget` directly. To use this API, you must configure CodeDeploy to send a notification or event to Amazon Simple Notification Service (or Amazon CloudWatch). You must then configure Amazon SNS (or CloudWatch) to call a Lambda function, and configure the Lambda function to call the `RegisterScalableTarget` API. The `RegisterScalableTarget` API must be called with the `SuspendedState` parameter set to `true` to suspend Auto Scaling operations, and `false` to resume them.
>
> The notification or event that CodeDeploy sends out must occur when a deployment starts (to trigger Auto Scaling suspend operations), or when a deployment succeeds, fails, or stops (to trigger Auto Scaling resume operations).
>
> For information about how to configure CodeDeploy to generate Amazon SNS notifications or CloudWatch events, see Monitoring deployments with Amazon CloudWatch Events. and Monitoring Deployments with Amazon SNS Event Notifications.

Only ALB supports gradual traffic routing, use AllAtOnce Traffic routing instead when you create/update Deployment group

Problem: You see the following error message while creating or updating a deployment group in CodeDeploy:

```
Only ALB supports gradual traffic routing, use AllAtOnce Traffic routing
instead when you create/update Deployment group.
```

Possible cause: This error might occur if you're using a Network Load Balancer and tried to use a predefined deployment configuration other than `CodeDeployDefault.ECSAllAtOnce`.

Possible fixes:

- Change your predefined deployment configuration to `CodeDeployDefault.ECSAllAtOnce`. This is the only predefined deployment configuration supported by Network Load Balancers.

  For more information about predefined deployment configurations, see Predefined deployment configurations for an Amazon ECS compute platform.
- Change your load balancer to an Application Load Balancer. Application Load Balancer's support all the predefined deployment configurations. For more information about creating a Application Load Balancer, see Set up a load balancer, target groups, and listeners for CodeDeploy Amazon ECS deployments.

Even though my deployment succeeded, the replacement task set fails the Elastic Load Balancing health checks, and my application is down

Problem: Even though CodeDeploy indicates that my deployment succeeded, the replacement task set fails the health checks from Elastic Load Balancing, and my application is down.

Possible cause: This issue might occur if you performed a CodeDeploy all-at-once deployment, and your replacement (green) task set contains bad code that is causing the Elastic Load Balancing health checks to fail. With the all-at-once deployment configuration, the load balancer's health checks start running on the replacement task set *after* traffic has been shifted to it (that is, *after* CodeDeploy's `AllowTraffic` lifecycle event occurs). That's why you will see health checks failing on the replacement task set after traffic has shifted, but not before. For information about the lifecycle events that CodeDeploy generates, see What happens during an Amazon ECS deployment.

Possible fixes:

- Change your deployment configuration from all-at-once to canary or linear. In a canary or linear configuration, the load balancer's health checks start running on the replacement task set while CodeDeploy installs your application in the replacement environment, and *before* traffic is shifted (that is, during the `Install` lifecycle event, and *before* the `AllowTraffic` event). By allowing

the checks to run during the application installation but before traffic is shifted, bad application code will be detected and cause deployment failures before the application becomes publicly available.

For information about how to configure canary or linear deployments, see Change deployment group settings with CodeDeploy.

For information about CodeDeploy lifecycle events that run during an Amazon ECS deployment, see What happens during an Amazon ECS deployment.

> **Note**
>
> Canary and linear deployment configurations are only supported with Application Load Balancers.

- If you want to keep your all-at-once deployment configuration, set up a test listener and check the health status of the replacement task set with the `BeforeAllowTraffic` lifecycle hook. For more information, see List of lifecycle event hooks for an Amazon ECS deployment.

Can I attach multiple load balancers to a deployment group?

No. If you want to use multiple Application Load Balancers or Network Load Balancers, use Amazon ECS rolling updates instead of CodeDeploy blue/green deployments. For more information about rolling updates, see Rolling update in the *Amazon Elastic Container Service Developer Guide*. For more information about using multiple load balancers with Amazon ECS, see Registering multiple target groups with a service in the *Amazon Elastic Container Service Developer Guide*.

Can I perform CodeDeploy blue/green deployments without a load balancer?

No, you cannot perform CodeDeploy blue/green deployments without a load balancer. If you are unable to use a load balancer, use Amazon ECS's rolling updates feature instead. For more information about Amazon ECS rolling updates, see Rolling update in the *Amazon Elastic Container Service Developer Guide*.

How can I update my Amazon ECS service with new information during a deployment?

To have CodeDeploy update your Amazon ECS service with a new parameter while it conducts a deployment, specify the parameter in the `resources` section of the AppSpec file. Only a few Amazon ECS parameters are supported by CodeDeploy, such as the task definition file and container name parameters. For a full list of Amazon ECS parameters that CodeDeploy can update, see AppSpec 'resources' section for Amazon ECS deployments.

> **Note**
>
> If you need to update your Amazon ECS service with a parameter that is not supported by CodeDeploy, complete these tasks:
>
> 1. Call Amazon ECS's `UpdateService` API with the parameter you want to update. For a full list of parameters that can be updated, see UpdateService in the *Amazon Elastic Container Service API Reference*.
> 2. To apply the change to the tasks, create a new Amazon ECS blue/green deployment. For more information, see Create an Amazon ECS Compute Platform deployment (console).

Troubleshoot AWS Lambda deployment issues

Topics

- AWS Lambda deployments fail after manually stopping a Lambda deployment that does not have configured rollbacks

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 shift traffic 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 CodeDeploy deployment.
4. From **Aliases**, choose the alias used in your CodeDeploy deployment, and then choose **Edit**.
5. From **Weighted alias**, 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 **Version**.
6. Choose **Save**.
7. Open the 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

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 CodeDeploy to enable automatic deployments to new EC2 instances in Amazon EC2 Auto Scaling groups. For more information, see Integrating CodeDeploy with Amazon EC2 Auto Scaling.

Troubleshoot instance issues

Topics

- Tags must be set correctly
- AWS CodeDeploy agent must be installed and running on instances
- Deployments do not fail for up to an hour when an instance is terminated during a deployment
- Analyzing log files to investigate deployment failures on instances
- Create a new CodeDeploy log file if it was accidentally deleted
- Troubleshooting “InvalidSignatureException – Signature expired: [time] is now earlier than [time]” deployment errors

Tags must be set correctly

Use the list-deployment-instances command to confirm the instances used for a deployment are tagged correctly. If an EC2 instance is missing in the output, use the 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*.

> **Note**
>
> If you tag an instance and immediately use 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 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 CodeDeploy agent is installed and running on an instance, see Verify the CodeDeploy agent is running.

To install, uninstall, or reinstall the CodeDeploy agent, see Install the CodeDeploy agent.

Deployments do not fail for up to an hour when an instance is terminated during a deployment

CodeDeploy provides a one-hour window for each deployment lifecycle event to run to completion. This provides ample time for long-running scripts.

If the scripts don't run to completion while a lifecycle event is in progress (for example, if an instance is terminated or the 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 CodeDeploy agent shuts down and cannot process more scripts.

If an instance is terminated between lifecycle events or before the first lifecycle event step starts, 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 CodeDeploy EC2/On-Premises deployments.

Create a new CodeDeploy log file if it was accidentally deleted

If you accidentally delete the deployment log file on an instance, 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:

```
systemctl stop codedeploy-agent
```

```
systemctl start 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

CodeDeploy requires accurate time references to perform its operations. If the date and time on your instance are not set correctly, they might not match the signature date of your deployment request, which CodeDeploy rejects.

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

CodeDeploy applications created after June 2017 use GitHub OAuth tokens for each AWS Region. The use of tokens tied to specific AWS Regions gives you more control over which 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. Remove the old token using one of the following methods:
   - To remove the old token using the API, use DeleteGitHubAccountToken.
   - To remove the old token using the AWS Command Line Interface:
     a. Go to the computer where the token resides.
     b. Make sure the AWS CLI is installed on this computer. For installation instructions, see Installing, updating, and uninstalling the AWS CLI in the *AWS Command Line Interface User Guide*
     c. Enter the following command on the computer where the token resides:

        ```
        aws delete-git-hub-account-token
        ```

        For details on the command syntax, see delete-git-hub-account-token.
2. Add a new OAuth token. For more information, see Integrating CodeDeploy with GitHub.

Maximum number of GitHub OAuth tokens exceeded

When you create a 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 CodeDeploy API to remove old tokens, see DeleteGitHubAccountToken.

Troubleshoot Amazon EC2 Auto Scaling issues

Topics

- General Amazon EC2 Auto Scaling troubleshooting
- "CodeDeployRole does not give you permission to perform operations in the following AWS service: AmazonAutoScaling" error
- Instances in an Amazon EC2 Auto Scaling group are continuously provisioned and terminated before a revision can be deployed
- Terminating or rebooting an Amazon EC2 Auto Scaling instance might cause deployments to fail
- Avoid associating multiple deployment groups with a single Amazon EC2 Auto Scaling group
- EC2 instances in an Amazon EC2 Auto Scaling group fail to launch and receive the error "Heartbeat Timeout"
- Mismatched Amazon EC2 Auto Scaling lifecycle hooks might cause automatic deployments to Amazon EC2 Auto Scaling groups to stop or fail
- "The deployment failed because no instances were found for your deployment group" error

General Amazon EC2 Auto Scaling troubleshooting

Deployments to EC2 instances in an Amazon EC2 Auto Scaling group can fail for the following reasons:

- **Amazon EC2 Auto Scaling continuously launches and terminates EC2 instances.** If CodeDeploy cannot automatically deploy your application revision, Amazon EC2 Auto Scaling continuously launches and terminates EC2 instances.

  Disassociate the Amazon EC2 Auto Scaling group from the CodeDeploy deployment group or change the configuration of your Amazon EC2 Auto Scaling group so that the desired number of instances matches the current number of instances (thus preventing Amazon EC2 Auto Scaling from launching any more EC2 instances). For more information, see Change deployment group settings with CodeDeploy or Manual Scaling for Amazon EC2 Auto Scaling.
- **The CodeDeploy agent is unresponsive.** The CodeDeploy agent might not be installed if initialization scripts (for example, cloud-init scripts) that run immediately after an EC2 instance is launched or started take more than one hour to run. CodeDeploy has a one-hour timeout for the CodeDeploy agent to respond to pending deployments. To address this issue, move your initialization scripts into your CodeDeploy application revision.

- **An EC2 instance in an Amazon EC2 Auto Scaling group reboots during a deployment.** Your deployment can fail if an EC2 instance is rebooted during a deployment or the CodeDeploy agent is shut down while processing a deployment command. For more information, see Terminating or rebooting an Amazon EC2 Auto Scaling instance might cause deployments to fail.
- **Multiple application revisions are deployed simultaneously to the same EC2 instance in an Amazon EC2 Auto Scaling group.** Deploying multiple application revisions to the same EC2 instance in an Amazon EC2 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 EC2 instances in an Amazon EC2 Auto Scaling group.
- **A deployment fails for new EC2 instances that are launched as part of an Amazon EC2 Auto Scaling group.** In this scenario, running the scripts in a deployment can prevent the launching of EC2 instances in the Amazon EC2 Auto Scaling group. (Other EC2 instances in the Amazon EC2 Auto Scaling group might appear to be running normally.) To address this issue, make sure that all other scripts are complete first:
  - **CodeDeploy agent is not included in your AMI** : If you use the **cfn-init** command to install the CodeDeploy agent while launching a new instance, place the agent installation script at the end of the `cfn-init` section of your CloudFormation template.
  - **CodeDeploy agent is included in your AMI** : Configure the AMI 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.

"CodeDeployRole does not give you permission to perform operations in the following AWS service: AmazonAutoScaling" error

Deployments that use an Auto Scaling group created with a launch template require the following permissions. These are in addition to the permissions granted by the `AWSCodeDeployRole` AWS managed policy.

- `EC2:RunInstances`
- `EC2:CreateTags`
- `iam:PassRole`

You might received this error if you are missing these permissions. For more information, see Tutorial: Use CodeDeploy to deploy an application to an Auto Scaling group, Creating a launch template for an Auto Scaling group, and Permissions in the *Amazon EC2 Auto Scaling User Guide*.

Instances in an Amazon EC2 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 Amazon EC2 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 Amazon EC2 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 Amazon EC2 Auto Scaling group health checks.
- An error in the application revision.

To work around this issue, follow these steps:

1. Manually create an EC2 instance that is not part of the Amazon EC2 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 prompts the Amazon EC2 Auto Scaling group to deploy the application revision to future instances in the Amazon EC2 Auto Scaling group.

> **Note**
>
> After you confirm that deployments are successful, delete the instance you created to avoid ongoing charges to your AWS account.

Terminating or rebooting an Amazon EC2 Auto Scaling instance might cause deployments to fail

If an EC2 instance is launched through Amazon EC2 Auto Scaling, and the instance is then terminated or rebooted, deployments to that instance might fail for the following reasons:

- During an in-progress deployment, a scale-in event or any other termination event causes the instance to detach from the Amazon EC2 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. CodeDeploy treats this as a timeout. The service fails all current and future deployments to the instance.

To address this issue:

- In general, make sure that all deployments are complete before the instance is terminated or rebooted. Make sure that all deployments start after the instance has started or been rebooted.
- Deployments can fail if you specify a Windows Server base Amazon Machine Image (AMI) for an Amazon EC2 Auto Scaling configuration and use the EC2Config service to set the computer name of the instance. To fix this issue, in the Windows Server base AMI, on the **General** tab of **EC2 Service Properties**, clear **Set Computer Name**. After you clear this check box, this behavior is disabled for all new Windows Server Amazon EC2 Auto Scaling instances launched with that Windows Server base AMI. For Windows Server Amazon EC2 Auto Scaling instances on which this behavior enabled, you do not need to clear this check box. Simply redeploy failed deployments to those instances after they have been rebooted.

Avoid associating multiple deployment groups with a single Amazon EC2 Auto Scaling group

As a best practice, you should associate only one deployment group with each Amazon EC2 Auto Scaling group.

This is because if Amazon EC2 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 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.

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, Amazon EC2 Auto Scaling immediately terminates the instance. When that first instance shuts down, the other running deployments start to fail. Because CodeDeploy has a one-hour timeout for the CodeDeploy agent to respond to pending deployments, it can take up to 60 minutes for each instance to time out.

For more information about Amazon EC2 Auto Scaling, see Under the hood: CodeDeploy and Auto Scaling integration.

EC2 instances in an Amazon EC2 Auto Scaling group fail to launch and receive the error "Heartbeat Timeout"

An Amazon EC2 Auto Scaling group might fail to launch new EC2 instances, generating a message similar to the following:

```
Launching a new EC2 instance <instance-Id>. Status Reason: Instance failed
to complete user's Lifecycle Action: Lifecycle Action with token<token-Id>
was abandoned: Heartbeat Timeout.
```

This 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 CodeDeploy quotas.
- The Auto Scaling group tried to launch too many EC2 instances too quickly. The API calls to RecordLifecycleActionHeartbeat or CompleteLifecycleAction for each new instance were throttled.
- An application in CodeDeploy was deleted before its associated deployment groups were updated or deleted.

When you delete an application or deployment group, CodeDeploy attempts to clean up any Amazon EC2 Auto Scaling hooks associated with it, but some hooks might remain. If you run a command to delete a deployment group, the leftover hooks are returned in the output. However, if you run a command to delete an application, the leftover hooks do 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 receive a “Heartbeat Timeout” error message, you can determine if leftover lifecycle hooks are the cause and resolve the problem by doing the following:

1. Do one of the following:

   - Call the delete-deployment-group command to delete the deployment group associated with the Auto Scaling group that is causing the heartbeat timeout.
   - Call the update-deployment-group command with a non-null empty list of Auto Scaling group names to detach all CodeDeploy-managed Auto Scaling lifecycle hooks.

     For example, enter the following AWS CLI command:

     ```
     aws deploy update-deployment-group --application-name my-example-app
     --current-deployment-group-name my-deployment-group --auto-scaling-
     groups
     ```

     As another example, if you are using the CodeDeploy API with Java, call `UpdateDeploymentGroup` and set `autoScalingGroups` to `new ArrayList<String>()`. This sets `autoScalingGroups` to an empty list and removes the existing list. Do not use `null`, which is the default, because this leaves `autoScalingGroups` as-is, which is not what you want.

   Examine the output of the call. If the output contains a `hooksNotCleanedUp` structure with a list of Amazon EC2 Auto Scaling lifecycle hooks, there are leftover lifecycle hooks.

2. Call the describe-lifecycle-hooks command, specifying the name of the Amazon EC2 Auto Scaling group associated with the EC2 instances that failed to launch. In the output, look for any of the following:

- Amazon EC2 Auto Scaling lifecycle hook names that correspond to the `hooksNotCleanedUp` structure you identified in step 1.
- Amazon EC2 Auto Scaling lifecycle hook names that contain the name of the deployment group associated with the Auto Scaling group that's failing.
- Amazon EC2 Auto Scaling lifecycle hook names that may have caused the heartbeat timeout for the CodeDeploy deployment.

3. If a hook falls into one of the categories listed in step 2, call the delete-lifecycle-hook command to delete it. Specify the Amazon EC2 Auto Scaling group and lifecycle hook in the call.

> **Important**
>
> Only delete hooks that are causing problems, as outlined in step 2. If you delete viable hooks, your deployments might fail, or CodeDeploy might not be able to deploy your application revisions to scaled out EC2 instances.

4. Call either the update-deployment-group or create-deployment-group command with the desired Auto Scaling group names. CodeDeploy re-installs the Auto Scaling hooks with new UUIDs.

> **Note**
>
> If you detach an Auto Scaling group from a CodeDeploy deployment group, any in-progress deployments to the Auto Scaling group may fail, and new EC2 instances that are scaled out by the Auto Scaling group will not receive your application revisions from CodeDeploy. To get Auto Scaling working again with CodeDeploy, you'll need to re-attach the Auto Scaling group to the deployment group and call a new `CreateDeployment` to start a fleet-wide deployment.

Mismatched Amazon EC2 Auto Scaling lifecycle hooks might cause automatic deployments to Amazon EC2 Auto Scaling groups to stop or fail

Amazon EC2 Auto Scaling and CodeDeploy use lifecycle hooks to determine which application revisions should be deployed to which EC2 instances after they are launched in Amazon EC2 Auto Scaling groups. Automatic deployments can stop or fail if lifecycle hooks and information about these hooks do not match exactly in Amazon EC2 Auto Scaling and CodeDeploy.

If deployments to an Amazon EC2 Auto Scaling group are failing, see if the lifecycle hook names in Amazon EC2 Auto Scaling and CodeDeploy match. If not, use these AWS CLI command calls.

First, get the list of lifecycle hook names for both the Amazon EC2 Auto Scaling group and the deployment group:

1. Call the describe-lifecycle-hooks command, specifying the name of the Amazon EC2 Auto Scaling group associated with the deployment group in 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 Amazon EC2 Auto Scaling group. In the output, in the `autoScalingGroups` list, find each item whose name value matches the Amazon EC2 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 Amazon EC2 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 Amazon EC2 Auto Scaling group. CodeDeploy creates new, replacement lifecycle hooks in the Amazon EC2 Auto Scaling group and associates the lifecycle hooks with the deployment

group. Automatic deployments should now resume as new instances are added to the Amazon EC2 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, do the following:

   a. Call the update-deployment-group command, but do not specify the name of the original Amazon EC2 Auto Scaling group.

   b. Call the **update-deployment-group** command again, but this time specify the name of the original Amazon EC2 Auto Scaling group. CodeDeploy re-creates the missing lifecycle hooks in the Amazon EC2 Auto Scaling group. Automatic deployments should now resume as new instances are added to the Amazon EC2 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 Amazon EC2 Auto Scaling group. Deployments do not occur automatically to instances that are already in the Amazon EC2 Auto Scaling group.

"The deployment failed because no instances were found for your deployment group" error

Read this section if you see the following CodeDeploy error:

```
The deployment failed because no instances were found for your deployment
group. Check your deployment group settings to make sure the tags for your
EC2 instances or Auto Scaling groups correctly identify the instances you
want to deploy to, and then try again.
```

Possible causes for this error are:

1. Your deployment group settings include tags for EC2 instances, on-premises instances, or Auto Scaling groups that are not correct. To fix this problem, check that your tags are correct, and then redeploy your application.
2. Your fleet scaled out after the deployment started. In this scenario, you see healthy instances in the `InService` state in your fleet, but you also see the error above. To fix this problem, redeploy your application.
3. Your Auto Scaling group does not include any instances that are in the `InService` state. In this scenario, when you try to do a fleet-wide deployment, the deployment fails with the error

message above because CodeDeploy needs at least one instance to be in the `InService` state. There are many reasons why you might have no instances in the `InService` state. A few of them include:

- You scheduled (or manually configured) the Auto Scaling group size to be `0`.
- Auto Scaling detected bad EC2 instances (for example, the EC2 instances had hardware failures), so canceled them all, leaving none in the `InService` state.
- During a scale out event from `0` to `1`, CodeDeploy deployed a previously successful revision (called a *last successful revision*) that had become unhealthy since the last deployment. This caused the deployment on the scaled-out instance to fail, which in turn caused Auto Scaling to cancel the instance, leaving no instances in the `InService` state.

  If you find that you have no instances in the `InService` state, troubleshoot the problem as described in the following procedure, To troubleshoot the error if there are no instances in the InService state.

To troubleshoot the error if there are no instances in the InService state

1. In the Amazon EC2 console, verify the **Desired Capacity** setting. If it is zero, set it to a positive number. Wait for the instance to be `InService`, which means the deployment succeeded. You have fixed the problem and can skip the remaining steps of this troubleshooting procedure. For information on setting the **Desired Capacity** setting, see Setting capacity limits on your Auto Scaling group in the *Amazon EC2 Auto Scaling User Guide*.
2. If Auto Scaling keeps attempting to launch new EC2 instances to meet the desired capacity but can never fulfill the scale out, it is usually due to a failing Auto Scaling lifecycle hook. Troubleshoot this problem as follows:

   a. To check which Auto Scaling lifecycle hook event is failing, see Verifying a scaling activity for an Auto Scaling group in the Amazon EC2 Auto Scaling User Guide.

   b. If the failing hook's name is `CodeDeploy-managed-automatic-launch-deployment-hook-`*`DEPLOYMENT_GROUP_NAME`*, go to CodeDeploy, find the deployment group, and find the failed deployment that was started by Auto Scaling. Then investigate why the deployment failed.

   c. If you understand why the deployment failed (for example, CloudWatch alarms were occurring) and you can fix the problem without changing the revision, then do so now.

   d. If, after investigation, you determine that CodeDeploy's *last successful revision* is no longer healthy, and there are zero healthy instances in your Auto Scaling group, you are in a

deployment deadlock scenario. To solve this issue, you must fix the bad CodeDeploy revision by temporarily removing CodeDeploy's lifecycle hook from the Auto Scaling group, and then reinstalling the hook and redeploying a new (good) revision. For instructions, see:

- To fix the deployment deadlock issue (CLI)
- To fix the deployment deadlock issue (console)

To fix the deployment deadlock issue (CLI)

1. (Optional) Block your CI/CD pipelines that are causing the CodeDeploy error so that unexpected deployments do not occur while you're fixing this problem.
2. Take note of your current Auto Scaling **DesiredCapacity** setting:

   ```
   aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name ASG_NAME
   ```

   You may need to scale back to this number at the end of this procedure.
3. Set your Auto Scaling **DesiredCapacity** setting to 1. This is optional if your desired capacity was greater than 1 to begin with. By decreasing it to 1, the instance will take less time to provision and deploy later, which speeds up troubleshooting. If your Auto Scaling desired capacity was originally set to `0`, you must increase it to 1. This is mandatory.

   aws autoscaling set-desired-capacity --auto-scaling-group-name *ASG_NAME* --desired-capacity 1

   > **Note**
   >
   > The remaining steps of this procedure assume you have set your **DesiredCapacity** to 1.

   At this point, Auto Scaling attempts to scale to one instance. Then, because the hook that CodeDeploy added is still present, CodeDeploy tries to deploy; the deployment fails; Auto Scaling cancels the instance; and Auto Scaling tries to re-launch an instance to reach desired capacity of one, which again fails. You are in a cancel-relaunch loop.
4. De-register the Auto Scaling group from the deployment group:

> **Warning**
>
> The following command will launch a new EC2 instance with no software on it. Before running the command, make sure that an Auto Scaling `InService` instance running no software is acceptable. For example, make sure the load balancer associated with the instance is not sending traffic to this host without software.

> **Important**
>
> Use the CodeDeploy command shown below to remove the hook. Do not remove the hook through the Auto Scaling service, because the removal will not be recognized by CodeDeploy.

```
aws deploy update-deployment-group --application-name APPLICATION_NAME
--current-deployment-group-name DEPLOYMENT_GROUP_NAME --auto-scaling-
groups
```

After running this command, the following occurs:

a. CodeDeploy de-registers the Auto Scaling group from the deployment group.

b. CodeDeploy removes the Auto Scaling lifecycle hook from the Auto Scaling group.

c. Since the hook that was causing a failed deployment is no longer present, Auto Scaling cancels the existing EC2 instance and immediately launches a new one to scale to the desired capacity. The new instance should soon move into `InService` state. The new instance does not include software.

5. Wait for the EC2 instance to enter the `InService` state. To verify its state, use the following command:

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
ASG_NAME --query AutoScalingGroups[0].Instances[*].LifecycleState
```

6. Add the hook back to the EC2 instance:

> **Important**
>
> Use the CodeDeploy command shown below to add the hook. Do not use the Auto Scaling service to add the hook, because the addition will not be recognized by CodeDeploy.

```
aws deploy update-deployment-group --application-name APPLICATION_NAME
--current-deployment-group-name DEPLOYMENT_GROUP_NAME --auto-scaling-
groups ASG_NAME
```

After running this command, the following occurs:

a. CodeDeploy re-installs the Auto Scaling lifecycle hook to the EC2 instance

b. CodeDeploy reregisters the Auto Scaling group with the deployment group.

7. Create a fleet-wide deployment with the Amazon S3 or GitHub revision that you know is healthy and want to use.

   For example, if the revision is a .zip file in an Amazon S3 bucket called `my-revision-bucket` with an object key of `httpd_app.zip`, enter the following command:

   ```
   aws deploy create-deployment --application-name APPLICATION_NAME
   --deployment-group-name DEPLOYMENT_GROUP_NAME --
   revision "revisionType=S3,s3Location={bucket=my-revision-
   bucket,bundleType=zip,key=httpd_app.zip}"
   ```

   Since there is now one `InService` instance in the Auto Scaling group, this deployment should work, and you should no longer see the error *The deployment failed because no instances were found for your deployment group*.

8. After the deployment succeeds, scale your Auto Scaling group back out to the original capacity, if you previously scaled it in:

   ```
   aws autoscaling set-desired-capacity --auto-scaling-group-name ASG_NAME
   --desired-capacity ORIGINAL_CAPACITY
   ```

To fix the deployment deadlock issue (console)

1. (Optional) Block your CI/CD pipelines that are causing the CodeDeploy error so that unexpected deployments do not occur while you're fixing this problem.
2. Go to the Amazon EC2 console, and take note of your Auto Scaling **Desired capacity** setting. You may need to scale back to this number at the end of this procedure. For information on finding this setting, see Setting capacity limits on your Auto Scaling group.
3. Set the desired number of EC2 instances to 1:

   This is optional if your desired capacity was greater than 1 to begin with. By decreasing it to 1, the instance will take less time to provision and deploy later, which speeds up troubleshooting. If your Auto Scaling **Desired capacity** was originally set to `0`, you must increase it to 1. This is mandatory.

   > **Note**
   >
   > The remaining steps of this procedure assume you have set your **Desired capacity** to 1.

   a. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/, and choose **Auto Scaling Groups** from the navigation pane.
   b. Choose the appropriate Region.
   c. Go to the problematic Auto Scaling group.
   d. In **Group details**, choose **Edit**.
   e. Set **Desired capacity** to **1**.
   f. Choose **Update**.
4. De-register the Auto Scaling group from the deployment group:

   > **Warning**
   >
   > The following sub-steps will launch a new EC2 instance with no software on it. Before running the command, make sure that an Auto Scaling `InService` instance running no software is acceptable. For example, make sure the load balancer associated with the instance is not sending traffic to this host without software.

a. Open the CodeDeploy console at https://console.aws.amazon.com/codedeploy/.

b. Choose the appropriate Region.

c. In the navigation pane, choose **Applications**.

d. Choose the name of your CodeDeploy application.

e. Choose the name of your CodeDeploy deployment group.

f. Choose **Edit**.

g. In **Environment configuration**, deselect **Amazon EC2 Auto Scaling groups**.

> **Note**
>
> The console does not allow you to save the configuration if there is no environment configuration defined. To bypass the check, temporarily add a tag of `EC2` or `On-premises` that you know won't resolve to any hosts. To add a tag, select **Amazon EC2 instances** or **On-premises instance**, and add a tag **Key** of **EC2** or **`On-premises`**. You can leave the tag **Value** empty.

h. Choose **Save changes**.

After completing these sub-steps, the following occurs:

i. CodeDeploy de-registers the Auto Scaling group from the deployment group.

ii. CodeDeploy removes the Auto Scaling lifecycle hook from the Auto Scaling group.

iii. Since the hook that was causing a failed deployment is no longer present, Auto Scaling cancels the existing EC2 instance and immediately launches a new one to scale to the desired capacity. The new instance should soon move into `InService` state. The new instance does not include software.

5. Wait for the EC2 instance to enter the `InService` state. To verify its state:

a. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

b. In the navigation pane, choose **Auto Scaling Groups**.

c. Choose your Auto Scaling group.

d. In the content pane, choose the **Instance Management** tab.

e. Under **Instances**, make sure that the **Lifecycle** column indicates **InService** next to the instance.

6. Re-register the Auto Scaling group with the CodeDeploy deployment group using the same method you used to remove it:

   a. Open the CodeDeploy console at https://console.aws.amazon.com/codedeploy/.
   b. Choose the appropriate Region.
   c. In the navigation pane, choose **Applications**.
   d. Choose the name of your CodeDeploy application.
   e. Choose the name of your CodeDeploy deployment group.
   f. Choose **Edit**.
   g. In **Environment configuration**, select **Amazon EC2 Auto Scaling groups** and select your Auto Scaling group from the list.
   h. Under **Amazon EC2 instances** or **On-premises instances**, find the tag you added and remove it.
   i. Deselect the check box next to **Amazon EC2 instances** or **On-premises instances**.
   j. Choose **Save changes**.

   This configuration re-installs the lifecycle hook into the Auto Scaling group.

7. Create a fleet-wide deployment with the Amazon S3 or GitHub revision that you know is healthy and want to use.

   For example, if the revision is a .zip file in an Amazon S3 bucket called `my-revision-bucket` with an object key of `httpd_app.zip`, do the following:

   a. In the CodeDeploy console, in the **Deployment Group** page, choose **Create deployment**.
   b. For **Revision type**, choose **My application is stored in Amazon S3**.
   c. For **Revision location**, choose `s3://my-revision-bucket/httpd_app.zip`.
   d. For **Revision file type**, choose `.zip`.
   e. Choose **Create deployment**.

   Since there is now one `InService` instance in the Auto Scaling group, this deployment should work, and you should no longer see the error *The deployment failed because no instances were found for your deployment group*.

8. After the deployment succeeds, scale your Auto Scaling group back out to the original capacity, if you previously scaled it in.

a. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/, and choose **Auto Scaling Groups** from the navigation pane.

b. Choose the appropriate Region.

c. Go to your Auto Scaling group.

d. In **Group details**, choose **Edit**.

e. Set **Desired capacity** back to its original value.

f. Choose **Update**.

Error codes for AWS CodeDeploy

This topic provides reference information about CodeDeploy errors.

Error Code	Description
`AGENT_ISSUE`	The deployment failed because of a problem with the CodeDeploy agent. Make sure the agent is installed and running on all instances in this deployment group. Learn more: • Verify the CodeDeploy agent is running • Install the CodeDeploy agent • Working with the CodeDeploy agent
`AUTO_SCALING_IAM_ROLE_PERMISSIONS`	The service role associated with your deployment group does not have the permission required to perform operations in the following AWS service. Learn more: • Step 2: Create a service role for CodeDeploy • Creating a Role to Delegate Permissions to an AWS Service

Error Code	Description
HEALTH_CONSTRAINTS	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. Learn more: • Instance Health • Troubleshoot instance issues • Troubleshoot EC2/On-Premises deployment issues
HEALTH_CONSTRAINTS_INVALID	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. Learn more: • Instance Health • Working with instances for CodeDeploy
IAM_ROLE_MISSING	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. Learn more: • Step 2: Create a service role for CodeDeploy • Change deployment group settings with CodeDeploy

Error Code	Description
`IAM_ROLE_PERMISSIONS`	CodeDeploy does not have the permissions required to assume a role, or the IAM role you're using does't give you permission to perform operations in an AWS service. Learn more: • Step 1: Setting up • Step 2: Create a service role for CodeDeploy • Step 4: Create an IAM instance profile for your Amazon EC2 instances

Error Code	Description
NO_INSTANCES	This might be caused by one of the following. • For an EC2/On-Premises blue/green deployment, if you use Amazon EC2 tags they might not be configured properly. In your CodeDeploy deployment group, make sure they are included in your blue instances and your green instances. You can confirm your instances are tagged properly using the Amazon EC2 console. • If you use an Amazon EC2 Auto Scaling group, your Auto Scaling group might not have enough capacity. Make sure your Auto Scaling group has enough capacity for your deployment. You can view the capacity of your Amazon EC2 Auto Scaling group by looking at its number of healthy instances using the Amazon EC2 console. • For an EC2/On-Premises blue/green deployment, the blue and green fleets might not be the same size. Make sure both fleets are the same size. Learn more: • Tagging Instances for Deployments • Tutorial: Use CodeDeploy to deploy an application to an Auto Scaling group • Create an application for a blue/green deployment (console)

Error Code	Description
`OVER_MAX_INSTANCES`	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. Learn more: • Change deployment group settings with CodeDeploy • CodeDeploy quotas • Request a Limit Increase
`THROTTLED`	The deployment failed because more requests were made than are permitted for AWS CodeDeploy by an IAM role. Try reducing the number of requests. Learn more: • Query API Request Rate

Error Code	Description
UNABLE_TO_SEND_ASG	The deployment failed because the deployment group isn't configured correctly with its Amazon EC2 Auto Scaling group. In the CodeDeploy console, delete the Amazon EC2 Auto Scaling group from the deployment group, and then add it again. Learn more: • Under the Hood: CodeDeploy and Auto Scaling Integration

Related topics

Troubleshooting CodeDeploy

CodeDeploy resources

The following related resources can help you as you work with CodeDeploy.

Reference guides and support resources

- AWS CodeDeploy API Reference — Descriptions, syntax, and usage examples about CodeDeploy actions and data types, including common parameters and error codes.
- CodeDeploy technical FAQs — Top questions from customers about CodeDeploy.
- 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

- CodeDeploy samples on GitHub — Samples and template scenarios for CodeDeploy.
- CodeDeploy Jenkins plugin — Jenkins plugin for CodeDeploy.
- CodeDeploy agent — Open-source version of the 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 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.
- CodeDeploy cmdlets in the AWS Tools for PowerShell — A set of Windows PowerShell cmdlets that expose the functionality of 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 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 *CodeDeploy User Guide*.

- **API version:** 2014-10-06

Change	Description	Date
CodeDeploy agent v1.8.1 release	The AWS CodeDeploy agent was updated to version 1.8.1. For more information, see Version history of the CodeDeploy agent.	February 3, 2026
CodeDeploy agent v1.8.0 release	The AWS CodeDeploy agent was updated to version 1.8.0. For more information, see Version history of the CodeDeploy agent.	July 31, 2025
CodeDeploy updated an existing AWS managed policy	The `AWSCodeDeployDeployerAccess` was updated. For more information, see AWS managed policy updates.	December 16, 2024
CodeDeploy updated an existing AWS managed policy	The `AWSCodeDeployReadOnlyAccess` was updated. For more information, see AWS managed policy updates.	December 16, 2024
CodeDeploy updated an existing AWS managed policy	The `AWSCodeDeployFullAccess` was updated. For more information, see AWS managed policy updates.	December 16, 2024

CodeDeploy agent v1.7.1 release	The AWS CodeDeploy agent was updated to version 1.7.1. For more information, see Version history of the CodeDeploy agent.	November 14, 2024
Updated Amazon S3 bucket names	Updated the example Amazon S3 buckets in this guide to use names reserved by AWS.	June 17, 2024
Added alternative text (alt text)	Updated all images in this guide to include alt text. Alt text is read by screen readers and makes our documenta tion more accessible to blind users.	May 22, 2024
CodeDeploy agent v1.7.0 release	The AWS CodeDeploy agent was updated to version 1.7.0. For more information, see Version history of the CodeDeploy agent.	March 6, 2024

Changed commands	The `sudo service codedeploy-agent` *`status|start|stop`* commands are no longer recommended because they do not use `systemd` for CodeDeploy agent process management, which is a best practice. To ensure `systemd` is used, use `systemctl` commands, as shown in the following example: `sytemctl start codedeploy-agent`. The following topics were updated with `systemctl` commands: Install the CodeDeploy agent for Amazon Linux or RHEL, Install the CodeDeploy agent for Ubuntu Server, Troubleshooting all lifecycle events skipped errors, and Create a new CodeDeploy log file if it was accidentally deleted.	January 12, 2024
Added topics	Added the Managing the CodeDeploy agent process and Referencing files in your lifecycle event scripts topics.	January 12, 2024
CodeDeploy now supports zonal configurations	Updated the Create a deployment configuration with CodeDeploy topic with zonal configuration information.	December 7, 2023

CodeDeploy now supports termination deployments	Added the Enabling termination deployments during Auto Scaling scale-in events topic to described the termination deployment feature. Also updated the AppSpec 'hooks' section for an EC2/On-Premises deployment, Create a deployment group for an in-place deployment (console), and Create a deployment group for an EC2/On-Premises blue/green deployment (console) topics to account for the feature.	December 7, 2023
Fixed JSON formatting	Fixed the formatting of the JSON code samples in the AppSpec 'resources' section (Amazon ECS and AWS Lambda deployments only) topic.	December 3, 2023
Added a troubleshooting topic	Added a Troubleshooting Amazon ECS deployment issues topic.	October 24, 2023
Updated the AppSpec file name	Updated the CodeDeploy AppSpec file reference to indicate that the AppSpec file must be named `appspec.yml` for EC2/On-Premises deployments.	October 5, 2023

CodeDeploy now supports multiple load balancers	Updated the Create a deployment group for an in-place deployment (console) , Create a deployment group for an EC2/On-Premises blue/green deployment (console) , and Set up a load balancer in Elastic Load Balancing for CodeDeploy Amazon EC2 deployments topics to indicate support for multiple load balancers.	September 26, 2023
Updated the Regions in the VPC topic	Updated the table in the Use CodeDeploy with Amazon Virtual Private Cloud topic to show additional Region support. Specifically, the Asia Pacific (Hyderabad), Asia Pacific (Melbourne), Europe (Milan), Europe (Spain), and Europe (Zurich) Regions were updated to show support for the agent endpoint.	September 22, 2023
Updated Regions in the resource kit topic	Added the following AWS Regions to the Resource kit bucket names by Region section: Asia Pacific (Osaka), Asia Pacific (Hyderabad), Canada (Central), Europe (Spain), Europe (Zurich), Middle East (UAE). Also updated IAM policies with these Regions and any others that were missing.	September 22, 2023

Shortened agent installation and update topics	Shortened the Install the CodeDeploy agent for Windows Server and Update the CodeDeploy agent on Windows Server topics. Removed redundant Amazon S3 bucket URLs and Amazon S3 copy commands.	September 21, 2023
Added the Asia Pacific (Jakarta) Region	Added Asia Pacific (Jakarta) to Resource kit bucket names by Region.	September 21, 2023
CodeDeploy updated an existing AWS managed policy	The AWSCodeDeployRole managed policy was updated. For more information, see AWS updates to AWS managed policies.	August 16, 2023
Added a limit	Added a limit to the CodeDeploy limits topic. The limit is *Maximum number of alarms associated with a deployment group*.	August 15, 2023
Fixed a step related to load balancers	Fixed the instructions in Create a deployment group for an EC2/On-Premises blue/green deployment (console). The load balancer step is now marked as optional.	August 3, 2023

Clarified wording in the Amazon ECS topic	Clarified the wording in Tutorial: Deploy an application into Amazon ECS. The wording now indicates you are deploying an application. Previously, the wording indicated you were deploying an Amazon ECS service.	August 3, 2023
CodeDeploy is now available in Israel (Tel Aviv) Region	CodeDeploy is now available in Israel (Tel Aviv) Region (il-central-1). Several topics, including those containing instructions for setting up the CodeDeploy agent, were updated to reflect the availability of this new Region.	July 31, 2023
Topic update	Updated the Troubleshoot EC2/On-Premises deployment issues topic with a tip about using a runbook to perform troubleshooting tasks automatically.	July 7, 2023
Topic update	Updated the AppSpec 'resources' section for Amazon ECS deployments topic with more information about the task definition ARN.	July 7, 2023
Topic update	Updated the Step 1: Install and configure the AWS CLI on the on-premises instance topic with troubleshooting information.	July 7, 2023

Topic update	Updated the Cross-service confused deputy prevention topic with information about Amazon ECS blue/green deployments through CloudFormation.	July 6, 2023
Topic update	The Cross-service confused deputy prevention topic was updated with information about Amazon ECS blue/green deployments through CloudFormation.	July 6, 2023
Topic update	Updated the Predefined deployment configurations for an EC2/On-Premises compute platform topic. Added a note about the `CodeDeployDefault.HalfAtATime` predefined deployment configuration's behavior with Auto Scaling groups.	June 29, 2023
Topic update	Updated the Infrastructure security in AWS CodeDeploy topic to indicate the new minimum and recommended versions of the Transport Layer Security (TLS) protocol.	June 28, 2023

Limit update	The following limit was changed: '*Maximum number of hours an EC2/On-Premises in-place deployment can run*'. For more information, see the Limits	June 27, 2023
Topic update	The Step 3: Limit the CodeDeploy user's permissions topic was updated with detailed instructions.	May 31, 2023
CodeDeploy updated an existing AWS managed policy	The AWSCodeDeployFullAccess managed policy was updated. For more information, see AWS updates to AWS managed policies.	May 16, 2023
CodeDeploy agent v1.6.0 release	The AWS CodeDeploy agent was updated to version 1.6.0. For more information, see Version history of the CodeDeploy agent.	March 30, 2023
CodeDeploy agent v1.5.0 release	The AWS CodeDeploy agent was updated to version 1.5.0. For more information, see Version history of the CodeDeploy agent.	March 3, 2023
Amazon ECS compute platform update	Deployments on an Amazon ECS compute platform are now supported in the Asia Pacific (Jakarta) Region.	February 8, 2023

CodeDeploy updated an existing AWS managed policy	The AWSCodeDeployRole managed policy was updated. For more information, see AWS updates to AWS managed policies.	February 3, 2023
Topic update	The Use CodeDeploy with Amazon Virtual Private Cloud topic was updated with new and changed AWS Regions.	February 2, 2023
Topic updates	CodeDeploy is now available in the Asia Pacific (Melbourne) Region (ap-southeast-4). Several topics, including those containing instructions for setting up the CodeDeploy agent, were updated to reflect the availability of these new regions.	January 26, 2023
Security best practices updates	The Getting started with CodeDeploy section and a few other sections were updated to conform to AWS security best practices.	January 23, 2023
CodeDeploy agent v1.4.1 release	The AWS CodeDeploy agent was updated to version 1.4.1. For more information, see Version history of the CodeDeploy agent.	December 6, 2022

Added a troubleshooting topic	Added a topic on how to troubleshoot errors caused by long file paths used with the CodeDeploy agent for Windows. For more informati on, see Long file paths cause "No such file or directory" errors.	December 6, 2022
Changed a limit	The following limit was changed: '*Maximum number of custom deployment configurations associated with an AWS account*'. The limit is now 200. For more informati on on limits, see the Limits topic.	September 7, 2022
CodeDeploy agent v1.4.0 release	The AWS CodeDeploy agent was updated to version 1.4.0. For more information, see Version history of the CodeDeploy agent.	August 31, 2022

Fixed a few limits.	The following limits were fixed: '*Maximum number of concurrent deployments associated with an AWS account*' is now 1000. '*Maximum number of instances in a single deployment*' is now 1000. '*Maximum number of instances that can be used by concurrent deployments that are in progress and associated with one account*' is now 1000. '*Maximum number of custom deployment configurations associated with an AWS account*' is now 100. For more information on limits, see the Limits topic.	August 8, 2022
Added a table that shows the CodeDeploy endpoints supported in each Region.	For more information, see Use CodeDeploy with Amazon Virtual Private Cloud.	April 20, 2022
Added a new limit for Amazon ECS blue/green deployments.	The maximum number of hours between the deployment of a revision and the shifting of traffic to the replacement environment during an Amazon ECS blue/green deployment is now 120. For more information, see Deployments in the Limits topic.	April 12, 2022

Added a topic on how to prevent the confused deputy problem	For more information, see AWS Identity and Access Management for AWS CodeDeploy.	March 14, 2022
CodeDeploy updated an existing AWS managed policy	The `AmazonEC2RoleforAWSCodeDeployLimited` role was updated. For more information, see AWS managed policy updates.	November 22, 2021
CodeDeploy updated an existing AWS managed policy	The `AWSCodeDeployRole` was updated. For more information, see AWS managed policy updates.	May 18, 2021
CodeDeploy agent v1.3.2 release	The AWS CodeDeploy agent was updated to version 1.3.2. For more information, see Version History of the CodeDeploy Agent.	May 6, 2021
CodeDeploy supports updating outdated Amazon EC2 instances	CodeDeploy now supports updating outdated Amazon EC2 instances automatically. For more information, see Configure advanced options for a deployment group.	February 23, 2021
CodeDeploy agent v1.3.1 release	The AWS CodeDeploy agent was updated to version 1.3.1. For more information, see Version History of the CodeDeploy Agent.	December 22, 2020

CodeDeploy agent v1.3.0 release	The AWS CodeDeploy agent was updated to version 1.3.0. For more information, see Version History of the CodeDeploy Agent.	November 10, 2020
CodeDeploy agent v1.2.1 release	The AWS CodeDeploy agent was updated to version 1.2.1. For more information, see Version History of the CodeDeploy Agent.	September 23, 2020
CodeDeploy supports Amazon VPC endpoints powered by AWS PrivateLink	If you use Amazon Virtual Private Cloud (Amazon VPC) to host your AWS resources , you can establish a private connection between your VPC and CodeDeploy. You can use this connection to enable CodeDeploy to communica te with your resources on your VPC without going through the public internet. For more information, see Use CodeDeploy with Amazon Virtual Private Cloud.	August 11, 2020
Updated CodeDeploy service limits	Updated the limit for both number of applications per account and number of deployment groups per application to 1000. For more information on CodeDeploy service limits, see CodeDeploy limits.	August 6, 2020

CodeDeploy agent v1.1.2 release	The AWS CodeDeploy agent was updated to version 1.1.2. For more information, see Version History of the CodeDeploy Agent.	August 4, 2020
CodeDeploy agent 1.1.0 release and integration with Amazon EC2 Systems Manager	Version 1.1.0 of the CodeDeploy agent is now available, for more informati on, see Version history of the CodeDeploy agent. You can now use Amazon EC2 Systems Manager to automatically manage your CodeDeplo y agent installation and updates on your Amazon EC2 or on-premises instances. For more information, see Install the CodeDeploy agent using Amazon EC2 Systems Manager.	June 30, 2020
CodeDeploy supports managing Amazon ECS blue/green deployments with CloudFormation	You can now use CloudForm ation to manage Amazon ECS blue/green deploymen ts through CodeDeploy. You generate your deploymen t by defining your green and blue resources and specifying the traffic routing and stabilization settings to use in CloudFormation. For more information, see Create an Amazon ECS blue/green deployment through CloudFormation.	May 19, 2020

CodeDeploy supports weighted traffic shifting for Amazon ECS blue/green deployments	CodeDeploy now supports weighted traffic shifting for Amazon ECS blue/green deployments. 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. The following topic has been updated to reflect this change: Deployment configurations on an Amazon ECS compute platform.	February 6, 2020
Updated security, authentication, and access control topics	The security, authentication, and access control information for CodeDeploy has been organized into a new Security chapter. For more information, see Security.	November 26, 2019
CodeDeploy supports notification rules	You can now use notification rules to notify users of important changes in deployments. For more information, see Create a notification rule.	November 5, 2019

Updated topics	CodeDeploy is now available in the Asia Pacific (Hong Kong) Region (ap-east-1) region. Several topics, including those containin g instructions for setting up the CodeDeploy agent, were updated to reflect the availability of this new region. You must explicitly enable access to this region. For more informaiton, see Managing AWS Regions.	April 25, 2019

Updated topics	AWS CodeDeploy now supports blue/green deployments of a containerized application in an Amazon ECS service. A CodeDeploy application that uses the new Amazon ECS compute platform deploys a containerized application to a new, replacement task set in the same Amazon ECS service. Several topics have been added and updated to reflect this change, including Overview of AWS CodeDeploy compute platforms, Deployments on an Amazon ECS compute platform, AppSpec file structure for Amazon ECS deployments, and Create an application for an Amazon ECS service deployment (Console).	November 27, 2018
Updated CodeDeploy agent	The AWS CodeDeploy agent was updated to version 1.0.1.1597. For more information, see Version history of the CodeDeploy agent.	November 15, 2018
Updated console	Procedures in this guide were updated to match the new design of the CodeDeploy console.	October 30, 2018

New minimum supported version of the CodeDeploy agent	The minimum supported version of the AWS CodeDeploy agent is now 1.8.x. For more information, see Version history of the 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	CodeDeploy is now available in the Europe (Paris) Region (eu-west-3) region. Several topics, including those containing instructions for setting up the CodeDeploy agent, were updated to reflect the availability of this new region.	December 19, 2017
Updated topics	CodeDeploy is now available in the China (Ningxia) Region. To use services in the China (Beijing) Region or China (Ningxia) Region, you must have an account and credentia ls 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 CodeDeploy Resource Kit bucket names and CodeDeploy agent installation procedures, are not included in this edition of the *CodeDeploy User Guide*. For more information:	December 11, 2017

Change	Description	Date Changed
	• CodeDeploy in *Getting Started with AWS in the China (Beijing) Region* • *CodeDeploy User Guide for the China Regions* (English version | Chinese version)	
Updated topics	CodeDeploy now supports the deployment of a Lambda function. An AWS Lambda deployment shifts 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 Serverles s Application Model (AWS SAM) so that you can use an AWS SAM deployment preference to manage the way in which traffic is shifted durng an AWS Lambda deploymen t. Several topics have been added and updated to reflect this change, including Overview of CodeDeploy compute platforms, Deployments on an AWS Lambda Compute Platform, Create an AWS Lambda Compute Platform deployment (console), Create an application for an AWS Lambda function deployment (console), and Add an AppSpec file for an AWS Lambda deployment.	November 28, 2017
New topic	CodeDeploy now supports deployments directly to a local machine or instance where the CodeDeploy agent is installed. You can locally test a deployment and, if it has errors, use 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 CodeDeploy agent to validate a deployment package on a local machine.	November 16, 2017

Change	Description	Date Changed
Updated topics	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 CodeDeploy with Elastic Load Balancing, Create an application for an in-place deployment (console), Deployment prerequisites, Integrating CodeDeploy with Elastic Load Balancing, and Create an application for an in-place deployment (console).	September 12, 2017
Updated topics	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 CodeDeploy with Elastic Load Balancing, Create an application with CodeDeploy, and Create a deployment group with CodeDeploy have been updated to reflect this additional support.	August 10, 2017

Change	Description	Date Changed
New and updated topics	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 Deployments. Other topics updated to reflect this support include Create an application for an in-place deployment (console), Create an application for a blue/green deployment (console), Create a deployment group for an in-place deployment (console), Create a deployment group for an EC2/On-Premises blue/green deployment (console), Deployments, and Step 5: Create an application and deployment group in Tutorial: Use CodeDeploy to deploy an application from GitHub.	July 31, 2017
Updated topic	Two additional methods for installing the CodeDeploy agent on Windows Server instances have been added to Install the CodeDeploy agent for Windows Server. 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

Change	Description	Date Changed
Updated topics	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 CodeDeploy applications with GitHub repositor ies. Each connection can support multiple repositor ies. 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, 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, Integrating CodeDeploy with GitHub, and Tutorial: Use CodeDeploy to deploy an application from GitHub.	May 30, 2017
Updated topics	In the past, if the 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. CodeDeplo y now provides options for how the agent handles these files: fail the deployment, retain the content, or overwrite the content. Create a deployment with CodeDeploy was updated to reflect this support, and the new section Rollback behavior with existing content was added to Redeploy and roll back a deployment with CodeDeploy.	May 16, 2017

Change	Description	Date Changed
Updated topics	A Classic Load Balancer in Elastic Load Balancing can now be assigned to a deployment group using the 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, Integrating CodeDeploy with Elastic Load Balancing, Create an application for an in-place deployment (console), Create a deployment group for an in-place deployment (console), and AppSpec 'hooks' section. A new section was added to the troubleshooting guide, Troubleshooting a failed ApplicationStop, BeforeBlockTraffic, or AfterBlockTraffic deployment lifecycle event.	April 27, 2017
Updated topics	A Classic Load Balancer in Elastic Load Balancing can now be assigned to a deployment group using the 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, Integrating CodeDeploy with Elastic Load Balancing, Create an application for an in-place deployment (console), Create a deployment group for an in-place deployment (console), and AppSpec 'hooks' section. A new section was added to the troubleshooting guide, Troubleshooting a failed ApplicationStop, BeforeBlockTraffic, or AfterBlockTraffic deployment lifecycle event.	May 1, 2017

Change	Description	Date Changed
Updated topics	CodeDeploy is now available in the China (Beijing) Region. To use services in the China (Beijing) Region or China (Ningxia) Region, you must have an account and credentia ls 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 CodeDeploy Resource Kit bucket names and CodeDeploy agent installation procedures, are not included in this edition of the *CodeDeploy User Guide*. For more information: • CodeDeploy in *Getting Started with AWS in the China (Beijing) Region* • *CodeDeploy User Guide for the China Regions* (English version | Chinese version)	March 29, 2017

Change	Description	Date Changed
New and updated topics	Several new topics have been introduced to reflect new 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 provides a high-level explanation of the blue/green methodology used by CodeDeploy. Additional new topics include Create an application for a blue/green deployment (console). Create a deployment group for an EC2/On-Premises blue/green deployment (console), and Set up a load balancer in Elastic Load Balancing for CodeDeploy Amazon EC2 deployments. Numerous topics have been updated as well, including Create a deployment with CodeDeploy, Working with deployment configurations in CodeDeploy, Create an application with CodeDeploy, Working with deployment groups in CodeDeploy , Working with deployments in CodeDeploy, and AppSpec 'hooks' section.	January 25, 2017
New and updated topics	A new topic, Use the register-on-premises-instance command (IAM Session ARN) to register an on-premises instance, 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 has also been updated to reflect this new support.	December 28, 2016

Change	Description	Date Changed
Updated topics	CodeDeploy is now available in the Europe (London) Region (eu-west-2). Several topics, including those containing instructions for setting up the CodeDeploy agent, were updated to reflect the availability of this new region.	December 13, 2016
Updated topics	CodeDeploy is now available in the Canada (Central) Region (ca-central-1). Several topics, including those containing instructions for setting up the CodeDeploy agent, were updated to reflect the availability of this new region.	December 8, 2016
Updated topics	CodeDeploy is now available in the US East (Ohio) Region (us-east-2). Several topics, including those containing instructions for setting up the CodeDeploy agent, were updated to reflect the availability of this new region.	October 17, 2016
New topics	A new section, Authentication and Access Control, provides comprehensive information about using AWS Identity and Access Management (IAM) and CodeDeploy to help secure access to your resources through the use of credentials. These credentials provide the permissio ns required to access AWS resources, such as retrievin g application revisions from Amazon S3 buckets and reading the tags on Amazon EC2 instances.	October 11, 2016
Updated topic	Update the CodeDeploy agent on Windows Server has been updated to reflect the availability of the new 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

Change	Description	Date Changed
Updated topics	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. CodeDeploy also now supports automatically rolling back a deployment if certain conditions are met, such as a deployment failure or an activated alarm. A number of topics have been updated to reflect these changes, including Create an application with CodeDeploy, Create a deployment group with CodeDeploy, Change deployment group settings with CodeDeploy, Deployments, Redeploy and roll back a deployment with CodeDeploy, and Product and service integrations with CodeDeploy, along with a new topic, Monitoring deployments with CloudWatch alarms in CodeDeploy.	September 15, 2016
New and updated topics	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 a 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.	September 9, 2016

Change	Description	Date Changed
Topic updates	The topics Integrating CodeDeploy with Elastic Load Balancing and Integration with other AWS services have been updated to reflect an additional load balancing option. CodeDeploy now supports the Classic Load Balancer and Application Load Balancer available in Elastic Load Balancing.	August 11, 2016
Topic updates	CodeDeploy is now available in the Asia Pacific (Mumbai) Region (ap-south-1). Several topics, including those containing instructions for setting up the CodeDeploy agent, were updated to reflect the availability of this new region.	June 27, 2016
Topic updates	CodeDeploy is now available in the Asia Pacific (Seoul) Region (ap-northeast-2). Several topics, including those containing instructions for setting up the CodeDeploy agent, were updated to reflect the availability of this new region. 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 CodeDeploy tutorials. For better usability, several long topics, including CodeDeploy AppSpec file reference and Troubleshooting CodeDeploy, have been divided into shorter topics. Configuration information for the CodeDeploy agent has been moved to a new topic, CodeDeploy agent configuration reference.	June 15, 2016

Change	Description	Date Changed
New and updated topics	Error codes for AWS CodeDeploy provides informati on about some of the error messages that might be displayed when CodeDeploy deployments fail. The following sections in Troubleshooting CodeDeploy were updated to better assist with resolving deployment problems: • EC2 instances in an Amazon EC2 Auto Scaling group fail to launch and receive the error "Heartbeat Timeout" • Avoid associating multiple deployment groups with a single Amazon EC2 Auto Scaling group	April 20, 2016
Topic updates	CodeDeploy is now available in the South America (São Paulo) Region (sa-east-1). Several topics, including those containing instructions for setting up the CodeDeploy agent, were updated to reflect the availability of this new region. Working with the CodeDeploy agent 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 CodeDeploy agent to archive.	March 10, 2016

Change	Description	Date Changed
New and updated topics	CodeDeploy now supports adding triggers to a deployment group to receive notifications about events related to deployments or instances in that deploymen t 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. Procedures were updated to reflect the redesign of the **Application details** page. The Deployments do not fail for up to an hour when an instance is terminated during a deployment section in Troubleshooting CodeDeploy has been updated. CodeDeploy quotas 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.	February 17, 2016

Change	Description	Date Changed
New and updated topics	CodeDeploy is now available in the US West (N. California) region (us-west-1). Several topics, including those containing instructions for setting up the CodeDeploy agent, were updated to reflect the addition of this new region. Choose a CodeDeploy repository type lists and describes the repository types now supported by CodeDeploy. This new topic will be updated as support for other repository types is introduced. Managing CodeDeploy agent operations was updated with information about the new `.version` file added to instances to report the current version of the CodeDeploy agent, as well as information about supported versions of the agent. Syntax highlighting for code samples, including JSON and YAML examples, has been added to the user guide. Add an application specification file to a revision for CodeDeploy has been reorganized as step-by-step instructions.	January 20, 2016
New topic	Deploy an application in a different AWS account 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

Change	Description	Date Changed
Topic update	The Product and service integrations with CodeDeploy 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 CodeDeploy integrations.	December 16, 2015
Topic updates	CodeDeploy is now available in the Asia Pacific (Singapore) Region (ap-southeast-1). Several topics, including those containing instructions for setting up the CodeDeploy agent, were updated to reflect the availability of this new region.	December 9, 2015
Topic updates	Working with the CodeDeploy agent was updated to reflect the new `:proxy_uri:` option in the CodeDeploy agent configuration file. CodeDeploy AppSpec file reference was updated with information about using a new environment variable, `DEPLOYMENT_GROUP_ID`, which hook scripts can access during a deployment lifecycle event.	December 1, 2015
Topic update	Step 2: Create a service role for CodeDeploy was updated to reflect a new procedure for creating a service role for CodeDeploy and to incorporate other improvements.	November 13, 2015
Topic updates	CodeDeploy is now available in the Europe (Frankfurt) Region (eu-central-1). Several topics, including those containing instructions for setting up the CodeDeploy agent, were updated to reflect the availability of this new region. The Troubleshooting CodeDeploy topic was updated with information about ensuring that time settings on instances are accurate.	October 19, 2015

Change	Description	Date Changed
New topics	CloudFormation templates for CodeDeploy reference was published to reflect new CloudFormation support for CodeDeploy actions. Created a Primary Components topic and introduced definition of a *target revision*.	October 1, 2015
Topic updates	Create a deployment group with CodeDeploy was updated to reflect the ability to locate instances for a deployment group using wildcard searches. Instance Health was updated to clarify the concept of *minimum healthy instances*.	August 31, 2015
Topic updates	CodeDeploy is now available in the Asia Pacific (Tokyo) Region (ap-northeast-1). Several topics, including those containing instructions for setting up the CodeDeploy agent, were updated to reflect the availability of this new region.	August 19, 2015
Topic updates	CodeDeploy now supports deployments to Red Hat Enterprise Linux (RHEL) on-premises instances and Amazon EC2 instances. For more information, see the following topics: • Operating systems supported by the CodeDeploy agent • Working with instances for CodeDeploy • Tutorial: Deploy WordPress to an Amazon EC2 instance (Amazon Linux or Red Hat Enterprise Linux and Linux, macOS, or Unix) • Tutorial: Deploy an application to an on-premises instance with CodeDeploy (Windows Server, Ubuntu Server, or Red Hat Enterprise Linux)	June 23, 2015

Change	Description	Date Changed
Topic update	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 CodeDeploy application, deployment group, and deployment lifecycle event, as well as the current CodeDeploy deployment identifier. For more information, see the end of the AppSpec 'hooks' section section in the CodeDeploy AppSpec file reference.	May 29, 2015
Topic updates	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: • A policy for enabling a user to register revisions with CodeDeploy only and then deploy them through CodeDeploy. • A policy for providing a user with full access to CodeDeploy resources. • A policy for providing a user with read-only access to CodeDeploy resources. • A policy to attach to a service role so that CodeDeplo y can identify Amazon EC2 instances by their Amazon EC2 tags, on-premises instance tags, or Amazon EC2 Auto Scaling group names and deploy application revisions to them accordingly. For more information, see the Customer-managed policy examples section in Authentication and Access Control.	May 29, 2015

Change	Description	Date Changed
Topic updates	CodeDeploy is now available in the Europe (Ireland) Region (eu-west-1) and the Asia Pacific (Sydney) Region (ap-southeast-2). Several topics, including those containin g instructions for setting up the CodeDeploy agent, were updated to reflect the availability of these new regions.	May 7, 2015
New topics	CodeDeploy now supports deployments to on-premises instance and Amazon EC2 instances. The following topics were added to describe this new support: • Working with On-Premises Instances • Tutorial: Deploy an application to an on-premises instance with CodeDeploy (Windows Server, Ubuntu Server, or Red Hat Enterprise Linux) • Working with On-Premises Instances	April 2, 2015
New topic	CodeDeploy resources was added.	April 2, 2015

Change	Description	Date Changed
Topic update	Troubleshooting CodeDeploy was updated: • A new Long-running processes can cause deployments to fail section describes steps you can take to identify and address deployment failures due to long-running processes. • The General Amazon EC2 Auto Scaling troubleshooting section was updated to show that CodeDeploy has increased its Amazon EC2 Auto Scaling timeout logic for the CodeDeploy agent from five minutes to one hour. • A new Mismatched Amazon EC2 Auto Scaling lifecycle hooks might cause automatic deployments to Amazon EC2 Auto Scaling groups to stop or fail section describes steps you can take to identify and address failed automatic deployments to Amazon EC2 Auto Scaling groups.	April 2, 2015

Change	Description	Date Changed
Topic updates	The following topics were updated to reflect new recommendations for creating your own custom policies and then attaching them to users and roles in IAM: • Configure an Amazon EC2 instance to work with CodeDeploy • Step 4: Create an IAM instance profile for your Amazon EC2 instances • Step 2: Create a service role for CodeDeploy Two sections were added to Troubleshooting CodeDeploy: • General troubleshooting checklist • Windows PowerShell scripts fail to use the 64-bit version of Windows PowerShell by default The AppSpec 'hooks' section section in the CodeDeploy AppSpec file reference was updated to more accurately describe the available deployment lifecycle events.	February 12, 2015
Topic updates	A new section was added to Troubleshooting CodeDeploy: EC2 instances in an Amazon EC2 Auto Scaling group fail to launch and receive the error "Heartbeat Timeout". A CloudBees section was added to Product and service integrations with CodeDeploy.	January 28, 2015

Change	Description	Date Changed
Topic updates	The following sections were added to Troubleshooting CodeDeploy: • Using some text editors to create AppSpec files and shell scripts can cause deployments to fail • Using Finder in macOS to bundle an application revision can cause deployments to fail • Troubleshooting a failed ApplicationStop, BeforeBlockTraffic, or AfterBlockTraffic deployment lifecycle event • Troubleshooting a failed DownloadBundle deployment lifecycle event with UnknownError: not opened for reading • General Amazon EC2 Auto Scaling troubleshooting	January 20, 2015
New topics	The Product and service integrations with CodeDeploy section was updated to include the following topics: • Integrating CodeDeploy with Amazon EC2 Auto Scaling • Tutorial: Use CodeDeploy to deploy an application to an Auto Scaling group • Monitoring Deployments • Integrating CodeDeploy with Elastic Load Balancing • Integrating CodeDeploy with GitHub • Tutorial: Use CodeDeploy to deploy an application from GitHub	January 9, 2015

Change	Description	Date Changed
Topic updates	• The Automatically deploy from CodePipeline with CodeDeploy section was added to Integrating CodeDeploy with GitHub. You can now automatic ally trigger a deployment from a GitHub repository whenever the source code in that repository is changed. • The Troubleshoot Amazon EC2 Auto Scaling issues section was added to Troubleshooting CodeDeploy. This new section describes how to troubleshoot common issues with deploying to Amazon EC2 Auto Scaling groups. • The new subsection "files Examples" was added to the AppSpec 'files' section (EC2/On-Premises deploymen ts only) section of CodeDeploy AppSpec file reference . This new subsection includes several examples of how to use the `files` section of an AppSpec file to instruct 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 was added. CodeDeploy is integrated with AWS CloudTrail, a service that captures API calls made by or on behalf of CodeDeploy in your AWS account and delivers the log files to an Amazon S3 bucket that you specify.	December 17, 2014
Initial public release	This is the initial public release of the *CodeDeploy User Guide*.	November 12, 2014

AWS Glossary

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