AWS CodeDeploy

Tejas Parikh (t.parikh@northeastern.edu)

CSYE 6225

Northeastern University

AWS CodeDeploy

- AWS CodeDeploy is a service that automates code deployments to any instance, including Amazon EC2 instances and instances running on-premises.
- AWS CodeDeploy makes it easier for you to rapidly release new features, helps you avoid downtime during application deployment, and handles the complexity of updating your applications.
- AWS CodeDeploy automates software deployments, eliminating the need for error-prone manual operations enabling you to easily deploy to one instance or thousands.

What AWS CodeDeploy Provides

- Automated Deployments
  - Repeatable Deployments
  - Auto Scaling Integration
- Minimize Downtime
  - Rolling and Blue/Green Updates
  - Stop and Rollback
- Centralized Control
  - Deployment Groups
  - Deployment History
- Easy To Adopt
  - Language and Architecture Agnostic

CodeDeploy Primary Components

- Application
- Compute platform
- Deployment configuration
- Deployment group
- Deployment type
- IAM instance profile
- Service role
- Target revision

Application

- 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

- Compute Platform - The platform on which CodeDeploy deploys an application
  - **EC2/On-Premises** - Describes instances of physical servers that can be Amazon EC2 cloud instances, on-premises servers, or both. Applications created using the EC2/On-Premises compute platform can be composed of executable files, configuration files, images, and more.
  - **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. Applications created using the AWS Lambda compute platform can manage the way in which traffic is directed to the updated Lambda function versions during a deployment by choosing a canary, linear, or all-at-once configuration.
  - **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 containerized application as a new replacement task set. CodeDeploy reroutes production traffic from the original application, or task set, to the replacement task set. The original task set is terminated after a successful deployment.

Deployment Configuration

- 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.

Deployment Group

- A set of individual instances.
- A deployment group contains individually tagged instances, Amazon EC2 instances in Amazon EC2 Auto Scaling groups, or both.

Deployment Type

- The method used to make the latest application revision available on instances in a deployment group.
- **In-place deployment**: The application on each instance in the deployment group is stopped, the latest application revision is installed, and the new version of the application is started and validated.
- **Blue/green deployment**: The instances in a deployment group (the original environment) are replaced by a different set of instances (the replacement environment) using these steps:
  - Instances are provisioned for the replacement environment.
  - The latest application revision is installed on the replacement instances.
  - An optional wait time occurs for activities such as application testing and system verification.
  - Instances in the replacement environment are registered with an Elastic Load Balancing load balancer, causing traffic to be rerouted to them. Instances in the original environment are deregistered and can be terminated or kept running for other uses.

IAM instance profile

- An IAM role that you attach to your Amazon EC2 instances.
- This profile includes the permissions required to access the Amazon S3 buckets or GitHub repositories where the applications are stored.

Revision

- 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).

Service Role

- An IAM role that grants permissions to an AWS service so it can access AWS resources.
- The policies you attach to the service role determine which AWS resources the service can access and the actions it can perform with those resources.

Target Revision

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

CodeDeploy Application Specification (AppSpec) Files

AWS CodeDeploy App Spec

- An application specification file (AppSpec file), which is unique to AWS CodeDeploy, is a YAML-formatted file 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.
- The AppSpec file is used to manage each deployment as a series of lifecycle events.
- Lifecycle event hooks, which are defined in the file, allow you to run scripts on an instance after most deployment lifecycle events.
- AWS CodeDeploy runs only those scripts specified in the file, but those scripts can call other scripts on the instance.
- You can run any type of script as long as it is supported by the operating system running on the instances.
- http://docs.aws.amazon.com/codedeploy/latest/userguide/writing-app-spec.html

AppSpec Template

- https://docs.aws.amazon.com/codedeploy/latest/userguide/application-revisions-appspec-file.html#add-appspec-file-server

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.

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.
- Required only if you are copying files from your revision to locations on the instance during deployment.

This section has the following structure:

```
files:
  - source: source-file-location
    destination: destination-file-location
```

Multiple `source` and `destination` pairs can be set.

The `source` instruction identifies a file or directory from your revision to copy to the instance:

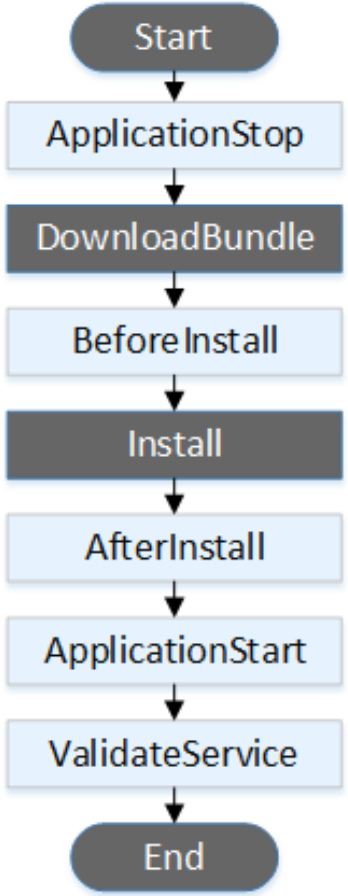
- If `source` refers to a file, only the specified files are copied to the instance.
- If `source` refers to a directory, then all files in the directory are copied to the instance.
- If `source` is a single slash ("/" for Amazon Linux, RHEL, and Ubuntu Server instances, or "\" for Windows Server instances), then all of the files from your revision are copied to the instance.

The paths used in `source` are relative 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.

List of lifecycle event hooks

- ApplicationStop
- DownloadBundle
- BeforeInstall
- Install
- AfterInstall
- ApplicationStart
- ValidateService

Run order of hooks in a deployment



AppSpec Hooks - ApplicationStop

- This deployment lifecycle event occurs even before the application revision is downloaded.
- You can specify scripts for this event to gracefully stop the application or remove currently installed packages in preparation of a deployment.
- The AppSpec file and scripts used for this deployment lifecycle event are from the previous successfully deployed application revision.
- **Note** : An AppSpec file does not exist on an instance before you deploy to it. For this reason, the ApplicationStop hook does not run the first time you deploy to the instance. You can use the ApplicationStop hook the second time you deploy to an instance.

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.

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.

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
```

- **location**: 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.
- **timeout**: The number of seconds to allow the script to execute before it is considered to have failed. The default is 3600 seconds (1 hour).
- **runas:** 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.

AppSpec File example for an EC2/On-Premises deployment

```yaml
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
```

CodeDeploy Agent Policy

- For EC2/On-Premises deployments, attach the `AWSCodeDeployRole` policy to your IAM role attached to the EC2 instance. 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.

Install the CodeDeploy agent for Ubuntu Server

- See https://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-agent-operations-install-ubuntu.html

Additional Resources

See Lecture Page