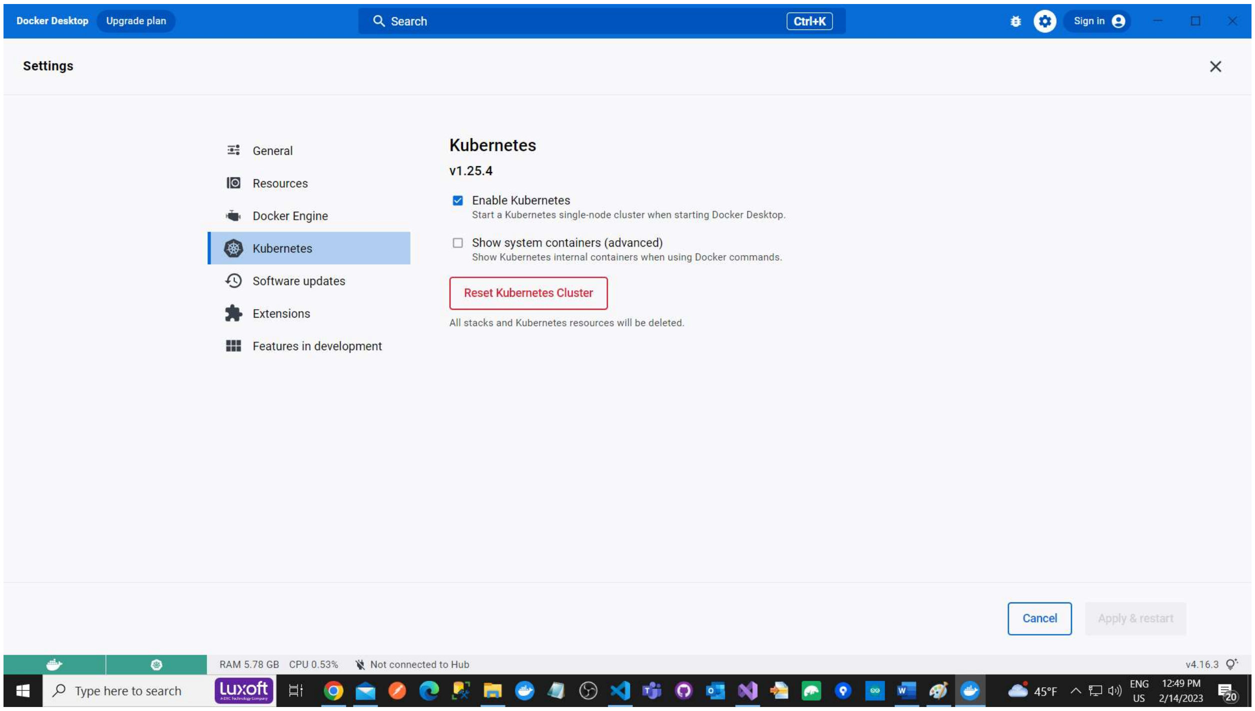
How to deploy on AWS EKS a MongoDb ReplicaSet

A. Prerequisites

0. Install Docker Desktop and enable Kubernetes.
https://www.docker.com/products/docker-desktop/



Kubernetes
v1.25.4

☑ 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.

1. Install AWS CLI
https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

2. Install kubectl
https://kubernetes.io/docs/tasks/tools/

3. install eksctl
https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html

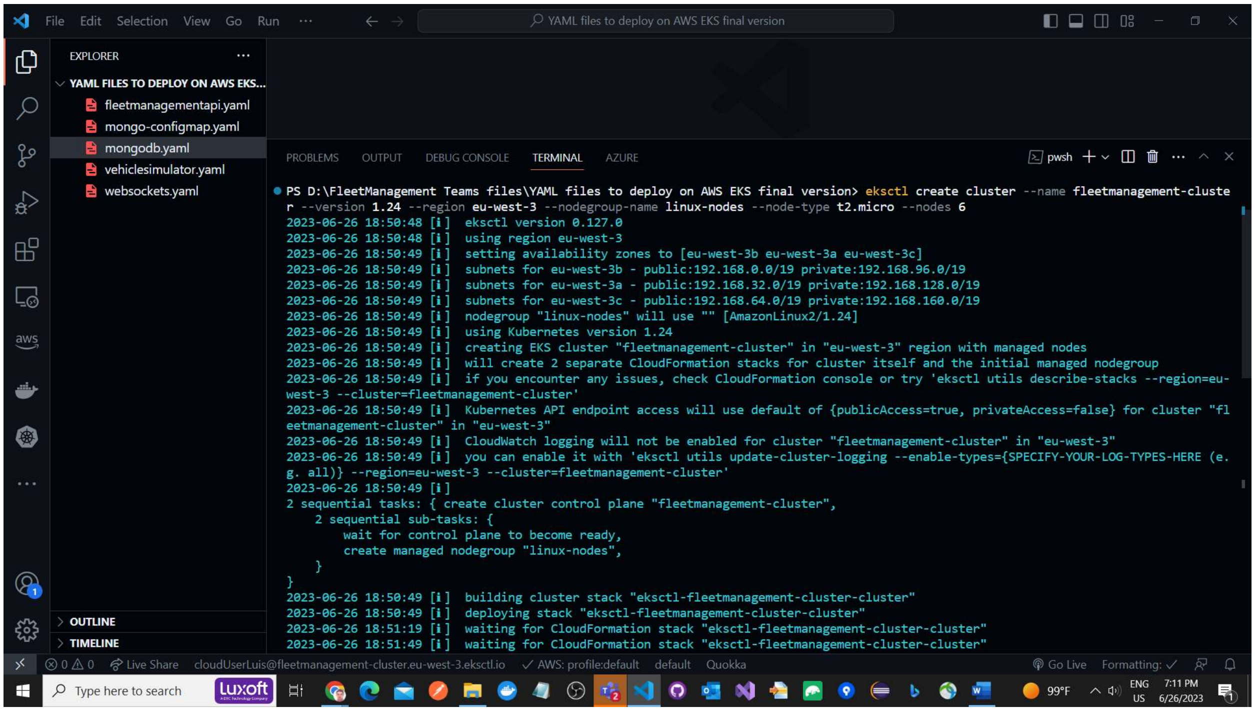
4. Login in the AWS Console.
Go to EKS service.

5. Create a new cluster with the following command:

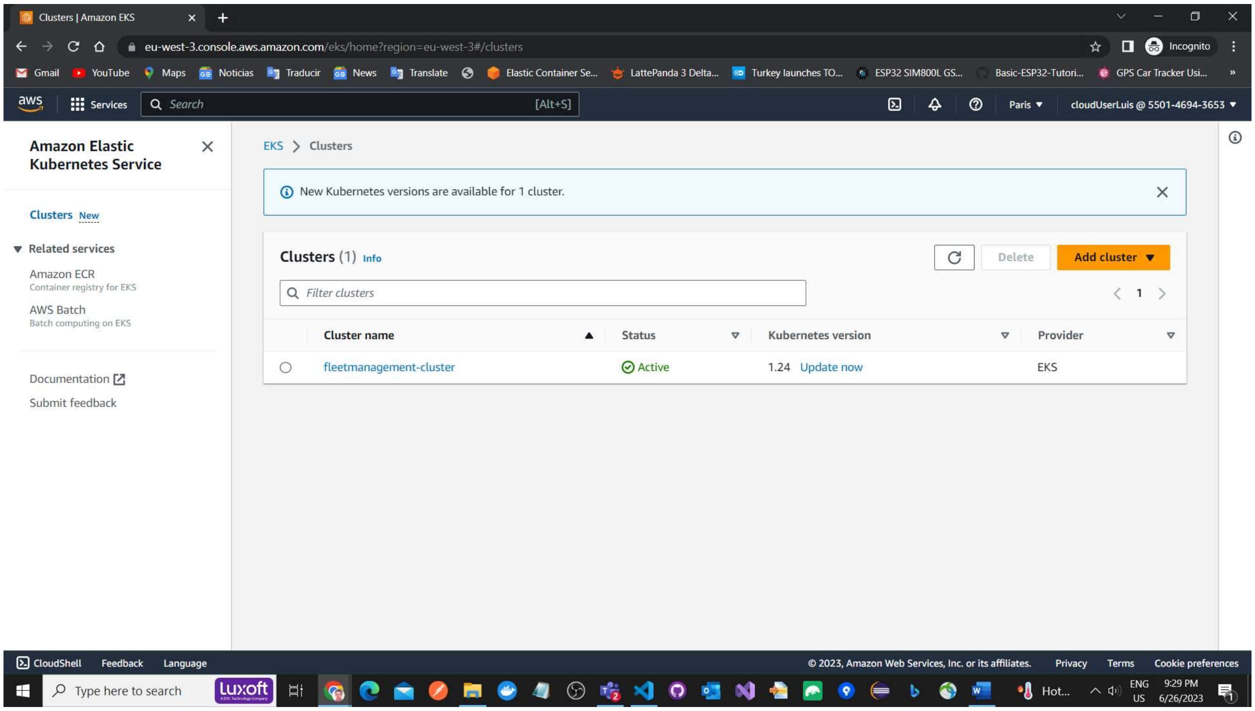
eksctl create cluster --name fleetmanagement-cluster --version 1.24 --region eu-west-3 --nodegroup-name linux-nodes --node-type t2.micro --nodes 6

When creating a new cluster we set:
-the cluster name: fleetmanagement-cluster
-the Kubernetes version: 1.24
-the region: eu-west-3
-the nodegroup-name: linux-nodes
-the node-type: t2.micro
-the number of nodes: 6

IMPORTANT!!! Set MINIMUN 6 Nodes for deploying the MongoDb ReplicaSet, the WebAPI and the ClientAPI application



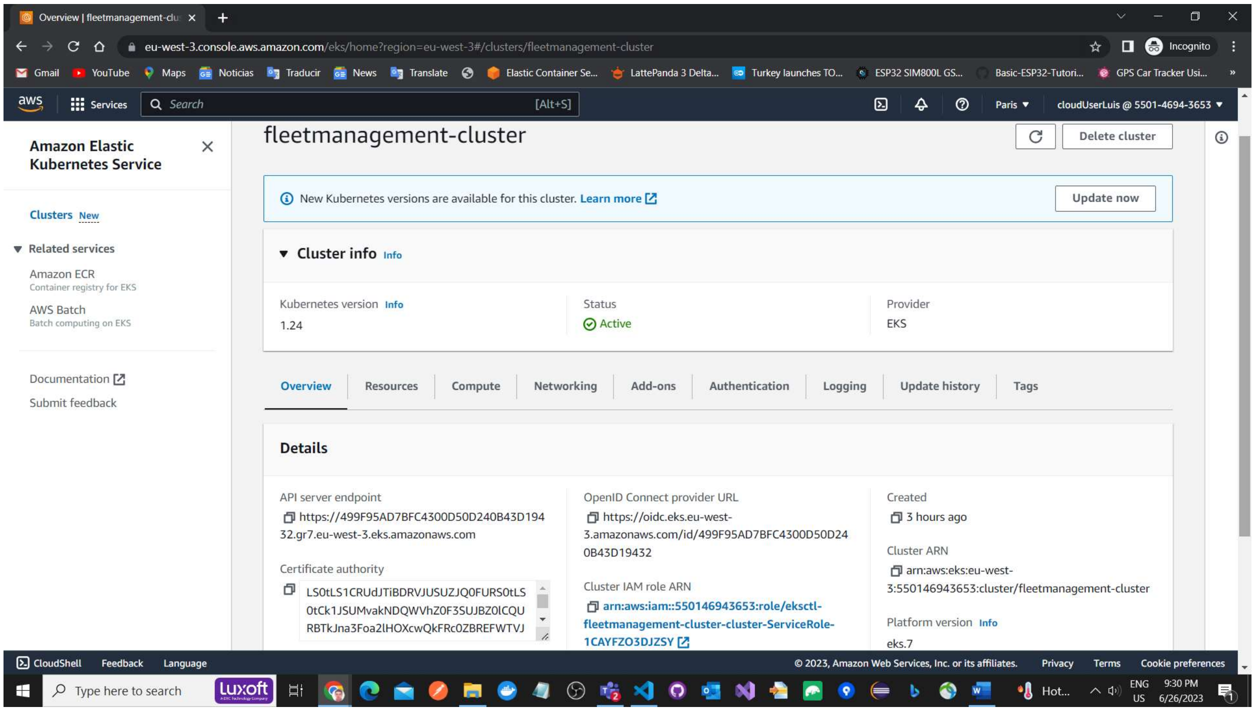
```
PS D:\FleetManagement Teams files\YAML files to deploy on AWS EKS final version> eksctl create cluster --name fleetmanagement-cluste
r --version 1.24 --region eu-west-3 --nodegroup-name linux-nodes --node-type t2.micro --nodes 6
2023-06-26 18:50:48 [ℹ]  eksctl version 0.127.0
2023-06-26 18:50:48 [ℹ]  using region eu-west-3
2023-06-26 18:50:49 [ℹ]  setting availability zones to [eu-west-3b eu-west-3a eu-west-3c]
2023-06-26 18:50:49 [ℹ]  subnets for eu-west-3b - public:192.168.0.0/19 private:192.168.96.0/19
2023-06-26 18:50:49 [ℹ]  subnets for eu-west-3a - public:192.168.32.0/19 private:192.168.128.0/19
2023-06-26 18:50:49 [ℹ]  subnets for eu-west-3c - public:192.168.64.0/19 private:192.168.160.0/19
2023-06-26 18:50:49 [ℹ]  nodegroup "linux-nodes" will use "" [AmazonLinux2/1.24]
2023-06-26 18:50:49 [ℹ]  using Kubernetes version 1.24
2023-06-26 18:50:49 [ℹ]  creating EKS cluster "fleetmanagement-cluster" in "eu-west-3" region with managed nodes
2023-06-26 18:50:49 [ℹ]  will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2023-06-26 18:50:49 [ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=eu-
west-3 --cluster=fleetmanagement-cluster'
2023-06-26 18:50:49 [ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "fl
eetmanagement-cluster" in "eu-west-3"
2023-06-26 18:50:49 [ℹ]  CloudWatch logging will not be enabled for cluster "fleetmanagement-cluster" in "eu-west-3"
2023-06-26 18:50:49 [ℹ]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.
g. all)} --region=eu-west-3 --cluster=fleetmanagement-cluster'
2023-06-26 18:50:49 [ℹ]
2 sequential tasks: { create cluster control plane "fleetmanagement-cluster",
    2 sequential sub-tasks: {
        wait for control plane to become ready,
        create managed nodegroup "linux-nodes",
    }
}
2023-06-26 18:50:49 [ℹ]  building cluster stack "eksctl-fleetmanagement-cluster-cluster"
2023-06-26 18:50:49 [ℹ]  deploying stack "eksctl-fleetmanagement-cluster-cluster"
2023-06-26 18:51:19 [ℹ]  waiting for CloudFormation stack "eksctl-fleetmanagement-cluster-cluster"
2023-06-26 18:51:49 [ℹ]  waiting for CloudFormation stack "eksctl-fleetmanagement-cluster-cluster"
```



New Kubernetes versions are available for 1 cluster.

Clusters (1) Info

	Cluster name	Status	Kubernetes version	Provider
○	fleetmanagement-cluster	Active	1.24 Update now	EKS



fleetmanagement-cluster

New Kubernetes versions are available for this cluster. Learn more

Cluster info Info

Kubernetes version Info
1.24

Status
Active

Provider
EKS

Details

API server endpoint
https://499F95AD7BFC4300D50D240B43D19432.gr7.eu-west-3.eks.amazonaws.com

Certificate authority
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMvakNDQWVhZ0F3SUJBZ0lCQURBTkJna3Foa2lHOXcwQkFRc0ZBREFWTVJ

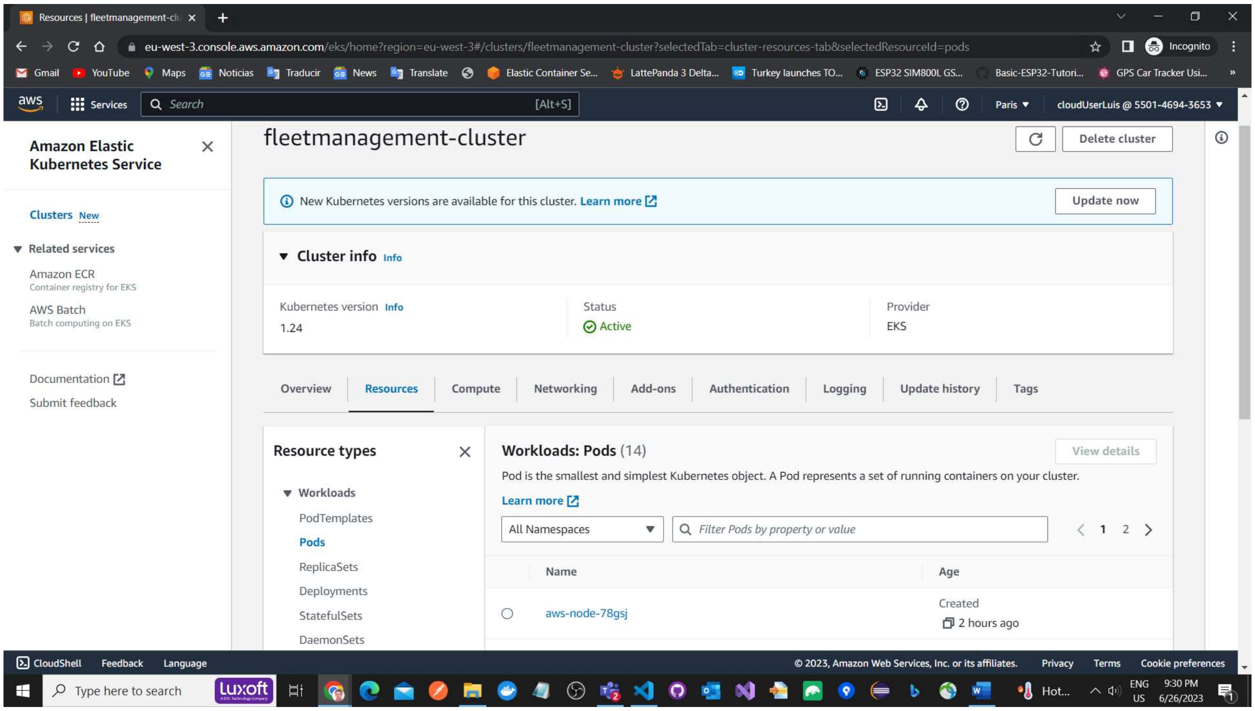
OpenID Connect provider URL
https://oidc.eks.eu-west-3.amazonaws.com/id/499F95AD7BFC4300D50D240B43D19432

Cluster IAM role ARN
arn:aws:iam::550146943653:role/eksctl-fleetmanagement-cluster-cluster-ServiceRole-1CAYFZO3DJZSY

Created
3 hours ago

Cluster ARN
arn:aws:eks:eu-west-3:550146943653:cluster/fleetmanagement-cluster

Platform version Info
eks.7



fleetmanagement-cluster

Delete cluster

ⓘ New Kubernetes versions are available for this cluster. Learn more

Update now

▼ Cluster info Info

Kubernetes version Info	Status	Provider
1.24	✓ Active	EKS

Overview | Resources | Compute | Networking | Add-ons | Authentication | Logging | Update history | Tags

Resource types

▼ Workloads
- PodTemplates
- Pods
- ReplicaSets
- Deployments
- StatefulSets
- DaemonSets

Workloads: Pods (14)

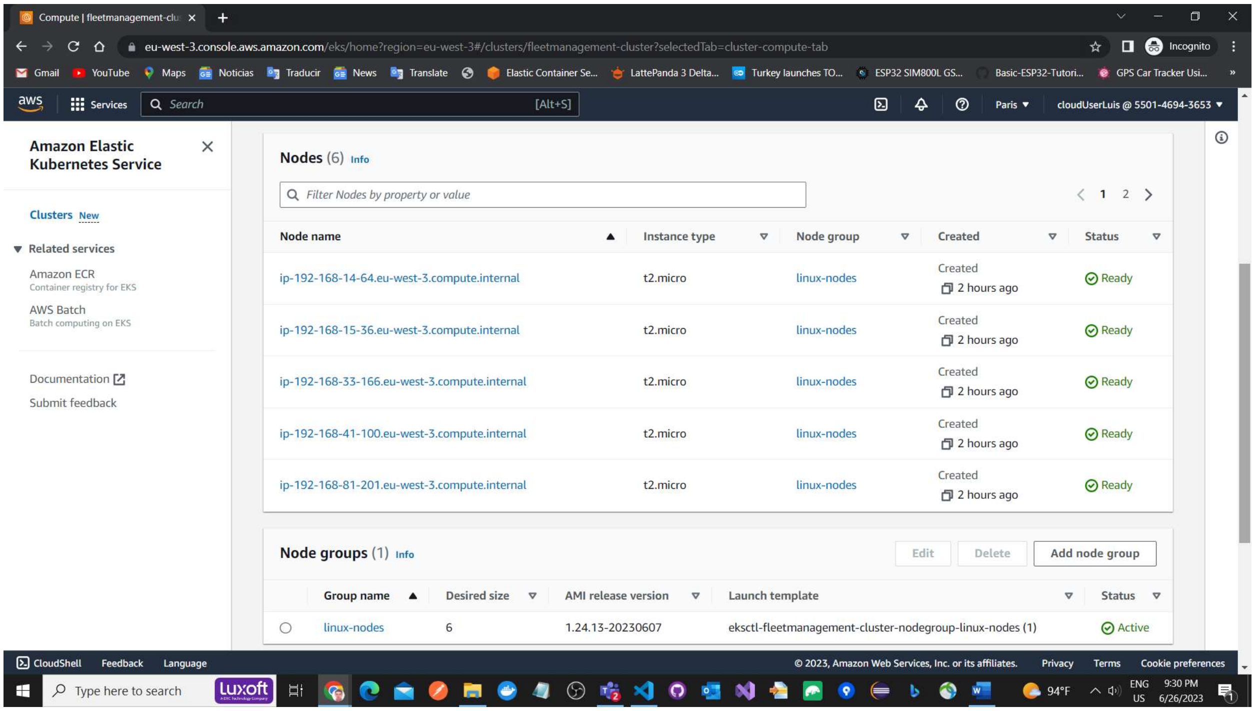
View details

Pod is the smallest and simplest Kubernetes object. A Pod represents a set of running containers on your cluster.

Learn more

All Namespaces ▾ | Filter Pods by property or value | ‹ 1 2 ›

	Name	Age
○	aws-node-78gsj	Created 2 hours ago



Nodes (6) Info

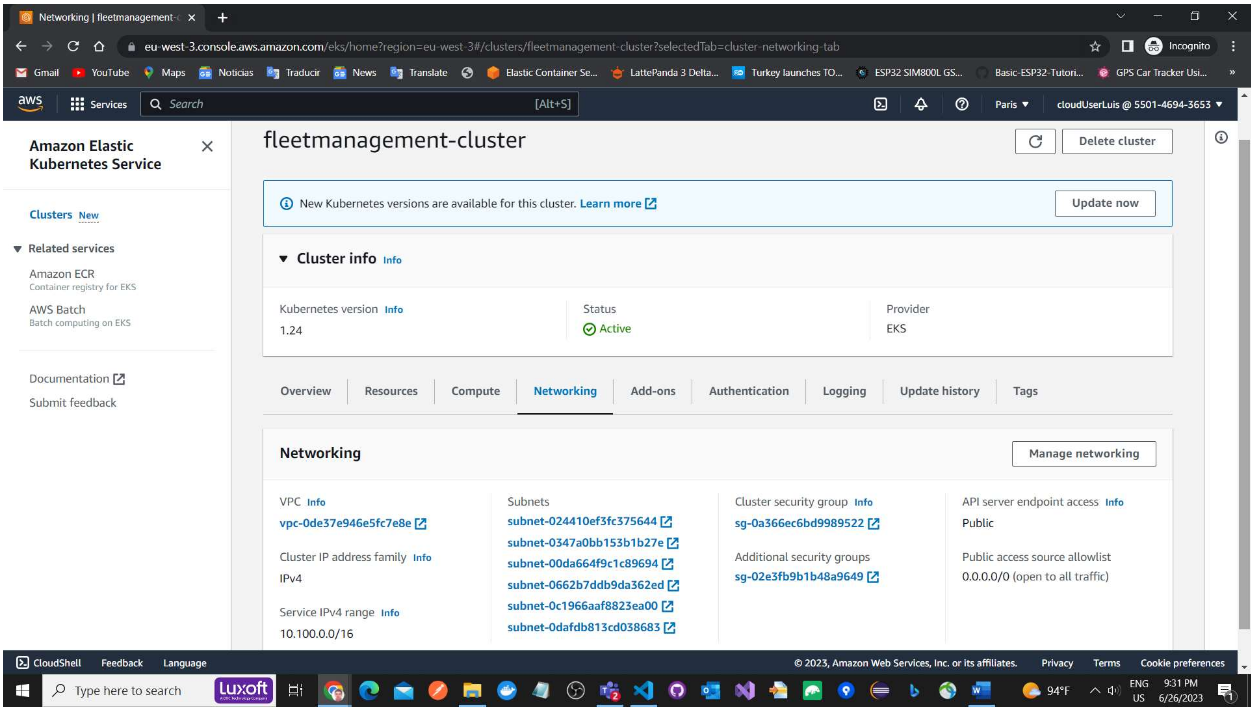
Filter Nodes by property or value | ‹ 1 2 ›

Node name ▲	Instance type	Node group	Created	Status
ip-192-168-14-64.eu-west-3.compute.internal	t2.micro	linux-nodes	Created 2 hours ago	✓ Ready
ip-192-168-15-36.eu-west-3.compute.internal	t2.micro	linux-nodes	Created 2 hours ago	✓ Ready
ip-192-168-33-166.eu-west-3.compute.internal	t2.micro	linux-nodes	Created 2 hours ago	✓ Ready
ip-192-168-41-100.eu-west-3.compute.internal	t2.micro	linux-nodes	Created 2 hours ago	✓ Ready
ip-192-168-81-201.eu-west-3.compute.internal	t2.micro	linux-nodes	Created 2 hours ago	✓ Ready

Node groups (1) Info

Edit | Delete | Add node group

	Group name ▲	Desired size	AMI release version	Launch template	Status
○	linux-nodes	6	1.24.13-20230607	eksctl-fleetmanagement-cluster-nodegroup-linux-nodes (1)	✓ Active



fleetmanagement-cluster

New Kubernetes versions are available for this cluster. Learn more

Cluster info Info

Kubernetes version Info: 1.24
Status: Active
Provider: EKS

Networking

VPC Info: vpc-0de37e946e5fc7e8e
Cluster IP address family Info: IPv4
Service IPv4 range Info: 10.100.0.0/16

Subnets:
subnet-024410ef3fc375644
subnet-0347a0bb153b1b27e
subnet-00da664f9c1c89694
subnet-0662b7ddb9da362ed
subnet-0c1966aaf8823ea00
subnet-0dafdb813cd038683

Cluster security group Info: sg-0a366ec6bd9989522
Additional security groups: sg-02e3fb9b1b48a9649

API server endpoint access Info: Public
Public access source allowlist: 0.0.0.0/0 (open to all traffic)

6. By default, CloudWatch logging not enabled for cluster "fleetmanagement-cluster" in "eu-west-3", you can enable it with:

```
eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=eu-west-3 --cluster=fleetmanagement-cluster --approve
```

For example:

```
eksctl utils update-cluster-logging --enable-types=all --region=eu-west-3 --cluster=fleetmanagement-cluster --approve
```

7. To list the clusters run the command:

```
kubectl config get-contexts
```

(OPTIONAL) you can delete a context with the following command.
For example:

```
kubectl config delete-context contextname
kubectl config delete-context luis.enriquez@dxc.com@test-cluster.eu-west-3.eksctl.io
```

8. Select the Cluster you have just created in AWS EKS.
To select a cluster where to deploy applications, run the command:

```
kubectl config use-context contextname
```

For example:

```
kubectl config use-context luis.enriquez@dxc.com@fleetmanagement-cluster.eu-west-3.eksctl.io
```

In this case we set to the context:
luis.enriquez@dxc.com@test-cluster.eu-west-3.eksctl.io
The contextName is composed by: IAMUser@clusterName.RegionName.eksctl.io
IAM user: luis.enriquez@dxc.com
clusterName: test-cluster
RegionName:eu-west-3

```
kubectl config use-context luis.enriquez@dxc.com@test-cluster.eu-west-3.eksctl.io
```

9. Create the “dev” namespace

```
kubectl create namespace dev
```

```
kubectl get nodes
```

```
kubectl get ns
```

```
kubectl get all
```

```
kubectl get pods
```

```
kubectl get all --namespace dev
```

```
kubectl get nodes --namespace dev
```

```
kubectl get ns --namespace dev
```

```
kubectl get pods --namespace dev
```

12. Prior to deploying our applications to EKS, we have to **create the applications docker images and upload them to AWS ECR**.

Create the private repositories in the AWS ECR, and to build and upload the images follow the instructions inside each repository.

For example:
-We create a mongodb private reposiroty.
-We login in the ECR repository:

```
aws ecr get-login-password --region eu-west-3 | docker login --username AWS --password-stdin 719220092744.dkr.ecr.eu-west-3.amazonaws.com
```

-We pull the mongo:latest image from Docker Hub to my local:

Go to Docker Hub: https://hub.docker.com/
Search for: mongo
Run the command to pull the image from Docker Hub to my local computer:

```
docker pull mongo:latest
```

-Rename the image before uploading to the AWS ECR:

```
docker tag mongo:latest 719220092744.dkr.ecr.eu-west-3.amazonaws.com/mongodb:latest
```

-Upload the image to AWS ECR:

```
docker push 719220092744.dkr.ecr.eu-west-3.amazonaws.com/mongodb:latest
```

13. Before start to deploy the applications in the AWS EKS cluster, we have to select the context (cluster name) for the deployment, and also it is convenient to create a new namespace where to deploy the applications, for example the "dev" namespace.

As we mentioned in the sections 7 and 8

```
kubectl config get-contexts
```

An start "*" is next to the used/actual cluster name.

To select or use another cluster run the command:

```
kubectl config use-context clustername
```

```
kubectl config use-context luis.enriquez@dxc.com@test1-cluster.eu-west-3.eksctl.io
```

For creating a new name space "dev"

```
kubectl create namespace dev
```

For listing the namespaces run the command

```
kubectl get ns
```

For deleting a namespace run the command

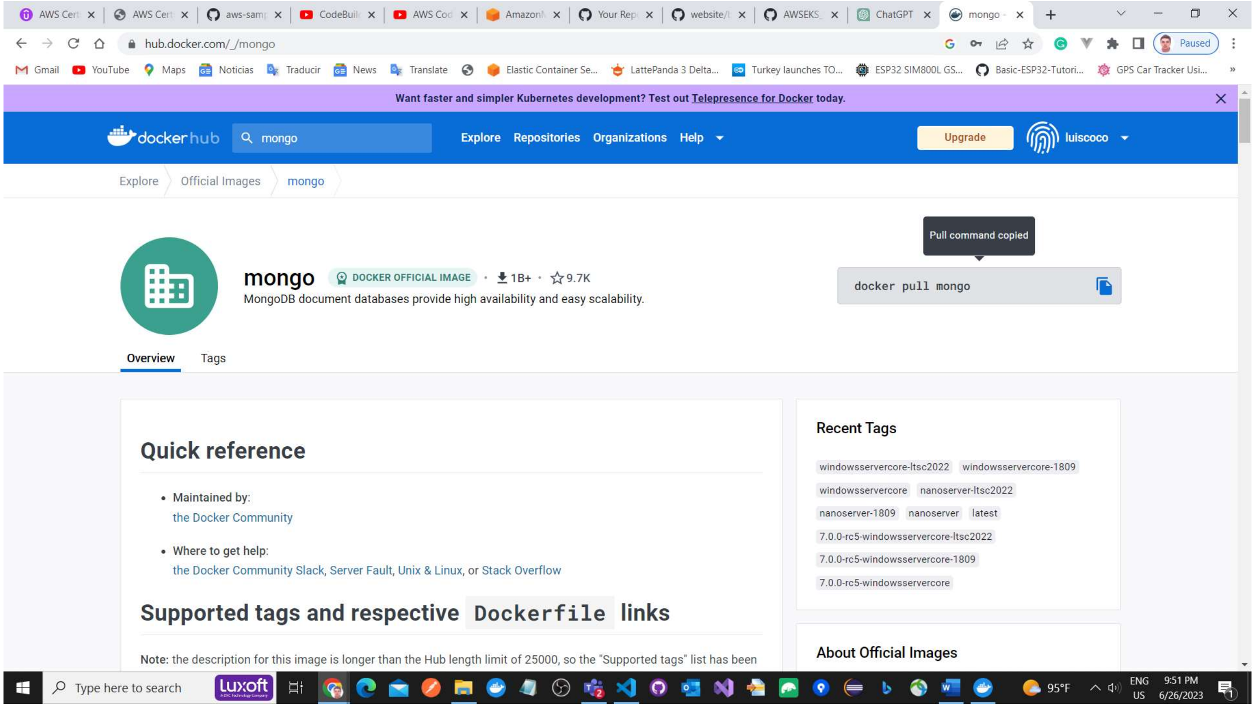
```
kubectl delete namespace namespacename
```

```
kubectl delete service mongodb-service -n dev
```

```
kubectl delete pod mongod-0 -n dev
```

```
kubectl delete statefulset mongod-0 -n dev
```

14. Now we are going to deploy our applications. First, we are going to deploy the MongoDbReplicaSet.



mongo DOCKER OFFICIAL IMAGE • 1B+ • 9.7K

MongoDB document databases provide high availability and easy scalability.

```
docker pull mongo
```

Quick reference

- Maintained by:
  the Docker Community
- Where to get help:
  the Docker Community Slack, Server Fault, Unix & Linux, or Stack Overflow

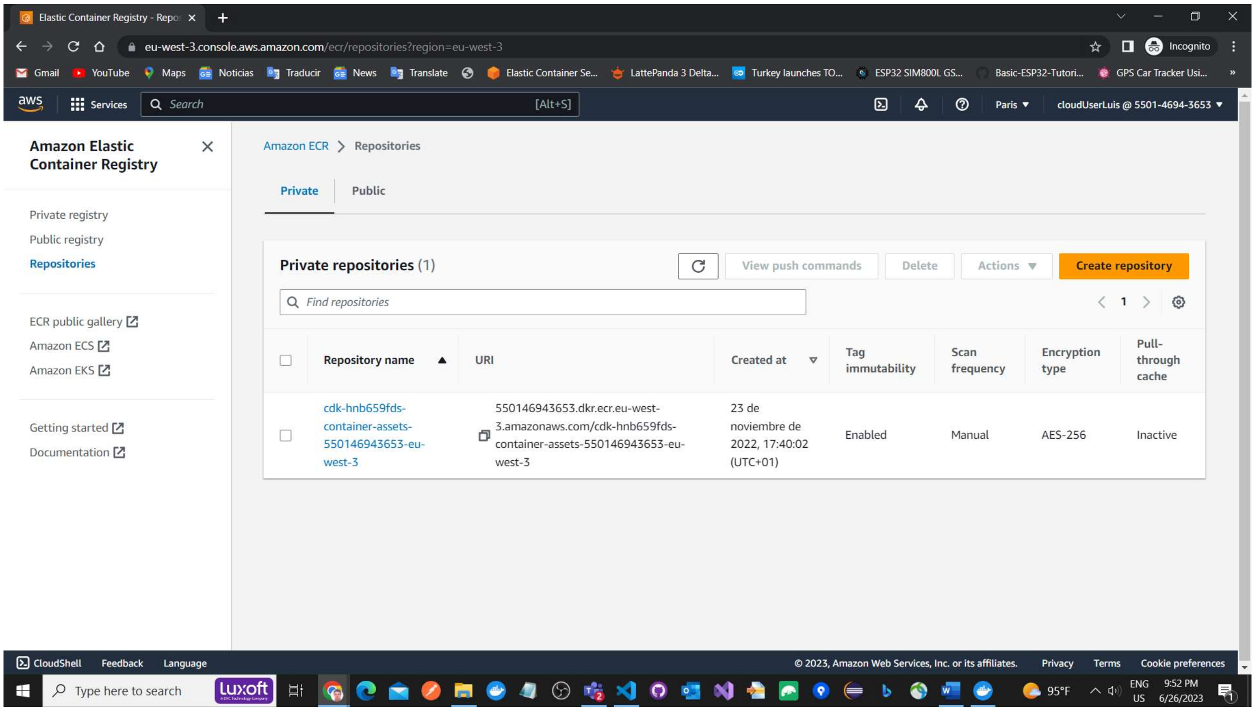
Supported tags and respective `Dockerfile` links

Note: the description for this image is longer than the Hub length limit of 25000, so the "Supported tags" list has been

Recent Tags

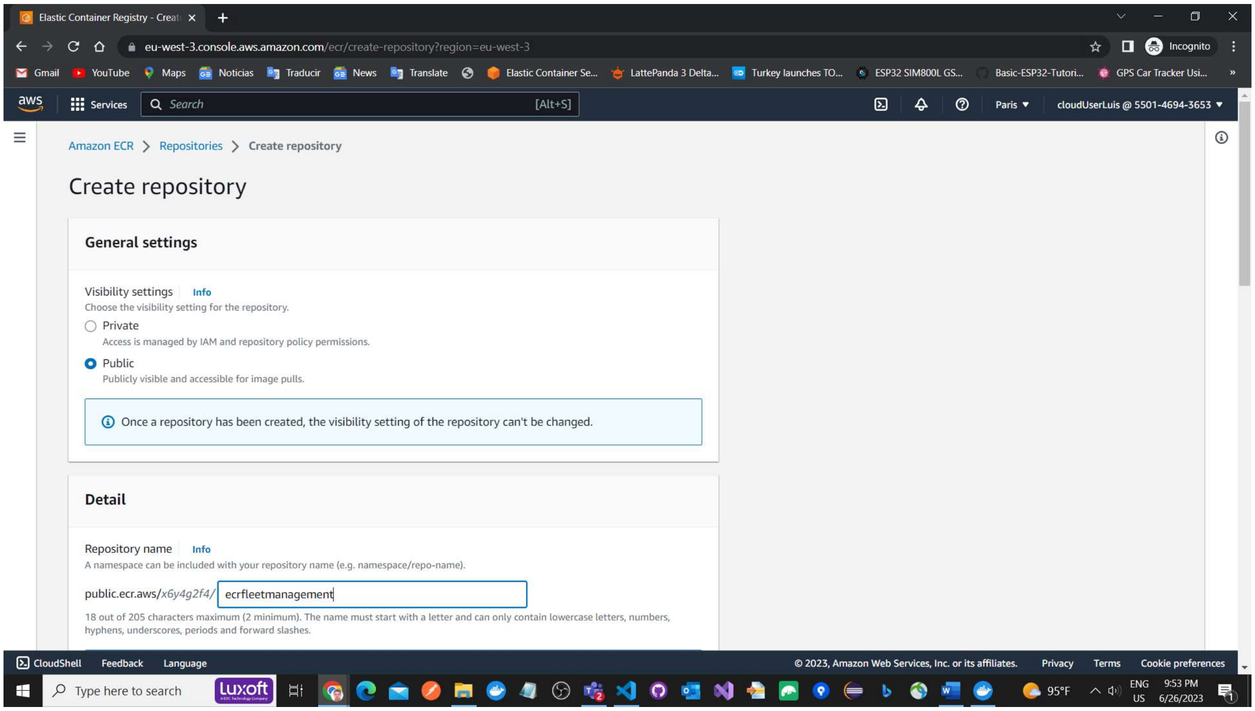
windowsservercore-ltsc2022 windowsservercore-1809 windowsservercore nanoserver-ltsc2022 nanoserver-1809 nanoserver latest 7.0.0-rc5-windowsservercore-ltsc2022 7.0.0-rc5-windowsservercore-1809 7.0.0-rc5-windowsservercore

About Official Images



Private repositories (1)

Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type	Pull-through cache
cdk-hnb659fds-container-assets-550146943653-eu-west-3	550146943653.dkr.ecr.eu-west-3.amazonaws.com/cdk-hnb659fds-container-assets-550146943653-eu-west-3	23 de noviembre de 2022, 17:40:02 (UTC+01)	Enabled	Manual	AES-256	Inactive



Create repository

General settings

Visibility settings Info

Choose the visibility setting for the repository.

○ Private

Access is managed by IAM and repository policy permissions.

● Public

Publicly visible and accessible for image pulls.

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

Detail

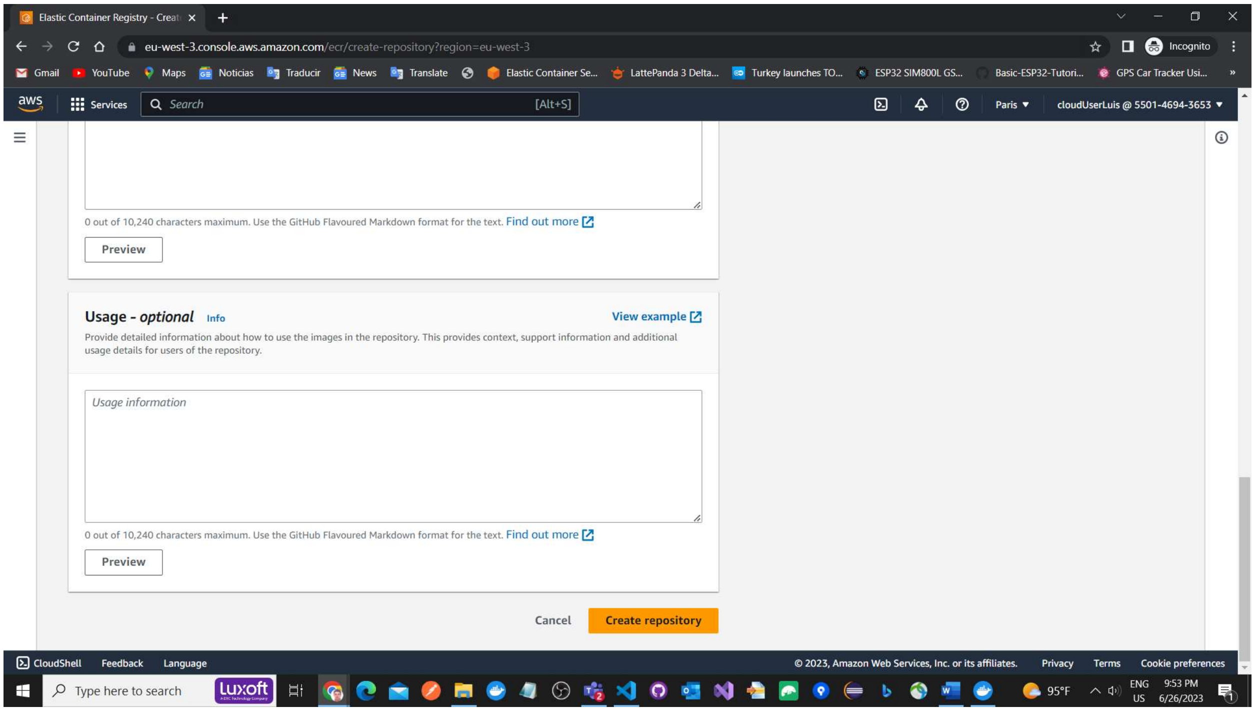
Repository name Info

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

public.ecr.aws/*x6y4g2f4*/ ecrfleetmanagement

18 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.

CloudShell
Feedback
Language
© 2023, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Type here to search
luxoft
95°F
ENG
US
9:53 PM
6/26/2023


Elastic Container Registry - Creat
eu-west-3.console.aws.amazon.com/ecr/create-repository?region=eu-west-3
Incognito
aws
Services
Search
[Alt+S]
Paris
cloudUserLuis @ 5501-4694-3653

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

Preview

Usage - *optional* Info

View example

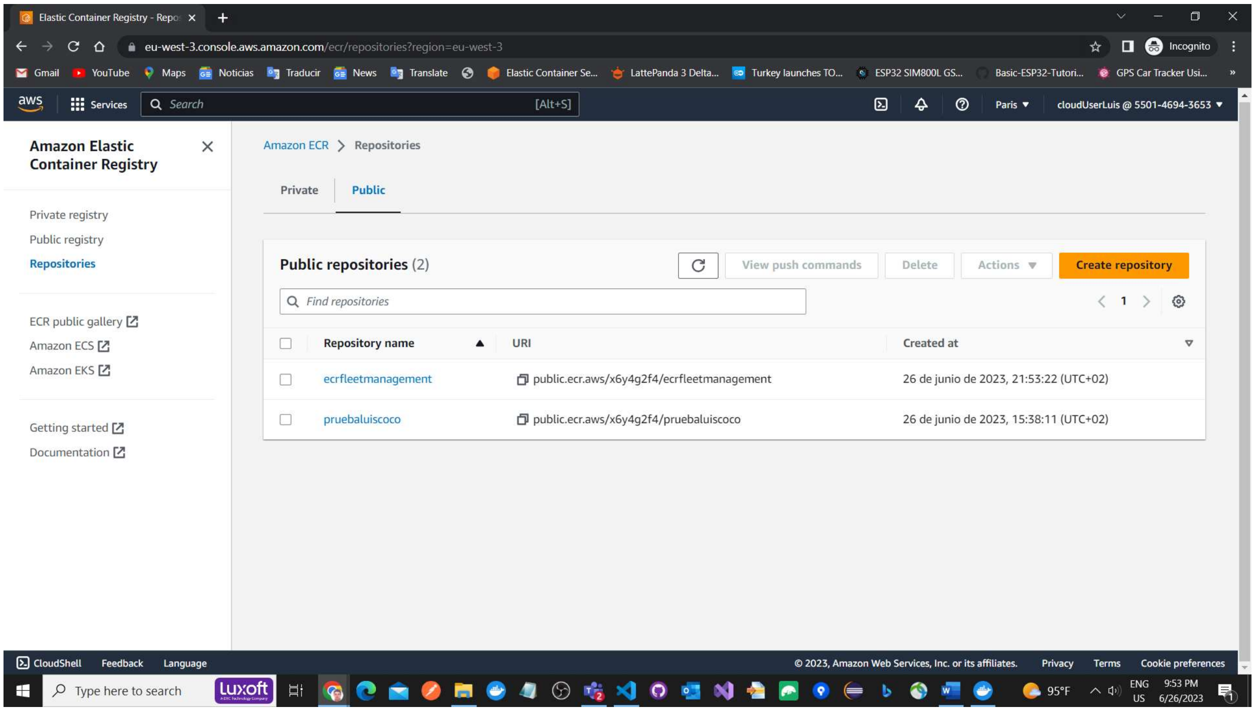
Provide detailed information about how to use the images in the repository. This provides context, support information and additional usage details for users of the repository.

Usage information

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

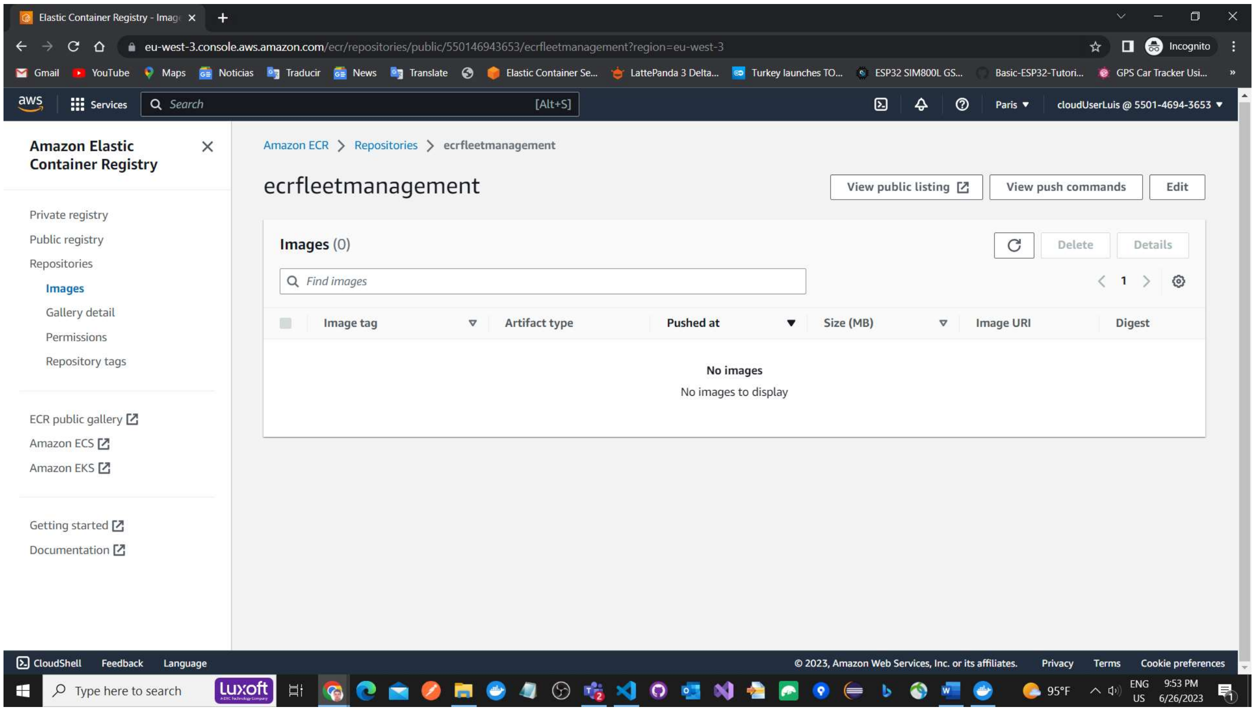
Preview

Cancel Create repository



Public repositories (2)

□	Repository name	URI	Created at
□	ecrfleetmanagement	public.ecr.aws/x6y4g2f4/ecrfleetmanagement	26 de junio de 2023, 21:53:22 (UTC+02)
□	pruebaluiscoco	public.ecr.aws/x6y4g2f4/pruebaluiscoco	26 de junio de 2023, 15:38:11 (UTC+02)



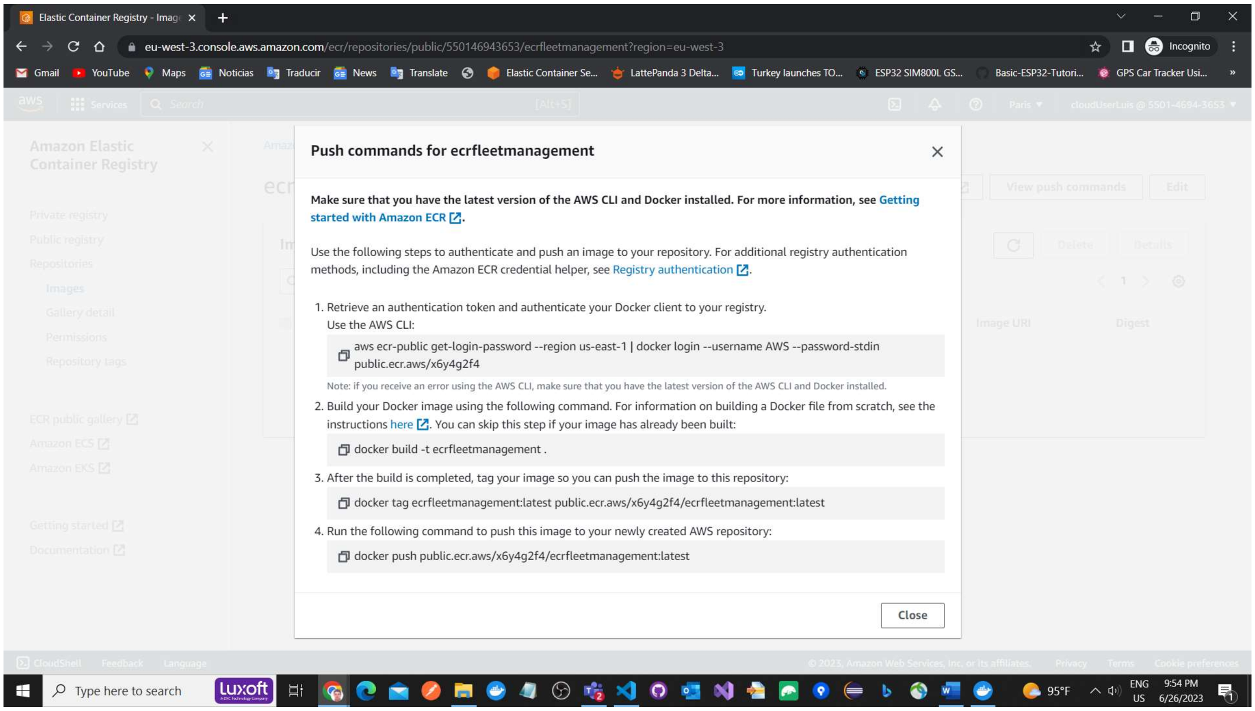
ecrfleetmanagement

Images (0)

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

No images

No images to display


Elastic Container Registry - Image
eu-west-3.console.aws.amazon.com/ecr/repositories/public/550146943653/ecrfleetmanagement?region=eu-west-3
View push commands
Edit
Close

Push commands for ecrfleetmanagement

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/x6y4g2f4
```

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 has already been built:

```
docker build -t ecrfleetmanagement .
```

3. After the build is completed, tag your image so you can push the image to this repository:

```
docker tag ecrfleetmanagement:latest public.ecr.aws/x6y4g2f4/ecrfleetmanagement:latest
```

4. Run the following command to push this image to your newly created AWS repository:

```
docker push public.ecr.aws/x6y4g2f4/ecrfleetmanagement:latest
```

Push commands for ecrfleetmanagement

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/x6y4g2f4

   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. After the build is completed, tag your image so you can push the image to this repository:
   docker tag mongo:latest public.ecr.aws/x6y4g2f4/mongo:latest

3. Run the following command to push this image to your newly created AWS repository:
   docker push public.ecr.aws/x6y4g2f4/mongo:latest

IMPORTANT!!! Copy the image URL from ECR to the ymal file in the image field.

IMPORTANT!!! Remove the Persistant Volume Claim from the **mongodb.yaml** file:

```
...
  #         volumeMounts:
  #           - name: mongodb-persistent-storage-claim
  #             mountPath: /data/db
  # volumeClaimTemplates:
  # - metadata:
  #     name: mongodb-persistent-storage-claim
  #   spec:
  #     accessModes: [ "ReadWriteOnce" ]
  #     resources:
  #       requests:
  #         storage: 1Gi
```

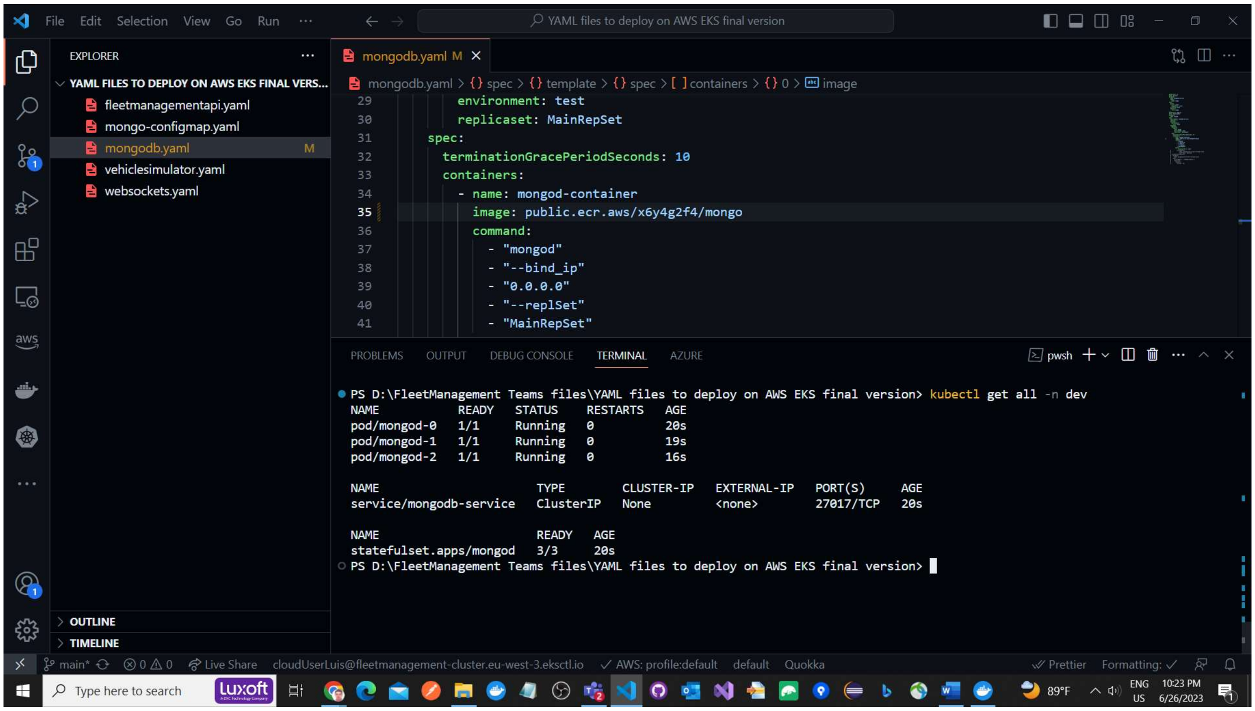
This is the code in the **mongodb.yaml** file:

```
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
  labels:
    name: mongo
spec:
  ports:
  - port: 27017
    targetPort: 27017
  clusterIP: None
  selector:
    role: mongo
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mongod
spec:
  serviceName: mongodb-service
  replicas: 3
  selector:
    matchLabels:
      role: mongo
  template:
```

```
metadata:
  labels:
    role: mongo
    environment: test
    replicaset: MainRepSet
spec:
  terminationGracePeriodSeconds: 10
  containers:
    - name: mongod-container
      image: 719220092744.dkr.ecr.eu-west-3.amazonaws.com/mongodb:latest
      command:
        - "mongod"
        - "--bind_ip"
        - "0.0.0.0"
        - "--replSet"
        - "MainRepSet"
      ports:
        - containerPort: 27017
```

We deploy the mongodb.yaml file running the command

kubectl apply -f mongodb.yaml --namespace dev

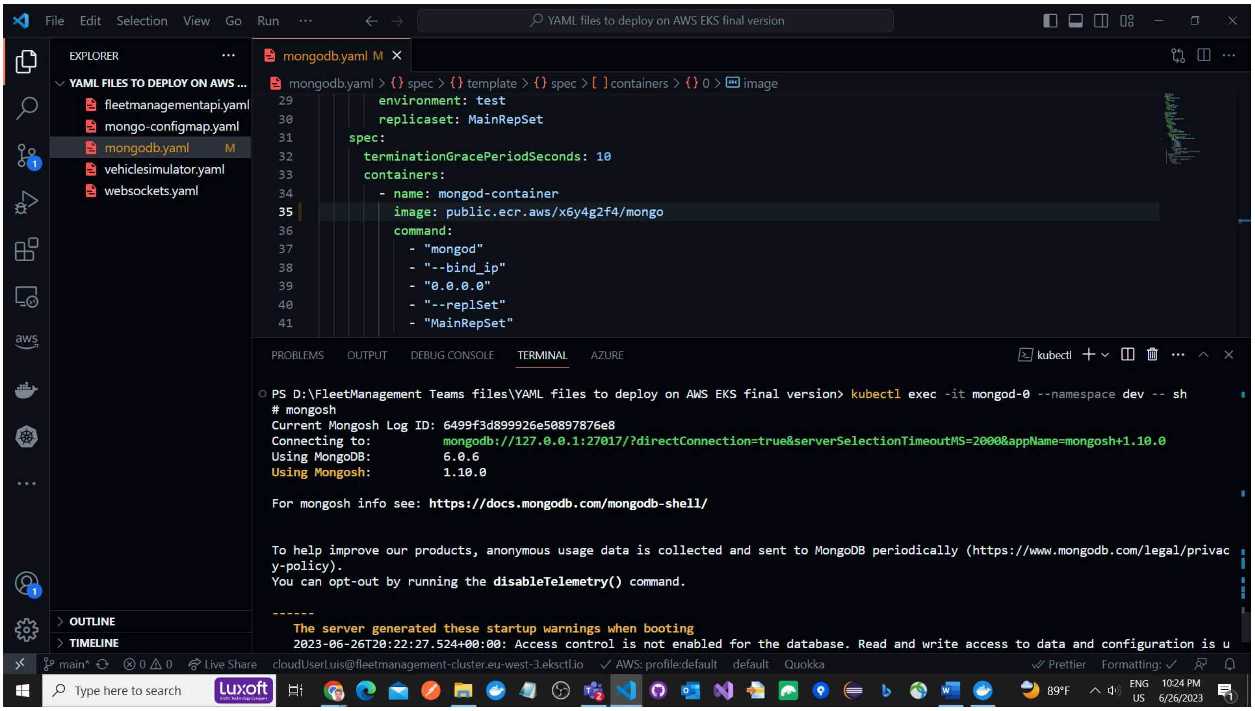

File Edit Selection View Go Run ···
YAML files to deploy on AWS EKS final version
EXPLORER
YAML FILES TO DEPLOY ON AWS EKS FINAL VERS...
fleetmanagementapi.yaml
mongo-configmap.yaml
mongodb.yaml M
vehiclesimulator.yaml
websockets.yaml
OUTLINE
TIMELINE
mongodb.yaml M
mongodb.yaml > {} spec > {} template > {} spec > [ ] containers > {} 0 > image
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL AZURE
pwsh
main* ⊗ 0 ⚠ 0 Live Share cloudUserLuis@fleetmanagement-cluster.eu-west-3.eksctl.io ✓ AWS: profile:default default Quokka
Prettier Formatting: ✓
Type here to search
89°F
ENG US
10:23 PM 6/26/2023

```
      environment: test
      replicaset: MainRepSet
  spec:
    terminationGracePeriodSeconds: 10
    containers:
      - name: mongod-container
        image: public.ecr.aws/x6y4g2f4/mongo
        command:
          - "mongod"
          - "--bind_ip"
          - "0.0.0.0"
          - "--replSet"
          - "MainRepSet"
```

```
PS D:\FleetManagement Teams files\YAML files to deploy on AWS EKS final version> kubectl get all -n dev
NAME             READY   STATUS    RESTARTS   AGE
pod/mongod-0     1/1     Running   0          20s
pod/mongod-1     1/1     Running   0          19s
pod/mongod-2     1/1     Running   0          16s

NAME                       TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)     AGE
service/mongodb-service    ClusterIP   None         <none>        27017/TCP   20s

NAME                       READY   AGE
statefulset.apps/mongod    3/3     20s
PS D:\FleetManagement Teams files\YAML files to deploy on AWS EKS final version>
```



```yaml
          environment: test
          replicaset: MainRepSet
      spec:
        terminationGracePeriodSeconds: 10
        containers:
          - name: mongod-container
            image: public.ecr.aws/x6y4g2f4/mongo
            command:
              - "mongod"
              - "--bind_ip"
              - "0.0.0.0"
              - "--replSet"
              - "MainRepSet"
```

```
PS D:\FleetManagement Teams files\YAML files to deploy on AWS EKS final version> kubectl exec -it mongod-0 --namespace dev -- sh
# mongosh
Current Mongosh Log ID: 6499f3d899926e50897876e8
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.10.0
Using MongoDB:          6.0.6
Using Mongosh:          1.10.0

For mongosh info see: https://docs.mongodb.com/mongodb-shell/


To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privac
y-policy).
You can opt-out by running the disableTelemetry() command.

------
   The server generated these startup warnings when booting
   2023-06-26T20:22:27.524+00:00: Access control is not enabled for the database. Read and write access to data and configuration is u
```

kubectl exec -it mongod-0 --namespace dev – sh

```
# mongosh
Current Mongosh Log ID: 6499f3d899926e50897876e8
Connecting to:
mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.10.0
Using MongoDB:          6.0.6
Using Mongosh:          1.10.0

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.
------
   The server generated these startup warnings when booting
   2023-06-26T20:22:27.524+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2023-06-26T20:22:27.525+00:00: You are running this process as the root user, which is not recommended
   2023-06-26T20:22:27.525+00:00: vm.max_map_count is too low
------

test>
```

15. Now we configure de ReplicaSet in MongoDb.

kubectl get pods --namespace dev

copy the mongodb pods names

kubectl exec -it mongod-0 --namespace dev -- sh

podName: mongodb-0

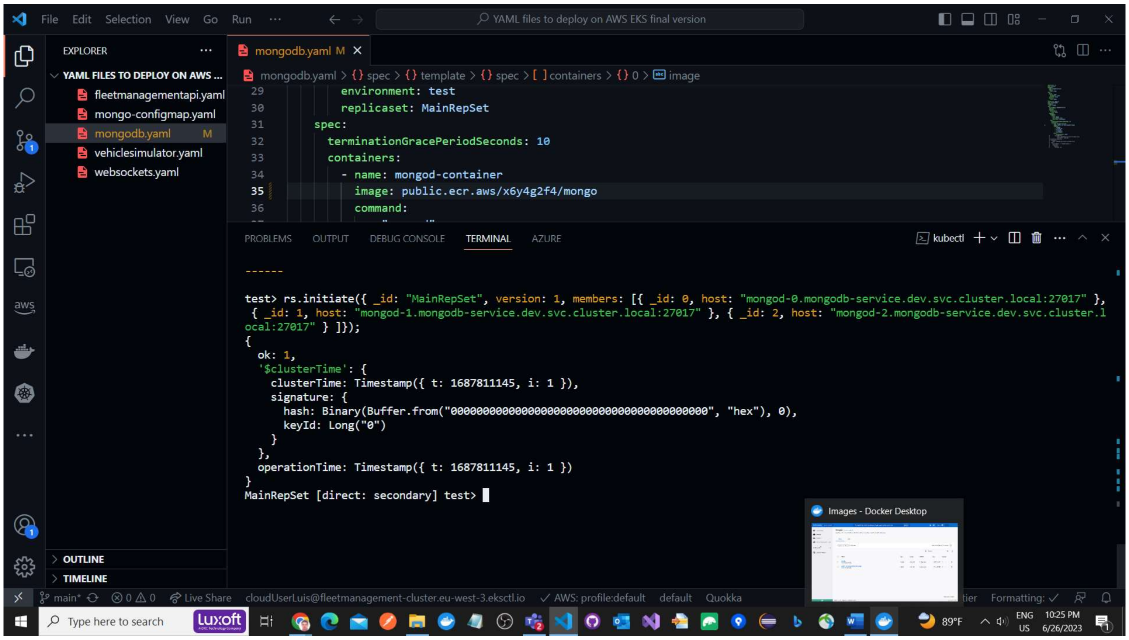
#**mongosh** or mongo

We have to set the namespace where we deployed the MongoDb.

In this example we set the "default" namespace:
>rs.initiate({ _id: "MainRepSet", version: 1, members: [{ _id: 0, host: "mongod-0.mongodb-service.default.svc.cluster.local:27017" }, { _id: 1, host: "mongod-1.mongodb-service.default.svc.cluster.local:27017" }, { _id: 2, host: "mongod-2.mongodb-service.default.svc.cluster.local:27017" }]});

Or

In this example we set the "dev" namespace
>**rs.initiate({ _id: "MainRepSet", version: 1, members: [{ _id: 0, host: "mongod-0.mongodb-service.dev.svc.cluster.local:27017" }, { _id: 1, host: "mongod-1.mongodb-service.dev.svc.cluster.local:27017" }, { _id: 2, host: "mongod-2.mongodb-service.dev.svc.cluster.local:27017" }]});**



```
        environment: test
        replicaset: MainRepSet
    spec:
      terminationGracePeriodSeconds: 10
      containers:
        - name: mongod-container
          image: public.ecr.aws/x6y4g2f4/mongo
          command:
```

```
------

test> rs.initiate({ _id: "MainRepSet", version: 1, members: [{ _id: 0, host: "mongod-0.mongodb-service.dev.svc.cluster.local:27017" },
 { _id: 1, host: "mongod-1.mongodb-service.dev.svc.cluster.local:27017" }, { _id: 2, host: "mongod-2.mongodb-service.dev.svc.cluster.l
ocal:27017" } ]});
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1687811145, i: 1 }),
    signature: {
      hash: Binary(Buffer.from("0000000000000000000000000000000000000000", "hex"), 0),
      keyId: Long("0")
    }
  },
  operationTime: Timestamp({ t: 1687811145, i: 1 })
}
MainRepSet [direct: secondary] test>
```

Run the status command to check the ReplicaSet was created:

>**rs.status();**

```
MainRepSet [direct: secondary] test> rs.status();
{
  set: 'MainRepSet',
  date: ISODate("2023-06-26T20:26:25.657Z"),
  myState: 1,
  term: Long("1"),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long("2000"),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1687811176, i: 1 }), t: Long("1") },
    lastCommittedWallTime: ISODate("2023-06-26T20:26:16.332Z"),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1687811176, i: 1 }), t: Long("1") },
    appliedOpTime: { ts: Timestamp({ t: 1687811176, i: 1 }), t: Long("1") },
    durableOpTime: { ts: Timestamp({ t: 1687811176, i: 1 }), t: Long("1") },
    lastAppliedWallTime: ISODate("2023-06-26T20:26:16.332Z"),
    lastDurableWallTime: ISODate("2023-06-26T20:26:16.332Z")
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1687811145, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate("2023-06-26T20:25:56.246Z"),
    electionTerm: Long("1"),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1687811145, i: 1 }), t: Long("-1") },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1687811145, i: 1 }), t: Long("-1") },
    numVotesNeeded: 2,
    priorityAtElection: 1,
    electionTimeoutMillis: Long("10000"),
    numCatchUpOps: Long("0"),
    newTermStartDate: ISODate("2023-06-26T20:25:56.313Z"),
    wMajorityWriteAvailabilityDate: ISODate("2023-06-26T20:25:56.791Z")
  },
  members: [
    {
      _id: 0,
      name: 'mongod-0.mongodb-service.dev.svc.cluster.local:27017',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 239,
      optime: { ts: Timestamp({ t: 1687811176, i: 1 }), t: Long("1") },
      optimeDate: ISODate("2023-06-26T20:26:16.000Z"),
      lastAppliedWallTime: ISODate("2023-06-26T20:26:16.332Z"),
      lastDurableWallTime: ISODate("2023-06-26T20:26:16.332Z"),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1687811156, i: 1 }),
      electionDate: ISODate("2023-06-26T20:25:56.000Z"),
      configVersion: 1,
      configTerm: 1,
      self: true,
      lastHeartbeatMessage: ''
    },
    {
      _id: 1,
      name: 'mongod-1.mongodb-service.dev.svc.cluster.local:27017',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 40,
      optime: { ts: Timestamp({ t: 1687811176, i: 1 }), t: Long("1") },
      optimeDurable: { ts: Timestamp({ t: 1687811176, i: 1 }), t: Long("1") },
      optimeDate: ISODate("2023-06-26T20:26:16.000Z"),
      optimeDurableDate: ISODate("2023-06-26T20:26:16.000Z"),
      lastAppliedWallTime: ISODate("2023-06-26T20:26:16.332Z"),
      lastDurableWallTime: ISODate("2023-06-26T20:26:16.332Z"),
      lastHeartbeat: ISODate("2023-06-26T20:26:24.296Z"),
      lastHeartbeatRecv: ISODate("2023-06-26T20:26:24.803Z"),
      pingMs: Long("1"),
      lastHeartbeatMessage: '',
      syncSourceHost: 'mongod-0.mongodb-service.dev.svc.cluster.local:27017',
      syncSourceId: 0,
      infoMessage: '',
      configVersion: 1,
      configTerm: 1
    },
    {
      _id: 2,
      name: 'mongod-2.mongodb-service.dev.svc.cluster.local:27017',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 40,
      optime: { ts: Timestamp({ t: 1687811176, i: 1 }), t: Long("1") },
      optimeDurable: { ts: Timestamp({ t: 1687811176, i: 1 }), t: Long("1") },
      optimeDate: ISODate("2023-06-26T20:26:16.000Z"),
      optimeDurableDate: ISODate("2023-06-26T20:26:16.000Z"),
      lastAppliedWallTime: ISODate("2023-06-26T20:26:16.332Z"),
      lastDurableWallTime: ISODate("2023-06-26T20:26:16.332Z"),
      lastHeartbeat: ISODate("2023-06-26T20:26:24.291Z"),
      lastHeartbeatRecv: ISODate("2023-06-26T20:26:23.790Z"),
      pingMs: Long("1"),
      lastHeartbeatMessage: '',
      syncSourceHost: 'mongod-1.mongodb-service.dev.svc.cluster.local:27017',
      syncSourceId: 1,
      infoMessage: '',
      configVersion: 1,
      configTerm: 1
    }
  ],
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1687811176, i: 1 }),
    signature: {
      hash: Binary(Buffer.from("0000000000000000000000000000000000000000", "hex"), 0),
      keyId: Long("0")
    }
  },
  operationTime: Timestamp({ t: 1687811176, i: 1 })
}
MainRepSet [direct: primary] test>
```

IMPORTANT NOTE: Copy the host names to set the connection string in “mongo-configmap.yaml” file, see section

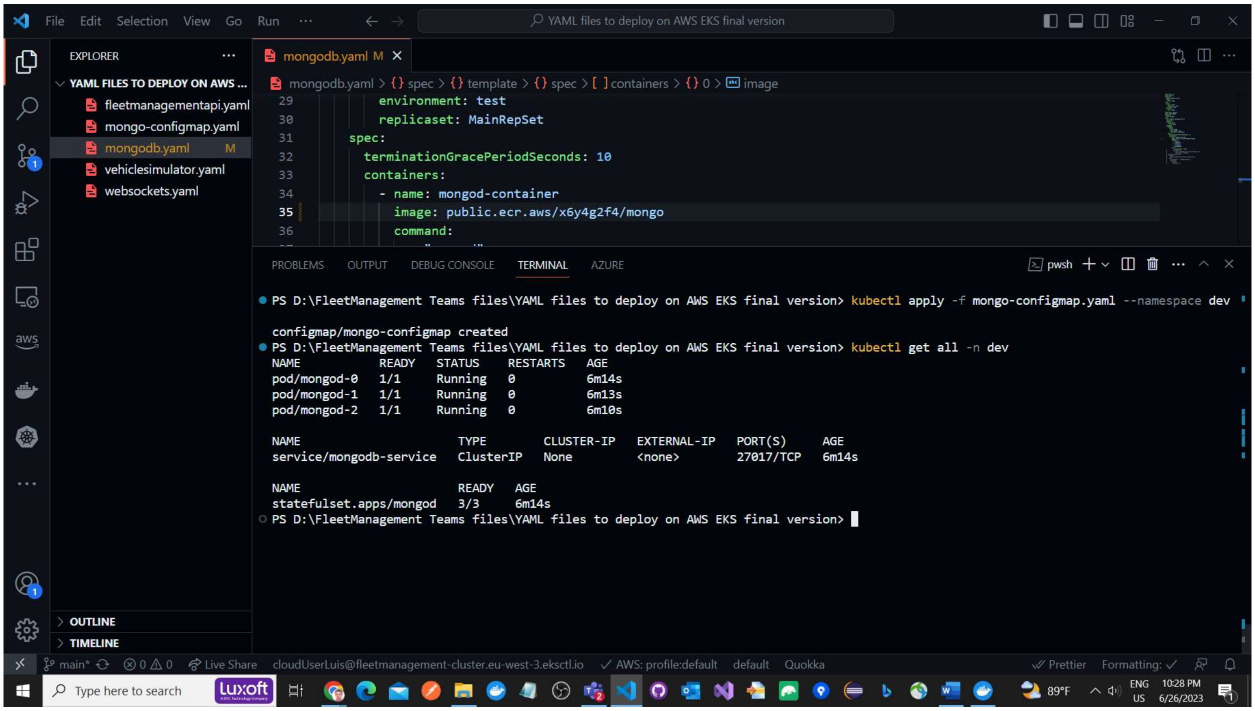
>**exit**
#**exit**
cls

15. This is the code for the **mongo-configmap.yaml**:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mongo-configmap
data:
connection_string: mongodb://mongod-0.mongodb-service.dev.svc.cluster.local:27017,mongod-1.mongodb-service.dev.svc.cluster.local:27017,mongod-2.mongodb-service.dev.svc.cluster.local:27017
```

Run the command

```
kubectl apply -f mongo-configmap.yaml --namespace dev
```


File Edit Selection View Go Run
YAML files to deploy on AWS EKS final version
EXPLORER
YAML FILES TO DEPLOY ON AWS ...
fleetmanagementapi.yaml
mongo-configmap.yaml
mongodb.yaml M
vehiclesimulator.yaml
websockets.yaml
OUTLINE
TIMELINE
mongodb.yaml M
mongodb.yaml > {} spec > {} template > {} spec > [ ] containers > {} 0 > image
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL AZURE
pwsh
main* Live Share cloudUserLuis@fleetmanagement-cluster.eu-west-3.eksctl.io AWS: profile:default default Quokka Prettier Formatting: ✓
Type here to search
89°F ENG US 10:28 PM 6/26/2023

```
        environment: test
        replicaset: MainRepSet
    spec:
      terminationGracePeriodSeconds: 10
      containers:
        - name: mongod-container
          image: public.ecr.aws/x6y4g2f4/mongo
          command:
```

```
PS D:\FleetManagement Teams files\YAML files to deploy on AWS EKS final version> kubectl apply -f mongo-configmap.yaml --namespace dev

configmap/mongo-configmap created
PS D:\FleetManagement Teams files\YAML files to deploy on AWS EKS final version> kubectl get all -n dev
NAME           READY   STATUS    RESTARTS   AGE
pod/mongod-0   1/1     Running   0          6m14s
pod/mongod-1   1/1     Running   0          6m13s
pod/mongod-2   1/1     Running   0          6m10s

NAME                     TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)     AGE
service/mongodb-service  ClusterIP   None         <none>        27017/TCP   6m14s

NAME                     READY   AGE
statefulset.apps/mongod  3/3     6m14s
PS D:\FleetManagement Teams files\YAML files to deploy on AWS EKS final version>
```