AMAZON AWS SOLUTIONS ARCHITECT

AWS CODEDEPLOY OVERVIEW

CODEDEPLOY

- AWS CodeDeploy is a deployment service that automates application deployments to Amazon EC2 instances, on-premises instances, or serverless Lambda functions.
- You can deploy a nearly unlimited variety of application content:
  - such as code
  - serverless AWS Lambda functions
  - web and configuration files
  - executables, packages, scripts, multimedia files
- AWS CodeDeploy can deploy application content that runs on a server and is stored in Amazon S3 buckets, GitHub repositories, or Bitbucket repositories
- AWS CodeDeploy can also deploy a serverless Lambda function. You do not need to make changes to your existing code before you can use AWS CodeDeploy

BENEFITS OF AWS CODEDEPLOY

- **Server and serverless applications**. AWS CodeDeploy lets you deploy both traditional applications on servers and applications that deploy a serverless AWS Lambda function version.
- **Automated deployments**. AWS CodeDeploy fully automates your application deployments across your development, test, and production environments. AWS CodeDeploy scales with your infrastructure so that you can deploy to one instance or thousands.
- **Minimize downtime**. AWS CodeDeploy helps maximize your application availability. During an in-place deployment, AWS CodeDeploy performs a rolling update across Amazon EC2 instances. You can specify the number of instances to be taken offline at a time for updates. During a blue/green deployment, the latest application revision is installed on replacement instances, and traffic is rerouted to these instances when you choose, either immediately or as soon as you are done testing the new environment. For both deployment types, AWS CodeDeploy tracks application health according to rules you configure.

BENEFITS OF AWS CODEDEPLOY

- **Stop and roll back**. You can automatically or manually stop and roll back deployments if there are errors.
- **Centralized control**. You can launch and track the status of your deployments through the AWS CodeDeploy console or the AWS CLI. You receive a report that lists when each application revision was deployed and to which Amazon EC2 instances.
- **Easy to adopt**. AWS CodeDeploy is platform-agnostic and works with any application. You can easily reuse your setup code. AWS CodeDeploy can also integrate with your software release process or continuous delivery toolchain.

CODEDEPLOY COMPUTE PLATFORMS

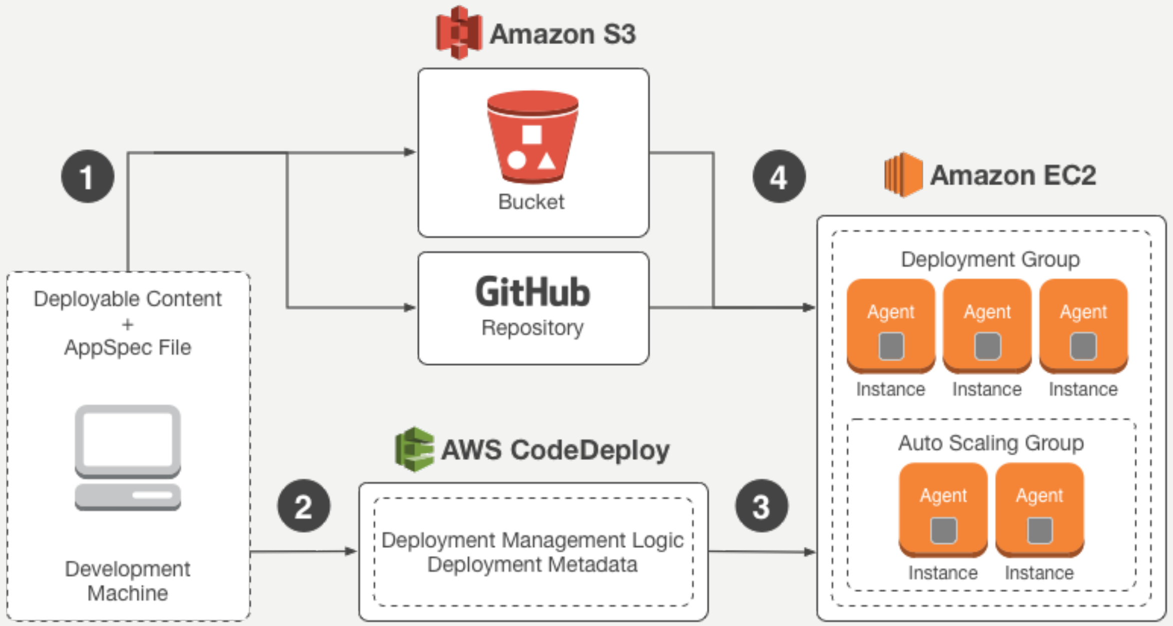
- **EC2/On-Premises**
  - Describes instances of physical servers that can be Amazon EC2 cloud instances, on-premises servers, or both
  - Applications created using the EC2/On-Premises compute platform can be composed of executable files, configuration files, images, and more.
  - Deployments that use 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
- **AWS Lambda**
  - Used to deploy applications that consist of updated versions of Lambda functions
  - AWS Lambda manages the Lambda functions in a serverless compute environment made up of a high-availability compute structure
  - All administration of the compute resources is performed by AWS Lambda.

CODEDEPLOY COMPONENTS

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

CODEDEPLOY 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.



OVERVIEW OF AN IN-PLACE DEPLOYMENT

1. You create deployable content on your local development machine or similar environment, and then you add an *application specification file* (AppSpec file). The AppSpec file is unique to AWS CodeDeploy. It defines the deployment actions you want AWS CodeDeploy to execute. You bundle your deployable content and the AppSpec file into an archive file, and then upload it to an Amazon S3 bucket or a GitHub repository. This archive file is called an *application revision* (or simply a *revision*).
2. You provide AWS CodeDeploy with information about your deployment, such as which Amazon S3 bucket or GitHub repository to pull the revision from and to which set of Amazon EC2 instances to deploy its contents. AWS CodeDeploy calls a set of Amazon EC2 instances a *deployment group*. A deployment group contains individually tagged Amazon EC2 instances, Amazon EC2 instances in Auto Scaling groups, or both.
3. Each time you successfully upload a new application revision that you want to deploy to the deployment group, that bundle is set as the *target revision* for the deployment group. In other words, the application revision that is currently targeted for deployment is the target revision. This is also the revision that will be pulled for automatic deployments.
4. The AWS CodeDeploy agent on each instance polls AWS CodeDeploy to determine what and when to pull from the specified Amazon S3 bucket or GitHub repository.
5. The AWS CodeDeploy agent on each instance pulls the target revision from the specified Amazon S3 bucket or GitHub repository and, using the instructions in the AppSpec file, deploys the contents to the instance

CODEDEPLOY DEPLOYMENT TYPES

- **Blue/green deployment**
  - **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:
      1. Instances are provisioned for the replacement environment.
      2. The latest application revision is installed on the replacement instances.
      3. An optional wait time occurs for activities such as application testing and system verification.
      4. 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.
  - **Blue/green on an AWS Lambda compute platform**
    - Traffic is shifted from your current serverless environment to one with your updated Lambda function versions.
    - You can specify Lambda functions that perform validation tests and choose the way in which the traffic shift occurs
    - All AWS Lambda compute platform deployments are blue/green deployments.

OVERVIEW OF BLUE/GREEN DEPLOYMENT

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

BLUE/GREEN DEPLOYMENT FOR LAMBDA

- If you're using the AWS Lambda compute platform, you must choose one of the following deployment configuration types to specify how traffic is shifted from the original AWS Lambda function version to the new AWS Lambda function version:
  - **Canary**: Traffic is shifted in two increments. You can choose from predefined canary options that specify the percentage of traffic shifted to your updated Lambda function version in the first increment and the interval, in minutes, before the remaining traffic is shifted in the second increment.
  - **Linear**: Traffic is shifted in equal increments with an equal number of minutes between each increment. You can choose from predefined linear options that specify the percentage of traffic shifted in each increment and the number of minutes between each increment.
  - **All-at-once**: All traffic is shifted from the original Lambda function to the updated Lambda function version at once.

BLUE/GREEN DEPLOYMENT FOR EC2

1. You already have instances or an Auto Scaling group that will serve as your original environment. The first time you run a blue/green deployment, you'll typically use instances that were already used in an in-place deployment.
2. In an existing AWS CodeDeploy application, you create a blue/green deployment group where, in addition to the options required for an in-place deployment, you specify the following:
   - The load balancer that will route traffic from your original environment to your replacement environment during the blue/green deployment process.
   - Whether to reroute traffic to the replacement environment immediately or wait for you to reroute it manually.
   - The rate at which traffic is routed to the replacement instances.
   - Whether the instances that are replaced are terminated or kept running.

BLUE/GREEN DEPLOYMENT FOR EC2

3. You create a deployment for this deployment group during which the following occur:
   - If you chose to copy an Auto Scaling group, instances are provisioned for your replacement environment.
   - The application revision you specify for the deployment is installed on the replacement instances.
   - 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.
   - Instances in the replacement environment are registered with an Elastic Load Balancing load balancer and traffic begins to be routed to them.
   - Instances in the original environment are deregistered and handled according to your specification in the deployment group, either terminated or kept running.

DEMO!

- Deploy an Application from Github to EC2 Instance

QUESTIONS?