😊 AWS Code Deploy

AWS CodeDeploy is a service provided by Amazon Web Services (AWS) that automates the deployment of applications to various computing services such as Amazon EC2, AWS Lambda, and on-premises servers. It is designed to help developers and IT operations teams manage application updates without causing downtime or disrupting normal operations.

Key Features:

1. **Automated Deployments:** CodeDeploy automates the deployment process, reducing the potential for human error and ensuring consistency.
2. **Support for Various Platforms:**
    - **EC2 Instances:** Deploy applications to one or more EC2 instances.
    - **AWS Lambda:** Update your Lambda functions in a safe, automated manner.
    - **On-Premises:** Deploy applications to on-premises servers, enabling a hybrid deployment model.
3. **Deployment Strategies:**
    - **In-Place Deployment:** Updates the application on the existing instances.
    - **Blue/Green Deployment:** Creates a new set of instances for the new version of the application, minimizing downtime and risk.
4. **Monitoring and Rollback:** CodeDeploy provides monitoring features to track the status of your deployments. It can automatically roll back to the previous version if there are any issues detected during the deployment process.
5. **Customizable Deployment Configurations:** You can define your deployment settings, such as the number of instances to be updated simultaneously and the order of deployment across instances.
6. **Integration with Other AWS Services:** CodeDeploy integrates seamlessly with other AWS services such as AWS CodePipeline, AWS CloudFormation, Amazon CloudWatch, and more, enabling a robust CI/CD pipeline.

Benefits:

- **Increased Agility:** Faster and more reliable deployments help teams release features and updates more frequently.
- **Reduced Downtime:** With strategies like blue/green deployments, application updates can be rolled out with minimal or no downtime.
- **Scalability:** Easily scale your deployments to thousands of instances.
- **Consistency:** Automated deployments ensure that applications are deployed the same way every time, reducing inconsistencies and errors.

- **Hybrid Deployments:** Ability to deploy applications both on AWS and on-premises environments.

Use Cases:

- **Web Applications:** Deploy updates to web servers running on EC2 instances.
- **Microservices:** Update microservices running on AWS Lambda or EC2 instances.
- **Hybrid Environments:** Deploy updates across both cloud and on-premises environments.
- **Continuous Integration/Continuous Deployment (CI/CD):** Integrate with tools like AWS CodePipeline to automate the end-to-end software release process.

Overall, AWS CodeDeploy is a powerful tool for automating the deployment of applications, ensuring reliable and scalable application delivery.

Detailed Features:

1. **Deployment Groups and Instances:**
    - **Deployment Groups:** A deployment group is a set of individual instances targeted for deployment. It can include EC2 instances, Lambda functions, or on-premises instances.
    - **Tagging:** Instances can be grouped using tags, auto-scaling group names, or manually specified instances.
2. **Deployment Types:**
    - **In-Place Deployment (Rolling Deployment):** Updates the application on the existing instances, one at a time or in groups, while keeping the application running. This method can involve a short downtime as the application is updated.
    - **Blue/Green Deployment:** Deploys the new version to a new set of instances (green environment) while the old version continues to serve traffic (blue environment). After testing, traffic is switched to the green environment. If issues are found, the traffic can easily be switched back to the blue environment.
3. **Lifecycle Hooks:**
    - **Hooks:** These are custom scripts that run at different stages of the deployment lifecycle. They allow you to run specific actions before, during, or after the deployment process. Examples include:
        - BeforeInstall
        - AfterInstall
        - ApplicationStart
        - ApplicationStop
        - BeforeBlockTraffic and AfterBlockTraffic

        - BeforeAllowTraffic and AfterAllowTraffic
    - These hooks enable tasks such as backups, configuration changes, and service restarts.

4. **Application Revisions:**
    - An application revision is a specific version of an application that you want to deploy. Revisions can be stored in Amazon S3, GitHub, or Bitbucket.
    - Revisions can include source code, configuration files, executables, and deployment scripts.
5. **Deployment Configurations:**
    - **Minimum Healthy Hosts:** Define the minimum number of healthy instances that should be available during the deployment.
    - **Traffic Control:** For Lambda deployments, CodeDeploy can control how traffic is shifted from the old version to the new version, either gradually or immediately.
6. **Monitoring and Logging:**
    - **Amazon CloudWatch:** CodeDeploy integrates with CloudWatch to provide metrics and alarms for deployment status and instance health.
    - **Logs:** Deployment logs are available in CloudWatch Logs, making it easier to troubleshoot issues.
7. **Rollback Mechanism:**
    - CodeDeploy can automatically roll back to the previous version if a deployment fails. This ensures that the application remains stable and reduces the risk of prolonged downtime.
8. **Integration with CI/CD Tools:**
    - **AWS CodePipeline:** CodeDeploy integrates with CodePipeline to automate the entire release process, from code commits to deployment.
    - **Third-Party Tools:** Integration with other CI/CD tools like Jenkins, GitLab, and CircleCI.

Detailed Benefits:

- **Flexibility:** Supports multiple deployment configurations and strategies, catering to different application needs and deployment requirements.
- **Reliability:** Automated rollbacks and lifecycle hooks ensure deployments are reliable and any issues can be quickly resolved.
- **Scalability:** Can handle large-scale deployments across thousands of instances.
- **Cost-Effective:** Automates many deployment tasks, reducing the need for manual intervention and thus lowering operational costs.

Example Use Cases:

1. **Web Application Deployment:**
    - Deploying a new version of a web application running on an EC2 Auto Scaling group with minimal downtime using blue/green deployments.
2. **Lambda Function Updates:**
    - Updating serverless applications by deploying new versions of AWS Lambda functions with traffic shifting for safe deployments.
3. **Microservices Architecture:**
    - Deploying updates to individual microservices independently, ensuring each service can be updated without affecting the others.
4. **On-Premises Applications:**
    - Automating the deployment of applications to on-premises servers, ensuring consistency and reliability across hybrid environments.
5. **CI/CD Pipelines:**
    - Integrating with AWS CodePipeline to automate the build, test, and deploy processes, ensuring fast and reliable delivery of new features and bug fixes.

Getting Started:

1. **Create a CodeDeploy Application:**
    - Define the application in the CodeDeploy console, specifying whether it is for EC2/On-Premises or Lambda.
2. **Define Deployment Groups:**
    - Specify the instances or Lambda functions to target in the deployment group. Use tags, auto-scaling groups, or manual selection to define the instances.
3. **Prepare Application Revision:**
    - Store the application revision (code, configuration, scripts) in a supported source like Amazon S3 or GitHub.
4. **Create Deployment Configuration:**
    - Define the deployment settings, such as the deployment type (in-place or blue/green), minimum healthy hosts, and traffic shifting for Lambda.
5. **Deploy the Application:**
    - Initiate the deployment from the CodeDeploy console, CLI, SDKs, or as part of a CI/CD pipeline.
6. **Monitor and Manage:**

- Use CloudWatch and CodeDeploy console to monitor the deployment progress and health. Configure alarms and notifications for deployment events.

AWS CodeDeploy simplifies and automates the deployment process, ensuring that application updates are efficient, reliable, and scalable.

In the upcoming labs we are going to learn about Code Deploy:

1. Understanding what AWS CodeDeoploy is and how to use it with AWS CodePipeline

2. Configuring an EC2 instance for CodeDeploy deployments.

3. Providing our deployment scripts and installation configuration to AWS CodeDeploy with our source code.

4. Monitoring deployment logs on an EC2 instance and streaming them to Amazon CloudWatch Logs for long-term storage.

5. Using CodeDeploy to deploy to multiple EC2 instances behind auto-scaling and load balancing.

6. Understanding how in-place all-at-once deployment works.

7. Understanding how in-place rolling deployment and automated rollbacks work.

8. Understanding what bule-green deployments are and how they work.