La prueba se realizó con las siguientes condiciones:

Sistema operativo Fedora Workstation 39, teniendo instalado Terraform, AWS CLI, Docker y Kubectl.

**Resumen de Pasos Realizados:**

A continuación describo los pasos que pude realizar, teniendo en cuenta que es la primera vez que utilizo Terraform y fué lo que mas me tomó tiempo comprender.

**Creación de Carpeta “terraform-despliegues”**

Esta carpeta contiene los archivos “.tf” que contienen la creación de los dos clúster en AWS. Acá se debe ir a una línea de comandos y definir estas variables de entorno de las llaves de acceso y la región que para el caso utilicé la región indicada en la variable:

export TF\_VAR\_access\_key="Llave de acceso obtenida en AWS"

export TF\_VAR\_secret\_key="Llave secreta obtenida en AWS"

export TF\_VAR\_region="us-east-2"

**Crear los clúster de Kubernetes usando Terraform:**

Entrar a la carpeta “/terraform-despliegues/eks-unificados” y realizar la inicialización y descarga de librerías mediante **terraform init** y posteriormente aplicar la infraestructura con **terraform apply**. Se configuraron los outputs para que se muestren al finalizar los cluster creados.

**Crear el Registry de imágenes usando Terraform:**

Decidí hacer este paso por separado de los cluster pues el Registry al final no lo estuve recreando tanto como los clúster. Para esto es de realizar **terraform init** y **terraform apply** en la ruta “/terraform-despliegues/registry”.

**Crear el proyecto java (hola mundo) utilizando VSCode:**

Inicialicé un proyecto java con maven con un único controller y un único método get de holaMundo en la ruta “/java/api\_prueba\_tecnica”.

**Crear Docker file:**

Al interior del proyecto Java cree el dockerfile para desplegar como contenedor luego.

**Crear archivos de despliegue en Kubernetes:**

Dentro del proyecto Java cree la carpeta k8s para posteriormente poder desplegarlo correctamente.

**Conectarme a AWS CLI:**

Usando **aws configure** se asignan las credenciales para el CLI de AWS.

**Conectar al registry creado en AWS:**

Para hacer login en docker y poder publicar la imagen de forma manual del API en el cluster de desarrollo, realizo el login usando **sudo aws ecr get-login-password --region us-east-2 | sudo docker login --username AWS --password-stdin cuentaws.dkr.ecr.us-east-2.amazonaws.com** de manera que pueda realizar push manual de las imagenes para probar la conexión al registry creado con terraform.

**Conectar Kubectl obteniendo los configs de AWS:**

Para obtener las configuraciones y acceso a los clusters creados con Terraform:

aws eks update-kubeconfig --name eks-despliegue

aws eks update-kubeconfig --name eks-desarrollo

De esa forma ya puedo moverme a los cluster y en cada uno crear un namespace usando la carpeta “/kubernetes-no-auto” donde esperaba instalar Jenkins para los siguientes pasos.

**Pasos no realizados:**

A continuación describo los pasos que pensaba realizar pero dado que me tomó mas tiempo entender Terraform, ya no pude realizar, de igual forma describo en orden los pasos que iba a ejecutar.

1. Desplegar en el clúster “eks-despliegue” Jenkins para poder programar ahí un pipeline.
2. Crear un repositorio git para la aplicación java y publicarla en github, creando ramas main, development y feature/carga\_inicial que sería la que iniciaría como primer pull/merge request posteriormente.
3. Inicializar Jenkins, iniciar sesión y configurar plugins para docker y kubernetes principalmente.
4. Configurar credenciales de Jenkins, de mi cuenta en github, de docker para el registry creado y para el kubect en el cluster de “eks-desarrollo”.
5. Crear un pipeline de jenkins, el cual se disparara al detectar cambios en una rama y cuyos pasos se harían de la siguiente forma:
   1. Configurar el patrón de nombres de la ramas para disparar el pipeline cuando corresponda, pensaba usar “desarrollo” para el caso.
   2. Clonar el repositorio en rama development.
   3. Aplicar validaciones de Sonarqube (esto nunca lo he hecho iba a investigar).
   4. Ejecutar mvn build.
   5. Ejecutar docker tag y docker push, usando el dockerfile que ya está en el proyecto Java.
   6. Ejecutar kubectl para los archivos en la carpeta k8s de la aplicación java, son el deployment y el service partiendo de que el paso anterior ya habría generado la imagen en el registry.