

**EXPERIMENT NO. 02**

Aim: To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

Theory:**AWSCodePipeline:**

Continuous deployment allows you to deploy revisions to a production environment automatically without explicit approval from a developer, making the entire software release process automated. You will create the pipeline using AWS CodePipeline, a service that builds, tests, and deploys your code every time there is a code change. You will use your GitHub account, an Amazon Simple Storage Service (S3) bucket, or an AWS CodeCommit repository as the source location for the sample app's code. You will also use AWS Elastic Beanstalk as the deployment target for the sample app. Your completed pipeline will be able to detect changes made to the source repository containing the sample app and then automatically update your live sample app.

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.

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 you can use 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. In this Experiment, you use AWS CodeBuild to build a collection of sample source

code input files (build input artifacts or build input) into a deployable version of the source code (build output artifact or build output). Specifically, you instruct CodeBuild to use Apache Maven, a common build tool, to build a set of Java class files into a Java Archive (JAR) file.

Steps:

1. Create the source code :

In this step, you create the source code that you want CodeBuild to build to the output bucket. This source code consists of two Java class files and an Apache Maven Project Object Model (POM) file.

- a. Using a text editor of your choice, create this file, name it *MessageUtil.java*, and then save it in the *src/main/java* directory. This class file creates as output the string of characters passed into it. The *MessageUtil* constructor sets the string of characters. The *printMessage* method creates the output. The salutation Message method outputs Hi! followed by the string of characters.
- b. Create this file, name it *TestMessageUtil.java*, and then save it in the */src/test/java* directory. This class file sets the message variable in the *MessageUtil* class to Robert. It then tests to see if the message variable was successfully set by checking whether the strings Robert and *Hi!Robert* appear in the output.
- c. Create this file, name it *pom.xml*, and then save it in the root (top level) directory. Apache Maven uses the instructions in this file to convert the *MessageUtil.java* and *TestMessageUtil.java* files into a file named *messageUtil-1.0.jar* and then run the specified tests.

2. Create the buildspec file:

In this step, you create a build specification (build spec) file. A buildspec is a collection of build commands and related settings, in YAML format, that CodeBuild uses to run a build. Without a build spec, CodeBuild cannot successfully convert your build input into build output or locate the build output artifact in the build environment to upload to your output bucket. Create this file, name it *buildspec.yml*, and then save it in the root (top level) directory.

```

version: 0.2

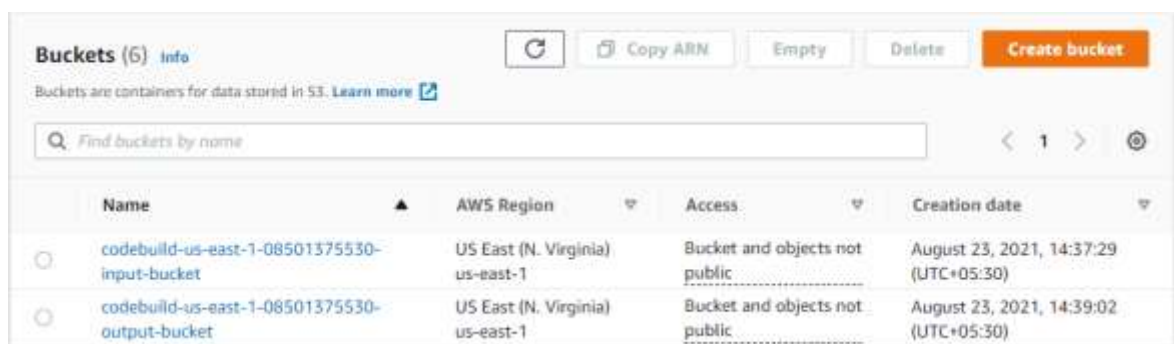
phases:
  install:
    runtime-versions:
      java: corretto11
  pre_build:
    commands:
      - echo Nothing to do in the pre_build phase...
  build:
    commands:
      - echo Build started on `date`
      - mvn install
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - target/messageUtil-1.0.jar

```

3. Create two S3 buckets

Although you can use a single bucket for this tutorial, two buckets makes it easier to see where the build input is coming from and where the build output is going.

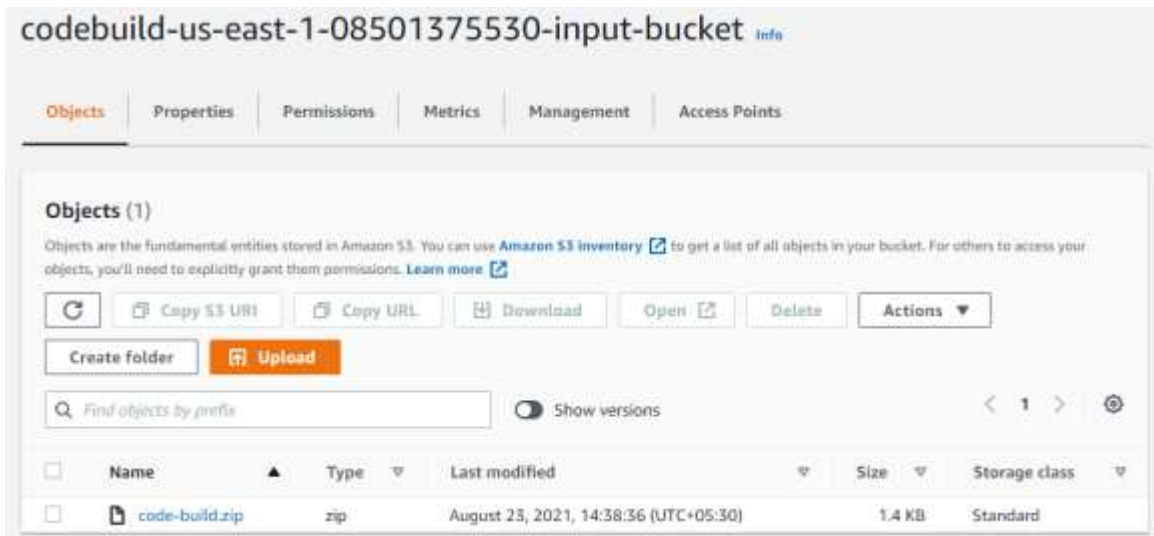
- One of these buckets (the input bucket) stores the build input. In this tutorial, the name of this input bucket is codebuild-region-ID-account-ID-input-bucket, where region-ID is the AWS Region of the bucket and account-ID is your AWS account ID.
- The other bucket (the output bucket) stores the build output. In this tutorial, the name of this output bucket is codebuild-region-ID-account-ID-output-bucket.



Name	AWS Region	Access	Creation date
codebuild-us-east-1-08501375530-input-bucket	US East (N. Virginia) us-east-1	Bucket and objects not public	August 23, 2021, 14:37:29 (UTC+05:30)
codebuild-us-east-1-08501375530-output-bucket	US East (N. Virginia) us-east-1	Bucket and objects not public	August 23, 2021, 14:39:02 (UTC+05:30)

4. Upload the source code and the buildspec file

In this step, you add the source code and build spec file to the input bucket. Using your operating system's zip utility, create a file named MessageUtil.zip that includes MessageUtil.java, TestMessageUtil.java, pom.xml, and buildspec.yml.



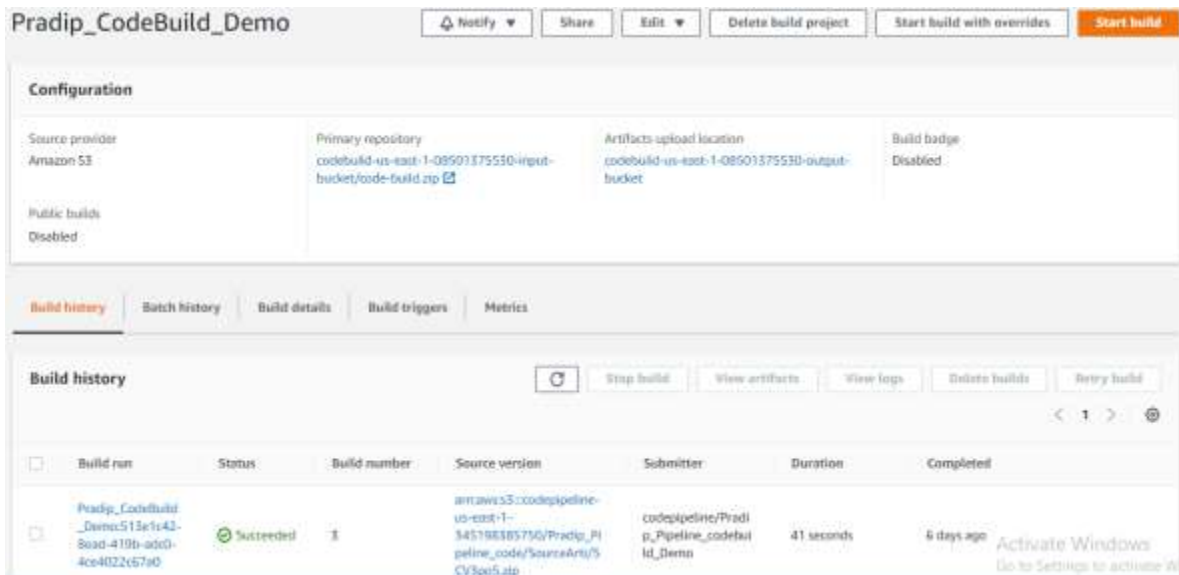
5. Create the build project

In this step, you create a build project that AWS CodeBuild uses to run the build. A *build project* includes information about how to run a build, including where to get the source code, which build environment to use, which build commands to run, and where to store the build output. A *build environment* represents a combination of operating system, programming language runtime, and tools that CodeBuild uses to run a build.

To create the build project

1. Sign in to the AWS Management Console and open the AWS CodeBuild console at <https://console.aws.amazon.com/codesuite/codebuild/home>.
2. Use the AWS region selector to choose an AWS Region where CodeBuild is supported.
3. If a CodeBuild information page is displayed, choose **Create build project**. Else, on the navigation pane, expand **Build**, choose **Build projects**, choose **Create build project**.
4. On the **Create build project** page, in **Project configuration**, for **Project name**, enter a name for this build project. Build project names must be unique across each AWS account. If you use a different name, be sure to use it throughout this tutorial. In **Source**, for **Source provider**, choose **Amazon S3**.
5. For **Bucket**, choose **codebuild-region-ID-account-ID-input-bucket**.
6. For **S3 object key**, enter **MessageUtil.zip**.
7. In **Environment**, for **Environment image**, leave **Managed image** selected.
8. For **Operating system**, choose **Amazon Linux 2**.
9. For **Runtime(s)**, choose **Standard**.

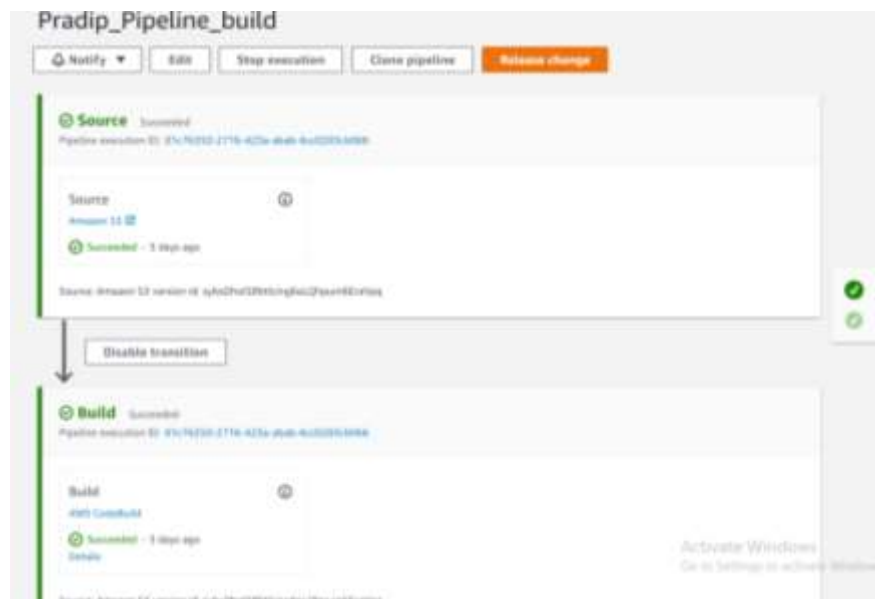
10. For **Image**, choose **aws/codebuild/amazonlinux2-x86_64-standard:3.0**.
11. In **Service role**, leave **New service role** selected, and leave **Role name** unchanged.
12. For **Buildspec**, leave **Use a buildspec file** selected.
13. In **Artifacts**, for **Type**, choose **Amazon S3**.
14. For **Bucket name**, choose **codebuild-region-ID-account-ID-output-bucket**.
15. Leave **Name** and **Path** blank.
16. Choose **Create build project**.



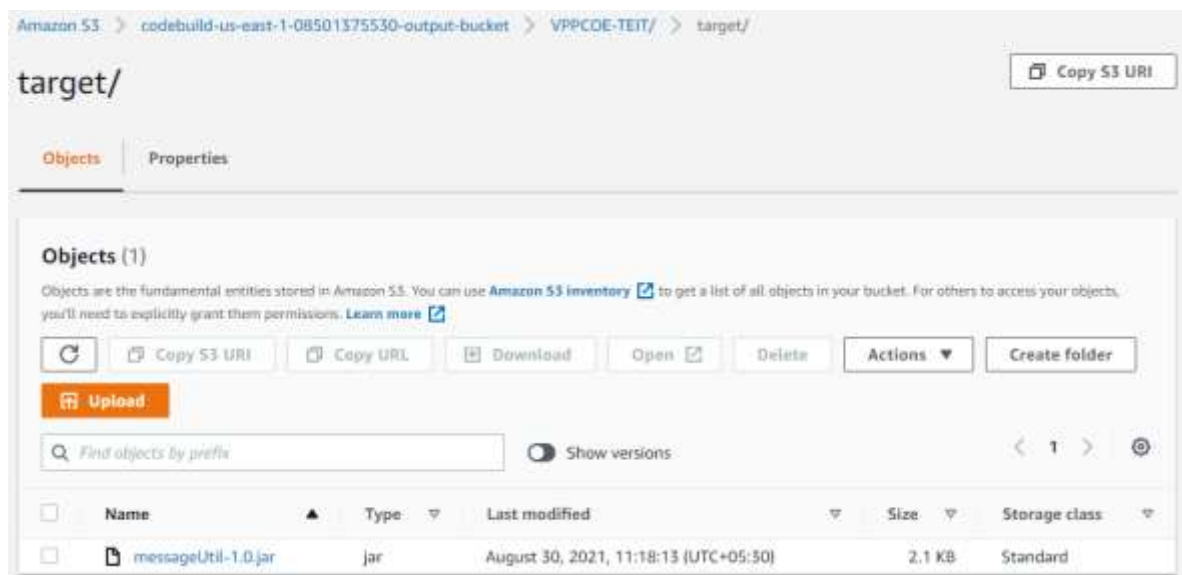
6. Create pipeline

1. Open the AWS CodePipeline console at <https://console.aws.amazon.com/codesuite/codepipeline/home>
2. In the AWS Region selector, choose the AWS Region where your build project AWS resources are located. This must be an AWS Region where CodeBuild is supported. For more information, see [AWS CodeBuild](#) in the *Amazon Web Services General Reference*.
3. Create a pipeline. If a CodePipeline information page is displayed, choose **Create pipeline**. If a **Pipelines** page is displayed, choose **Create pipeline**.
4. On the **Step 1: Choose pipeline settings** page, for **Pipeline name**, enter a name for the pipeline. If you choose a different name, be sure to use it throughout this procedure.
5. For **Role name**, Choose **Existing service role**, and then choose the CodePipeline service role you created or identified as part of this topic's prerequisites.

6. For **Artifact store**, Choose Custom location if you already have an existing artifact store you have created, such as an S3 artifact bucket, in the same AWS Region as your pipeline. Choose Next.
7. On the Step 2: Add source stage page,
If your source code is stored in an S3 bucket, choose Amazon S3. For Bucket, select the S3 bucket that contains your source code. For S3 object key, enter the name of the file contains the source code (for example, file-name.zip). Choose Next.
8. On the Step 3: Add build stage page, for Build provider, choose CodeBuild.
9. If you already have a build project you want to use, for Project name, choose the name of the build project and skip to the next step in this procedure. If you choose an existing build project, it must have build output artifact settings already defined.
10. On the **Step 4: Add deploy stage** page, do one of the following:
 - If you do not want to deploy the build output artifact, choose **Skip**, and confirm this choice when prompted.
 - If you want to deploy the build output artifact, for **Deploy provider**, choose a deployment provider, and then specify the settings when prompted.
 - Choose **Next**.
11. On the **Review** page, review your choices, and then choose **Create pipeline**.
12. After the pipeline runs successfully, you can get the build output artifact. With the pipeline displayed in the CodePipeline console, in the **Build** action, choose the tooltip. Make a note of the value for **Output artifact**.



13. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
14. In the list of buckets, open the bucket used by the pipeline. The name of the bucket should follow the format `codepipeline-region-ID-random-number`. You can use the AWS CLI to run the CodePipeline **get-pipeline** command to get the name of the bucket, where `my-pipeline-name` is the display name of your pipeline. In the output, the pipeline object contains an `artifactStore` object, which contains a `location` value with the name of the bucket.
15. Open the folder that matches the name of your pipeline (depending on the length of the pipeline's name, the folder name might be truncated), and then open the folder that matches the value for **Output artifact** that you noted earlier.



16. Extract the contents of the file. If there are multiple files in that folder, extract the contents of the file with the latest **Last Modified** timestamp. (You might need to give the file the `.zip` extension so that you can work with it in your system's ZIP utility.) The build output artifact is in the extracted contents of the file.

Conclusion:

We Build Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline.