How - code commit, build, deploy, pipeline-Getting started overview

What Is AWS CodeCommit?

AWS CodeCommit is a version control service hosted by Amazon Web Services that you can use to privately store and manage assets (such as documents, source code, and binary files) in the cloud.

Introducing CodeCommit

CodeCommit is a secure, highly scalable, managed source control service that hosts private Git repositories. CodeCommit eliminates the need for you to manage your own source control system or worry about scaling its infrastructure. You can use 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.

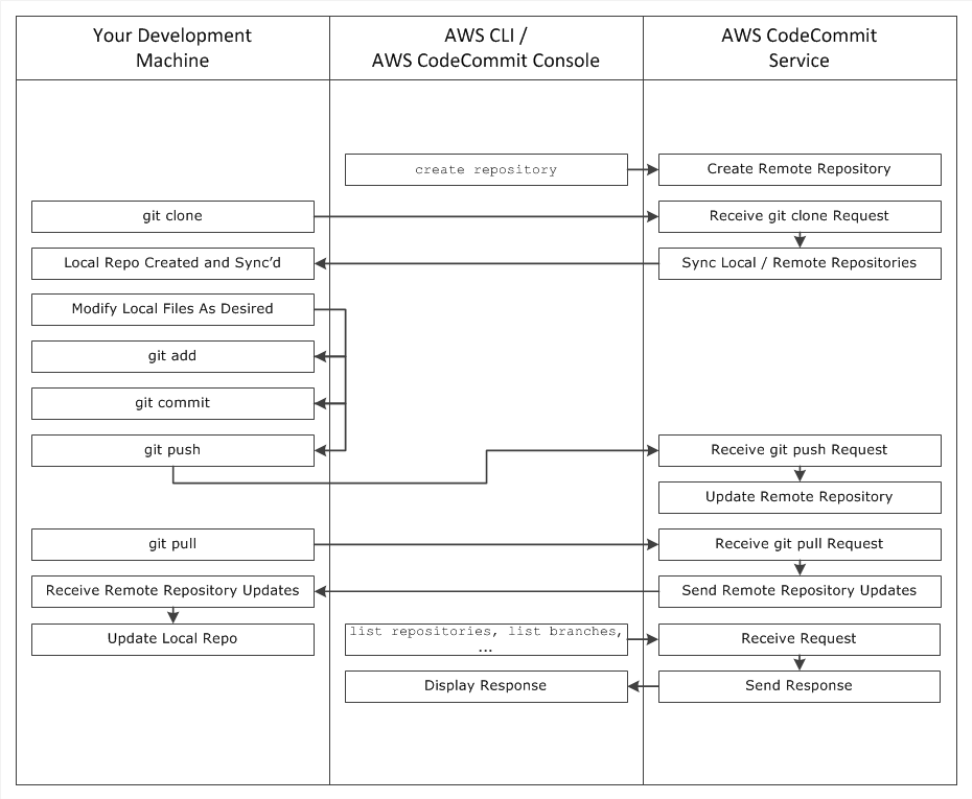
With CodeCommit, you can:

- **Benefit from a fully managed service hosted by AWS**. CodeCommit provides high service availability and durability and eliminates the administrative overhead of managing your own hardware and software. There is no hardware to provision and scale and no server software to install, configure, and update.
- **Store your code securely**. CodeCommit repositories are encrypted at rest as well as in transit.
- **Work collaboratively on code.** CodeCommit repositories support pull requests, where users can review and comment on each other's code changes before merging them to branches; notifications that automatically send emails to users about pull requests and comments; and more.
- **Easily scale your version control projects**. CodeCommit repositories can scale up to meet your development needs. The service can handle repositories with large numbers of files or branches, large file sizes, and lengthy revision histories.
- **Store anything, anytime**. CodeCommit has no limit on the size of your repositories or on the file types you can store.
- **Integrate with other AWS and third-party services**. CodeCommit keeps your repositories close to your other production resources in the AWS Cloud, which helps increase the speed and frequency of your development lifecycle. It is integrated with IAM and can be used with other AWS services and in parallel with other repositories. For more information, see Product and Service Integrations with AWS CodeCommit.
- **Easily migrate files from other remote repositories**. You can migrate to CodeCommit from any Git-based repository.
- **Use the Git tools you already know**. CodeCommit supports Git commands as well as its own AWS CLI commands and APIs.

How Does CodeCommit Work?

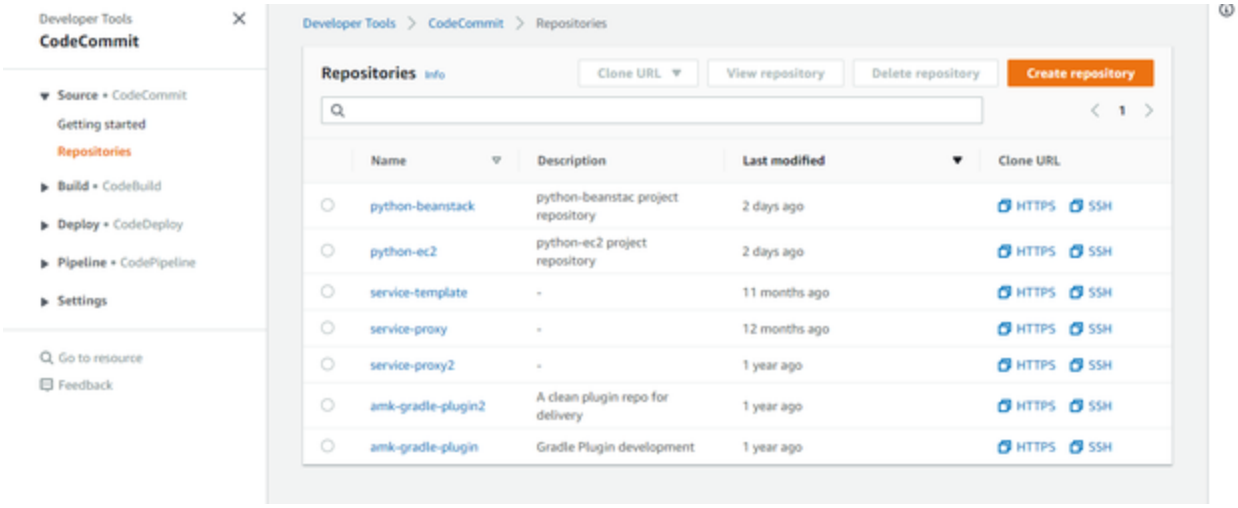
CodeCommit is familiar to users of Git-based repositories, but even those unfamiliar should find the transition to CodeCommit relatively simple. CodeCommit provides a console for the easy creation of repositories and the listing of existing repositories and branches. In a few simple steps, users can find information about a repository and clone it to their computer, creating a local repo where they can make changes and then push them to the CodeCommit repository. Users can work from the command line on their local machines or use a GUI-based editor.

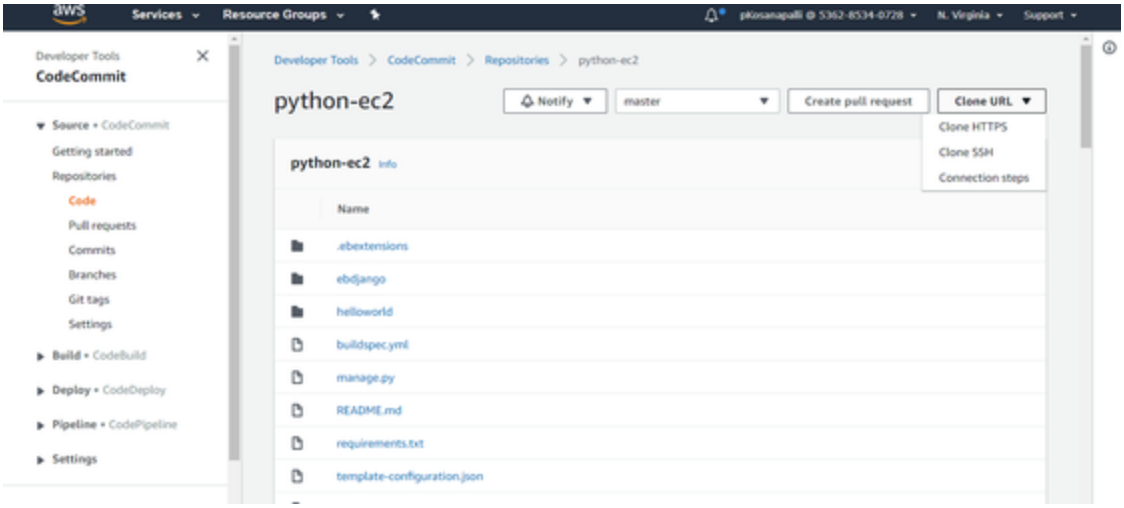
The following figure shows how you use your development machine, the AWS CLI or CodeCommit console, and the CodeCommit service to create and manage repositories:



1. Use the AWS CLI or the CodeCommit console to create a CodeCommit repository.
2. From your development machine, use Git to run **git clone**, specifying the name of the CodeCommit repository. This creates a local repo that connects to the CodeCommit repository.
3. Use the local repo on your development machine to modify (add, edit, and delete) files, and then run **git add** to stage the modified files locally. Run **git commit** to commit the files locally, and then run **git push** to send the files to the CodeCommit repository.
4. Download changes from other users. Run **git pull** to synchronize the files in the CodeCommit repository with your local repo. This ensures you're working with the latest version of the files.

You can use the AWS CLI or the CodeCommit console to track and manage your repositories.





How Is CodeCommit Different from File Versioning in Amazon S3?

CodeCommit is optimized for team software development. It manages batches of changes across multiple files, which can occur in parallel with changes made by other developers. Amazon S3 versioning supports the recovery of past versions of files, but it's not focused on collaborative file tracking features that software development teams need.

What Is AWS CodeBuild?

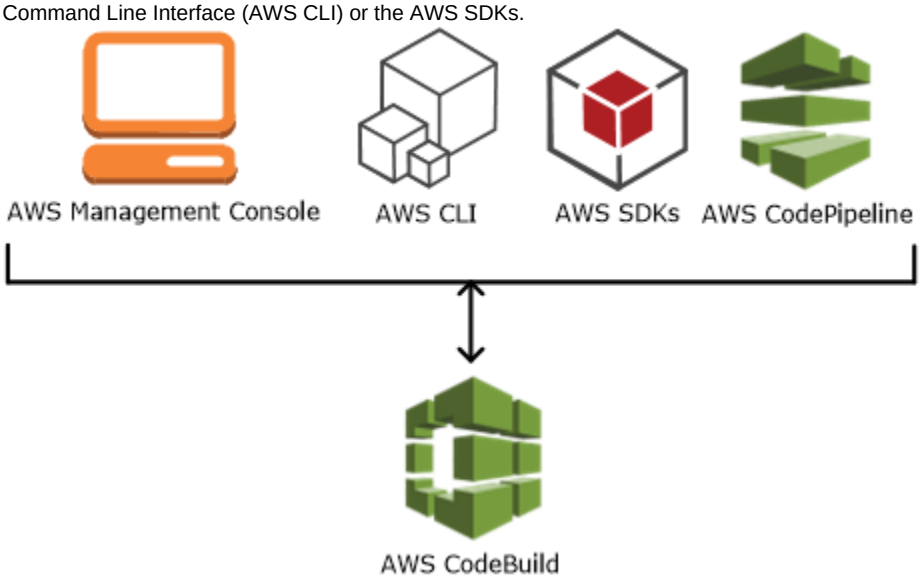
AWS CodeBuild is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy. CodeBuild eliminates the need to provision, manage, and scale your own build servers. It provides prepackaged build environments for popular programming languages and build tools such as Apache Maven, Gradle, and more. You can also customize build environments in CodeBuild to use your own build tools. CodeBuild scales automatically to meet peak build requests.

CodeBuild provides these benefits:

- **Fully managed** – CodeBuild eliminates the need to set up, patch, update, and manage your own build servers.
- **On demand** – CodeBuild scales on demand to meet your build needs. You pay only for the number of build minutes you consume.
- **Out of the box** – CodeBuild provides preconfigured build environments for the most popular programming languages. All you need to do is point to your build script to start your first build.

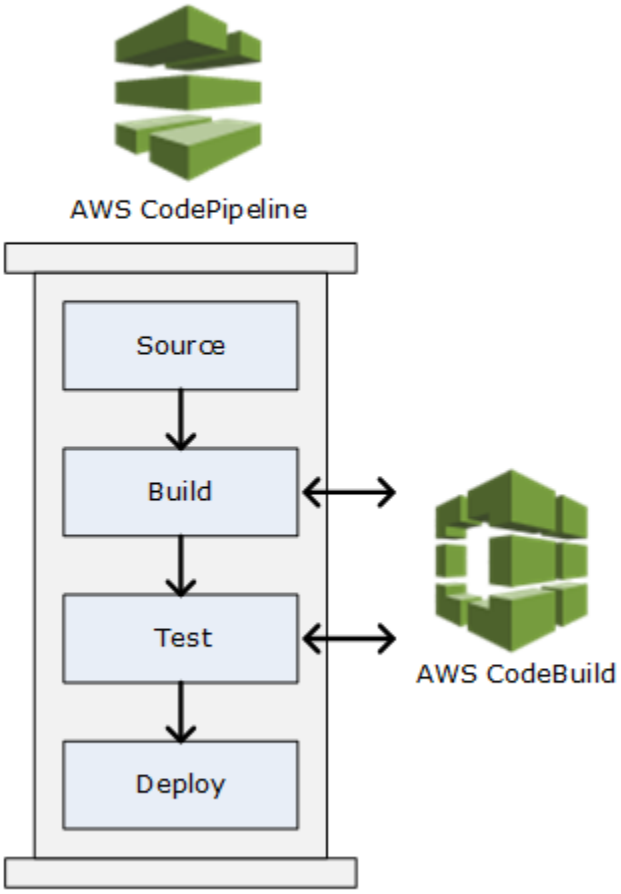
How to Run CodeBuild

You can run CodeBuild by using the CodeBuild or AWS CodePipeline console. You can also automate the running of CodeBuild by using the AWS Command Line Interface (AWS CLI) or the AWS SDKs.



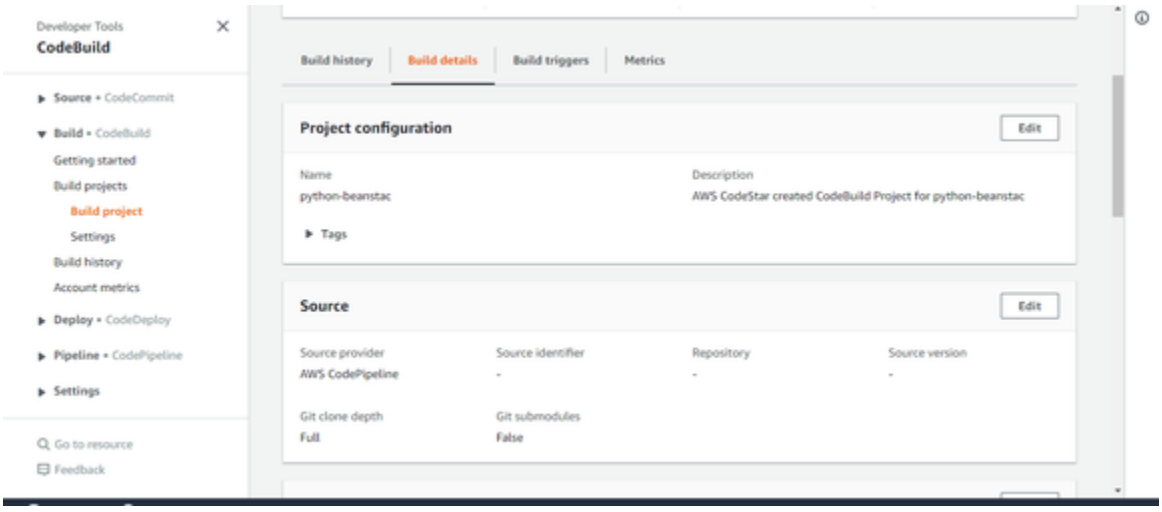
To run CodeBuild by using the CodeBuild console, AWS CLI, or AWS SDKs.

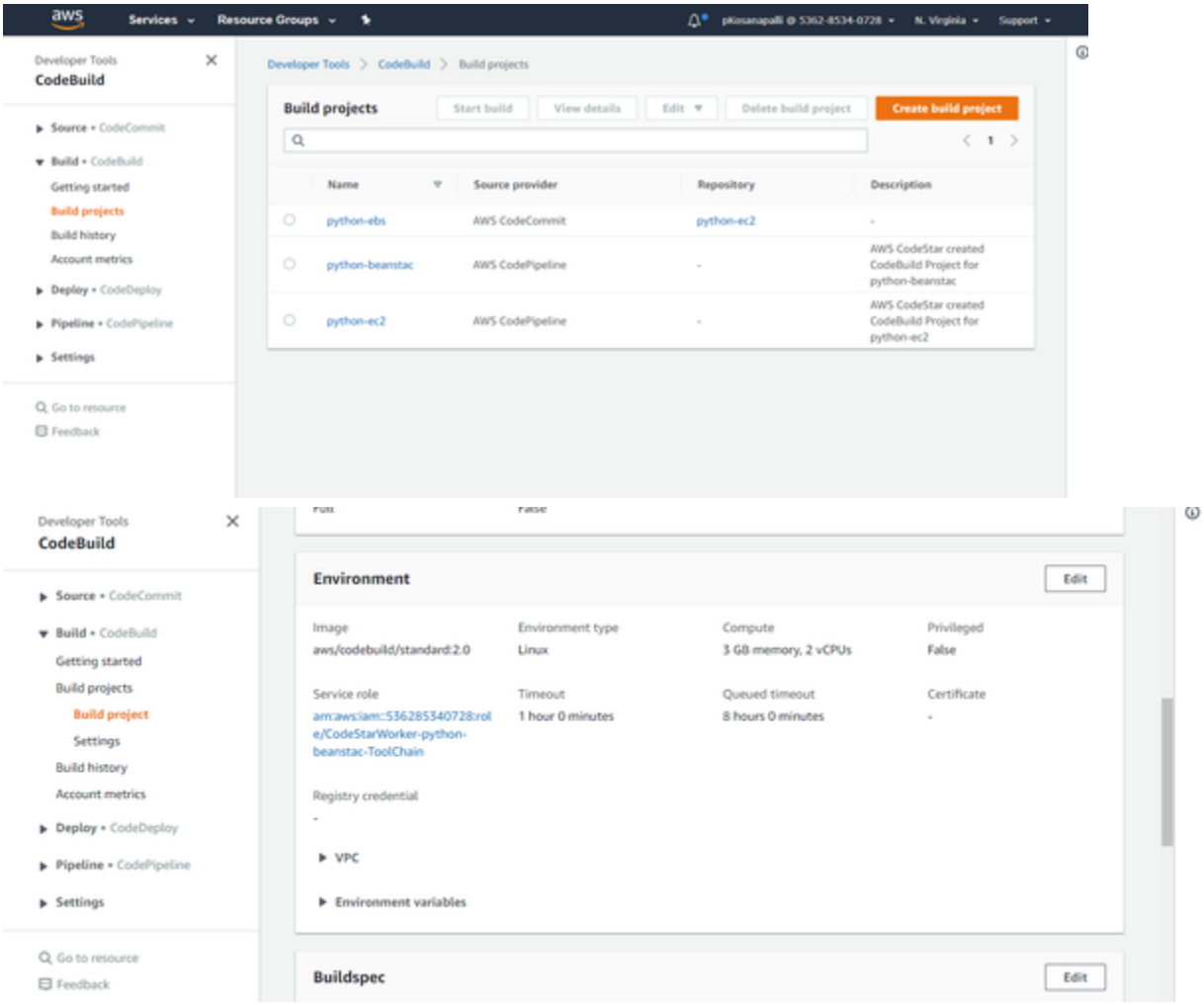
As the following diagram shows, you can add CodeBuild as a build or test action to the build or test stage of a pipeline in AWS CodePipeline. AWS CodePipeline is a continuous delivery service that enables you to model, visualize, and automate the steps required to release your code. This includes building your code. A *pipeline* is a workflow construct that describes how code changes go through a release process.



To use CodePipeline to create a pipeline and then add a CodeBuild build or test action, see Use AWS CodePipeline with CodeBuild.

The CodeBuild console also provides a way to quick rch 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 Viewing Resources in the Console.





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.

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.

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.
- **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.
- **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.

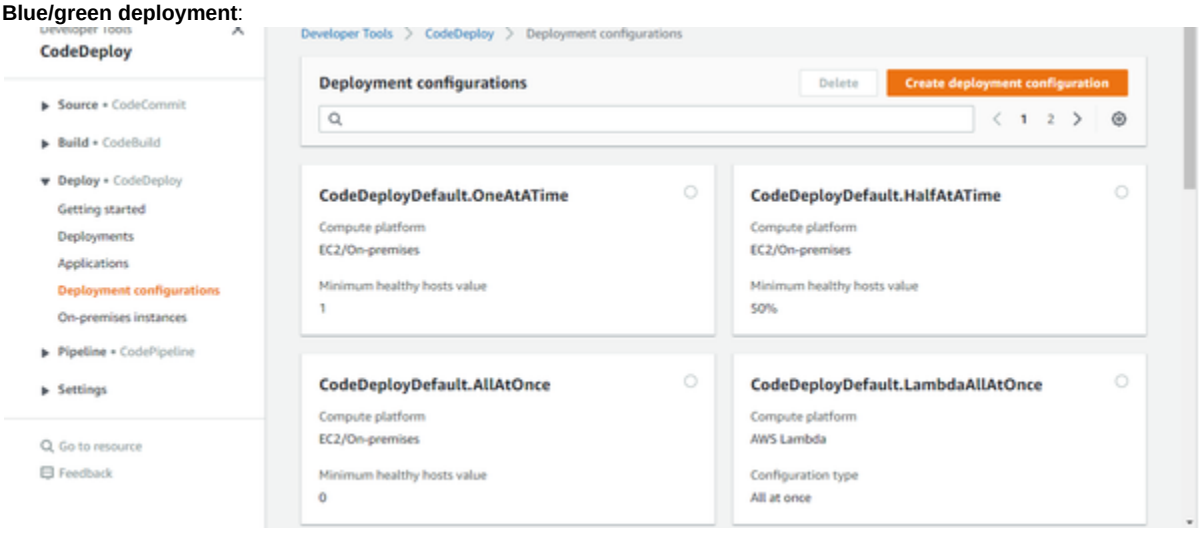
The following table describes how CodeDeploy components are used with each compute platform.

- 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 Deployment Types

CodeDeploy provides two deployment type options:

- **In-place deployment**:
- **Blue/green deployment**:



What Is AWS CodePipeline?

AWS CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software. You can quickly model and configure the different stages of a software release process. CodePipeline automates the steps required to release your software changes continuously.

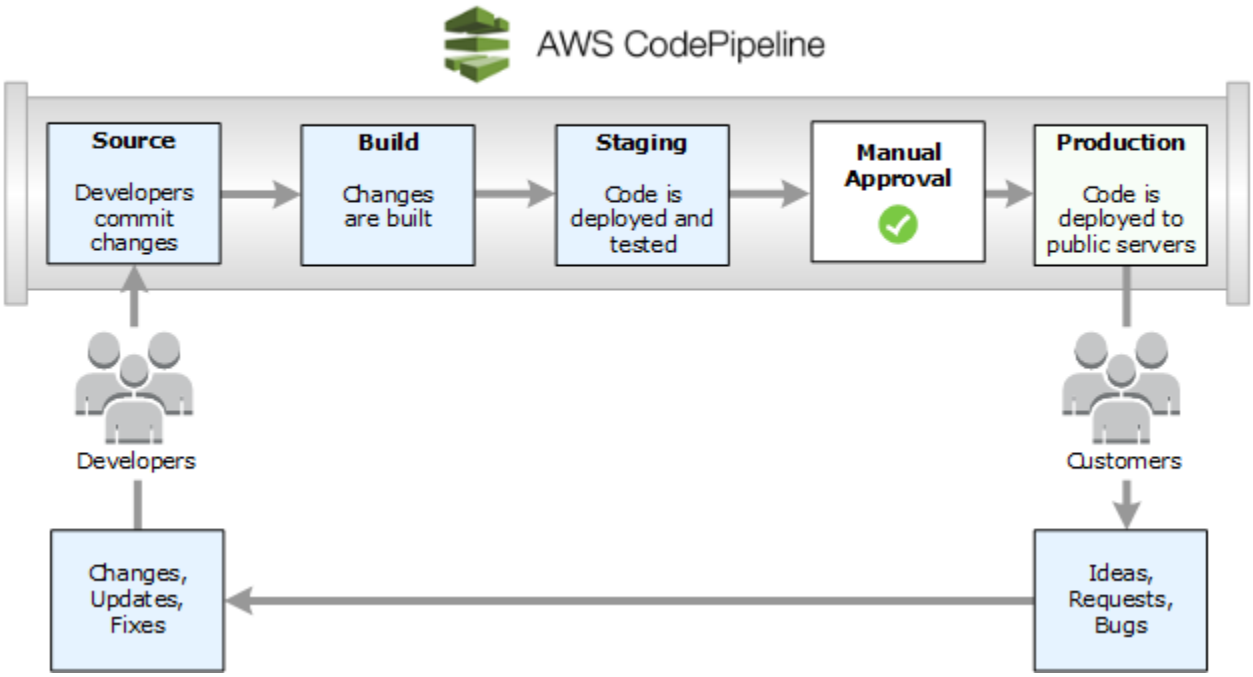
What Can I Do with CodePipeline?

You can use CodePipeline to help you automatically build, test, and deploy your applications in the cloud. Specifically, you can:

- **Automate your release processes**: CodePipeline fully automates your release process from end to end, starting from your source repository through build, test, and deployment. You can prevent changes from moving through a pipeline by including a manual approval action in any stage except a Source stage. You can release when you want, in the way you want, on the systems of your choice, across one instance or multiple instances.
- **Establish a consistent release process**: Define a consistent set of steps for every code change. CodePipeline runs each stage of your release according to your criteria.
- **Speed up delivery while improving quality**: You can automate your release process to allow your developers to test and release code incrementally and speed up the release of new features to your customers.
- **Use your favorite tools**: You can incorporate your existing source, build, and deployment tools into your pipeline. For a full list of AWS services and third-party tools currently supported by CodePipeline, see Product and Service Integrations with CodePipeline.
- **View progress at a glance**: You can review real-time status of your pipelines, check the details of any alerts, retry failed actions, view details about the source revisions used in the latest pipeline execution in each stage, and manually rerun any pipeline.
- **View pipeline history details**: You can view details about executions of a pipeline, including start and end times, run duration, and execution IDs.

A Quick Look at CodePipeline

The following diagram shows an example release process using CodePipeline.



In this example, when developers commit changes to a source repository, CodePipeline automatically detects the changes. Those changes are built, and if any tests are configured, those tests are run. After the tests are complete, the built code is deployed to staging servers for testing. From the staging server, CodePipeline runs additional tests, such as integration or load tests. Upon the successful completion of those tests, and after a manual approval action that was added to the pipeline is approved, CodePipeline deploys the tested and approved code to production instances.

CodePipeline can deploy applications to Amazon EC2 instances by using CodeDeploy, AWS Elastic Beanstalk, or AWS OpsWorks Stacks. CodePipeline can also deploy container-based applications to services by using Amazon ECS. Developers can also use the integration points provided with CodePipeline to plug in other tools or services, including build services, test providers, or other deployment targets or systems.

A pipeline can be as simple or as complex as your release process requires.

A Quick Look at Input and Output Artifacts

CodePipeline integrates with development tools to check for code changes and then build and deploy through all stages of the continuous delivery process.

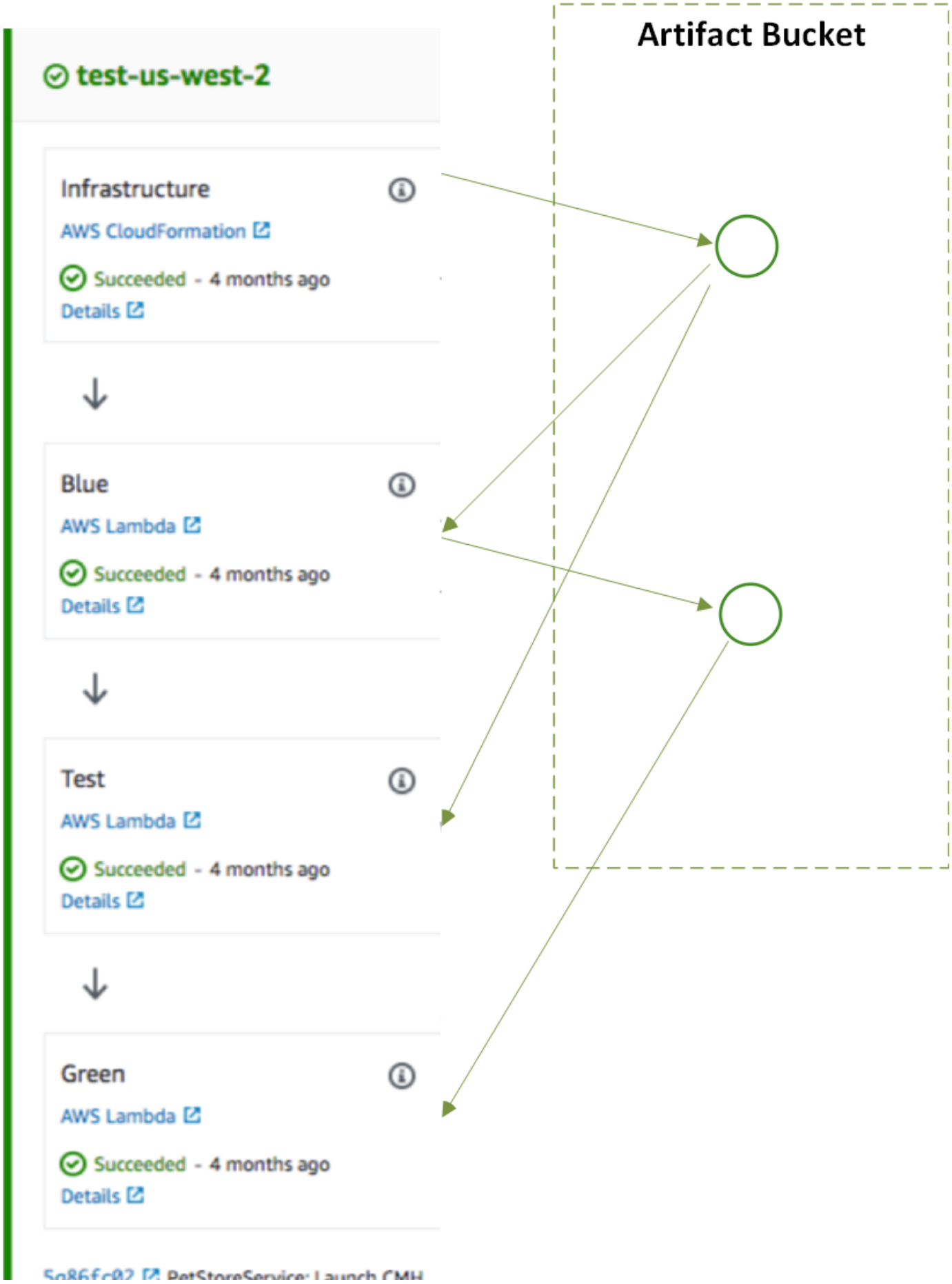
Stages use input and output artifacts that are stored in the artifact store for your pipeline. An artifact store is an Amazon S3 bucket. Your artifact store is in the same AWS Region as the pipeline to store items for all pipelines in that Region associated with your account. Every time you use the console to create another pipeline in that Region, CodePipeline creates a folder for that pipeline in the bucket. It uses that folder to store artifacts for your pipeline as the automated release process runs. When you create or edit a pipeline, you must have an artifact bucket in the pipeline Region and then you must have one artifact bucket per Region where you are executing an action.

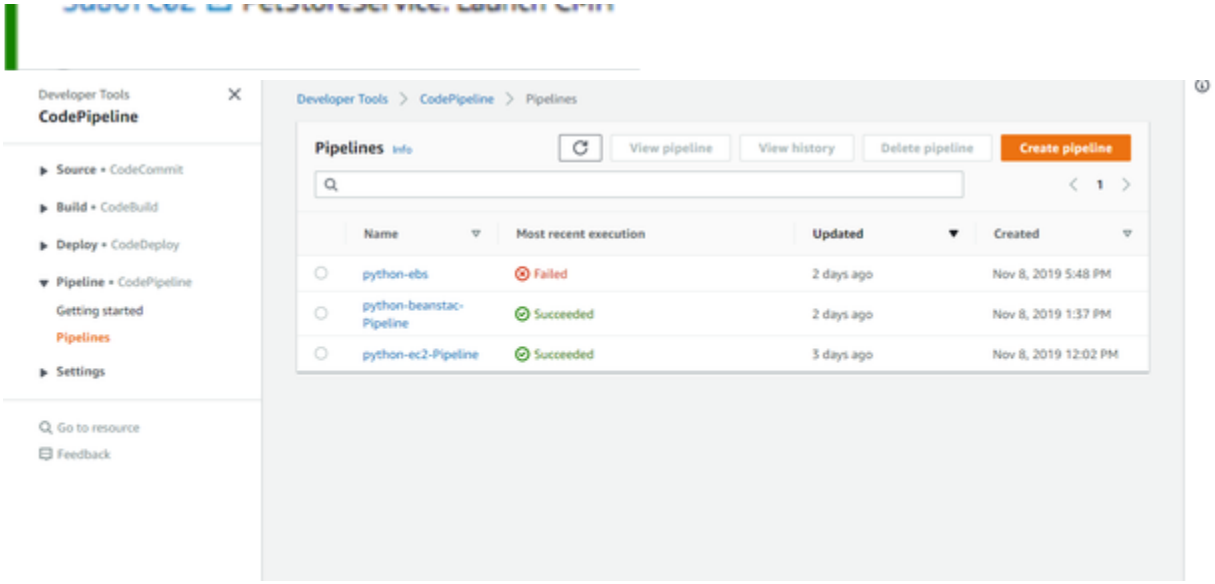
CodePipeline zips and transfers the files for input or output artifacts as appropriate for the action type in the stage. For example, at the start of a build action, CodePipeline retrieves the input artifact (any files to be built) and provides the artifact to the build action. After the action is complete, CodePipeline takes the output artifact (the built application) and saves it to the output artifact bucket for use in the next stage.

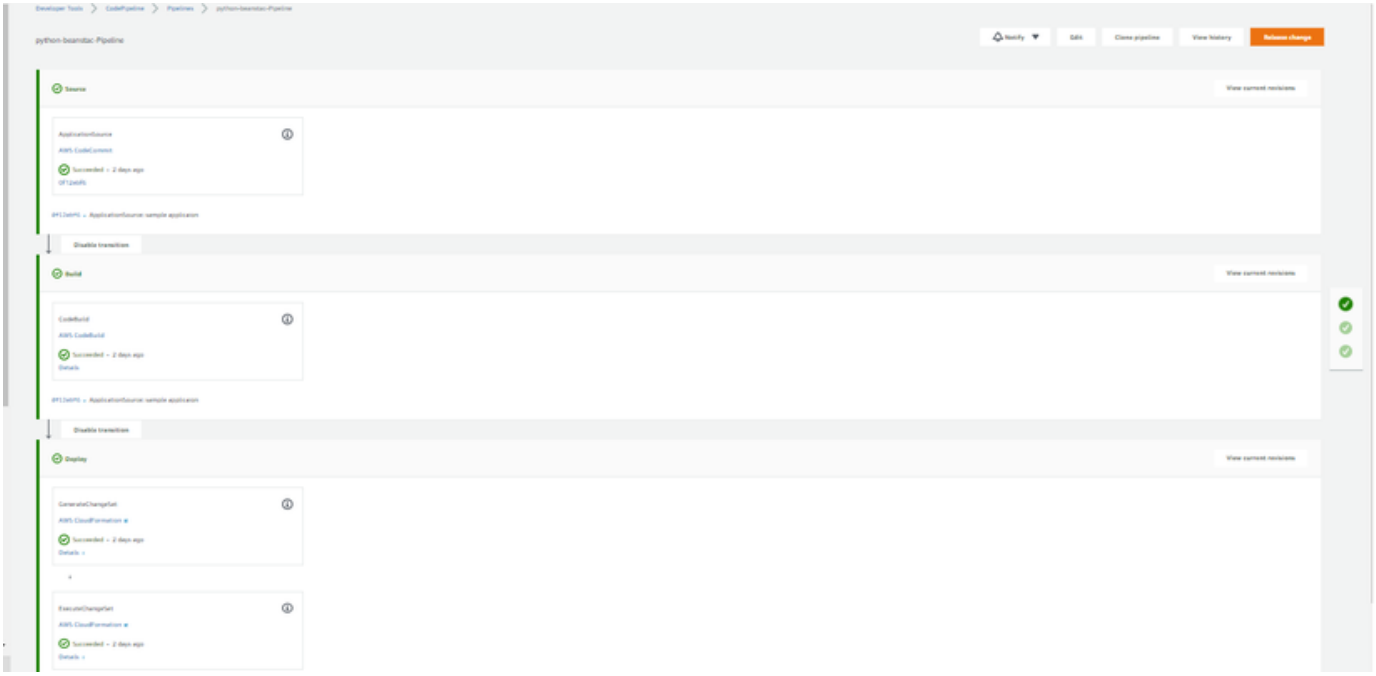
When you use the **Create Pipeline** wizard to configure or choose stages:

1. CodePipeline triggers your pipeline to run when there is a commit to the source repository, providing the output artifact from the Source stage.
2. The output artifact from the previous step is ingested as an input artifact to the Build stage. An output artifact from the Build stage can be an updated application or an updated Docker image built to a container.
3. The output artifact from the previous step is ingested as an input artifact to the Deploy stage, such as staging or production environments in the AWS Cloud. You can deploy applications to a deployment fleet, or you can deploy container-based applications to tasks running in Amazon ECS clusters.

The following diagram shows a high-level artifact workflow between stages in CodePipeline.


Artifact Bucket
test-us-west-2
Infrastructure
AWS CloudFormation
Succeeded - 4 months ago
Details
Blue
AWS Lambda
Succeeded - 4 months ago
Details
Test
AWS Lambda
Succeeded - 4 months ago
Details
Green
AWS Lambda
Succeeded - 4 months ago
Details





AWS CodePipeline Adds Support for Amazon ECS and AWS Fargate

- AWS CodePipelinenow supports deployments toAmazon Elastic Container Service(Amazon ECS) andAWS Fargate. This makes it easy to create acontinuous deliverypipeline for container-based applications.
- When you push a code change, AWS CodePipeline automatically calls the specified services to create your Docker image, run tests, and update your running containers. For example, you can create a pipeline where a code change in AWS CodeCommit triggers AWS CodeBuild to build a new image and upload it to Amazon Elastic Container Registry.
- CodePipeline then automatically instructs Amazon ECS to deploy the updated containers onto your managed cluster with Amazon ECS or AWS Fargate.
- Additionally, CodePipeline integrates with many third party CI/CD services, so you can use a pipeline with your existing tools.

thats all done !!!