

1. Optimizing Docker Image

Multi-stage Dockerfile to reduce the image size. ensuring only production dependencies are included.

```
FROM node:18 AS build
```

```
WORKDIR /app
```

```
COPY package.json package-lock.json* ./
```

```
RUN npm install
```

```
COPY . .
```

```
# Stage 2: Final Image (Alpine)
```

```
FROM node:18-alpine
```

```
RUN addgroup -S appgroup && adduser -S appuser -G appgroup
```

```
WORKDIR /app
```

```
COPY --from=build /app /app
```

```
RUN chown -R appuser:appgroup /app
```

```
USER appuser
```

```
EXPOSE 5000
```

```
CMD ["npm", "start"]
```

Single-stage Dockerfile

```
FROM node:18
```

```
WORKDIR /app
```

```
COPY package.json package-lock.json* ./
```

```
RUN npm install
```

```
COPY . .
```

```
EXPOSE 5000
```

```
CMD ["npm", "start"]
```

difference in size and security between the original and optimized images

Size: The single-stage build produces a larger image as it includes unnecessary development dependencies, increasing the attack surface. The multi-stage build optimizes this by producing a smaller final image containing only runtime essentials.

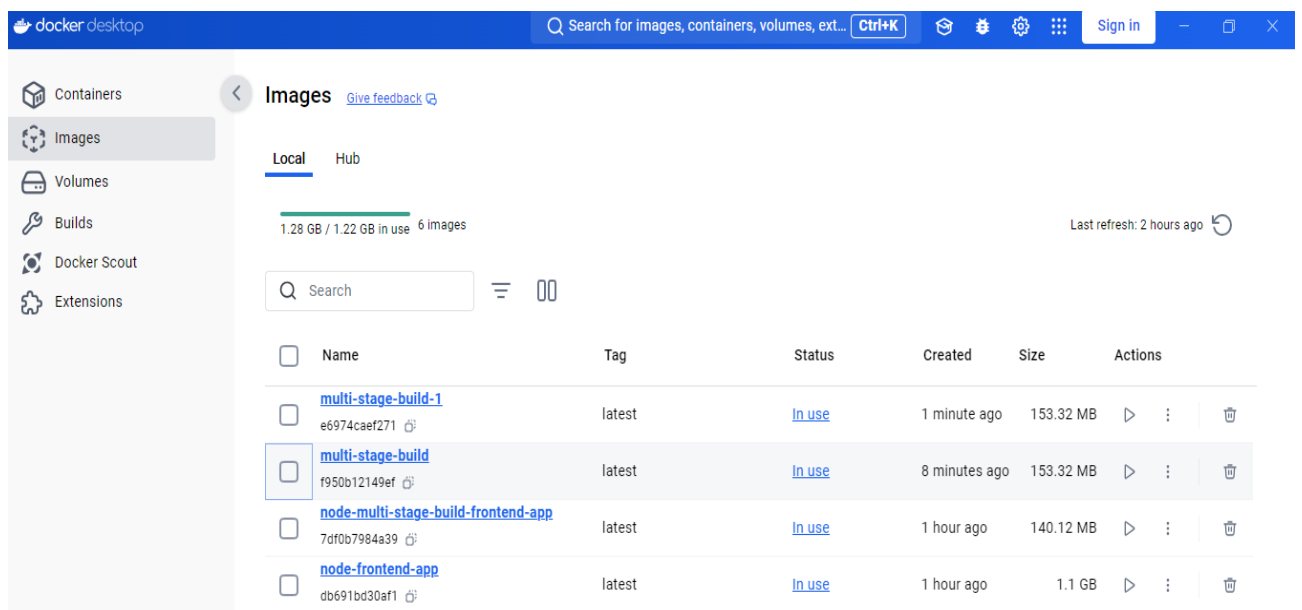
Attack Surface: The single-stage image, with its full Node.js environment, is more vulnerable to exploits targeting unused tools. In contrast, the multi-stage build minimizes this risk by using a lightweight Alpine base image.

Dependency Risks: The multi-stage build reduces the chances of including vulnerable development dependencies, which could be exploited if present in the runtime environment.

Surface Hardening: Alpine's minimal design further enhances security by including fewer packages, making it harder for attackers to find exploitable vulnerabilities.

Best Practices: Multi-stage builds align with security best practices by ensuring only production-ready code and dependencies are deployed, reducing the risk of unintended exposures.

Single stage build Image size is **1.1 GB**, while multi stage build image size is **153.32 MB**.



The screenshot shows the Docker Desktop interface. On the left is a sidebar with navigation options: Containers, Images (selected), Volumes, Builds, Docker Scout, and Extensions. The main panel is titled 'Images' and shows a 'Local' tab. A progress bar indicates '1.28 GB / 1.22 GB in use' for '6 images'. Below this is a search bar and a table of local images. The table has columns for Name, Tag, Status, Created, Size, and Actions. Four images are listed, all with the 'latest' tag and 'In use' status. The 'multi-stage-build' image is highlighted, showing a size of 153.32 MB. The 'node-frontent-app' image is the largest at 1.1 GB.

Name	Tag	Status	Created	Size	Actions
multi-stage-build-1 e6974caef271	latest	In use	1 minute ago	153.32 MB	▶ ⋮ 🗑
multi-stage-build f950b12149ef	latest	In use	8 minutes ago	153.32 MB	▶ ⋮ 🗑
node-multi-stage-build-frontend-app 7df0b7984a39	latest	In use	1 hour ago	140.12 MB	▶ ⋮ 🗑
node-frontent-app db691bd30af1	latest	In use	1 hour ago	1.1 GB	▶ ⋮ 🗑

2. Deployment of node app on Minikube Cluster via Helm

Prerequisites:- Before you begin, make sure you have the following tools installed and running on your local machine:

- kubectl (Kubernetes command-line tool)
- Docker
- Minikube

Clone the project repository to your local machine:

Github repo [url:- https://github.com/shivam086r/Cloudrove-EKS-Helm-CICD.git](https://github.com/shivam086r/Cloudrove-EKS-Helm-CICD.git)

Commands:-

```
git clone https://github.com/shivam086r/Cloudrove-EKS-Helm-CICD.git
```

```
cd Cloudrove-EKS-Helm-CICD
```

```
helm install myapp ./myapp-helm-chart
```

```
kubectl get svc
```

```
minikube service nodejs-app --url
```

(Note:- You can access the service via this URL)



Host Information

Welcome to User Page

Insert new user to list		User list			
Name:	<input type="text"/>	<table border="1"><thead><tr><th>Name</th><th>Email</th></tr></thead><tbody></tbody></table>	Name	Email	
Name	Email				
Email:	<input type="text"/>				
<input type="button" value="SUBMIT"/>					

Adding ingress controller

Commands:-

minikube addons enable ingress

kubectl get pods -A | grep nginx

```
Shivam@DESKTOP-SVEAR2R MINGW64 /d/Clouddrove-project (main)
$ helm list
NAME          NAMESPACE    REVISION    UPDATED              STATUS    CHART
nodejs-app    default       1           2025-01-12 14:09:34.5473505 +0530 IST  deployed  nodejs-app-0.1.0

Shivam@DESKTOP-SVEAR2R MINGW64 /d/Clouddrove-project (main)
$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nodejs-app-6d76745d57-6c22d        1/1     Running   0           114m
nodejs-app-6d76745d57-qhgw7        1/1     Running   0           114m
nodejs-app-6d76745d57-z7t7k        1/1     Running   1 (113m ago) 114m

Shivam@DESKTOP-SVEAR2R MINGW64 /d/Clouddrove-project (main)
$ kubectl get svc
NAME          TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes   ClusterIP   10.96.0.1     <none>         443/TCP          3h52m
nodejs-app    NodePort    10.111.252.35 <none>         80:30080/TCP     114m

Shivam@DESKTOP-SVEAR2R MINGW64 /d/Clouddrove-project (main)
$ kubectl get ingress
NAME          CLASS    HOSTS          ADDRESS          PORTS    AGE
ingress-example  nginx   nodejs.local   192.168.49.2    80       115m

Shivam@DESKTOP-SVEAR2R MINGW64 /d/Clouddrove-project (main)
$ kubectl get pods -A | grep nginx
ingress-nginx   ingress-nginx-admission-create-tbp5d        0/1     Completed    0       3m51s
ingress-nginx   ingress-nginx-admission-patch-hr9d1         0/1     Completed    0       3m51s
ingress-nginx   ingress-nginx-controller-768f948f8f-t6mr9    1/1     Running      0       3m51s

Shivam@DESKTOP-SVEAR2R MINGW64 /d/Clouddrove-project (main)
$ |
```

###Adding Ingress Controller IP with hostname inside the local hosts file

###for linux open host file and add IP and domain

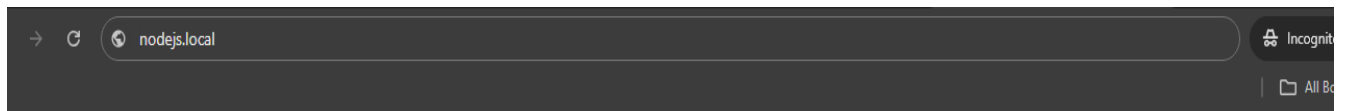
/etc/hosts

192.168.49.2 nodejs.local

###for windows open host file and add IP and domain

C:\Windows\System32\drivers\etc\hosts

192.168.49.2 nodejs.local



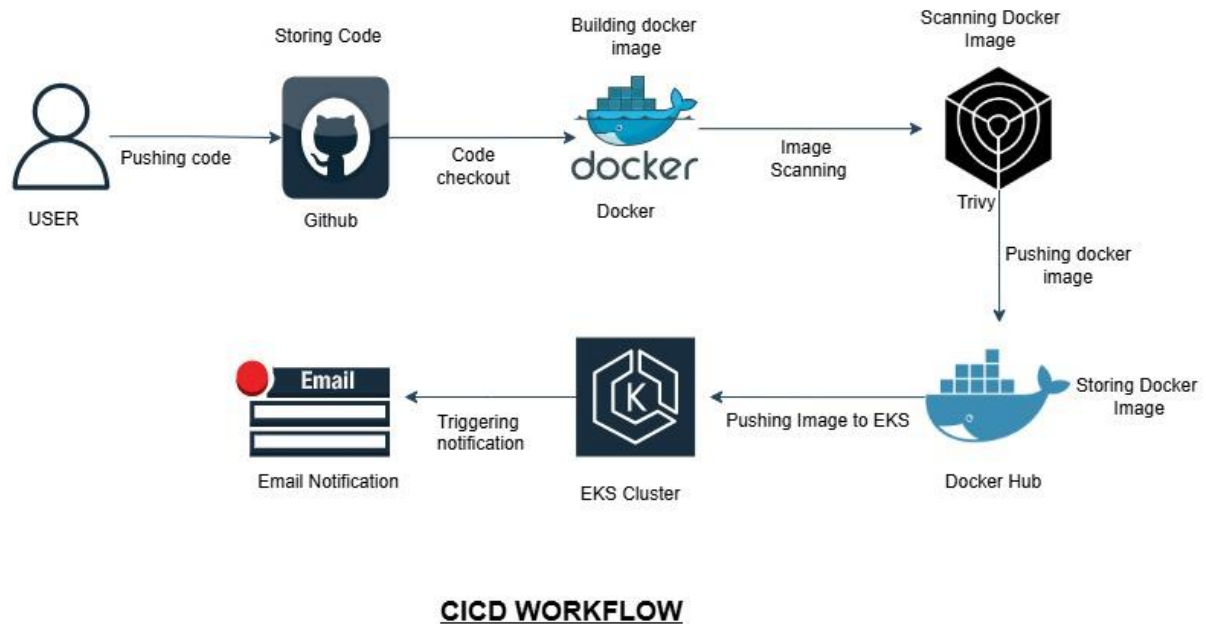
Data Server From Backend

Host: **nodejs-app-6c6fdf875f-w9rnh**, Private IP: **10.244.0.9**, Public IP: **152.59.30.231**

Welcome to User Page

Insert new user to list	User list		
Name: <input type="text"/>	<table><thead><tr><th>Name</th><th>Email</th></tr></thead></table>	Name	Email
Name	Email		
Email: <input type="text"/>			
<input type="button" value="SUBMIT"/>			

3.EKS Cluster Creation with EKSCTL, Deploying app on EKS via Helm and CICD configuration



Prerequisite:

- ➔ AWS CLI and kubectl should be installed on your machine
- ➔ Create IAM user with admin access and create access key id and secret access key

Commands:-

aws --version

aws configure

(provide your access-key-id, secret-access-key and region)

Cloning the Repo

git clone <https://github.com/shivam086r/Clouddrove-project.git>

cd Clouddrove-EKS-Helm-CICD

Creating EKS Cluster via eksctl :-

Commands:-

```
eksctl create cluster -f cluster-config.yaml
```

configuring oidc-provider and add-on vpc-cni

Commands:-

```
eksctl utils associate-iam-oidc-provider --region us-east-1 --cluster node-cluster --approve
```

```
eksctl update add-on --name vpc-cni --cluster node-cluster --region us-east-1
```

Cluster

node-cluster

[Refresh](#) [Delete cluster](#) [Upgrade version](#) [View dashboard](#)

- ❗ Your current IAM principal doesn't have access to Kubernetes objects on this cluster.
This may be due to the current user or role not having Kubernetes RBAC permissions to describe cluster resources or not having an entry in the cluster's auth config map. [Learn more](#)
- ❗ End of standard support for Kubernetes version 1.30 is July 28, 2025. On that date, your cluster will enter the extended support period with additional fees. For more information, see the [pricing page](#). [Upgrade now](#)

▼ Cluster info [Info](#)

Status ✔ Active	Kubernetes version Info 1.30	Support period ⓘ Standard support until July 28, 2025	Provider EKS
Cluster health issues ✔ 0	Upgrade insights ✔ 4 ❌ 1	Node health issues ✔ 0	

Worker-Node

Instances (1) [Info](#)

Last updated
less than a minute ago



Connect

Instance state ▼

Actions ▼

Launch

Find Instance by attribute or tag (case-sensitive)

All states ▼

Instance state = running X

Clear filters

<input type="checkbox"/>	Name 🔗	Instance ID	Instance state ▼	Instance type ▼	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	node-cluster-demo-nodes-Node	i-032b6eb58c6402a4b	Running 🔍 🔍	t2.medium	✔ 2/2 checks passed	View alarms +	us-east-1c

Deploying app on EKS Cluster

Updating kubeconfig file

Command:-

```
aws eks --region us-east-1 update-kubeconfig --name <cluster-name>
```

Deploying app via helm-Chart

Command:-

kubectl create namespace app-namespace

helm install myapp ./myapp-helm-chart

kubectl get all -n app-namespace

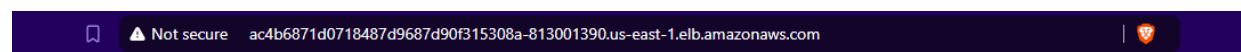
```
Shivam@DESKTOP-SVEAR2R MINGW64 /d/CloudDrove-update-project/test-clouddrove (main)
$ kubectl get all -n app-namespace
NAME                                READY    STATUS    RESTARTS   AGE
pod/mongodb-8444b757d6-bc6gx        1/1      Running   0           49m
pod/nodejs-app-6d78fcb5c8-7cfkb     1/1      Running   0           6m30s
pod/nodejs-app-6d78fcb5c8-ggc42     1/1      Running   0           6m32s

NAME                                TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
service/mongodb-service             ClusterIP      10.100.57.136   <none>            27017/TCP        49m
service/nodejs-service              LoadBalancer  10.100.24.73    ac4b6871d0718487d9687d90f315308a-813001390.us-east-1.elb.amazonaws.com  80:30675/TCP    49m

NAME                READY    UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mongodb  1/1      1             1           49m
deployment.apps/nodejs-app  2/2      2             2           49m

NAME                DESIRED   CURRENT   READY   AGE
replicaset.apps/mongodb-8444b757d6  1         1         1       49m
replicaset.apps/nodejs-app-6d78fcb5c8  2         2         2       6m34s
replicaset.apps/nodejs-app-7bbbdd85d4  0         0         0       49m
```

Accessing app on browser via Load balancer URL and adding Data



Data Server From Backend

Host: **nodejs-app-6d78fcb5c8-7cfkb**, Private IP: **192.168.10.22**, Public IP: **18.234.228.212**

User added successfully

Insert new user to list

Name:

Email:

User list

Name	Email
raj256	suresh@123.com
raj@	raj@gmail.com
hello	shivam123@gmail.com

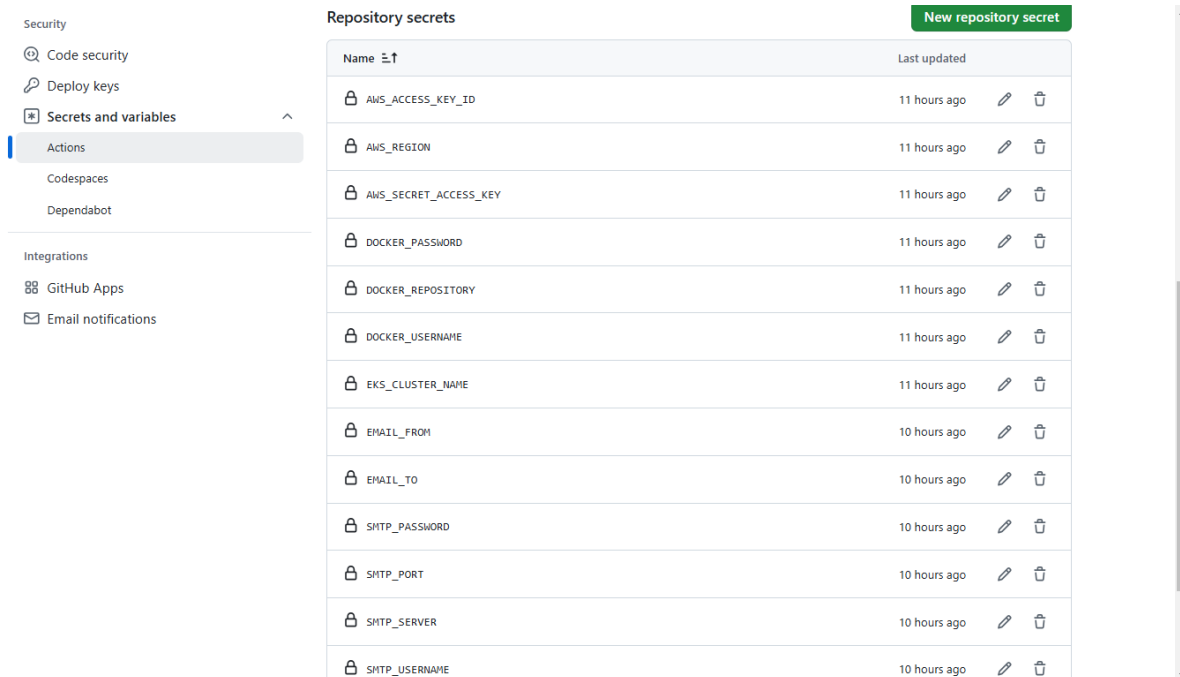
Checking /health endpoint:-

```
< > ↻ 🔒 Not secure ac4b6871d0718487d9687d90f315308a-813001390.us-east-1.elb.amazonaws.com/health 🔍 🛡️
Pretty-print ☐
{"status":"healthy","message":"Server is running smoothly"}
```


Setting up CICD pipeline

creating `.github/workflow/deploy.yaml` file and adding all the necessary CICD steps there

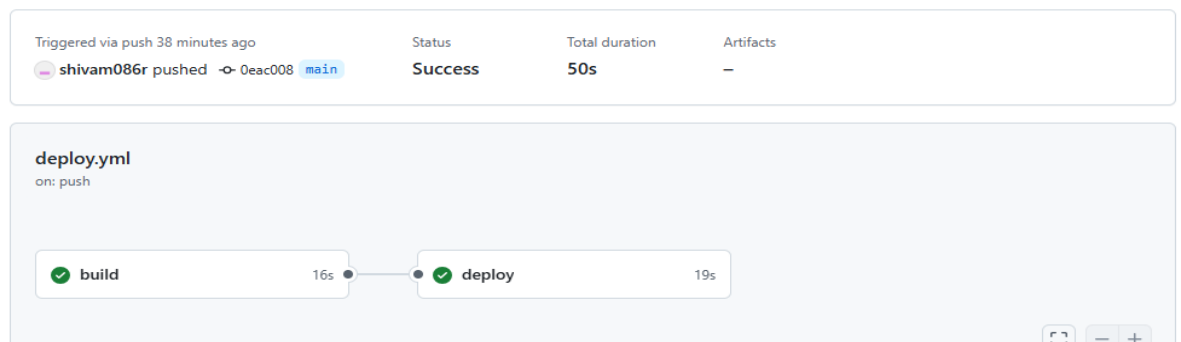
In your repository settings, add the following secrets:



The screenshot shows the 'Repository secrets' page in GitHub. On the left, the 'Security' sidebar is open, with 'Secrets and variables' selected. The main area displays a table of repository secrets. A 'New repository secret' button is in the top right corner. The table has two columns: 'Name' and 'Last updated'. It lists 14 secrets, all created 10 or 11 hours ago, with edit and delete icons for each.

Name	Last updated
AWS_ACCESS_KEY_ID	11 hours ago
AWS_REGION	11 hours ago
AWS_SECRET_ACCESS_KEY	11 hours ago
DOCKER_PASSWORD	11 hours ago
DOCKER_REPOSITORY	11 hours ago
DOCKER_USERNAME	11 hours ago
EKS_CLUSTER_NAME	11 hours ago
EMAIL_FROM	10 hours ago
EMAIL_TO	10 hours ago
SMTP_PASSWORD	10 hours ago
SMTP_PORT	10 hours ago
SMTP_SERVER	10 hours ago
SMTP_USERNAME	10 hours ago

After pushing the changes in repo, Github actions work flow stages :-



The screenshot shows a GitHub Actions workflow run for the 'deploy.yml' file. The workflow was triggered by a push 38 minutes ago by user 'shivam086r' to the 'main' branch. The status is 'Success' with a total duration of 50s. The workflow diagram shows two stages: 'build' (16s) and 'deploy' (19s), both of which completed successfully, indicated by green checkmarks.

Triggered via push 38 minutes ago
shivam086r pushed -> 0eac008 main
Status: Success
Total duration: 50s
Artifacts: -

deploy.yml
on: push

build (16s) → deploy (19s)

Build stage workflow:-

build
succeeded 40 minutes ago in 16s

Search logs

> Set up job

> Checkout code

> Set up Docker Hub credentials

> Build Docker image

> Push Docker image to Docker Hub

> Post Set up Docker Hub credentials

> Post Checkout code

> Complete job

0s

1s

0s

9s

3s

0s

0s

0s

Deploy-stage-workflow:-

deploy
succeeded 42 minutes ago in 19s

Search logs

> Set up job

> Checkout code

> Set up AWS credentials

> Set up kubectl

> Set kubeconfig

> Update the Docker image in Kubernetes deployment

> Check Kubernetes deployment status

> Post Set up AWS credentials

> Post Checkout code

> Complete job

2s

1s

1s

1s

4s

5s




1s

0s

0s

0s

Updated changes on the webpage after pushing new code to GitHub and deploying the new Docker image to the EKS cluster via the CI/CD pipeline

  Not secure ac4b6871d0718487d9687d90f315308a-813001390.us-east-1.elb.amazonaws.com 

Hello Everyone, Have a great day

Host: **nodejs-app-66cbd7b59c-dcwlg**, Private IP: **192.168.5.13**, Public IP: **18.234.228.212**

Welcome to User Page

Insert new user to list

Name:

Email:

User list








Name	Email
raj256	suresh@123.com
raj@	raj@gmail.com
hello	shivam123@gmail.com

Verifying the health of the deployed application using a `/health` endpoint.

Adding code for `/health` endpoint code in `index.js` file and commit the code:

```
// Health check route
app.get('/health', (req, res) => {
  res.status(200).json({
    status: 'healthy',
    message: 'Server is running smoothly'
  });
});
```

Checking health Endpoint

     Not secure ac4b6871d0718487d9687d90f315308a-813001390.us-east-1.elb.amazonaws.com/health  

Pretty-print ☐

```
{"status": "healthy", "message": "Server is running smoothly"}
```