

Deploying MongoDB using Docker with authentication enabled



Vishal Lokam · [Follow](#)

6 min read · Sep 19, 2023



3



1



What is MongoDB?

MongoDB is a free and open-source cross-platform document-oriented database program. Classified as a NoSQL database program. MongoDB uses JSON-like documents.

That being said, let's start this journey by learning about the basics required to run MongoDB in a Docker container.

Basics

To start a basic MongoDB server instance, run the following command:

```
docker container run --name some-mongo -d mongo:tag
```

where **some-mongo** will be the name of the container, and **tag** will specify the MongoDB version. To learn more about the MongoDB docker image tags check [here](#)

Setting up Volumes

Volumes are the preferred way to store data persistently for docker containers. Volumes are completely managed by Docker. Even if the container goes down for some reason and a new container takes its place, the new container would still be able to access all the persisted data.

```
docker run --name some-mongo -v mymongodata:/data/db -d mongo
```

Here **-v** option indicates which volume to use. In case the volume is not present, a new volume is created.

Enabling Authentication

To start MongoDB instance in authentication mode automatically, pass environment variables `MONGO_INITDB_ROOT_USERNAME` , `MONGO_INITDB_ROOT_PASSWORD` while initializing the instance.

The above variables will create a new user and set the password in `admin` authentication database and given the role of `root` , which is a "superuser" role.

```
docker container run -d --network some-network --name some-mongo -e MONGO_INITDB
```

Authentication in MongoDB is fairly complex, so complex user setup is done using `/docker-entrypoint-initdb.d/` . We will learn about this shortly.

At this point, login into the mongo shell inside the container is possible and then users can be manually added. But it defeats the purpose of using Docker to deploy MongoDB.

In case one needs to login into the mongo shell inside the MongoDB container one can use the below command to create another container that will connect to the MongoDB instance.

```
docker run -it --rm --network some-network mongo mongosh --host some-mongo -u mo
```

Here, after the `-u` option, pass the username that was created in the earlier section and `-p` would be the password that was created in the earlier section

This container will automatically delete once the user exits out of the shell because of the `--rm` flag.

Create a shell script to add users

To add users automatically apart from root users when a new instance of MongoDB is created, a shell/JavaScript is required. Create a shell script which can be named anything. I have named it `entrypoint.sh`. This script will login into the mongo shell and create new users.

```
#!/usr/bin/env bash
echo "Creating mongo users..."
mongosh --authenticationDatabase admin --host localhost -u mongoadmin -p mongopa
echo "Mongo users created."
```

Lets break down the above shell script.

`#!/usr/bin/env bash` uses the environment variable present in the system, then executes commands with a defined interpreter in our case `bash`.

`mongosh` JavaScript and Node.js 16.x REPL environment for interacting with MongoDB deployments. It comes Pre-installed with the official Mongo image.

`--authenticationDatabase admin` Database to search for the user. In our case `admin`.

`--host localhost` as we are connecting on the same instance

`-u mongoadmin -p mongopasswd` super user/root user created during instance initialization along with the password.

app_db_name database name for which user needs to be created.

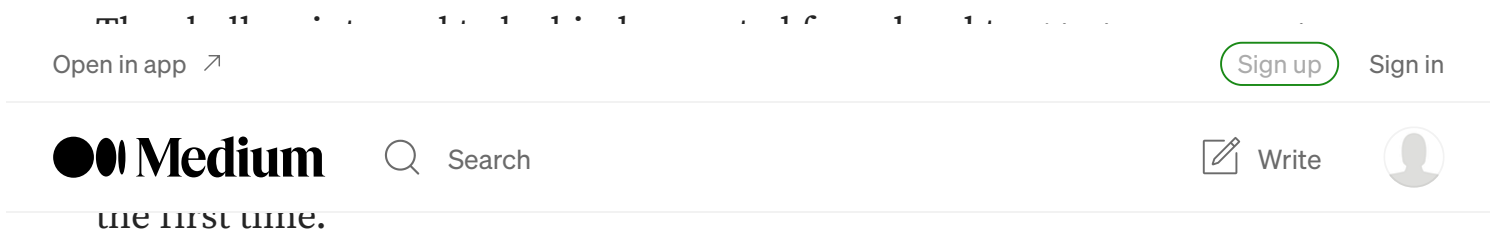
`--eval` An option that allows the shell to evaluate JavaScript.

```
db.createUser({user: 'devUser', pwd: 'devUserPass', roles: [{role:
'readWrite', db: 'app_db_name '}]}); creates a new user devUser and assign
a password devUserPass with a role role: 'readWrite' for a specific database
db: 'app_db_name '
```

For more information on built-in roles check out [Built-in roles](#)

Using the Shell script

Put the shell script in a folder. I have named the folder as `mongo-entrypoint` but any name is fine given that it is the same name used as the bind mount during deployment in the later section. Bind mounting will also help in case we want to create multiple scripts.



Now lets look at the completed docker container that included everything that we have learned

Complete Docker command

```
docker container run -p 27017:27017 --network todolist-network --name todolist-m
```

-p 27017:27017 publish port 27017 on host(left side of the :) and route data to the port 27017 on the container (right side of the :)

--network todolist-network create a new network todolist-network

--name todolist-mongodb name of the container

-e MONGO_INITDB_ROOT_USERNAME="mongoadmin" initialize root user

-e MONGO_INITDB_ROOT_PASSWORD="mongopasswd" initialize root user password

-v mongodbddata:/data/db Volume for storing data persistently

-v "\$(pwd)/mongo-entrypoint:/docker-entrypoint-initdb.d" bind mounting the user creation script

mongo:5.0 MongoDB version

Create a Docker compose file

Let's create a docker compose file for easier usage. This compose file will have the same settings that are mentioned in the above docker command.

```
version: "3.8"
services:
  mongo-db:
    image: mongo:5.0
    container_name: mongoDB
    ports:
      - 27017:27017
    environment:
      MONGO_INITDB_ROOT_USERNAME: mongoadmin
      MONGO_INITDB_ROOT_PASSWORD: mongopasswd
    volumes:
      - mymongodata:/data/db
```

```
- type: bind
  source: ./mongo-entrypoint
  target: /docker-entrypoint-initdb.d/

volumes:
  mymongodata:
```

This code will run the same container as the Docker command shown above. We can run the compose file using command `docker compose up`

Need for adding health checks for the MongoDB container

One problem that usually occurs is when another container depends on the new user created by the shell script. The problem is that Docker will spin up the MongoDB container with root user and password. As soon as this is done, Docker indicate that the container is successfully online.

This will indicate all the other containers which are dependent on MongoDB container to start deployment.

However, the users that are supposed to be created by the script are not yet created. Let's say we have another container that is dependent on the MongoDB container but it will use the authentication of the new user that will be created by the script.

This new container will crash because database authentication with MongoDB will fail as the new user is not yet created in the MongoDB container.

There are two ways to fix this issue.

1. To ensure that even if the container crashes, it still constantly attempts to deploy a new container and reconnect with MongoDB container until the

new users are created, apply the policy `restart: always`. Note that this is not an ideal solution.

2. Ideal solution would be to add a health check to MongoDB, checkout the code below. The health check will verify whether a new user has been created or not. Below I have a NodeJS application that will be deployed only when the health check on the MongoDB container is successful. This application is looking for user `devUser` in `todolistDB` database. So the user in the script should be added accordingly

```
version: "3.8"

services:
  todolist-app:
    image: vishallokam/todolist-docker:latest
    container_name: todolist_app
    ports:
      - '3000:3000'
    environment:
      MONGODB_SERVER_URL: "mongoDB:27017"
      DB_USER: devUser
      DB_PASS: devUserPass
    depends_on:
      mongo-db:
        condition: service_healthy

  mongo-db:
    image: mongo:5.0
    container_name: mongoDB
    ports:
      - 27017:27017
    environment:
      MONGO_INITDB_ROOT_USERNAME: mongoadmin
      MONGO_INITDB_ROOT_PASSWORD: mongopasswd
    volumes:
      - mymongodata:/data/db
      - type: bind
        source: ./mongo-entrypoint
        target: /docker-entrypoint-initdb.d/
    healthcheck:
      test: echo 'db.runCommand({find:"app_db_name.devUser"}).ok' | mongosh --au
      interval: 10s
      timeout: 10s
      retries: 3
```



```
start_period: 20s

volumes:
  mymongodata:
```

Let us look at the health check more closely

```
healthcheck:
  test: echo 'db.runCommand({find:"app_db_name.devUser"}).ok' | mongosh --au
  interval: 10s
  timeout: 60s
  retries: 3
  start_period: 20s
```

Health check in MongoDB is implemented using the above code. It will check if the user is created or not.

test: This will run the command that will check if the new user is created or not.

interval: determines the interval between two checks.

timeout: If the HEALTHCHECK command exceeds the specified duration, it is categorized as a failure. The default duration is the 30s.

retries: If it reaches the specified number of retries, the state is considered to be unhealthy.

start_period: time interval after which start the health check.

Now let us look at the application container.

```
depends_on:  
  mongo-db:  
    condition: service_healthy
```

Application container has the above policy. Application container will only be deployed when the new users are created.

So there you have it — the ins and outs of deploying MongoDB to containers with authentication enabled. Until next time...

Thank you for reading, please follow for more such content 📖.

Docker

Mongodb

DevOps

Containerization




Written by Vishal Lokam

1 Follower

Follow



More from Vishal Lokam

 Vishal Lokam

What is GitHub Actions: A Beginner's Guide on automating workflows(Part 1)

If your code resides on GitHub or if you plan to store it there, it's highly beneficial to explore GitHub Actions as a CI/CD solution

4 min read · Oct 4, 2023




See all from Vishal Lokam



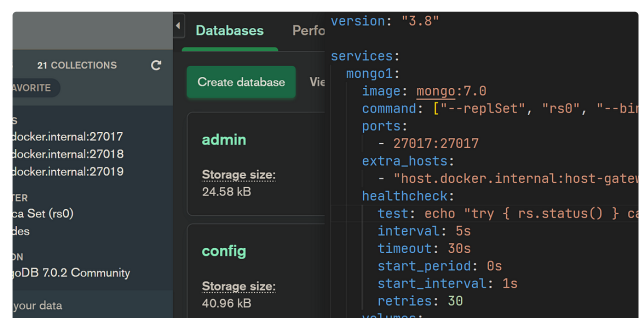
Recommended from Medium

nongol

 Sumanta Mukhopadhyay

MongoDB: A Developer's Quick Primer

MongoDB is a popular NoSQL database that offers a flexible schema, high availability, an...



 Anthony Simmon in Workleap

The only local MongoDB replica set with Docker Compose guide you'll...

Learn to run a local MongoDB replica set using Docker Compose. Enable access to...

2 min read · Oct 28, 2023



10



147



6



Lists



Coding & Development

11 stories · 445 saves



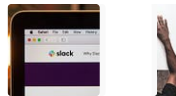
General Coding Knowledge

20 stories · 912 saves



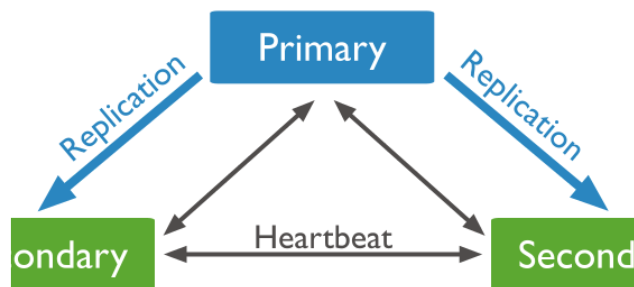
Natural Language Processing

1189 stories · 664 saves



Productivity

237 stories · 323 saves



Ojo Joseph

MongoDB Replica Set with Docker-compose

This post walks you through how to set up a Mongo db replica set with docker-compose...

2 min read · Jan 14, 2024



71



142



Johannes Johansson

Implementing the pgvector extension for a PostgreSQL...

If you're looking to implement vectorized search using PostgreSQL and Python, you're...

3 min read · Sep 15, 2023



Muhammad naufal adli in Dev Genius

How to write good Unit Testing for iOS Apps in SwiftUI

Unit testing is a crucial aspect of software development, ensuring that each componen...

★ · 3 min read · 4 days ago



160



2



3



1



See more recommendations



Harshvardhan Singh

MySQL on Docker

(bonus MySQL 8 and 5.7 on M1/M2 mac)

2 min read · Sep 10, 2023