



Creating our first Code Pipeline

AWS CodePipeline is a fully managed continuous integration and continuous delivery (CI/CD) service provided by Amazon Web Services (AWS). It automates the steps required to release software changes, such as building, testing, and deploying, enabling rapid and reliable application updates.

Key Features of AWS CodePipeline:

1. **Pipeline Automation:** Automates the entire software release process.
 2. **Integration:** Integrates seamlessly with AWS services (like CodeCommit, CodeBuild, and Elastic Beanstalk) and third-party tools (like GitHub, Jenkins, and Bitbucket).
 3. **Continuous Delivery:** Supports workflows for continuous integration and continuous deployment.
 4. **Custom Actions:** Allows defining custom actions using Lambda or other tools for unique requirements.
 5. **Parallel and Sequential Steps:** Supports defining pipelines with multiple stages and steps, executed sequentially or in parallel.
 6. **Pay-as-you-go Pricing:** Charges based on pipeline usage, with no upfront costs.
-

How AWS CodePipeline Works:

1. **Source:** CodePipeline listens for changes in the source repository (e.g., CodeCommit, GitHub, or S3).
 2. **Build:** Automatically triggers a build process (e.g., via AWS CodeBuild or another build tool).
 3. **Test:** Runs automated tests to validate changes.
 4. **Deploy:** Deploys the application to specified environments (e.g., Elastic Beanstalk, EC2, or Lambda).
-

Example Workflow:

1. **Source Stage:**
 - A developer pushes code changes to a GitHub repository.
 - CodePipeline detects the change and triggers the pipeline.
2. **Build Stage:**

- CodePipeline invokes AWS CodeBuild to compile the source code, run tests, and produce an artifact.

3. Test Stage:

- Artifacts are deployed to a test environment where integration tests are run.

4. Deploy Stage:

- Validated artifacts are deployed to production (e.g., an EC2 instance, an Elastic Beanstalk environment, or a Lambda function).

Benefits of AWS CodePipeline:

1. **Automation:** Reduces manual intervention in the release process.
 2. **Speed:** Enables fast, frequent, and reliable software updates.
 3. **Scalability:** Handles multiple pipelines and scales automatically based on the workload.
 4. **Flexibility:** Supports custom actions, integration with third-party tools, and multi-cloud deployments.
 5. **Security:** Built-in integration with AWS IAM for secure access control.
-

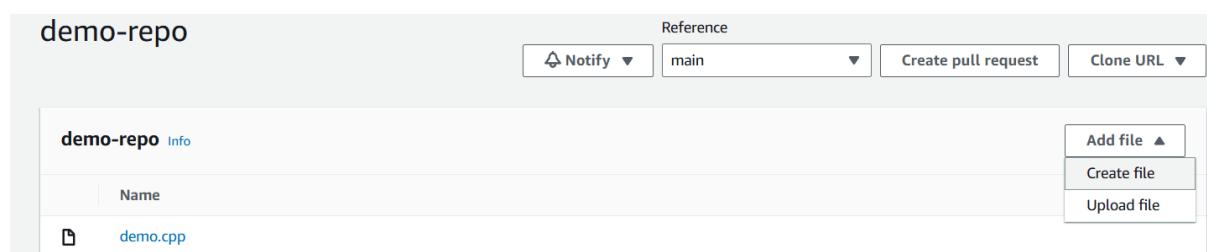
Common Use Cases:

- **Continuous Delivery:** Streamline deployments to production environments.
- **Multi-Environment Deployments:** Deploy applications to different environments (e.g., development, staging, production).
- **Testing Pipelines:** Automate testing for every code change.
- **Microservices:** Manage independent pipelines for different microservices.

AWS CodePipeline is an essential tool for implementing DevOps practices and ensuring efficient, automated, and reliable software delivery pipelines.

😊 To begin with the Lab:

1. The first step for us is to create an appspec.yml file in the code commit repository because the pipeline expects it to be there. So, basically this appspec.yml file should be added to the root of our repository.
2. So, go to your repository and click on Create file.



3. Here you need to use the below code and give the details then click on commit changes.

```
version: 0.0
os: linux
files:
- source: my-app
  destination: /tmp
```

demo-repo [Info](#)

The code editor uses the Tab key to control indentation. To navigate away from the code editor, use Escape plus Tab keys.

```
1 version: 0.0
2 os: linux
3 files:
4   - source: my-app
5     destination: /tmp|
```

Commit changes to main

File name
For example, file.txt

appspec.yml

demo-repo/appspec.yml

Author name

cloudfreeks

Email address

cf@gmail

4. After that from code build open your project and click on Edit your project. Then scroll down to build spec file.
5. Here on line 11, you need to add the wildcard characters so that it picks up both of your files from the repository.

Buildspec

Build specifications

Insert build commands

Store build commands as build project configuration

Use a buildspec file

Store build commands in a YAML-formatted buildspec file

Build commands [Info](#)

```
1 version: 0.2
2
3 phases:
4   build:
5     commands:
6       - echo "Building the application"
7       - g++ demo.cpp -o my-app # Compile the demo.cpp file into an executable named my-app
8
9 artifacts:
10  files:
11    - '**/*' # Include the compiled executable as an artifact
12
```

6. Now we are ready to create our code pipeline so click on Create Pipeline.
7. Also, before that we can just clear out the tmp directory from our EC2 instance as well.

```
[ec2-user@ip-172-31-5-236 tmp]$ ls
[ec2-user@ip-172-31-5-236 tmp]$ rm
```

Developer Tools > CodePipeline > Pipelines

Pipelines Info				
C View history Release change Delete pipeline Create pipeline				
<input type="text"/> Search				
Name	Latest execution status	Latest source revisions	Latest execution started	Most recent executions
No results There are no results to display.				

8. On step 1 choose build custom pipeline and click on Next.

Choose creation option [Info](#)

Step 1 of 6

Creation options

Choose one of the following options to create your pipeline.

Create pipeline from template

Create a pipeline from a pre-built template for common scenarios.

Build custom pipeline

Build a pipeline from scratch to meet your specific needs.

Cancel

Next

9. On step 2 just give it a name and move to next step.

Choose pipeline settings Info

Step 2 of 6

Pipeline settings

Pipeline name

Enter the pipeline name. You cannot edit the pipeline name after it is created.

No more than 100 characters

10. Here you need to choose code commit as your source provider and choose your repository.

Add source stage Info

Step 3 of 6

Source

Source provider

This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.



Repository name

Choose a repository that you have already created where you have pushed your source code.



Branch name

Choose a branch of the repository



11. On step 4 choose other build providers and choose AWS Code build then choose your project. Move to step 5.

Add build stage Info

Step 4 of 6

Build - optional

Build provider

Choose the tool you want to use to run build commands and specify artifacts for your build action.

Commands

Other build providers

AWS CodeBuild

Project name

Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

demo-build-project



or

[Create project](#)

Environment variables - optional

Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. [Learn more](#)

12. On the last step we have to choose code deploy as our deploy provider and choose the build artifact as input artifact then choose our application name and deployment group.

Deploy - optional

Deploy provider

Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

AWS CodeDeploy

Region

Asia Pacific (Mumbai)

Input artifacts

Choose an input artifact for this action. [Learn more](#)

BuildArtifact

Defined by: Build

No more than 100 characters

Application name

Choose an application that you have already created in the AWS CodeDeploy console. Or create an application in the AWS CodeDeploy console and then return to this task.

demo-code-deploy



Deployment group

Choose a deployment group that you have already created in the AWS CodeDeploy console. Or create a deployment group in the AWS CodeDeploy console and then return to this task.

demo-deployment-group



Configure automatic rollback on stage failure

Enable automatic retry on stage failure

13. Move to the review page and create your pipeline. Below you can see that our first step has been completed.

The screenshot shows the 'Source' stage of a pipeline named 'demo-first-pipeline'. The stage is listed as 'Succeeded'. The execution mode is 'QUEUED'. Pipeline execution ID is 'a871c9eb-be61-47b9-aae5-d9855de35137'. The stage details show 'Source' (AWS CodeCommit) with a status of 'Succeeded - Just now' (beac7149). A 'View details' button is present. A note at the bottom states 'beac7149 Source: Added appspec.yml'.

14. Our second step has also been completed for the build stage.

The screenshot shows the 'Build' stage of the same pipeline. The stage is listed as 'Succeeded'. The execution mode is 'QUEUED'. Pipeline execution ID is 'a871c9eb-be61-47b9-aae5-d9855de35137'. The stage details show 'Build' (AWS CodeBuild) with a status of 'Succeeded - Just now' (beac7149). A 'View details' button is present. A note at the bottom states 'beac7149 Source: Added appspec.yml'.

15. The last step for code deploy has also been completed.

The screenshot shows the 'Deploy' stage of the pipeline. The stage is listed as 'Succeeded'. The execution mode is 'QUEUED'. Pipeline execution ID is 'a871c9eb-be61-47b9-aae5-d9855de35137'. The stage details show 'Deploy' (AWS CodeDeploy) with a status of 'Succeeded - Just now' (beac7149). A 'View details' button is present. A note at the bottom states 'beac7149 Source: Added appspec.yml'.

16. Also, if you go to your instance and do the object listing in your tmp directory then you will see the my-app binary there.

```
[ec2-user@ip-172-31-5-236 tmp]$ ls  
my-app  
[ec2-user@ip-172-31-5-236 tmp]$ █
```

17. In the end if you go to the code commit and open your repository, the demo.cpp file and edit file then save it.

Edit a file

demo-repo / demo.cpp [Info](#)

The code editor uses the Tab key to control indentation. To navigate away from the code editor, use Escape plus Tab keys.

```
1 #include <iostream>
2
3 int main()
4 {
5     std::cout << "Hello World\n";
6 }
```

18. So, the code pipeline will detect the change and it will start to run again. Below you can see that once it has detected the changes the execution is in progress.

Developer Tools > CodePipeline > Pipelines

Pipelines [Info](#)

Name	Latest execution status	Latest source revisions	Latest execution started	Most recent executions
demo-first-pipeline (Type: V2 Execution mode: QUEUED)	In progress	-	Just now	View details