Aim: Integrate Kubernetes and Docker. Automate the process of running containerized applications developed in above using Kubernetes

Objective:

Containerize a Node.js application using Docker.

Push the image to Docker Hub.

Deploy and manage the application using Kubernetes.

Automate deployment using GitHub Actions CI/CD workflow.

Expose and scale the application using Kubernetes tools.

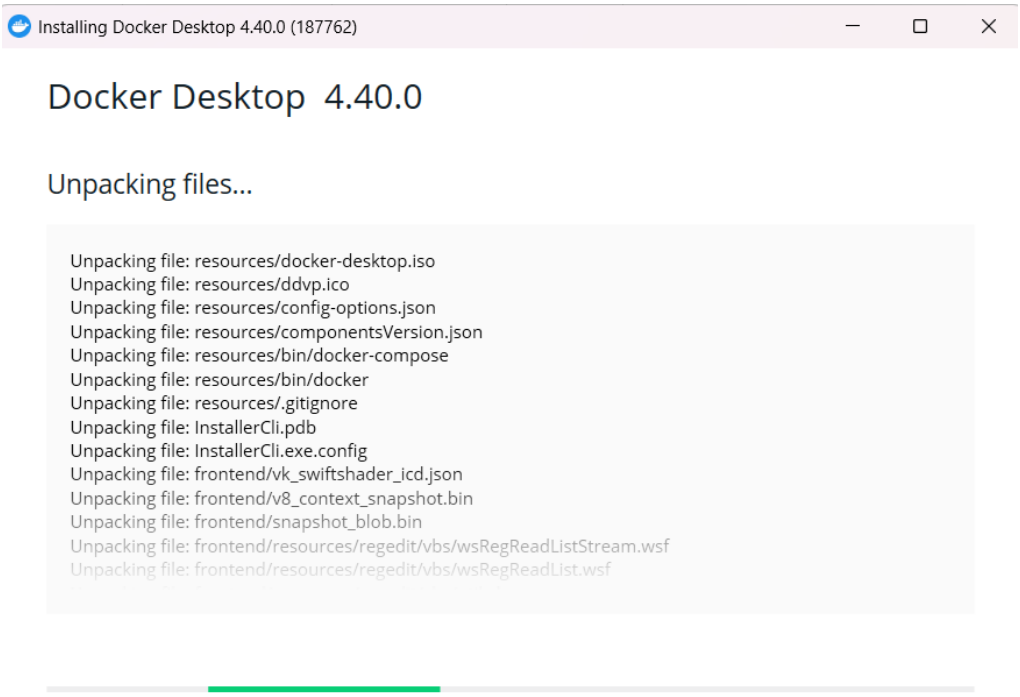
Description:

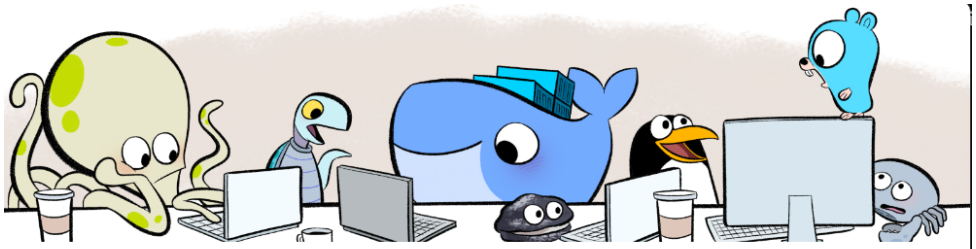
This project demonstrates how Docker and Kubernetes can be integrated to streamline application development and deployment. A simple Node.js web server is containerized using Docker, uploaded to Docker Hub, and deployed to a Kubernetes cluster. The deployment process is automated with GitHub Actions, allowing continuous integration and delivery. This ensures consistency, scalability, and reliability of the app in a production-like environment.

Steps Required:

1. Containerize the Application
2. Push Image to Docker Hub
3. Create Kubernetes Manifests
4. Verify Deployment
5. Automate Deployment with GitHub Actions
6. Scale and Manage the Application

Implementation:





Docker Subscription Service Agreement

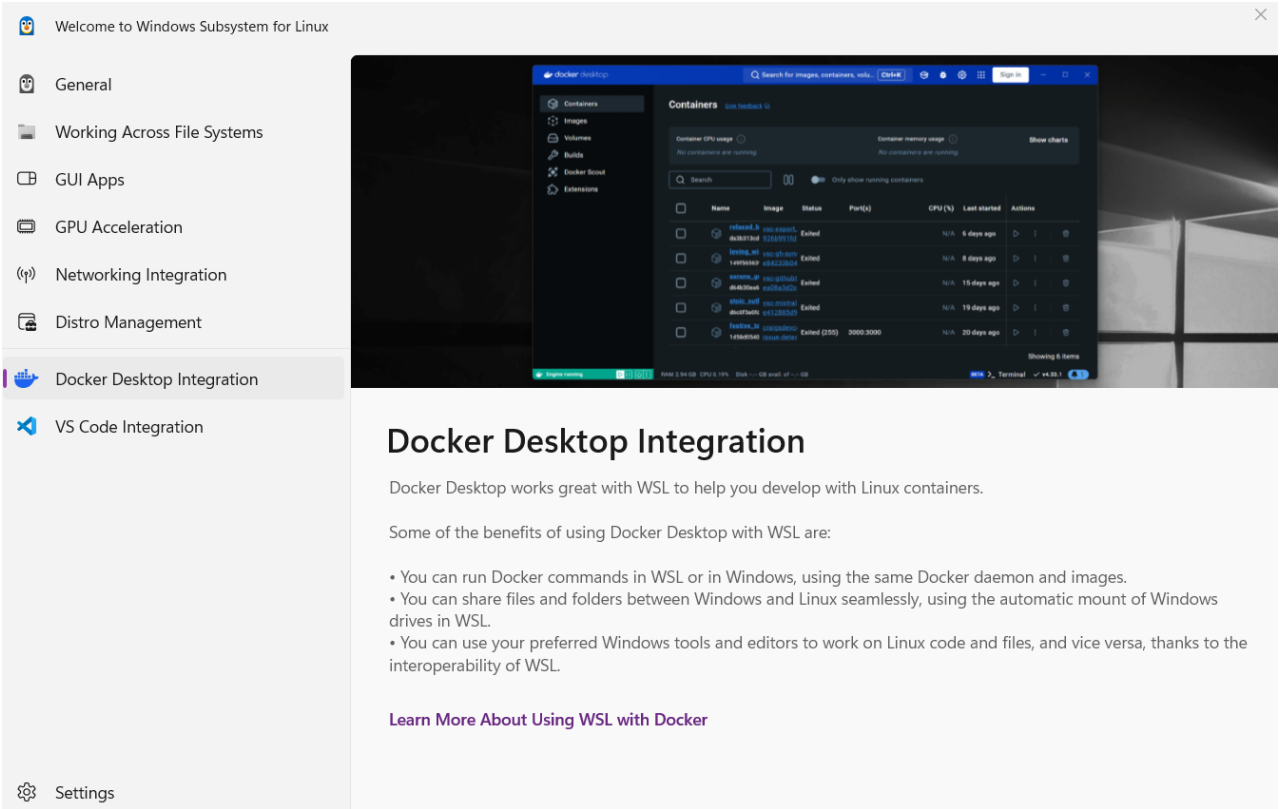
By selecting **accept**, you agree to the Subscription Service Agreement, the Docker Data Processing Agreement, and the Data Privacy Policy.

Commercial use of Docker Desktop at a company of more than 250 employees OR more than $10 million in annual revenue requires a paid subscription (Pro, Team, or Business). See subscription details

View Full Terms | Accept | Close

```
and 'wsl.exe --install <Distro>' to install.
PS C:\WINDOWS\system32> wsl --list --online
>>
The following is a list of valid distributions that can be installed.
Install using 'wsl.exe --install <Distro>'.

NAME                            FRIENDLY NAME
AlmaLinux-8                     AlmaLinux OS 8
AlmaLinux-9                     AlmaLinux OS 9
AlmaLinux-Kitten-10             AlmaLinux OS Kitten 10
Debian                          Debian GNU/Linux
FedoraLinux-42                  Fedora Linux 42
SUSE-Linux-Enterprise-15-SP5    SUSE Linux Enterprise 15 SP5
SUSE-Linux-Enterprise-15-SP6    SUSE Linux Enterprise 15 SP6
Ubuntu                          Ubuntu
Ubuntu-24.04                    Ubuntu 24.04 LTS
archlinux                       Arch Linux
kali-linux                      Kali Linux Rolling
openSUSE-Tumbleweed             openSUSE Tumbleweed
openSUSE-Leap-15.6              openSUSE Leap 15.6
Ubuntu-18.04                    Ubuntu 18.04 LTS
Ubuntu-20.04                    Ubuntu 20.04 LTS
Ubuntu-22.04                    Ubuntu 22.04 LTS
OracleLinux_7_9                 Oracle Linux 7.9
OracleLinux_8_7                 Oracle Linux 8.7
OracleLinux_9_1                 Oracle Linux 9.1
PS C:\WINDOWS\system32> wsl --install -d Ubuntu
>>
Downloading: Ubuntu
[========                  14.6%                          ]
```




```
PS C:\WINDOWS\system32> wsl -d Ubuntu
>>
Provisioning the new WSL instance Ubuntu
This might take a while...
Create a default Unix user account: geeta
New password:
Retype new password:
passwd: password updated successfully
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 5.15.167.4-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun Apr 20 07:14:58 UTC 2025

  System load:  0.06               Processes:             32
  Usage of /:   0.1% of 1006.85GB  Users logged in:       0
  Memory usage: 5%                 IPv4 address for eth0: 172.17.247.16
  Swap usage:   0%


This message is shown once a day. To disable it please create the
/home/geeta/.hushlogin file.
geeta@Geeta:/mnt/c/WINDOWS/system32$
```

```
C:\Users\geeta>docker --version
Docker version 28.0.4, build b8034c0

C:\Users\geeta>docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:c41088499908a59aae84b0a49c70e86f4731e588a737f1637e73c8c09d995654
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Developing a Simple Containerized Application

Let’s create a simple Node.js web application that serves a "Hello, Docker!" message. We will containerize this application using Docker.

Step 1: Create the Application

1. **Create a new directory for your application.**

```
PS C:\Users\geeta\OneDrive\Desktop> mkdir my-docker-app


    Directory: C:\Users\geeta\OneDrive\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        20-04-2025     14:21                my-docker-app
```

Create the Node.js application:

- Create a file called app.js with the following content:

```
JS app.js > ...
const http = require('http');
const port = 8080;

const requestHandler = (req, res) => {
  res.write('Hello, Docker!');
  res.end();
};

const server = http.createServer(requestHandler);

server.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

Create a package.json file:

- Run npm init -y to generate a basic package.json file, and then run: npm install http

```
PS C:\Users\geeta\OneDrive\Desktop\my-docker-app> npm init -y
Wrote to C:\Users\geeta\OneDrive\Desktop\my-docker-app\package.json:

{
  "name": "my-docker-app",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "type": "commonjs",
  "description": ""
}
```

```
npm install http
>> C:\Users\geeta\OneDrive\Desktop\my-docker-app>

added 1 package, and audited 2 packages in 1s

found 0 vulnerabilities
PS C:\Users\geeta\OneDrive\Desktop\my-docker-app>
```

Step 2: Create the Dockerfile

1. **Create a Dockerfile in the same directory with the following content:**

```
# Use official Node.js image as the base
FROM node:23-slim


# Set the working directory inside the container
WORKDIR /usr/src/app

# Copy package.json and install dependencies
COPY package.json ./
RUN npm install

# Copy the rest of the application code
COPY . .

# Expose the port on which the app will run
EXPOSE 8080

# Command to run the application
CMD [ "node", "app.js" ]
```

2. App.js

```
const http = require('http');
const port = 8080;

const requestHandler = (req, res) => {
  res.write('Hello, Docker!');
  res.end();
};

const server = http.createServer(requestHandler);

server.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

Step 3: Build the Docker Image

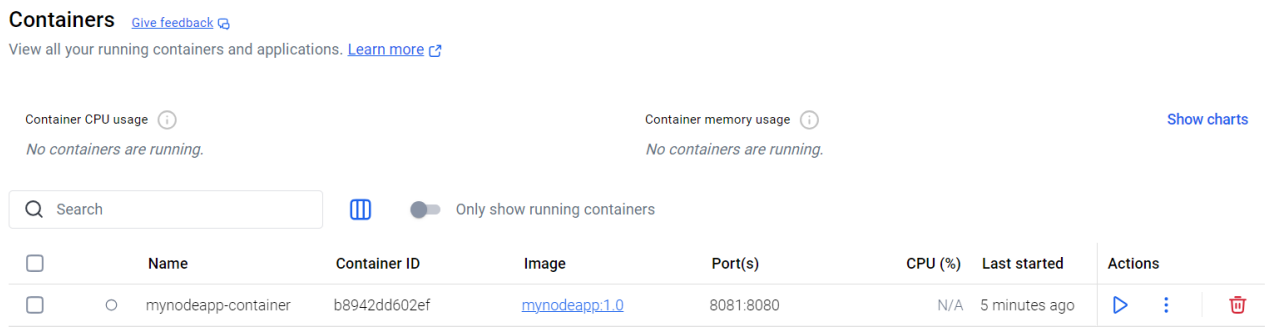
1. **Build the Docker image using the command: docker build -t mynodeapp:1.0 .**

```
PS C:\Users\geeta\OneDrive\Desktop\my-docker-app> docker build -t mynodeapp:1.0 .
[+] Building 164.2s (11/11) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 431B
 => [internal] load metadata for docker.io/library/node:14
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/5] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa
 => => resolve docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa
 => => sha256:0d27a8e861329007574c6766fba946d48e20d2c8e964e873de352603f22c4ceb 450B / 450B
 => => sha256:0c8cc2f24a4dcb64e602e086fc9446b0a541e8acd9ad72d2e90df3ba22f158b3 2.29MB / 2.29MB
 => => sha256:5f32ed3c3f278edda4fc571c880b5277355a29ae8f52b52cdf865f058378a590 35.24MB / 35.24MB
 => => sha256:6f51ee005deac0d99898e41b8ce60ebf250ebe1a31a0b03f613aec6bbc9b83d8 4.19kB / 4.19kB
 => => sha256:d9a8df5894511ce28a05e2925a75e8a4acbd0634c39ad734fdfba8e23d1b1569 191.85MB / 191.85MB
 => => sha256:1de76e268b103d05fa8960e0f77951ff54b912b63429c34f5d6adfd09f5f9ee2 51.88MB / 51.88MB
 => => sha256:3d2201bd995cccf12851a50820de03d34a17011dcbb9ac9fdf3a50c952cbb131 10.00MB / 10.00MB
 => => sha256:b253aeafeaa7e0671bb60008df01de101a38a045ff7bc656e3b0fbfc7c05cca5 7.86MB / 7.86MB
 => => sha256:2ff1d7c41c74a25258bfa6f0b8adb0a727f84518f55f65ca845ebc747976c408 50.45MB / 50.45MB
 => => extracting sha256:2ff1d7c41c74a25258bfa6f0b8adb0a727f84518f55f65ca845ebc747976c408
 => => extracting sha256:b253aeafeaa7e0671bb60008df01de101a38a045ff7bc656e3b0fbfc7c05cca5
 => => extracting sha256:3d2201bd995cccf12851a50820de03d34a17011dcbb9ac9fdf3a50c952cbb131
 => => extracting sha256:1de76e268b103d05fa8960e0f77951ff54b912b63429c34f5d6adfd09f5f9ee2
 => => extracting sha256:d9a8df5894511ce28a05e2925a75e8a4acbd0634c39ad734fdfba8e23d1b1569
 => => extracting sha256:6f51ee005deac0d99898e41b8ce60ebf250ebe1a31a0b03f613aec6bbc9b83d8
 => => extracting sha256:5f32ed3c3f278edda4fc571c880b5277355a29ae8f52b52cdf865f058378a590
 => => extracting sha256:0c8cc2f24a4dcb64e602e086fc9446b0a541e8acd9ad72d2e90df3ba22f158b3
 => => extracting sha256:0d27a8e861329007574c6766fba946d48e20d2c8e964e873de352603f22c4ceb
 => [internal] load build context
 => => transferring context: 2.47kB
 => [auth] library/node:pull token for registry-1.docker.io
 => [2/5] WORKDIR /usr/src/app
 => [3/5] COPY package.json ./
```

Step 4: Run the Docker Container

1. **Run the container: docker run -d -p 8081:8080 --name mynodeapp-container mynodeapp:1.0**

```
PS C:\Users\geeta\OneDrive\Desktop\my-docker-app> docker run -d -p 8081:8080 --name mynodeapp-container mynodeapp:1.0
b8942dd602ef929d9bc547720ff42af7ee1ff2a4635671c0397c090d3e233a93
PS C:\Users\geeta\OneDrive\Desktop\my-docker-app>
```



Images / mynodeapp:1.0

mynodeapp:1.0 IN USE

f0b7b86abe3c

CREATED 8 minutes ago | SIZE 1.34 GB | Recommended fixes | Run

Analyzed by docker scout

Layers (20)

0	ADD file:40953ed6e6f96703b2e0c13288437c2aaf8b3df3...	130.11 MB
1	CMD ["bash"]	0 B
2	set -eux; apt-get update; apt-get install -y --no-install-reco...	18.92 MB
3	set -ex; if ! command -v gpg > /dev/null; then apt-get upda...	18.78 MB
4	apt-get update && apt-get install -y --no-install-recommen...	160.46 MB
5	set -ex; apt-get update; apt-get install -y --no-install-recom...	543.22 MB
6	groupadd --gid 1000 node && useradd --uid 1000 --gid nod...	405.5 KB
7	ENV NODE_VERSION=14.21.3	0 B
8	ARCH= && dpkgArch="$(dpkg --print-architecture)" && cas...	117.58 MB
9	ENV YARN_VERSION=1.22.19	0 B

Vulnerabilities | Packages | Give feedback

This image has not been analyzed

You can use Docker Scout to analyze local images and list its vulnerabilities.

Start analysis

Enable background indexing in Settings so your results are always ready.

RAM 3.19 GB CPU 0.00% Disk: 2.84 GB used (limit 1006.85 GB) Terminal v4.40.0

Docker Content Management A | localhost:8081 | Semester 2 - Google Drive

localhost:8081

V26 | Profile | 2025 | DQ | S2 | Apply | J

```
Hello, Docker!
```

Containers Give feedback

View all your running containers and applications. Learn more

Container CPU usage: 0.00% / 1200% (12 CPUs available)

Container memory usage: 8.16MB / 7.45GB

Search | Only show running containers

	Name	Container ID	Image	Port(s)	Actions
	elastic_yonath	9893804698cc	mynodeapp	8080:8080	
	bold_pascal	b64d18a81900	mynodeapp	8081:8080	

Showing 2 items

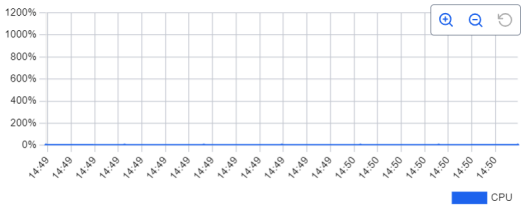
RAM 1.40 GB CPU 0.00% Disk: 2.84 GB used (limit 1006.85 GB) v4.40.0

Resource usage

Container CPU usage

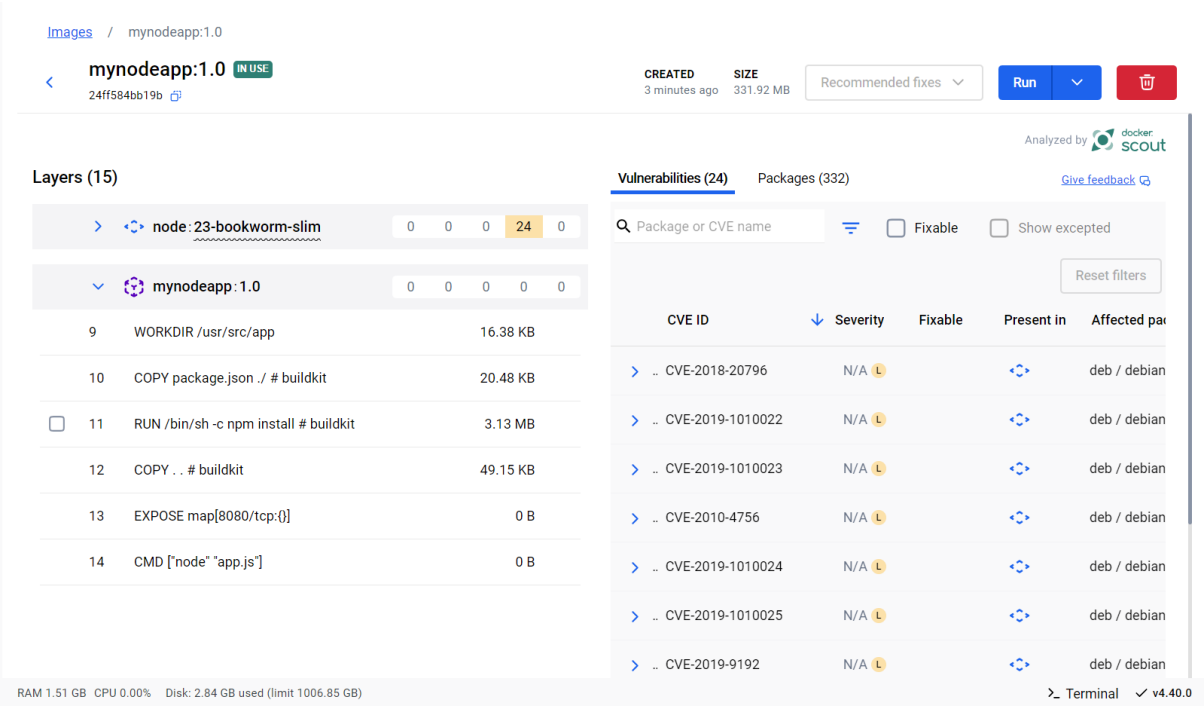
0.00% / 1200%

12 CPUs available



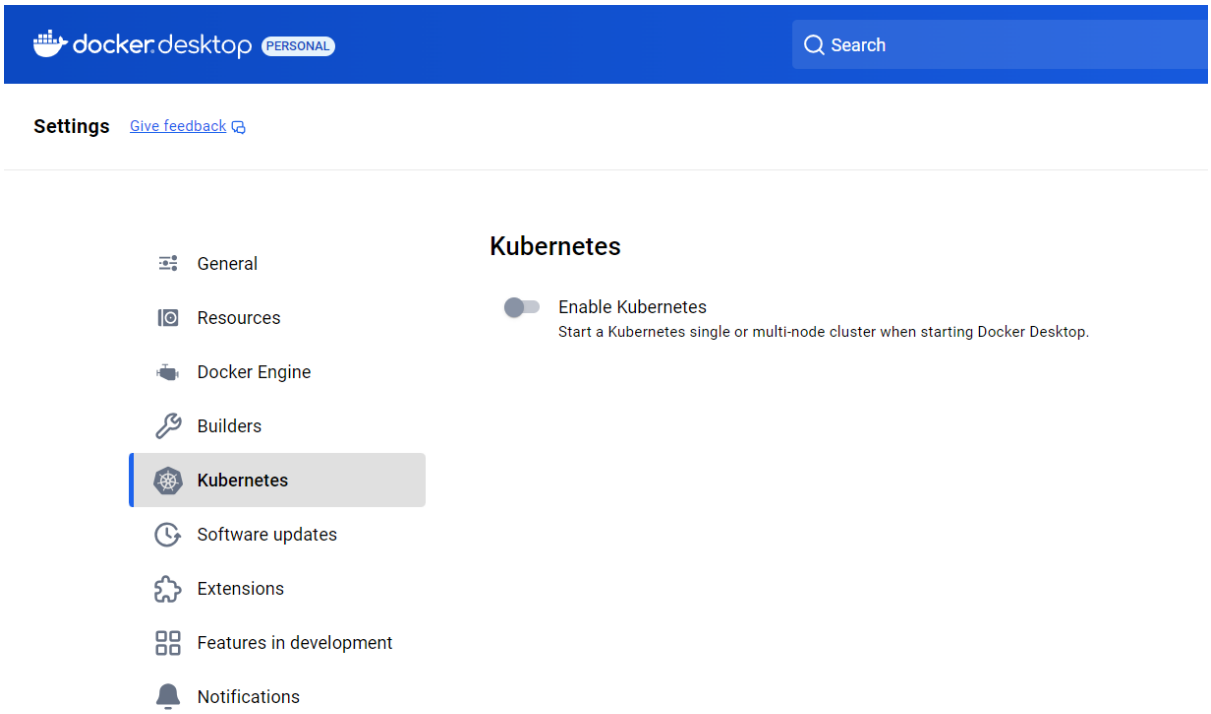
Container memory usage

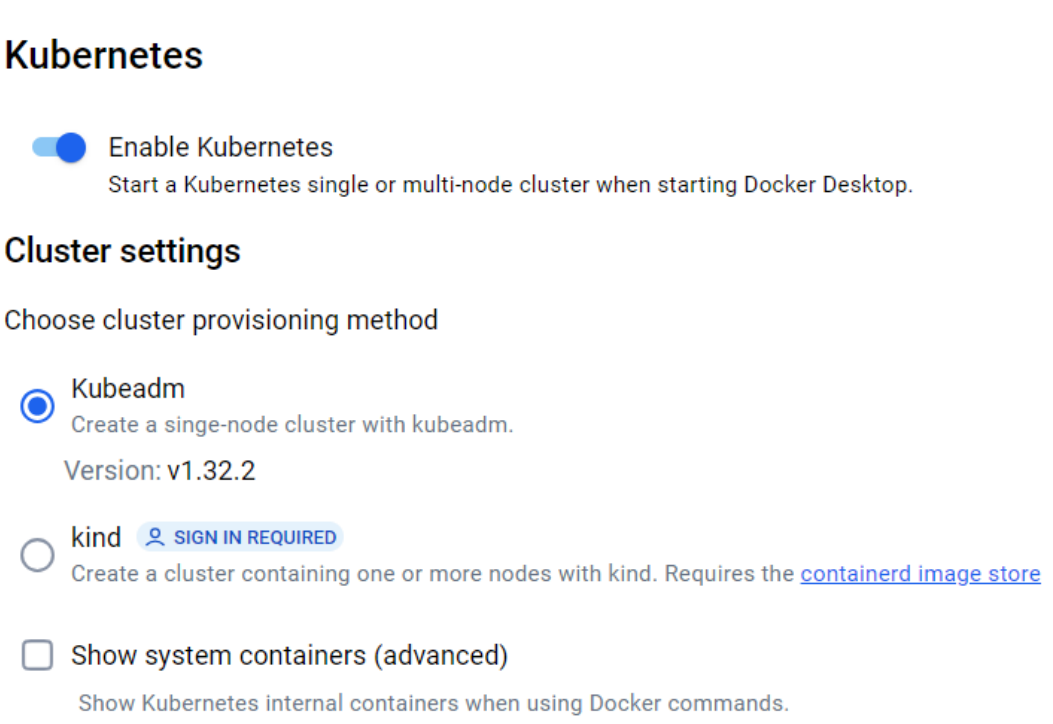
8.16MB / 7.45GB

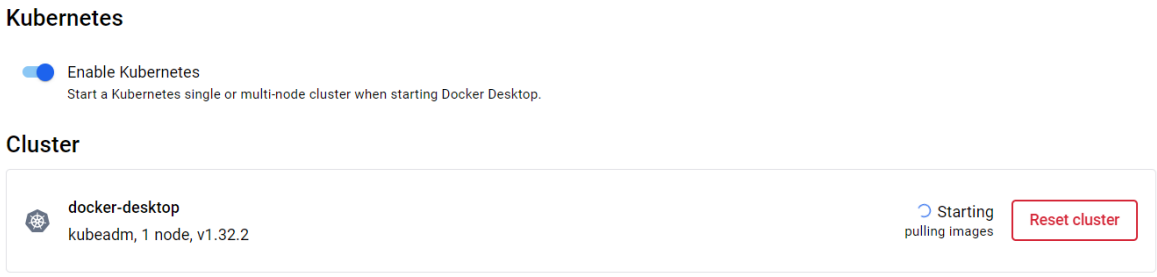


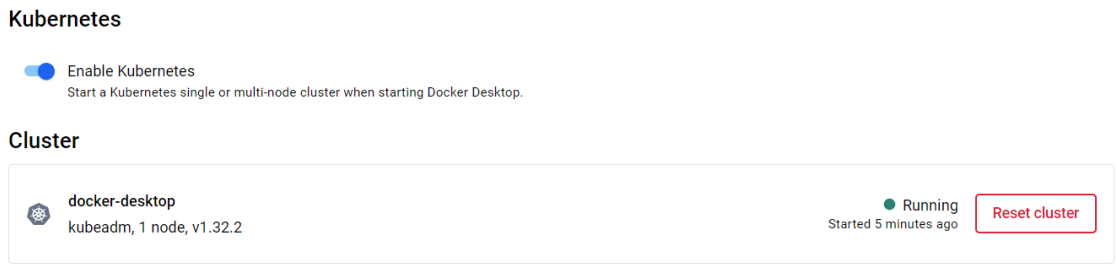
Step 1: Install Kubernetes (if not already installed)

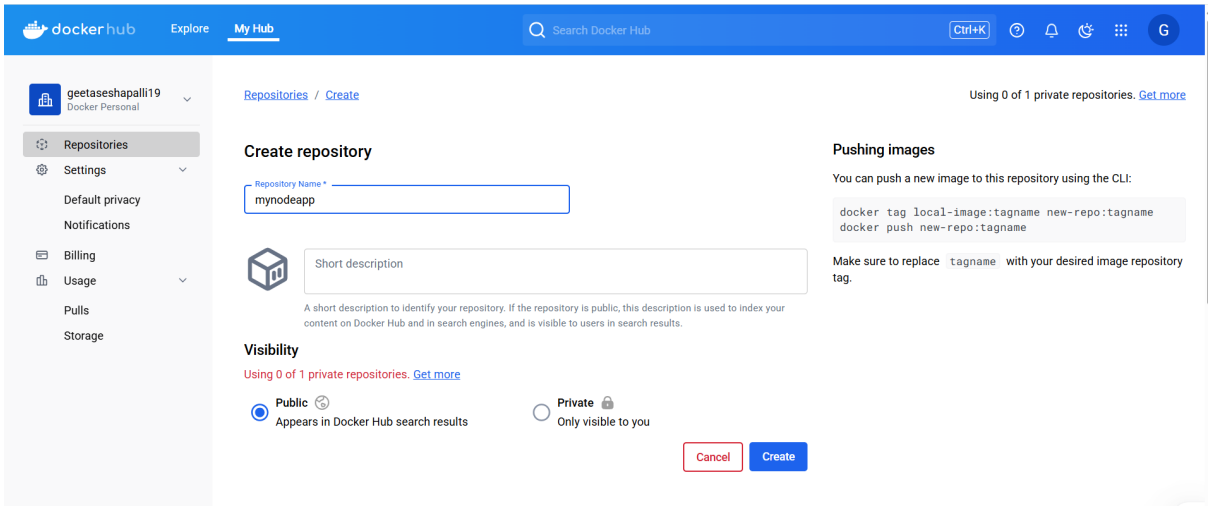
If you don't have Kubernetes installed on your system yet, you can install **Minikube** (a local Kubernetes cluster) or use **Docker Desktop** which provides an in-built Kubernetes cluster.

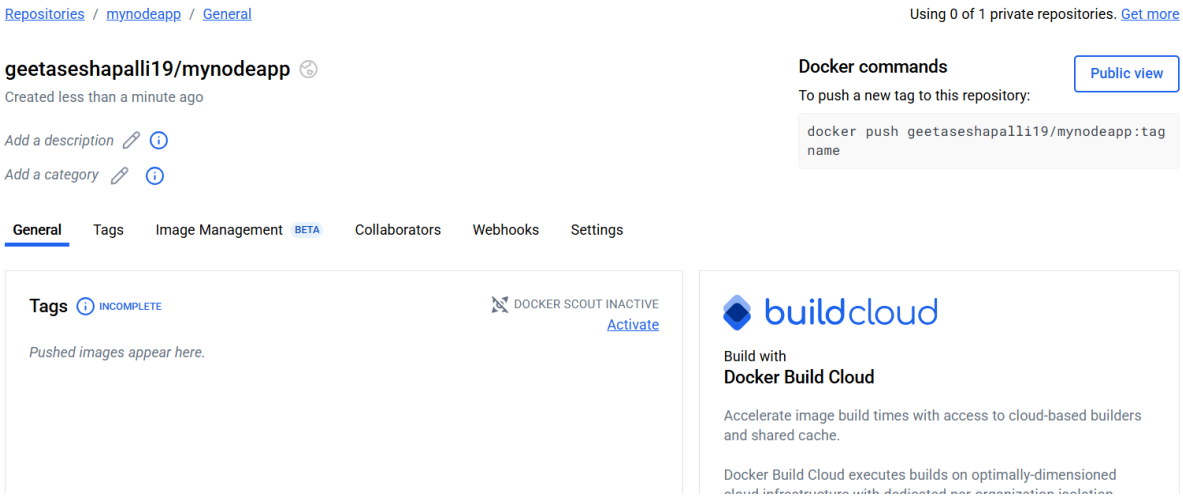



Kubernetes
Enable Kubernetes
Start a Kubernetes single or multi-node cluster when starting Docker Desktop.
Cluster settings
Choose cluster provisioning method
Kubeadm
Create a singe-node cluster with kubeadm.
Version: v1.32.2
kind
SIGN IN REQUIRED
Create a cluster containing one or more nodes with kind. Requires the containerd image store
Show system containers (advanced)
Show Kubernetes internal containers when using Docker commands.


Kubernetes
Enable Kubernetes
Start a Kubernetes single or multi-node cluster when starting Docker Desktop.
Cluster
docker-desktop
kubeadm, 1 node, v1.32.2
Starting
pulling images
Reset cluster


Kubernetes
Enable Kubernetes
Start a Kubernetes single or multi-node cluster when starting Docker Desktop.
Cluster
docker-desktop
kubeadm, 1 node, v1.32.2
Running
Started 5 minutes ago
Reset cluster






```
PS C:\Users\geeta\OneDrive\Desktop\exp4\my-docker-app> docker tag mynodeapp:1.0 geetaseshapalli19/mynodeapp:1.0
PS C:\Users\geeta\OneDrive\Desktop\exp4\my-docker-app> docker push geetaseshapalli19/mynodeapp:1.0
The push refers to repository [docker.io/geetaseshapalli19/mynodeapp]
58e674205d2a: Pushed
af8df1bc3450: Pushed
64c9cb827123: Pushed
1667adbcc2a4: Pushed
3814b81dd163: Pushed
8a628cdd7ccc: Mounted from library/redis
809b04e40f4e: Pushed
c7a0e0208e7b: Pushed
3ba11eb6e58f: Pushed
50d5199701e6: Pushed
1.0: digest: sha256:24ff584bb19b23c4726a075c4297b455471d450342d238c605cb9390c5058e3d size: 856
PS C:\Users\geeta\OneDrive\Desktop\exp4\my-docker-app>
```

Repositories / mynodeapp / General

geetaseshapalli19/mynodeapp

Last pushed less than a minute ago • Repository size: 76.9 MB

Add a description

Add a category

General Tags Image Management BETA Collaborators Webhooks Settings

Tags

DOCKER SCOUT INACTIVE
Activate

This repository contains 0 tag(s).

Tag	OS	Type	Pulled	Pushed
1.0		Image	less than 1 day	less than a minute

See all

To verify if you have Kubernetes installed and running, run:

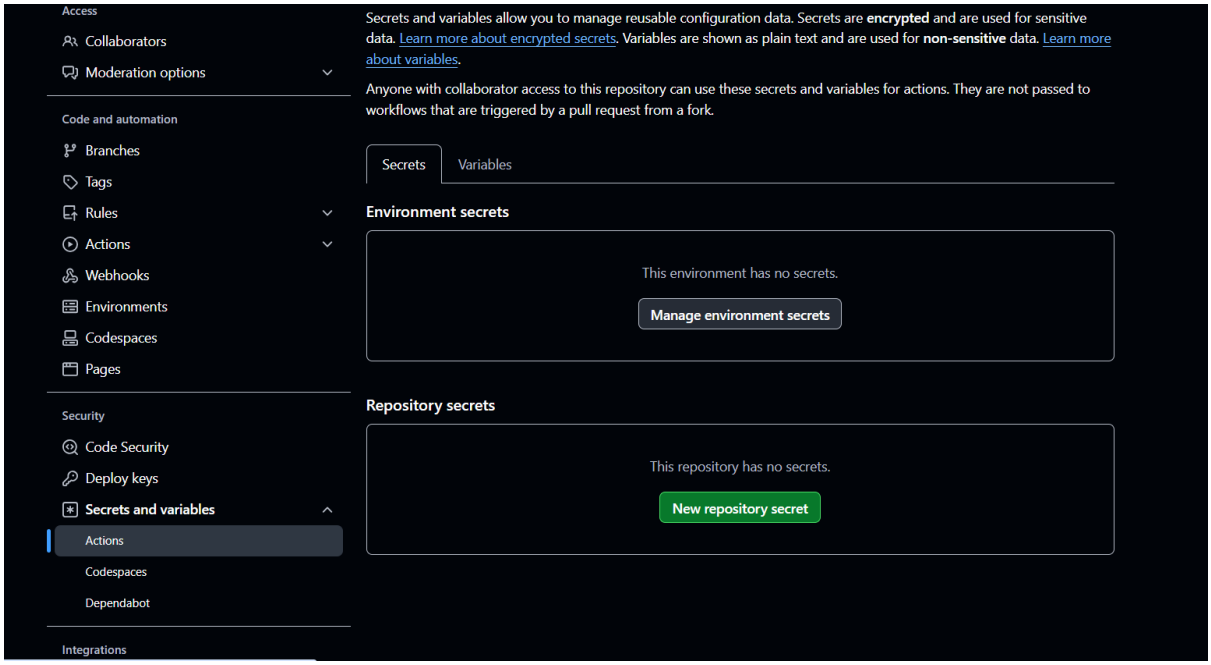
```
C:\Users\geeta>kubectl version
Client Version: v1.32.2
Kustomize Version: v5.5.0
Unable to connect to the server: EOF
```

```
PS C:\Users\geeta\OneDrive\Desktop\exp4\my-docker-app> kubectl apply -f deployment.yaml
deployment.apps/mynodeapp-deployment created
PS C:\Users\geeta\OneDrive\Desktop\exp4\my-docker-app> kubectl apply -f service.yaml
service/mynodeapp-service unchanged
```

```
PS C:\Users\geeta\OneDrive\Desktop\exp4\my-docker-app> kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
mynodeapp-deployment-74c998d876-84c9m   1/1     Running   0          69s
mynodeapp-deployment-74c998d876-jlwkq   1/1     Running   0          69s
mynodeapp-deployment-74c998d876-lj7r2   1/1     Running   0          69s
```

```
PS C:\Users\geeta\OneDrive\Desktop\exp4\my-docker-app> kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
mynodeapp-deployment  3/3     3            3           102s
PS C:\Users\geeta\OneDrive\Desktop\exp4\my-docker-app>
```

```
PS C:\Users\geeta\OneDrive\Desktop\exp4\my-docker-app> kubectl get services
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes          ClusterIP      10.96.0.1       <none>        443/TCP          17m
mynodeapp-service   LoadBalancer   10.99.216.180   localhost     8081:32370/TCP   3m
PS C:\Users\geeta\OneDrive\Desktop\exp4\my-docker-app>
```




```
PS C:\Users\geeta\OneDrive\Desktop\exp4\my-docker-app> kubectl scale deployment mynodeapp-deployment --replicas=3
deployment.apps/mynodeapp-deployment scaled
PS C:\Users\geeta\OneDrive\Desktop\exp4\my-docker-app> kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
mynodeapp-deployment  3/3     3            3           30m
PS C:\Users\geeta\OneDrive\Desktop\exp4\my-docker-app> kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
mynodeapp-deployment-74c998d876-84c9m 1/1     Running   0          30m
mynodeapp-deployment-74c998d876-jlwkq 1/1     Running   0          30m
mynodeapp-deployment-74c998d876-lj7r2 1/1     Running   0          30m
PS C:\Users\geeta\OneDrive\Desktop\exp4\my-docker-app>
```

k8s_mynodeapp_mynodea	c90a6c85eb07	24ff584bb19b	0%	33 minutes ago		
k8s_mynodeapp_mynodea	b377872571bb	24ff584bb19b	0%	33 minutes ago		
k8s_mynodeapp_mynodea	664bfa018fb2	24ff584bb19b	0%	33 minutes ago		

Creating Kubernetes Deployment

To create a Kubernetes **Deployment** to manage the WordSearch app container on the Kubernetes cluster. The deployment ensures that the application is running as expected, with specified replicas.

Create a file named wordsearch-deployment.yaml:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordsearch-deployment
spec:
  replicas: 2  # You can increase this for more replicas
  selector:
    matchLabels:
      app: wordsearch
  template:
    metadata:
      labels:
        app: wordsearch
    spec:
      containers:
      - name: wordsearch
        image: geetaseshapalli19/wordsearch-app:latest
        ports:
        - containerPort: 80
        resources:
          requests:
            memory: "64Mi"  # Minimum memory requested
            cpu: "250m"      # Minimum CPU requested
          limits:
            memory: "128Mi"  # Maximum memory allowed
            cpu: "500m"      # Maximum CPU allowed
```

- **replicas**: Number of pod instances that should run.
- **matchLabels**: A label selector to find the pods to manage.
- **containers**: Defines the Docker container to use (in this case, your WordSearch app container).
- **containerPort**: Exposes port 80 in the container (since Nginx listens on this port).

Creating Kubernetes Service

- To create a Kubernetes **Service** to expose your application. The service will allow external traffic to access the deployed app.
- Create a file named `wordsearch-service.yaml`:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: wordsearch-service
spec:
  selector:
    app: wordsearch
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer  # or NodePort if you are running Kubernetes locally
```

Deploying to Kubernetes

- Now that you have your deployment and service YAML files, it's time to deploy them to Kubernetes.
- Run the following commands to apply the Kubernetes configurations:

Deploy the WordSearch app

kubectl apply -f wordsearch-deployment.yaml

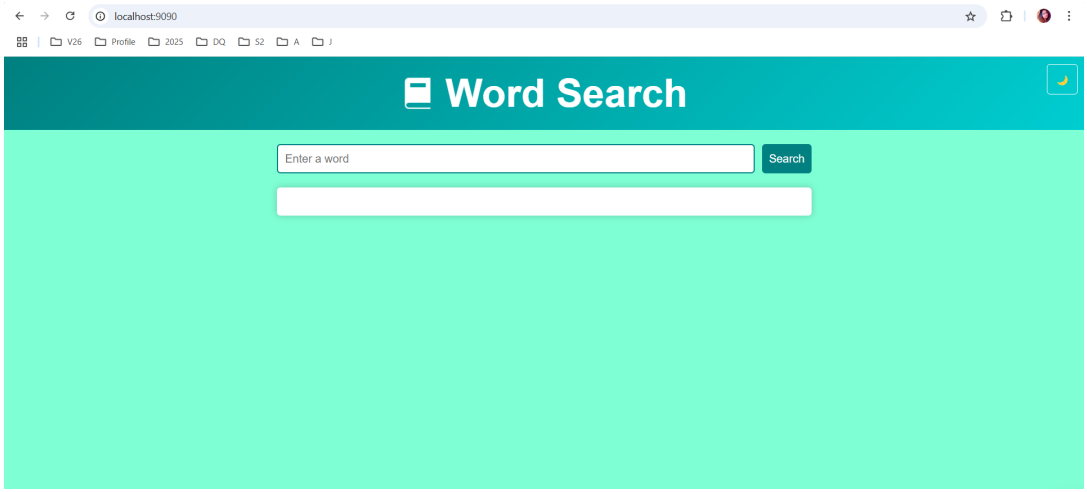
Expose the app via a service

kubectl apply -f wordsearch-service.yaml

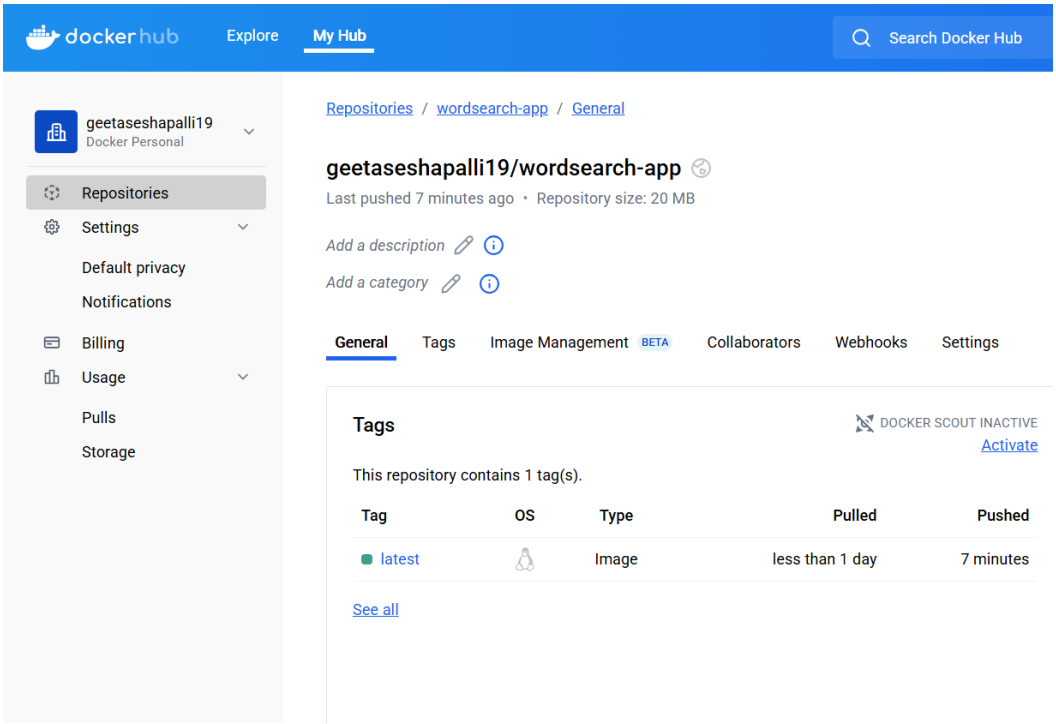
Check pods:

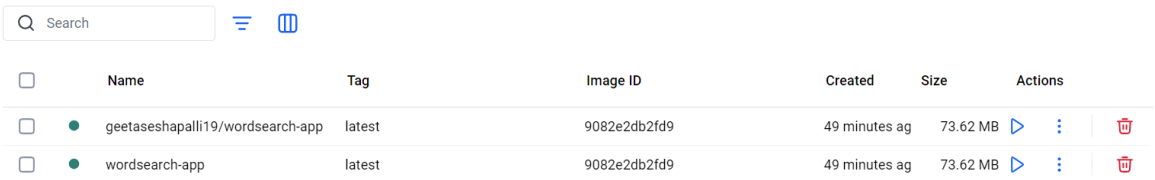
```
deployment.apps/wordsearch-deployment configured
PS C:\Users\geeta\OneDrive\Desktop\WordSearch> kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
wordsearch-deployment-595ffbff7b-4mmwv   1/1     Running   0          8s
wordsearch-deployment-595ffbff7b-8nhlp   1/1     Running   0          4s
PS C:\Users\geeta\OneDrive\Desktop\WordSearch> kubectl get service wordsearch-service
NAME                 TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
wordsearch-service   LoadBalancer   10.107.113.125   <pending>     80:30862/TCP   6m28s
PS C:\Users\geeta\OneDrive\Desktop\WordSearch> kubectl port-forward service/wordsearch-service 9090:80
Forwarding from 127.0.0.1:9090 -> 80
Forwarding from [::1]:9090 -> 80
Handling connection for 9090
Handling connection for 9090
```

Access the app in your browser:



Description: http://localhost:9090





Description: Kubernetes

```
PS C:\Users\geeta\OneDrive\Desktop\WordSearch> kubectl describe deployment wordsearch-deployment
Name:                   wordsearch-deployment
Namespace:              default
CreationTimestamp:      Sun, 27 Apr 2025 18:26:45 +0530
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 3
Selector:               app=wordsearch
Replicas:               2 desired | 2 updated | 2 total | 2 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=wordsearch
  Containers:
   wordsearch:
    Image:      geetaseshapalli19/wordsearch-app:latest
    Port:       80/TCP
    Host Port:  0/TCP
    Limits:
      cpu:     500m
      memory:  128Mi
    Requests:
      cpu:         250m
      memory:      64Mi
    Environment:   <none>
    Mounts:        <none>
  Volumes:         <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
```

Description: Kubernetes Description of file

```
      memory:      64Mi
    Environment:   <none>
    Mounts:        <none>
  Volumes:         <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  wordsearch-deployment-6f99dcc59 (0/0 replicas created), wordsearch-deployment-595ffbff7b (0/0 replicas created)
NewReplicaSet:   wordsearch-deployment-6c4649968f (2/2 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  29m   deployment-controller  Scaled up replica set wordsearch-deployment-6f99dcc59 from 0 to 2
  Normal  ScalingReplicaSet  23m   deployment-controller  Scaled up replica set wordsearch-deployment-595ffbff7b from 0 to 1
  Normal  ScalingReplicaSet  23m   deployment-controller  Scaled down replica set wordsearch-deployment-6f99dcc59 from 2 to 1
  Normal  ScalingReplicaSet  23m   deployment-controller  Scaled up replica set wordsearch-deployment-595ffbff7b from 1 to 2
  Normal  ScalingReplicaSet  23m   deployment-controller  Scaled down replica set wordsearch-deployment-6f99dcc59 from 1 to 0
  Normal  ScalingReplicaSet  108s  deployment-controller  Scaled up replica set wordsearch-deployment-6c4649968f from 0 to 1
  Normal  ScalingReplicaSet  105s  deployment-controller  Scaled down replica set wordsearch-deployment-595ffbff7b from 2 to 1
  Normal  ScalingReplicaSet  105s  deployment-controller  Scaled up replica set wordsearch-deployment-6c4649968f from 1 to 2
  Normal  ScalingReplicaSet  102s  deployment-controller  Scaled down replica set wordsearch-deployment-595ffbff7b from 1 to 0
PS C:\Users\geeta\OneDrive\Desktop\WordSearch>
```

Description: Kubernetes Description of file

```
PS C:\Users\geeta\OneDrive\Desktop\WordSearch> git add .
PS C:\Users\geeta\OneDrive\Desktop\WordSearch> git commit -m "Kubernetes Changes"
[main df2ed15] Kubernetes Changes
 2 files changed, 31 insertions(+)
 create mode 100644 wordsearch-deployment.yaml
 create mode 100644 wordsearch-service.yaml
PS C:\Users\geeta\OneDrive\Desktop\WordSearch> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 705 bytes | 352.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/geeta-seshapalli/Word-Search.git
   8e1d1fb..df2ed15  main -> main
PS C:\Users\geeta\OneDrive\Desktop\WordSearch>
```

Description: Kubernetes Changes in Git

```
> C:\Program Files\Git\bin\git.exe checkout -b main df2ed15d48c567c1be0b97122b4e7b7a74c05e76 # timeout=10
Commit message: "Kubernetes Changes"
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Install Dependencies)
[Pipeline] echo
No dependencies to install.
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Run Tests)
[Pipeline] echo
No tests configured, skipping.
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
No build step configured, skipping.
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy to Staging)
[Pipeline] echo
Deployment to staging server complete!
[Pipeline] }
[Pipeline] // stage
```

Description: Kubernetes in Jenkins Status

```
[Pipeline] echo
Deployment to staging server complete!
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Notify)
[Pipeline] echo
Deployment completed!
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
Build and Deployment successful!
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Description: Kubernetes in Jenkins Status

REPOSITORY LINK: https://github.com/geeta-seshapalli/Word-Search.git

Conclusion:

We have Kubernetes integrated with Docker and have automated the deployment process for your containerized application. We can scale, update, and manage your app in Kubernetes with ease.