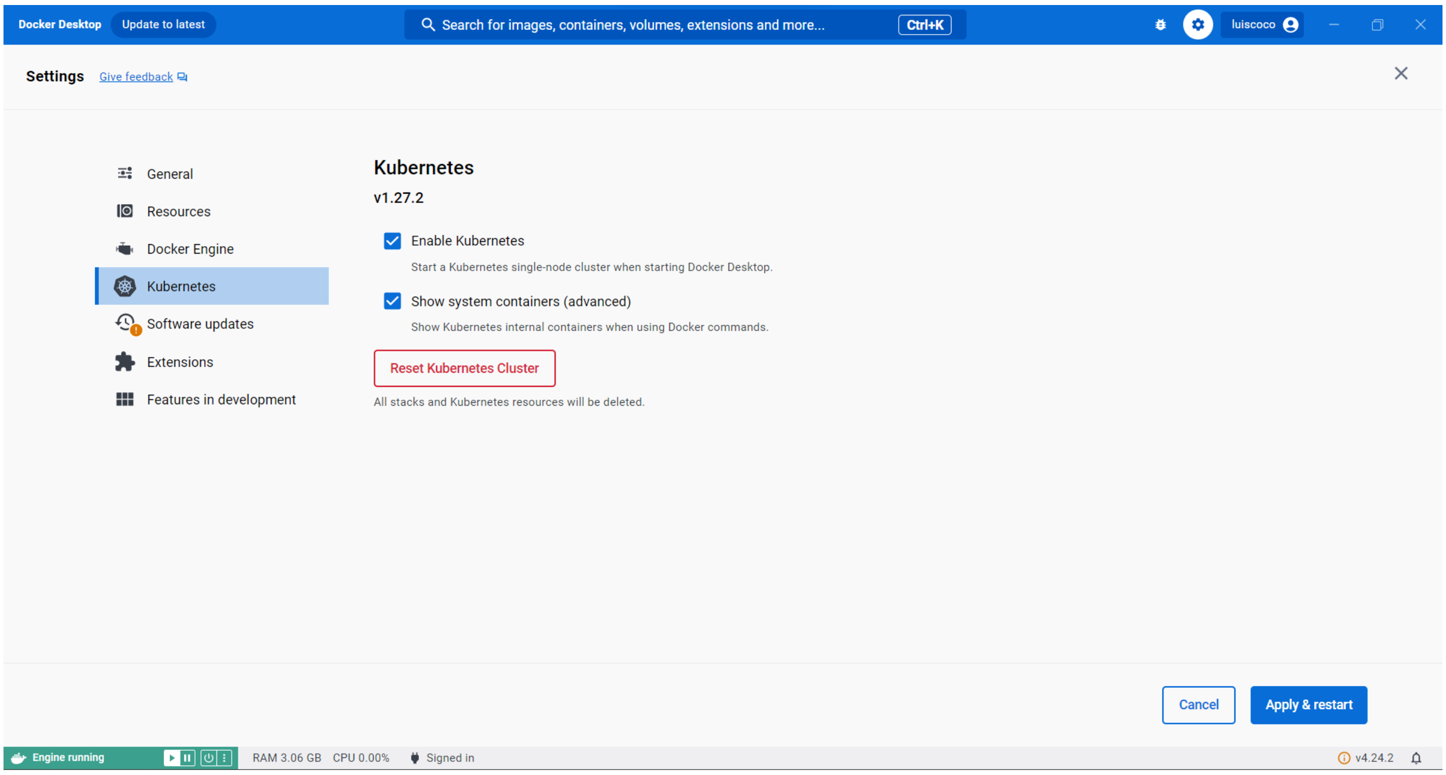
How to deploy SpringBoot WebAPI to AWS EKS

1. Prerequisites

1.1. Install and run Docker Desktop

https://docs.docker.com/desktop/install/windows-install/

Run Docker Desktop and enable Kubernetes



Kubernetes

v1.27.2

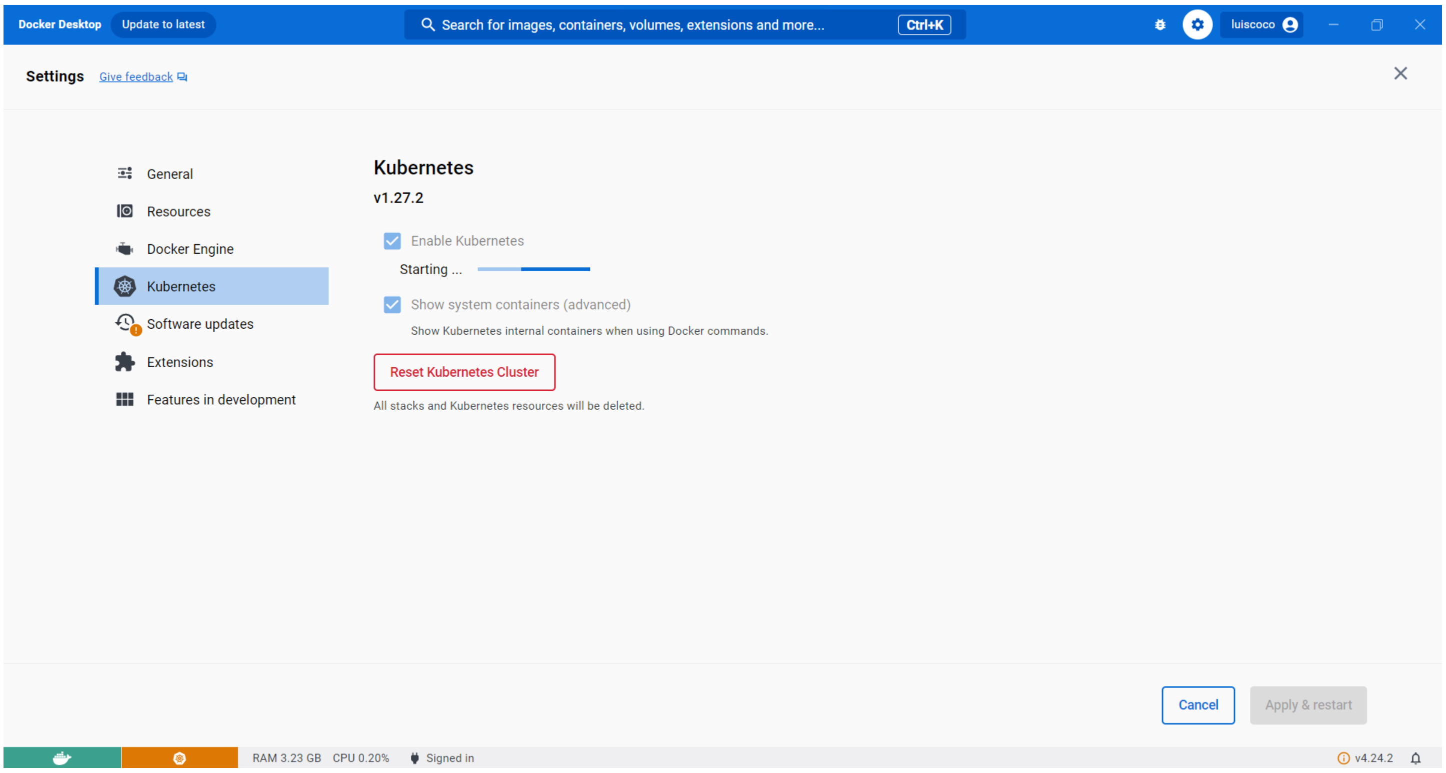
☑ Enable Kubernetes

Start a Kubernetes single-node cluster when starting Docker Desktop.

☑ Show system containers (advanced)

Show Kubernetes internal containers when using Docker commands.

All stacks and Kubernetes resources will be deleted.



Docker Desktop Update to latest Search for images, containers, volumes, extensions and more... Ctrl+K luiscoco

Settings Give feedback

General
Resources
Docker Engine
Kubernetes
Software updates
Extensions
Features in development

Kubernetes

v1.27.2

Enable Kubernetes

Starting ...

Show system containers (advanced)

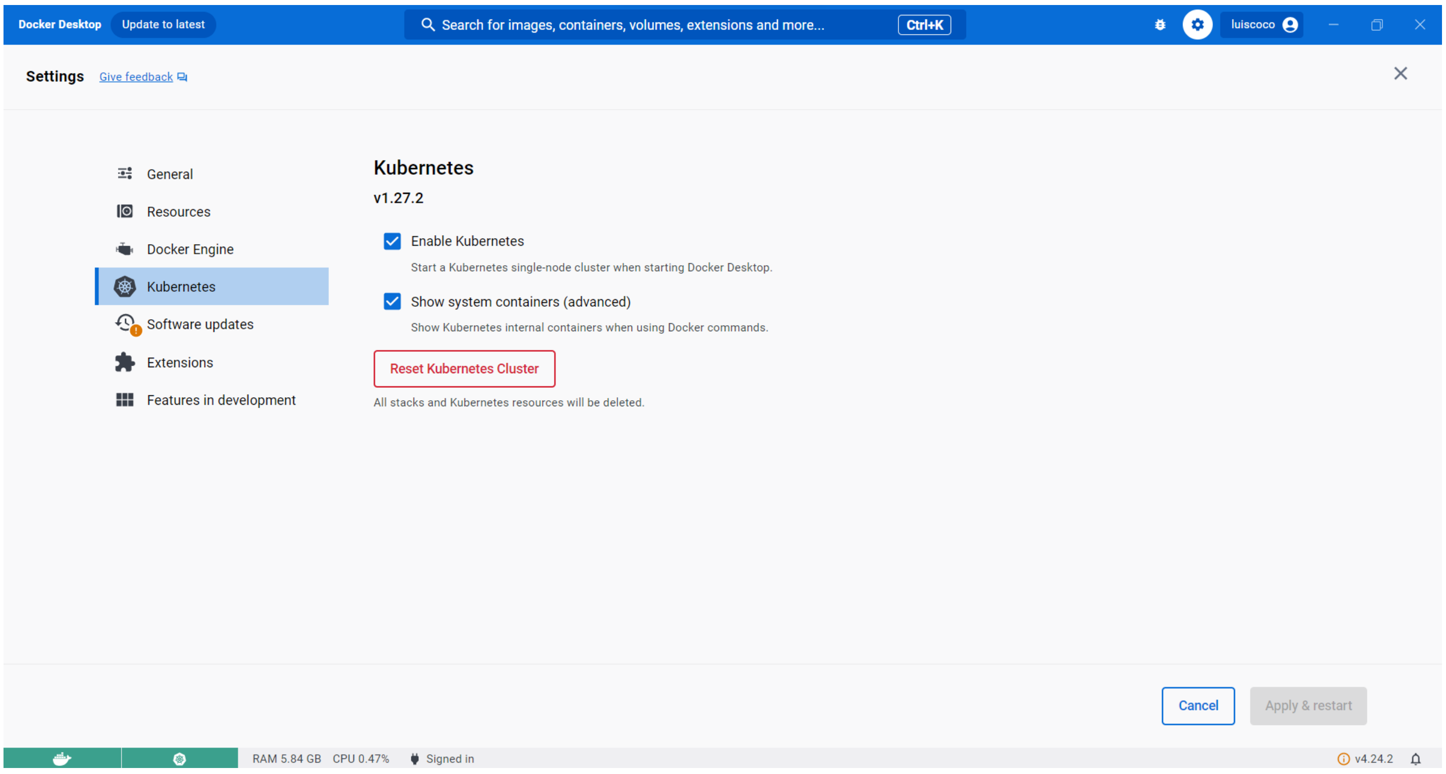
Show Kubernetes internal containers when using Docker commands.

Reset Kubernetes Cluster

All stacks and Kubernetes resources will be deleted.

Cancel Apply & restart

RAM 3.23 GB CPU 0.20% Signed in v4.24.2



Docker Desktop Update to latest Search for images, containers, volumes, extensions and more... Ctrl+K luiscoco

Settings Give feedback

General
Resources
Docker Engine
Kubernetes
Software updates
Extensions
Features in development

Kubernetes

v1.27.2

Enable Kubernetes

Start a Kubernetes single-node cluster when starting Docker Desktop.

Show system containers (advanced)

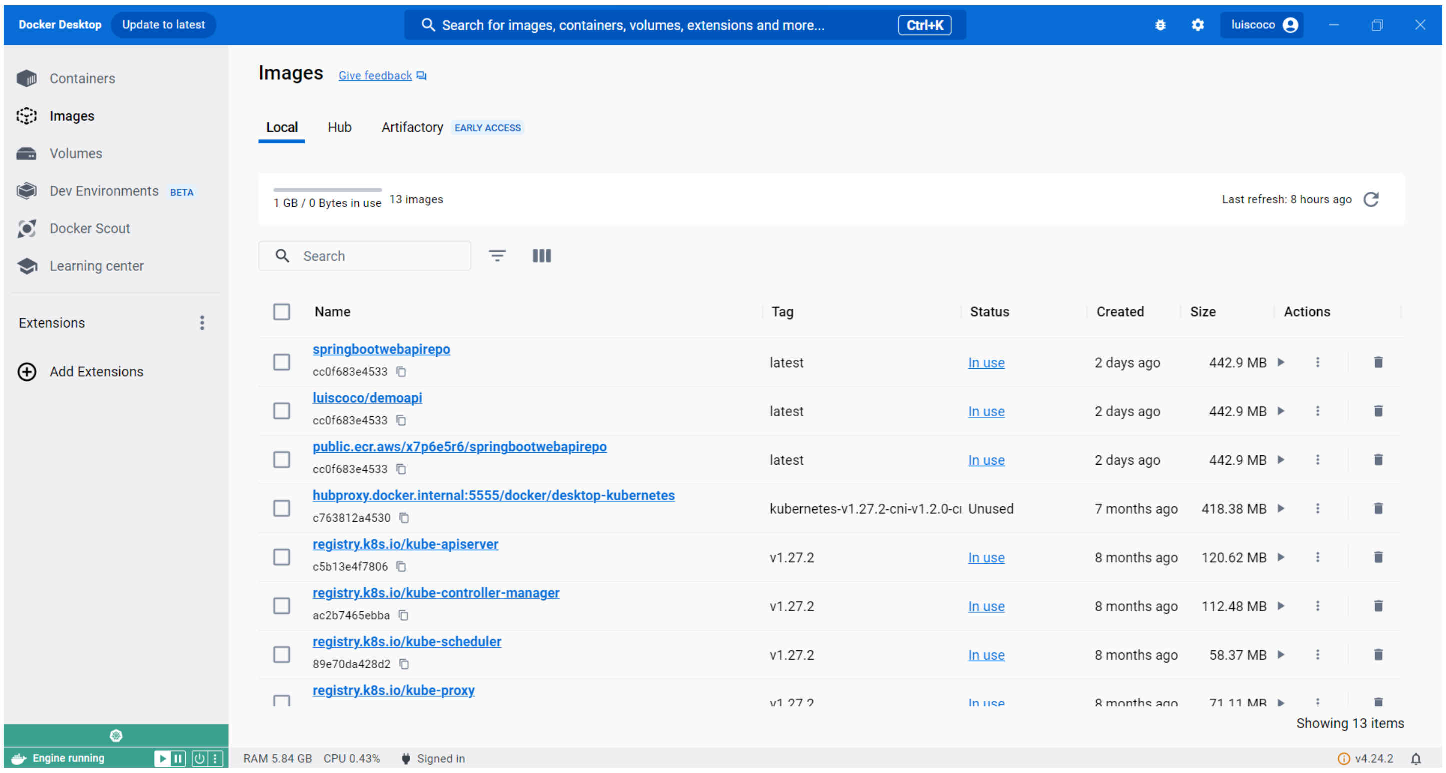
Show Kubernetes internal containers when using Docker commands.

Reset Kubernetes Cluster

All stacks and Kubernetes resources will be deleted.

Cancel Apply & restart

RAM 5.84 GB CPU 0.47% Signed in v4.24.2



Images

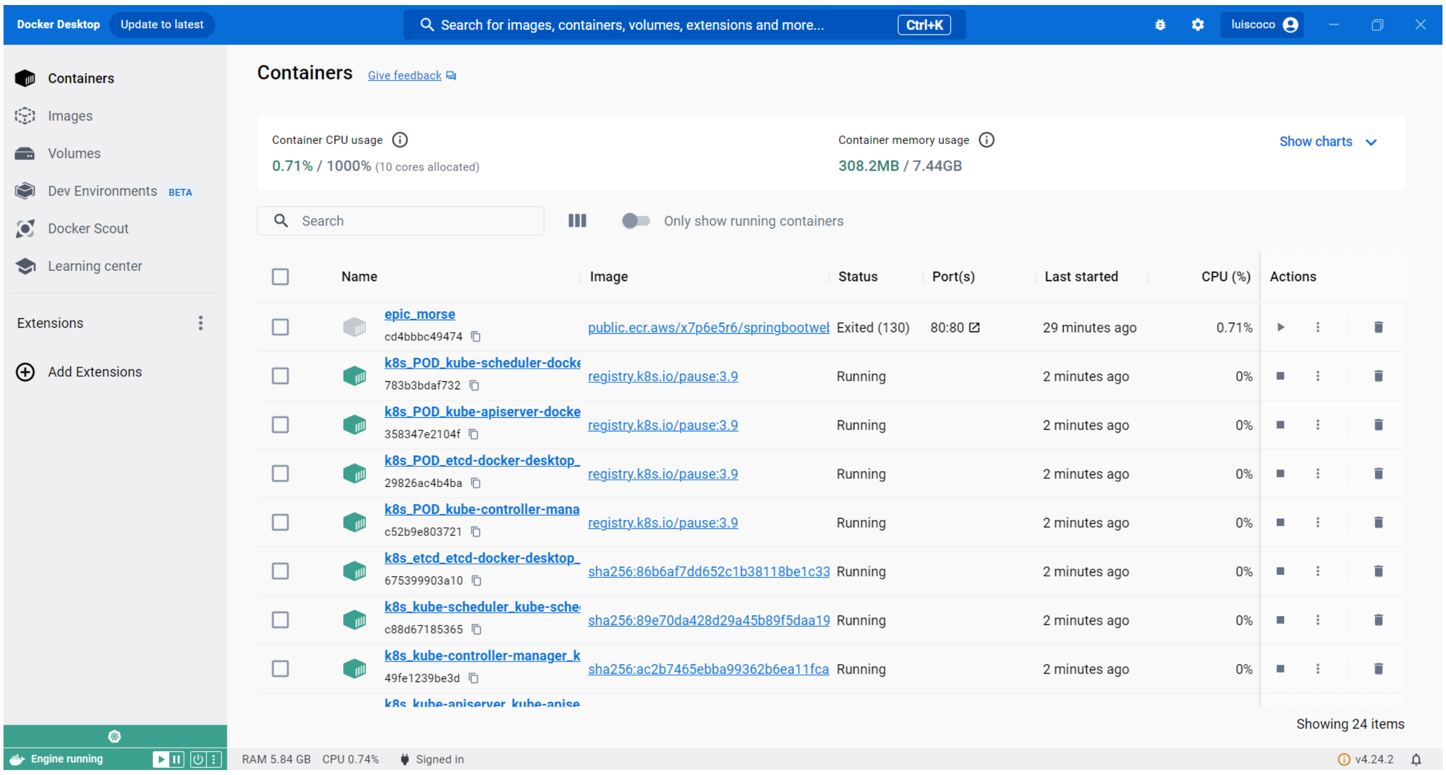
1 GB / 0 Bytes in use 13 images

Last refresh: 8 hours ago

Name	Tag	Status	Created	Size	Actions
springbootwebapirepo cc0f683e4533	latest	In use	2 days ago	442.9 MB	
luiscoco/demoapi cc0f683e4533	latest	In use	2 days ago	442.9 MB	
public.ecr.aws/x7p6e5r6/springbootwebapirepo cc0f683e4533	latest	In use	2 days ago	442.9 MB	
hubproxy.docker.internal:5555/docker/desktop-kubernetes c763812a4530	kubernetes-v1.27.2-cni-v1.2.0-cı	Unused	7 months ago	418.38 MB	
registry.k8s.io/kube-apiserver c5b13e4f7806	v1.27.2	In use	8 months ago	120.62 MB	
registry.k8s.io/kube-controller-manager ac2b7465ebba	v1.27.2	In use	8 months ago	112.48 MB	
registry.k8s.io/kube-scheduler 89e70da428d2	v1.27.2	In use	8 months ago	58.37 MB	
registry.k8s.io/kube-proxy	v1.27.2	In use	8 months ago	71.11 MB	

Showing 13 items

RAM 5.84 GB CPU 0.43% Signed in v4.24.2



Containers

Container CPU usage
0.71% / 1000% (10 cores allocated)

Container memory usage
308.2MB / 7.44GB

Only show running containers

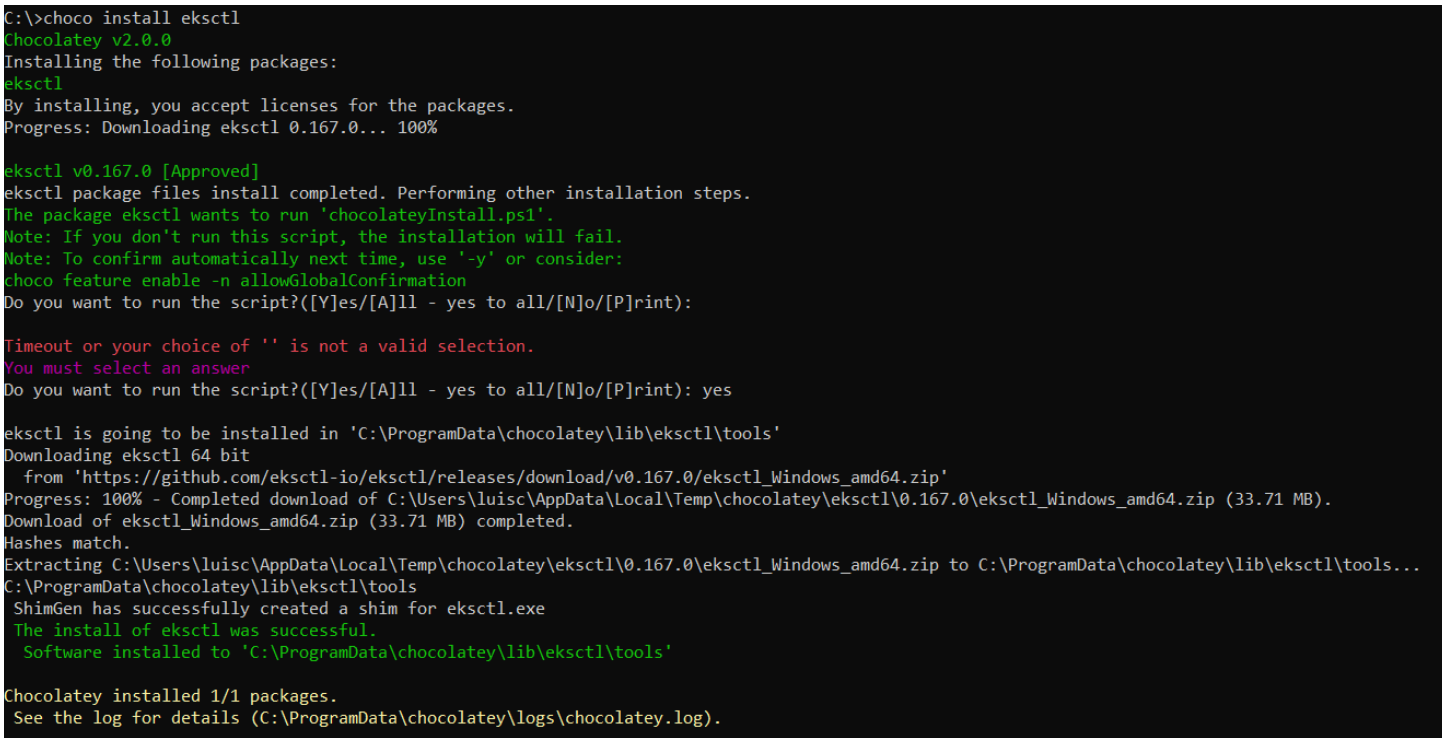
Name	Image	Status	Port(s)	Last started	CPU (%)	Actions
epic_morse cd4bbbc49474	public.ecr.aws/x7p6e5r6/springbootwel	Exited (130)	80:80	29 minutes ago	0.71%	
k8s_POD_kube-scheduler-docke 783b3bdaf732	registry.k8s.io/pause:3.9	Running		2 minutes ago	0%	
k8s_POD_kube-apiserver-docke 358347e2104f	registry.k8s.io/pause:3.9	Running		2 minutes ago	0%	
k8s_POD_etcd-docker-desktop_ 29826ac4b4ba	registry.k8s.io/pause:3.9	Running		2 minutes ago	0%	
k8s_POD_kube-controller-mana c52b9e803721	registry.k8s.io/pause:3.9	Running		2 minutes ago	0%	
k8s_etcd_etcd-docker-desktop_ 675399903a10	sha256:86b6af7dd652c1b38118be1c33	Running		2 minutes ago	0%	
k8s_kube-scheduler_kube-sche c88d67185365	sha256:89e70da428d29a45b89f5daa19	Running		2 minutes ago	0%	
k8s_kube-controller-manager_k 49fe1239be3d	sha256:ac2b7465ebba99362b6ea11fca	Running		2 minutes ago	0%	
k8s_kube-apiserver_kube-apise						

Showing 24 items

RAM 5.84 GB CPU 0.74% Signed in v4.24.2

1.2. Install eksctl:

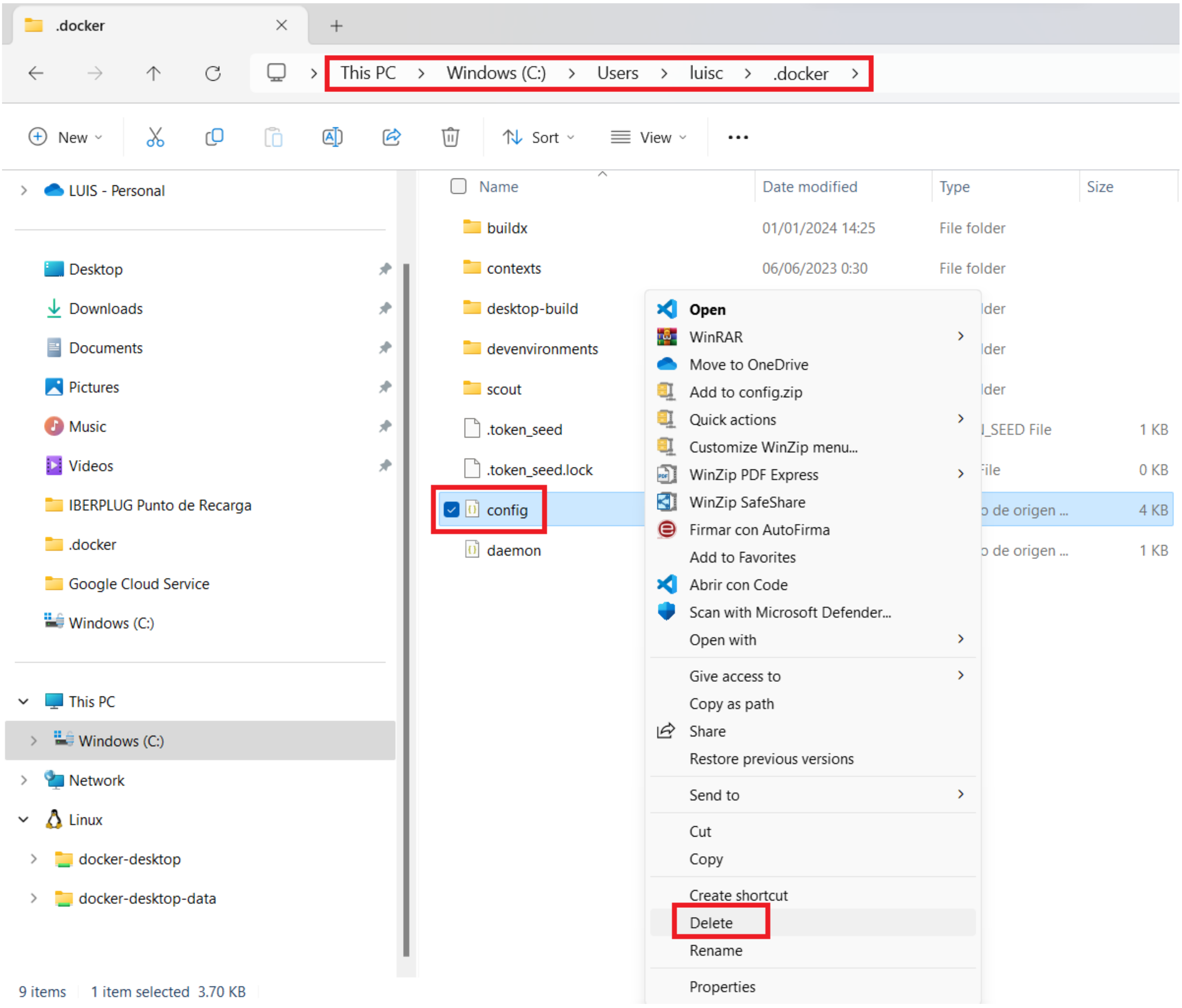
```
choco install eksctl
```



```
C:\>choco install eksctl
Chocolatey v2.0.0
Installing the following packages:
eksctl
By installing, you accept licenses for the packages.
Progress: Downloading eksctl 0.167.0... 100%

eksctl v0.167.0 [Approved]
eksctl package files install completed. Performing other installation steps.
The package eksctl wants to run 'chocolateyInstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint):

Timeout or your choice of '' is not a valid selection.
You must select an answer
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint): yes

eksctl is going to be installed in 'C:\ProgramData\chocolatey\lib\eksctl\tools'
Downloading eksctl 64 bit
  from 'https://github.com/eksctl-io/eksctl/releases/download/v0.167.0/eksctl_Windows_amd64.zip'
Progress: 100% - Completed download of C:\Users\luisc\AppData\Local\Temp\chocolatey\eksctl\0.167.0\eksctl_Windows_amd64.zip (33.71 MB).
Download of eksctl_Windows_amd64.zip (33.71 MB) completed.
Hashes match.
Extracting C:\Users\luisc\AppData\Local\Temp\chocolatey\eksctl\0.167.0\eksctl_Windows_amd64.zip to C:\ProgramData\chocolatey\lib\eksctl\tools...
C:\ProgramData\chocolatey\lib\eksctl\tools
 ShimGen has successfully created a shim for eksctl.exe
 The install of eksctl was successful.
  Software installed to 'C:\ProgramData\chocolatey\lib\eksctl\tools'

Chocolatey installed 1/1 packages.
 See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
```

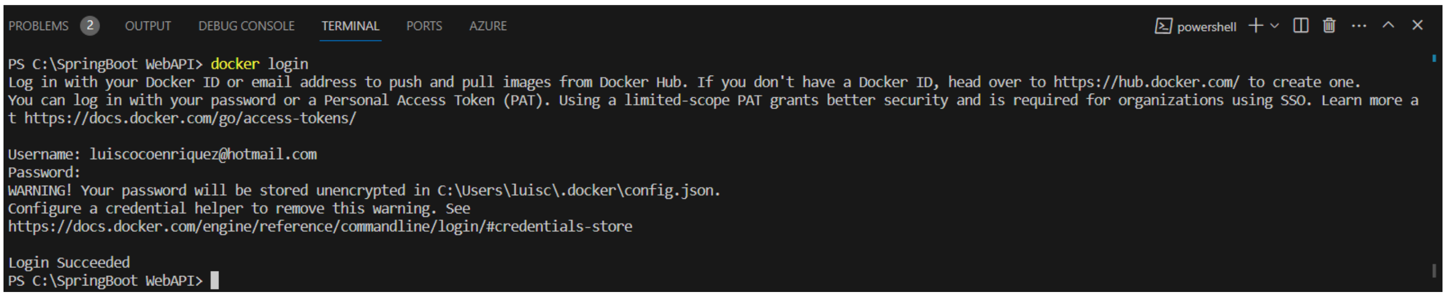
1.3. Run Docker

Delete the config.json file in this location: C:\Users\luisc.docker\config.json



Run the command:

```
docker login
```



```
PS C:\SpringBoot WebAPI> docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants better security and is required for organizations using SSO. Learn more a
t https://docs.docker.com/go/access-tokens/

Username: luiscocoenriquez@hotmail.com
Password:
WARNING! Your password will be stored unencrypted in C:\Users\luisc\.docker\config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
PS C:\SpringBoot WebAPI>
```

Delete the letter "s" in the word "credsStore"

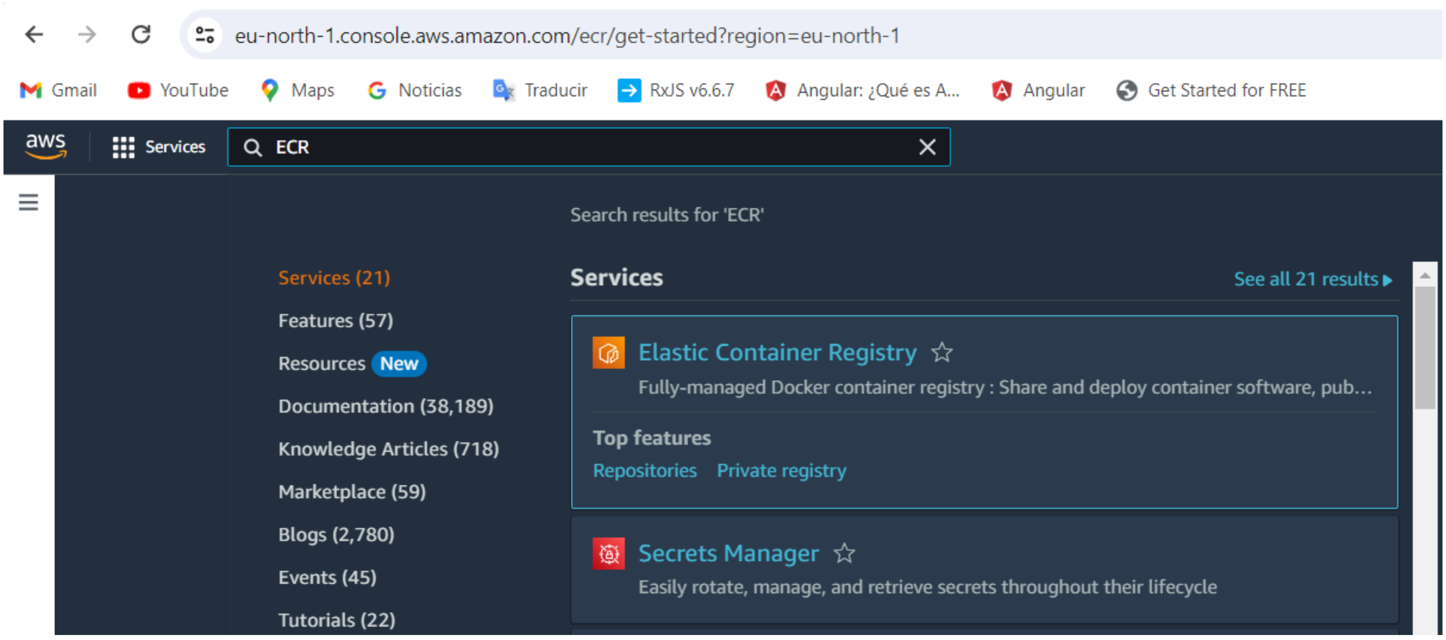
```
{
    "auths": {
        "https://index.docker.io/v1/": {}
    },
    "credsStore": "wincred"
}
```

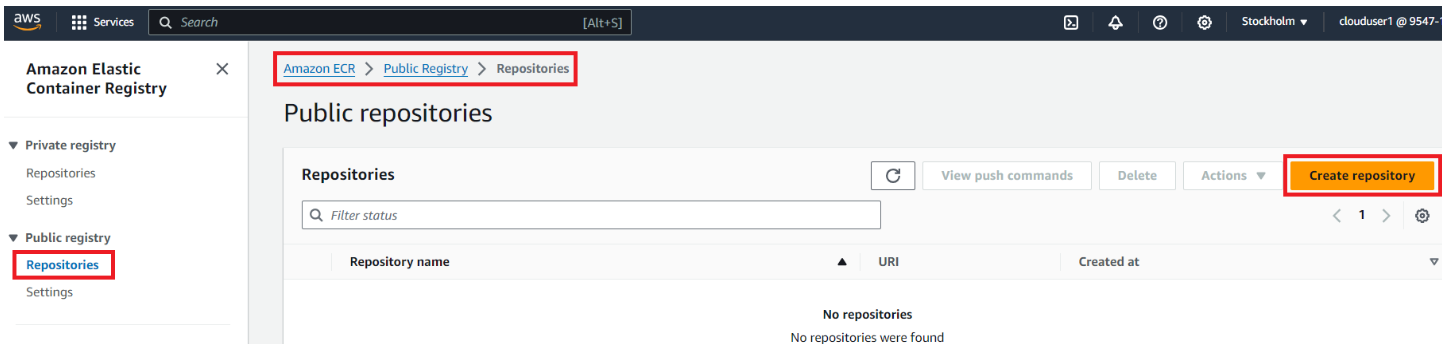
This is the new config.json file

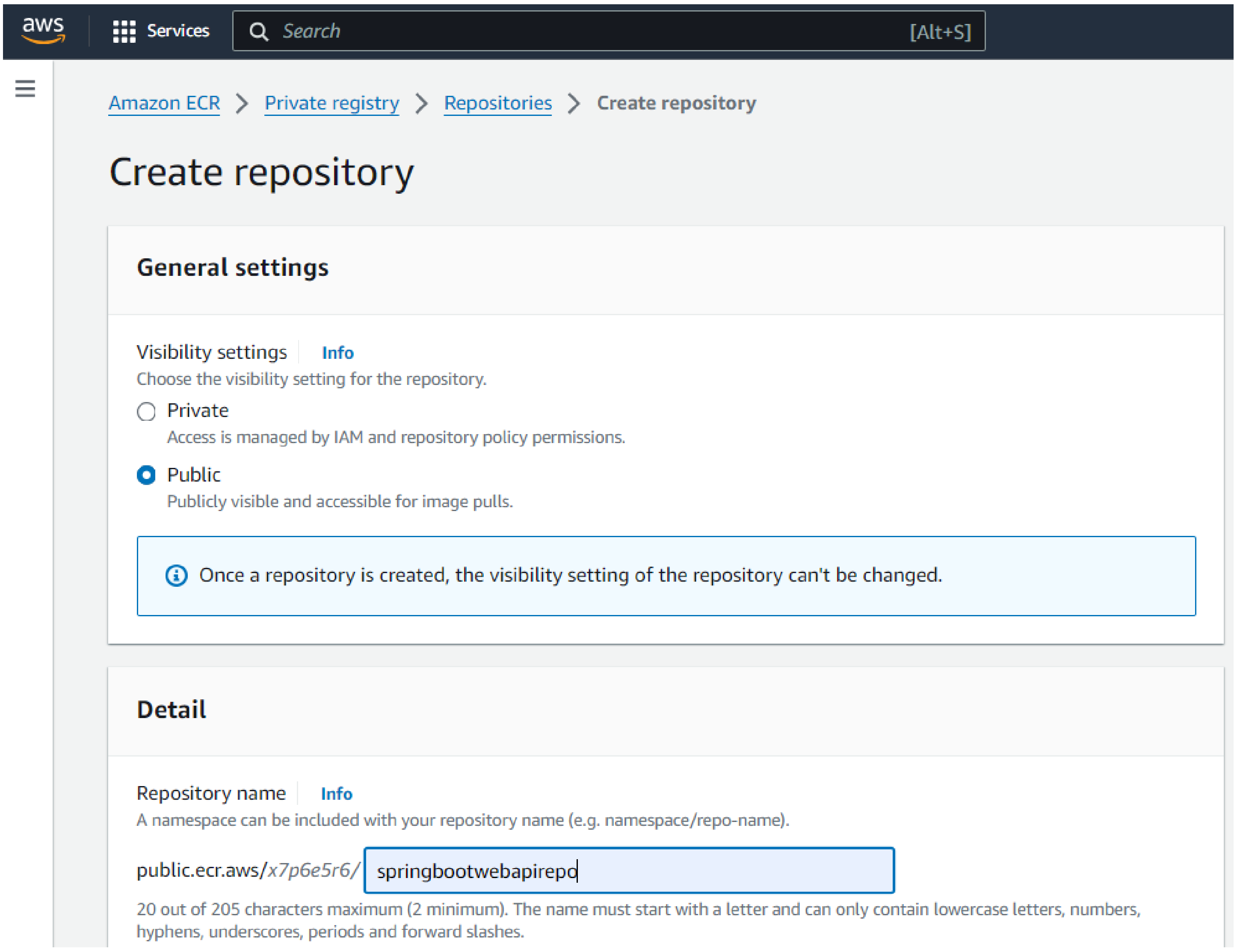
```
{
    "auths": {
        "https://index.docker.io/v1/": {}
    },
    "credStore": "wincred"
}
```

2. Create AWS ECR public repo

We navigate to AWS ECR and we create a new public repo







Amazon ECR > Private registry > Repositories > Create repository

Create repository

General settings

Visibility settings Info
Choose the visibility setting for the repository.

Private [✓ Public]

- Private — Access is managed by IAM and repository policy permissions.
- Public — Publicly visible and accessible for image pulls.

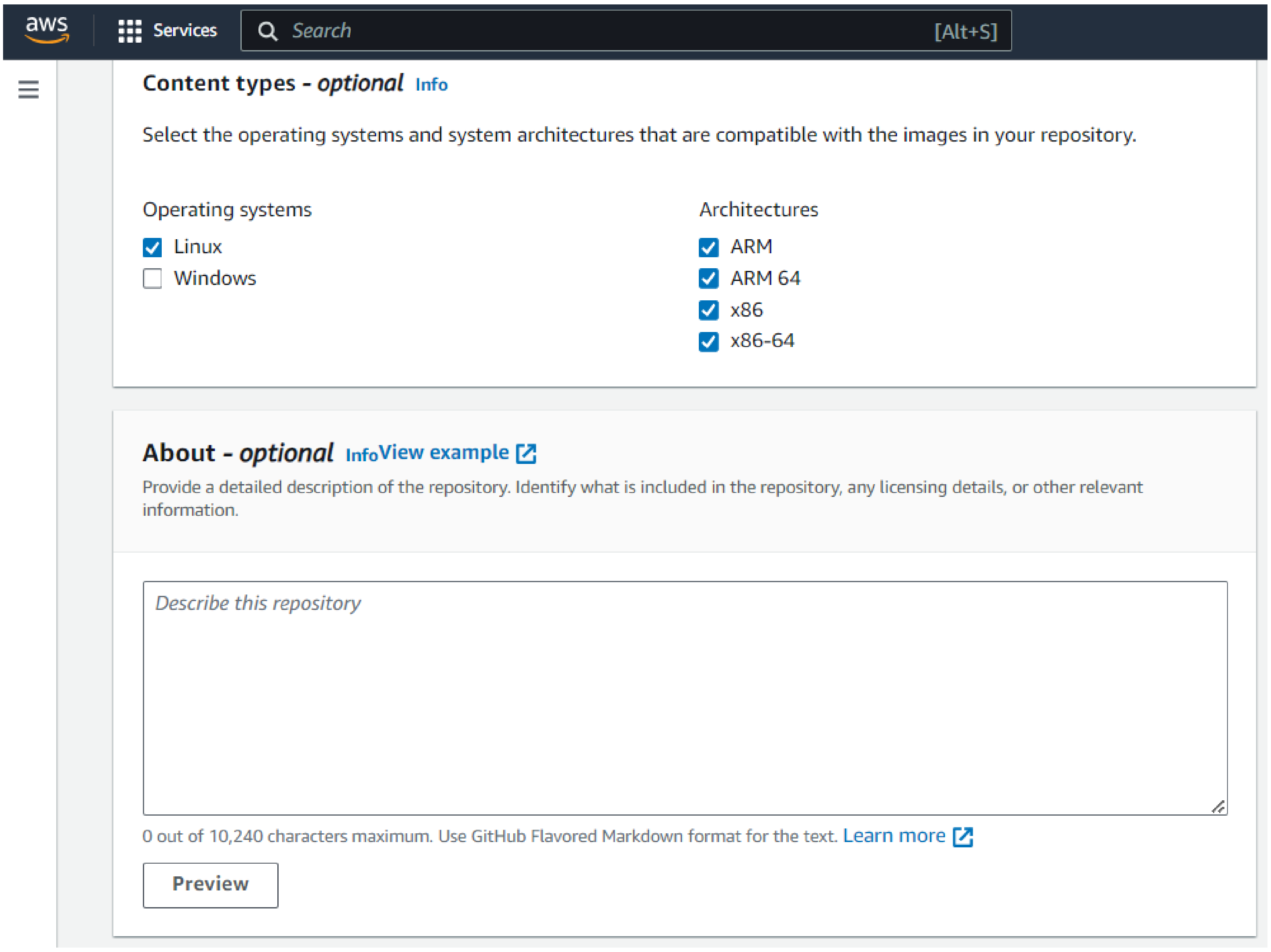
Once a repository is created, the visibility setting of the repository can't be changed.

Detail

Repository name Info
A namespace can be included with your repository name (e.g. namespace/repo-name).

public.ecr.aws/x7p6e5r6/ springbootwebapirepo

20 out of 205 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.



aws Services Search [Alt+S]

Content types - *optional* Info

Select the operating systems and system architectures that are compatible with the images in your repository.

Operating systems

- ✓ Linux
- ☐ Windows

Architectures

- ✓ ARM
- ✓ ARM 64
- ✓ x86
- ✓ x86-64

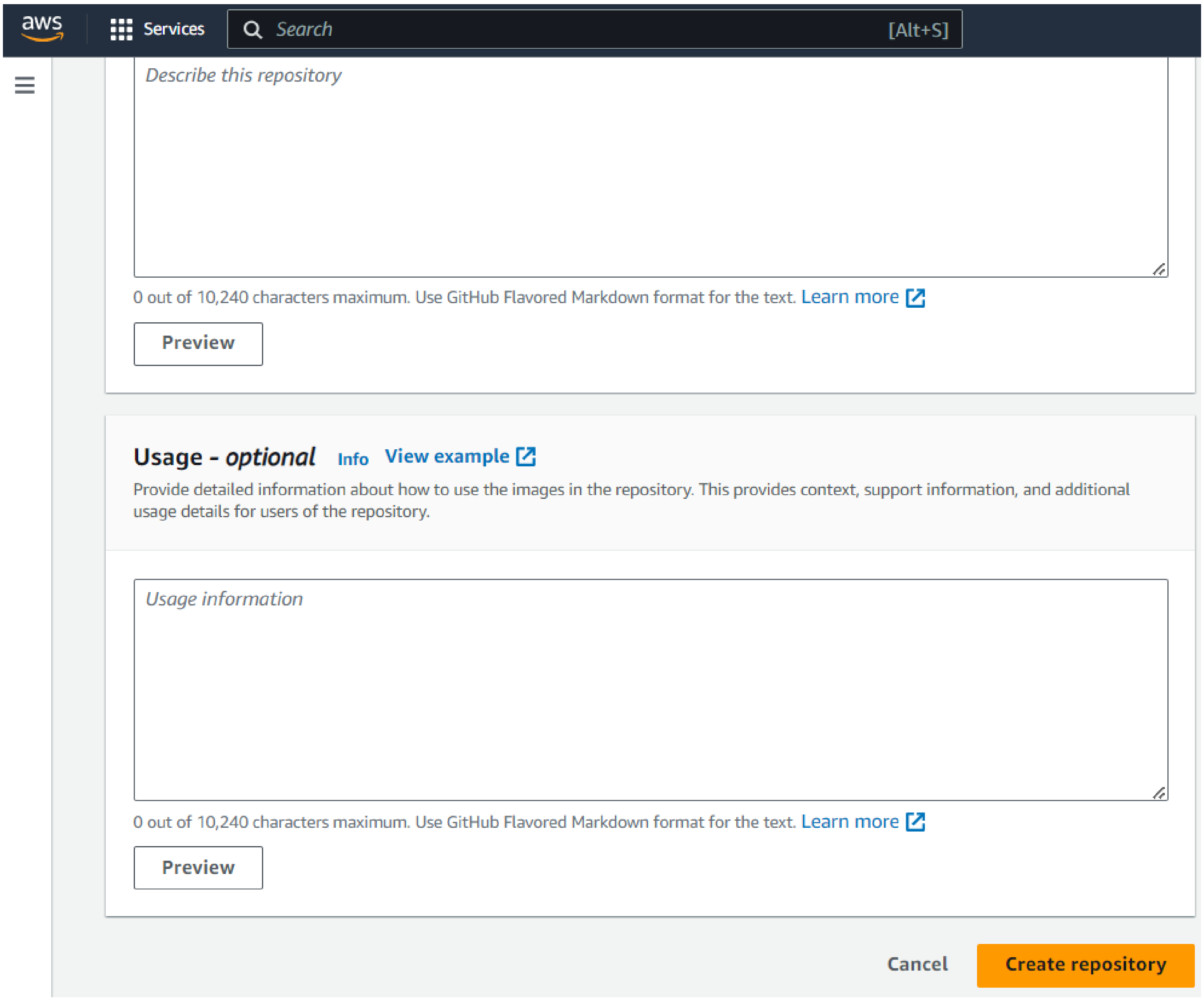
About - *optional* Info View example

Provide a detailed description of the repository. Identify what is included in the repository, any licensing details, or other relevant information.

Describe this repository

0 out of 10,240 characters maximum. Use GitHub Flavored Markdown format for the text. Learn more

Preview



Describe this repository

0 out of 10,240 characters maximum. Use GitHub Flavored Markdown format for the text. Learn more

Preview

Usage - *optional* Info View example

Provide detailed information about how to use the images in the repository. This provides context, support information, and additional usage details for users of the repository.

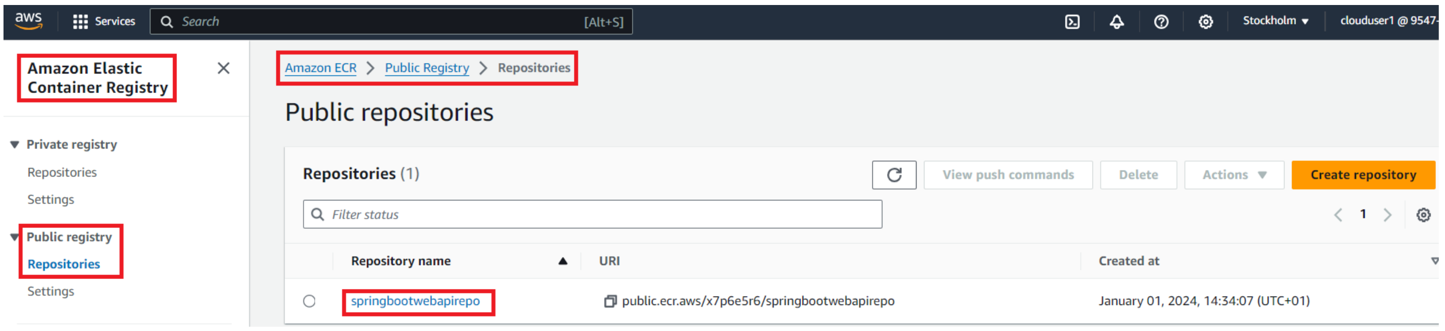
Usage information

0 out of 10,240 characters maximum. Use GitHub Flavored Markdown format for the text. Learn more

Preview

Cancel Create repository

We click on the repo and we upload the application docker image

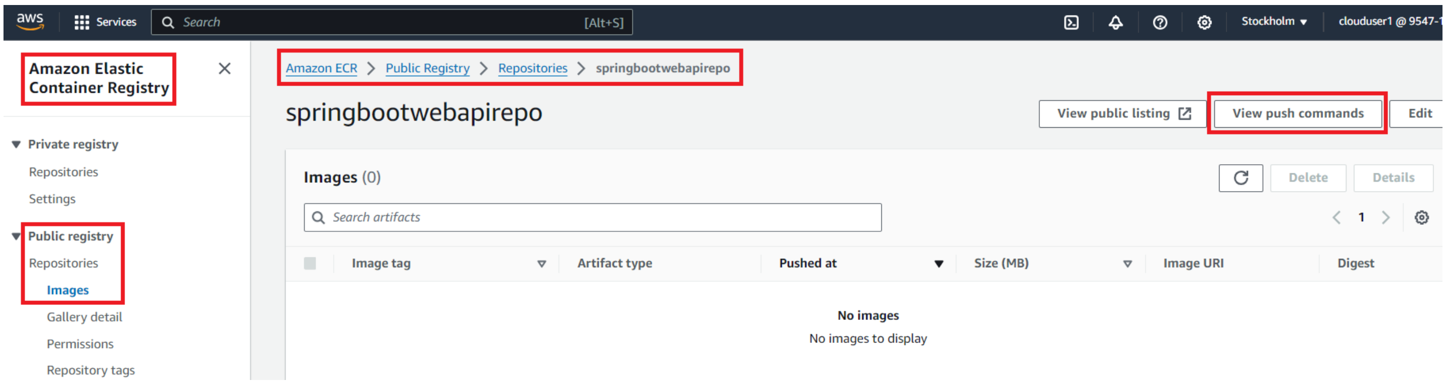


Amazon ECR > Public Registry > Repositories

Public repositories

Repositories (1)

	Repository name	URI	Created at
○	springbootwebapirepo	public.ecr.aws/x7p6e5r6/springbootwebapirepo	January 01, 2024, 14:34:07 (UTC+01)



Amazon ECR > Public Registry > Repositories > springbootwebapirepo

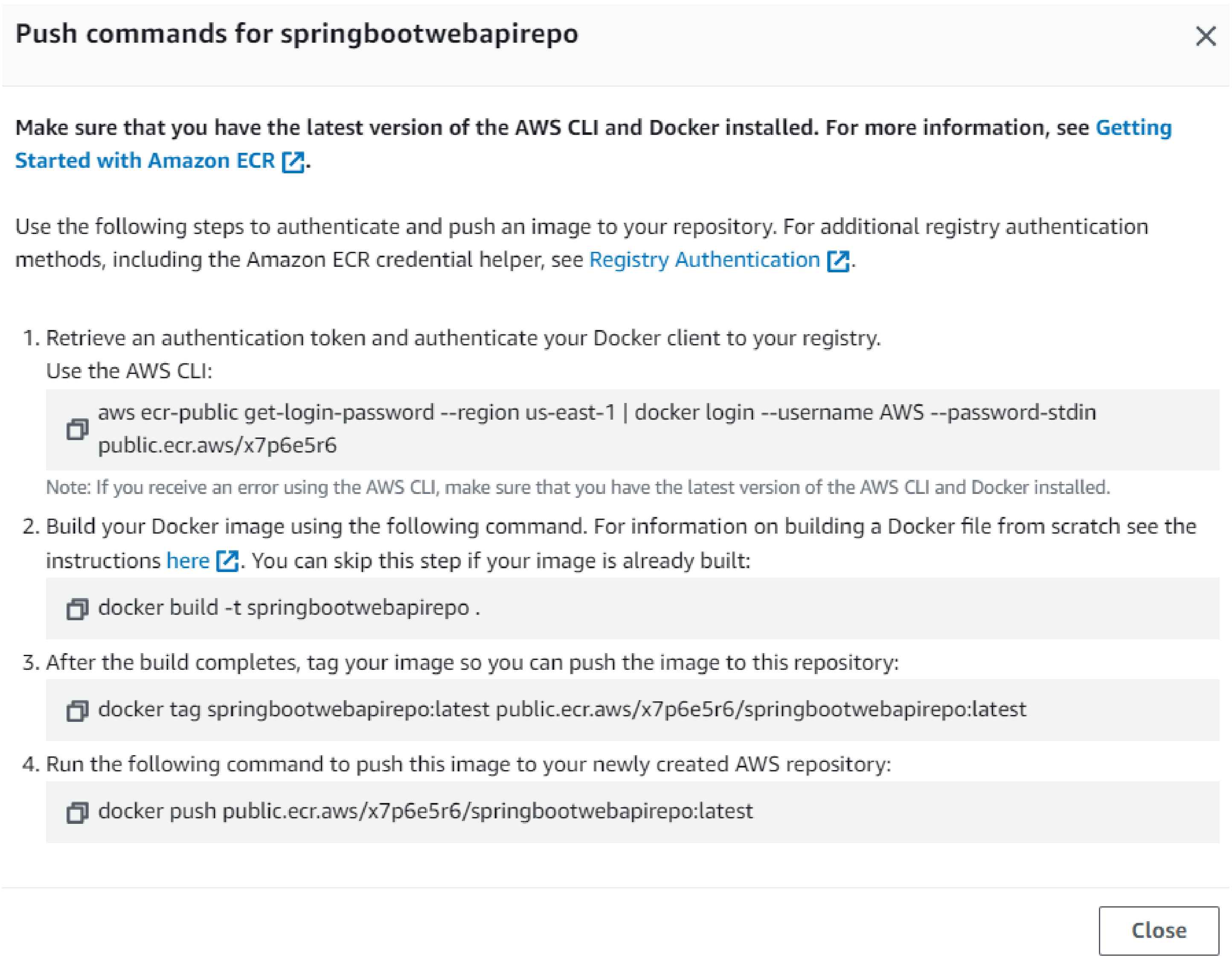
springbootwebapirepo

Images (0)

	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest

No images
No images to display

These are the commnads we have to execute in VSCode Terminal Window



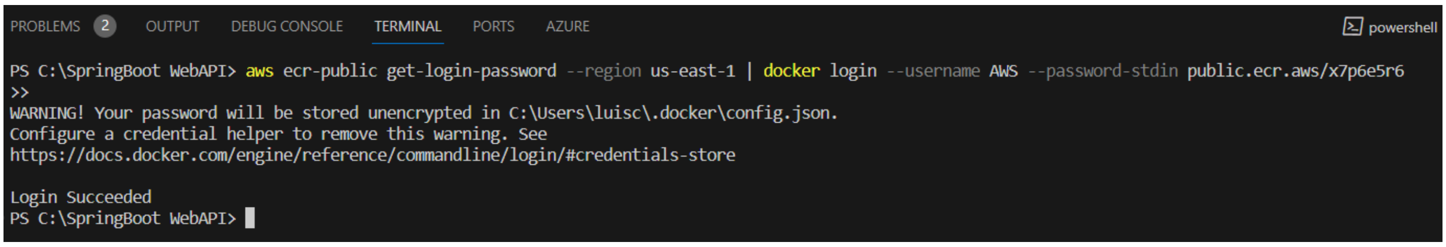
Push commands for springbootwebapirepo

Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see Getting Started with Amazon ECR.

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see Registry Authentication.

1. Retrieve an authentication token and authenticate your Docker client to your registry.
   Use the AWS CLI:
   ```
   aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws/x7p6e5r6
   ```
   Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.
2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions here. You can skip this step if your image is already built:
   ```
   docker build -t springbootwebapirepo .
   ```
3. After the build completes, tag your image so you can push the image to this repository:
   ```
   docker tag springbootwebapirepo:latest public.ecr.aws/x7p6e5r6/springbootwebapirepo:latest
   ```
4. Run the following command to push this image to your newly created AWS repository:
   ```
   docker push public.ecr.aws/x7p6e5r6/springbootwebapirepo:latest
   ```

```
aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-
```



```
PS C:\SpringBoot WebAPI> aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws/x7p6e5r6
>>
WARNING! Your password will be stored unencrypted in C:\Users\luisc\.docker\config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
PS C:\SpringBoot WebAPI>
```

3. Create a Docker image and push it to Docker Desktop

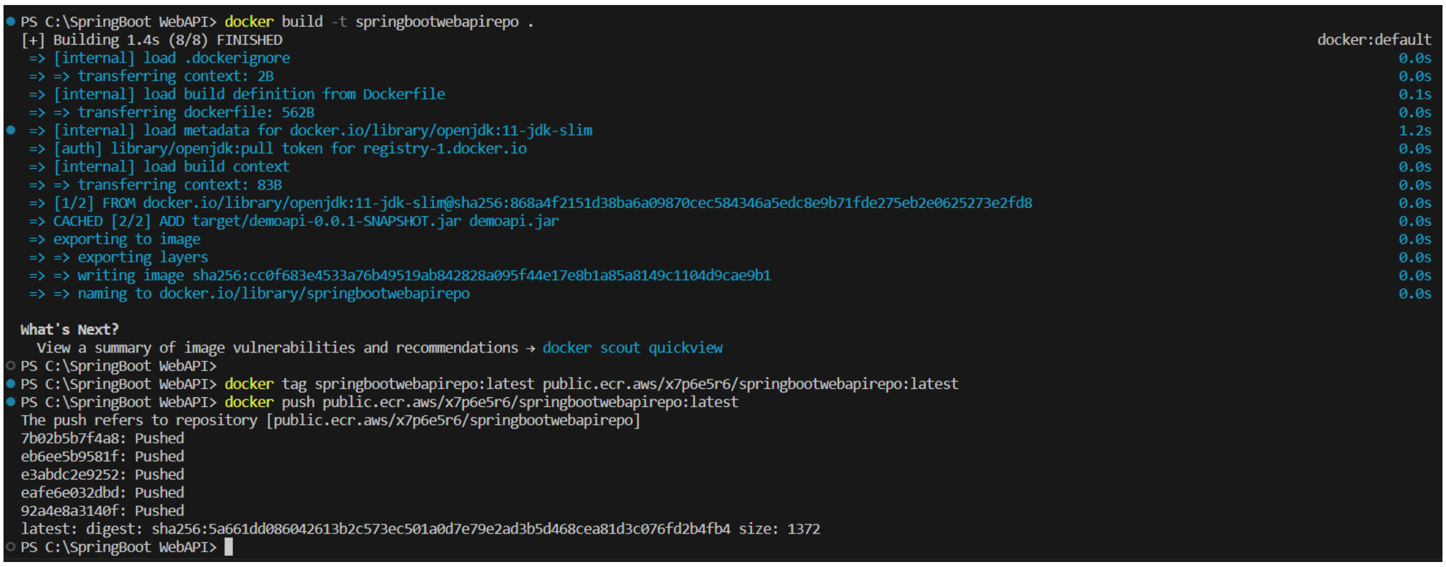
This is the Dockerfile

```
# Start with a base image containing Java runtime
FROM openjdk:11-jdk-slim as build

# Add Maintainer Info
```

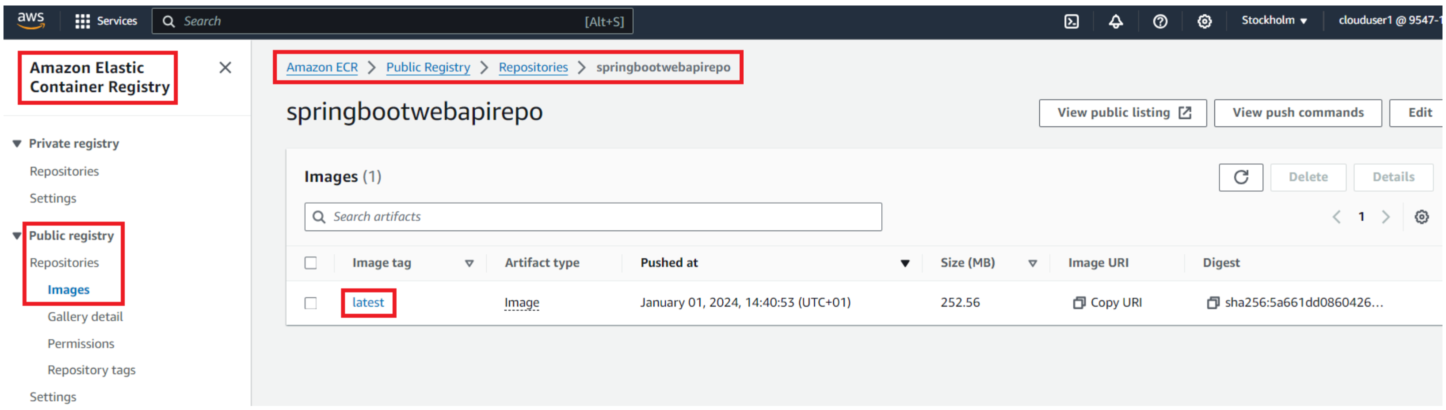
```
LABEL maintainer="your_email@example.com"

# Add a volume pointing to /tmp
VOLUME /tmp

# Make port 8080 available to the world outside this container
EXPOSE 80

# The application's jar file
ARG JAR_FILE=target/demoapi-0.0.1-SNAPSHOT.jar

# Add the application's jar to the container
ADD ${JAR_FILE} demoapi.jar

# Run the jar file
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/demoapi.jar"]
```

Build the Docker image and push it to AWS ECR with these commands:

```
docker build -t springbootwebapirepo .

docker tag springbootwebapirepo:latest public.ecr.aws/x7p6e5r6/springbootwebapirepo:latest

docker push public.ecr.aws/x7p6e5r6/springbootwebapirepo:latest
```



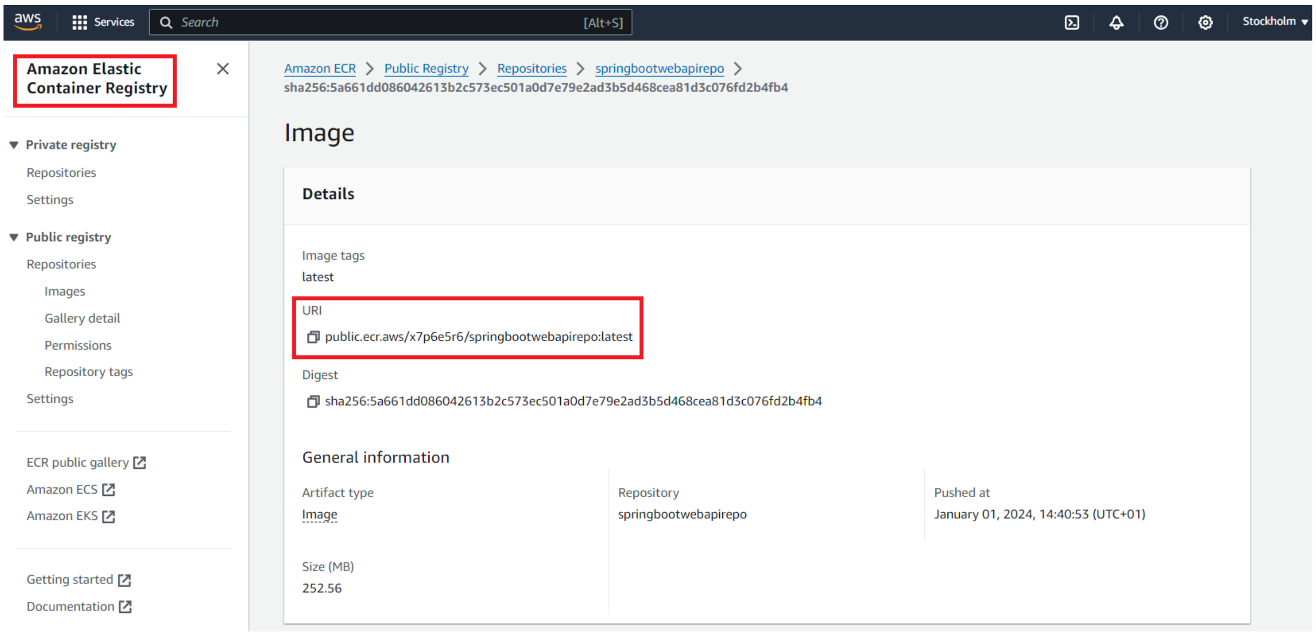
```
PS C:\SpringBoot WebAPI> docker build -t springbootwebapirepo .
[+] Building 1.4s (8/8) FINISHED                                                                       docker:default
 => [internal] load .dockerignore                                                                                0.0s
 => => transferring context: 2B                                                                                  0.0s
 => [internal] load build definition from Dockerfile                                                             0.1s
 => => transferring dockerfile: 562B                                                                             0.0s
 => [internal] load metadata for docker.io/library/openjdk:11-jdk-slim                                           1.2s
 => [auth] library/openjdk:pull token for registry-1.docker.io                                                   0.0s
 => [internal] load build context                                                                                0.0s
 => => transferring context: 83B                                                                                 0.0s
 => [1/2] FROM docker.io/library/openjdk:11-jdk-slim@sha256:868a4f2151d38ba6a09870cec584346a5edc8e9b71fde275eb2e0625273e2fd8  0.0s
 => CACHED [2/2] ADD target/demoapi-0.0.1-SNAPSHOT.jar demoapi.jar                                               0.0s
 => exporting to image                                                                                           0.0s
 => => exporting layers                                                                                          0.0s
 => => writing image sha256:cc0f683e4533a76b49519ab842828a095f44e17e8b1a85a8149c1104d9cae9b1                     0.0s
 => => naming to docker.io/library/springbootwebapirepo                                                          0.0s

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\SpringBoot WebAPI>
PS C:\SpringBoot WebAPI> docker tag springbootwebapirepo:latest public.ecr.aws/x7p6e5r6/springbootwebapirepo:latest
PS C:\SpringBoot WebAPI> docker push public.ecr.aws/x7p6e5r6/springbootwebapirepo:latest
The push refers to repository [public.ecr.aws/x7p6e5r6/springbootwebapirepo]
7b02b5b7f4a8: Pushed
eb6ee5b9581f: Pushed
e3abdc2e9252: Pushed
eafe6e032dbd: Pushed
92a4e8a3140f: Pushed
latest: digest: sha256:5a661dd086042613b2c573ec501a0d7e79e2ad3b5d468cea81d3c076fd2b4fb4 size: 1372
PS C:\SpringBoot WebAPI>
```



springbootwebapirepo

Images (1)

	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
☐	latest	Image	January 01, 2024, 14:40:53 (UTC+01)	252.56	Copy URI	sha256:5a661dd0860426...



Amazon ECR > Public Registry > Repositories > springbootwebapirepo > sha256:5a661dd086042613b2c573ec501a0d7e79e2ad3b5d468cea81d3c076fd2b4fb4

Image

Details

Image tags
latest

URI
public.ecr.aws/x7p6e5r6/springbootwebapirepo:latest

Digest
sha256:5a661dd086042613b2c573ec501a0d7e79e2ad3b5d468cea81d3c076fd2b4fb4

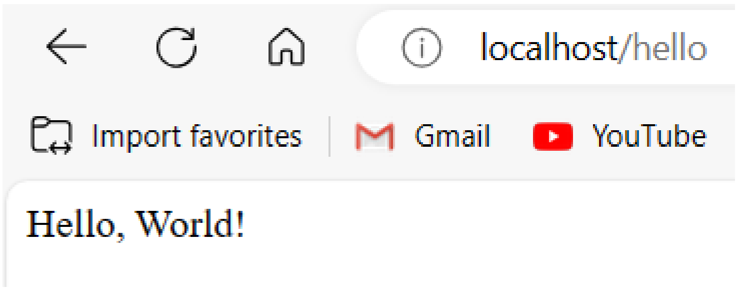
General information

Artifact type	Repository	Pushed at
Image	springbootwebapirepo	January 01, 2024, 14:40:53 (UTC+01)

Size (MB)
252.56

Run the Docker container with this command:

```
docker run -p 80:80 public.ecr.aws/x7p6e5r6/springbootwebapirepo:latest
```



Hello, World!

4. Create the AWS EKS cluster

We run this command for creating AWS EKS:

```
eksctl create cluster ^
--name springbootwebapi-cluster ^
--version 1.25 ^--region eu-west-3 ^
--nodegroup-name linux-nodes ^
--node-type t2.micro ^
--nodes 4
```

5. Deploy you application in AWS EKS Elastic Kluster

To deploy your application in AWS Elastic Kubernetes Service (EKS), you'll need to create two YAML files: one for the deployment (**deployment.yml**) and one for the service (**service.yml**).

Below are sample YAML files that you can use and modify according to your requirements.

Deployment YAML (deployment.yml)

This file defines the deployment of your application in the Kubernetes cluster.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demoapi-deployment
spec:
  replicas: 1  # The number of Pods to run
  selector:
    matchLabels:
      app: demoapi
  template:
    metadata:
      labels:
        app: demoapi
    spec:
      containers:
        - name: demoapi
          image: public.ecr.aws/x7p6e5r6/springbootwebapirepo:latest
          ports:
            - containerPort: 80
```

IMPORTANT: pay attention replace set the image name.

See section 3 ... image: public.ecr.aws/x7p6e5r6/springbootwebapirepo:latest ...

Service YAML (service.yml)

This file defines how your application is exposed.

```
apiVersion: v1
kind: Service
metadata:
  name: demoapi-service
spec:
  type: LoadBalancer  # Exposes the service externally using a load balancer
  selector:
    app: demoapi
  ports:
    - protocol: TCP
      port: 80  # The port the load balancer listens on
      targetPort: 80  # The port the container accepts traffic on
```

Deploying to AWS EKS

After creating these files, you can use the kubectl command-line tool to apply these configurations to your EKS cluster:

Deploy the Application:

```
kubectl apply -f deployment.yml
```

Create the Service

```
kubectl apply -f service.yml
```

Verify the Deployment:

Check the status of the deployment:

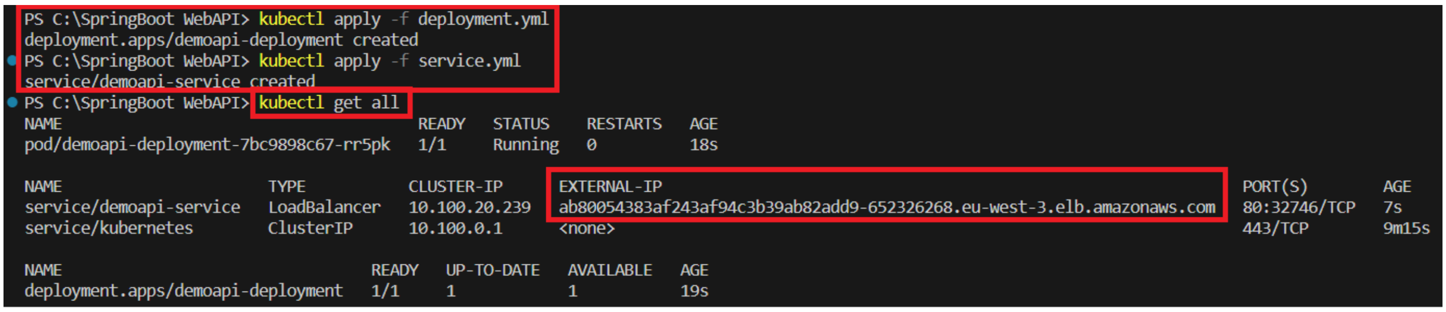
```
kubectl get deployments
```

Check the status of the pods:

```
kubectl get pods
```

```
kubectl get service demoapi-service
```

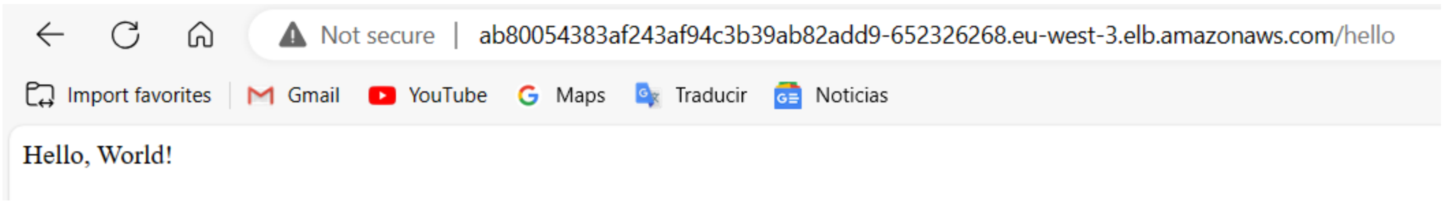
```
kubectl get all
```



```
PS C:\SpringBoot WebAPI> kubectl apply -f deployment.yml
deployment.apps/demoapi-deployment created
PS C:\SpringBoot WebAPI> kubectl apply -f service.yml
service/demoapi-service created
PS C:\SpringBoot WebAPI> kubectl get all
NAME                                      READY   STATUS    RESTARTS   AGE
pod/demoapi-deployment-7bc9898c67-rr5pk   1/1     Running   0          18s

NAME                      TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(S)        AGE
service/demoapi-service   LoadBalancer   10.100.20.239   ab80054383af243af94c3b39ab82add9-652326268.eu-west-3.elb.amazonaws.com   80:32746/TCP   7s
service/kubernetes        ClusterIP      10.100.0.1      <none>                                                                     443/TCP        9m15s

NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/demoapi-deployment   1/1     1            1           19s
```

Use the external IP address to access your application in a web browser: **http://<EXTERNAL-IP>/hello**

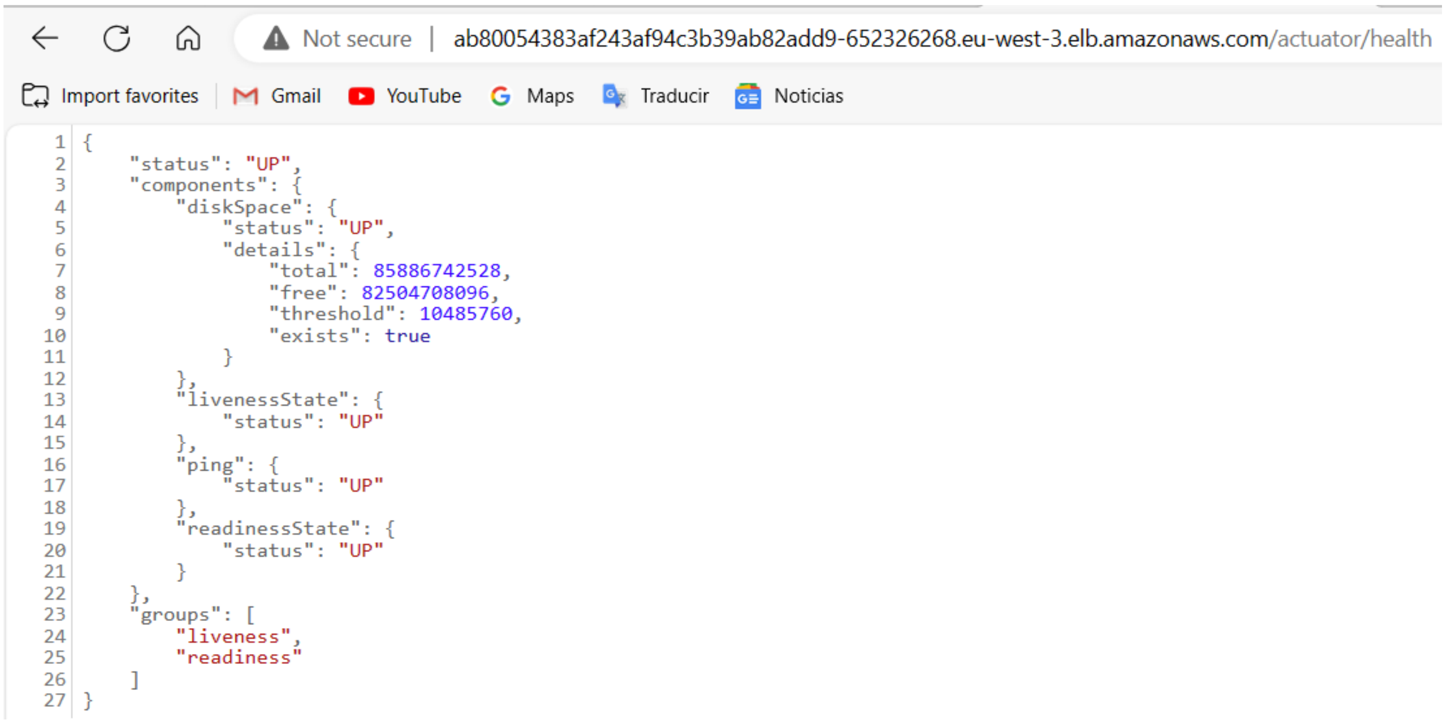
http://ab80054383af243af94c3b39ab82add9-652326268.eu-west-3.elb.amazonaws.com/hello



Not secure
ab80054383af243af94c3b39ab82add9-652326268.eu-west-3.elb.amazonaws.com/hello
Import favorites
Gmail
YouTube
Maps
Traducir
Noticias

Hello, World!

http://ab80054383af243af94c3b39ab82add9-652326268.eu-west-3.elb.amazonaws.com/actuator/health


Not secure
ab80054383af243af94c3b39ab82add9-652326268.eu-west-3.elb.amazonaws.com/actuator/health
Import favorites
Gmail
YouTube
Maps
Traducir
Noticias

```
{
    "status": "UP",
    "components": {
        "diskSpace": {
            "status": "UP",
            "details": {
                "total": 85886742528,
                "free": 82504708096,
                "threshold": 10485760,
                "exists": true
            }
        },
        "livenessState": {
            "status": "UP"
        },
        "ping": {
            "status": "UP"
        },
        "readinessState": {
            "status": "UP"
        }
    },
    "groups": [
        "liveness",
        "readiness"
    ]
}
```