To create the **GitLab CI/CD** pipeline that builds and pushes a Docker image to **DockerHub**, and deploys it to a **Kubernetes cluster** after the code is merged:

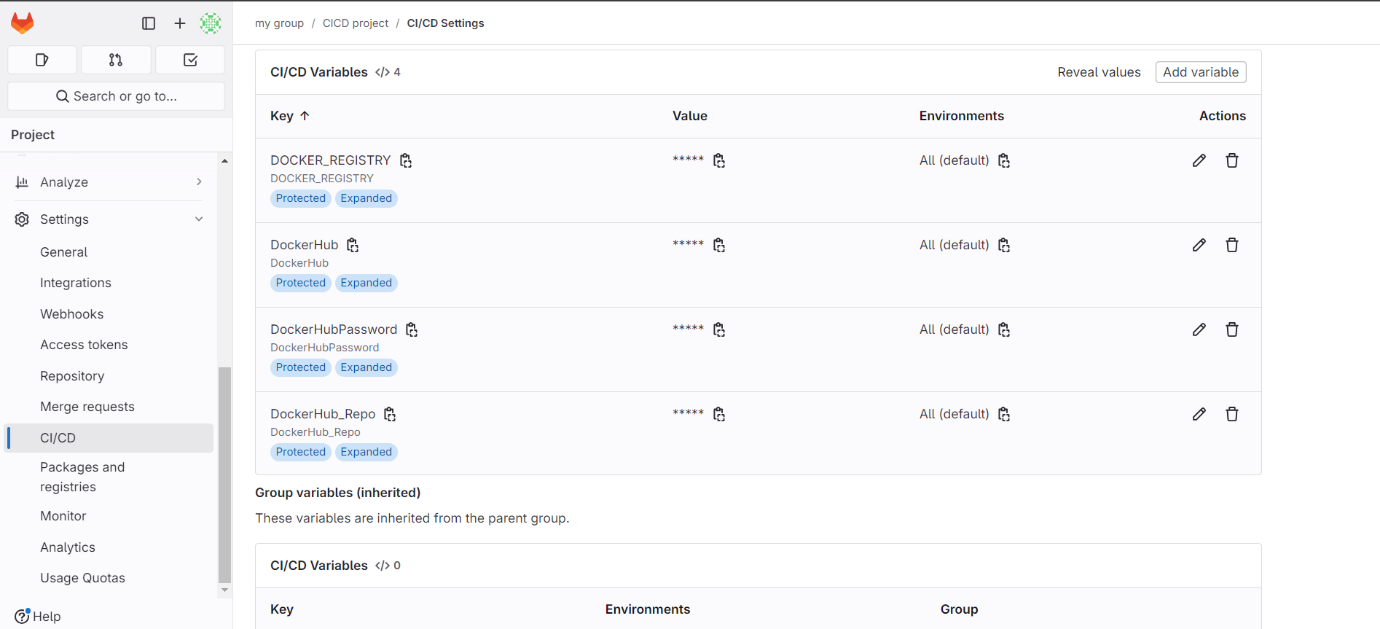
* **you can follow these steps:**

**1. Pre-requisites:**

* Docker Hub credentials (username, password/access token).
* Kubernetes cluster configured (with kubectl access).
* A deployment or service in Kubernetes for your application.
* Docker and Kubernetes credentials stored in GitLab CI/CD Variables.

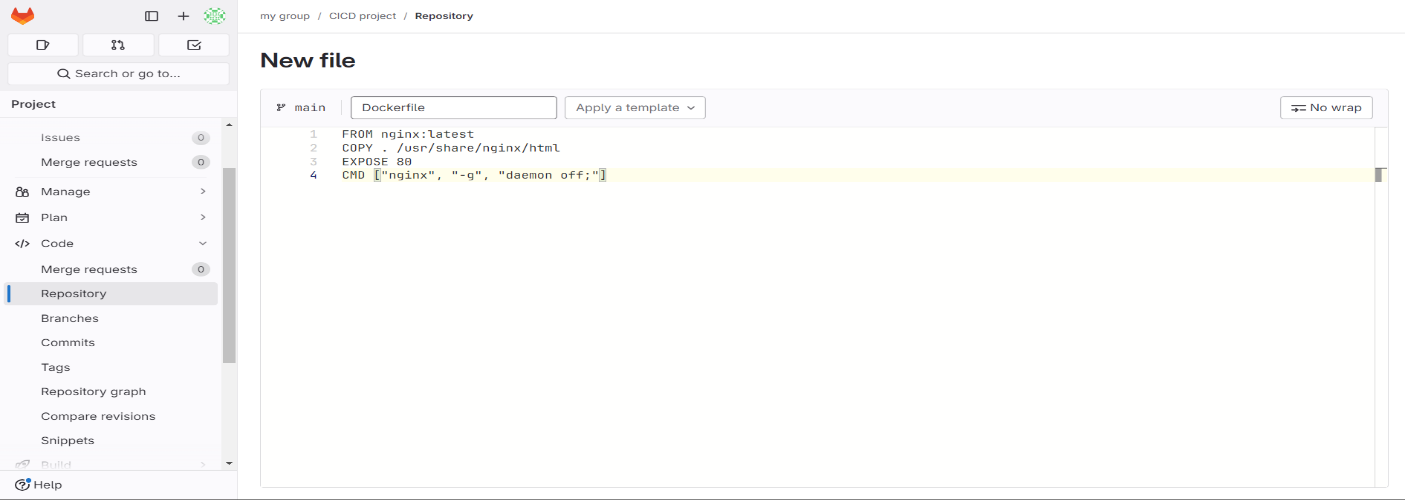
**2. Set Up GitLab CI/CD Variables:**

* In your GitLab project, go to **Settings > CI / CD > Variables** and add the following environment variables:
* DOCKER\_HUB: Your Docker Hub username.
* DOCKER\_PASSWORD: Your Docker Hub access token.
* KUBE\_CONTEXT: <PROJECT\_NAME>/<REPO.NAME>:<CLUSTER\_NAME>
* DOCKER\_REPO: The name of your Docker image (e.g., your\_username/yourapp).



**3.Configure Docker for Kubernetes**

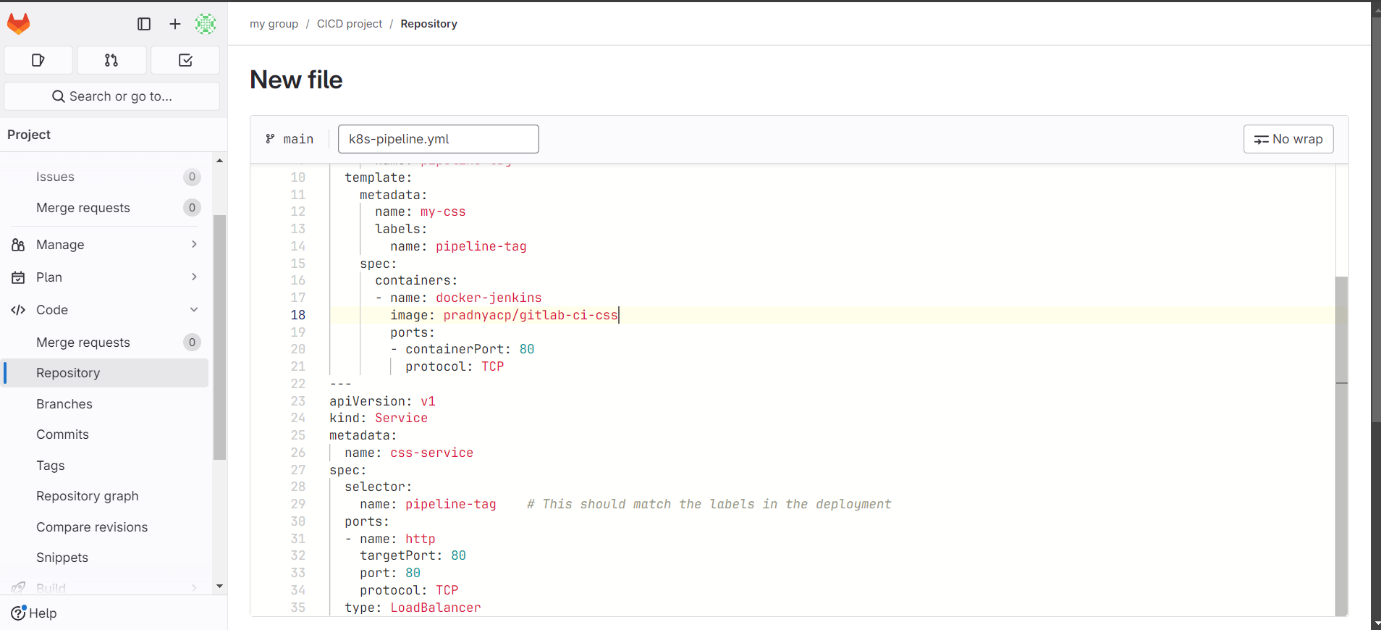
* Install Docker on your machine to build and manage Docker images for your application.
* You will need a Docker image to deploy to Kubernetes.
* Create **index.html** file, upload code in it and upload it to GitLab repository.
* **Dockerfile**: Create a Dockerfile for your project in GitLab repository.



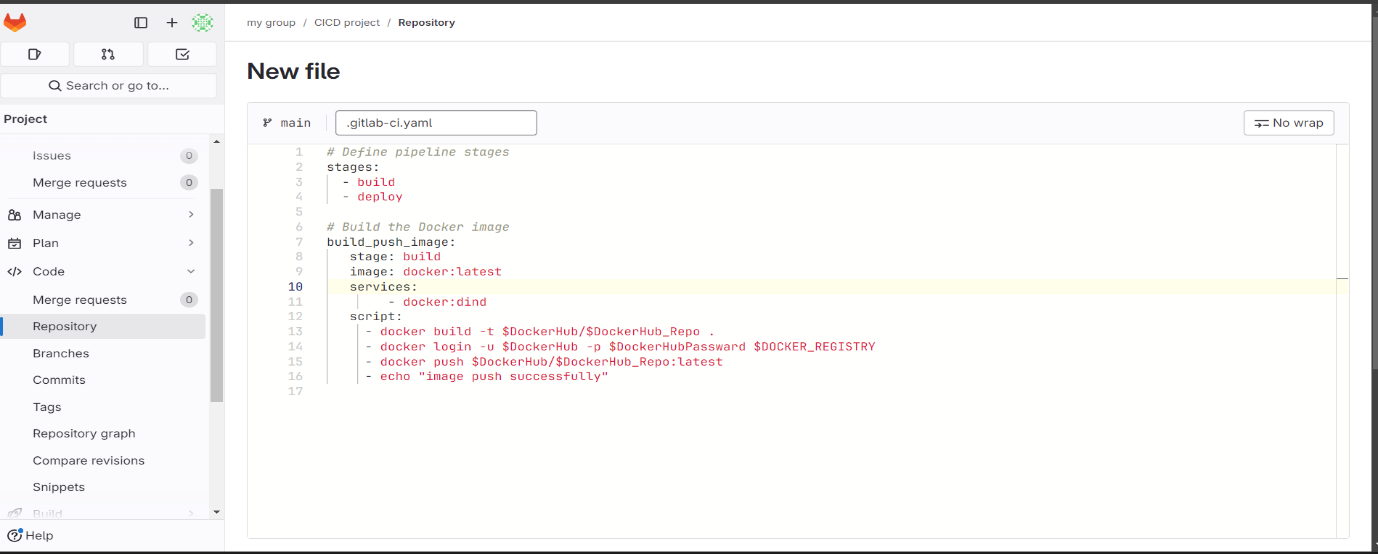
**4.Create Kubernetes Deployment Files**

In your repository, create Kubernetes YAML files for deployment and services:

**K8s-pipeline.yml**:

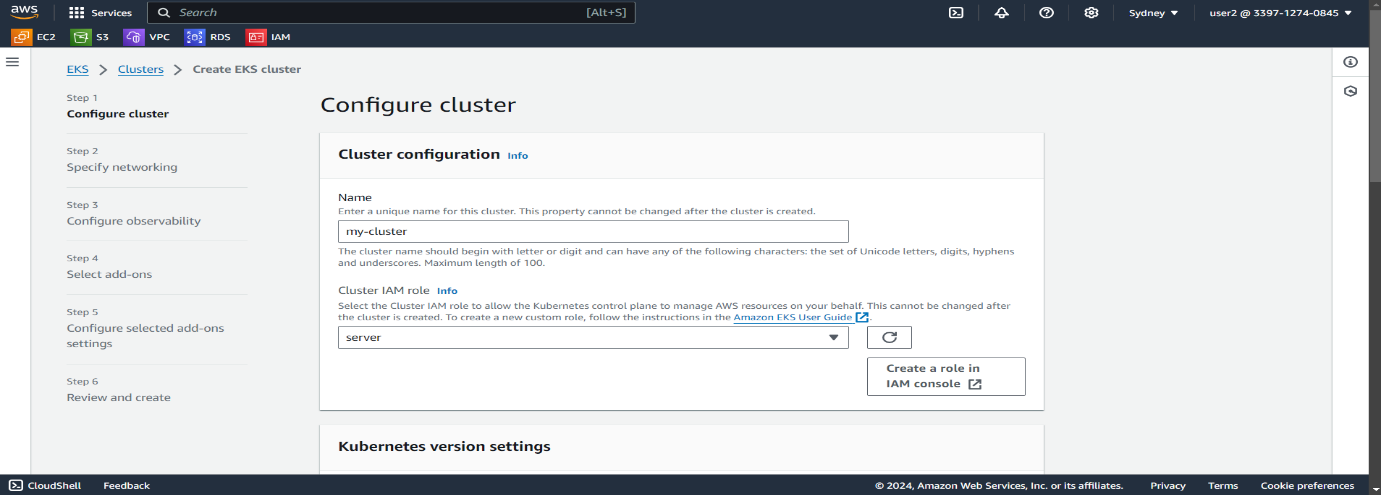


**5.Set up .gitlab-ci.yml File**

Create a **.gitlab-ci.yml** file in the root of your repository to define your CI/CD pipeline:

**6.Create the EKS Cluster Using eksctl:**

1. Create an EKS Cluster

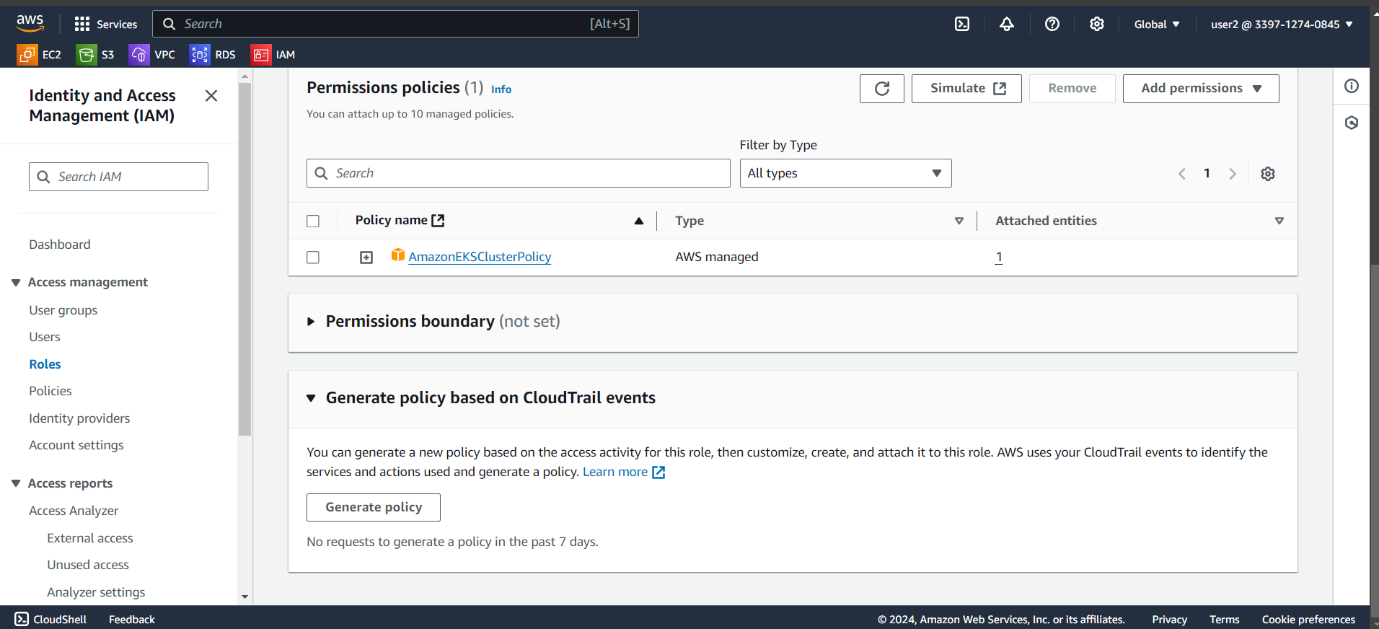


**Attach IAM Policies to the EKS Cluster Role**

When creating the EKS cluster, ensure that the IAM role assigned to the EKS control plane has sufficient permissions to manage AWS resources. Attach the following AWS managed policies to the EKS cluster's IAM role:

**Policies to Attach for EKS Cluster Role:**

* **AmazonEKSClusterPolicy**

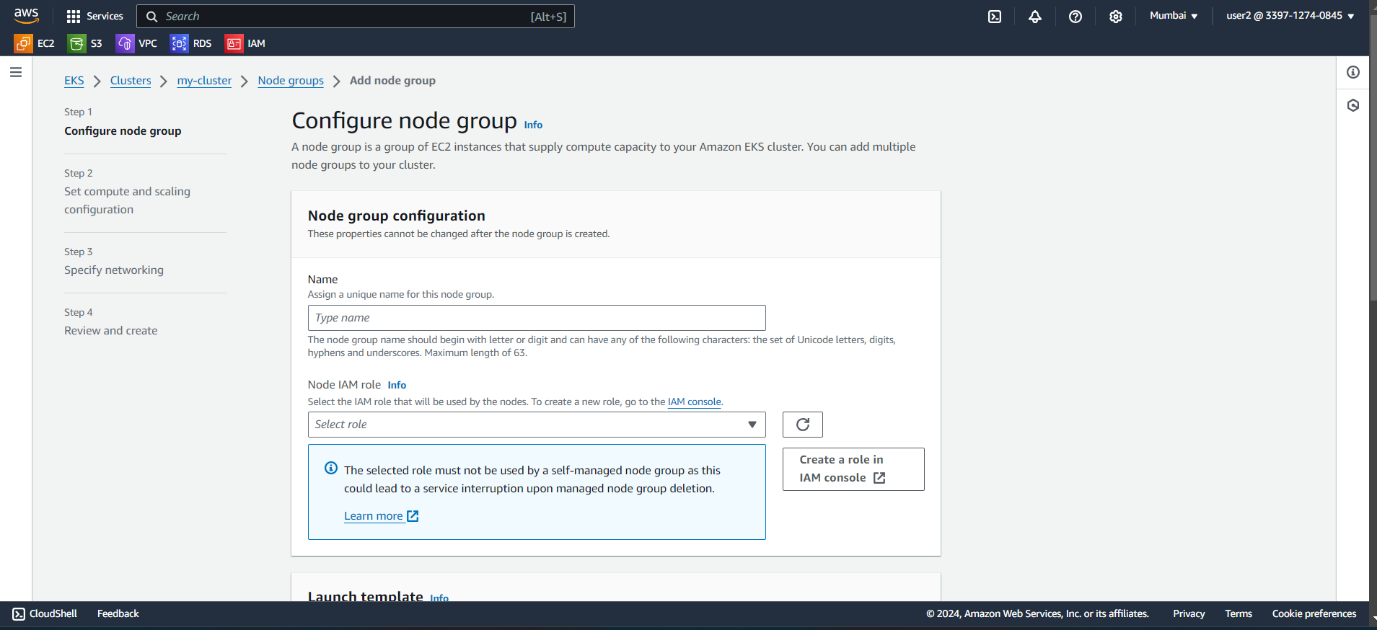


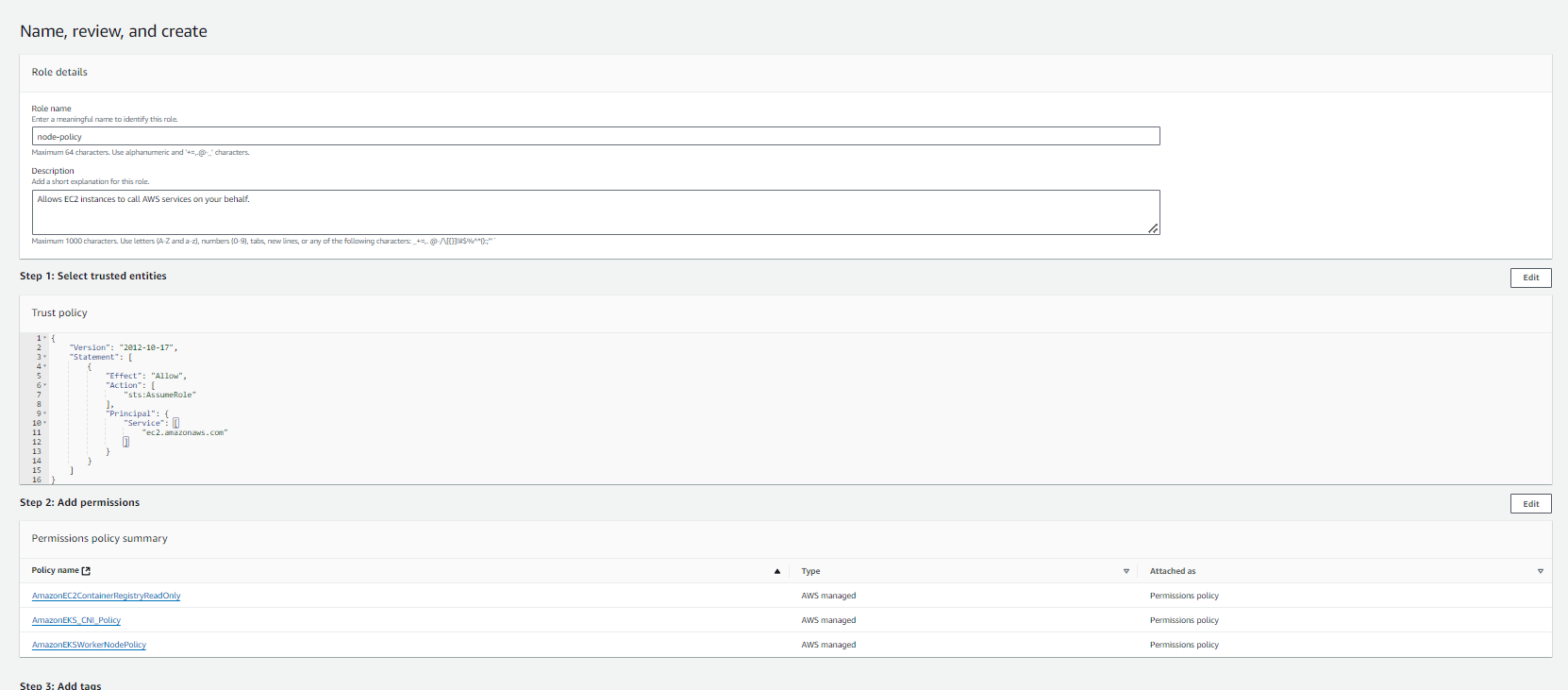
**Attach IAM Policies to the Worker Node Role**

The worker nodes in the EKS cluster require certain permissions to interact with the Kubernetes control plane and perform various functions such as networking and pulling images from the Amazon Elastic Container Registry (ECR).

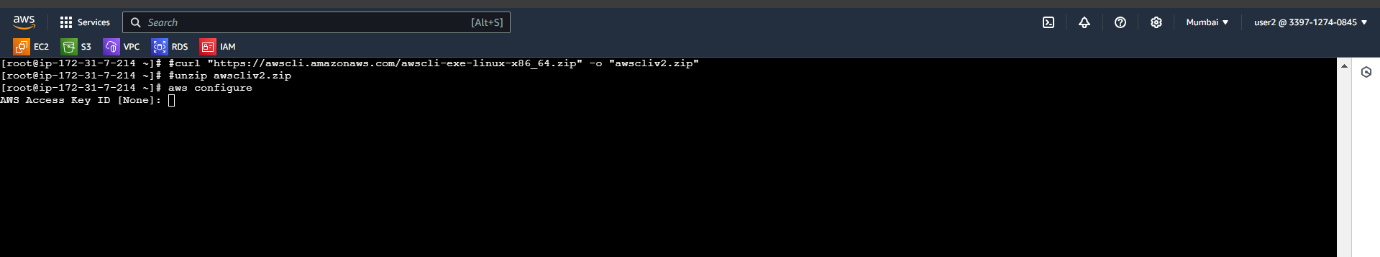
**Policies to Attach for Worker Node Role**

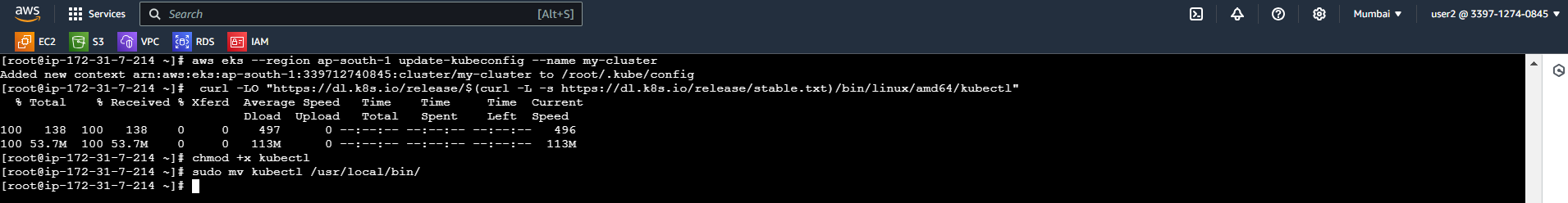
* **AmazonEKSWorkerNodePolicy**
* **AmazonEC2ContainerRegistryReadOnly**
* **AmazonEKS\_CNI\_Policy**

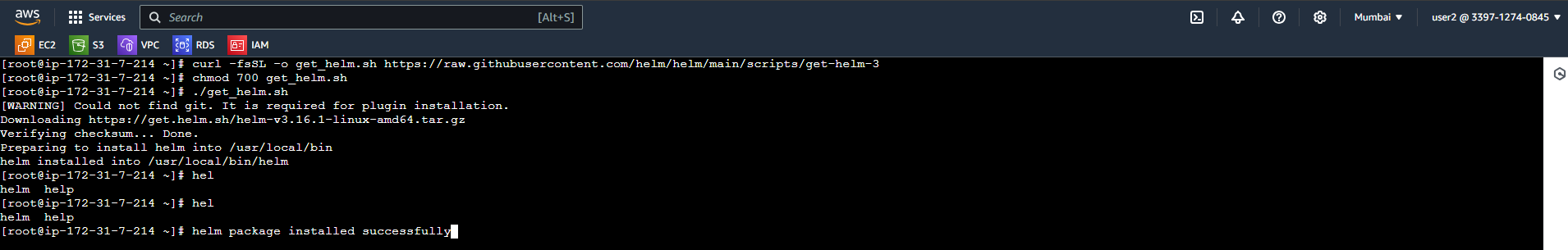




* **After the successfully created cluster and nodes you need to enter to node instance and configure services: (kubectl, helm, aws-cli):**

****

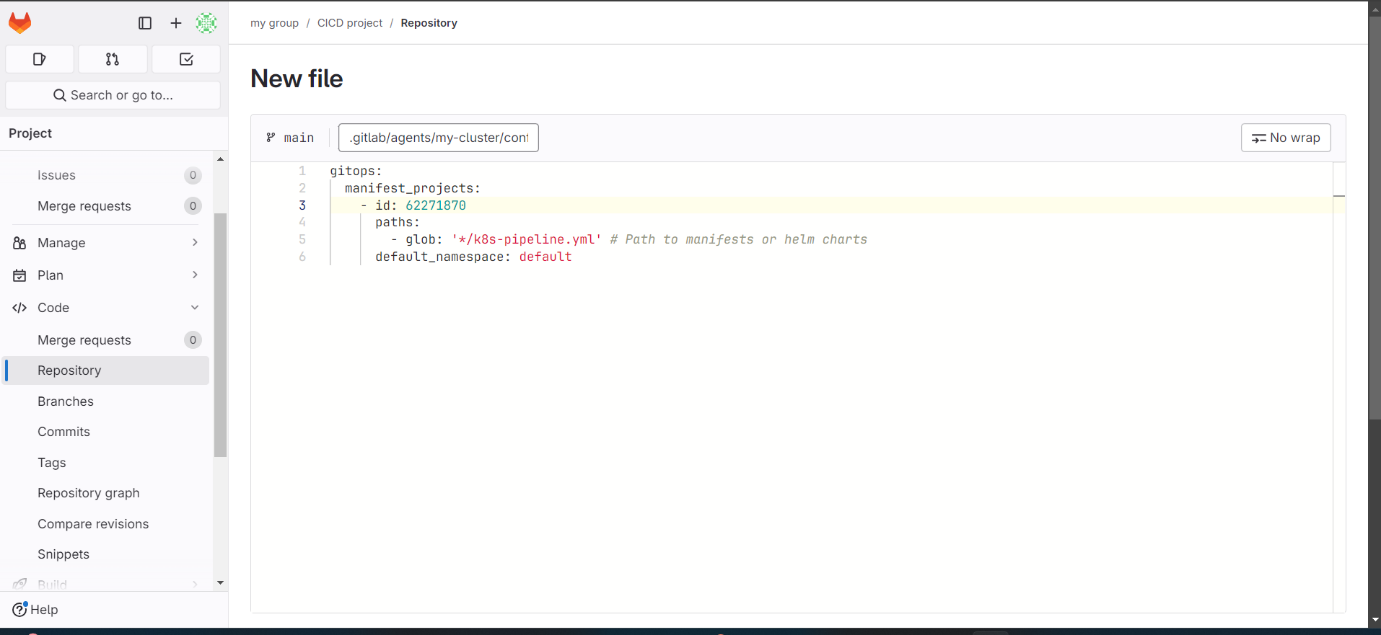
****

****

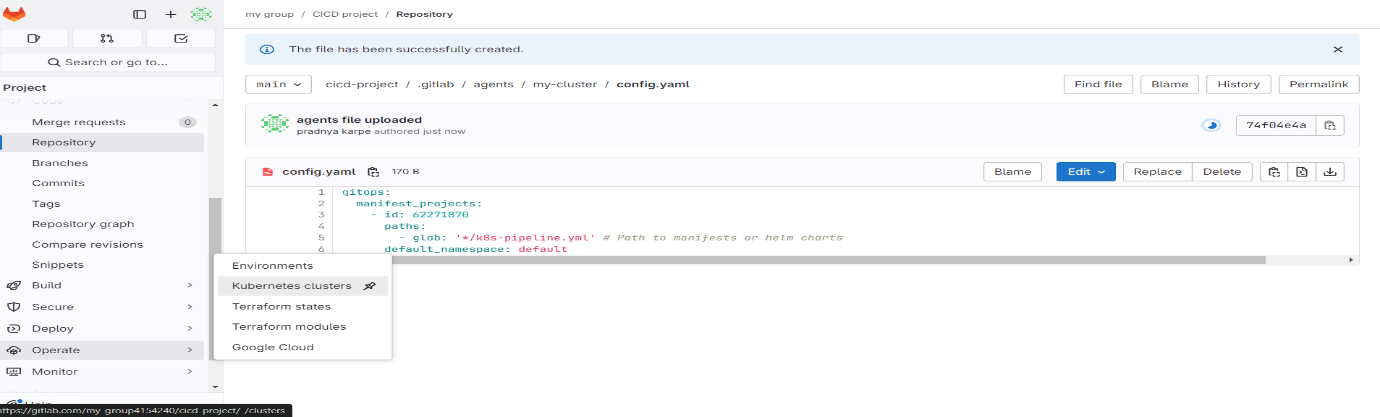
**Enable GitLab Agent on GitLab**

* **Go to your GitLab project in which you want to set up the agent**.

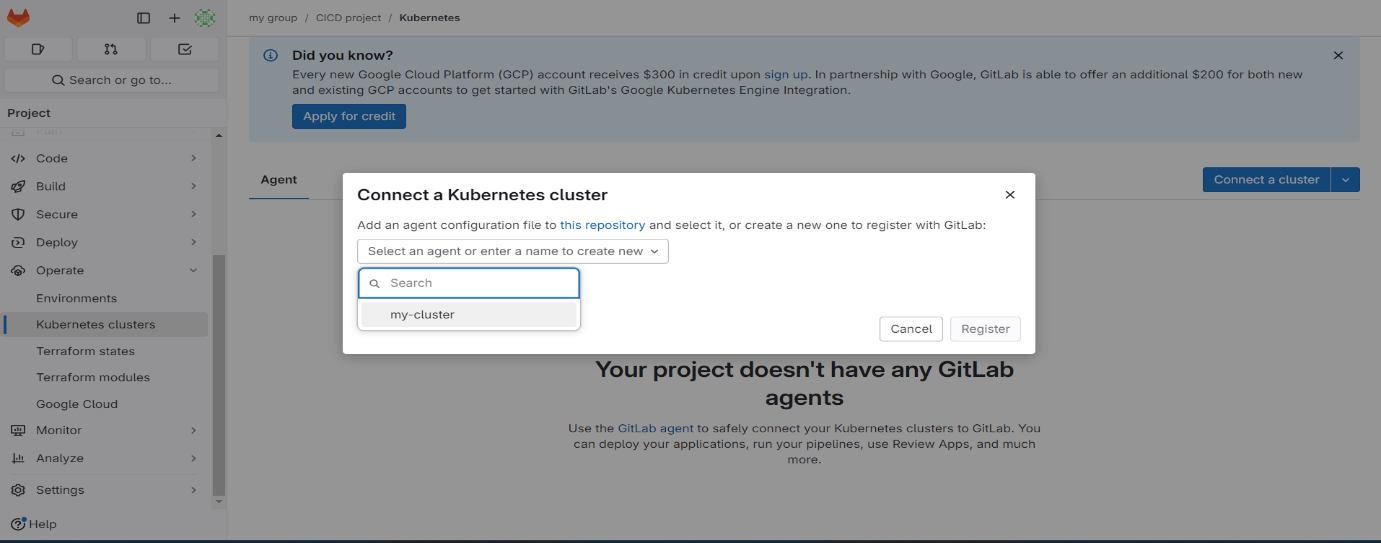
The **.gitlab/agents/<agent-name>/config.yml** file is a configuration file used by the GitLab Agent to manage and sync your Kubernetes cluster with your GitLab repository.



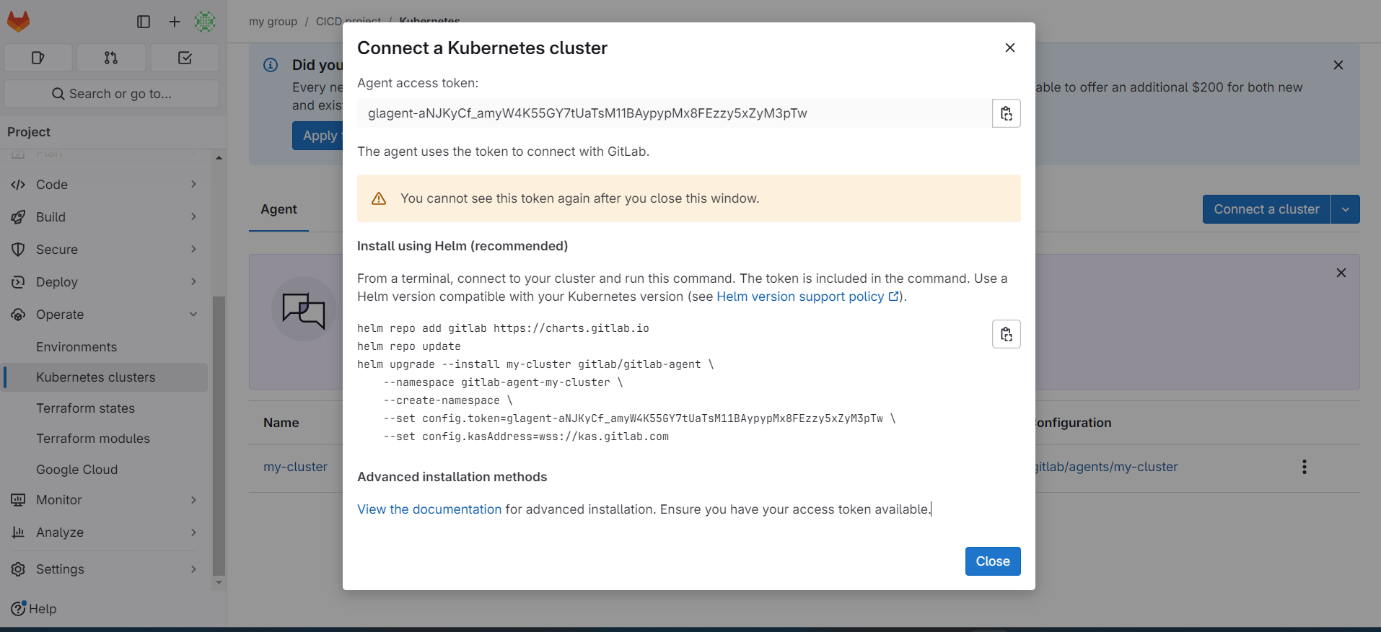
* Navigate to Operate > Kubernetes Clusters in the left sidebar.



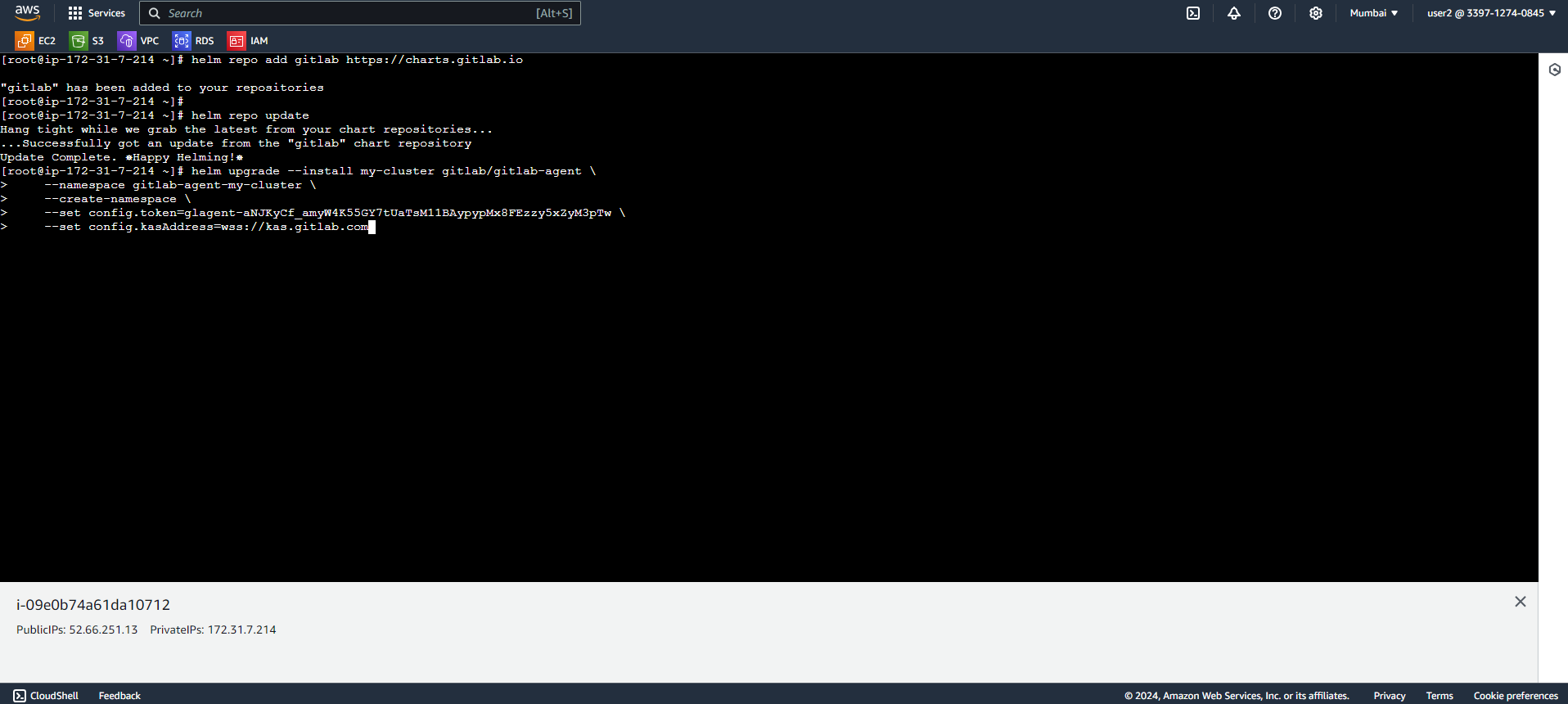
* Click the **"Connect a cluster (Agent)"** button.
* Select the **"GitLab Agent"** option.



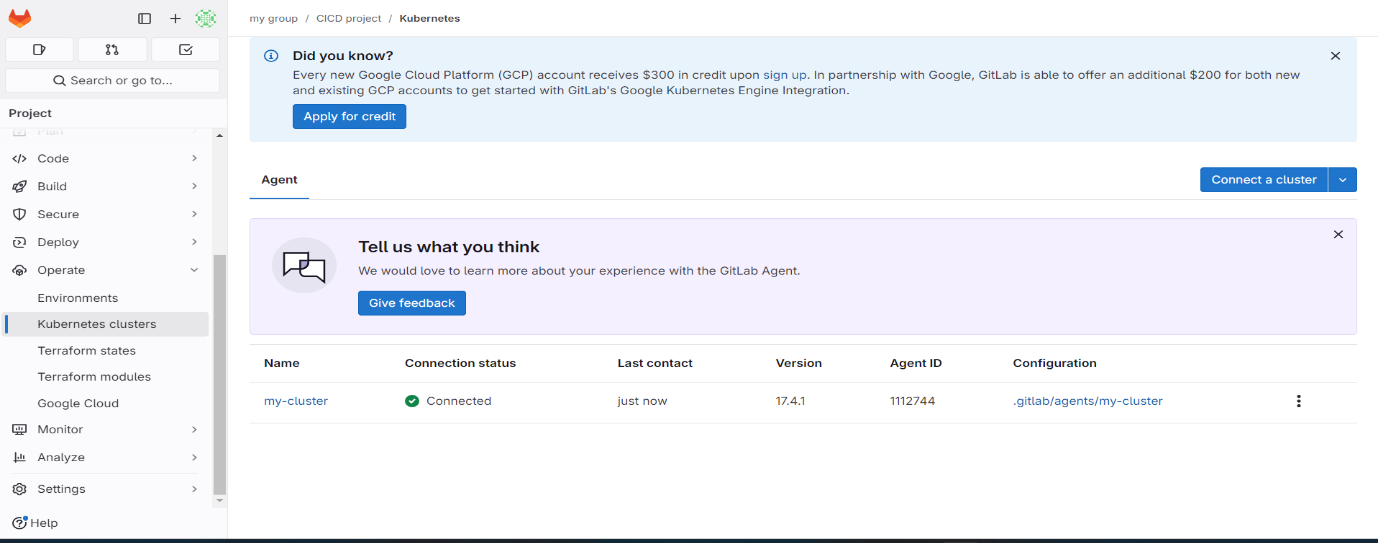
* Now, it will generate some commands to connect Kubernetes cluster through helm package manager:



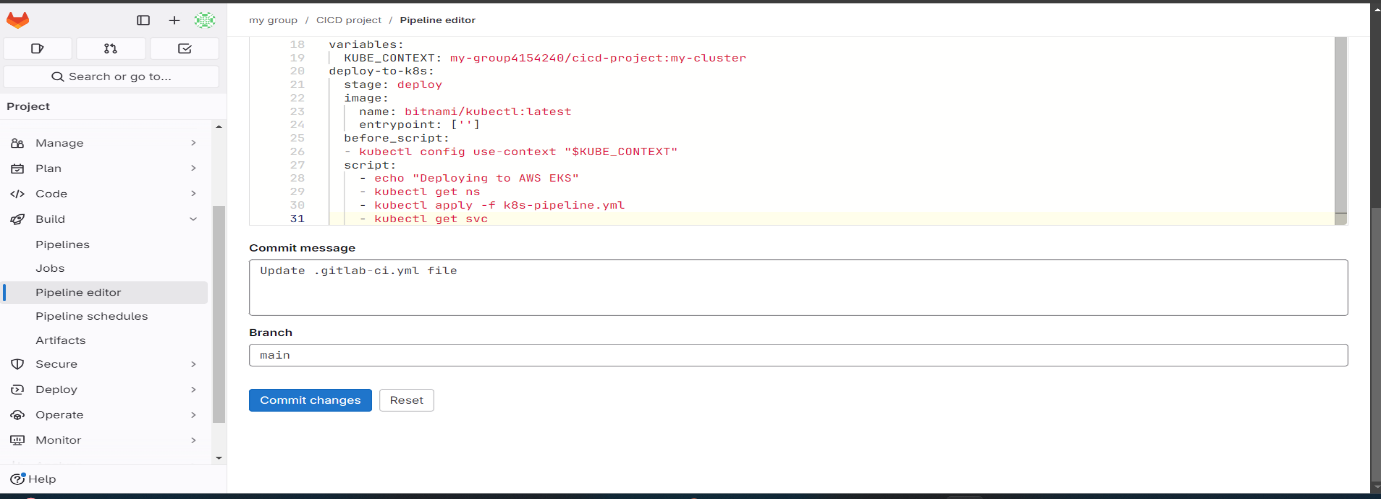
* Hit these generated commands on AWS-EKS node:



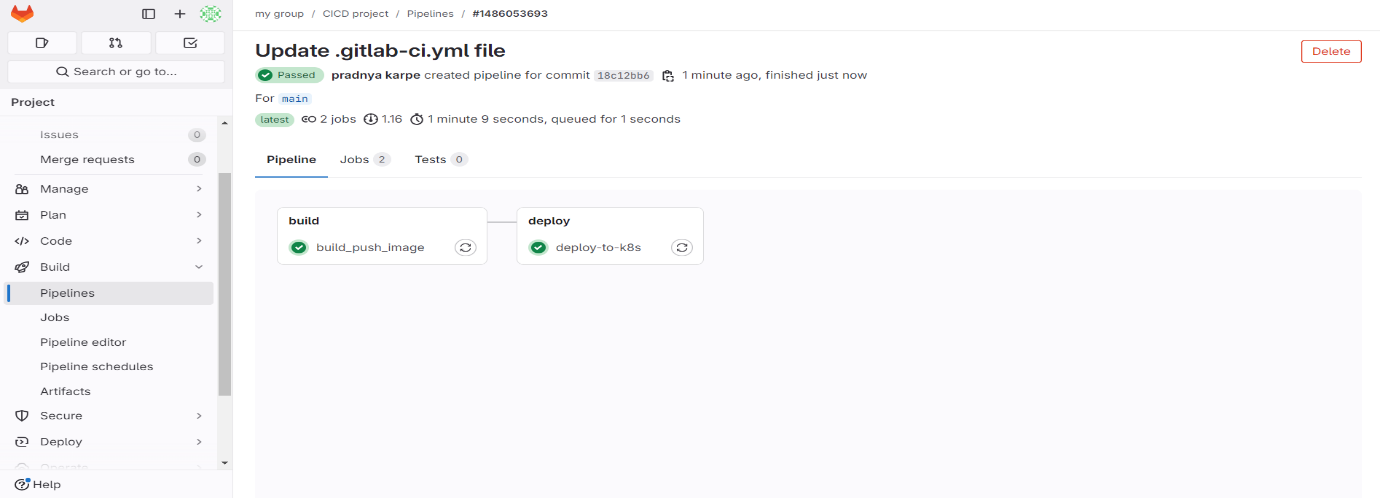
* Now, it will show the status of connection:

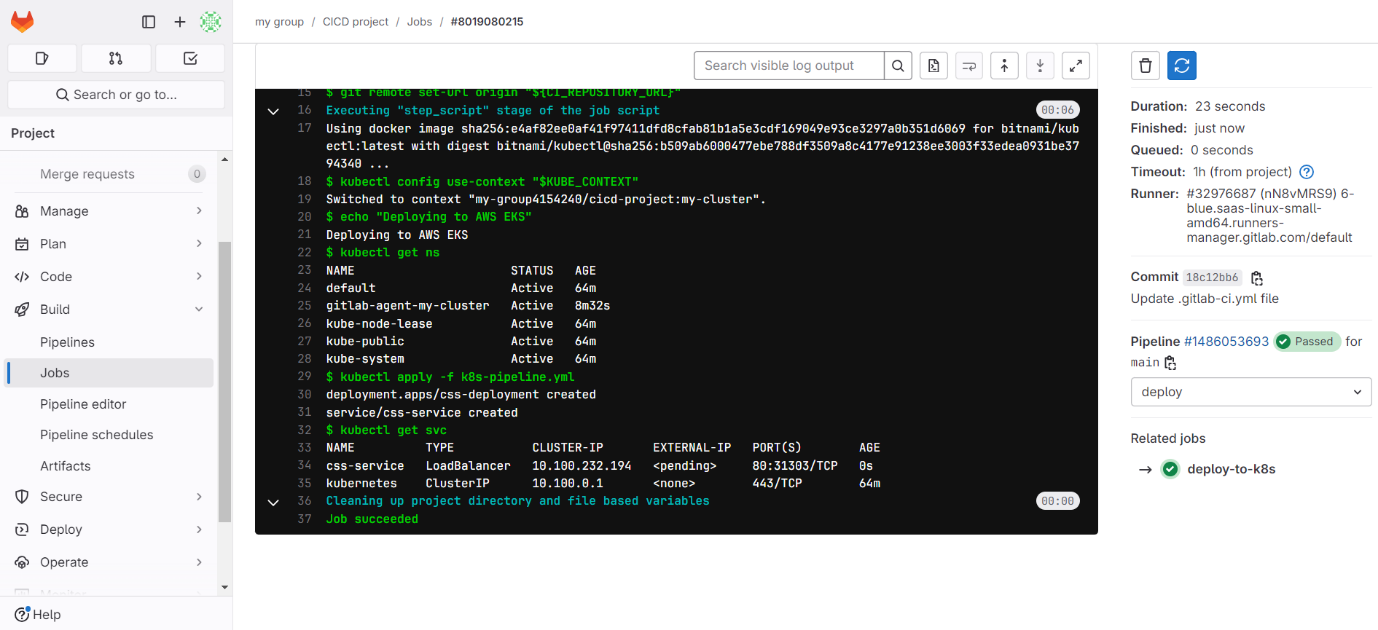


* **Add deploy stage in .gitlab-ci.yml file:**

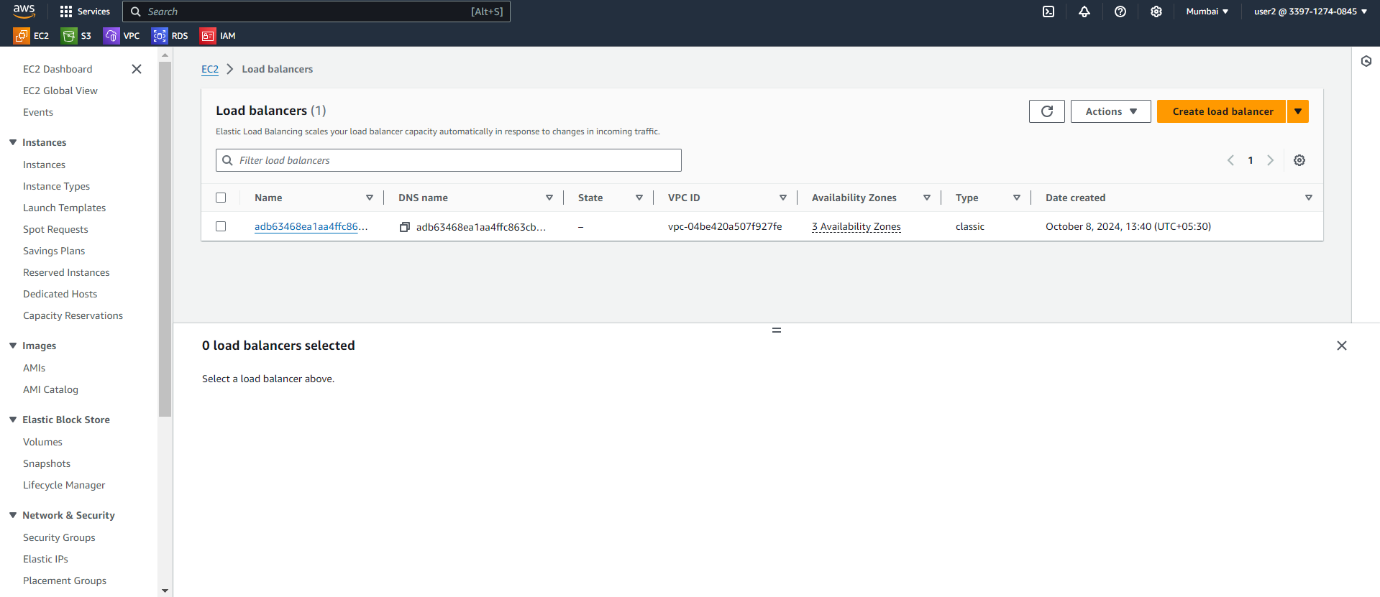
****

* **Commit the changes, automatically pipeline will be trigger and you will get status of pipeline on pipeline option:**





* **We will get load balancer as we have mentioned in our service.yml file (k8s-pipeline.yml)**



* Once, hit the DNS of load balancer on browser then we will able to access our page:

