|  |
| --- |
|  |
| Problem Statement |

# Problem Statement

In this project you will apply the skills and knowledge which were developed throughout the Cloud DevOps Nanodegree program. These include:

Working in AWS

* Using Jenkins to implement Continuous Integration and Continuous Deployment
* Building pipelines
* Working with Ansible and Terraform to deploy clusters
* Building Kubernetes clusters
* Building Docker containers in pipelines

As a capstone project, the directions are rather more open-ended than they were in the previous projects in the program. You will also be able to make some of your own choices in this capstone, for the type of deployment you implement, which services you will use, and the nature of the application you develop.

You will develop a CI/CD pipeline for micro services applications with either blue/green deployment or rolling deployment. You will also develop your Continuous Integration steps as you see fit, but must at least include typographical checking (aka “linting”). To make your project stand out, you may also choose to implement other checks such as security scanning, performance testing, integration testing, etc.!

Once you have completed your Continuous Integration you will set up Continuous Deployment, which will include:

* Pushing the built Docker container(s) to the Docker repository (create your own custom Registry within your cluster) ; and
* Deploying these Docker container(s) to a small Kubernetes cluster. For your Kubernetes cluster you can use your own KIND Kubernetes cluster. To deploy your Kubernetes cluster, use either Ansible. Preferably, run these from within Jenkins as an independent pipeline.

### Project Tasks

#### Step 1: Propose and Scope the Project

* Plan what your pipeline will look like.
* Decide which options you will include in your Continuous Integration phase.
* Use Jenkins.
* Pick a deployment type - either rolling deployment .
* For the Docker application you can either use an application which you come up with, or use an open-source application pulled from the Internet, or if you have no idea, you can use an Nginx “Hello World, my name is (student name)” application.

#### Step 2: Use Jenkins, and implement rolling deployment.

* Create your Jenkins master box with either Jenkins and install the plugins you will need.
* Set up your environment to which you will deploy code.

#### Step 3: build your own Kubernetes cluster.

* Use Ansible to build your “infrastructure”; i.e., the Kubernetes Cluster.
* It should create the EC2 instances (if you are building your own), set the correct networking settings, and deploy software to these instances.
* As a final step, the Kubernetes cluster will need to be initialized. The Kubernetes cluster initialization can either be done by hand, or with Ansible at the student’s discretion.

#### Step 4: Build your pipeline

* Construct your pipeline in your GitHub repository.
* Set up all the steps that your pipeline will include.
* Configure a deployment pipeline.
* Include your Dockerfile/source code in the Git repository.

#### Step 5: Test your pipeline

* Perform builds on your pipeline.
* Verify that your pipeline works as you designed it.
* Take a screenshot of the Jenkins pipeline showing deployment and a screenshot of your AWS EC2 page showing the newly created or modified (for rolling) instances.

## File structure

### Application:

* app application folder
* requirements.txt dependencies of app

### CI/CD:

* Jenkinsfile file with Pipeline configuration
* kubernetes/deployment.yml file used for deploy to cluster k8s
* kubernetes/aws-auth-cm.yaml used to allow auth into k8s in aws eks

### Docker:

* Dockerfile use this file to deploy an image for the app to be runned on a container

**System Requirements:**

Jenkins master - VM1

Build server - VM1

AWS Account

K8s – VM2, VM3