Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2022, 9(2):123-142**

Research Article

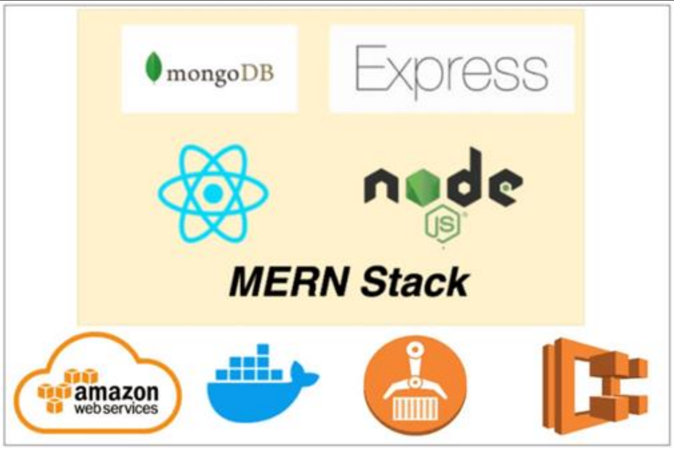
ISSN: 2394-2630
CODEN(USA): JSERBR

Architecting and Deploying MERN Stack Applications on AWS ECS

Bhargav Bachina

Abstract This paper presents a comprehensive guide for deploying MERN (MongoDB, Express.js, React.js, Node.js) stack applications on Amazon Web Services (AWS) Elastic Container Service (ECS). With the increasing popularity of MERN stack for web development and the growing adoption of cloud computing, efficient deployment strategies are essential. Leveraging AWS ECS offers scalability, reliability, and ease of management. The paper provides an overview of MERN stack components, AWS ECS features, and a step-by-step deployment process covering containerization, task definition creation, cluster setup, service configuration, and load balancing. Practical examples, code snippets, and best practices are discussed to empower developers and DevOps engineers in efficiently deploying MERN stack applications on AWS ECS, facilitating streamlined development workflows and robust, cloud-native solutions.

Keywords AWS, Programming, Software Development, Web Development, Cloud Computing



AWS provides more than 200 services and it's very important to know which service you should select for your needs. Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast, container management service that makes it easy to run, stop, and manage Docker containers on a cluster of Amazon EC2 instances. Amazon ECS lets you launch and stop container-based applications with simple API calls, allows you to get the state of your cluster from a centralized service, and gives you access to many familiar Amazon EC2 features.
In this post, we are going to deploy the MERN Stack on AWS ECS. First, we dockerize our app and push that image to Amazon ECR and run that app on Amazon ECS. We will also see how to access it from the browser.

- Prerequisites
- Example Project
- Set up a MongoDB Atlas

- Build For Production
- Externalize Environment Variables
- Dockerize the Project
- Running the WebApp on Docker
- Pushing Docker Image To ECR
- Deploying On AWS ECS
- Access the WebApp from the browser.
- Cleaning Up
- Summary
- Conclusion

1. Prerequisites

If you are new to web development, go through the below link on how to develop and build MERN Stack.

How To Develop and Build MERN Stack
(https://medium.com/bb-tutorials-and-thoughts/how-to-develop-and-build-mern-stack-9a7a1099624)

The other prerequisites to this post are Docker essentials. We are not going to discuss the basics such as what is a container or Docker. Below are the prerequisites you should know before going through this article.

A. Docker Essentials

You need to understand Docker concepts such as creating images, container management, etc. Below are some of the links that you can understand about Docker if you are new.

- Docker Docs (https://docs.docker.com/)
- Docker — A Beginner's guide to Dockerfile with a sample project (https://medium.com/bb-tutorials-and-thoughts/docker-a-beginners-guide-to-dockerfile-with-a-sample-project-6c1ac1f17490)
- Docker — Image creation and Management (https://medium.com/bb-tutorials-and-thoughts/docker-image-creation-and-management-9d91e4c277b1)
- Docker — Container Management With Examples Understanding (https://medium.com/bb-tutorials-and-thoughts/docker-container-management-with-examples-c280906158a8)
- Docker Volumes with an example (https://medium.com/bb-tutorials-and-thoughts/understanding-docker-volumes-with-an-example-d898cb5e40d7)

B. Kubernetes Essentials

You need to understand Kubernetes' essentials as well along with Docker essentials. Here are some of the docs to help you understand the concepts of Kubernetes.

- Kubernetes Docs (https://kubernetes.io/docs/concepts/)
- How to Get Started with Kubernetes (https://medium.com/bb-tutorials-and-thoughts/how-to-get-started-with-kubernetes-e06ea82d23b)
- Some Example **Projects** (https://medium.com/bb-tutorials-and-thoughts/docker/home)

C. AWS Prerequisites

Amazon is the leading cloud provider and pioneered Cloud Computing. AWS provides more than 200 services, and it's very important to know which service you should select for your needs.

If you are new to AWS or just getting started you can see the following article.

How To Get Started With AWS
(https://medium.com/bb-tutorials-and-thoughts/how-to-get-started-with-aws-9731a4f855a7)

2. Example Project

Here is an example of a simple tasks application that creates, retrieves, edits, and deletes tasks. We actually run the API on the NodeJS server and you can use MongoDB to save all these tasks.

https://miro.medium.com/v2/resize:fit:720/1*JN0njPe-UZ16utRelcQn5w.gif

As you add users we are making an API call to the nodejs server to store them and get the same data from the server when we retrieve them. You can see network calls in the following video.

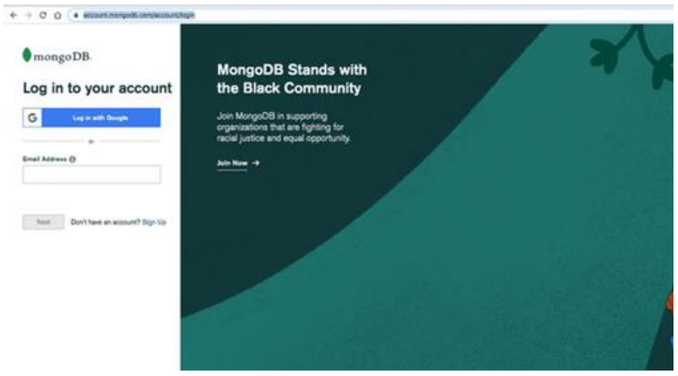
https://miro.medium.com/v2/resize:fit:720/1*j7EWjQuwx76WTOILWDvXHQ.gif

Here is a Github link to this project. You can clone it and run it on your machine.
// clone the projectgit clone https://github.com/bbachi/mern-stack-example
// React Codecd uinpm installnpm start
// API codecd apinpm installnpm run dev

3. Set up a MongoDB Atlas

The core of MongoDB Cloud is **MongoDB Atlas** (https://www.mongodb.com/cloud/atlas), a fully managed cloud database for modern applications. Atlas is the best way to run MongoDB, the leading modern database. There are two ways to deploy MongoDB on AWS and **you can check them here on this page** (https://aws.amazon.com/quickstart/architecture/mongodb/). We are using fully-managed MongoDB Cluster for this post.

Let's create your MongoDB Account here
(https://account.mongodb.com/account/login) .You can either log in with any of your Gmail accounts or you can provide any other email address to create the account.



mongoDB
Log in to your account
Log in with Google
Email Address
Next
Don't have an account? Sign Up
MongoDB Stands with the Black Community
Join MongoDB in supporting organizations that are fighting for racial justice and equal opportunity.
Join Now →

Figure 1: MongoDB Atlas login

Once you log in with your account you will see the dashboard below where you can create clusters.

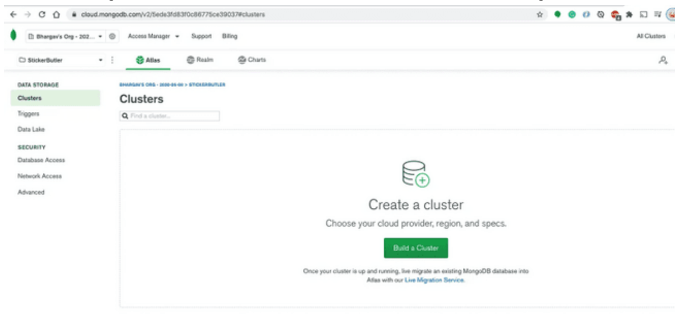
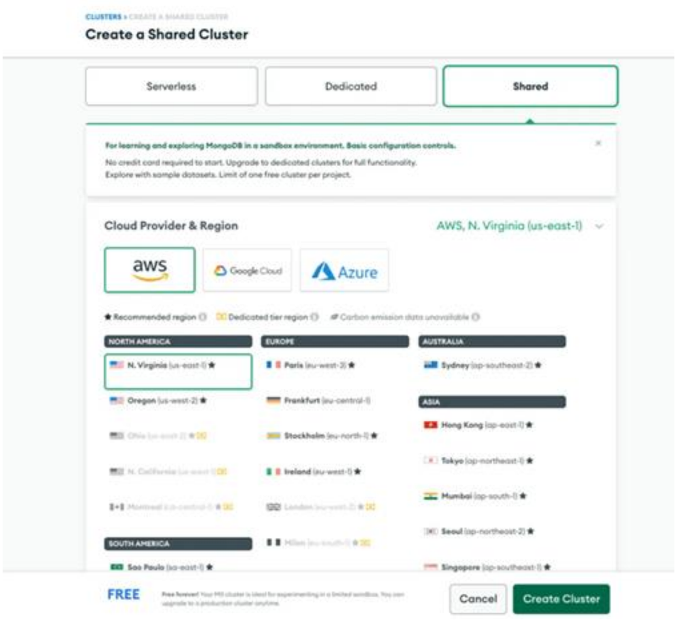


Figure 2: MongoDB Dashboard

Let's create a cluster called todo-cluster by clicking on the build a cluster and selecting all the details below. Make sure you select AWS Cloud.


CLUSTERS › CREATE A SHARED CLUSTER

Create a Shared Cluster

Serverless | Dedicated | Shared

For learning and exploring MongoDB in a sandbox environment. Basic configuration controls.
No credit card required to start. Upgrade to dedicated clusters for full functionality.
Explore with sample datasets. Limit of one free cluster per project.

Cloud Provider & Region — AWS, N. Virginia (us-east-1)

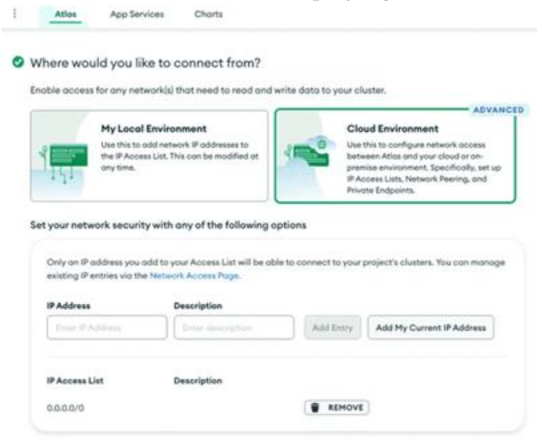
★ Recommended region · Dedicated tier region · Carbon emission data unavailable

NORTH AMERICA	EUROPE	AUSTRALIA / ASIA
N. Virginia (us-east-1) ★	Paris (eu-west-3) ★	Sydney (ap-southeast-2) ★
Oregon (us-west-2) ★	Frankfurt (eu-central-1)	Hong Kong (ap-east-1) ★
Ohio (us-east-2) ★	Stockholm (eu-north-1) ★	Tokyo (ap-northeast-1) ★
N. California (us-west-1)	Ireland (eu-west-1) ★	Mumbai (ap-south-1) ★
Montreal (ca-central-1) ★	London (eu-west-2) ★	Seoul (ap-northeast-2) ★
SOUTH AMERICA: Sao Paulo (sa-east-1) ★	Milan (eu-south-1) ★	Singapore (ap-southeast-1) ★

FREE — Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Figure 3: Creating a Cluster

Make sure you select the Cloud Environment since we are deploying this on AWS Cloud.


Where would you like to connect from?

Enable access for any network(s) that need to read and write data to your cluster.

My Local Environment — Use this to add network IP addresses to the IP Access List. This can be modified at any time.

Cloud Environment (ADVANCED) — Use this to configure network access between Atlas and your cloud or on-premise environment. Specifically, set up IP Access Lists, Network Peering, and Private Endpoints.

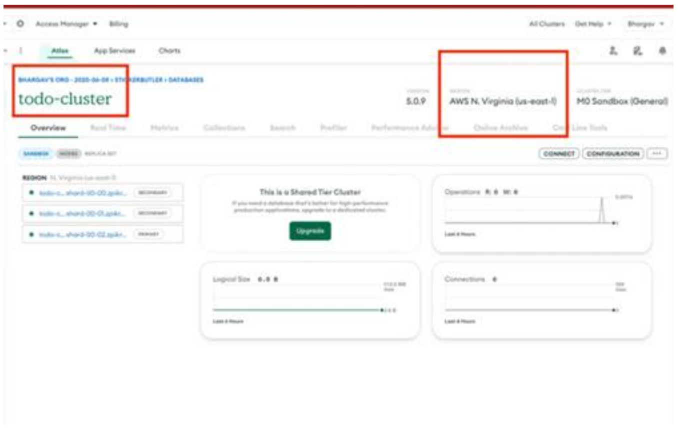
Set your network security with any of the following options

Only an IP address you add to your Access List will be able to connect to your project's clusters. You can manage existing IP entries via the Network Access Page.

IP Access List	Description
0.0.0.0/0	

Figure 4: Cloud Environment

Here is the cluster we created below.

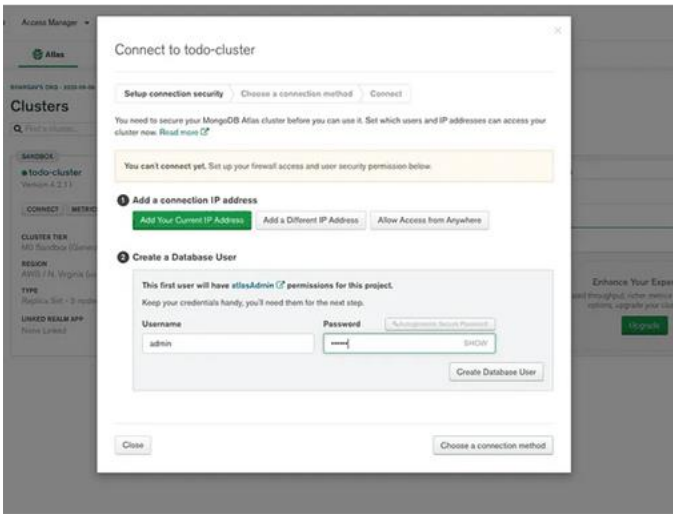

todo-cluster

VERSION 5.0.9 · REGION AWS N. Virginia (us-east-1) · CLUSTER TIER M0 Sandbox (General)

This is a Shared Tier Cluster

Figure 5: todo-cluster

You can click on the connect button to see the details about connecting to the cluster. You need to create a user and Allow Access from anywhere for now.


Connect to todo-cluster
Setup connection security
Choose a connection method
Connect
Add Your Current IP Address
Add a Different IP Address
Allow Access from Anywhere
Create Database User
Close
Choose a connection method

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. Read more

You can't connect yet. Set up your firewall access and user security permission below.

1 Add a connection IP address

2 Create a Database User

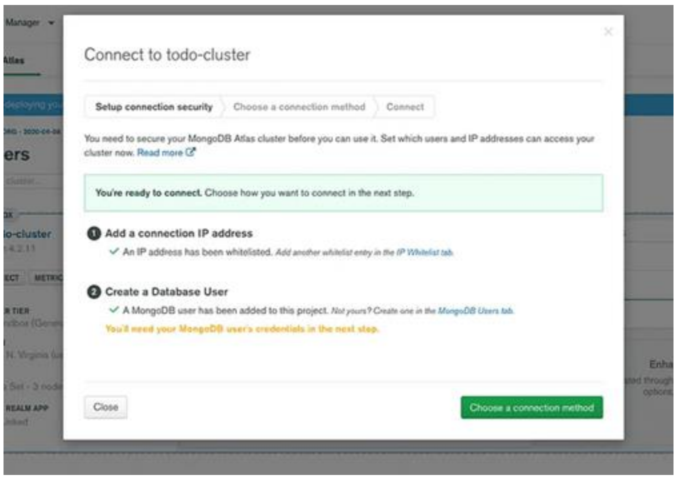
This first user will have atlasAdmin permissions for this project.

Keep your credentials handy, you'll need them for the next step.

Username: admin

Password: •••••

Figure 6: Connecting to cluster


Connect to todo-cluster
Setup connection security
Choose a connection method
Connect
Close
Choose a connection method

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. Read more

You're ready to connect. Choose how you want to connect in the next step.

1 Add a connection IP address

✓ An IP address has been whitelisted. *Add another whitelist entry in the IP Whitelist tab.*

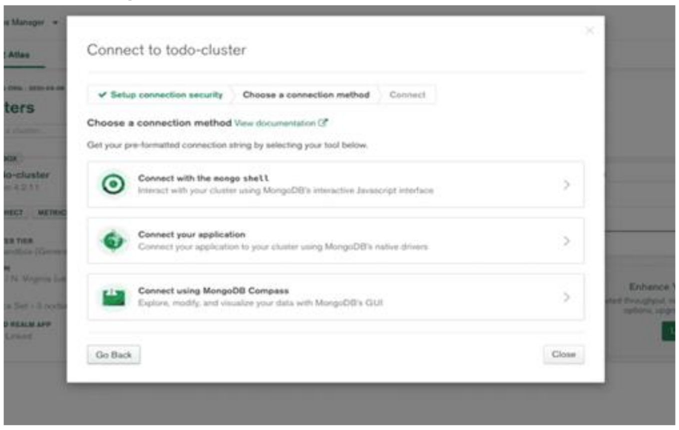
2 Create a Database User

✓ A MongoDB user has been added to this project. *Not yours? Create one in the MongoDB Users tab.*

You'll need your MongoDB user's credentials in the next step.

Figure 7: Connecting to cluster

You can see three ways of connecting to the cluster on the next screen.



Choose a connection method View documentation

Get your pre-formatted connection string by selecting your tool below.

Connect with the mongo shell
Interact with your cluster using MongoDB's interactive Javascript interface

Connect your application
Connect your application to your cluster using MongoDB's native drivers

Connect using MongoDB Compass
Explore, modify, and visualize your data with MongoDB's GUI

Figure 8: Ways of connecting

We will see all these three ways to connect to the cluster in the next sections.

A. Create a Database

We have created a cluster and it’s time to create a database. Click on the collections to create a new database as below.

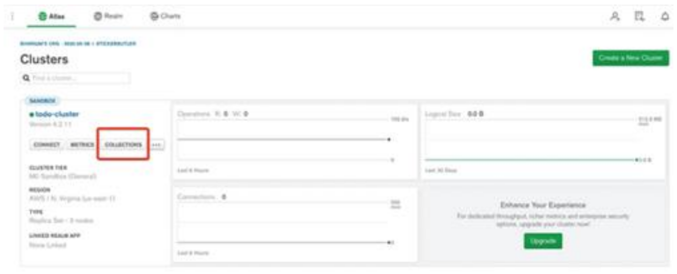


Figure 9: Collections

Click on the Add My Own Data Button to create a new database.

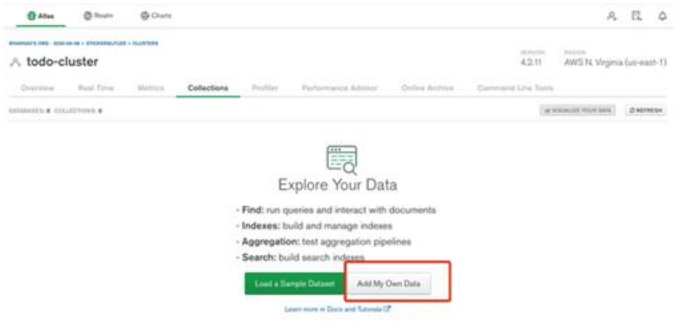
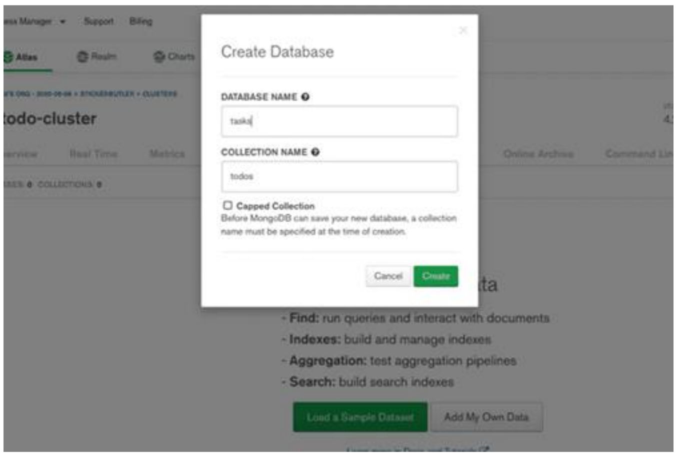

Atlas
Realm
Charts
todo-cluster
4.2.11
AWS N. Virginia (us-east-1)
Overview
Real Time
Metrics
Collections
Profiler
Performance Advisor
Online Archive
Command Line Tools
Explore Your Data
- Find: run queries and interact with documents
- Indexes: build and manage indexes
- Aggregation: test aggregation pipelines
- Search: build search indexes
Load a Sample Dataset
Add My Own Data
Learn more in Docs and Tutorials

Figure 10: Add My Own Data

I have given a database name as tasks and the collection name is todos.


Create Database

DATABASE NAME: tasks

COLLECTION NAME: todos

Capped Collection

Before MongoDB can save your new database, a collection name must be specified at the time of creation.
Cancel
Create
todo-cluster
Find: run queries and interact with documents
Indexes: build and manage indexes
Aggregation: test aggregation pipelines
Search: build search indexes
Load a Sample Dataset
Add My Own Data

Figure 11: Creating a Database

You will see the below dashboard once the database is created. We have a database with empty collections.

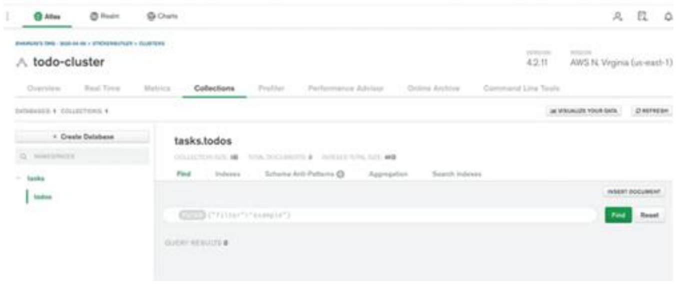
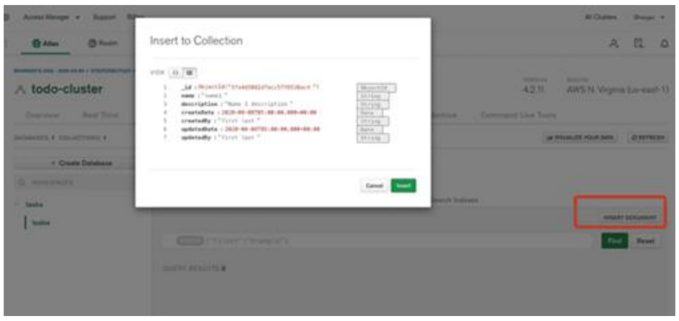


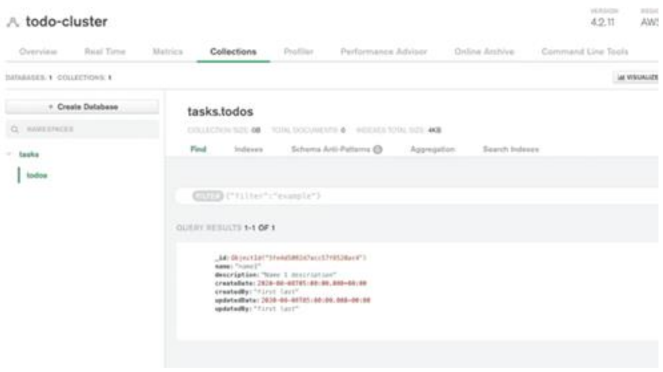
Figure 12: Empty Collection

Let’s insert the first document into the collection by clicking the button insert document



```
_id : ObjectId("[illegible]")
name : "name1"
description : "Name 1 description"
createDate : [illegible]
createdBy : "first last"
updatedDate : [illegible]
updatedBy : "first last"
```

Figure 13: Inserting the Document



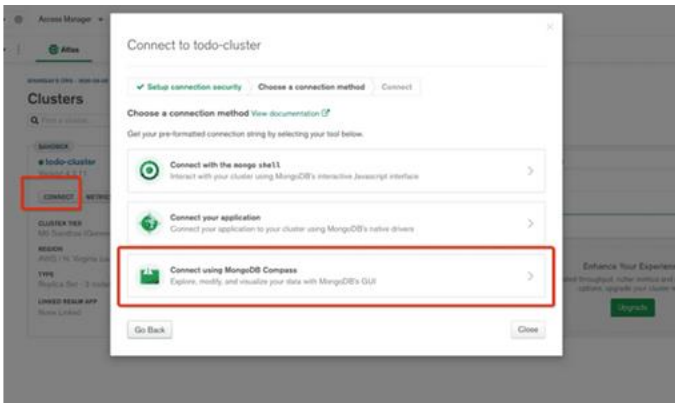
```
_id: ObjectId("[illegible]")
name: "name1"
description: "Name 1 description"
createDate: [illegible]
createdBy: "first last"
updatedDate: [illegible]
updatedBy: "first last"
```

Figure 14: Document Inserted

B. Connect with Mongo Compass

We have seen three ways we can connect to this cluster and read the collections. Let’s connect to the database with Mongo Compass. **The first thing we need to do is to download and install Mongo Compass from this link** (https://www.mongodb.com/try/download/compass).

Let’s get a connection string from the Atlas dashboard as below.



Choose a connection method View documentation

Get your pre-formatted connection string by selecting your tool below.

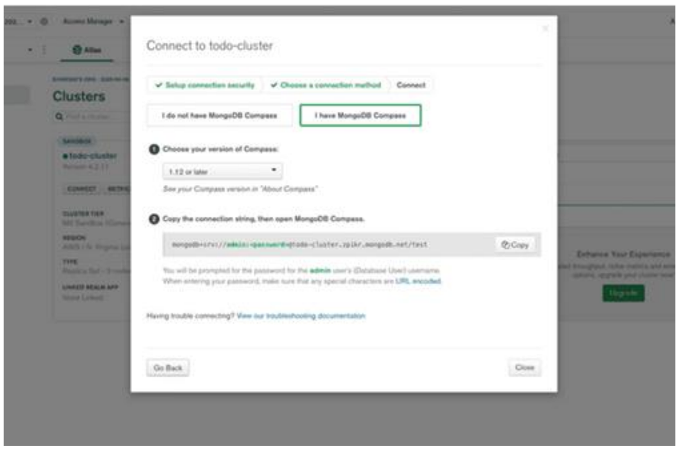
Connect with the mongo shell
Interact with your cluster using MongoDB's interactive Javascript interface

Connect your application
Connect your application to your cluster using MongoDB's native drivers

Connect using MongoDB Compass
Explore, modify, and visualize your data with MongoDB's GUI

Figure 15: Connect with MongoDB Compass

Replace the password with the password that you created above.


Connect to todo-cluster

✓ Setup connection security ✓ Choose a connection method Connect

I do not have MongoDB Compass [✓ I have MongoDB Compass]

1 Choose your version of Compass:

1.12 or later

See your Compass version in "About Compass"

2 Copy the connection string, then open MongoDB Compass.

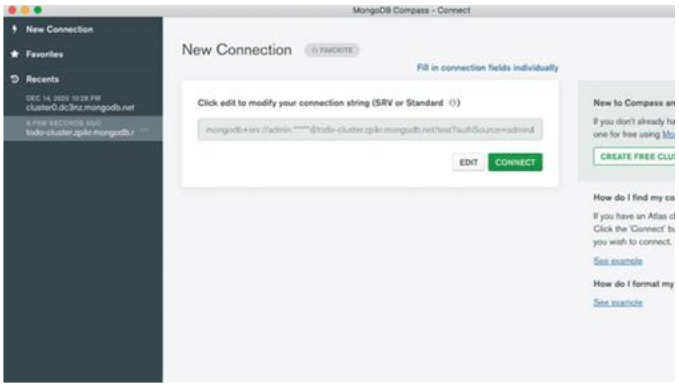
```
mongodb+srv://admin:<password>@todo-cluster.zpikr.mongodb.net/test
```

You will be prompted for the password for the admin user's (Database User) username.
When entering your password, make sure that any special characters are URL encoded.

Having trouble connecting? View our troubleshooting documentation

Figure 16: Connection String

Let's connect to the database with the connection string


MongoDB Compass - Connect

New Connection

Fill in connection fields individually

Click edit to modify your connection string (SRV or Standard)

```
mongodb+srv://admin:*****@todo-cluster.zpikr.mongodb.net/test?authSource=admin&
```

Figure 17: Connect with Connection String

You can actually see the same collection in the MongoDB Compass as well.

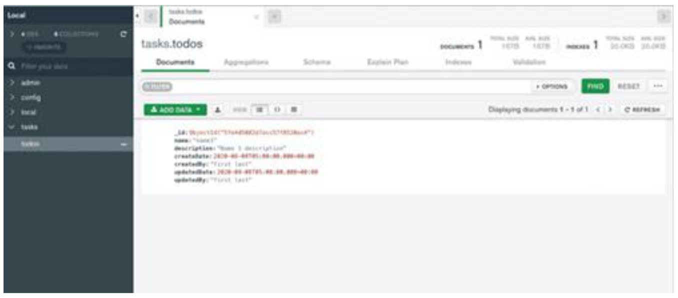


Figure 18: MongoDB Compass

Here is the connection string that you can connect to MongoDB.
mongodb+srv://admin123:admin123@todo-cluster.zpikr.mongodb.net/?retryWrites=true&w=majority

4. Build for Production

There are so many ways you can build MERN Stack for production and it depends on the use case or where we are deploying the MERN Stack. This article explains different ways to build MERN Stack for production.
How to Build MERN Stack for Production (https://medium.com/bb-tutorials-and-thoughts/how-to-build-mern-stack-for-production-1462e70a35cb)

5. Externalize Environment Variables

Reading environment variables is one of the most common things that we do when we are building apps. It doesn't matter whether you are developing front end app or backend API you have so many variables that should be outside of your application source code that makes your app or API more configurable. For example, if you want to hide logger statements in production or do something else based on the environment you can pass this as an environment variable. If you want to change later all you need to change is in one place.

Reading Environment Variables In NodeJS REST API
(https://medium.com/bb-tutorials-and-thoughts/reading-environment-variables-in-nodejs-rest-api-e75bb04b813d)

When it comes to this application, there are two environment variables one is the Mongo Connection string and another one is PORT.

PORT=80MONGO_CONNECTION_STRING=mongodb+srv://admin123:admin123@todocluster.zpikr.mongodb.net/?retryWrites=true&w=majority

You have to put these in the webpack.config.js file so that these values are used when we dockerize the app for **production**.

https://gist.github.com/bbachi/aa25aec5b82320d28cc5ee137bb8b8cf#file-webpack-config-js

7. Dockerize the WebApp

Amazon EKS is a managed service that makes it easy for you to run Kubernetes on AWS. The first thing you need to do is to dockerize your project.

We use the multi-stage builds for efficient docker images. Building efficient Docker images are very important for faster downloads and lesser surface attacks. In this multi-stage build, building a React app and putting those static assets in the build folder is the first step. The second step involves building the API. Finally, the third step involves taking those static build files and API build and serving the React static files through the API server.

We need to update the server.js file in the NodeJS API to let Express know about the React static assets and send the index.html as a default route. Here is the updated server.js file. Notice the line numbers **41** and **20**.

https://gist.github.com/bbachi/e828985a85cfb9da08164afa88549211#file-server-js

Let's build an image with the Dockerfile. Here are the things we need for building an image.

A. Stage 1

- Start from the base image node:14-slim
- There are two package.json files: one is for the nodejs server and another is for React UI. We need to copy these into the Docker file system and install all the dependencies.
- We need this step first to build images faster in case there is a change in the source later. We don't want to repeat installing dependencies every time we change any source files.
- Copy all the source files.
- Install all the dependencies.
- Run npm run build to build the React App and all the assets will be created under build a folder within the ui folder.

B. Stage 2

- Start from the base image node:14-slim
- Copy the nodejs package.json into ./api folder
- Install all the dependencies
- Finally, copy the server.js into the same folder

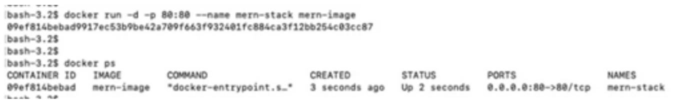
C. Stage 3

- Start from the base image node:14-slim
- Copy all the built files from UI Build
- Copy all the built files from API Build
- Finally, run this command node api.bundle.js

Here is the complete Dockerfile for the entire build.

https://gist.github.com/bbachi/06eecfc6c956d01c99180523c2677c15#file-dockerfile

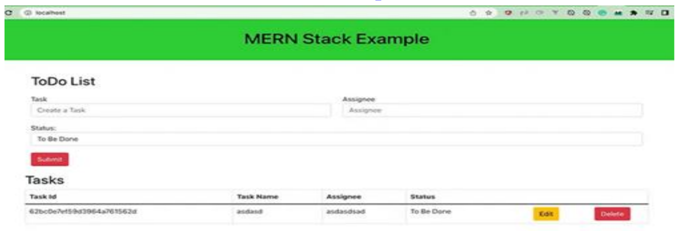
Let's build the image with the following command.
// build the imagedocker build -t mern-image
// check the imagesdocker images
Once the Docker image is built. You can run the image with the following command.
// run the containerdocker run -d -p 80:80 --name mern-stack mern-image
// list the containerdocker ps
// logsdocker logs mern-stack
// exec into running containerdocker exec -it mern-stack /bin/sh



```
bash-3.2$ docker run -d -p 80:80 --name mern-stack mern-image
09ef814bebad9917ec53b9be42a709f663f932401fc884ca3f12bb254c03cc87
bash-3.2$
bash-3.2$
bash-3.2$ docker ps
CONTAINER ID   IMAGE        COMMAND                  CREATED         STATUS         PORTS                NAMES
09ef814bebad   mern-image   "docker-entrypoint.s…"   3 seconds ago   Up 2 seconds   0.0.0.0:80->80/tcp   mern-stack
```

Figure 19: docker ps

You can access the application on the web at this address http://localhost



localhost

MERN Stack Example

ToDo List

Task: Create a Task
Assignee: Assignee
Status: To Be Done

Submit

Tasks

Task Id	Task Name	Assignee	Status		
62bc0e7ef59d3964a761562d	asdasd	asdasdsad	To Be Done	Edit	Delete

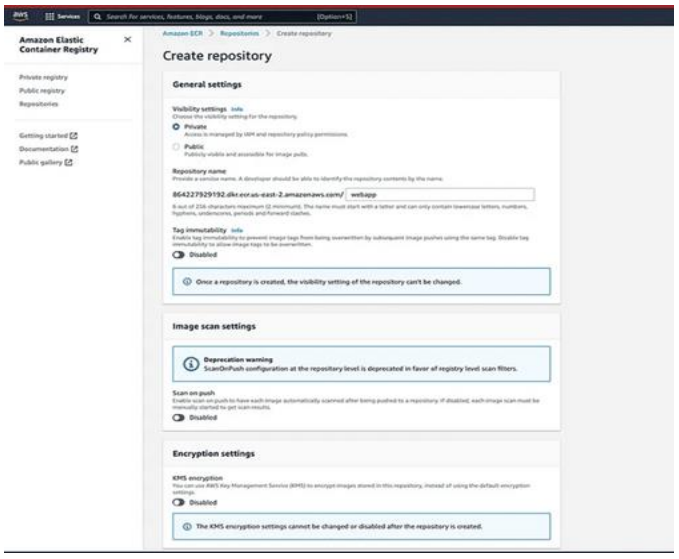
Figure 20: MERN Stack Running on port 80

8. Pushing Docker Image to ECR

Amazon Elastic Container Registry (ECR) is a fully-managed **Docker** (https://aws.amazon.com/docker/) container registry that makes it easy for developers to store, manage, and deploy Docker container images. Amazon ECR is integrated with **Amazon Elastic Container Service (ECS)** (https://aws.amazon.com/ecs/), simplifying your development to production workflow.
Amazon EKS works with Amazon ECR and ECR public. But, in this post, we see how we can use Amazon ECR to store our Docker images. Once you set up the Amazon account and create an IAM user with Administrator access the first thing you need to create a Docker repository.
You can create your first repository either by AWS console or AWS CLI

A. AWS Console

Creating a repository with an AWS console is straightforward and all you need to give a name.



aws
Services
Search for services, features, blogs, docs, and more [Option+S]
Amazon Elastic Container Registry
Private registry
Public registry
Repositories
Getting started
Documentation
Public gallery
Amazon ECR > Repositories > Create repository

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.

Repository name
Provide a concise name. A developer should be able to identify the repository contents by the name.
864227929192.dkr.ecr.us-east-2.amazonaws.com/ webapp
6 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

Tag immutability Info
Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.
Disabled

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

Image scan settings

Deprecation warning
ScanOnPush configuration at the repository level is deprecated in favor of registry level scan filters.

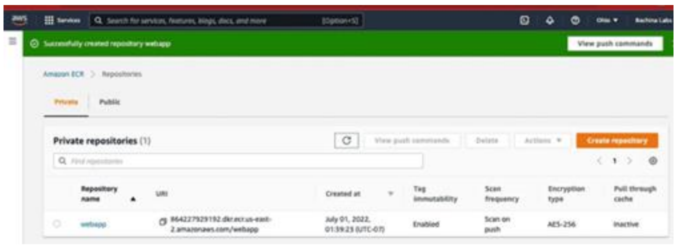
Scan on push
Enable scan on push to have each image automatically scanned after being pushed to a repository. If disabled, each image scan must be manually started to get scan results.
Disabled

Encryption settings

KMS encryption
You can use AWS Key Management Service (KMS) to encrypt images stored in this repository, instead of using the default encryption settings.
Disabled

The KMS encryption settings cannot be changed or disabled after the repository is created.

Figure 21: Creating a Repository



Services
Search for services, features, blogs, docs, and more [Option+S]
Ohio
Bachina Labs

Successfully created repository webapp

Private repositories (1)

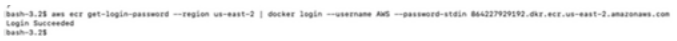
Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type	Pull through cache
webapp	864227929192.dkr.ecr.us-east-2.amazonaws.com/webapp	July 01, 2022, 01:39:23 (UTC-07)	Enabled	Scan on push	AES-256	Inactive

Figure 22: Repository

B. AWS CLI

The first thing you need to do is authenticate to your default registry. Here is the command to authenticate to your default registry. You need to make sure you are putting the correct regions and account id in the command.

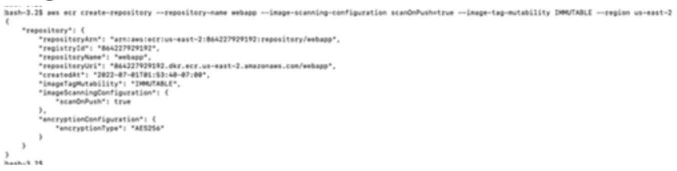
aws ecr get-login-password **--region us-east-2** | docker login --username AWS --password-stdin **aws_account_id**.dkr.ecr.**us-east-2**.amazonaws.com



```
bash-3.2$ aws ecr get-login-password --region us-east-2 | docker login --username AWS --password-stdin 864227929192.dkr.ecr.us-east-2.amazonaws.com
Login Succeeded
bash-3.2$
```

Figure 23: Authenticating to ECR

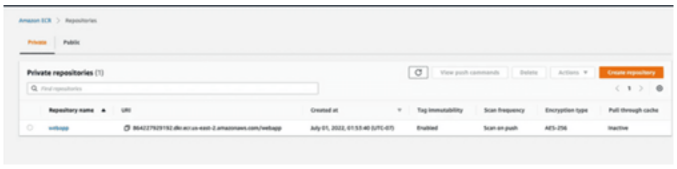
aws ecr create-repository --repository-name rest-api --image-scanning-configuration scanOnPush=true --image-tag-mutability IMMUTABLE --region us-east-2



```
bash-3.2$ aws ecr create-repository --repository-name webapp --image-scanning-configuration scanOnPush=true --image-tag-mutability IMMUTABLE --region us-east-2
{
    "repository": {
        "repositoryArn": "arn:aws:ecr:us-east-2:864227929192:repository/webapp",
        "registryId": "864227929192",
        "repositoryName": "webapp",
        "repositoryUri": "864227929192.dkr.ecr.us-east-2.amazonaws.com/webapp",
        "createdAt": "2022-07-01T01:53:40-07:00",
        "imageTagMutability": "IMMUTABLE",
        "imageScanningConfiguration": {
            "scanOnPush": true
        },
        "encryptionConfiguration": {
            "encryptionType": "AES256"
        }
    }
}
bash-3.2$
```

Figure 23: Creating a Repository

You will have the same result as well.



Private repositories (1)

Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type	Pull through cache
webapp	864227929192.dkr.ecr.us-east-2.amazonaws.com/webapp	July 01, 2022, 01:53:40 (UTC-07)	Enabled	Scan on push	AES-256	Inactive

Figure 24: Repository

C. Tagging your local Docker image and Pushing

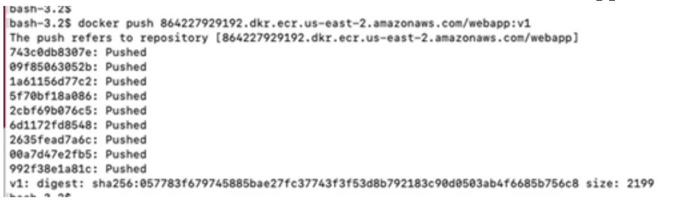
You created a Docker image on your local machine earlier. It's time to tag that image with this repository URI in the above image.

docker tag webapp:latest 864227929192.dkr.ecr.us-east-2.amazonaws.com/webapp:v1

Once you tag the image and it's time to push the Docker image into your repository.

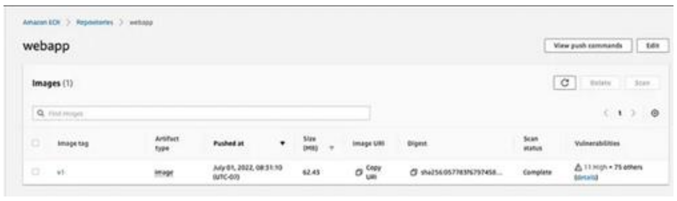
// list the imagesdocker images

// push the imagedocker push 864227929192.dkr.ecr.us-east-2.amazonaws.com/webapp:v1



```
bash-3.2$
bash-3.2$ docker push 864227929192.dkr.ecr.us-east-2.amazonaws.com/webapp:v1
The push refers to repository [864227929192.dkr.ecr.us-east-2.amazonaws.com/webapp]
743c0db8307e: Pushed
09f85063052b: Pushed
1a61156d77c2: Pushed
5f70bf18a086: Pushed
2cbf69b076c5: Pushed
6d1172fd8548: Pushed
2635fead7a6c: Pushed
00a7d47e2fb5: Pushed
992f38e1a81c: Pushed
v1: digest: sha256:057783f679745885bae27fc37743f3f53d8b792183c90d0503ab4f6685b756c8 size: 2199
```

Figure 25: Pushing Docker Image


Amazon ECR > Repositories > webapp

webapp

Images (1)

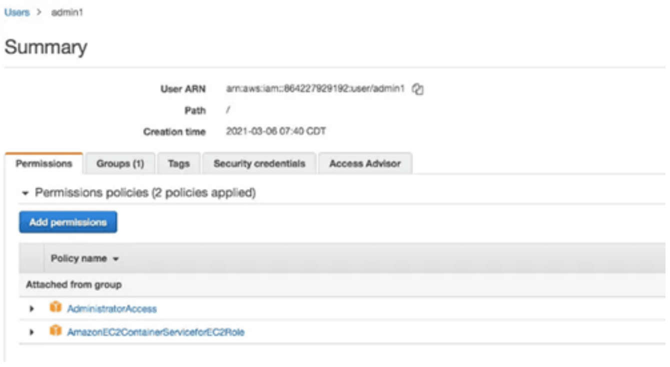
Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Scan status	Vulnerabilities
v1	Image	July 01, 2022, 08:51:10 (UTC-07)	62.45	Copy URI	sha256:0577837675 7458...	Complete	11 High • 75 others (details)

Figure 26: Docker Image Pushed to Repository

9. Deploying on AWS ECS

Amazon ECS makes it easy to deploy, manage, and scale Docker containers running applications, services, and batch processes. Amazon ECS places containers across your cluster based on your resource needs and is integrated with familiar features like Elastic Load Balancing, EC2 security groups, EBS volumes, and IAM roles. **You can explore more on the AWS documentation here** (https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html).

The Amazon ECS container agent makes calls to the Amazon ECS API on your behalf. Container instances that run the agent require an IAM policy and role for the service to know that the agent belongs to you. We need to add the **AmazonEC2ContainerServiceforEC2Role** policy to the user we created below.


Users > admin1

Summary

User ARN arn:aws:iam::864227929192:user/admin1
Path /
Creation time 2021-03-06 07:40 CDT

Permissions policies (2 policies applied)

Policy name

Attached from group
AdministratorAccess
AmazonEC2ContainerServiceforEC2Role

Figure 27: The policy added to the user we created

These are the objects of AWS ECS and how they are related.

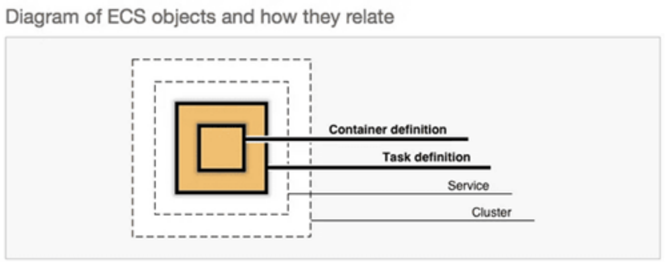
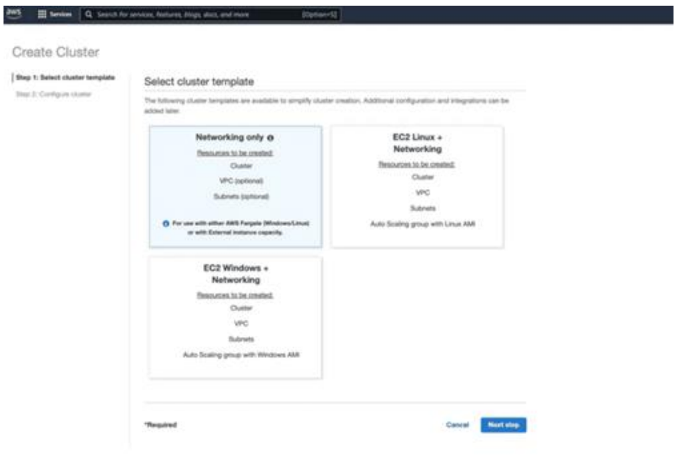


Figure 28: ECS objects

The first step is to create a cluster and you can select the Fargate one. We are launching using AWS using Fargate. AWS Fargate is a technology that you can use with Amazon ECS to run **containers** (https://aws.amazon.com/what-are-containers) without having to manage servers or clusters of Amazon EC2 instances. With AWS Fargate, you no longer have to provision, configure, or scale clusters of virtual machines to run containers. This removes the need to choose server types, decide when to scale your clusters or optimize cluster packing.



Create Cluster

Step 1: Select cluster template
Step 2: Configure cluster

Select cluster template

The following cluster templates are available to simplify cluster creation. Additional configuration and integrations can be added later.

Networking only
Resources to be created:
Cluster
VPC (optional)
Subnets (optional)
For use with either AWS Fargate (Windows/Linux) or with External instance capacity.

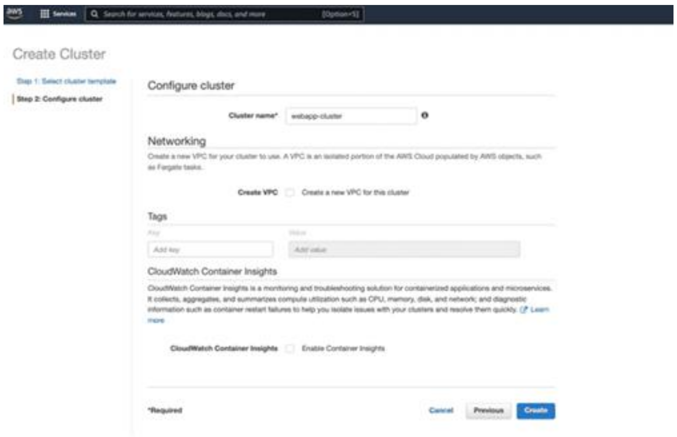
EC2 Linux + Networking
Resources to be created:
Cluster
VPC
Subnets
Auto Scaling group with Linux AMI

EC2 Windows + Networking
Resources to be created:
Cluster
VPC
Subnets
Auto Scaling group with Windows AMI

*Required

Figure 29: Creating a cluster

You need to give it a name and some other configuration if needed.



Create Cluster

Step 1: Select cluster template
Step 2: Configure cluster

Configure cluster

Cluster name* webapp-cluster

Networking

Create a new VPC for your cluster to use. A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Fargate tasks.

Create VPC ☐ Create a new VPC for this cluster

Tags

Key: Add key Value: Add value

CloudWatch Container Insights

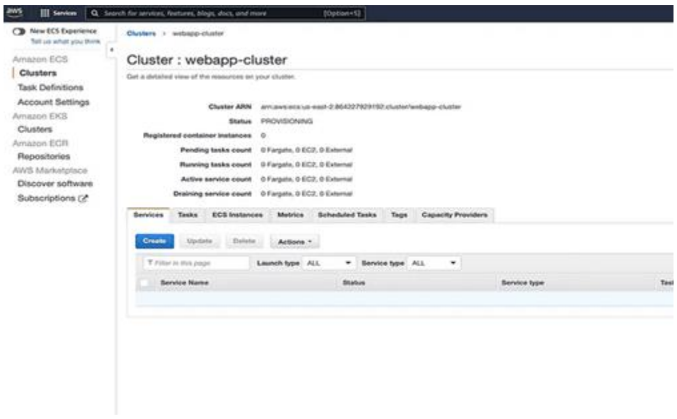
CloudWatch Container Insights is a monitoring and troubleshooting solution for containerized applications and microservices. It collects, aggregates, and summarizes compute utilization such as CPU, memory, disk, and network; and diagnostic information such as container restart failures to help you isolate issues with your clusters and resolve them quickly. Learn more

CloudWatch Container Insights ☐ Enable Container Insights

*Required

Figure 30: Webapp-Cluster

You can cluster as below once created.



Clusters > webapp-cluster

Cluster : webapp-cluster

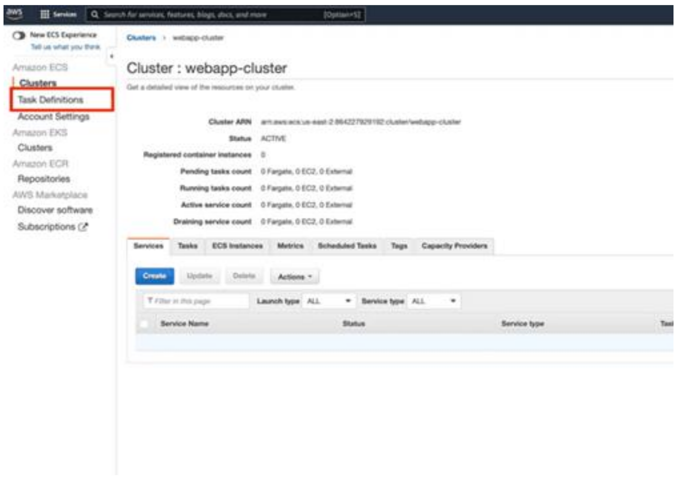
Get a detailed view of the resources on your cluster.

Cluster ARN	arn:aws:ecs:us-east-2:864207929192:cluster/webapp-cluster
Status	PROVISIONING
Registered container instances	0
Pending tasks count	0 Fargate, 0 EC2, 0 External
Running tasks count	0 Fargate, 0 EC2, 0 External
Active service count	0 Fargate, 0 EC2, 0 External
Draining service count	0 Fargate, 0 EC2, 0 External

Service Name	Status	Service type	Tas

Figure 31: Cluster Created

Now, we need to create a task definition so that we can deploy that as a service on this cluster. Let's click on the tasks tab and go to the task definitions page. Let's click on the hamburger icon on the left and click on the task definitions.


Clusters > webapp-cluster

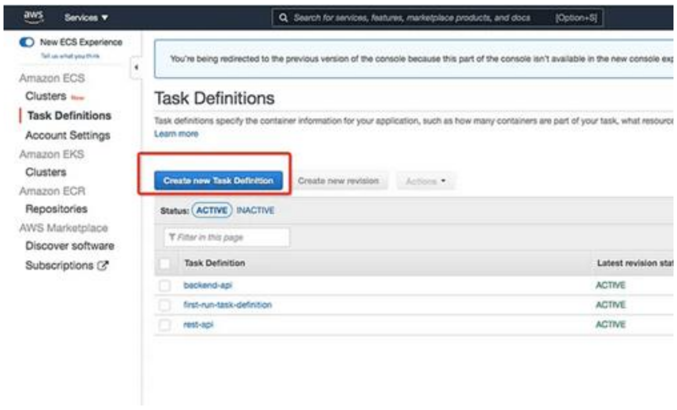
Cluster : webapp-cluster

Get a detailed view of the resources on your cluster.

Cluster ARN	arn:aws:ecs:us-east-2:864227929192:cluster/webapp-cluster
Status	ACTIVE
Registered container instances	0
Pending tasks count	0 Fargate, 0 EC2, 0 External
Running tasks count	0 Fargate, 0 EC2, 0 External
Active service count	0 Fargate, 0 EC2, 0 External
Draining service count	0 Fargate, 0 EC2, 0 External

Figure 32: Task Definitions

Let's click on the create task definition.


You're being redirected to the previous version of the console because this part of the console isn't available in the new console exp

Task Definitions

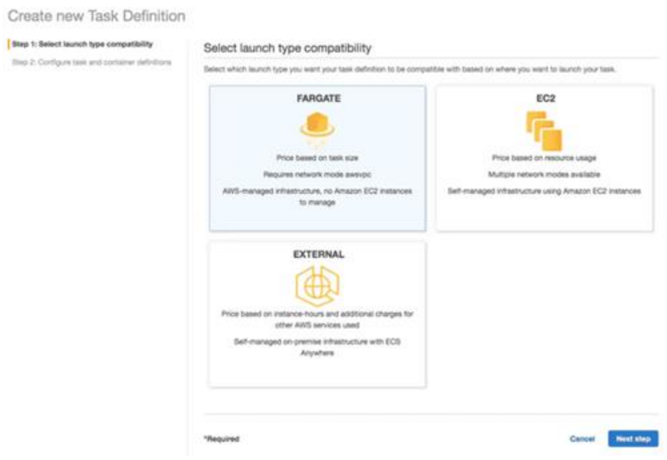
Task definitions specify the container information for your application, such as how many containers are part of your task, what resourc

Learn more

Task Definition	Latest revision stat
backend-api	ACTIVE
first-run-task-definition	ACTIVE
rest-api	ACTIVE

Figure 33: Creating Task Definition

Select the Fargate on the next screen.


Create new Task Definition

Step 1: Select launch type compatibility

Step 2: Configure task and container definitions

Select launch type compatibility

Select which launch type you want your task definition to be compatible with based on where you want to launch your task.

FARGATE
Price based on task size
Requires network mode awsvpc
AWS-managed infrastructure, no Amazon EC2 instances to manage

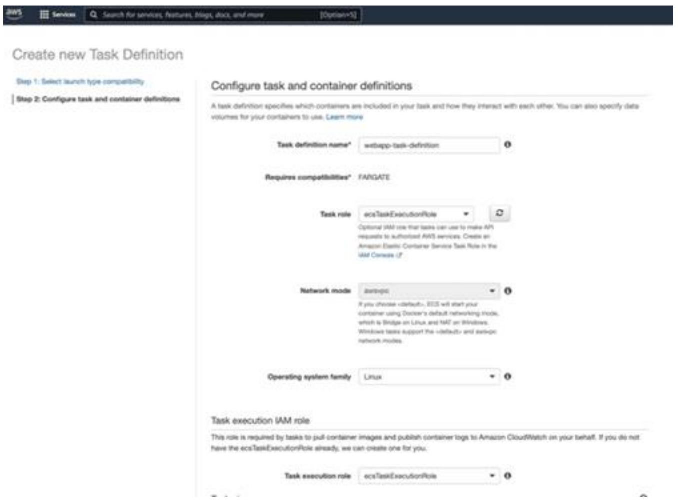
EC2
Price based on resource usage
Multiple network modes available
Self-managed infrastructure using Amazon EC2 instances

EXTERNAL
Price based on instance-hours and additional charges for other AWS services used
Self-managed on-premise infrastructure with ECS Anywhere

*Required

Figure 34: Fargate

Let's configure task definition on the next screen such as name, CPU, memory, etc.



Create new Task Definition

Step 1: Select launch type compatibility
Step 2: Configure task and container definitions

Configure task and container definitions

A task definition specifies which containers are included in your task and how they interact with each other. You can also specify data volumes for your containers to use. Learn more

Task definition name* webapp-task-definition

Requires compatibilities* FARGATE

Task role ecsTaskExecutionRole

Optional IAM role that tasks can use to make API requests to authorized AWS services. Create an Amazon Elastic Container Service Task Role in the IAM Console

Network mode awsvpc

If you choose <default>, ECS will start your container using Docker's default networking mode, which is Bridge on Linux and NAT on Windows. Windows tasks support the <default> and awsvpc network modes.

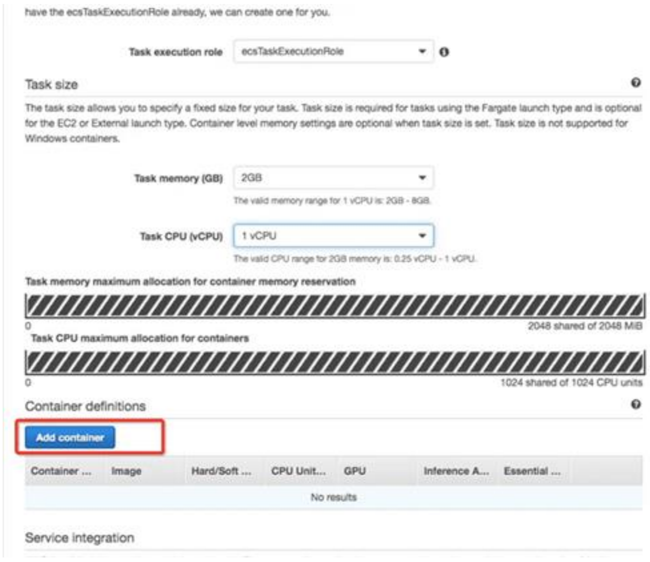
Operating system family Linux

Task execution IAM role

This role is required by tasks to pull container images and publish container logs to Amazon CloudWatch on your behalf. If you do not have the ecsTaskExecutionRole already, we can create one for you.

Task execution role ecsTaskExecutionRole

Figure 35: Configuring Task Definition



have the ecsTaskExecutionRole already, we can create one for you.

Task execution role ecsTaskExecutionRole

Task size

The task size allows you to specify a fixed size for your task. Task size is required for tasks using the Fargate launch type and is optional for the EC2 or External launch type. Container level memory settings are optional when task size is set. Task size is not supported for Windows containers.

Task memory (GB) 2GB

The valid memory range for 1 vCPU is: 2GB - 8GB.

Task CPU (vCPU) 1 vCPU

The valid CPU range for 2GB memory is: 0.25 vCPU - 1 vCPU.

Task memory maximum allocation for container memory reservation

0 2048 shared of 2048 MiB

Task CPU maximum allocation for containers

0 1024 shared of 1024 CPU units

Container definitions

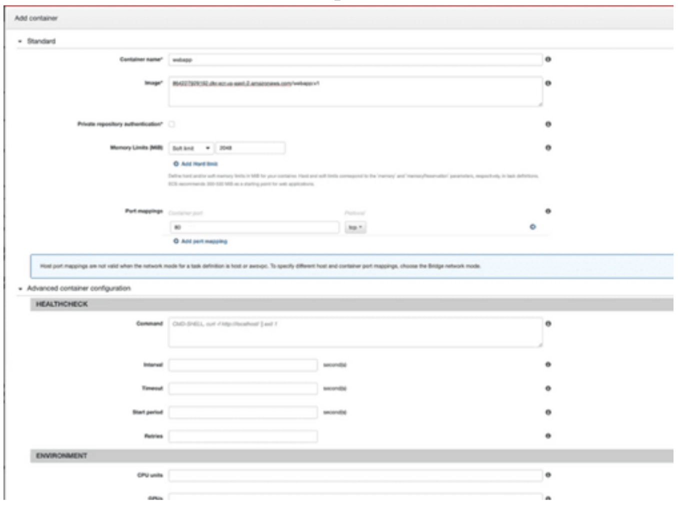
Add container

Container ...	Image	Hard/Soft ...	CPU Unit...	GPU	Inference A...	Essential ...
No results						

Service integration

Figure 36: Configuring Task Definition

You need to add container details from the ECR that we pushed in the above sections.



Add container

Standard

Container name* webapp

Image* [illegible].dkr.ecr.us-east-2.amazonaws.com/webapp:v1

Private repository authentication*

Memory Limits (MiB) Soft limit 2048

Add Hard limit

Define hard and/or soft memory limits in MiB for your container. Hard and soft limits correspond to the 'memory' and 'memoryReservation' parameters, respectively, in task definitions. ECS recommends 300-500 MiB as a starting point for web applications.

Port mappings Container port 80 Protocol tcp

Add port mapping

Host port mappings are not valid when the network mode for a task definition is host or awsvpc. To specify different host and container port mappings, choose the Bridge network mode.

Advanced container configuration

HEALTHCHECK

Command CMD-SHELL, curl -f http://localhost/ || exit 1

Interval second(s)

Timeout second(s)

Start period second(s)

Retries

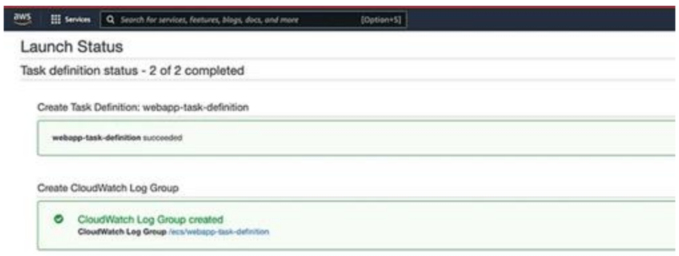
ENVIRONMENT

CPU units

GPUs

Figure 37: Adding a container

Once you add the container and hit the create button and you can see the status of the task definition on the next screen.



Launch Status

Task definition status - 2 of 2 completed

Create Task Definition: webapp-task-definition

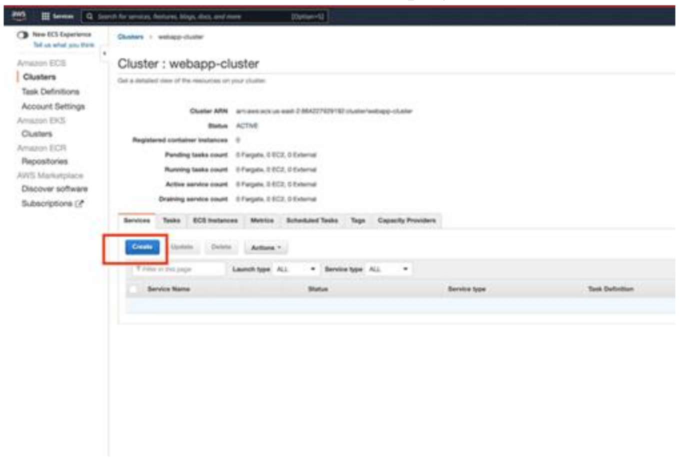
webapp-task-definition succeeded

Create CloudWatch Log Group

CloudWatch Log Group created
CloudWatch Log Group /ecs/webapp-task-definition

Figure 38: Launch Status

Let's go to the cluster that we created above and click on the deploy button under the service section.



Cluster : webapp-cluster

Get a detailed view of the resources on your cluster

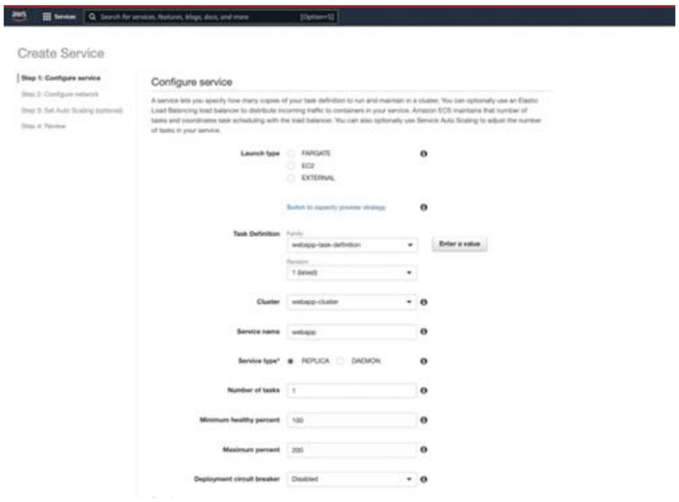
Cluster ARN	[illegible]
Status	ACTIVE
Registered container instances	0
Pending tasks count	0 Fargate, 0 EC2, 0 External
Running tasks count	0 Fargate, 0 EC2, 0 External
Active service count	0 Fargate, 0 EC2, 0 External
Draining service count	0 Fargate, 0 EC2, 0 External

Launch type ALL Service type ALL

Service Name	Status	Service type	Task Definition

Figure 39: Deploying the service

On the next screen, we need to select the same task definition that we have created above.


Services
Step 1: Configure service
Step 2: Configure network
Step 3: Set Auto Scaling (optional)
Step 4: Review
Switch to capacity provider strategy
Enter a value

Create Service

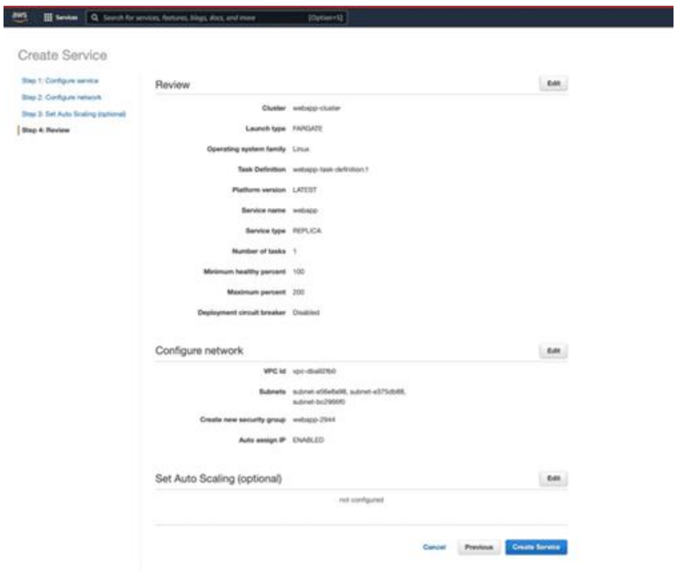
Configure service

A service lets you specify how many copies of your task definition to run and maintain in a cluster. You can optionally use an Elastic Load Balancing load balancer to distribute incoming traffic to containers in your service. Amazon ECS maintains that number of tasks and coordinates task scheduling with the load balancer. You can also optionally use Service Auto Scaling to adjust the number of tasks in your service.

Launch type	FARGATE / EC2 / EXTERNAL
Task Definition	Family: webapp-task-definition; Revision: 1 (latest)
Cluster	webapp-cluster
Service name	webapp
Service type*	[✓ REPLICA] DAEMON
Number of tasks	1
Minimum healthy percent	100
Maximum percent	200
Deployment circuit breaker	Disabled

Figure 40: Deployment Configuration

Finally, click on the deploy button.


Search for services, features, blogs, docs, and more
[Option+S]
Step 1: Configure service
Step 2: Configure network
Step 3: Set Auto Scaling (optional)
Step 4: Review
Edit
Cancel
Previous
Create Service

Create Service

Review

Cluster	webapp-cluster
Launch type	FARGATE
Operating system family	Linux
Task Definition	webapp-task-definition:1
Platform version	LATEST
Service name	webapp
Service type	REPLICA
Number of tasks	1
Minimum healthy percent	100
Maximum percent	200
Deployment circuit breaker	Disabled

Configure network

VPC Id	vpc-dba02fb0
Subnets	subnet-a56e8a98, subnet-a375db88, subnet-bc2966f0
Create new security group	webapp-2944
Auto assign IP	ENABLED

Set Auto Scaling (optional)

not configured

Figure 41: Create Service

Once you click on the Create Service button, you can see the status below.

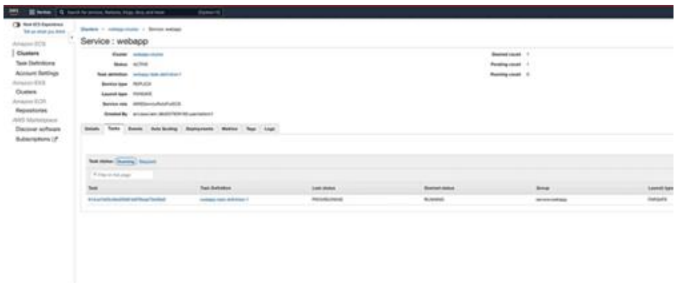
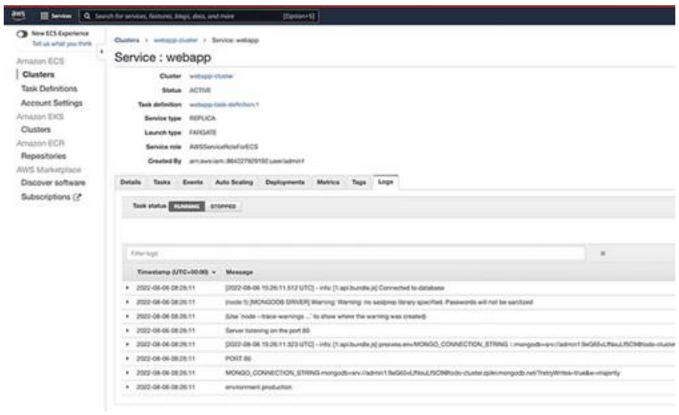


Figure 42: Deployment Successful

10: Access the WebApp from the browser

We know the web app runs on port 80 from the logs. You can see the logs when you click on the service and go to the details page.



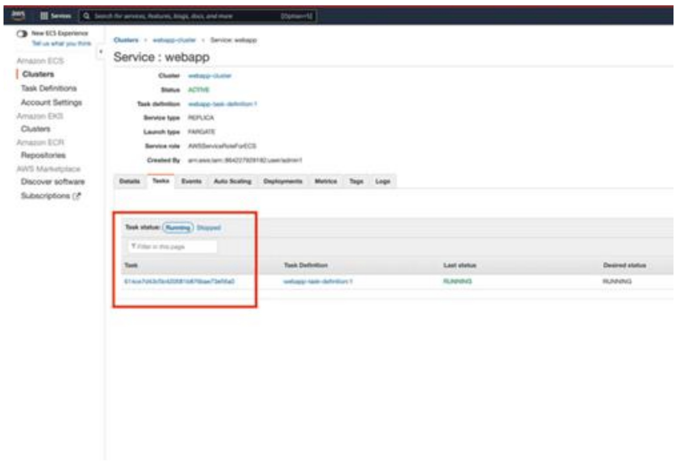
Service : webapp

Cluster	webapp-cluster
Status	ACTIVE
Task definition	webapp-task-definition:1
Service type	REPLICA
Launch type	FARGATE
Service role	AWSServiceRoleForECS
Created By	arn:aws:iam::864327929192:user/admin1

```
Timestamp (UTC+00:00)   Message
2022-08-06 08:26:11     [2022-08-06 15:26:11.512 UTC] - info: [1.api.bundle.js] Connected to database
2022-08-06 08:26:11     (node:1) [MONGODB DRIVER] Warning: Warning: no saslprep library specified. Passwords will not be sanitized
2022-08-06 08:26:11     (Use `node --trace-warnings ...` to show where the warning was created)
2022-08-06 08:26:11     Server listening on the port 80
2022-08-06 08:26:11     [2022-08-06 15:26:11.323 UTC] - info: [1.api.bundle.js] process.env.MONGO_CONNECTION_STRING : mongodb+srv://admin1:[illegible]@todo-cluster
2022-08-06 08:26:11     PORT 80
2022-08-06 08:26:11     MONGO_CONNECTION_STRING mongodb+srv://admin1:[illegible]@todo-cluster.zpkn.mongodb.net/?retryWrites=true&w=majority
2022-08-06 08:26:11     environment production
```

Figure 43: WebApp Logs

Now, we need to find the public IP Address of the service. You need to go to the Configuration and tasks tab and click on the specific task definition that is running your ECR Image.



Clusters > webapp-cluster > Service: webapp

Service : webapp

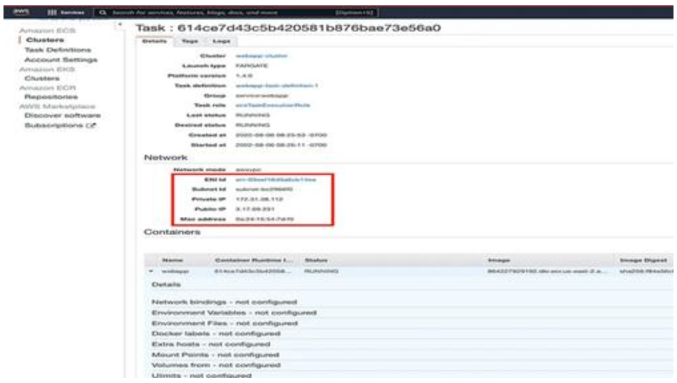
Cluster	webapp-cluster
Status	ACTIVE
Task definition	webapp-task-definition:1
Service type	REPLICA
Launch type	FARGATE
Service role	AWSServiceRoleForECS
Created By	arn:aws:iam::[illegible]:user/admin1

Task status: Running Stopped

Task	Task Definition	Last status	Desired status
614ce7d43c5b420581b876bae73e56a0	webapp-task-definition:1	RUNNING	RUNNING

Figure 44: Tasks

Once you are on the task page you can take the public IP address from there or you can go to the particular EC2 instance.



Task : 614ce7d43c5b420581b876bae73e56a0

Cluster	webapp-cluster
Launch type	FARGATE
Platform version	1.4.0
Task definition	webapp-task-definition:1
Group	service:webapp
Task role	ecsTaskExecutionRole
Last status	RUNNING
Desired status	RUNNING
Created at	2022-08-06 08:25:53 -0700
Started at	2022-08-06 08:26:11 -0700

Network

Network mode	awsvpc
ENI Id	eni-[illegible]
Subnet Id	subnet-[illegible]
Private IP	172.31.28.112
Public IP	3.17.69.231
Mac address	[illegible]

Containers

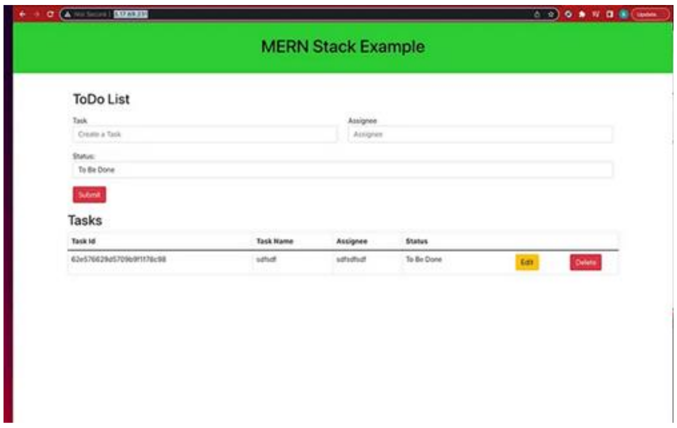
Name	Container Runtime I...	Status	Image	Image Digest
webapp	614ce7d43c5b42058...	RUNNING	[illegible].dkr.ecr.us-east-2.a...	sha256:[illegible]

Details

Network bindings - not configured
Environment Variables - not configured
Environment Files - not configured
Docker labels - not configured
Extra hosts - not configured
Mount Points - not configured
Volumes from - not configured
Ulimits - not configured

Figure 45: Public IPAddress

Now, we know the WebApp runs on port 80 and you can access it with the below address.
http://3.17.69.231/



MERN Stack Example

ToDo List

Task	Assignee
Create a Task	Assignee

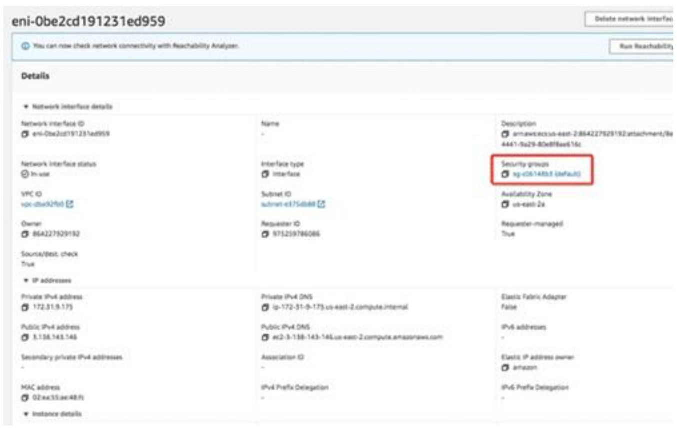
Status:
To Be Done

Tasks

Task Id	Task Name	Assignee	Status		
62e576629d5709b9f1f76c98	sdfsdf	sdfsdfsdf	To Be Done	Edit	Delete

Figure 46: Accessing the WebApp

One thing you need to do is that you have to add an inbound rule for all the traffic. Let’s click on the ENI Id and go to the Security Groups and add an inbound rule as below.


eni-0be2cd191231ed959

You can now check network connectivity with Reachability Analyzer.

Details

▼ Network interface details

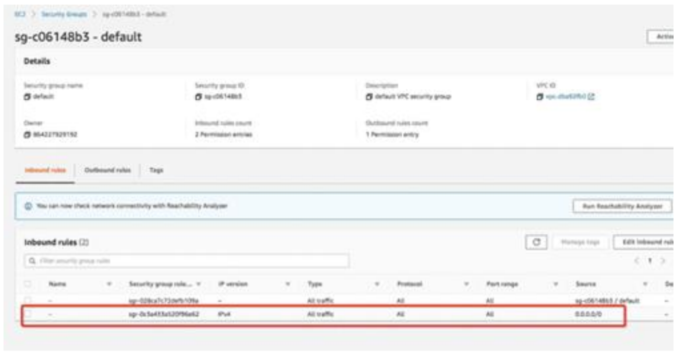
Network interface ID eni-0be2cd191231ed959	Name -	Description arn:aws:ecs:us-east-2:864227929192:attachment/8e 4441-9a29-80e8f8ae616c
Network interface status In-use	Interface type interface	Security groups sg-c06148b3 (default)
VPC ID vpc-dba92fb0	Subnet ID subnet-e375db88	Availability Zone us-east-2a
Owner 864227929192	Requester ID 975259786086	Requester-managed True
Source/dest. check True		

▼ IP addresses

Private IPv4 address 172.31.9.175	Private IPv4 DNS ip-172-31-9-175.us-east-2.compute.internal	Elastic Fabric Adapter False
Public IPv4 address 3.138.143.146	Public IPv4 DNS ec2-3-138-143-146.us-east-2.compute.amazonaws.com	IPv6 addresses -
Secondary private IPv4 addresses -	Association ID -	Elastic IP address owner amazon
MAC address 02:ea:55:ae:48:fc	IPv4 Prefix Delegation -	IPv6 Prefix Delegation -

▼ Instance details

Figure 47: Security Groups


EC2 > Security Groups > sg-c06148b3 - default

sg-c06148b3 - default

Details

Security group name default	Security group ID sg-c06148b3	Description default VPC security group	VPC ID vpc-dba92fb0
Owner 864227929192	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry	

You can now check network connectivity with Reachability Analyzer

Inbound rules (2)

Name	Security group rule...	IP version	Type	Protocol	Port range	Source
-	sgr-028ca7c72defb109a	-	All traffic	All	All	sg-c06148b3 / default
-	sgr-0c5a433a520f96a62	IPv4	All traffic	All	All	0.0.0.0/0

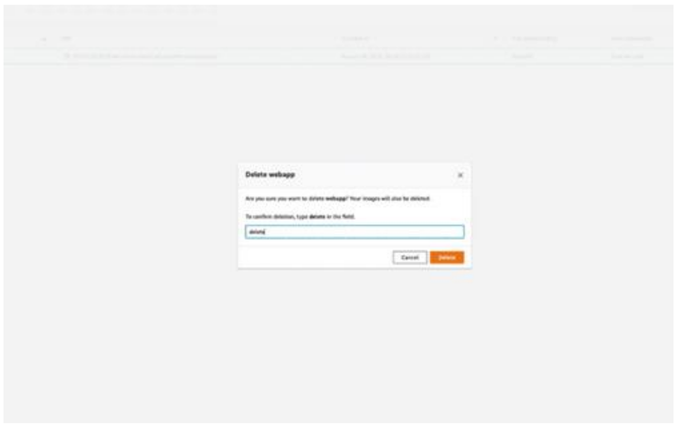
Figure 48: InBound Rule

11. Cleaning Up

We need to clean up all the resources that we used here if you don't want to incur any extra charges.

A. ECR

We need to remove the repository in the ECR.


Delete webapp

Are you sure you want to delete **webapp**? Your images will also be deleted.

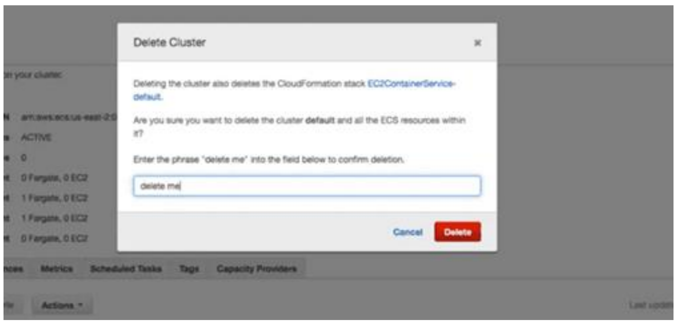
To confirm deletion, type **delete** in the field.

delete

Figure 49: Deleting the repository

B. ECS

We need to delete the cluster on the ECS.


Delete Cluster

Deleting the cluster also deletes the CloudFormation stack EC2ContainerService-default.

Are you sure you want to delete the cluster **default** and all the ECS resources within it?

Enter the phrase "delete me" into the field below to confirm deletion.

delete me

Figure 50: Deleting Cluster

12: Summary

- Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast, container management service that makes it easy to run, stop, and manage Docker containers on a cluster of Amazon EC2 instances.
- There are two launch types: EC2 and Fargate
- With AWS Fargate, you no longer have to provision, configure, or scale clusters of virtual machines to run containers.
- We need to add **AmazonEC2ContainerServiceforEC2Role** policy to the user so that The Amazon ECS container agent makes calls to the Amazon ECS API on your behalf.
- You can use any Docker image registry such as Docker Hub, AWS ECR, etc.
- Clean up all the resources after you practice this.

13. Conclusion

In this article, we've explored the process of deploying the MERN Stack on AWS ECS Fargate. Moving forward, subsequent discussions will delve into additional aspects such as automating deployments, configuring environment variables, establishing custom domain links, and other relevant configuration procedures. These forthcoming insights will provide a comprehensive understanding of deploying and managing MERN Stack applications on AWS ECS Fargate, contributing valuable insights to the field of cloud-based application deployment and management.

References

[1]. AWS Docs for ECS https://docs.aws.amazon.com/ecs/
[2]. Official Docker Guides https://docs.docker.com/get-started/overview/
[3]. Official React Documentation https://react.dev/
[4]. Official MongoDB Docs https://www.mongodb.com/docs/