Task Overview: You are tasked with deploying a Docker-based Node.js application that interacts with a MongoDB database on AWS. This will involve setting up the containers, managing them through AWS ECS or EKS, and ensuring optimal configuration for scalability and security.

Key Components of the Task:

1. **Docker Configuration:** Prepare Dockerfiles and use Docker Compose to manage multi-container setups.
2. **AWS Deployment:** Utilize AWS services such as ECS/EKS, ECR, ELB, and Auto Scaling.
3. **CI/CD Pipeline:** Implement a CI/CD pipeline using tools like Jenkins or AWS CodePipeline.
4. **Documentation:** Provide clear documentation detailing the deployment process and your CI/CD setup.
5. **Optimal Deployment Discussion:** Include a report discussing your choices of AWS configurations and services.

GitHub Repo: https://github.com/kumarsuresh03/Task.git

Step1: Set Up Docker Configuration

- **Prepare Dockerfiles for Node.js and MangoDB**
- **Node.js Dockerfile(Dockerfile)**

```
“”  FROM node:14

WORKDIR /usr/src/app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["npm", "start"]  “”
```

- **MongoDB Dockerfile**

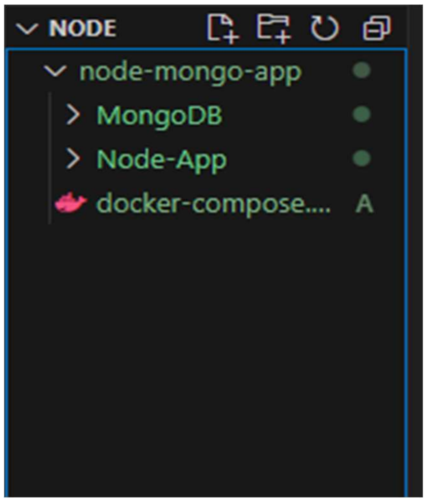
```
“”

FROM mongo:latest

EXPOSE 27017 “”
```

- **Docker-compose.yml**

```
“”version: '3.8'
```

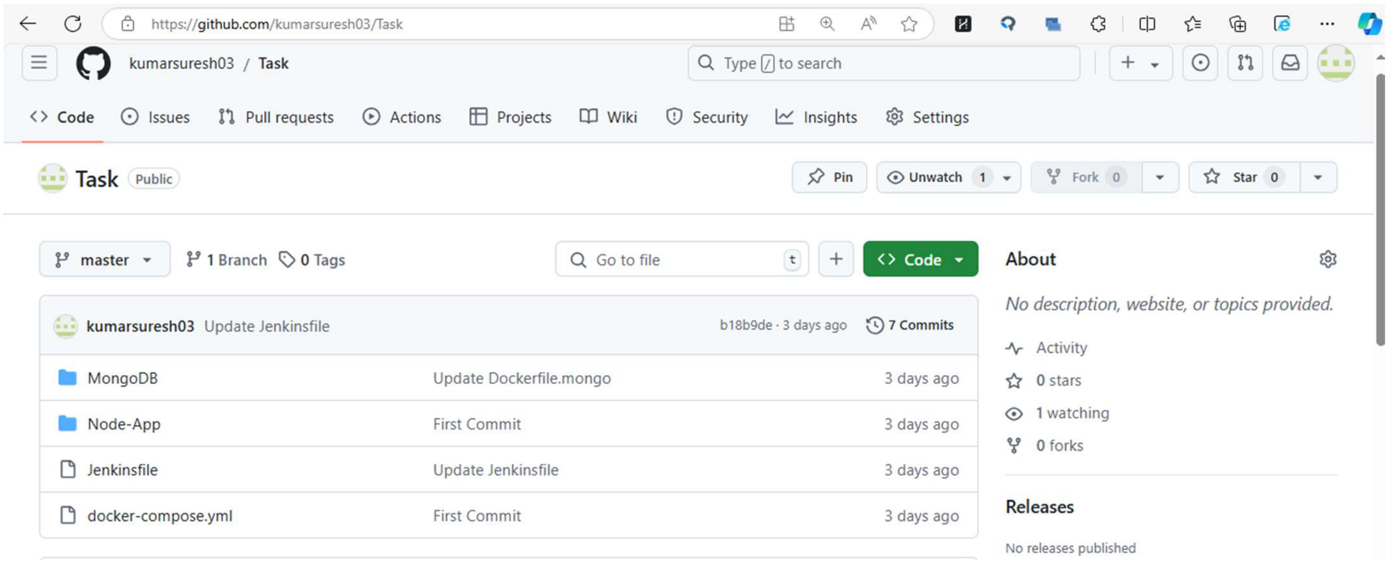
```
services:
  app:
   build:
      context: .
    ports:
      - "3000:3000"
    depends_on:
      - mongo
  mongo:
    image: mongo:latest
    ports:
      - "27017:27017"
    volumes:
      - mongo-data:/data/db  # Persist MongoDB data
volumes:
  mongo-data: “”
```

- **This is the format I have Created File for Pushing the Github repo**



Step2: CI/CD Pipeline with Jenkins

- **After pushing the file into GitHub Repo and add the 'Jenkinsfile' we need to create the pipeline**

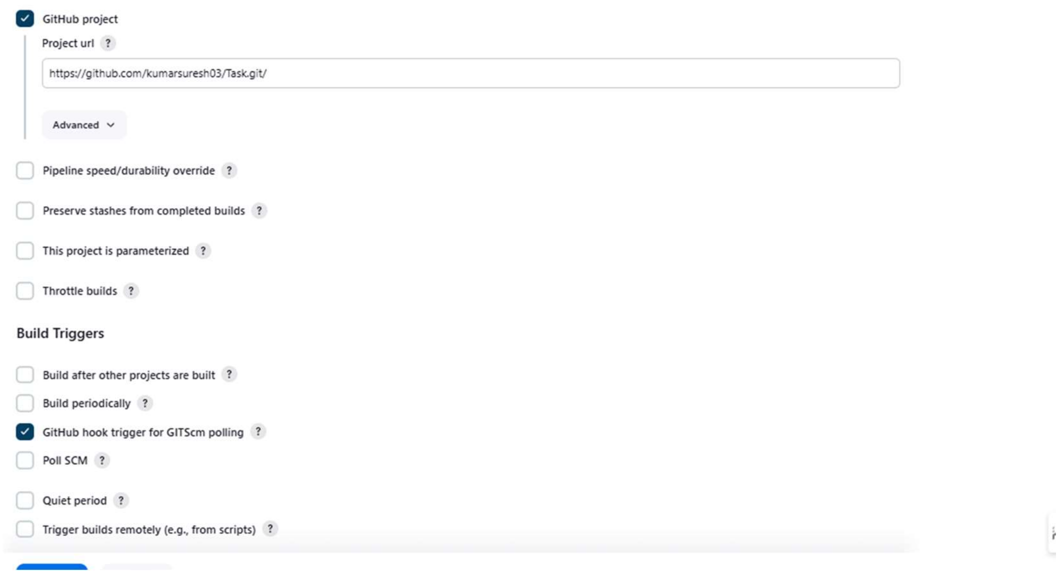


kumarsuresh03 Update Jenkinsfile b18b9de · 3 days ago 7 Commits

MongoDB	Update Dockerfile.mongo	3 days ago
Node-App	First Commit	3 days ago
Jenkinsfile	Update Jenkinsfile	3 days ago
docker-compose.yml	First Commit	3 days ago

No description, website, or topics provided.

- **To create a pipeline job in Jenkins with the name "Task" and configure it to run the pipeline defined in the Jenkinsfile**
- **Configure the Pipeline Job**



- [x] GitHub project

Project url ?

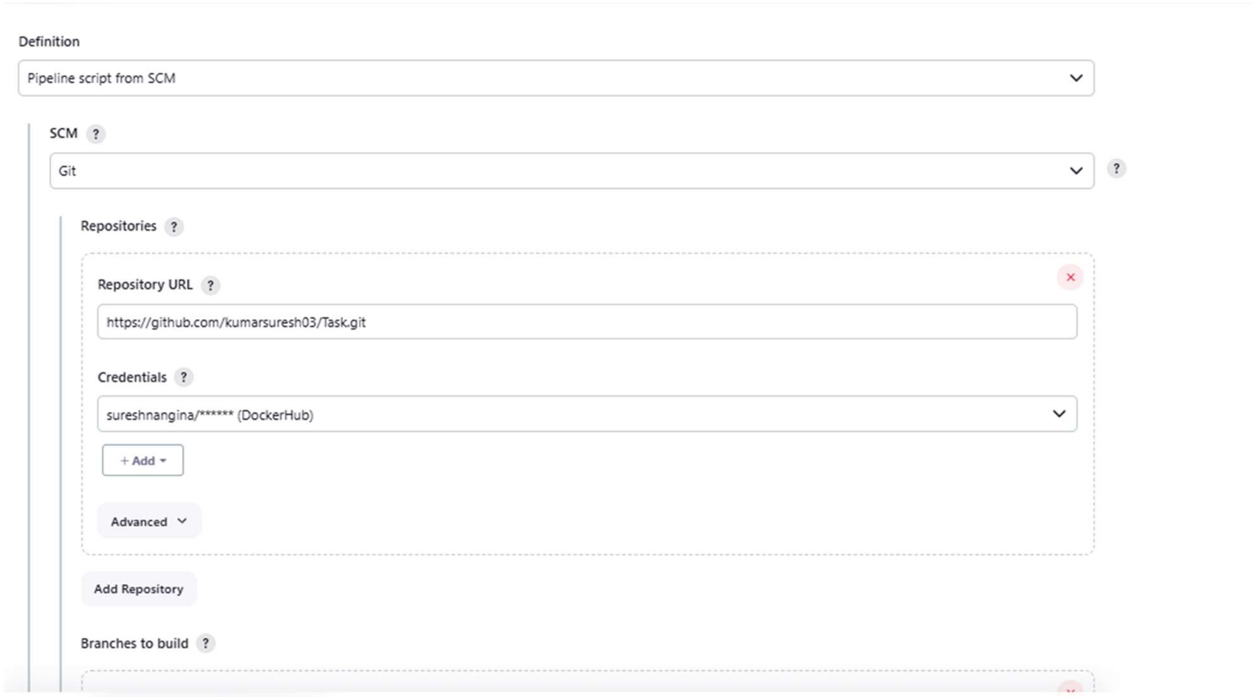
https://github.com/kumarsuresh03/Task.git/

- [ ] Pipeline speed/durability override ?
- [ ] Preserve stashes from completed builds ?
- [ ] This project is parameterized ?
- [ ] Throttle builds ?

Build Triggers

- [ ] Build after other projects are built ?
- [ ] Build periodically ?
- [x] GitHub hook trigger for GITScm polling ?
- [ ] Poll SCM ?
- [ ] Quiet period ?
- [ ] Trigger builds remotely (e.g., from scripts) ?

Credentials: Choose the credentials that have access to your repository (if needed).



Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

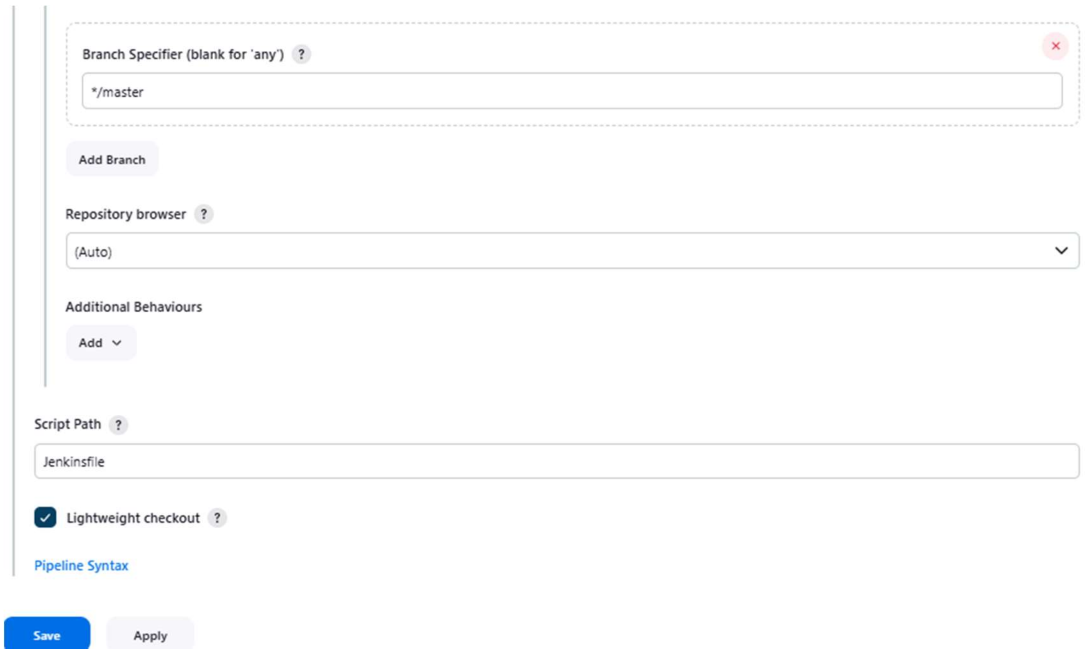
https://github.com/kumarsuresh03/Task.git

Credentials

sureshnangina/****** (DockerHub)

Branches to build

Script Path: Jenkinsfile is located in the root of your repository, leave it as Jenkinsfile.



Branch Specifier (blank for 'any')

*/master

Repository browser

(Auto)

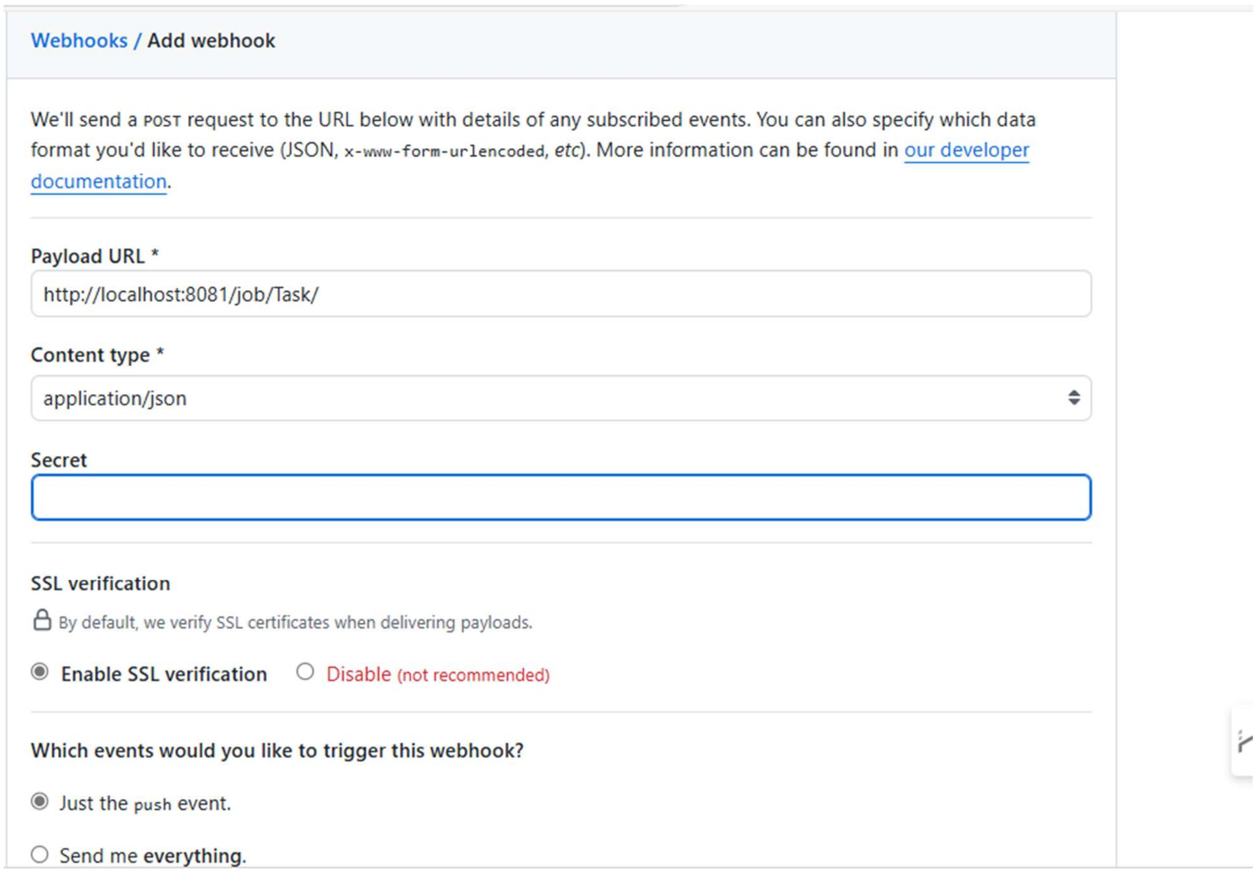
Additional Behaviours

Script Path

Jenkinsfile

✓ Lightweight checkout

Configure Webhook in Your Repository



Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, *etc*). More information can be found in our developer documentation.

Payload URL *
http://localhost:8081/job/Task/

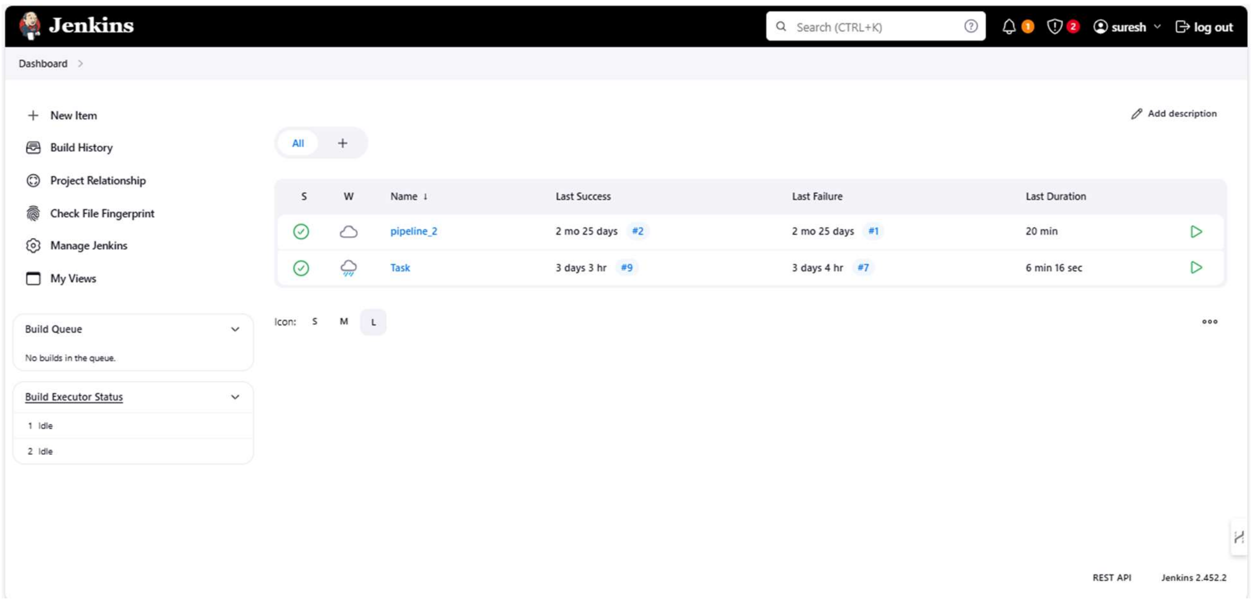
Content type *
application/json

Secret

SSL verification
By default, we verify SSL certificates when delivering payloads.
[✓ Enable SSL verification] Disable (not recommended)

Which events would you like to trigger this webhook?
[✓ Just the `push` event.]
Send me **everything**.

Successfully created pipeline job and configure it for execution in Jenkins



Dashboard >

- New Item
- Build History
- Project Relationship
- Check File Fingerprint
- Manage Jenkins
- My Views

Build Queue
No builds in the queue.

Build Executor Status
1 Idle
2 Idle

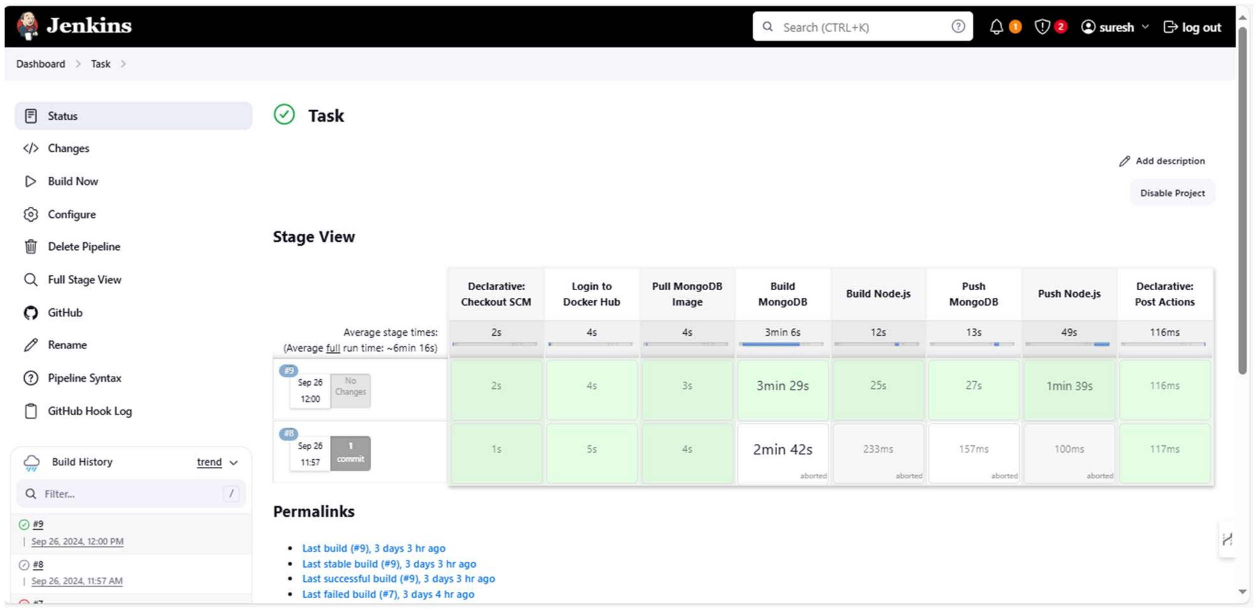
Add description

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓		pipeline_2	2 mo 25 days #2	2 mo 25 days #1	20 min
✓		Task	3 days 3 hr #9	3 days 4 hr #7	6 min 16 sec

Icon: S M L

REST API Jenkins 2.452.2



Dashboard > Task >

- Status
- Changes
- Build Now
- Configure
- Delete Pipeline
- Full Stage View
- GitHub
- Rename
- Pipeline Syntax
- GitHub Hook Log

Build History trend
Filter...
#9 Sep 26, 2024, 12:00 PM
#8 Sep 26, 2024, 11:57 AM

✓ Task

Add description
Disable Project

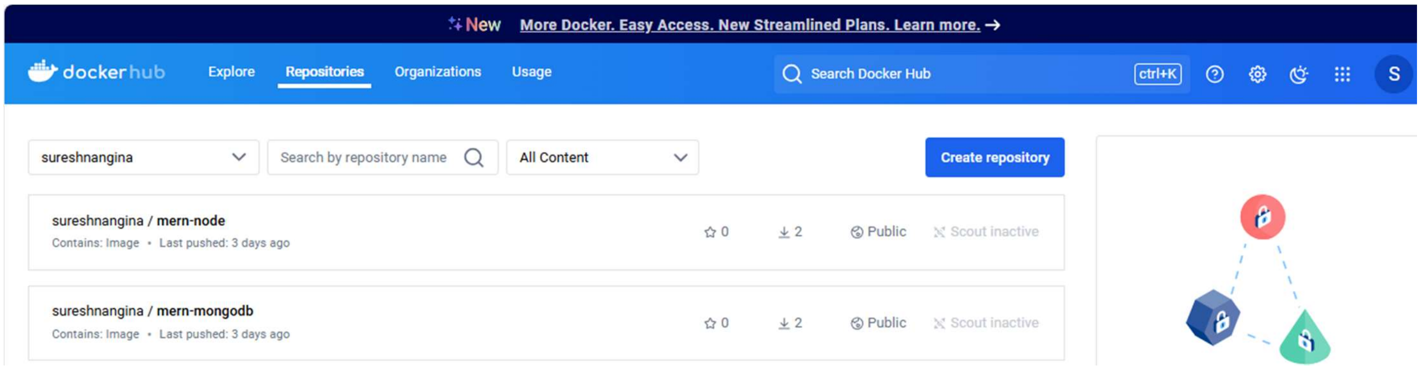
Stage View

	Declarative: Checkout SCM	Login to Docker Hub	Pull MongoDB Image	Build MongoDB	Build Node.js	Push MongoDB	Push Node.js	Declarative: Post Actions
Average stage times: (Average full run time: ~6min 16s)	2s	4s	4s	3min 6s	12s	13s	49s	116ms
#9 Sep 26 12:00 No Changes	2s	4s	3s	3min 29s	25s	27s	1min 39s	116ms
#8 Sep 26 11:57 1 commit	1s	5s	4s	2min 42s aborted	233ms aborted	157ms aborted	100ms aborted	117ms

Permalinks

- Last build (#9), 3 days 3 hr ago
- Last stable build (#9), 3 days 3 hr ago
- Last successful build (#9), 3 days 3 hr ago
- Last failed build (#7), 3 days 4 hr ago

Successfully build the Docker image and push into Docker Hub under the specified repository.



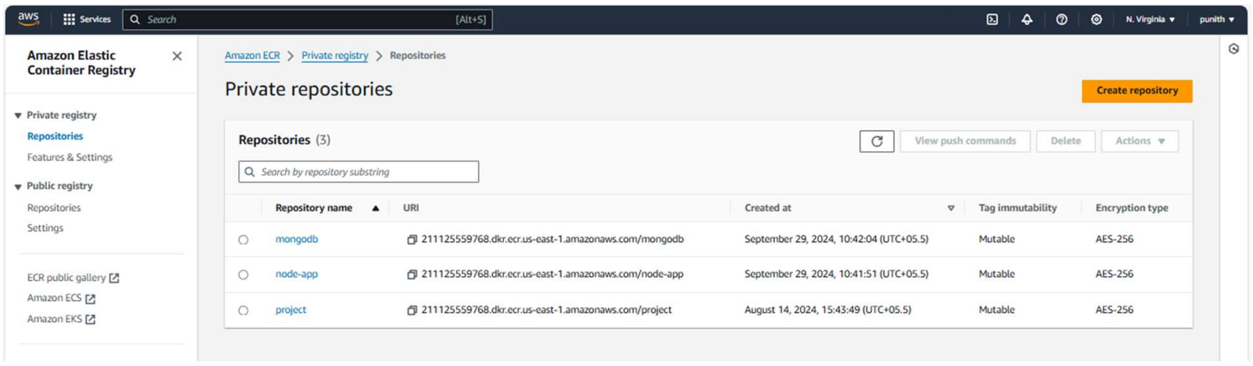
Repository		Stars	Pulls	Visibility	Scout
sureshnangina / mern-node	Contains: Image • Last pushed: 3 days ago	0	2	Public	Scout inactive
sureshnangina / mern-mongodb	Contains: Image • Last pushed: 3 days ago	0	2	Public	Scout inactive

Step 3: AWS Deployment

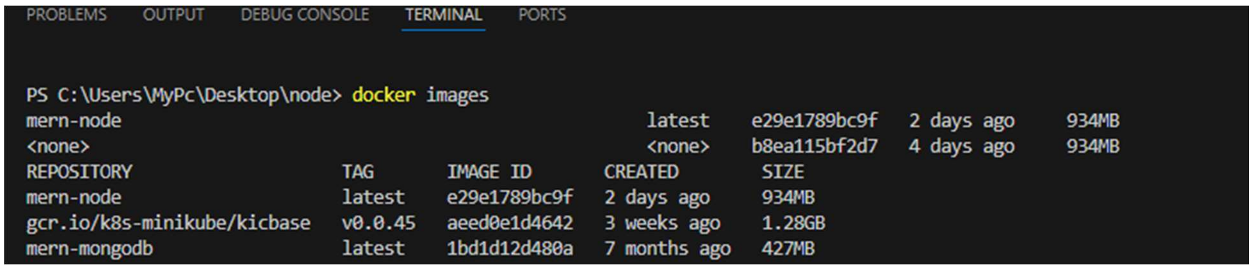
Push Docker Images to Amazon ECR

Create ECR Repositories:

- Go to AWS ECR and create repositories for both the Node.js app and MongoDB.



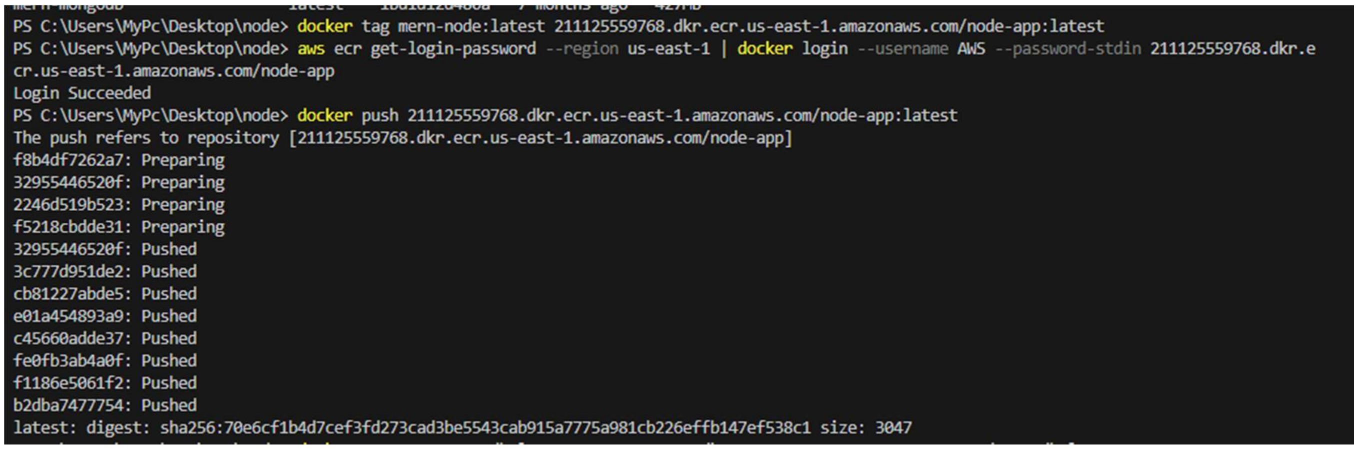
Repository name	URI	Created at	Tag immutability	Encryption type
mongodb	211125559768.dkr.ecr.us-east-1.amazonaws.com/mongodb	September 29, 2024, 10:42:04 (UTC+05.5)	Mutable	AES-256
node-app	211125559768.dkr.ecr.us-east-1.amazonaws.com/node-app	September 29, 2024, 10:41:51 (UTC+05.5)	Mutable	AES-256
project	211125559768.dkr.ecr.us-east-1.amazonaws.com/project	August 14, 2024, 15:43:49 (UTC+05.5)	Mutable	AES-256



```
PS C:\Users\MyPc\Desktop\node> docker images
mern-node                                         latest    e29e1789bc9f   2 days ago    934MB
<none>                                            <none>    b8ea115bf2d7   4 days ago    934MB
REPOSITORY                    TAG       IMAGE ID       CREATED        SIZE
mern-node                     latest    e29e1789bc9f   2 days ago     934MB
gcr.io/k8s-minikube/kicbase   v0.0.45   aeed0e1d4642   3 weeks ago    1.28GB
mern-mongodb                  latest    1bd1d12d480a   7 months ago   427MB
```

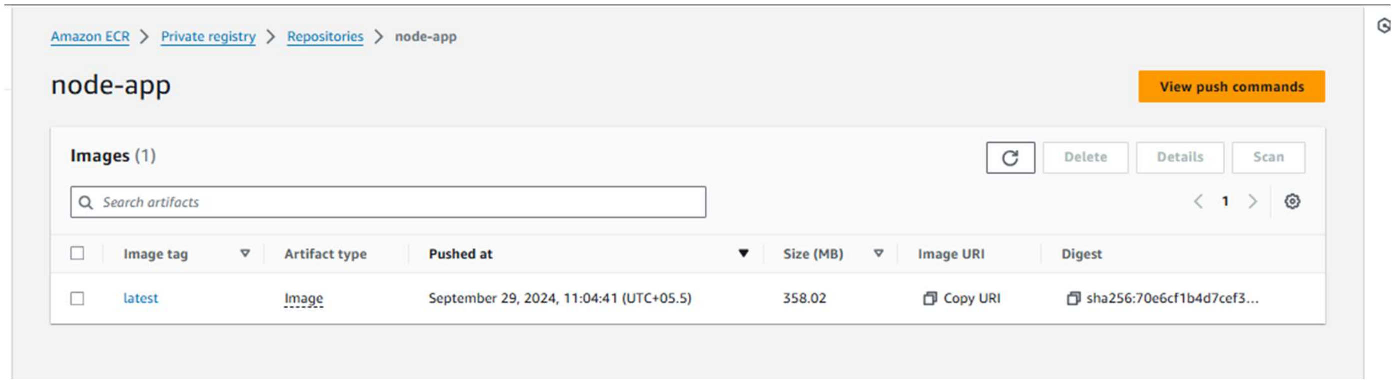
Firstly we need push the ‘mern-node’ image into ECR repo

1. **Tag the image:** docker tag mern-node:latest 211125559768.dkr.ecr.us-east-1.amazonaws.com/node-app:latest
2. **Log in to ECR**: aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 211125559768.dkr.ecr.us-east-1.amazonaws.com
3. **Push the image**: docker push 211125559768.dkr.ecr.us-east-1.amazonaws.com/node-app:latest



```
PS C:\Users\MyPc\Desktop\node> docker tag mern-node:latest 211125559768.dkr.ecr.us-east-1.amazonaws.com/node-app:latest
PS C:\Users\MyPc\Desktop\node> aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 211125559768.dkr.e
cr.us-east-1.amazonaws.com/node-app
Login Succeeded
PS C:\Users\MyPc\Desktop\node> docker push 211125559768.dkr.ecr.us-east-1.amazonaws.com/node-app:latest
The push refers to repository [211125559768.dkr.ecr.us-east-1.amazonaws.com/node-app]
f8b4df7262a7: Preparing
32955446520f: Preparing
2246d519b523: Preparing
f5218cbdde31: Preparing
32955446520f: Pushed
3c777d951de2: Pushed
cb81227abde5: Pushed
e01a454893a9: Pushed
c45660adde37: Pushed
fe0fb3ab4a0f: Pushed
f1186e5061f2: Pushed
b2dba7477754: Pushed
latest: digest: sha256:70e6cf1b4d7cef3fd273cad3be5543cab915a7775a981cb226effb147ef538c1 size: 3047
```

Successfully Pushed image into ECR repo



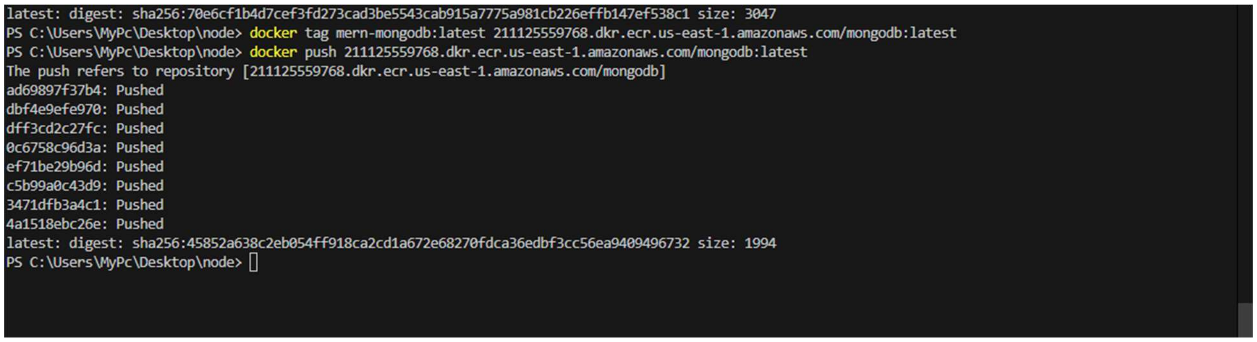
Amazon ECR > Private registry > Repositories > node-app

node-app

Images (1)

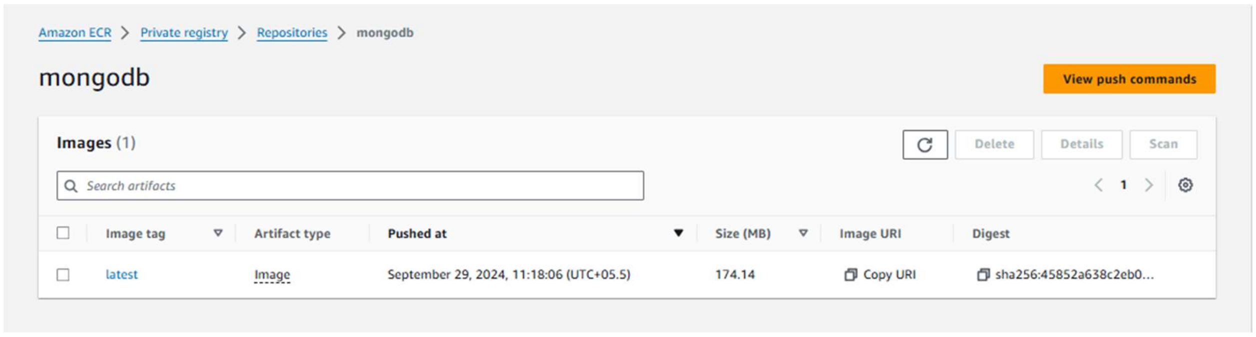
Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
latest	Image	September 29, 2024, 11:04:41 (UTC+05.5)	358.02	Copy URI	sha256:70e6cf1b4d7cef3...

we need push the ‘mern-mongodb’ image into ECR repo



```
latest: digest: sha256:70e6cf1b4d7cef3fd273cad3be5543cab915a7775a981cb226effb147ef538c1 size: 3047
PS C:\Users\MyPc\Desktop\node> docker tag mern-mongodb:latest 211125559768.dkr.ecr.us-east-1.amazonaws.com/mongodb:latest
PS C:\Users\MyPc\Desktop\node> docker push 211125559768.dkr.ecr.us-east-1.amazonaws.com/mongodb:latest
The push refers to repository [211125559768.dkr.ecr.us-east-1.amazonaws.com/mongodb]
ad69897f37b4: Pushed
dbf4e9efe970: Pushed
dff3cd2c27fc: Pushed
0c6758c96d3a: Pushed
ef71be29b96d: Pushed
c5b99a0c43d9: Pushed
3471dfb3a4c1: Pushed
4a1518ebc26e: Pushed
latest: digest: sha256:45852a638c2eb054ff918ca2cd1a672e68270fdca36edbf3cc56ea9409496732 size: 1994
PS C:\Users\MyPc\Desktop\node> 
```

Successfully Pushed image into ECR repo


Amazon ECR > Private registry > Repositories > mongodb

mongodb

Images (1)

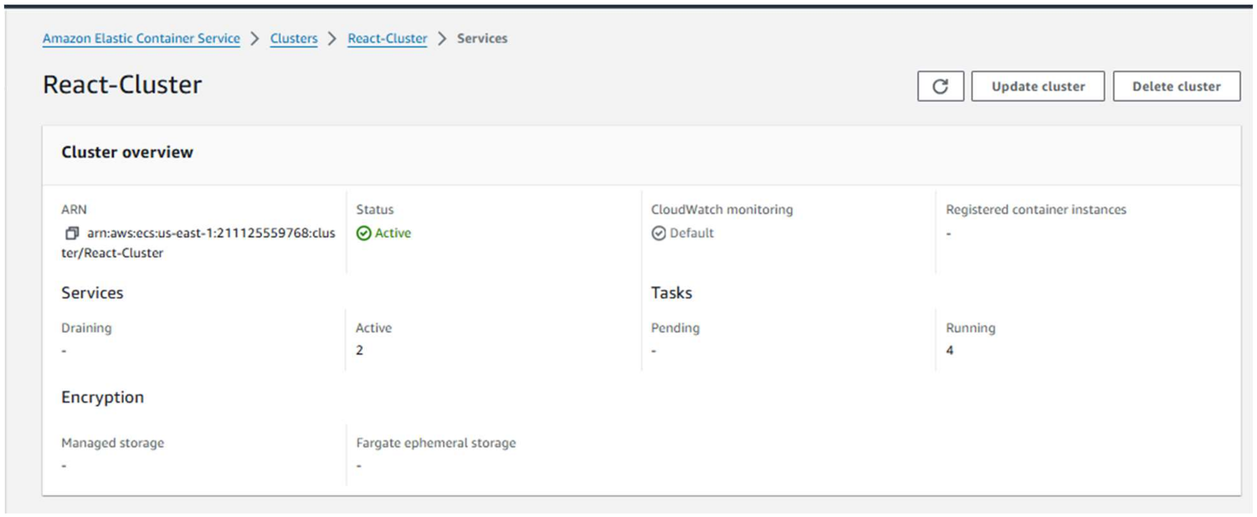
	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
☐	latest	Image	September 29, 2024, 11:18:06 (UTC+05.5)	174.14	Copy URI	sha256:45852a638c2eb0...

Step4 :Deploy to AWS ECS

Using ECS Fargate for Serverless Containers:

1. **Create a Cluster:**

- In the ECS dashboard, create an ECS cluster using the **Fargate** launch type.
- Name is React-Cluster


Amazon Elastic Container Service > Clusters > React-Cluster > Services

React-Cluster

Cluster overview

ARN	Status	CloudWatch monitoring	Registered container instances
arn:aws:ecs:us-east-1:211125559768:cluster/React-Cluster	Active	Default	-

Services

Draining	Active
-	2

Tasks

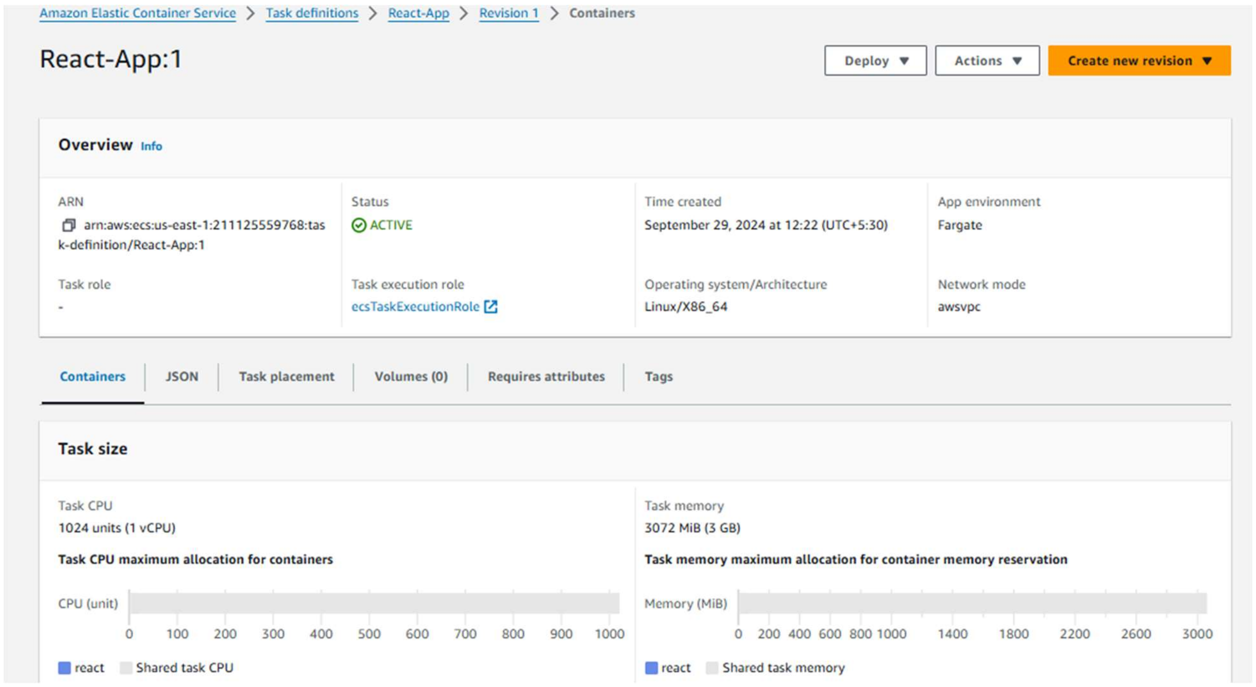
Pending	Running
-	4

Encryption

Managed storage	Fargate ephemeral storage
-	-

2. **Define Task Definitions:**

- Create a new **Task Definition** for the Node.js app and MongoDB.
- For the Node.js app, specify:
  Container Image: Your ECR image URL for the Node.js app
  Port Mappings: Port 3000.


Amazon Elastic Container Service > Task definitions > React-App > Revision 1 > Containers

React-App:1

Overview Info

ARN	Status	Time created	App environment
arn:aws:ecs:us-east-1:211125559768:task-definition/React-App:1	ACTIVE	September 29, 2024 at 12:22 (UTC+5:30)	Fargate
Task role	Task execution role	Operating system/Architecture	Network mode
-	ecsTaskExecutionRole	Linux/X86_64	awsvpc

Task size

Task CPU
1024 units (1 vCPU)

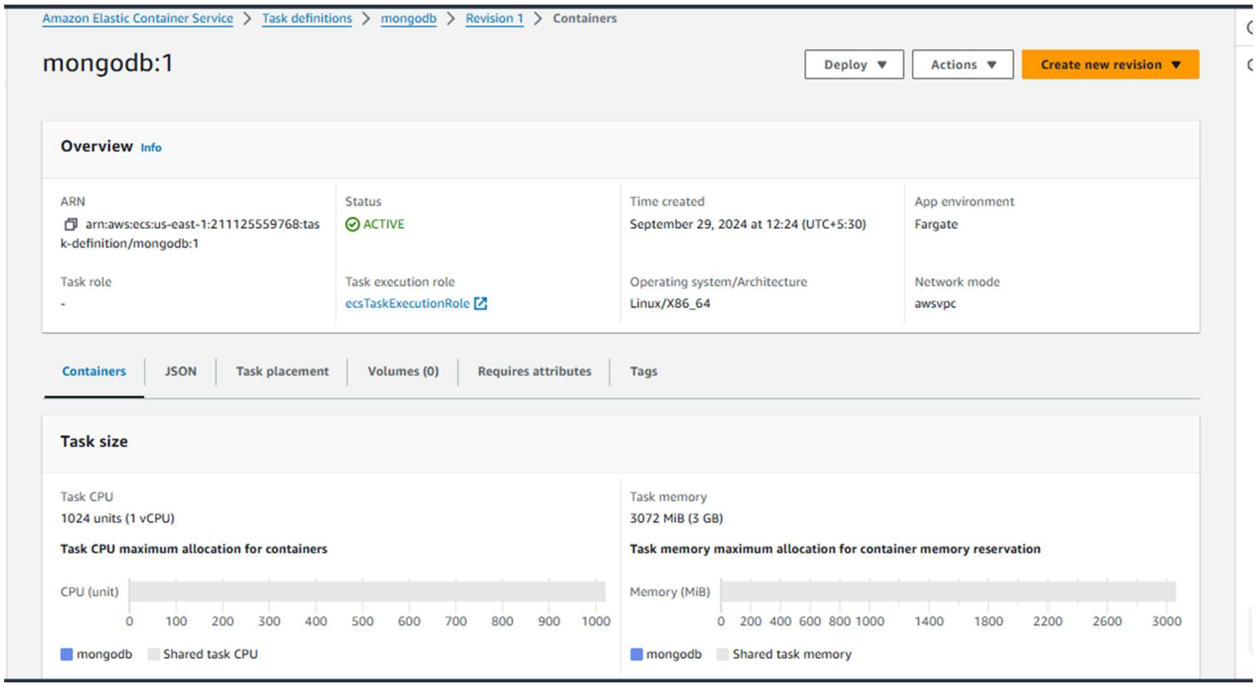
Task CPU maximum allocation for containers

Task memory
3072 MiB (3 GB)

Task memory maximum allocation for container memory reservation

For MongoDB, specify:

- Container Image: Your ECR image URL for MongoDB.
- Port Mappings: Port 27017.


Amazon Elastic Container Service > Task definitions > mongodb > Revision 1 > Containers

mongodb:1

Overview Info

ARN	Status	Time created	App environment
arn:aws:ecs:us-east-1:211125559768:task-definition/mongodb:1	ACTIVE	September 29, 2024 at 12:24 (UTC+5:30)	Fargate
Task role	Task execution role	Operating system/Architecture	Network mode
-	ecsTaskExecutionRole	Linux/X86_64	awsvpc

Task size

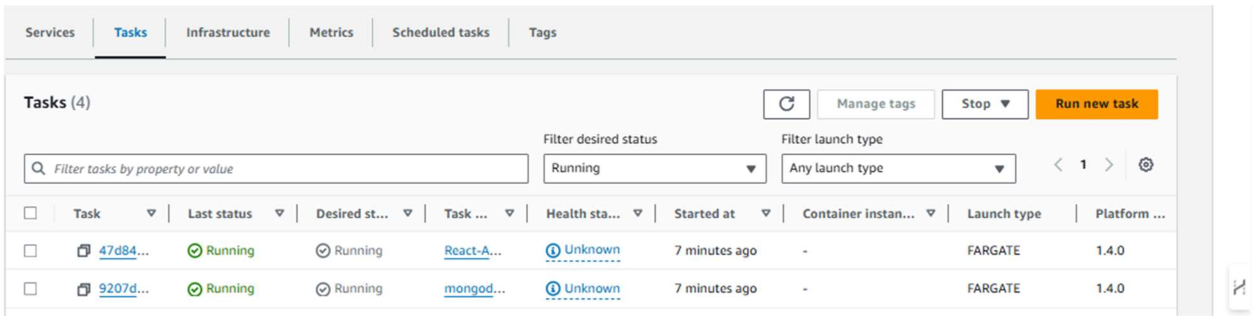
Task CPU
1024 units (1 vCPU)

Task CPU maximum allocation for containers

Task memory
3072 MiB (3 GB)

Task memory maximum allocation for container memory reservation

Successfully created the 2 tasks



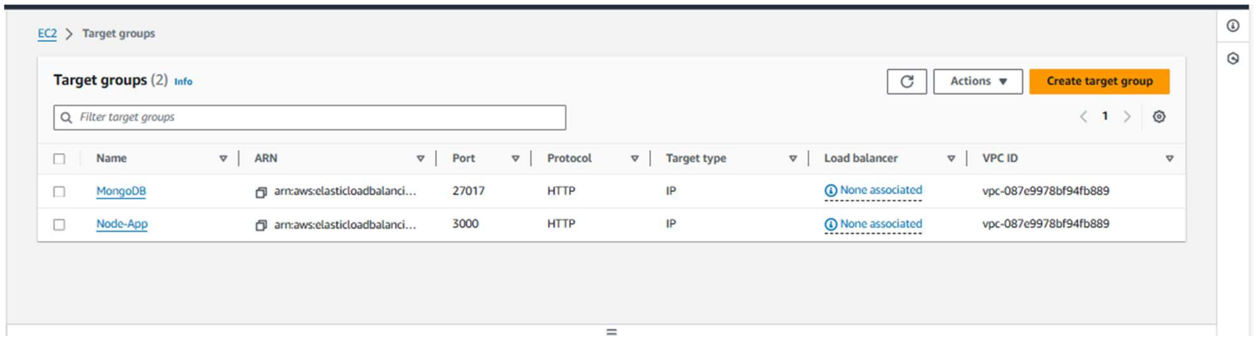
Services | Tasks | Infrastructure | Metrics | Scheduled tasks | Tags

Tasks (4)

Filter desired status: Running
Filter launch type: Any launch type

Task	Last status	Desired st...	Task ...	Health sta...	Started at	Container instan...	Launch type	Platform ...
47d84...	Running	Running	React-A...	Unknown	7 minutes ago	-	FARGATE	1.4.0
9207d...	Running	Running	mongod...	Unknown	7 minutes ago	-	FARGATE	1.4.0

Now, Create the load balancer before we creating load we need to create "Target groups"

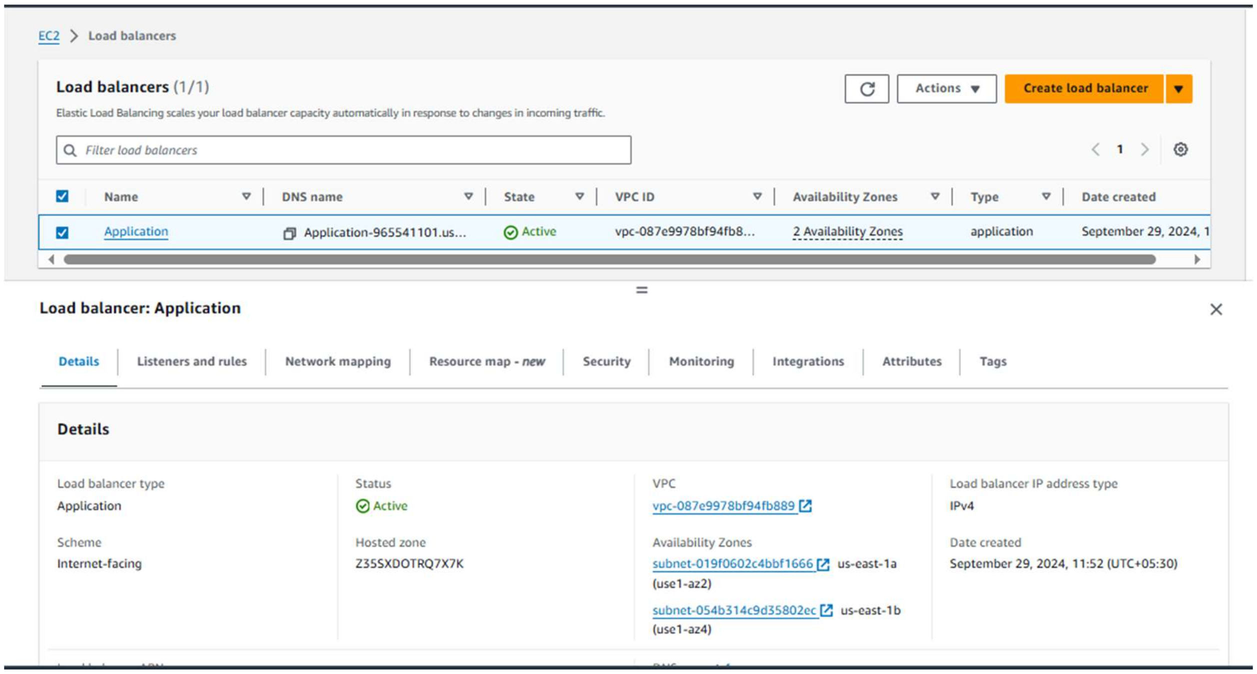


EC2 > Target groups

Target groups (2) Info

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
MongoDB	arn:aws:elasticloadbalanci...	27017	HTTP	IP	None associated	vpc-087e9978bf94fb889
Node-App	arn:aws:elasticloadbalanci...	3000	HTTP	IP	None associated	vpc-087e9978bf94fb889

Now ,Successfully created Load balancer



EC2 > Load balancers

Load balancers (1/1)

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

Name	DNS name	State	VPC ID	Availability Zones	Type	Date created
Application	Application-965541101.us...	Active	vpc-087e9978bf94fb8...	2 Availability Zones	application	September 29, 2024, 1

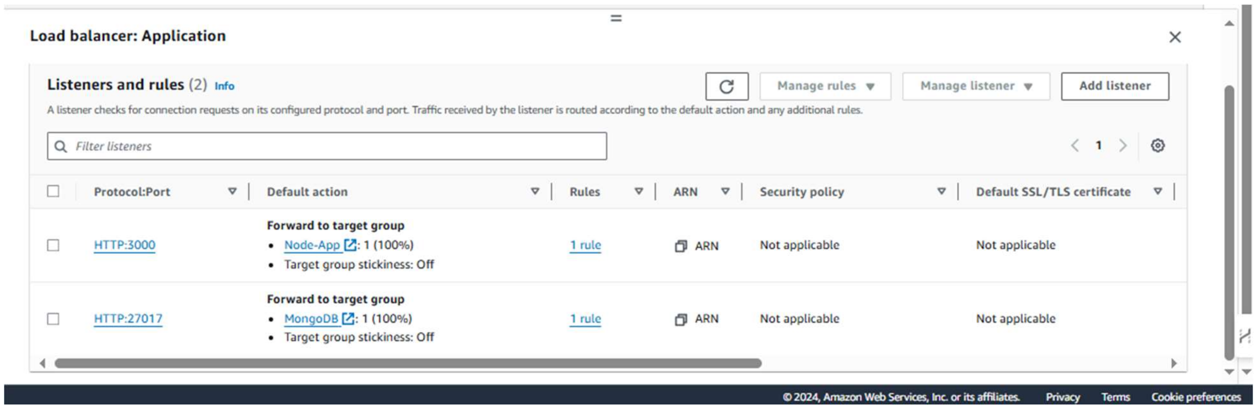
Load balancer: Application

Details | Listeners and rules | Network mapping | Resource map - new | Security | Monitoring | Integrations | Attributes | Tags

Details

Load balancer type: Application
Scheme: Internet-facing
Status: Active
Hosted zone: Z35SXDOTRQ7X7K
VPC: vpc-087e9978bf94fb889
Availability Zones: subnet-019f0602c4bbf1666 us-east-1a (use1-az2); subnet-054b314c9d35802ec us-east-1b (use1-az4)
Load balancer IP address type: IPv4
Date created: September 29, 2024, 11:52 (UTC+05:30)

In Load balancer we have added Listener Rules For node-app we have given the port number is 3000 and for mongodb we have given the port number is 27017



Load balancer: Application

Listeners and rules (2) Info

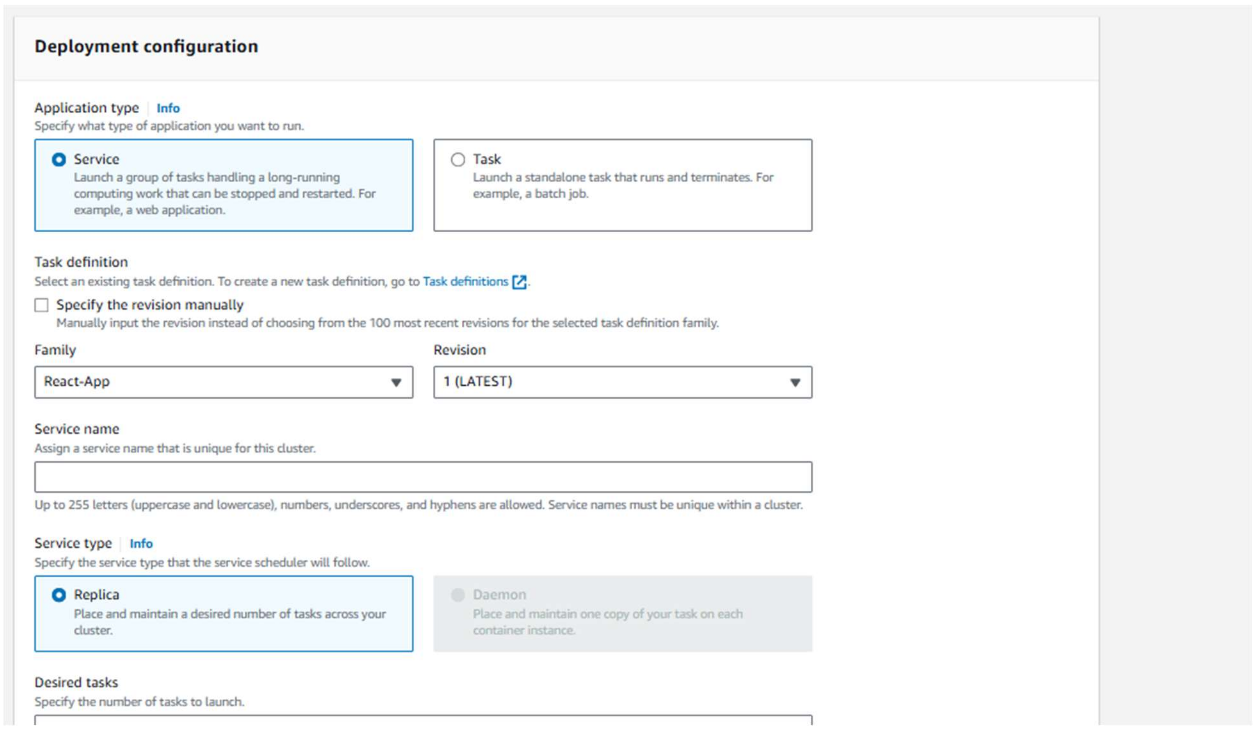
A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate
HTTP:3000	Forward to target group • Node-App: 1 (100%) • Target group stickiness: Off	1 rule	ARN	Not applicable	Not applicable
HTTP:27017	Forward to target group • MongoDB: 1 (100%) • Target group stickiness: Off	1 rule	ARN	Not applicable	Not applicable

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Create ECS Service:

- **Use your task definition to create a service in ECS.**



Deployment configuration

Application type Info
Specify what type of application you want to run.

[✓ Service] Launch a group of tasks handling a long-running computing work that can be stopped and restarted. For example, a web application.

Task — Launch a standalone task that runs and terminates. For example, a batch job.

Task definition
Select an existing task definition. To create a new task definition, go to Task definitions.

☐ Specify the revision manually
Manually input the revision instead of choosing from the 100 most recent revisions for the selected task definition family.

Family	Revision
React-App	1 (LATEST)

Service name
Assign a service name that is unique for this cluster.

Up to 255 letters (uppercase and lowercase), numbers, underscores, and hyphens are allowed. Service names must be unique within a cluster.

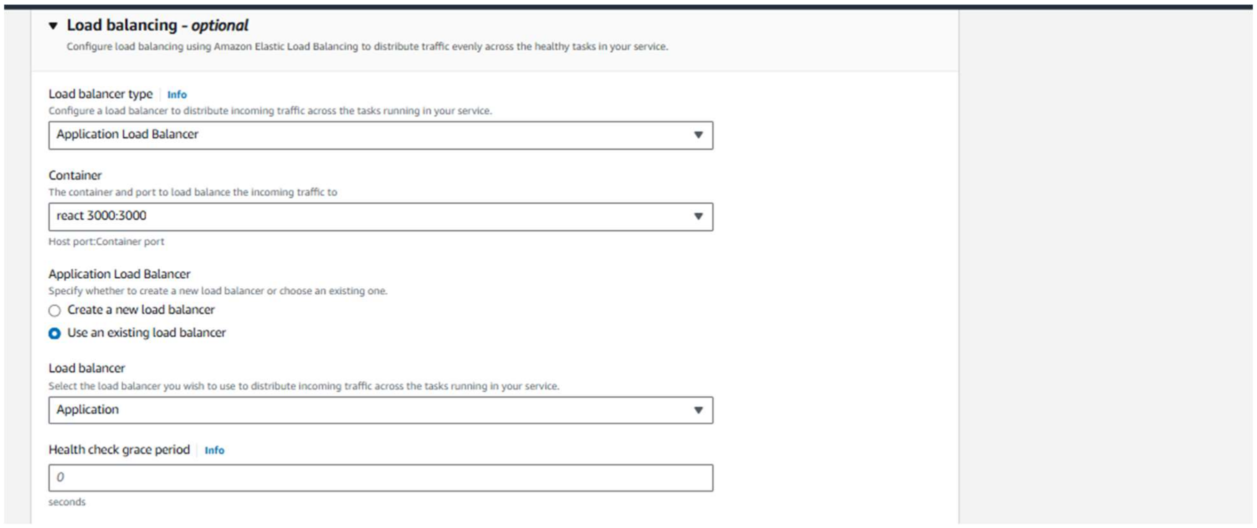
Service type Info
Specify the service type that the service scheduler will follow.

[✓ Replica] Place and maintain a desired number of tasks across your cluster.

Daemon — Place and maintain one copy of your task on each container instance.

Desired tasks
Specify the number of tasks to launch.

- **Attach an Application Load Balancer (ALB) to handle external traffic.**



▼ Load balancing - *optional*

Configure load balancing using Amazon Elastic Load Balancing to distribute traffic evenly across the healthy tasks in your service.

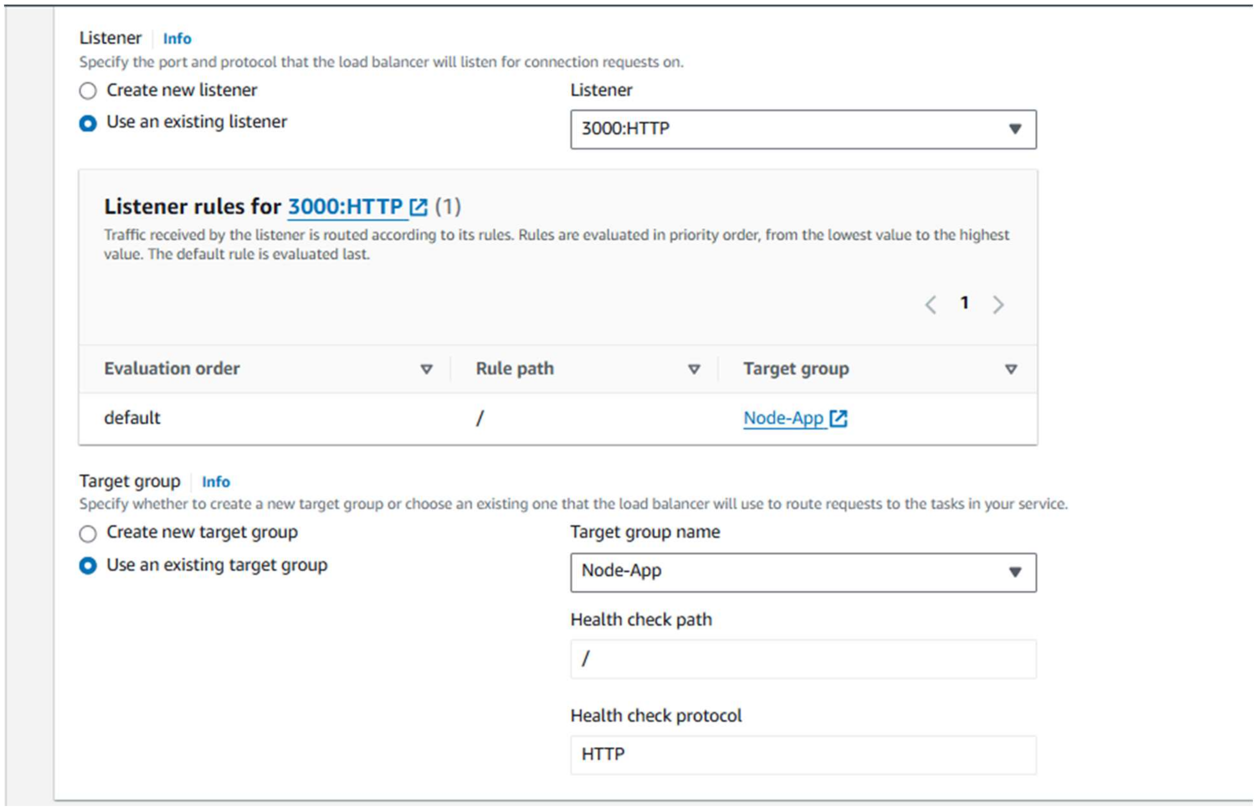
Load balancer type | Info
Configure a load balancer to distribute incoming traffic across the tasks running in your service.
Application Load Balancer

Container
The container and port to load balance the incoming traffic to
react 3000:3000
Host port:Container port

Application Load Balancer
Specify whether to create a new load balancer or choose an existing one.
- ○ Create a new load balancer
- ● Use an existing load balancer

Load balancer
Select the load balancer you wish to use to distribute incoming traffic across the tasks running in your service.
Application

Health check grace period | Info
0
seconds



Listener | Info
Specify the port and protocol that the load balancer will listen for connection requests on.
- ○ Create new listener
- ● Use an existing listener

Listener: 3000:HTTP

Listener rules for 3000:HTTP (1)

Traffic received by the listener is routed according to its rules. Rules are evaluated in priority order, from the lowest value to the highest value. The default rule is evaluated last.

Evaluation order	Rule path	Target group
default	/	Node-App

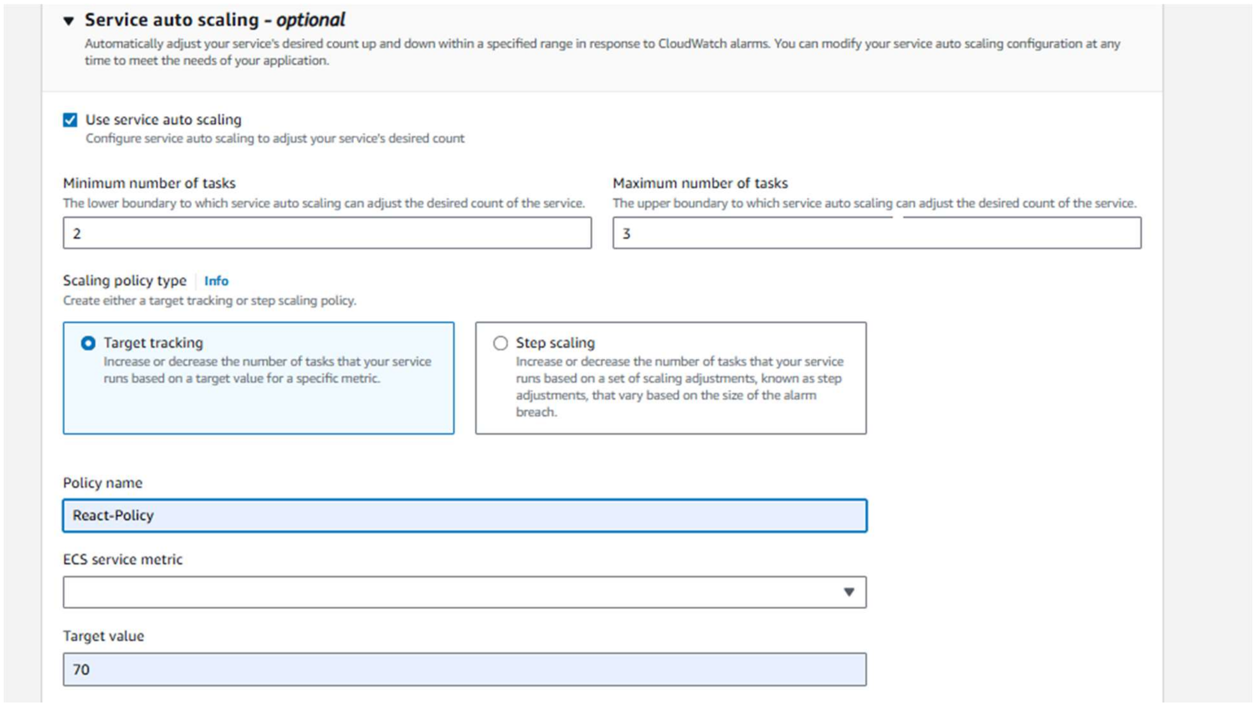
Target group | Info
Specify whether to create a new target group or choose an existing one that the load balancer will use to route requests to the tasks in your service.
- ○ Create new target group
- ● Use an existing target group

Target group name: Node-App

Health check path: /

Health check protocol: HTTP

- **Configure Auto Scaling for your service.**



Service auto scaling - *optional*

Automatically adjust your service's desired count up and down within a specified range in response to CloudWatch alarms. You can modify your service auto scaling configuration at any time to meet the needs of your application.

☑ Use service auto scaling
Configure service auto scaling to adjust your service's desired count

Minimum number of tasks The lower boundary to which service auto scaling can adjust the desired count of the service.	Maximum number of tasks The upper boundary to which service auto scaling can adjust the desired count of the service.
2	3

Scaling policy type | Info
Create either a target tracking or step scaling policy.

● Target tracking
Increase or decrease the number of tasks that your service runs based on a target value for a specific metric.

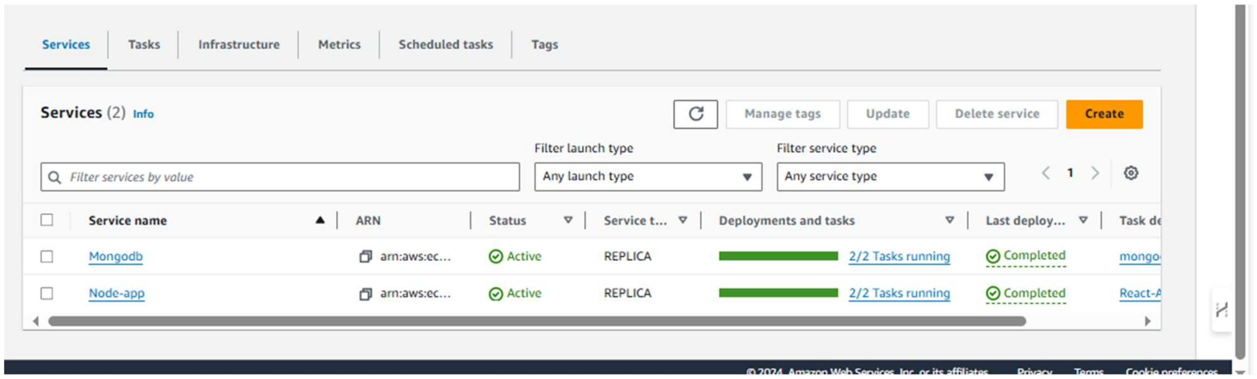
○ Step scaling
Increase or decrease the number of tasks that your service runs based on a set of scaling adjustments, known as step adjustments, that vary based on the size of the alarm breach.

Policy name
React-Policy

ECS service metric

Target value
70

Now, Successfully Deployed Node-app and Mongodb in services



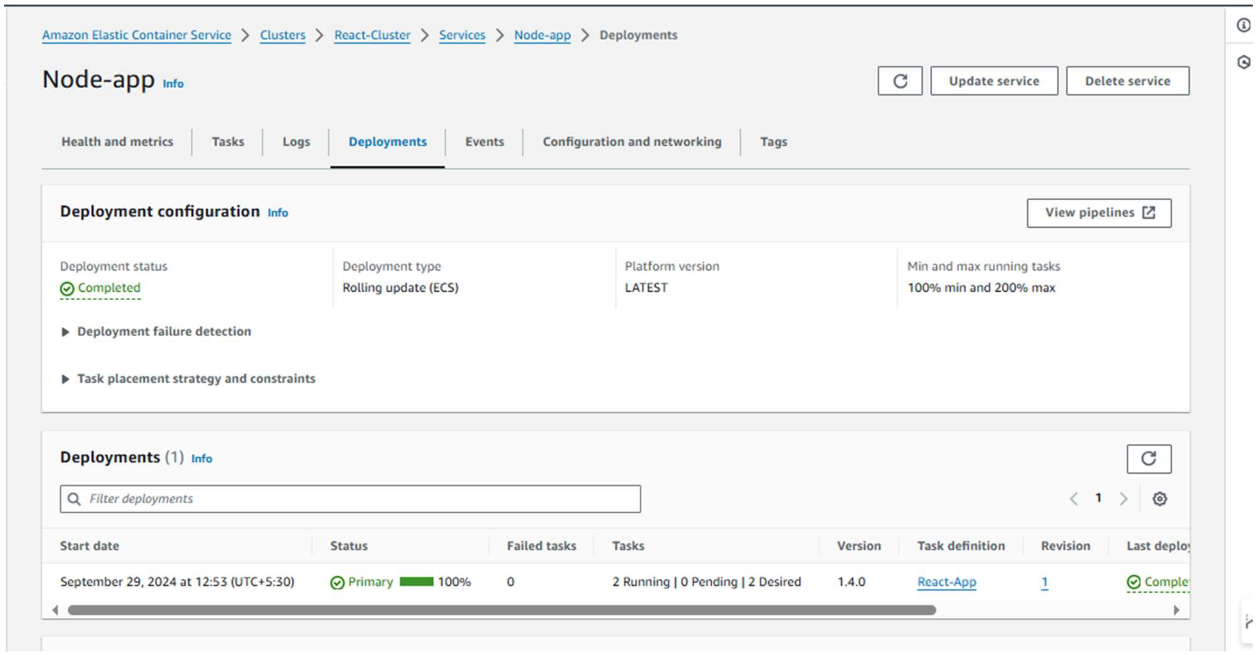
Services (2) Info

Filter services by value

Service name	ARN	Status	Service t...	Deployments and tasks	Last deploy...	Task de
Mongodb	arn:aws:ec...	Active	REPLICA	2/2 Tasks running	Completed	mongo
Node-app	arn:aws:ec...	Active	REPLICA	2/2 Tasks running	Completed	React-A

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Node-app is successfully Deployed



Amazon Elastic Container Service > Clusters > React-Cluster > Services > Node-app > Deployments

Node-app Info

Health and metrics | Tasks | Logs | Deployments | Events | Configuration and networking | Tags

Deployment configuration Info

Deployment status: Completed
Deployment type: Rolling update (ECS)
Platform version: LATEST
Min and max running tasks: 100% min and 200% max

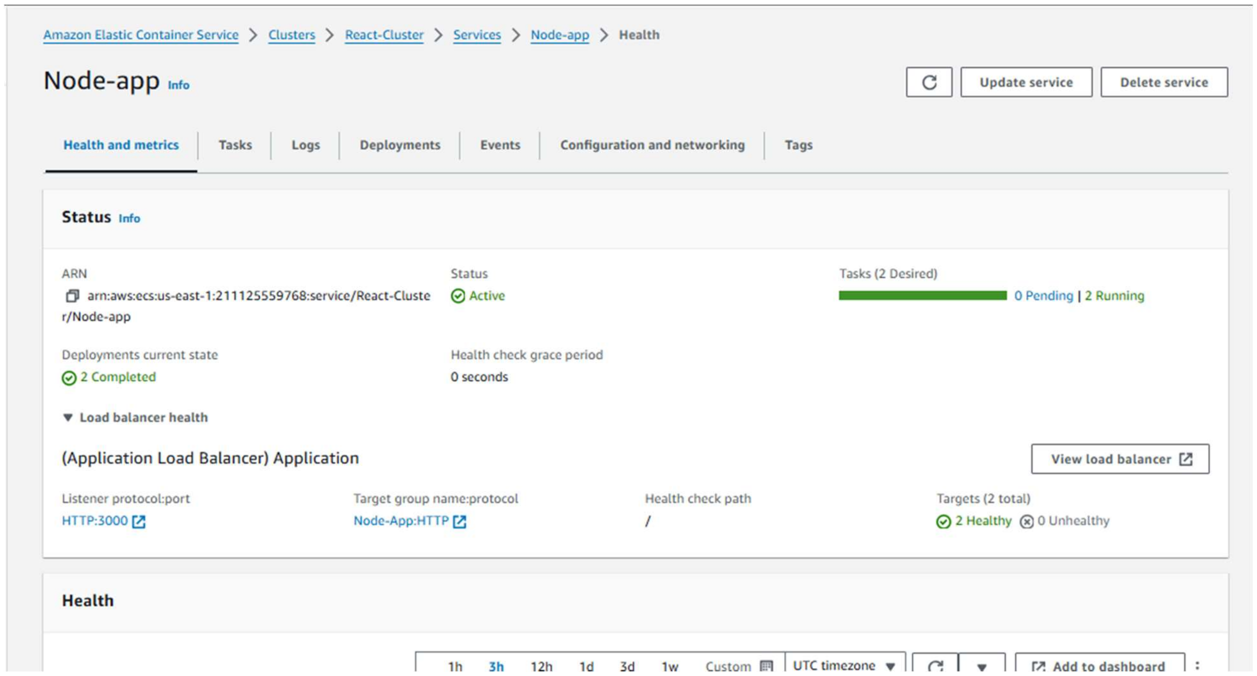
▶ Deployment failure detection

▶ Task placement strategy and constraints

Deployments (1) Info

Start date	Status	Failed tasks	Tasks	Version	Task definition	Revision	Last deplo
September 29, 2024 at 12:53 (UTC+5:30)	Primary 100%	0	2 Running | 0 Pending | 2 Desired	1.4.0	React-App	1	Comple

Health and metrics In services for Node-app



Amazon Elastic Container Service > Clusters > React-Cluster > Services > Node-app > Health

Node-app Info

Health and metrics | Tasks | Logs | Deployments | Events | Configuration and networking | Tags

Status Info

ARN: arn:aws:ecs:us-east-1:211125559768:service/React-Cluster/Node-app
Status: Active
Tasks (2 Desired): 0 Pending | 2 Running
Deployments current state: 2 Completed
Health check grace period: 0 seconds

▼ Load balancer health

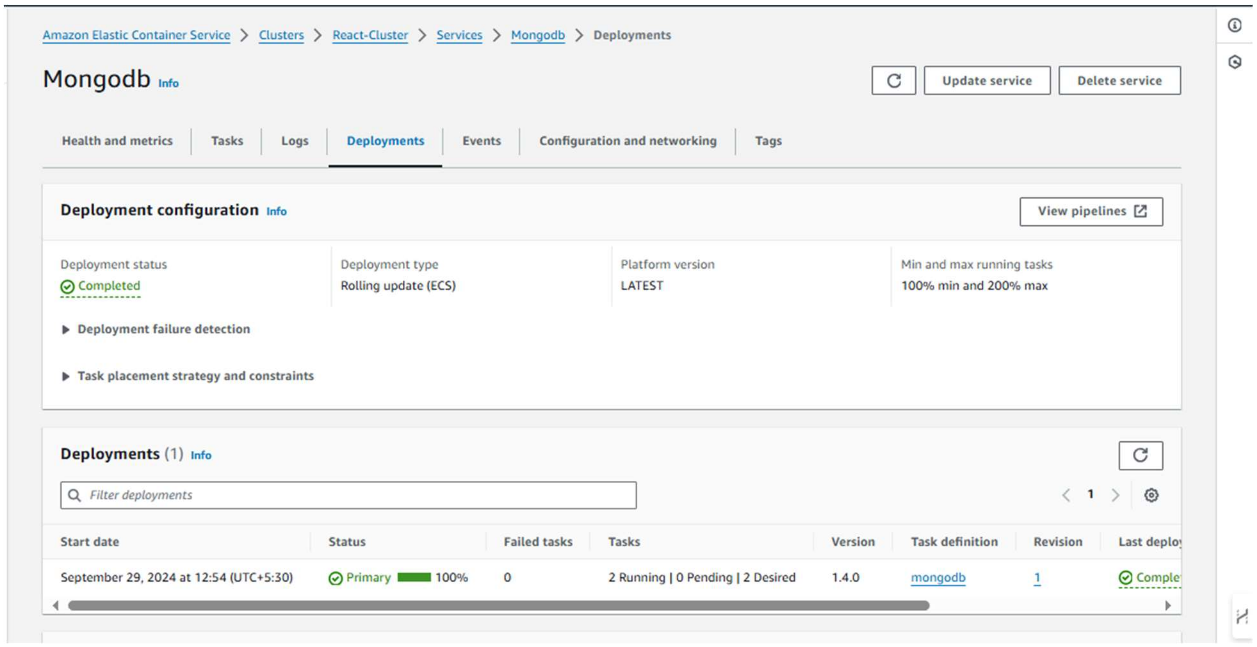
(Application Load Balancer) Application

Listener protocol:port: HTTP:3000
Target group name:protocol: Node-App:HTTP
Health check path: /
Targets (2 total): 2 Healthy 0 Unhealthy

Health

1h 3h 12h 1d 3d 1w Custom | UTC timezone | Add to dashboard

Mongodb is successfully Deployed


Amazon Elastic Container Service > Clusters > React-Cluster > Services > Mongodb > Deployments

Mongodb Info

Deployment configuration Info

Deployment status	Deployment type	Platform version	Min and max running tasks
Completed	Rolling update (ECS)	LATEST	100% min and 200% max

▶ Deployment failure detection

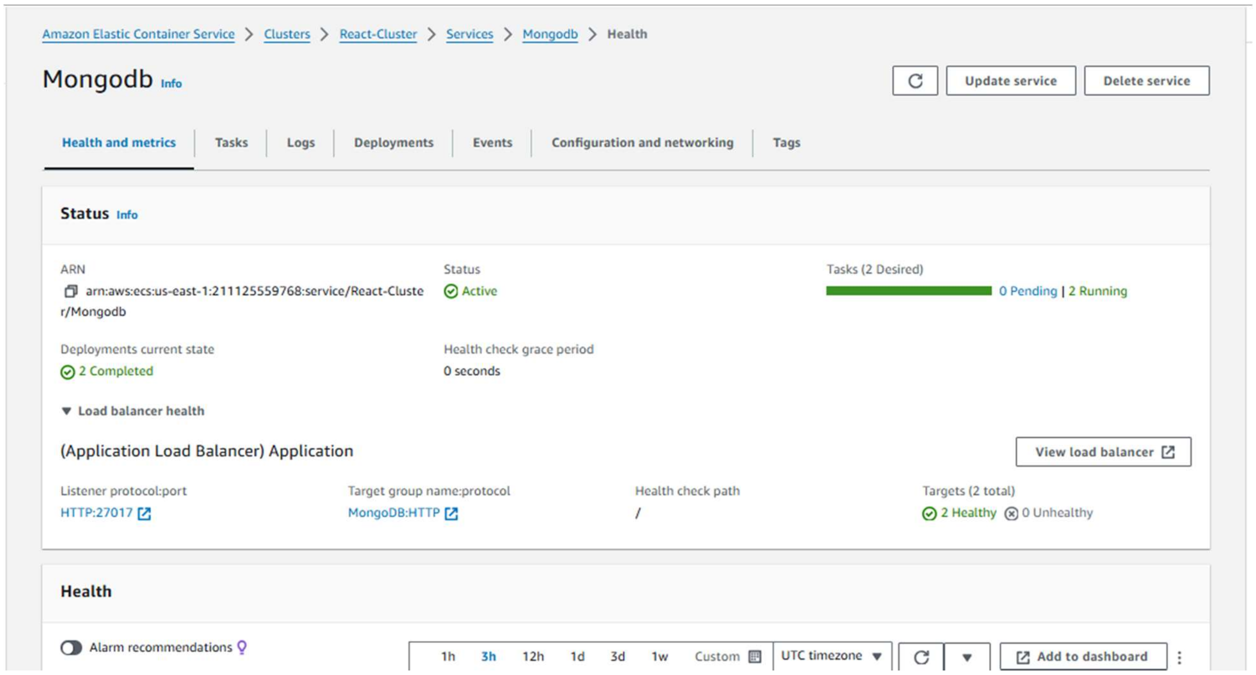
▶ Task placement strategy and constraints

Deployments (1) Info

Filter deployments

Start date	Status	Failed tasks	Tasks	Version	Task definition	Revision	Last deploy
September 29, 2024 at 12:54 (UTC+5:30)	Primary 100%	0	2 Running | 0 Pending | 2 Desired	1.4.0	mongodb	1	Comple

And check the health and matrics


Amazon Elastic Container Service > Clusters > React-Cluster > Services > Mongodb > Health

Mongodb Info

Status Info

ARN
arn:aws:ecs:us-east-1:211125559768:service/React-Cluster/Mongodb

Status
Active

Tasks (2 Desired)
0 Pending | 2 Running

Deployments current state
2 Completed

Health check grace period
0 seconds

▼ Load balancer health

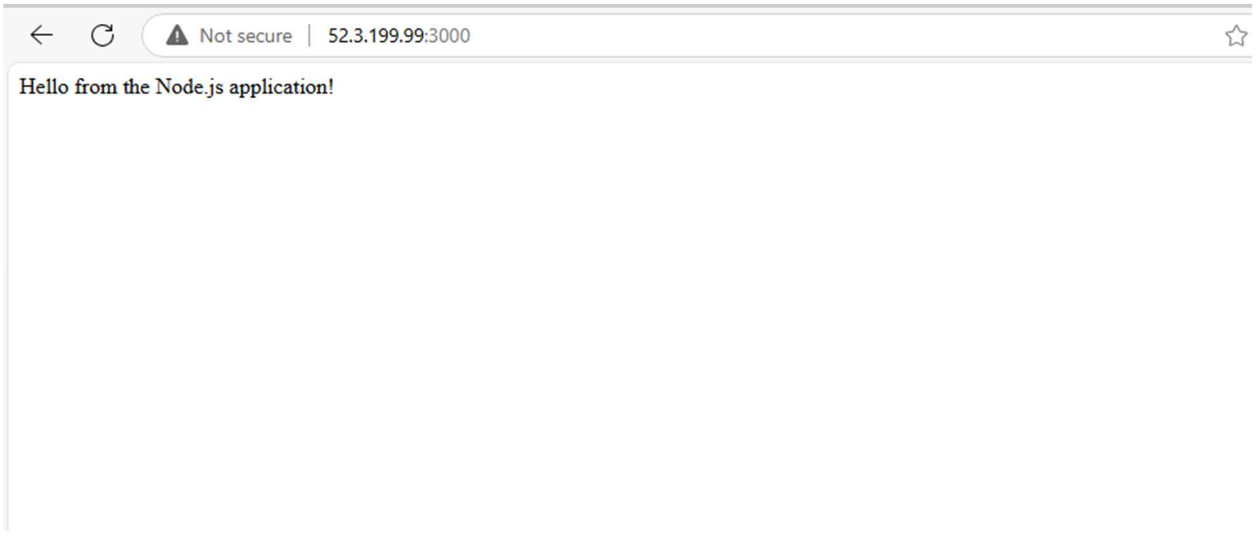
(Application Load Balancer) Application

Listener protocol:port	Target group name:protocol	Health check path	Targets (2 total)
HTTP:27017	MongoDB:HTTP	/	2 Healthy 0 Unhealthy

Health

Now we will run the webserver those two containers

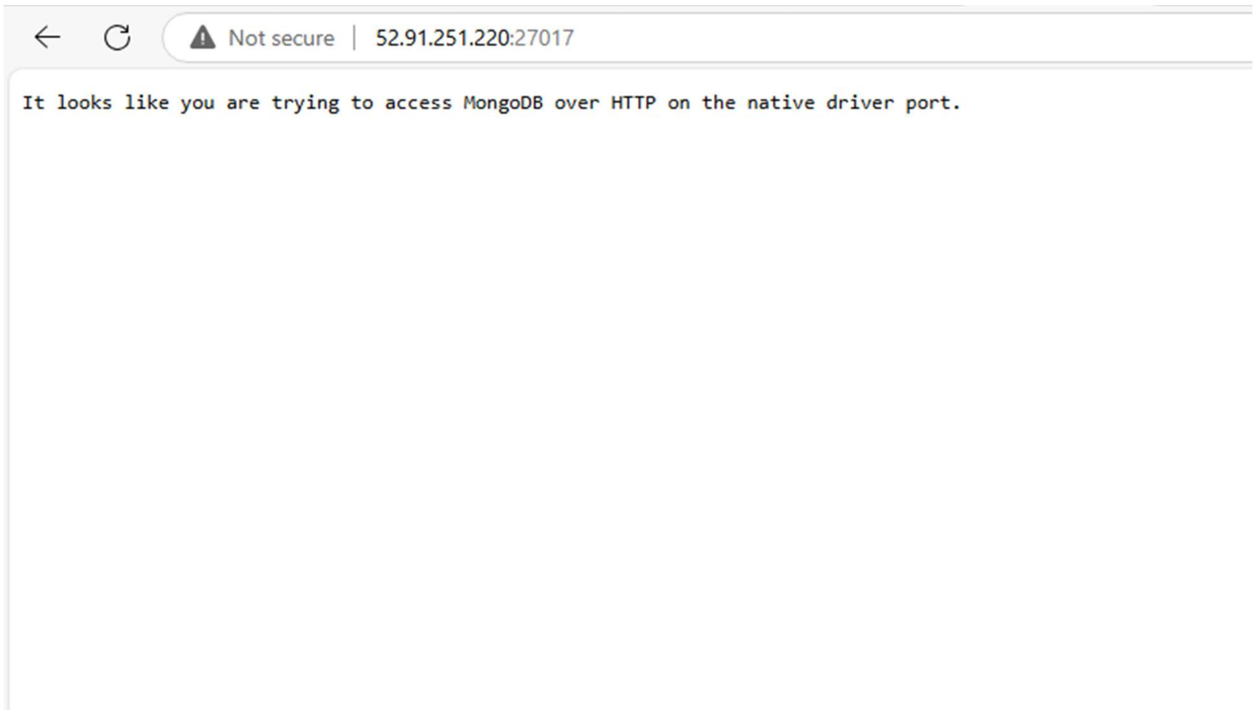
Node-app : 52.3.199.99:3000



Not secure | 52.3.199.99:3000

```
Hello from the Node.js application!
```

MongoDB: 52.91.251.220:27017



Not secure | 52.91.251.220:27017

```
It looks like you are trying to access MongoDB over HTTP on the native driver port.
```