



CI/CD ON AMAZON WEB SERVICES (AWS)

Dinesh Kumar Shanmugasundaram
October 2016

ABSTRACT

Over the past decade there has been tremendous change and advancement in how applications are built and deployed by developers as a result of the ever-increasing demands from end users. In the past, enterprise release cycles lasted up to a quarter, sometimes even longer. However, application development moves at a much faster pace today. Applications are deployed within hours and delivered directly to consumer devices in the cloud. Cloud and mobile based computing allow development teams to build innovative and disruptive apps and disseminate them to millions of users without having to worry about infrastructure. This is one of the main reasons why startups and even agile teams within large corporations have been so successful. These agile and nimble teams can get real-time feedback from their customers, rapidly make changes to their designs and features, and ship the updated version back to their customers, all within a few hours of finishing them. Building and releasing applications at this pace needs end-to-end automation for consistent and repeatable results. Development teams need tools to manage these processes, test the applications automatically, and deploy the tested applications onto their target environments.

Application development used to be a time consuming and difficult process for developers who worked in isolation to merge their module to the master branch upon completion. This batched process can lead to the accumulation of minor bugs which can remain unfixed for long durations, delaying application delivery.

This whitepaper is intended for existing and potential AWS users – especially Architects, Developers, and SysOps administrators who prefer to implement their application development and deployments using the CI/CD model on AWS. This whitepaper highlights, the AWS services and features that can be leveraged to implement continuous integration and continuous delivery for the application development cycle. It also provides an overview, the stages involved, benefits of CI/CD, and hands-on examples. The techniques described allow users to scale applications while continuously integrating and deploying application changes that may be incurred.

Onica is an AWS Premier Consulting Partner. We specialize in guiding our customers with DevOps challenges on their journey into the cloud. Our goal is to increase automation and decrease the 20th century approach to technology thinking. We strive to give you continual measurements for achievement, on demand demonstrations, and milestones for approvals and rejections. Onica wants to help your teams' talent break through by automating your workloads, because we know that downtime and repetition cost organizations money.

WHAT WE'LL COVER

This whitepaper assumes familiarity with fundamental AWS services such as Elastic Compute Cloud (EC2), Simple Storage Service (S3), Identity and Access Management (IAM), and the basics of Git. We will cover the following services of AWS with a hands-on demonstration.

- **CI/CD Overview:** Why CI/CD, example architecture diagram of CI/CD on AWS, security, and cost
- **Automation Components of CI/CD:** Introduction to CodeCommit, CodePipeline, and CodeDeploy
- **Hands-On Demonstration of the following:**
 - **CodeCommit:** Create, connect, and delete CodeCommit repository
 - **CodePipeline:** Create, configure, trigger, and delete CodePipeline
 - **CodeDeploy:** Create, configure, trigger, and delete CodeDeploy

OVERVIEW

Continuous Integration (CI) is a software development practice that requires developers to regularly merge their code changes into a central repository (like git), which triggers the continuous integration tool to automatically build and run unit tests on the new code changes to immediately surface any functional or integration errors.

Continuous Delivery (CD) is an extension of continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage. When continuous delivery is implemented properly, developers will always have a deployment-ready build artifact that has passed through a standardized test process.

With CI/CD, code is committed to a central repository where it will be merged to the mainline branch followed by automated build creation, and testing. It will then be pushed to a non-production testing or staging environment. Tests may include UI testing, load testing, integration testing, API reliability testing, etc. Some of the key benefits of CI/CD are to automate the software release process, identify and fix bugs quickly, improve developer productivity, deliver software updates faster, etc. With the cloud, it's easy and cost-effective to automate the creation and replication of multiple environments for testing, which was previously difficult to do in an on-premises environment.

CI/CD as a process can be executed either in a manual fashion or it can be done using tools. As a common practice tools work, more effectively to enforce the discipline and avoid manual intervention. Aws provides fully-managed tools for the functions mentioned above.

WHY CI/CD

It is critically important for every team and especially the leaders of those teams to get their development workflows in order. Below are some of the benefits of using continuous integration and continuous delivery.

Improve Developer Productivity: CI/CD helps deliver productive results by reducing manual tasks, thus leading to lesser errors and bugs deployed to customers.

Find and Address Bugs Quicker: With automation and more frequent testing, your team can discover and address bugs early before they grow into larger problems later.

Deliver Updates Faster: Continuous integration helps your team deliver updates to their customers faster and more frequently. When continuous delivery is implemented properly, you will always have a deployment-ready build artifact that has passed through a standardized test process.

Automate the Software Release Process: Continuous delivery lets your team automatically build, test, and prepare code changes for release to production so that your software delivery is more efficient and rapid.

EXAMPLE WORKFLOW DIAGRAM

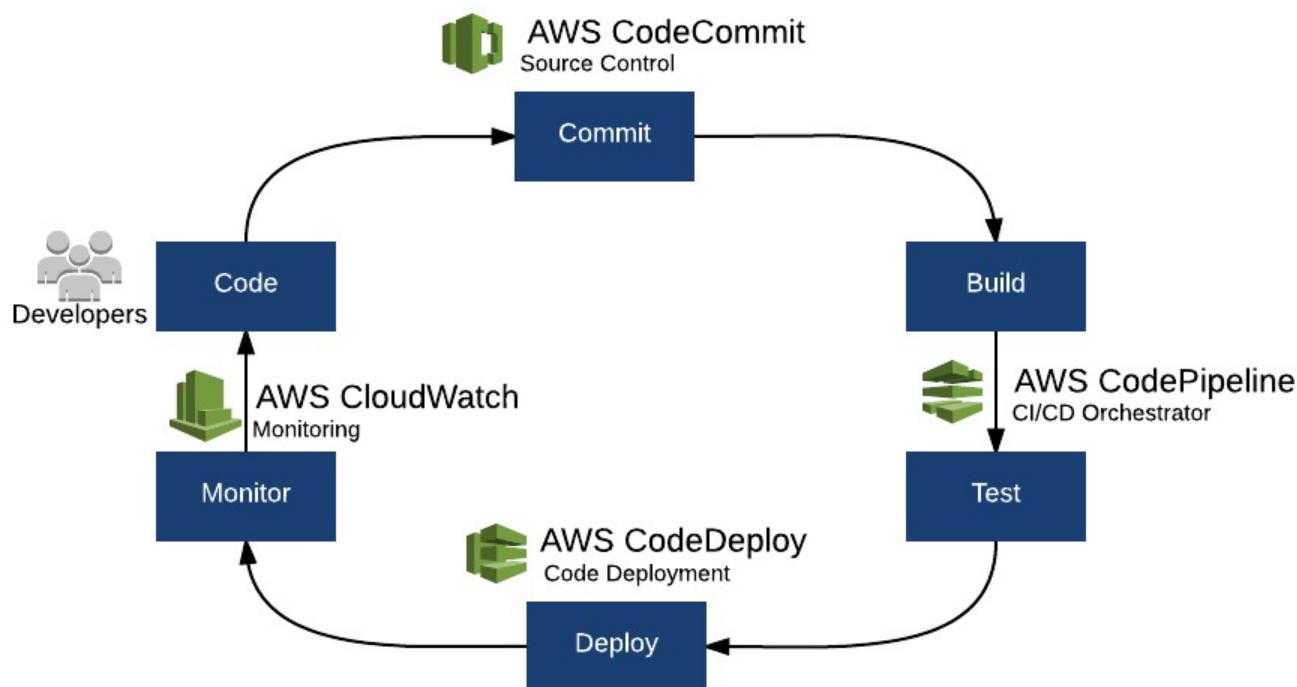


Figure 1: CI/CD on AWS Workflow

Security

IT and technology management should evaluate security before adopting any new systems and services. We've outlined AWS' and CI/CD security controls below.

Shared Responsibility Model

When evaluating security in AWS, it is important to understand the AWS Shared Responsibility Model (<https://aws.amazon.com/compliance/shared-responsibility-model/>). The underlying facilities, physical hypervisor resources, and the services themselves in the AWS cloud have undergone many extensive third-party audits and certifications for security compliance. To summarize, the Shared Responsibility Model identifies that AWS is responsible for the security of the AWS infrastructure as a service, while the customer is responsible for the secure use of those services.

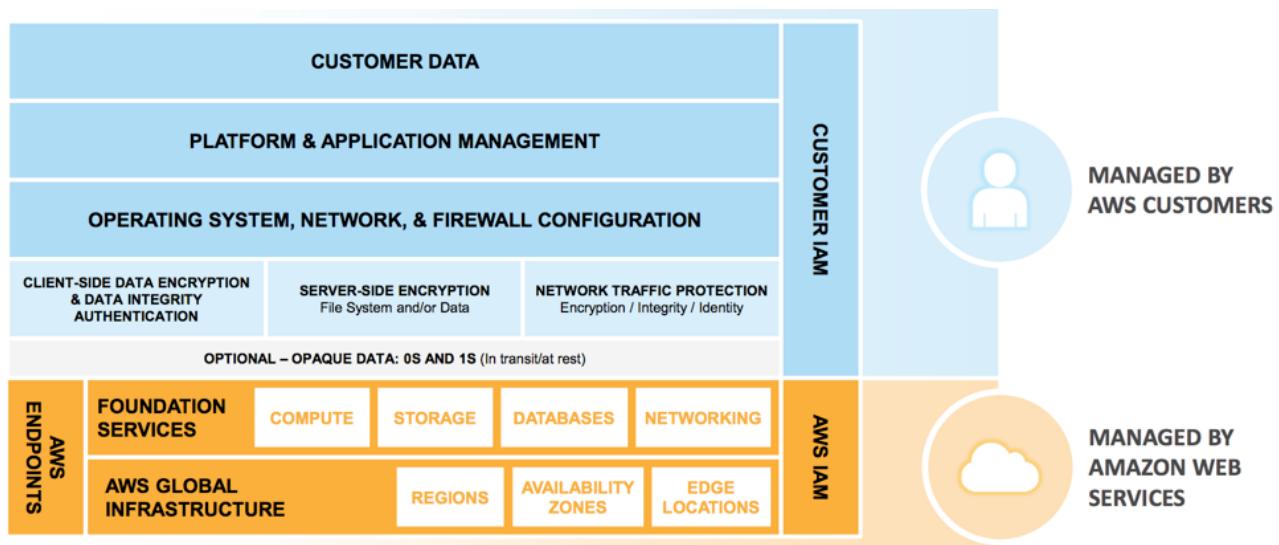


Figure 2: Shared Responsibility Model for Infrastructure Services

Continuous integration and continuous delivery tools such as CodeCommit, CodePipeline, and CodeDeploy are fully managed services hosted by AWS which provide high service availability and durability, and eliminate the administrative overhead of managing your own hardware and software. However, identity access and management allowing access to create/delete/modify repositories, pipelines, and deployment groups are the customer's responsibility. Inbound and outbound traffic of target EC2 instances to which AWS CodeDeploy deploys source code needs to be controlled with Security Groups and Network ACLs.

Security in the Virtual Private Cloud (VPC)

Before beginning to architect security practices across environments, basic data classification should be in place. Organize and classify data into segments such as publicly available and privately available. The instances in the public subnet can send outbound traffic directly to the Internet, whereas the instances in the private subnet can't. Instead the instances in the private subnet can access the Internet by using a network address translation (NAT) gateway that resides in the public subnet. The database servers can connect to the Internet for software updates using the NAT gateway, but the Internet cannot establish connections to the database servers. VPC endpoint enables instances in VPC to use their private IP addresses to

communicate to S3. Using a S3 bucket policy, the S3 bucket can be restricted so that it can be accessed only from EC2 instances from a specific VPC, preventing the traffic from going through the Internet.

Identity & Access Management Roles

When launching EC2 instances running in an AWS VPC, AWS actions such as downloading and uploading data in S3, provisioning instances, and so forth are all authorized and authenticated using Identity and Access Management (IAM) roles. This allows security administrators to implement least privileges policies to pipelines so that users only have access to specific S3 locations and other resources as needed. Additionally, using IAM roles allows these actions without storing AWS API keys anywhere. IAM role keys are rotated automatically at regular intervals. Any and all API actions to an AWS account can be logged and monitored using AWS CloudTrail.

Storage Security

Customers are also responsible for their storage security. It is recommended that customers use the encryption options provided by AWS Key Management Service (KMS) to encrypt their data. For example, CodePipeline stores artifacts in S3 which has a server-side encryption option, that will encrypt objects at the storage layer for protection. Also, another example would be EBS volumes that are attached to EC2 instances, which can be encrypted at rest using KMS.

COST

Below are the cost breakdowns for the main components of CI/CD on AWS.

CodeCommit

Anyone with an AWS account can get started with AWS CodeCommit for free. Your account gets five active users per month for free (within limits), after which you pay \$1 per additional active users per month. There are no upfront fees or commitments.

Free	\$1 per month
First five active users Receives: <ul style="list-style-type: none">• Unlimited repositories• 50 GB-month of storage• 10,000 Git requests/month	Each active user* above the first five Receives: <ul style="list-style-type: none">• Unlimited repositories• 10 GB-month of storage per active user• 2,000 Git requests/month per active user

CodePipeline

Free Tier: AWS CodePipeline offers new and existing customers one active pipeline for free each month for up to one year.

After Free Tier: With AWS CodePipeline, there are no upfront fees or commitments. You pay only for what you use. AWS CodePipeline costs \$1 per active* pipeline per month after the free tier. There is no charge for pipelines that are idle and have no new code changes moving through them. An active pipeline is a pipeline that has at least one code change that moves through it during the month (i.e. making an update to your code in your source code repository or rerunning your previous code change through your pipeline).

Note: *Additional charges may incur for storing and accessing your pipeline artifacts in Amazon S3 and for triggering actions from other AWS and third-party services that you connect to your pipeline.*

CodeDeploy

EC2: There is no additional charge for code deployments to Amazon EC2 instances through AWS CodeDeploy.

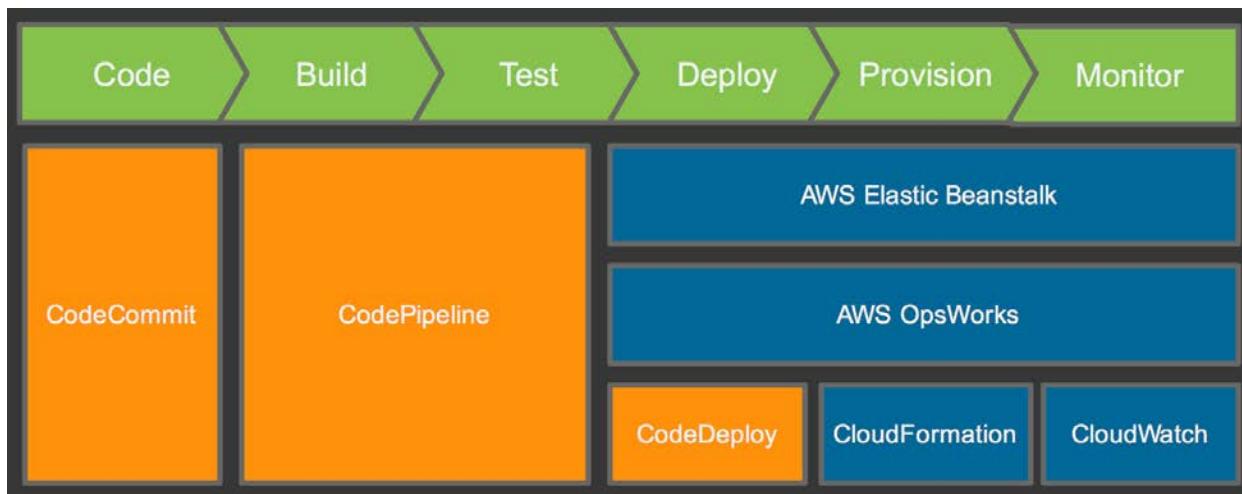
On-Premises: You pay \$0.02 per on-premises instance update using AWS CodeDeploy. There are no minimum fees and no upfront commitments. For example, a deployment to three instances equals three instance updates. You will only be charged if CodeDeploy performs an update to an instance. You will not be charged for any instances skipped during the deployment.

Note: *You only pay for any other AWS resources (e.g. S3 buckets) you may use in conjunction with CodeDeploy to store and run your application. You only pay for what you use, as you use it; there are no minimum fees and no upfront commitments.*

AUTOMATION COMPONENTS OF CI/CD

The diagram below represents the cloud software development lifecycle and the equivalent AWS Services. The core AWS automation components of continuous integration and continuous deployments are:

- **CodeCommit:** AWS CodeCommit is a secure, highly scalable, managed source control service that hosts private Git repositories.
- **CodePipeline:** AWS CodePipeline is a continuous delivery service for fast and reliable application updates.
- **CodeDeploy:** AWS CodeDeploy is a service that automates code deployments to any instance, including Amazon EC2 instances and instances running on-premises.



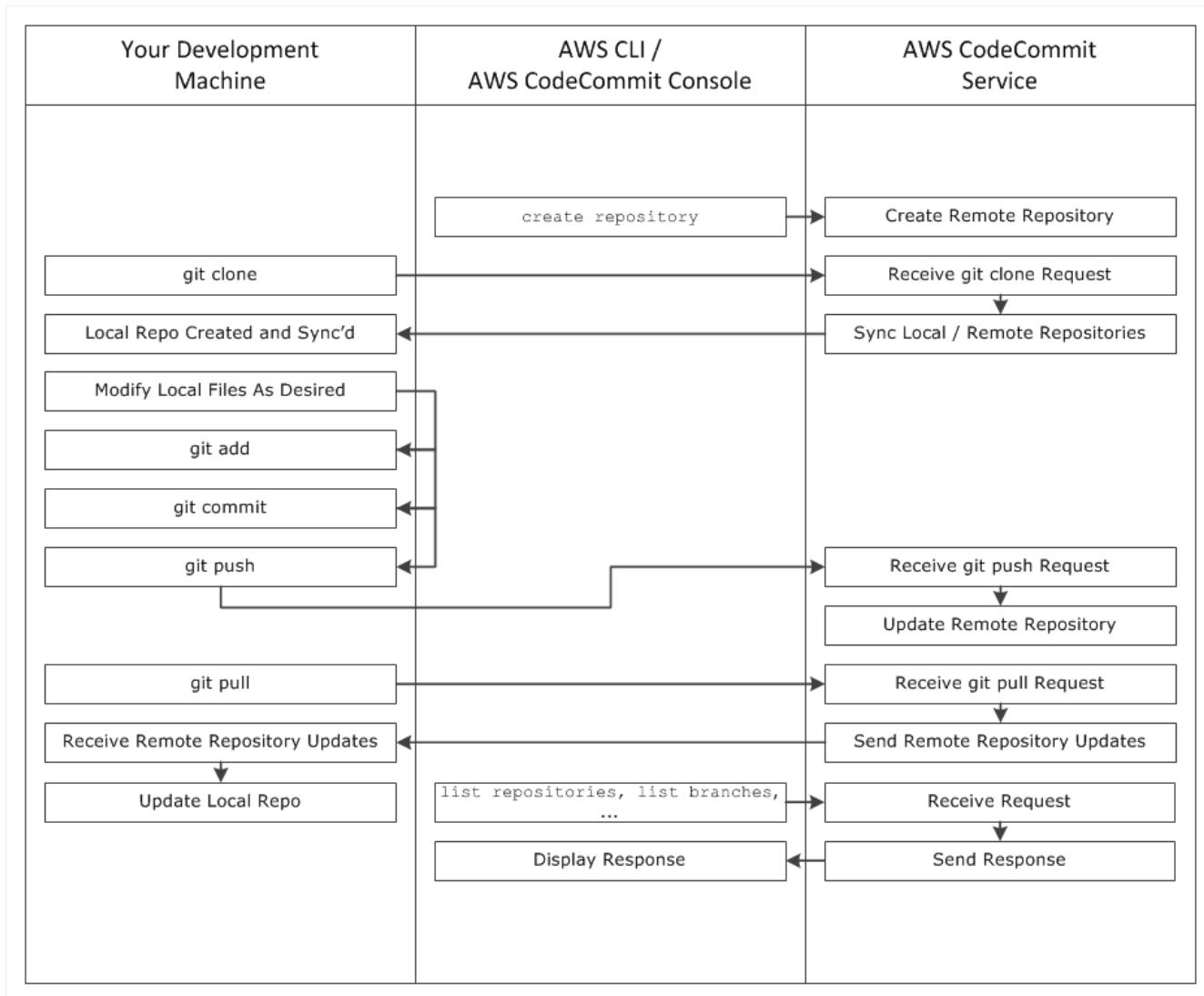
AWS CODECOMMIT

AWS CodeCommit is a secure, highly scalable, managed source control service that hosts private Git repositories. AWS CodeCommit eliminates the need for you to manage your own source control system or worry about scaling its infrastructure. You can use AWS CodeCommit to store anything from code to binaries. It supports the standard functionality of Git, so it works seamlessly with your existing Git-based tools.

AWS CodeCommit can create triggers that send notifications or automatically run code whenever a change occurs in your repository. Notifications can be sent from Amazon Simple Notification Service (Amazon SNS) or invoke AWS Lambda functions in response to the repository events you choose (e.g. commit to a branch, branch or tag creation, and branch or tag deletion). This helps you customize and automate your development workflow.

Using Amazon SNS, you can configure notifications (e.g. SMS or email) and create HTTP web hooks that are triggered by repository events. This lets you send notifications to users and other services. For example, you can create a trigger that will start a build on your continuous integration server using an HTTP web hook with Amazon SNS. You can also notify Amazon Simple Queue Service (SQS) about changes to your repositories using Amazon SNS notifications.

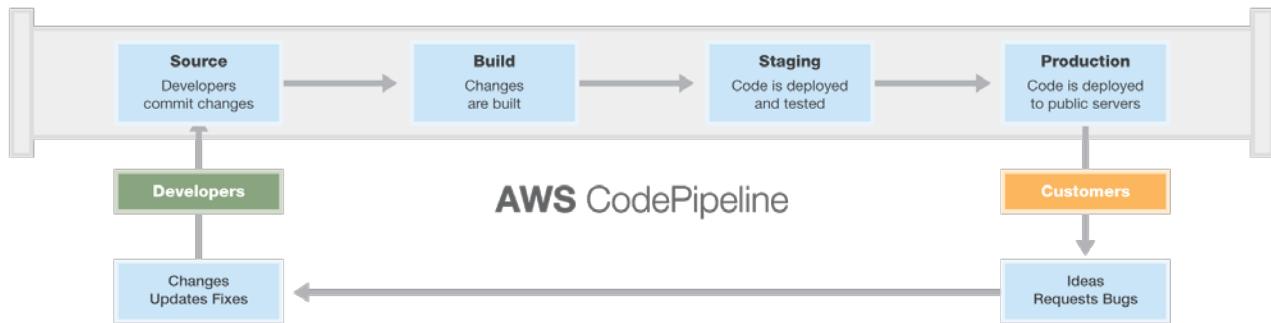
The following figure shows how to use your development machine, the AWS CLI or AWS CodeCommit console, and the AWS CodeCommit service to create and manage repositories:



AWS CODE PIPELINE

AWS CodePipeline is a continuous delivery service for fast and reliable application updates. CodePipeline builds, tests, and deploys your code every time there is a code change, based on the release process models you define. This enables you to rapidly and reliably deliver features and updates. You can easily build out an end-to-end solution by using the pre-built plugins for popular third-party services like GitHub or

integrating your own custom plugins into any stage of your release process. With AWS CodePipeline, you only pay for what you use. There are no upfront fees or long-term commitments.



The pipeline structure has the following requirements:

- A pipeline must contain at least two stages
- The first stage of a pipeline must contain at least one source action and can only contain source actions
- Only the first stage of a pipeline may contain source actions
- At least one stage in each pipeline must contain an action that is not a source action
- All stage names within a pipeline must be unique
- Stage names cannot be edited within the AWS CodePipeline console. If you edit a stage name by using the AWS CLI and the stage contains an action with one or more secret parameters (such as an OAuth token), the value of those secret parameters will not be preserved. You must manually type the value of the parameters (which are masked by four asterisks in the JSON returned by the AWS CLI) and include them in the JSON structure.

The version number of a pipeline is automatically generated and updated every time you update the pipeline.

AWS CodePipeline allows you to model the different stages of your software release process and each stages can have multiple actions. Here are the valid actions categories and providers.

Action Category	Providers
Approval	Manual Approval
Source	Amazon S3 AWS CodeCommit GitHub

Build	Jenkins Solano CI
Test	Jenkins Apica LoadTest BlazeMeter Ghost Inspector UI Testing HPE Strom Runner Load Runscope API Monitoring
Deploy	AWS CodeDeploy AWS Elastic Beanstalk AWS OpsWorks
Invoke	AWS Lambda

CODE DEPLOY

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. You can use AWS CodeDeploy to automate software deployments, eliminating the need for error-prone manual operations. The service also scales with your infrastructure so that you can easily deploy to one instance or thousands of instances.

Below is the AWS CodeDeploy lifecycle events flow diagram:

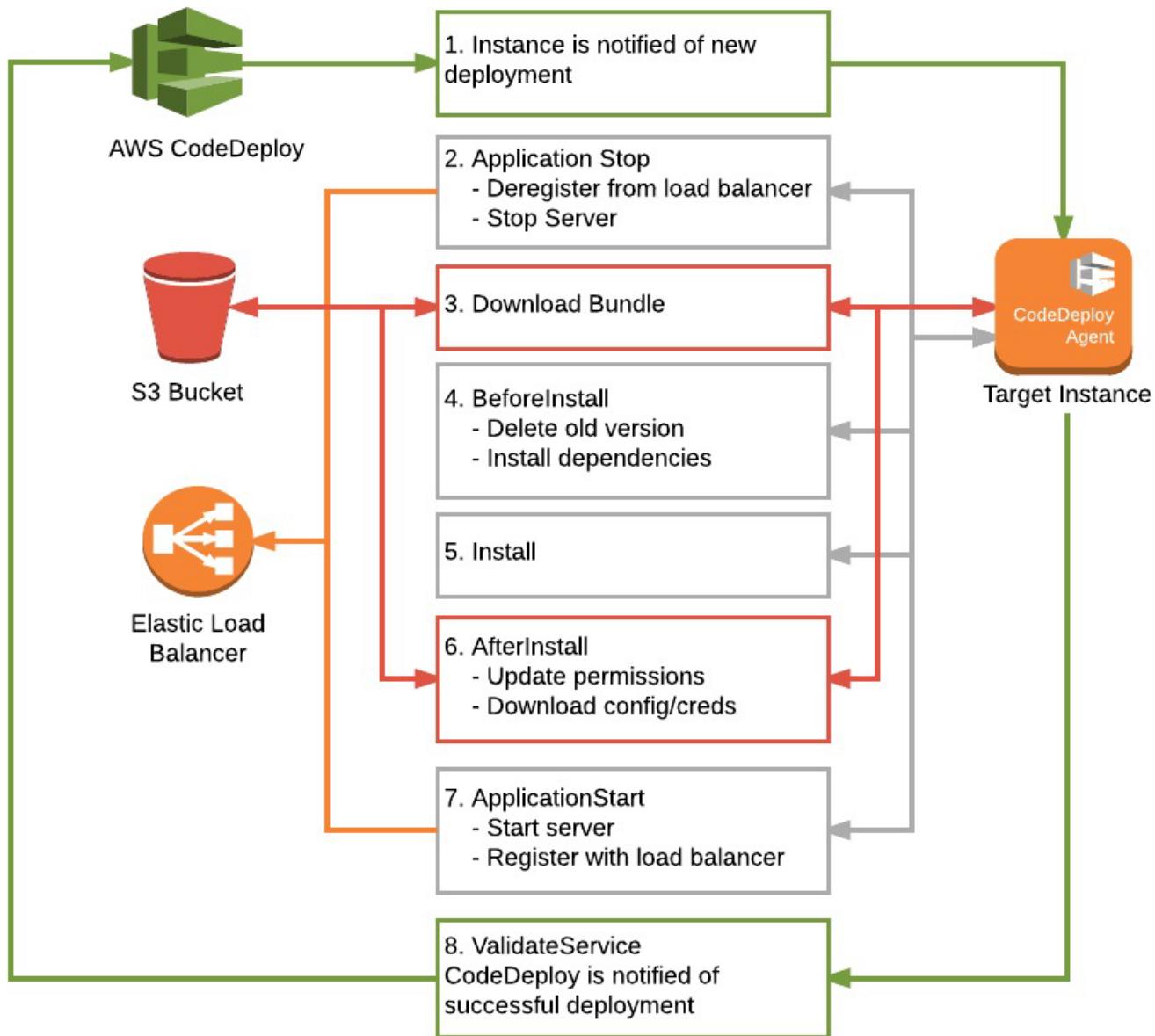


Figure 3: AWS CodeDeploy lifecycle events

EXAMPLE ARCHITECTURE DIAGRAM

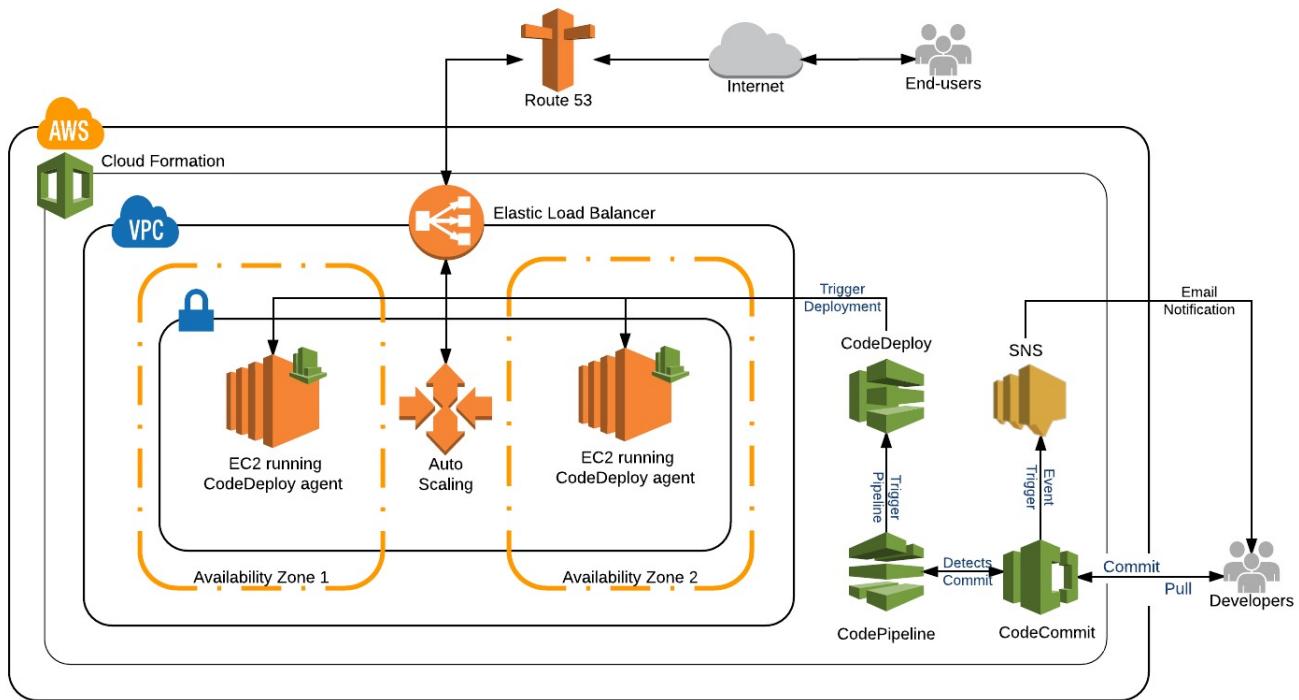


Figure 4: CI/CD on AWS Architecture Diagram

Hands-On Demonstration

This demonstration will walk you through how to create a CodeCommit repository, CodeDeploy, and CodePipeline using CloudFormation. This also includes the steps to connect to the CodeCommit repository and commit changes which trigger your deployment to the sample stack created.

PREREQUISITES

To follow along with the hands-on demonstration, you will need the following files, software, and resources:

- Download and unzip the example AWS Sample Linux App code to your local machine: https://s3-us-west-2.amazonaws.com/cis-samples/codecommit/SampleApp_Linux.zip
- Access to an AWS account
- AWS CLI tool installed on your local machine
Instructions: <http://docs.aws.amazon.com/cli/latest/userguide/installing.html>

To maintain focus on continuous integration and continuous deployment on AWS, we kindly ask that you please refer to the links above to download, install, and set up the required software, tools, and accounts before moving further.

Total Hands-On Estimated Duration: 1 Hour

- **CodeCommit Creation:** Approximately one-minute passive wait time to complete
- **CodeCommit Configurations:** Approximately 15 – 20 minutes
- **CodePipeline Creation:** Approximately 5 – 10 minutes passive wait time to complete, including the CodeDeploy creation and nested CloudFormation
- **CodePipeline/CodeDeploy Peek:** Approximately 5 – 10 minutes
- **Cleanup:** Approximately 20 minutes (including 10 minutes of passive wait time to complete)

Disclaimer

Users should note that, throughout this demonstration, you will be creating AWS resources in your own AWS account. This includes: the CodeCommit repository, EC2 Instances, and more. Your account will be charged for these resources accordingly at a standard hourly rate. With that in consideration, all templates and parameters were designed for minimum specifications to minimize costs.

For maximum cost efficiency, we recommend completing the hands-on demonstration end-to-end in one session and immediately terminate all templates when you have completed the demonstration.

CREATION OF CODECOMMIT REPOSITORY

In this section, we are going to create a new CodeCommit repository, connect to it, and make an initial commit. There are multiple ways to create a CodeCommit repository, we are going to use the AWS CloudFormation template. This template creates the CodeCommit repository, an IAM user to connect to repository, and a SNS topic configured to send email notifications on event triggers.

To proceed, please follow the steps below:

1. Log into the AWS Management Console for an account that you own.
2. Click the blue rectangle shape below. This will open up the CloudFormation page in the “US East (N.Virginia)” region, also known as “us-east-1”

[Click here to create a CodeCommit repository](#)
3. You should have been taken to the “Select Template” page after clicking the above link. Click “Next” to accept the defaults and proceed.
4. On the “Specify Details” page, provide a valid email address to receive an email notification for any repository event triggers (E.g.: commit to a branch, branch or tag creation, and branch or tag deletion) and click “Next” to proceed.
5. On the “Options” page, you can ignore the tagging options as it is optional and click “Next” to proceed.
6. You should now be on the “Review” page. Scroll down to find a checkbox with the following text: “**I acknowledge that AWS CloudFormation might create IAM resources.**” Check the box and click “Create”.
7. You should have been redirected to the CloudFormation management page. You should see a Stack Name such as “**CodeCommit-Demo**” that has a status ‘**CREATE_IN_PROGRESS**’
8. Wait for the status to update to ‘**CREATE_COMPLETE**’ in the AWS Management Console to complete the template. This may take a minute to complete.
9. Click on the check box next to the Stack Name “**CodeCommit-Demo**” and click on the “Output” tabs on the lower half of the browser window to view and see the outputs of CloudFormation which will be used in the subsequent section below.
10. Check your email for SNS notification subscription confirmation, click on the link to “Confirm subscription.” (Optional – to get the CodeCommit repository event trigger notifications)

CONNECT TO CODECOMMIT REPOSITORY

There are a couple of ways to connect to an existing CodeCommit repository. One way is via HTTPS, the other is via SSH. We're going to focus on connecting via SSH. For additional information, see [Connect to an AWS CodeCommit Repository](#): <http://docs.aws.amazon.com/codecommit/latest/userguide/how-to-connect.html>

Prerequisites

- Install Git (1.7.9 or later supported). If you don't have Git installed, install it now.

CloudFormation that was launched in the previous section would have created a IAM user "codecommit-demo". Now you need to create and attach a SSH public key to the IAM user.

To do that follow below steps:

1. Open a terminal and type the following command.

Note: These instructions are designed for Mac/Linux environments.

```
cd ~/.ssh  
ssh-keygen
```

2. When prompted after typing ssh-keygen, provide a name for SSH key E.g.: codecommit and leave all fields blank and just hit enter

3. Copy the public key, created from the previous step.

```
cat codecommit.pub
```

4. In AWS

Management Console, navigate to Identity and Access Management (IAM) and select "Users" in the left pane > On the right pane, click on the IAM user "codecommit-demo". Select the Security Credentials tab and scroll down to see "Upload SSH public key" button under SSH Keys for AWS CodeCommit. Click "Upload SSH public key" > paste the public key copied in the previous step and click "Upload SSH public key" button to upload.

Alternatively, if AWS CLI is installed in the local machine, use following command to upload the SSH public key.

```
aws iam upload-ssh-public-key --user-name "codecommit-demo" --ssh-public-key-body  
"<paste-public-key-here>"
```

5. Note down the "SSHPublicKeyId" from the previous step to be used in the next steps.

6. Open terminal and edit `~/.ssh/config`.

```
Host git-codecommit.*.amazonaws.com
User [SSHPublicKeyId]
IdentityFile ~/.ssh/codecommit
```

7. To verify your SSH connection works, type

```
ssh git-codecommit.us-east-1.amazonaws.com
```

8. Clone a local copy of the CodeCommit repo, that was created from the CodeCommit-Demo CloudFormation template and make an initial commit using following commands.

Note: SSH URL can be found in Output section of the CodeCommit-Demo CloudFormation.

```
#Change directory to user home
cd ~

#Clone CodeCommit repo
git clone ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/codecommit-demo

#Change directory
cd codecommit-demo

#Download the sample application
Wget
https://s3-us-west-2.amazonaws.com/cis-samples/codecommit/SampleApp\_Linux.zip

#Unzip
unzip SampleApp_Linux.zip

#Clean up
rm -f SampleApp_Linux.zip

#Commands to push the code to CodeCommit repo
git add -A

git commit -m "Initial Commit"

git push origin master
```

9. Note that you should have received an email notification, as a result of “commit” event trigger on CodeCommit repository.

CREATION OF CODEPIPELINE & CODEDEPLOY

In this section, we are going to create CodePipeline and CodeDeploy stacks using CloudFormation. The CodePipeline CloudFormation template creates a S3 bucket, CodePipeline, CodeDeploy stacks, required IAM roles, and policies. The CodePipeline template also triggers a nested CloudFormation template which is responsible for creating an EC2 instance ready for the CodeDeploy deployments.

Prerequisites

- The AWS account that's running the demonstration should have a Key Pair in the N. Virginia region. If not create one. For additional information, see [Creating a Key Pair](#):
<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html>

To proceed, please follow the steps below:

1. Log into the AWS Management Console for an account that you own.
2. Click the blue rectangle shape below. This will open up the CloudFormation page in the “US East (N.Virginia)” region, also known as “us-east-1”

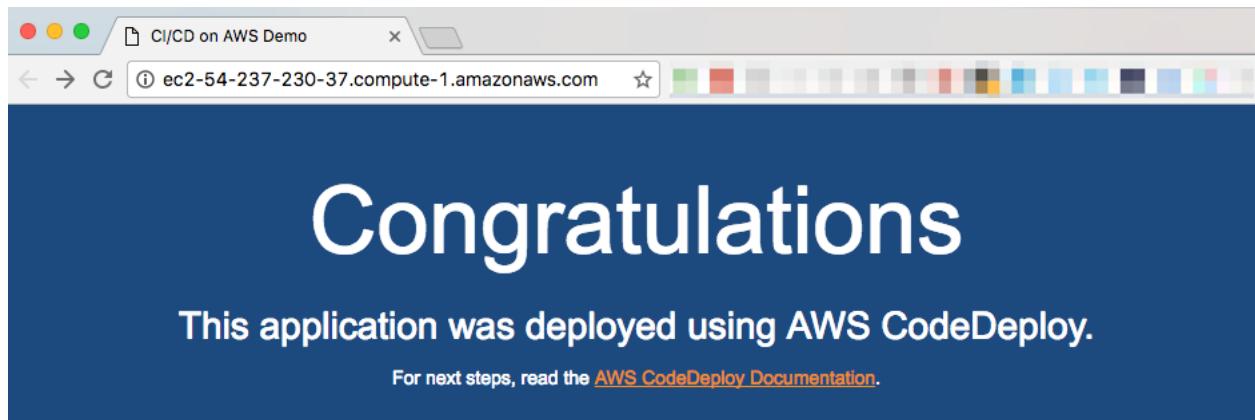
[Click here to create CodePipeline and CodeDeploy stacks](#)
3. You should have been taken to the “Select Template” page after clicking the above link. Click “Next” to accept the defaults and proceed.
4. On the “Specify Details” page, under the “Dynamic Configuration” section > select a EC2 Key Pair Name and click “Next” to proceed.
5. On the “Options” page, you can ignore the tagging options as it is optional and click “Next” to proceed.
6. You should now be on the “Review” page. Scroll down to find a checkbox with the following texts: “**I acknowledge that AWS CloudFormation might create IAM resources**” and “**I acknowledge that AWS CloudFormation might create IAM resources with custom names.**” Check the box and click “Create”.
7. You should have been redirected to the CloudFormation management page. You should see a Stack Name such as “**CodePipeline-Demo**” that has a status ‘**CREATE_IN_PROGRESS**’
8. “**CodePipeline-Demo**” triggers a nested CloudFormation stack such as “**CodePipeline-Demo-CodeDeployEC2InstancesStack-XXXXXXXXXXXX**”

9. Wait for the status to update to `CREATE_COMPLETE` in the AWS Management Console. “CodePipeline-Demo” and “CodePipeline-Demo-CodeDeployEC2InstancesStack-XXXXXXXXXXXX” combined may take about 6 – 10 minutes to complete.
10. Click on the check box next to the Stack Name “CodePipeline-Demo” and click on the “Output” tabs on the lower half of the browser window to view the outputs of CloudFormation which will be used in the subsequent section below.
11. Click on the check box next to the Stack Name “CodePipeline-Demo-CodeDeployEC2InstancesStack-XXXXXXXXXXXX” to view the outputs of this CloudFormation which will be used in the subsequent section below.

ACCESS THE APPLICATION

Once the “CodePipeline-Demo” CloudFormation stack’s status updates to `CREATE_COMPLETE`, scroll down to output tab of “CodeDeployEC2InstancesStack-XXXXXXXXXXXX” and click on the ApplicationURL to access the application.

Congratulations! Sample Linux code is deployed to the EC2 instances.



COMMIT CHANGES TO CODECOMMIT

To make some changes to the source code and commit changes to the CodeCommit repository and see these changes get deployed through CodePipeline, please do the following.

1. Navigate to the location, where the “codecommit-demo” repository was cloned.

2. Make following changes to the code

```
#Change directory to repo location  
cd ~/codecommit-demo  
  
#Edit index.html to make background color change  
sed -i.bak s/1A4073/94B823/ index.html && rm index.html.bak
```

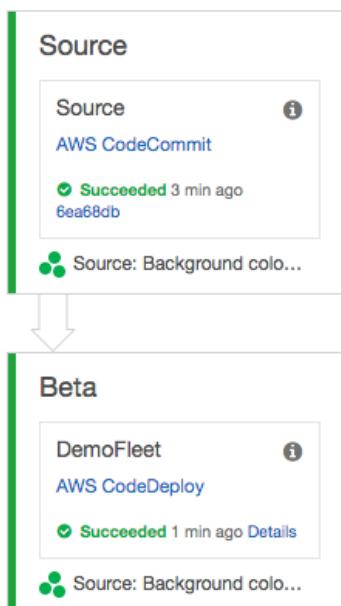
3. Commit the changes to the codecommit-demo repo

```
#Commands to push the code to CodeCommit repo  
git add -A  
  
git commit -m "Background color changed"  
  
git push origin master
```

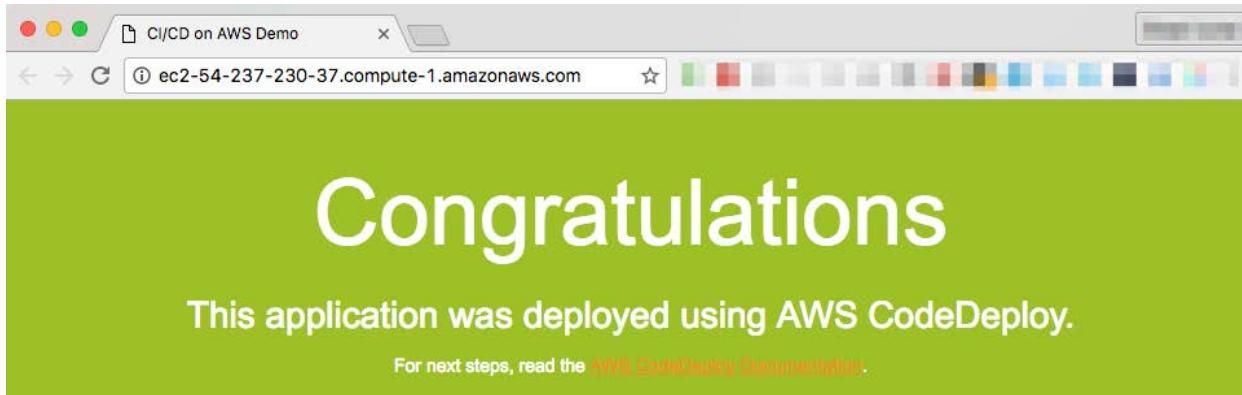
4. Once the code change is committed, CodePipeline will detect that the new code has been pushed to the CodeCommit repo and trigger the pipeline.

Note: Please wait a few seconds for CodePipeline to detect the changes, this will take less than a minute. Also note you should have received an email for code push to repo.

5. Reload the Application URL, once the pipeline status updates to succeeded.



6. Congratulations! Changes made were deployed to the EC2 instances.



CLEAN UP

When you are done with the demonstration, please closely follow the order of the cleanup steps below to avoid facing issue with CloudFormation deletion.

1. Empty S3 bucket "**codepipeline-demo-us-east-1-<your-aws-account-number>**"
 - a. In AWS Management Console, navigate to S3 in the AWS Management console, look for bucket "**codepipeline-demo-us-east-1-<your-aws-account-number>**". Select the bucket  and click on the actions dropdown from top menu section > choose **Empty bucket** to empty the bucket. CloudFormation cannot delete a bucket when it is not empty.
2. Delete "**CodePipeline-Demo**" CloudFormation stack
 - a. In AWS Management Console, navigate to CloudFormation. Click the checkbox for "**CodePipeline-Demo**" stack > click on the actions dropdown from top menu section > click "Delete Stack."
 - b. Wait for the "**CodePipeline-Demo**" stack reaches 'DELETE_COMPLETE' status and removed from the CloudFormation stack list.
3. Delete SSH key attached to CodeCommit IAM user "codecommit-demo".
 - a. In AWS Management Console, navigate to Identity and Access Management (IAM) and select Users in the left pane > On the right pane, click on the IAM user "codecommit-demo." Select Security Credentials tab and scroll down to see "SSH Keys for AWS Code Commit" section. Delete SSH Key that was uploaded in earlier step.
4. Delete "**CodeCommit-Demo**" CloudFormation stack
 - a. In AWS Management Console, navigate to CloudFormation. Click the checkbox for "**CodeCommit-Demo**" stack > click on the actions dropdown from top menu section > click "Delete Stack."
 - b. Wait for the "**CodeCommit-Demo**" stack reaches 'DELETE_COMPLETE' status and removed from the CloudFormation stack list.

5. Clean up local machine

a. Delete local copy of the repo

```
#Change directory to repo location  
cd ~  
rm -Rf codecommit-demo
```

b. Remove CodeCommit config in SSH configuration

```
vi ~/.ssh/config  
#Remove the three lines that look like below  
Host git-codecommit.*.amazonaws.com  
User [SSHPublicKeyId]  
IdentityFile ~/.ssh/codecommit
```

CONCLUSION

Today, faster software development has become a competitive advantage for many businesses with the automation of software development processes facilitating speed and consistency. Continuous integration and continuous delivery tools, such as CodeCommit, CodePipeline, and CodeDeploy are fully managed services hosted by AWS providing high service availability and durability, eliminating the administrative overhead of managing your own hardware and software. Tools like AWS CloudFormation and Jenkins CI will take the CI/CD to the next level of automation. This whitepaper addresses and covers CI/CD on AWS with a hands-on demonstration

Further Reading

Onica DevOps Practice

<https://www.onica.com/amazon-web-services/devops/>

AWS DevOps

<https://aws.amazon.com/devops/what-is-devops/>

AWS CodeCommit

<https://aws.amazon.com/codecommit/>

AWS CodePipeline

<https://aws.amazon.com/codepipeline/>

AWS CodeDeploy

<https://aws.amazon.com/codedeploy/>

References

1. AWS CodeCommit Documentation
<http://docs.aws.amazon.com/codecommit/latest/userguide/welcome.html>
2. AWS CodePipeline Documentation
<http://docs.aws.amazon.com/codepipeline/latest/userguide/welcome.html>
3. AWS CodeDeploy Documentation
<http://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html>
4. CloudFormation User Guide
<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html>