**Setting up a CI/CD pipeline for AWS withCodePipeline**

*In this project, I deployed a NodeJS application from Github into AWS Elastic Beanstalk and build a CI/CD pipeline.*

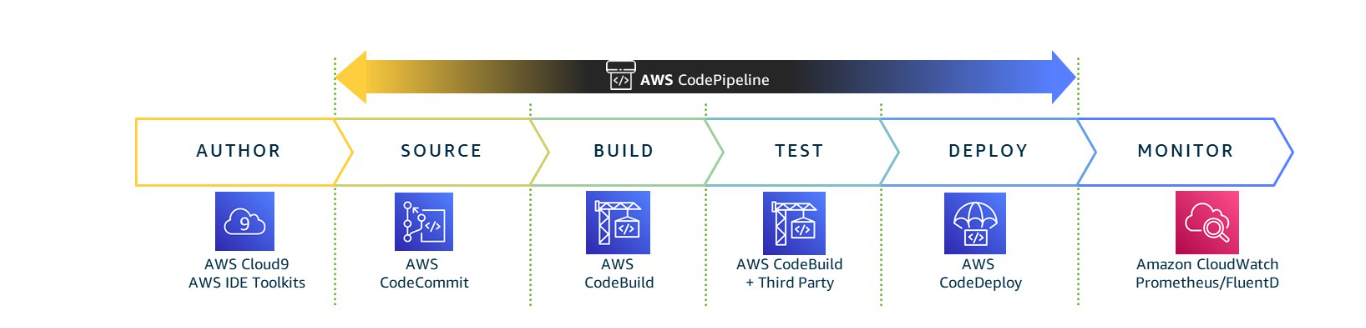
**Continuous Intergation/Continous Delivery(Deployment) (CI/CD)**

CI/CD as a whole can be pictured as a pipeline, where new code is submitted on one end, tested over a series of stages (source, build, test, staging, and production), and then published as production-ready code.Lets break this down further.

**Continuous Integration** - This is the first step in CI/CD. Developers first push the code to a code repository (**Github**,CodeCommit, Bitbucket etc). As soon as the code is pushed, there will be a test/build stage that checks the code to give developers feedback on the tests and checks that have passed/failed. In this project, I am using **CodeBuild**, but you could also use Jenkins. This enables us to find bugs early.

**Continuous delivery/deployment -** Allows you to deploy revisions to a production environment automatically without explicit approval from a developer, making the entire software release process automated.With this, we can shift from say one release every three months to 5 or more releases a day.

**Setting up the environment**

****

**Stage 1:** The image above represents the CI/CD steps for AWS . For source code you could use AWS Code Commit, Github or a third party repository. In this case i tried with both **CodeCommit and Github. CodeCommit**  is a fully-managed source control service that makes it easy for companies to host secure and highly scalable private Git repositories.It eliminates the need to operate (setup/manage)your own source control system or worry about scaling its infrastructure.It is build on a durable and highly redundant architecture, and automatically scales so you never run out of space. It also encrypts files in transit and at rest and can store code, binary and images.

**Stage 2:** For building and testing I used **AWS CodeBuild,** but you could also use Jenkins CI or any other third party CI server.

**Stage 3:** To deploy and provision the infrastructure, you can use AWS Elastic Beanstalk or CloudFormation. To save time, we will use **Elastic Bean Stalk.** Elastic Beanstalk lets you easily host web applications without needing to launch, configure, or operate virtual servers on your own. It automatically provisions and operates the infrastructure (e.g. virtual servers, load balancers, etc.) and provides the application stack (e.g. OS, language and framework, web and application server, etc.) for you.

**Stage 4:** To orchestrate everything, I used AWS CodePipeline, a service that builds, tests, and deploys your code every time there is a code change.

Note: I created an IAM user (DevOps Engineer 1) and added her to a Group with policy permissions to create CI/CD pipeline. Attach the managed policy named AWSCodePipelineFullAccess to the user (or to the IAM group to which the user belongs). Using an AWS root account is not recommended.

\*\*Goal:Completed pipeline will be able to detect changes made to the source repository containing the sample app and then automatically update the live sample app.

