GitLab-CI-CD pipeline created using DockerHub and Deployed on EKS-Cluster

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:

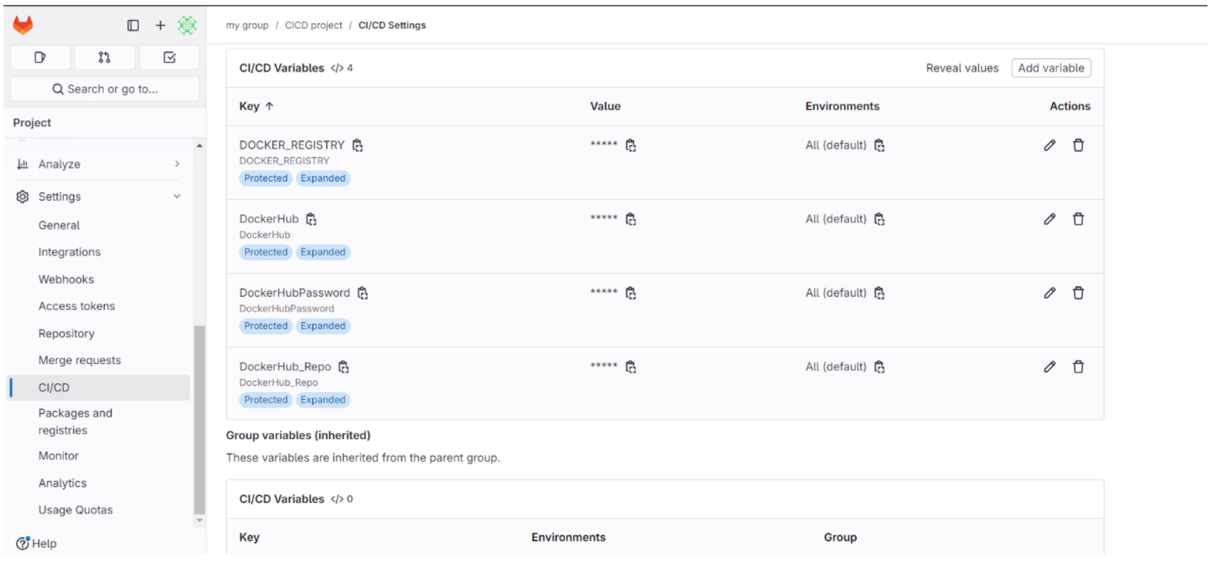
- **you can follow these steps:**

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).



CI/CD Variables 4

Key	Value	Environments	Actions
DOCKER_REGISTRY DOCKER_REGISTRY Protected Expanded	*****	All (default)	
DockerHub DockerHub Protected Expanded	*****	All (default)	
DockerHubPassword DockerHubPassword Protected Expanded	*****	All (default)	
DockerHub_Repo DockerHub_Repo Protected Expanded	*****	All (default)	

Group variables (inherited)

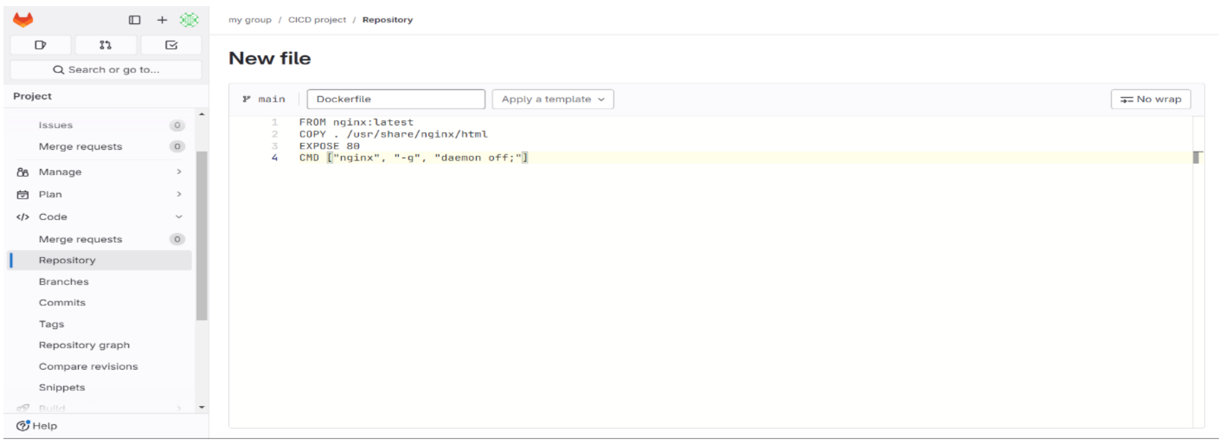
These variables are inherited from the parent group.

CI/CD Variables 0

Key	Environments	Group

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.



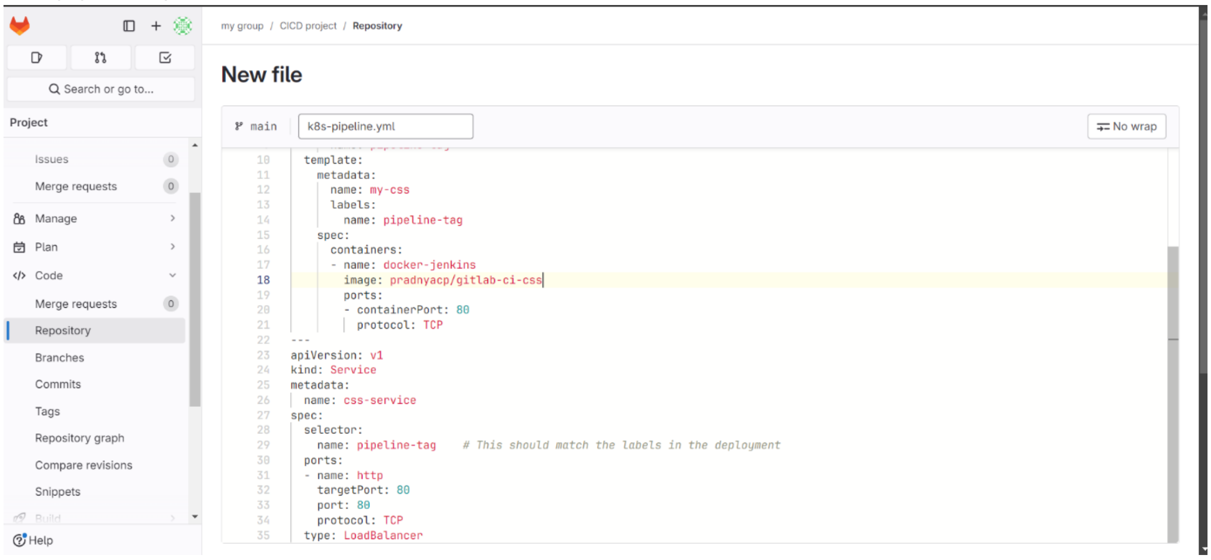
```
FROM nginx:latest
COPY . /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

GitLab-CI-CD pipeline created using DockerHub and Deployed on EKS-Cluster

4.Create Kubernetes Deployment Files

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

K8s-pipeline.yml:



my group / CICD project / Repository

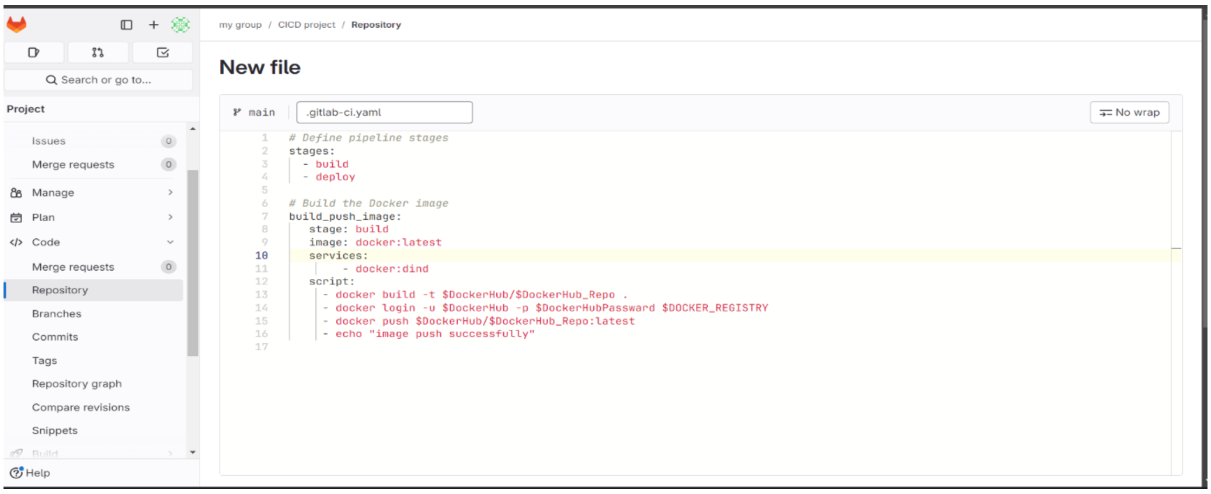
New file

main k8s-pipeline.yml

```yaml
  template:
    metadata:
      name: my-css
      labels:
        name: pipeline-tag
    spec:
      containers:
      - name: docker-jenkins
        image: pradnyacp/gitlab-ci-css
        ports:
        - containerPort: 80
          protocol: TCP
---
apiVersion: v1
kind: Service
metadata:
  name: css-service
spec:
  selector:
    name: pipeline-tag    # This should match the labels in the deployment
  ports:
  - name: http
    targetPort: 80
    port: 80
    protocol: TCP
  type: LoadBalancer
```

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:



my group / CICD project / Repository

New file

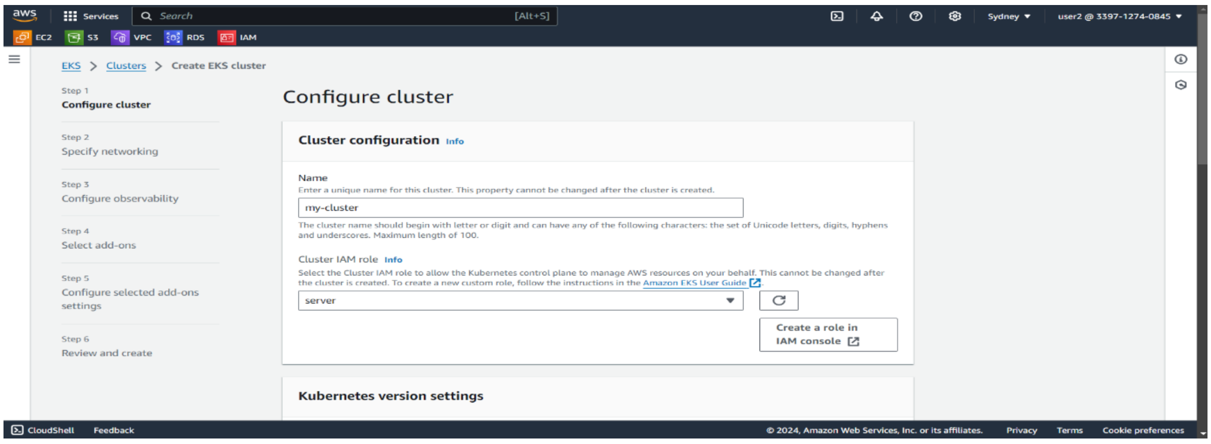
main .gitlab-ci.yaml

```yaml
# Define pipeline stages
stages:
  - build
  - deploy

# Build the Docker image
build_push_image:
   stage: build
   image: docker:latest
   services:
          - docker:dind
   script:
     - docker build -t $DockerHub/$DockerHub_Repo .
     - docker login -u $DockerHub -p $DockerHubPassward $DOCKER_REGISTRY
     - docker push $DockerHub/$DockerHub_Repo:latest
     - echo "image push successfully"
```

6.Create the EKS Cluster Using eksctl:

1. Create an EKS Cluster



EKS > Clusters > Create EKS cluster

Step 1
Configure cluster

Step 2
Specify networking

Step 3
Configure observability

Step 4
Select add-ons

Step 5
Configure selected add-ons settings

Step 6
Review and create

Configure cluster

Cluster configuration Info

Name
Enter a unique name for this cluster. This property cannot be changed after the cluster is created.
my-cluster
The cluster name should begin with letter or digit and can have any of the following characters: the set of Unicode letters, digits, hyphens and underscores. Maximum length of 100.

Cluster IAM role Info
Select the Cluster IAM role to allow the Kubernetes control plane to manage AWS resources on your behalf. This cannot be changed after the cluster is created. To create a new custom role, follow the instructions in the Amazon EKS User Guide.
server

Create a role in IAM console

Kubernetes version settings

© 2024, Amazon Web Services, Inc. or its affiliates.

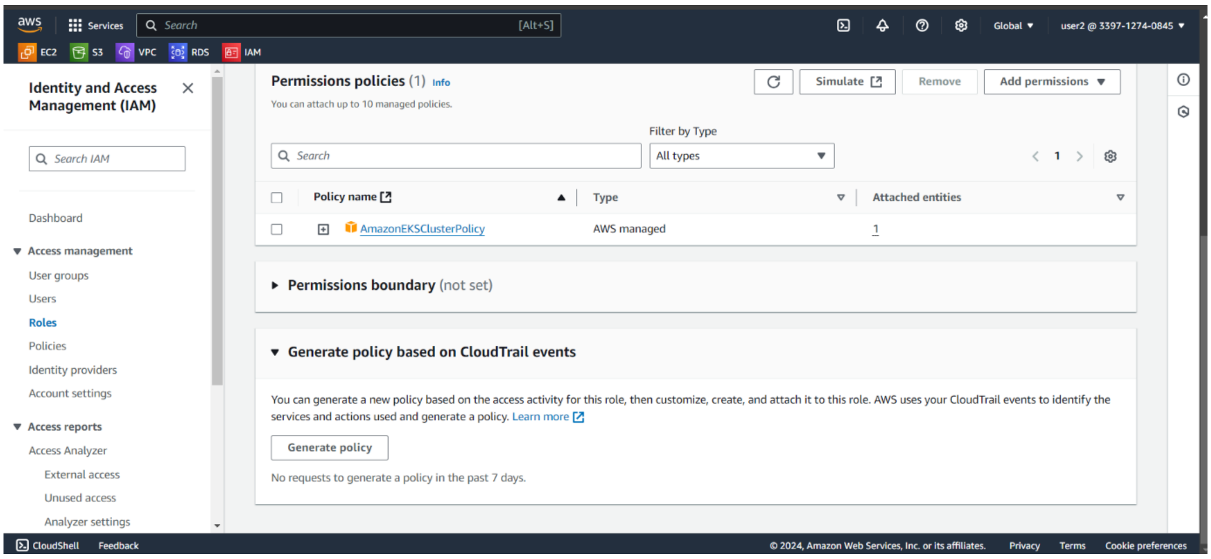
GitLab-CI-CD pipeline created using DockerHub and Deployed on EKS-Cluster

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**



Permissions policies (1) Info

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
AmazonEKSClusterPolicy	AWS managed	1

Permissions boundary (not set)

Generate policy based on CloudTrail events

You can generate a new policy based on the access activity for this role, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. Learn more

Generate policy

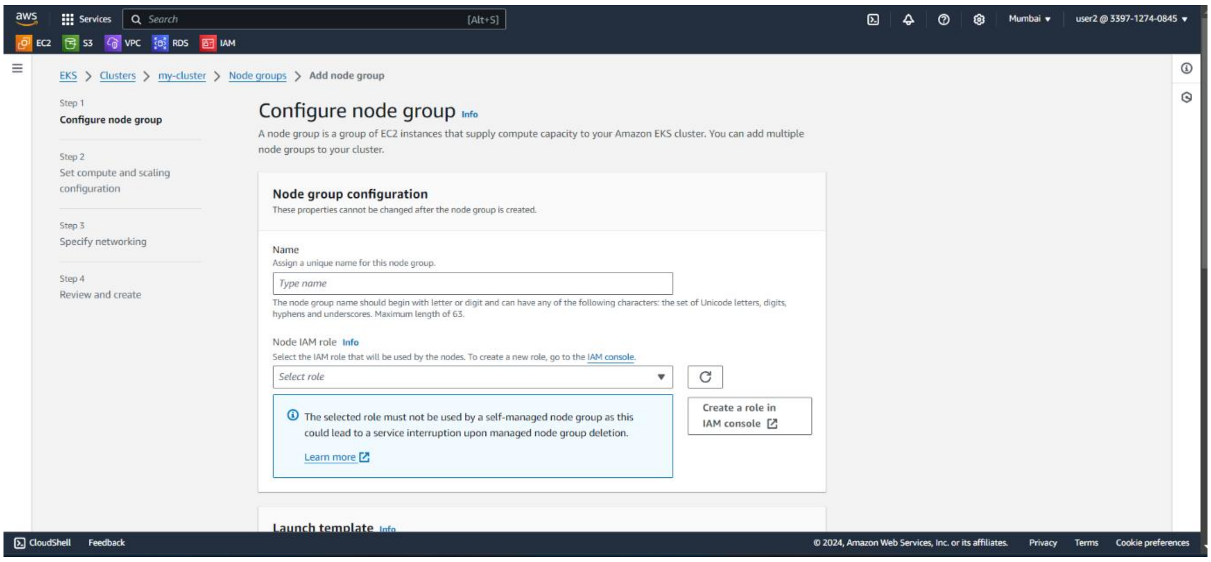
No requests to generate a policy in the past 7 days.

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**



Configure node group Info

A node group is a group of EC2 instances that supply compute capacity to your Amazon EKS cluster. You can add multiple node groups to your cluster.

Node group configuration

These properties cannot be changed after the node group is created.

Name

Assign a unique name for this node group.

Type name

The node group name should begin with letter or digit and can have any of the following characters: the set of Unicode letters, digits, hyphens and underscores. Maximum length of 63.

Node IAM role Info

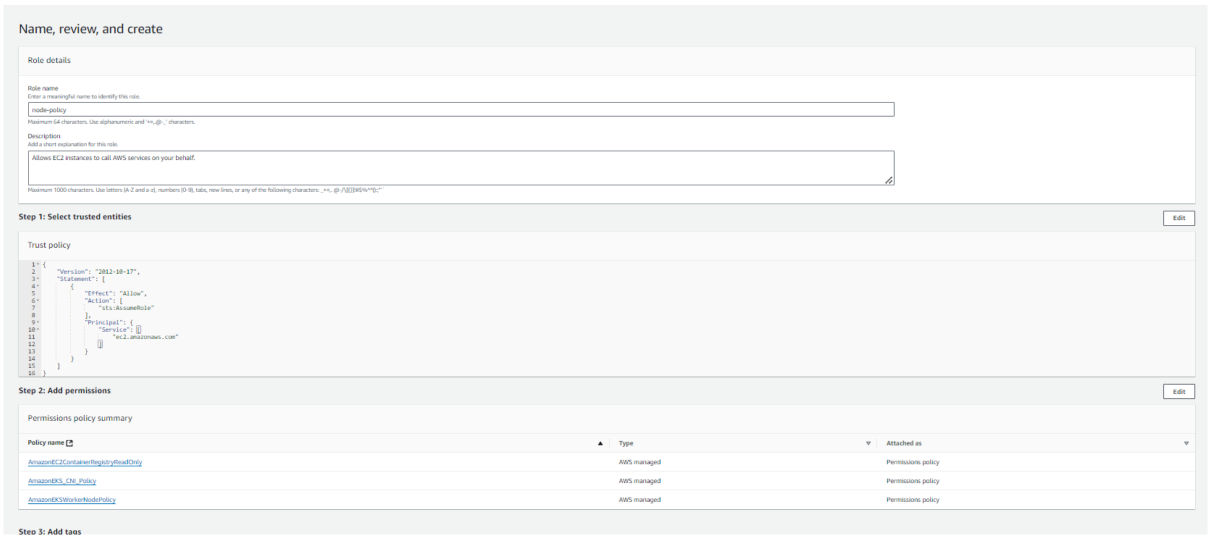
Select the IAM role that will be used by the nodes. To create a new role, go to the IAM console.

Select role

The selected role must not be used by a self-managed node group as this could lead to a service interruption upon managed node group deletion.

Learn more

Launch template Info



Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.
node-policy
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.

Description
Add a short explanation for this role.
Allows EC2 instances to call AWS services on your behalf.
Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,.@-/\[{()}]!#$%^*();:"'

Step 1: Select trusted entities

Trust policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

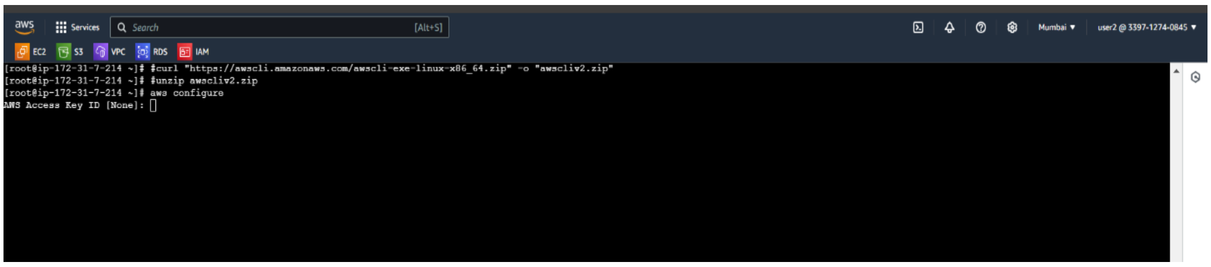
Step 2: Add permissions

Permissions policy summary

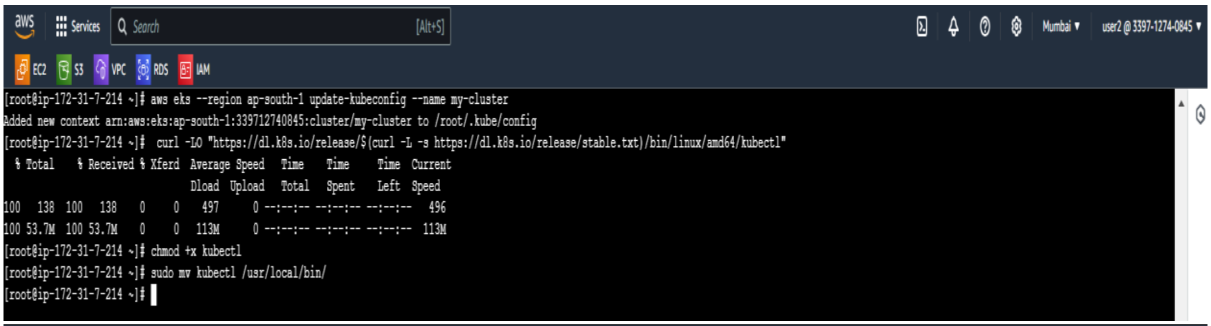
Policy name	Type	Attached as
AmazonEC2ContainerRegistryReadOnly	AWS managed	Permissions policy
AmazonEKS_CNI_Policy	AWS managed	Permissions policy
AmazonEKSWorkerNodePolicy	AWS managed	Permissions policy

Step 3: Add tags

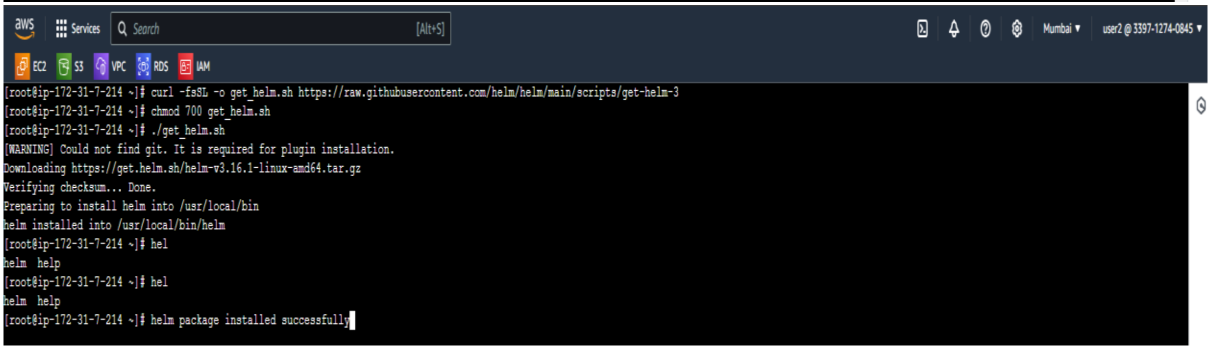
- **After the successfully created cluster and nodes you need to enter to node instance and configure services: (kubectl, helm, aws-cli):**



```
[root@ip-172-31-7-214 ~]# #curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
[root@ip-172-31-7-214 ~]# #unzip awscliv2.zip
[root@ip-172-31-7-214 ~]# aws configure
AWS Access Key ID [None]: 
```



```
[root@ip-172-31-7-214 ~]# aws eks --region ap-south-1 update-kubeconfig --name my-cluster
Added new context arn:aws:eks:ap-south-1:339712740845:cluster/my-cluster to /root/.kube/config
[root@ip-172-31-7-214 ~]#  curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    497      0 --:--:-- --:--:-- --:--:--   496
100 53.7M  100 53.7M    0     0   113M      0 --:--:-- --:--:-- --:--:--  113M
[root@ip-172-31-7-214 ~]# chmod +x kubectl
[root@ip-172-31-7-214 ~]# sudo mv kubectl /usr/local/bin/
[root@ip-172-31-7-214 ~]# 
```



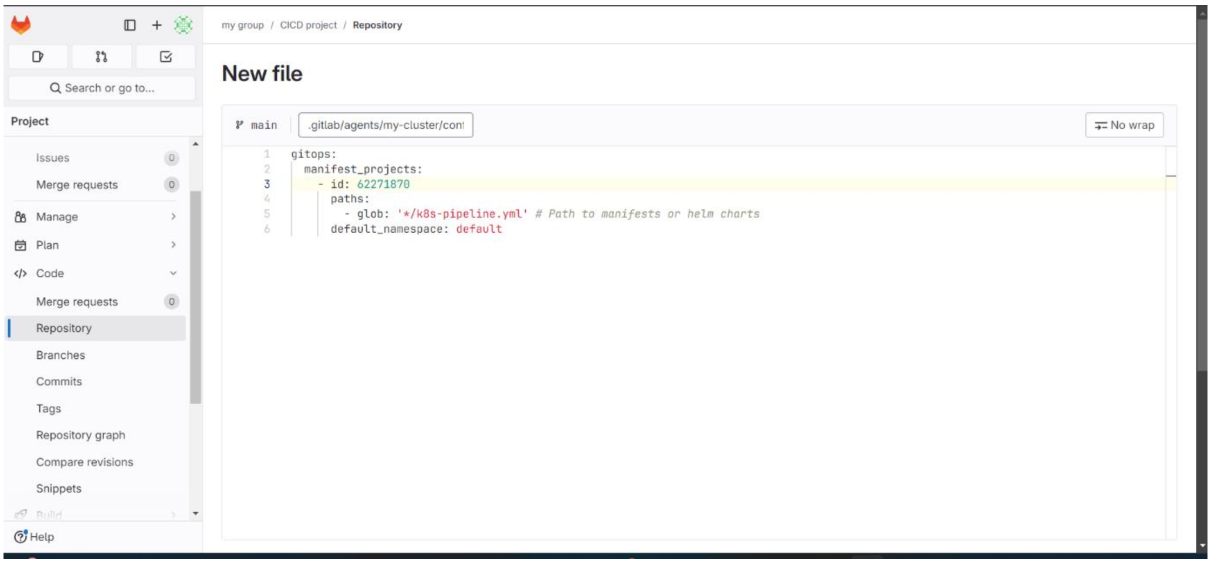
```
[root@ip-172-31-7-214 ~]# curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
[root@ip-172-31-7-214 ~]# chmod 700 get_helm.sh
[root@ip-172-31-7-214 ~]# ./get_helm.sh
[WARNING] Could not find git. It is required for plugin installation.
Downloading https://get.helm.sh/helm-v3.16.1-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
[root@ip-172-31-7-214 ~]# hel
helm  help
[root@ip-172-31-7-214 ~]# hel
helm  help
[root@ip-172-31-7-214 ~]# helm package installed successfully
```

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.

GitLab-CI-CD pipeline created using DockerHub and Deployed on EKS-Cluster

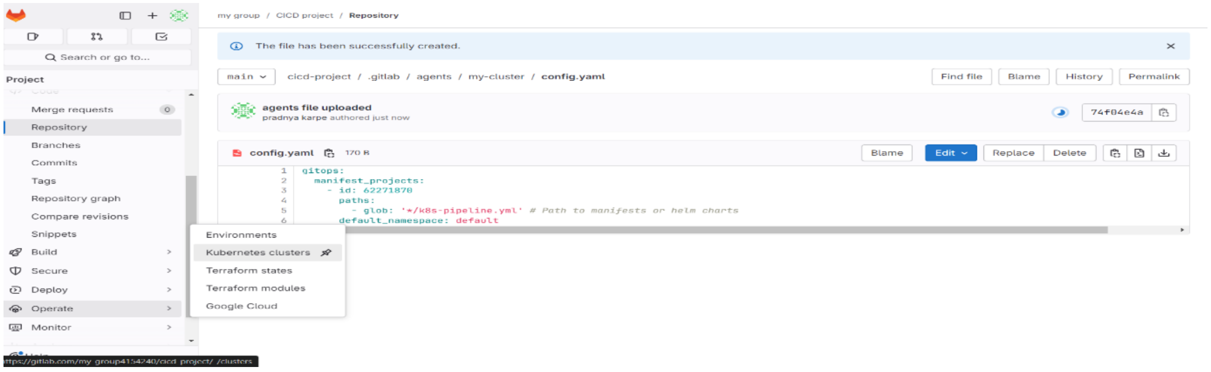

my group / CICD project / Repository

New file

main .gitlab/agents/my-cluster/conf

```
gitops:
  manifest_projects:
    - id: 62271870
      paths:
        - glob: '*/k8s-pipeline.yml' # Path to manifests or helm charts
      default_namespace: default
```

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


my group / CICD project / Repository

The file has been successfully created.

main / cicd-project / .gitlab / agents / my-cluster / config.yaml

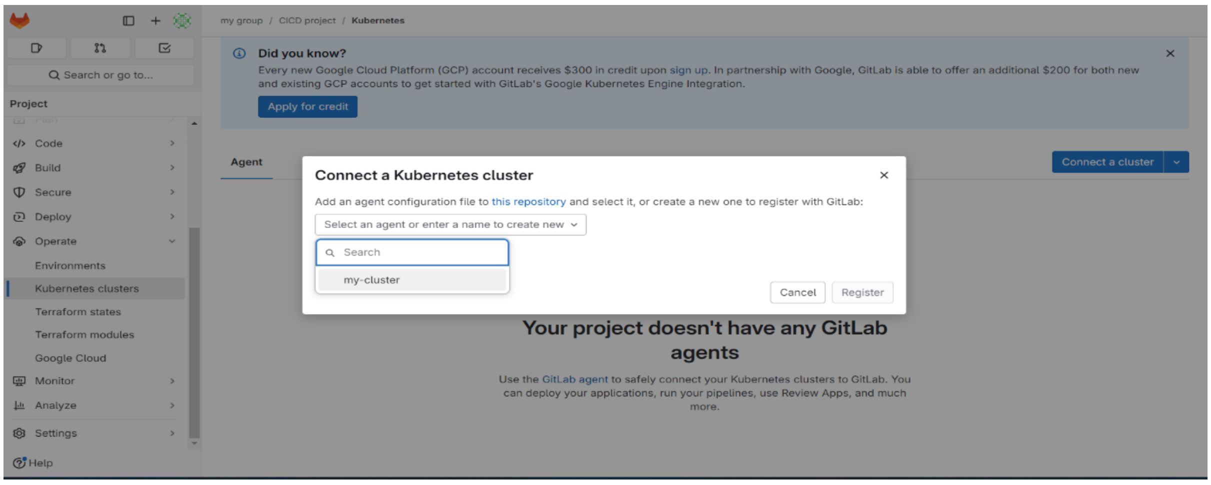
agents file uploaded
pradnya karpe authored just now
74f04e4a

config.yaml 170 B

```
gitops:
  manifest_projects:
    - id: 62271870
      paths:
        - glob: '*/k8s-pipeline.yml' # Path to manifests or helm charts
      default_namespace: default
```

ttps://gitlab.com/my_group4154240/cicd_project/_/clusters

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


my group / CICD project / Kubernetes

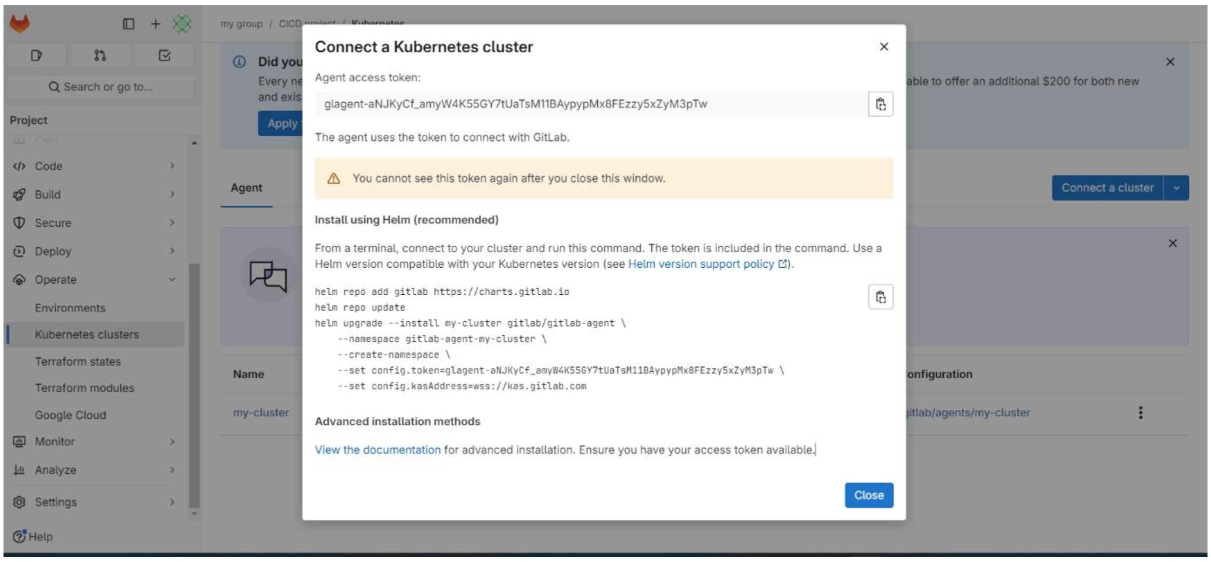
Did you know?
Every new Google Cloud Platform (GCP) account receives $300 in credit upon sign up. In partnership with Google, GitLab is able to offer an additional $200 for both new and existing GCP accounts to get started with GitLab's Google Kubernetes Engine Integration.

Connect a Kubernetes cluster
Add an agent configuration file to this repository and select it, or create a new one to register with GitLab:
Select an agent or enter a name to create new
my-cluster

Your project doesn't have any GitLab agents
Use the GitLab agent to safely connect your Kubernetes clusters to GitLab. You can deploy your applications, run your pipelines, use Review Apps, and much more.

GitLab-CI-CD pipeline created using DockerHub and Deployed on EKS-Cluster

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



Connect a Kubernetes cluster

Agent access token:

glagent-aNJKyCf_amyW4K55GY7tUaTsM11BAypypMx8FEzzy5xZyM3pTw

The agent uses the token to connect with GitLab.

You cannot see this token again after you close this window.

Install using Helm (recommended)

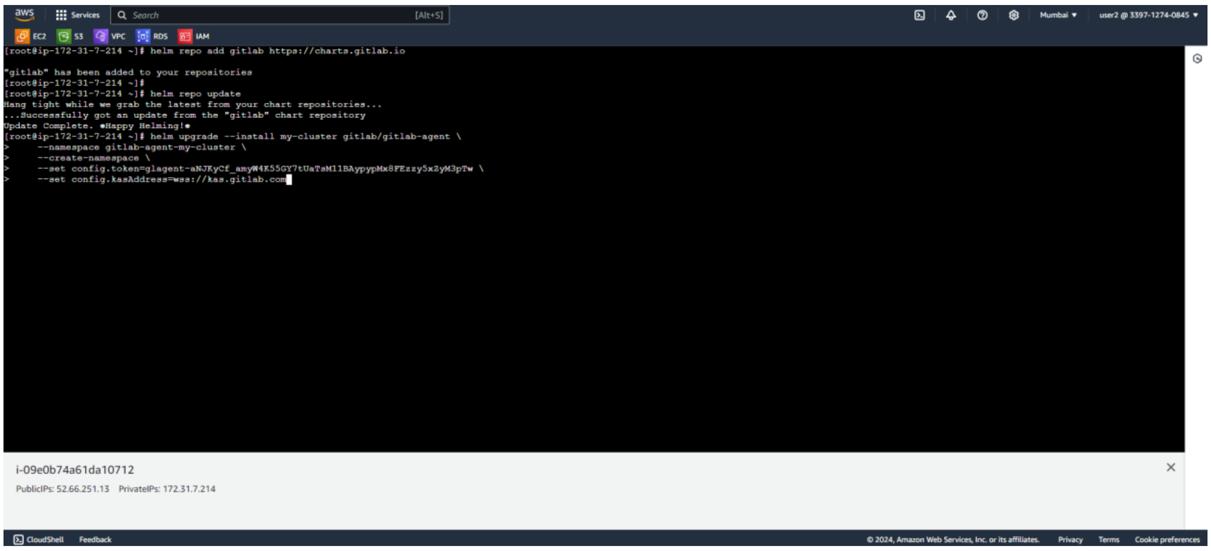
From a terminal, connect to your cluster and run this command. The token is included in the command. Use a Helm version compatible with your Kubernetes version (see Helm version support policy).

```
helm repo add gitlab https://charts.gitlab.io
helm repo update
helm upgrade --install my-cluster gitlab/gitlab-agent \
    --namespace gitlab-agent-my-cluster \
    --create-namespace \
    --set config.token=glagent-aNJKyCf_amyW4K55GY7tUaTsM11BAypypMx8FEzzy5xZyM3pTw \
    --set config.kasAddress=wss://kas.gitlab.com
```

Advanced installation methods

View the documentation for advanced installation. Ensure you have your access token available.

- Hit these generated commands on AWS-EKS node:



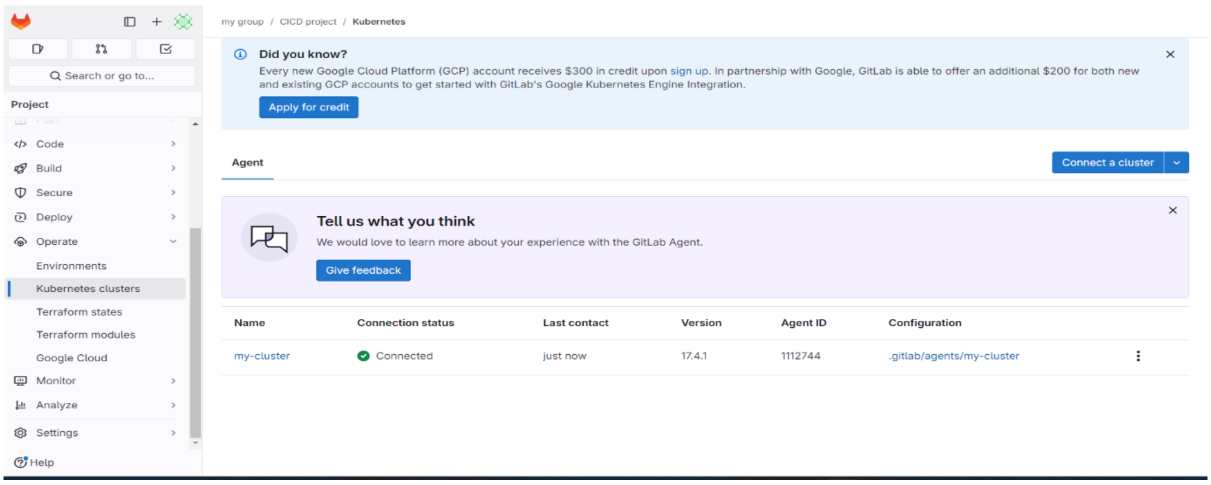
```
[root@ip-172-31-7-214 ~]# helm repo add gitlab https://charts.gitlab.io

"gitlab" has been added to your repositories
[root@ip-172-31-7-214 ~]#
[root@ip-172-31-7-214 ~]# helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "gitlab" chart repository
Update Complete. ⎈Happy Helming!⎈
[root@ip-172-31-7-214 ~]# helm upgrade --install my-cluster gitlab/gitlab-agent \
>     --namespace gitlab-agent-my-cluster \
>     --create-namespace \
>     --set config.token=glagent-aNJKyCf_amyW4K55GY7tUaTsM11BAypypMx8FEzzy5xZyM3pTw \
>     --set config.kasAddress=wss://kas.gitlab.com
```

i-09e0b74a61da10712

PublicIPs: 52.66.251.13 PrivateIPs: 172.31.7.214

- Now, it will show the status of connection:



Did you know?

Every new Google Cloud Platform (GCP) account receives $300 in credit upon sign up. In partnership with Google, GitLab is able to offer an additional $200 for both new and existing GCP accounts to get started with GitLab's Google Kubernetes Engine Integration.

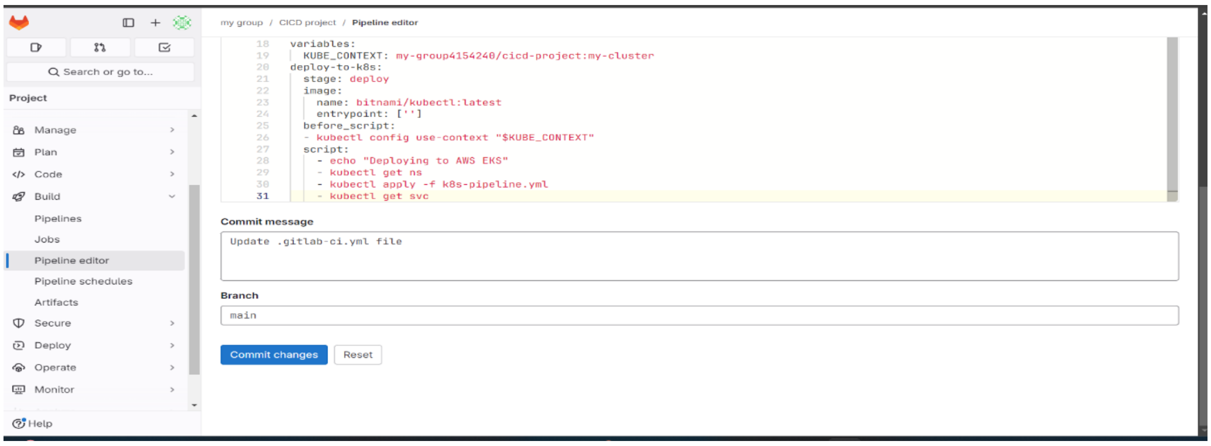
Tell us what you think

We would love to learn more about your experience with the GitLab Agent.

Name	Connection status	Last contact	Version	Agent ID	Configuration
my-cluster	Connected	just now	17.4.1	1112744	.gitlab/agents/my-cluster

GitLab-CI-CD pipeline created using DockerHub and Deployed on EKS-Cluster

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

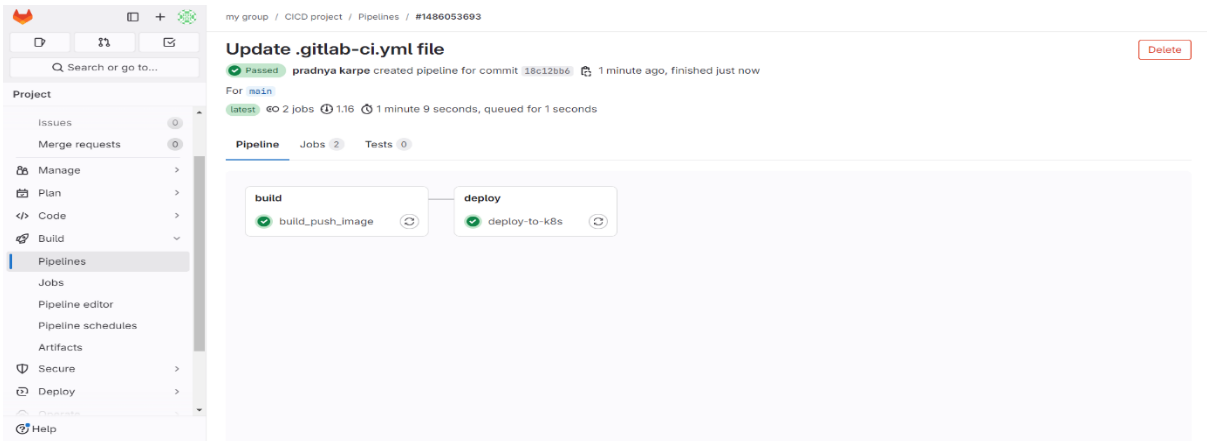


```
variables:
  KUBE_CONTEXT: my-group4154240/cicd-project:my-cluster
deploy-to-k8s:
  stage: deploy
  image:
    name: bitnami/kubectl:latest
    entrypoint: ['']
  before_script:
  - kubectl config use-context "$KUBE_CONTEXT"
  script:
    - echo "Deploying to AWS EKS"
    - kubectl get ns
    - kubectl apply -f k8s-pipeline.yml
    - kubectl get svc
```

Commit message: Update .gitlab-ci.yml file

Branch: main

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

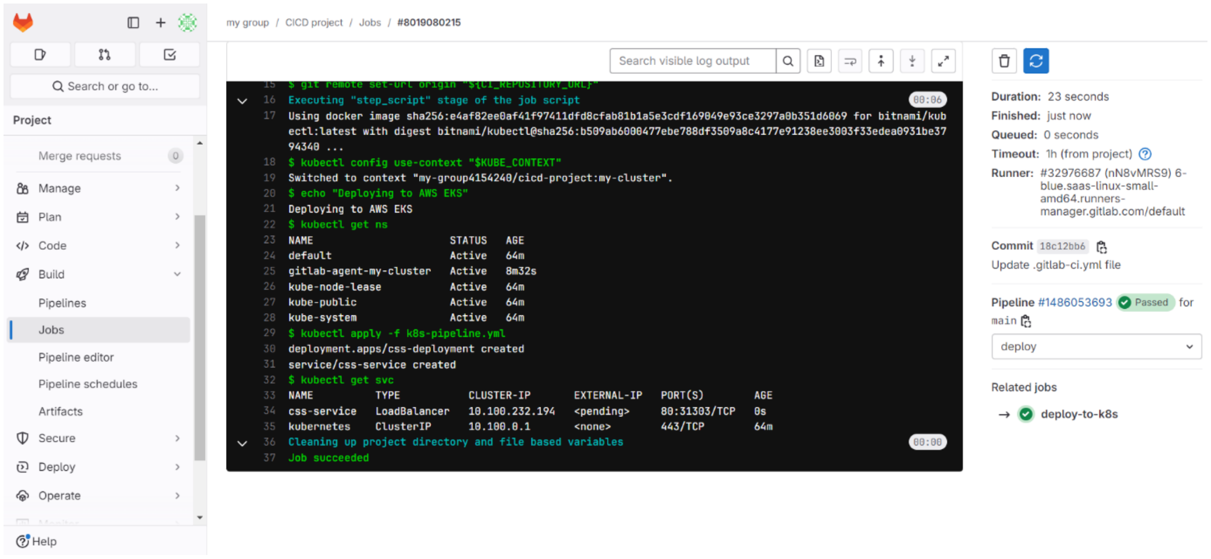


Update .gitlab-ci.yml file

Passed pradnya karpe created pipeline for commit 18c12bb6 1 minute ago, finished just now

For main

latest 2 jobs 1.16 1 minute 9 seconds, queued for 1 seconds



```
$ git remote set-url origin "${CI_REPOSITORY_URL}"
Executing "step_script" stage of the job script                                          00:06
Using docker image sha256:e4af82ee0af41f97411dfd8cfab81b1a5e3cdf169049e93ce3297a0b351d6069 for bitnami/kub
ectl:latest with digest bitnami/kubectl@sha256:b509ab6000477ebe788df3509a8c4177e91238ee3003f33edea0931be37
94340 ...
$ kubectl config use-context "$KUBE_CONTEXT"
Switched to context "my-group4154240/cicd-project:my-cluster".
$ echo "Deploying to AWS EKS"
Deploying to AWS EKS
$ kubectl get ns
NAME                      STATUS   AGE
default                   Active   64m
gitlab-agent-my-cluster   Active   8m32s
kube-node-lease           Active   64m
kube-public               Active   64m
kube-system               Active   64m
$ kubectl apply -f k8s-pipeline.yml
deployment.apps/css-deployment created
service/css-service created
$ kubectl get svc
NAME          TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
css-service   LoadBalancer   10.100.232.194   <pending>     80:31303/TCP   0s
kubernetes    ClusterIP      10.100.0.1       <none>        443/TCP        64m
Cleaning up project directory and file based variables                                   00:00
Job succeeded
```

Duration: 23 seconds
Finished: just now
Queued: 0 seconds
Timeout: 1h (from project)
Runner: #32976687 (nN8vMRS9) 6-blue.saas-linux-small-amd64.runners-manager.gitlab.com/default

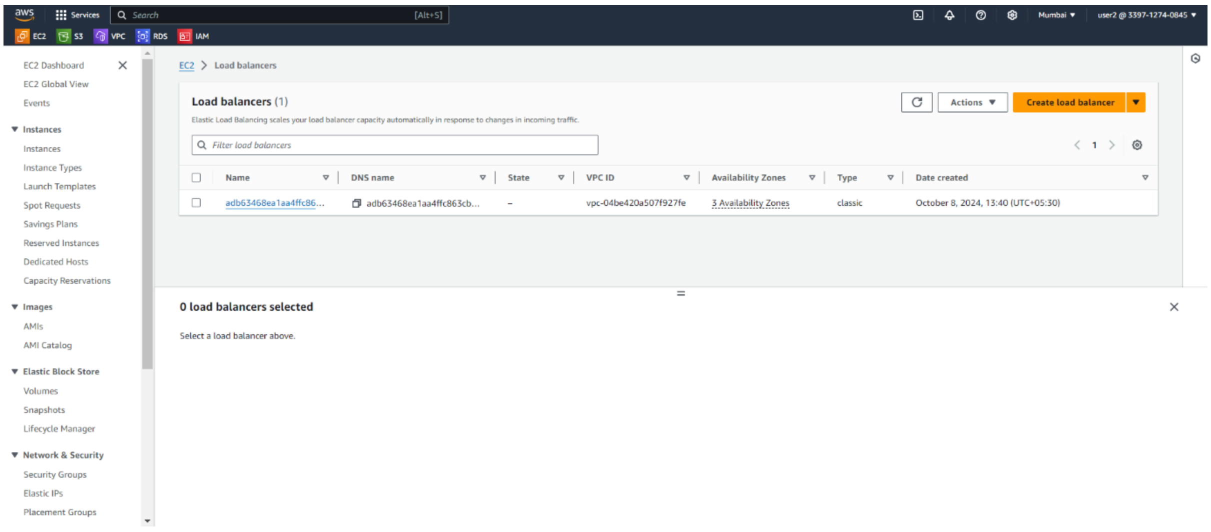
Commit 18c12bb6
Update .gitlab-ci.yml file

Pipeline #1486053693 Passed for main

deploy

Related jobs
→ deploy-to-k8s

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



EC2 > Load balancers

Load balancers (1)

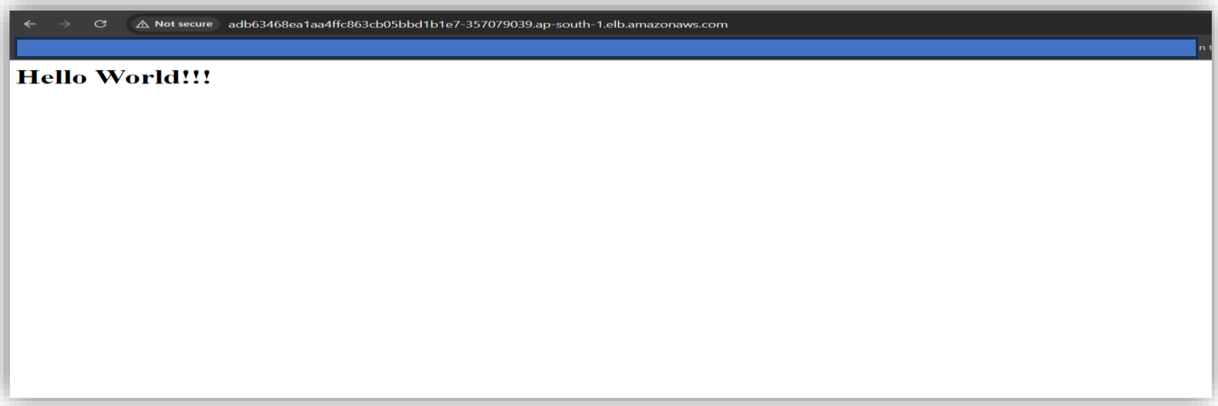
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Name	DNS name	State	VPC ID	Availability Zones	Type	Date created
adb63468ea1aa4ffc86...	adb63468ea1aa4ffc863cb...	-	vpc-04be420a507f927fe	3 Availability Zones	classic	October 8, 2024, 13:40 (UTC+05:30)

0 load balancers selected

Select a load balancer above.

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



Not secure adb63468ea1aa4ffc863cb05bbd1b1e7-357079039.ap-south-1.elb.amazonaws.com

Hello World!!!