**Integrating a Node.js application running on AWS Beanstalk as a docker container with Gitlab CI/CD**

Many large enterprise organizations constantly look for various AWS services that will help them quickly and reliably build their web applications in a cost-effective way. One such service to build and deploy these web applications in a quicker way is AWS Elastic Beanstalk. With AWS Elastic Beanstalk, we can quickly deploy and manage these applications in AWS cloud without worrying about the infrastructure. AWS Elastic Beanstalk supports applications developed in various programming languages and can be deployable as Docker containers. At the same time, organizations want to automate their development process using continuous integration and continuous delivery and deployment methods. GitLab CI/CD is such kind of tool built into GitLab for software development through continuous methodologies.

In this blog post, I will walk you through step by step process to deploy a simple Node.js application as a docker container hosted in Gitlab’s container registry as a Docker image into AWS Elastic Beanstalk service. We will see a process to create a pipeline where pushing an image into GitLab container registry, will automatically update in the AWS Elastic Beanstalk environment. Basically, any update to the Node.js application code, or the docker file or any other configuration file committed to GitLab repository, will build a new docker image and pushed to Gitlab container registry and eventually AWS Beanstalk will deploy the new artifact onto the environment.

# Solution Overview

The steps we will follow in this blog post are:

1. Create a Virtual Private Cloud (VPC), an Amazon S3 bucket and an RDS PostgreSQL database.
2. Provision a sample Gitlab environment with container registry, runner and docker service in an EC2 instance.
3. Download a simple Node.js application and configure it.
4. Create a docker image and push it to Gitlab container registry.
5. Provision a sample AWS Elastic Beanstalk application and environment with a CloudFormation script.
6. Configure Dockerrun.aws.json and. ebextensions files to deploy it into AWS Beanstalk environment.
7. Update the sample code and verify the if it is updated the environment in AWS Elastic Beanstalk.

The following diagram explains how the services work together.

