ASP.NET WEB API FOR MONGODB AND KUBERNETES WITH IDENTITY AND JWT AUTHENTICATION

TECHNICAL DOCUMENT AS A PROOF OF CONCEPT CONTAINING THE OBTAINED I+D CONCLUSIONS OF THE DESCRIBED SOLUTION

Contents

Doc. Version	Date	Author	Description
1.0	2025-12-24	https://github.com/kraftcoding	First release of the solution

Requirements

The next table shows the requirements for the solution described.

Requirement	Description
RQ-001 | ASP.NET web API (.Net 8)	Framework (.Net 8) based Web API to authenticate and manage books
RQ-002 | Web API Authentication and Authorization	JWT and Identity based authentication and authorization procedures
RQ-003 | MongoDB	NoSQL storage
RQ-004 | Docker	Docker to run Kubernetes
RQ-005 | Kubernetes	Containerization sytem

Overview

This document contains the technical details of the solution that will show you how to build a web API using ASP.Net Core (.Net 8) that can perform CRUD (Create, Read, Update and Delete) operations on a MongoDB NoSQL database.

To ensure security, we will protect the API using JWT (JSON web tokens) authentication. In addition, we will demonstrate how to store user credentials in a MongoDB database by using Asp.NetCore Authentication and Identity.

Finally we will containerize and deploy the MongoDB service using Docker and Kubernetes.

Scope

The scope of this document is to offer a proof of concept with a detailed explain of how to building modern, stateless back-end applications and services on distributed platforms offering scalability and resiliency to your application, using the technologies described before.

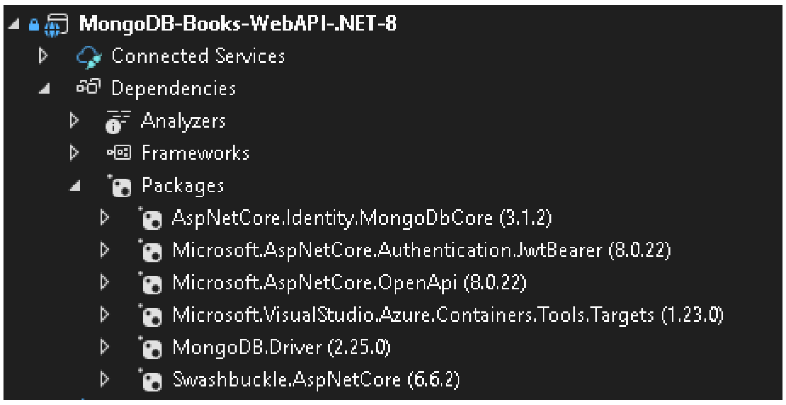
RQ-01 | ASP.NET web API

NOTE: All about JWT and Identity based authentication and authorization is discussed separately.

A Web API or Web Service API is an application processing interface between a web server and web browser. All web services are APIs but not all APIs are web services. REST API is a special type of Web API that uses the standard architectural style explained above.

To deploy a web API for this purpose, we run Visual Studio and create a new project, we select the "ASP.NET Core Web API" template. We set the Project name "BookStoreApi" and the solution path. We select the Framework (.Net 8) and rest of the options: Configure https, use controllers and enable openAPI support.

After the solution is created, add the NuGet libraries:



```
MongoDB-Books-WebAPI-.NET-8
    Connected Services
    Dependencies
        Analyzers
        Frameworks
        Packages
            AspNetCore.Identity.MongoDbCore (3.1.2)
            Microsoft.AspNetCore.Authentication.JwtBearer (8.0.22)
            Microsoft.AspNetCore.OpenApi (8.0.22)
            Microsoft.VisualStudio.Azure.Containers.Tools.Targets (1.23.0)
            MongoDB.Driver (2.25.0)
            Swashbuckle.AspNetCore (6.6.2)
```

We add the entity model, we create a "Models" folder/directory to the project root. We add a "Book.cs" class to the "Entities" directory/folder.

Model

In the "Book" class, we require the "Id" property for mapping the object to the MongoDB collection. We use the [BsonId] annotation to make this property the document's primary key. We also use the [BsonRepresentation(BsonType.ObjectId)] annotation to allow passing the parameter as type string instead of an ObjectId structure.

Mongo handles the conversion from string to ObjectId. The BookName property is annotated with the [BsonElement] attribute. The attribute's value of "Name" represents the property name in the MongoDB collection. It is optional to use this attribute to rename the C# model property to a MongoDB column name.

We configure the MongoDb connection information in the "appsettings.json" file. We set the connection string, the database name and the books collectionname:

- "mongodb://username:password@localhost:27017/", "DatabaseName":"Bookstore", "BookCollectionName": "Books".

  NOTE: It is needed to forward ports to make it available.

We add a “BookStoreDbSettings.cs” class to the “Configuration” directory/folder. We configure the “BookStoreSettings” in the middleware “Program.cs” file. Now we add a CRUD operations to the project. We add a “Services” directory/folder to the project root. Inside this folder we implement the repository interface and class. We create the “IbookRepository.cs” interface. We implement the interface with the “MongoDbBooksRepository.cs” class. We configure the repository in the middleware.

Controller

Finally, we add the controller.

For this purpose, we create the “BooksController.cs” class inside the “Controllers” directory/folder. The controller includes the “IBooksRepository” dependency injection. By means of this service we access the functions (GetAsync, CreateAsync, UpdateAsync, RemoveAsync).

Now we can interchange information with the MongoDb database.

Tests

To test the web API during development we are going to use Swagger which offers a web-based UI OpenAPI specification that provides information about the API functions.

Port forwarding is needed to connect to database (see MongoDB section).

Postman HTTP Request collection is added.

RQ-02 | Web API authentication and authorization

Authentication is the process of verifying user credentials, while authorization is the process of determining whether a user has permission to access specific modules within an application. In order to secure an ASP.NET Core Web API application, we can implement JWT (JSON web token) authentication.

Additionally, we can use authorization in ASP.NET Core to grant access to various functionalities of the application based on a user's privileges. To store user credentials securely, we utilize a MongoDB database.

JSON web token (JWT) is an open standard (RFC 7519) that defines a secure and compact way to transmit information between different parties as a JSON object.

Since JWTs are digitally signed, the transmitted data can be trusted and verified. JWTs can be signed using a secret key (with the HMAC algorithm) or a public/private key pair using RSA or ECDSA.

The compact form of a JSON web token consists of three parts separated by dots (.), namely the Header, Payload, and Signature. This structure makes JWTs easily transferable and straightforward to process. Typically, a JWT looks like this: xxxx.yyyy.zzzz.

Implementation

We install the Nugget packages: “AspNetCore.Identity.MongoDbCore” and “Microsoft.AspNetCore.Authentication.JwtBearer”.

After loading the Nugget packages, we start our implementation creating a new directory/folder called “Authorization”, where to store all the models to manage the authentication and authorization process. We first define our Identity “User” and “Role” classes.

The Identity User class inherits from “MongoIdentityUser” class and is mapped to “Users” collections. The Identity Role class inherits from “MongoIdentityRole” class and is mapped to “Roles” collections. Also, we create a static class “UserRoles” to add two constant values “Admin” and “User” as roles. You can add as many roles as you wish.

We create a “User.cs” class for new user registration. In this class we store the personal information for each user (Name, Email and Password). We store the login user credentials (name and password) inside the “LoginModel.cs” class.

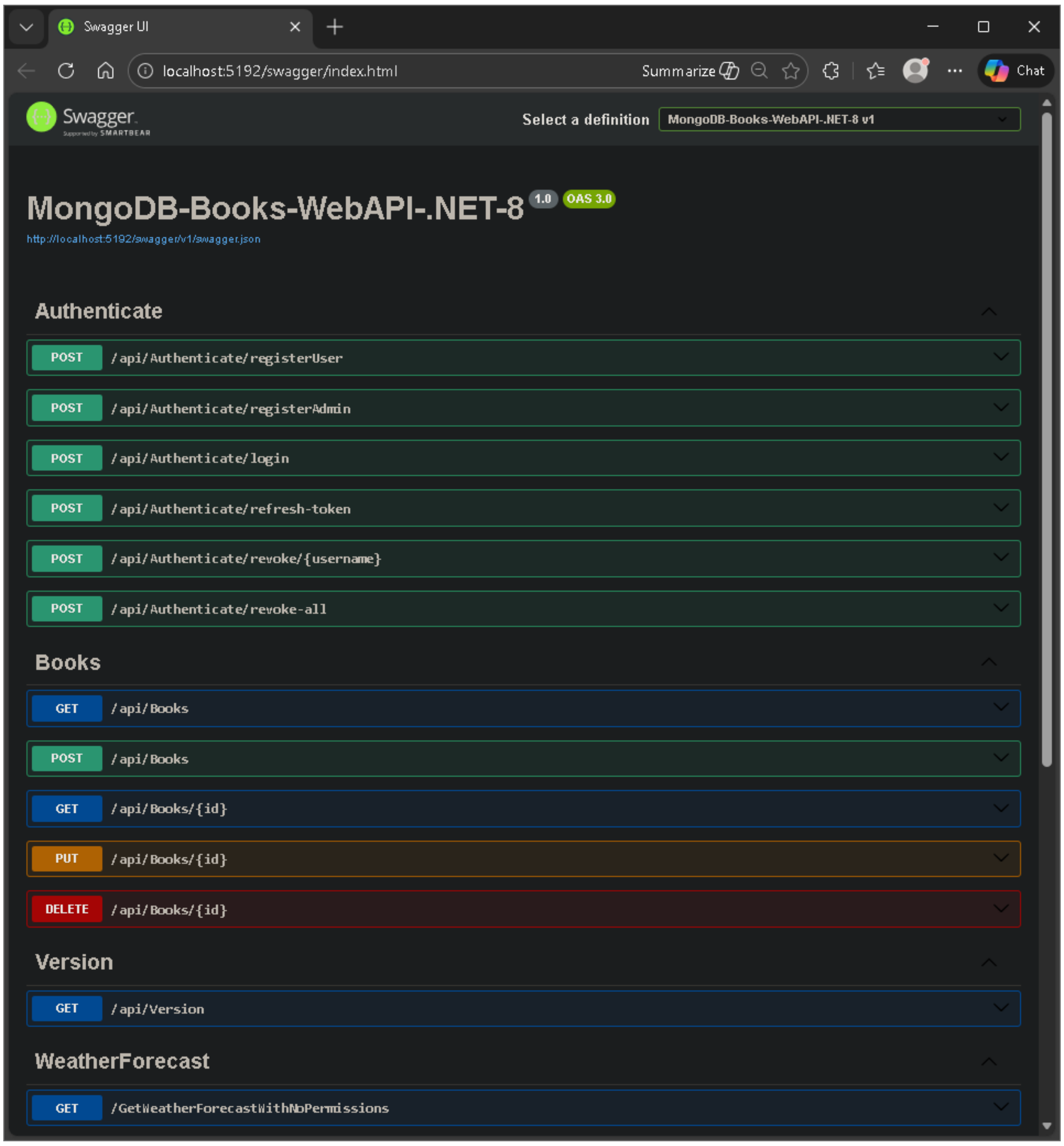
To store the information required by the JWT authorization we create the “TokenModel.cs” class. In this class we store the access token and the refresh token. The access token is valid for the time period we establish in the “appsettings.json” file. After that period, we must create a new token with the refresh token.

To manage the “http” responses we create the “Response.cs” class. In this class we define the response message and the response status code.

After creating all the authentication and authorization models classes inside the “Authorization” directory/folder, we proceed with the “appsettings. json” file modifications. We must include a new section “JWT”, inside this new section in the “appsettings. json” file we define five new variables (ValidAudience, ValidIssuer, Secret, TokenValidityInMinutes and RefreshTokenValidityInDays).

After setting the JWT configuration variables we must modify the middleware (Program.cs file) to include the authentication and authorization scenario.

Here the Swagger page with the API definition page.


Swagger UI
localhost:5192/swagger/index.html
Summarize
Chat
Swagger
Select a definition
MongoDB-Books-WebAPI-.NET-8 v1

MongoDB-Books-WebAPI-.NET-8 1.0 OAS 3.0
http://localhost:5192/swagger/v1/swagger.json

Authenticate

```
POST    /api/Authenticate/registerUser
POST    /api/Authenticate/registerAdmin
POST    /api/Authenticate/login
POST    /api/Authenticate/refresh-token
POST    /api/Authenticate/revoke/{username}
POST    /api/Authenticate/revoke-all
```

Books

```
GET     /api/Books
POST    /api/Books
GET     /api/Books/{id}
PUT     /api/Books/{id}
DELETE  /api/Books/{id}
```

Version

```
GET     /api/Version
```

WeatherForecast

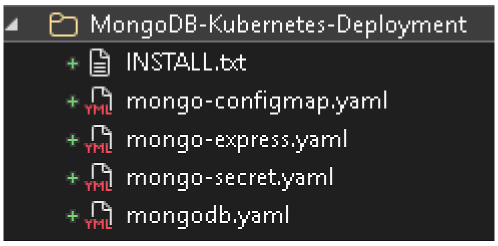
```
GET     /GetWeatherForecastWithNoPermissions
```

RQ-03 | MongoDB

MongoDB refers to a popular NoSQL database that stores data in flexible, JSON-like documents within "collections," rather than traditional tables and rows, offering dynamic schemas for evolving applications and high scalability for various data types (structured, semi-structured, unstructured). It's known for its agility, ease of use with application objects, and features like sharding and replication, making it great for modern, demanding software.

Installation

Instructions and files for deployment are in the next folder:



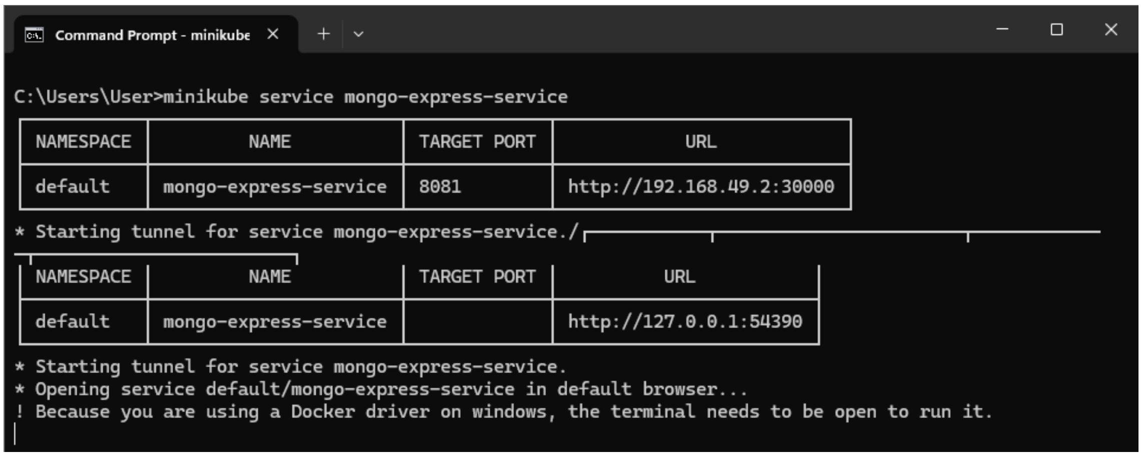
```
MongoDB-Kubernetes-Deployment
    INSTALL.txt
    mongo-configmap.yaml
    mongo-express.yaml
    mongo-secret.yaml
    mongodb.yaml
```

Mongo Express

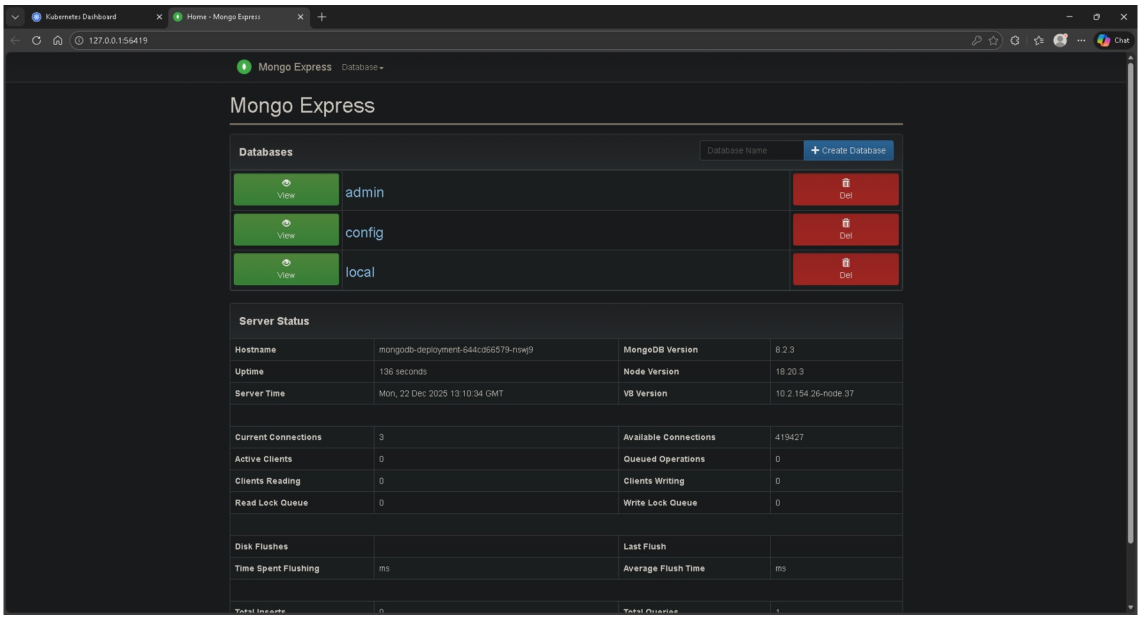
Mongo-Express is an interactive lightweight Web-Based Administrative Tool to effectively manage MongoDB Databases. Written with Node. js, Express, and Bootstrap3, Mongo Express can be used to simplify several MongoDB Admin tasks. Using Mongo Express, you can add, delete or modify databases, collections, and documents.

To launch it, execute:

```
minikube service mongo-express-service.
```



```
C:\Users\User>minikube service mongo-express-service
| NAMESPACE |         NAME          | TARGET PORT |            URL             |
| default   | mongo-express-service | 8081        | http://192.168.49.2:30000  |
* Starting tunnel for service mongo-express-service./
| NAMESPACE |         NAME          | TARGET PORT |           URL           |
| default   | mongo-express-service |             | http://127.0.0.1:54390  |
* Starting tunnel for service mongo-express-service.
* Opening service default/mongo-express-service in default browser...
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```



Mongo Express

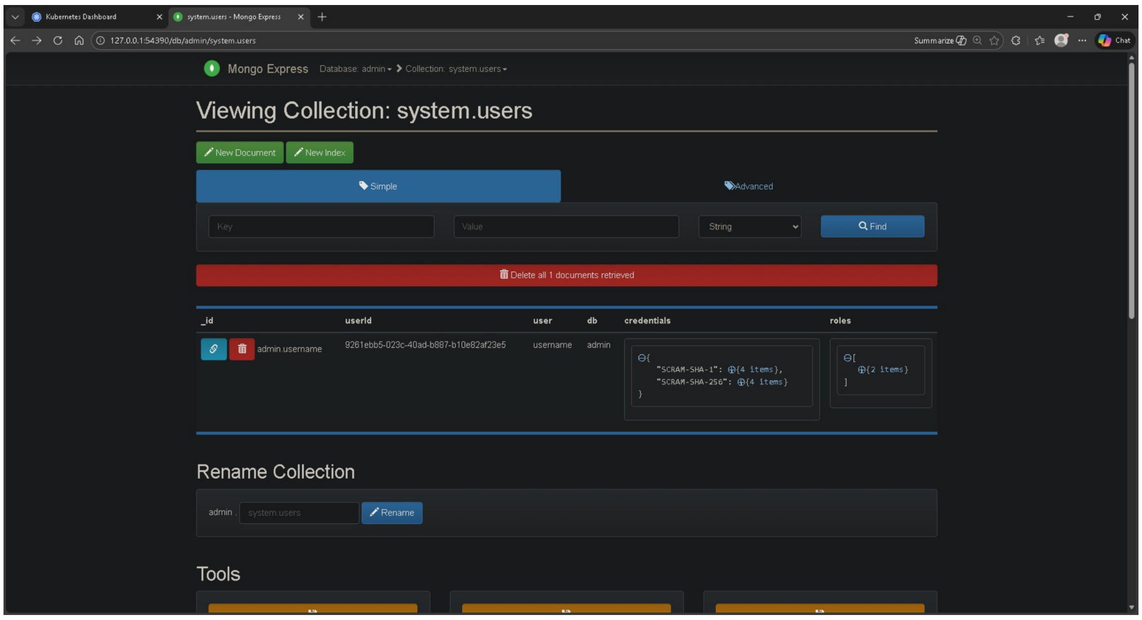
Databases

View	admin	Del
View	config	Del
View	local	Del

Server Status

Hostname	mongodb-deployment-644cd66579-nswj9	**MongoDB Version**	8.2.3
Uptime	136 seconds	**Node Version**	18.20.3
Server Time	Mon, 22 Dec 2025 13:10:34 GMT	**V8 Version**	10.2.154.26-node.37
Current Connections	3	**Available Connections**	419427
Active Clients	0	**Queued Operations**	0
Clients Reading	0	**Clients Writing**	0
Read Lock Queue	0	**Write Lock Queue**	0
Disk Flushes		**Last Flush**	
Time Spent Flushing	ms	**Average Flush Time**	ms
Total Inserts	0	**Total Queries**	1

This is the default user we have by default.



Viewing Collection: system.users

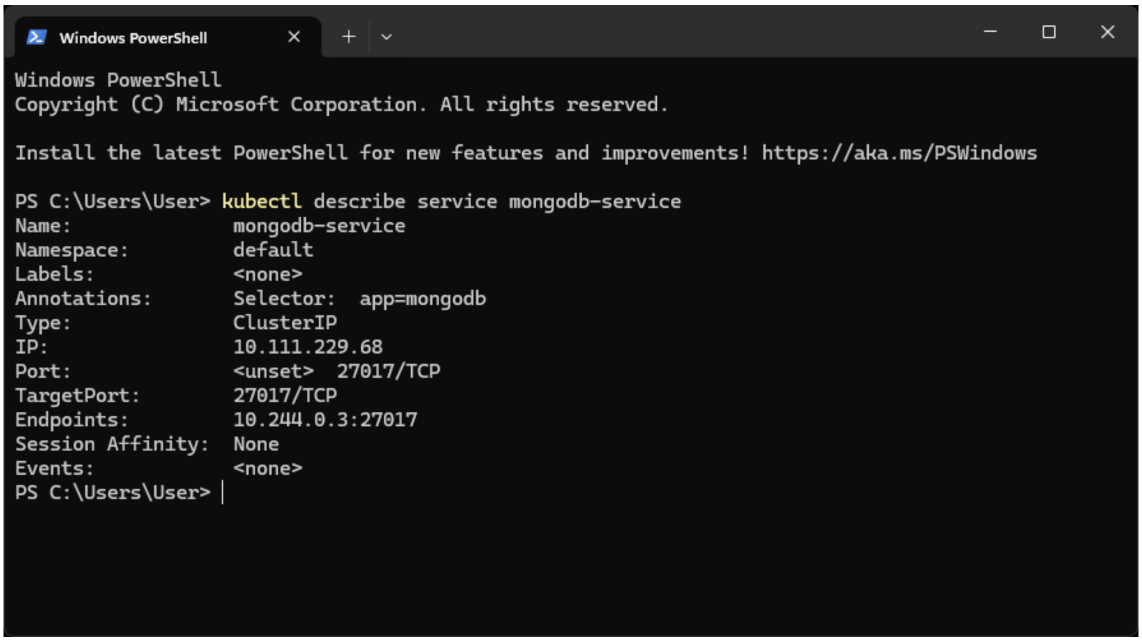
_id	userId	user	db	credentials	roles
admin.username	9261ebb5-023c-40ad-b887-b10e82af23e5	username	admin	⊖{ "SCRAM-SHA-1": ⊕{4 items}, "SCRAM-SHA-256": ⊕{4 items} }	⊖[⊕{2 items}]

Rename Collection

admin . system.users

We can have relevant information of the cluster too, by executing the next command:

```
kubectl describe service mongodb-service
```



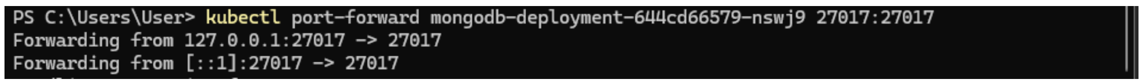
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\User> kubectl describe service mongodb-service
Name:              mongodb-service
Namespace:         default
Labels:            <none>
Annotations:       Selector:  app=mongodb
Type:              ClusterIP
IP:                10.111.229.68
Port:              <unset>  27017/TCP
TargetPort:        27017/TCP
Endpoints:         10.244.0.3:27017
Session Affinity:  None
Events:            <none>
PS C:\Users\User>
```

Access shell with admin privileges

To access the shell or a client with administrator privileges first we need to forward ports to make the service available.

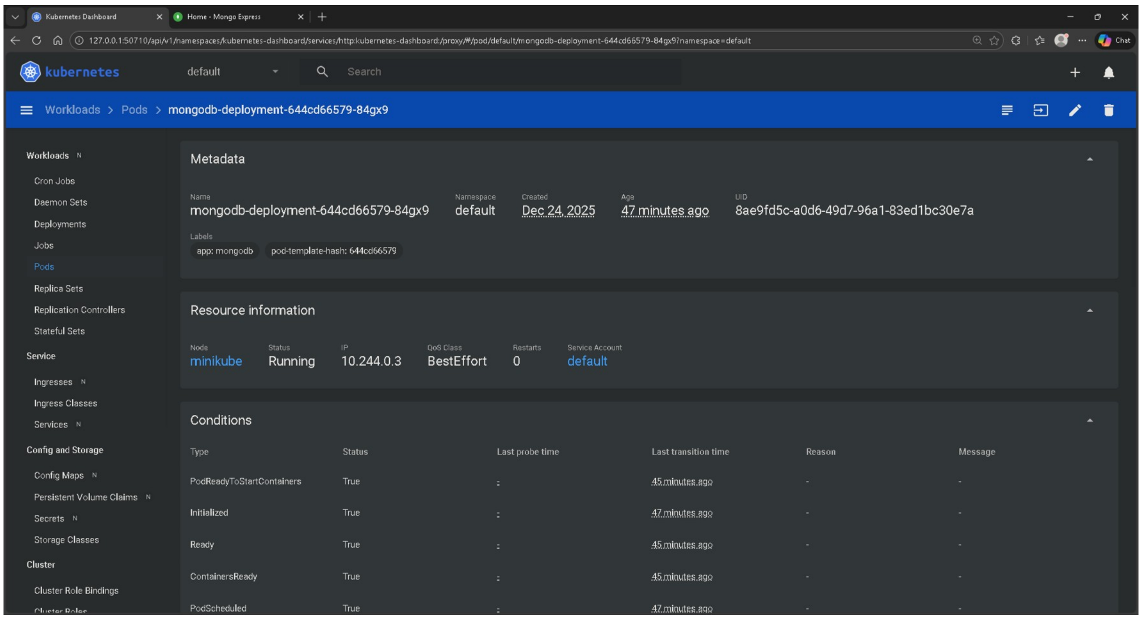
So to do that we will execute the next commnad:

```
kubectl port-forward  mongodb-deployment-644cd66579-84gx9 27017:27017
```



```
PS C:\Users\User> kubectl port-forward mongodb-deployment-644cd66579-nswj9 27017:27017
Forwarding from 127.0.0.1:27017 -> 27017
Forwarding from [::1]:27017 -> 27017
```

Where “mongodb-deployment-644cd66579-84gx9” is the pod’s name and we can get it in the description page of the minikube dashboard.



127.0.0.1:50710/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/#/pod/default/mongodb-deployment-644cd66579-84gx9?namespace=default

Workloads > Pods > mongodb-deployment-644cd66579-84gx9

Metadata

Name	Namespace	Created	Age	UID
mongodb-deployment-644cd66579-84gx9	default	Dec 24, 2025	47 minutes ago	8ae9fd5c-a0d6-49d7-96a1-83ed1bc30e7a

Labels: app: mongodb, pod-template-hash: 644cd66579

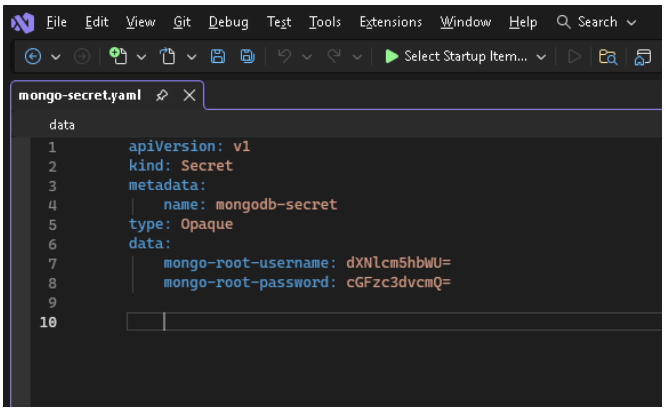
Resource information

Node	Status	IP	QoS Class	Restarts	Service Account
minikube	Running	10.244.0.3	BestEffort	0	default

Conditions

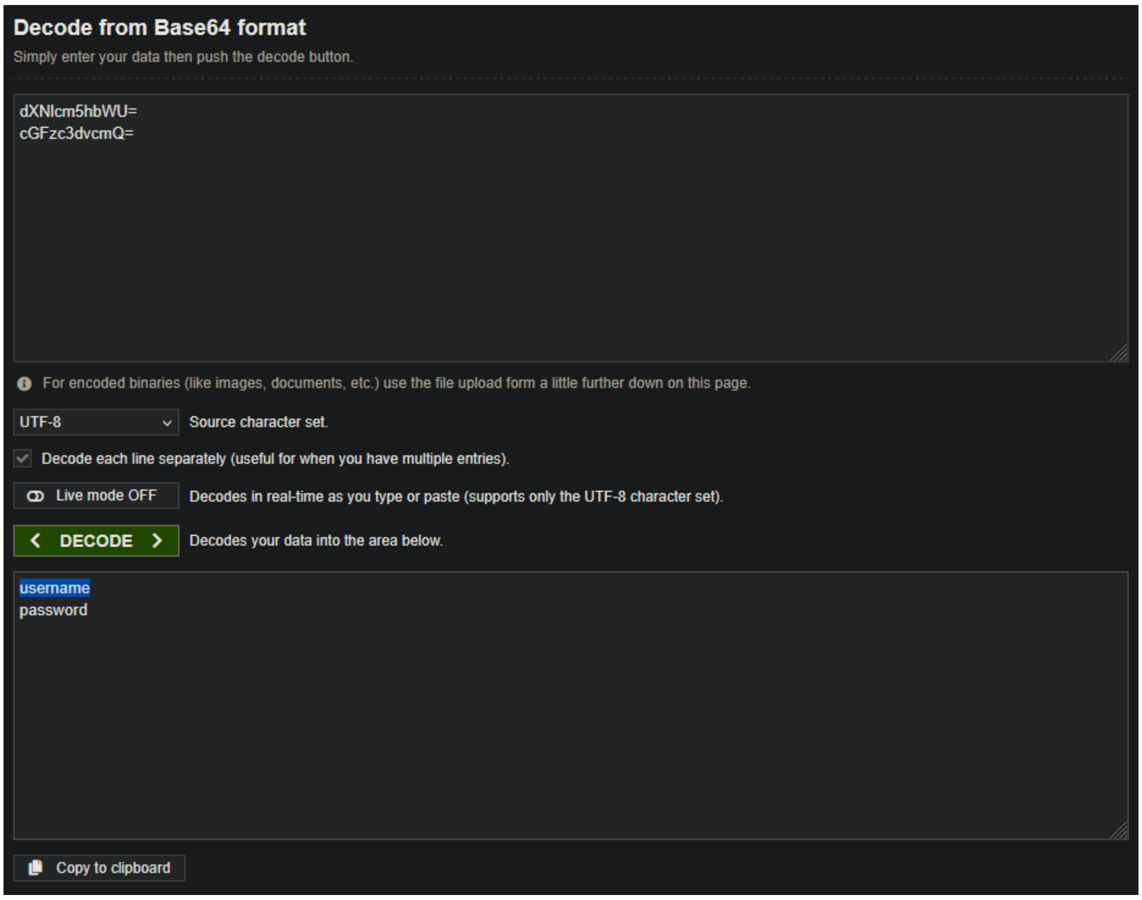
Type	Status	Last probe time	Last transition time	Reason	Message
PodReadyToStartContainers	True	-	45 minutes ago	-	-
Initialized	True	-	47 minutes ago	-	-
Ready	True	-	45 minutes ago	-	-
ContainersReady	True	-	45 minutes ago	-	-
PodScheduled	True	-	47 minutes ago	-	-

Ow we need the user and password that were used during installation.


File Edit View Git Debug Test Tools Extensions Window Help Search
Select Startup Item...
mongo-secret.yaml
data

```yaml
apiVersion: v1
kind: Secret
metadata:
    name: mongodb-secret
type: Opaque
data:
    mongo-root-username: dXNlcm5hbWU=
    mongo-root-password: cGFzc3dvcmQ=

```

We decode them.



Decode from Base64 format

Simply enter your data then push the decode button.

```
dXNlcm5hbWU=
cGFzc3dvcmQ=
```

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Source character set.

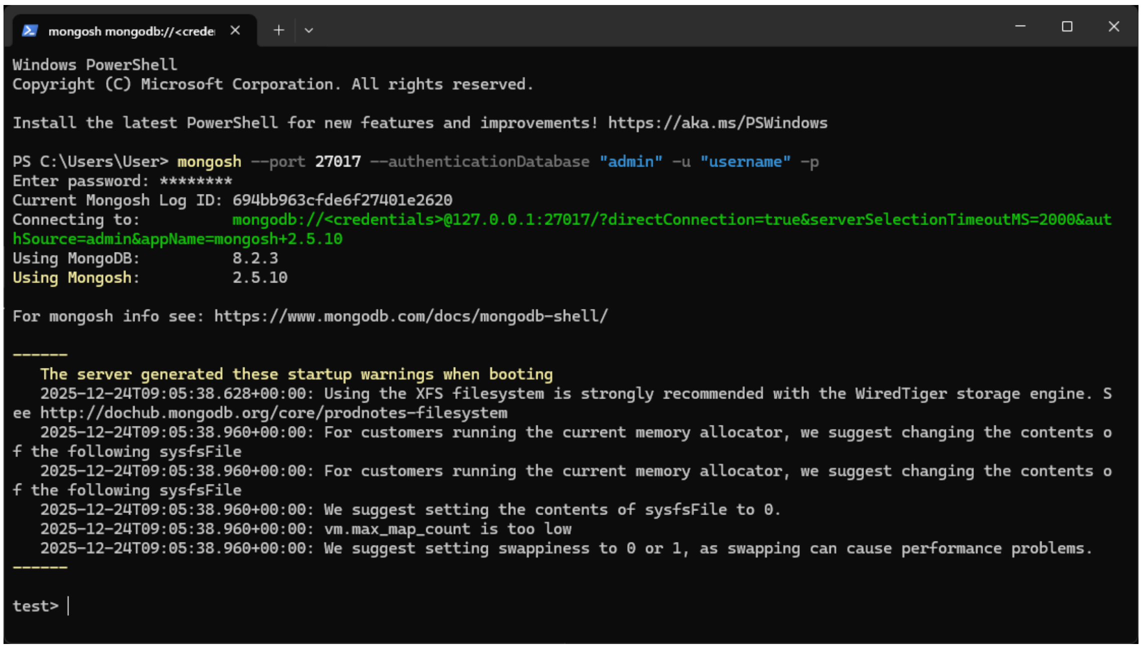
[✓] Decode each line separately (useful for when you have multiple entries).

Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

DECODE Decodes your data into the area below.

```
username
password
```

Now we can login with the shell i.e.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\User> mongosh --port 27017 --authenticationDatabase "admin" -u "username" -p
Enter password: ********
Current Mongosh Log ID: 694bb963cfde6f27401e2620
Connecting to:          mongodb://<credentials>@127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&aut
hSource=admin&appName=mongosh+2.5.10
Using MongoDB:          8.2.3
Using Mongosh:          2.5.10

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

------
   The server generated these startup warnings when booting
   2025-12-24T09:05:38.628+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. S
ee http://dochub.mongodb.org/core/prodnotes-filesystem
   2025-12-24T09:05:38.960+00:00: For customers running the current memory allocator, we suggest changing the contents o
f the following sysfsFile
   2025-12-24T09:05:38.960+00:00: For customers running the current memory allocator, we suggest changing the contents o
f the following sysfsFile
   2025-12-24T09:05:38.960+00:00: We suggest setting the contents of sysfsFile to 0.
   2025-12-24T09:05:38.960+00:00: vm.max_map_count is too low
   2025-12-24T09:05:38.960+00:00: We suggest setting swappiness to 0 or 1, as swapping can cause performance problems.
------

test> 
```

RQ-04 | Docker

Docker is a software platform that allows you to build, test, and deploy applications quickly. Docker packages software into standardized units called containers that have everything the software needs to run including libraries, system tools, code, and runtime.

Minikube uses Docker to run Kubernetes node but you can use any VM engine:

```
minikube start --driver=docker
```

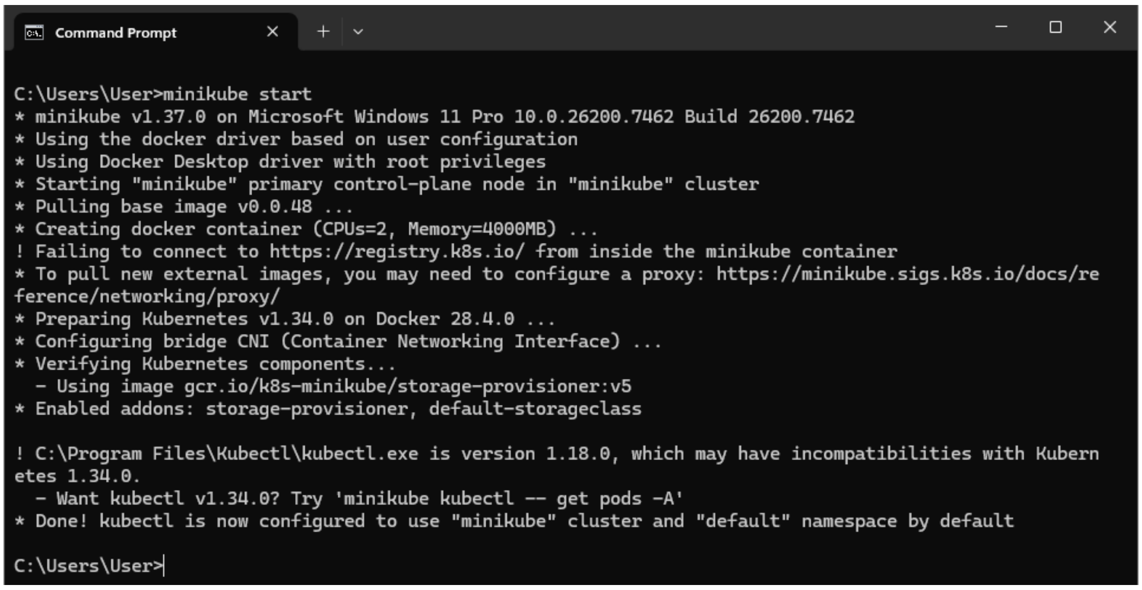
```
minikube start --driver=VMware (for VMWare virtualization)
```

RQ-05 | Kubernetes Single Node (Minikube)

Kubernetes is an open source system to deploy, scale, and manage containerized applications anywhere. This project shows you how to run a sample app on Kubernetes using Minikube.

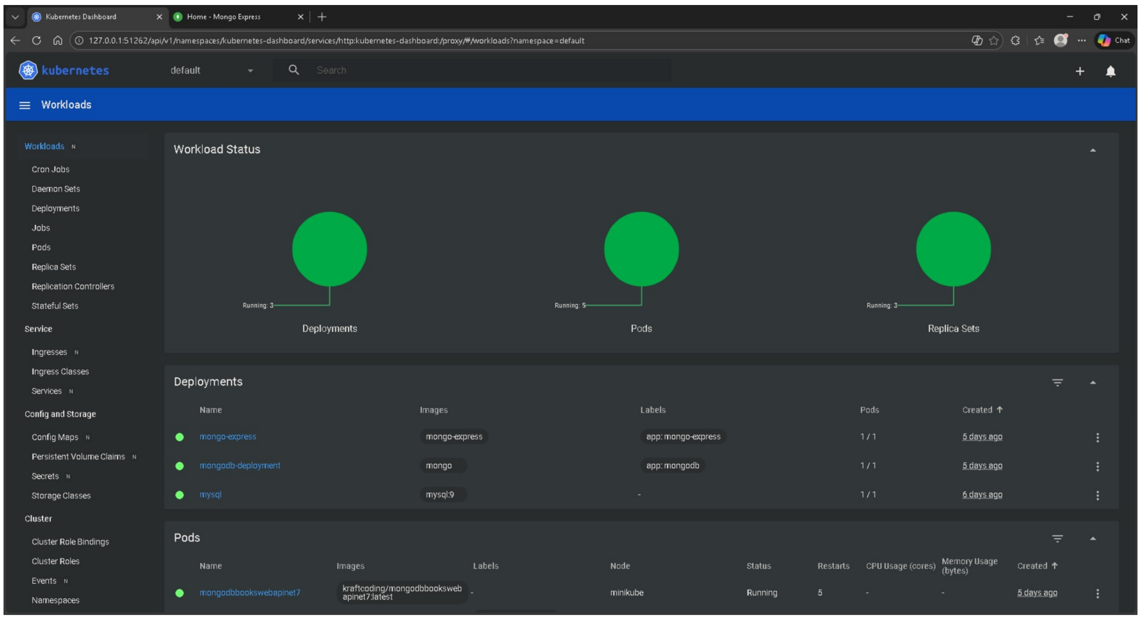
Minikube is an open-source tool that lets you run a single-node Kubernetes cluster on your local machine (laptop/PC) for learning, development, and testing, providing a lightweight, portable environment that mimics a real cluster without needing complex cloud infrastructure. It runs inside a virtual machine or container on your OS (Linux, macOS, Windows) and includes components like the control plane and worker node, allowing you to practice deploying applications using tools like kubectl.

To start a cluster, run: "minikube start":



```
C:\Users\User>minikube start
* minikube v1.37.0 on Microsoft Windows 11 Pro 10.0.26200.7462 Build 26200.7462
* Using the docker driver based on user configuration
* Using Docker Desktop driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.48 ...
* Creating docker container (CPUs=2, Memory=4000MB) ...
! Failing to connect to https://registry.k8s.io/ from inside the minikube container
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/re
ference/networking/proxy/
* Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass

! C:\Program Files\Kubectl\kubectl.exe is version 1.18.0, which may have incompatibilities with Kubern
etes 1.34.0.
  - Want kubectl v1.34.0? Try 'minikube kubectl -- get pods -A'
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

C:\Users\User>
```

To start the Dashboard, run: "minikube dashboard":



Deployments

	Name	Images	Labels	Pods	Created ↑
●	mongo-express	mongo-express	app: mongo-express	1 / 1	5 days ago
●	mongodb-deployment	mongo	app: mongodb	1 / 1	5 days ago
●	mysql	mysql:9	-	1 / 1	6 days ago

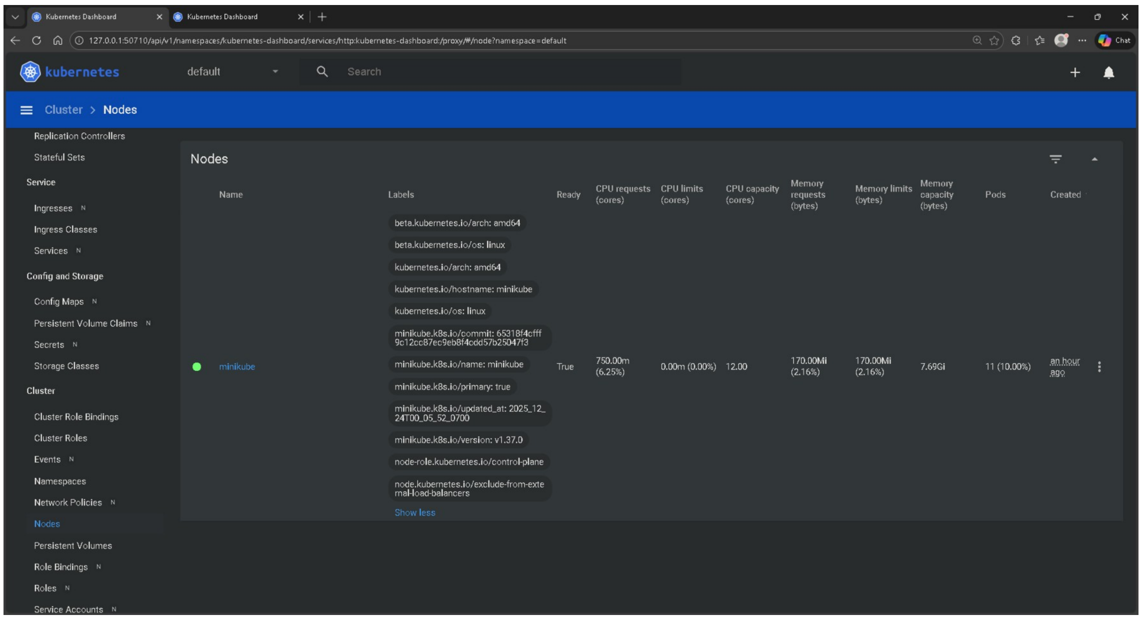
Pods

	Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created ↑
●	mongodbbookswebapinet7	kraftcoding/mongodbbookswebapinet7:latest	-	minikube	Running	5	-	-	5 days ago

This will create too an image and a container for the Kubernetes single node in Docker.

Node Management

By using the minikube dashboard we can access to the node’s information to monitor the components that comprises together to the load and state of the processes, just like a small dedicated linux based OS to manage dockerized applications.

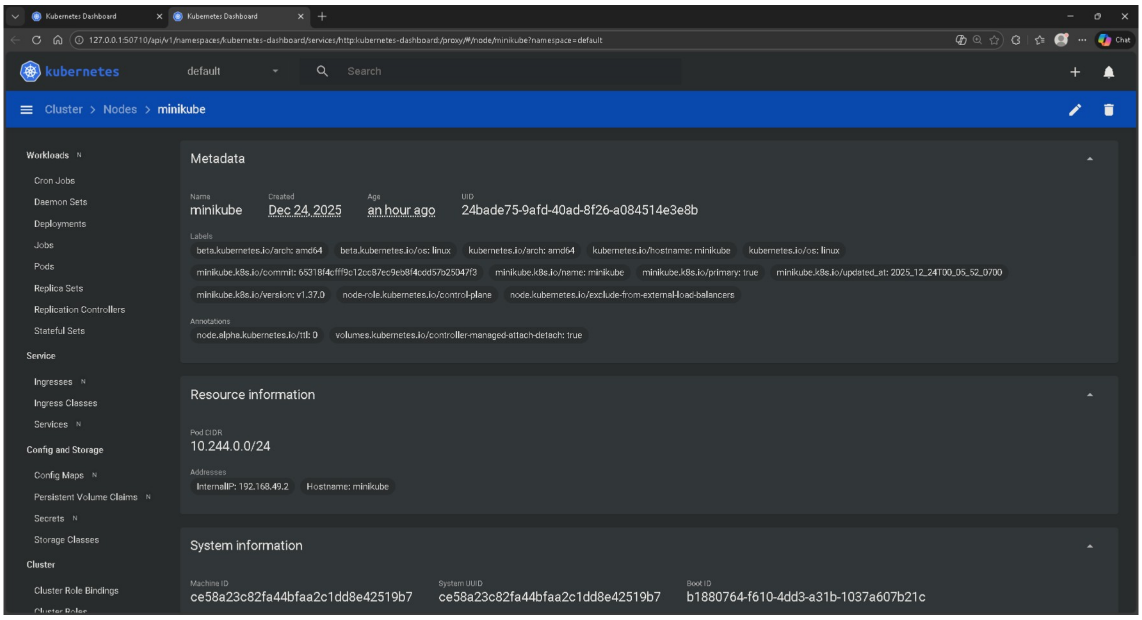


Nodes

	Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	CPU capacity (cores)	Memory requests (bytes)	Memory limits (bytes)	Memory capacity (bytes)	Pods	Created
●	minikube	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux kubernetes.io/arch: amd64 kubernetes.io/hostname: minikube kubernetes.io/os: linux minikube.k8s.io/commit: 65318f4cfff9c12cc87ec9eb8f4cdd57b25047f3 minikube.k8s.io/name: minikube minikube.k8s.io/primary: true minikube.k8s.io/updated_at: 2025_12_24T00_05_52_0700 minikube.k8s.io/version: v1.37.0 node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-from-external-load-balancers Show less	True	750.00m (6.25%)	0.00m (0.00%)	12.00	170.00Mi (2.16%)	170.00Mi (2.16%)	7.69Gi	11 (10.00%)	an hour ago

Here the node monitorig page with interesting information about the node that we might need to have in account.

- Internal IP: 192.168.49.2
- Hostname: minikube



Metadata

Name: minikube
Created: Dec 24, 2025
Age: an hour ago
UID: 24bade75-9afd-40ad-8f26-a084514e3e8b

Labels
beta.kubernetes.io/arch: amd64
beta.kubernetes.io/os: linux
kubernetes.io/arch: amd64
kubernetes.io/hostname: minikube
kubernetes.io/os: linux
minikube.k8s.io/commit: 65318f4cfff9c12cc87ec9eb8f4cdd57b25047f3
minikube.k8s.io/name: minikube
minikube.k8s.io/primary: true
minikube.k8s.io/updated_at: 2025_12_24T00_05_52_0700
minikube.k8s.io/version: v1.37.0
node-role.kubernetes.io/control-plane
node.kubernetes.io/exclude-from-external-load-balancers

Annotations
node.alpha.kubernetes.io/ttl: 0
volumes.kubernetes.io/controller-managed-attach-detach: true

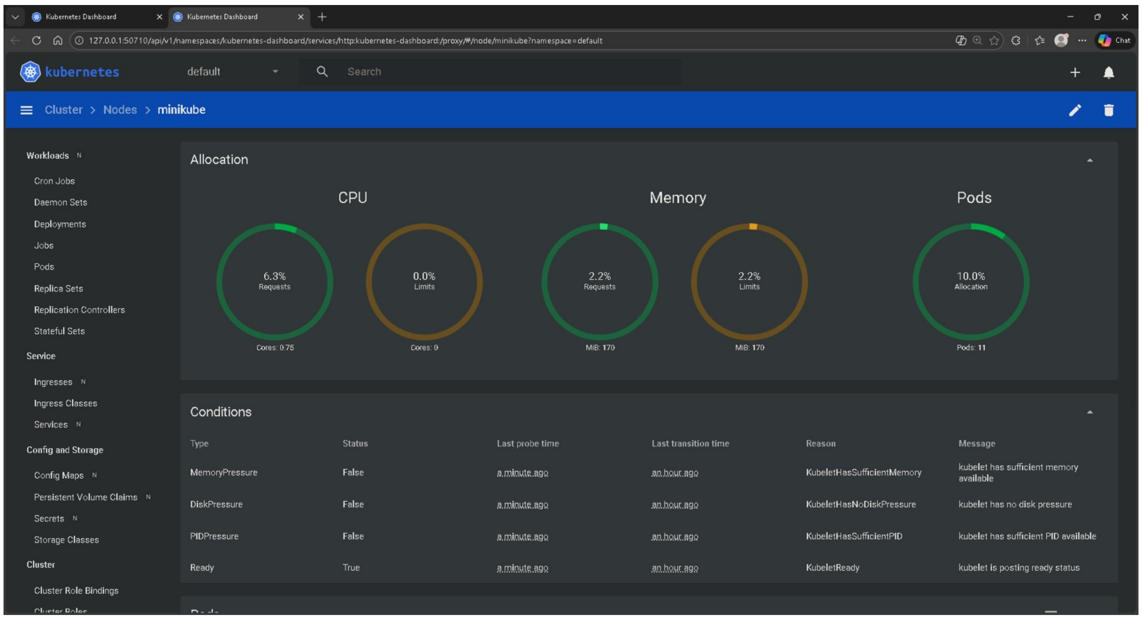
Resource information

Pod CIDR: 10.244.0.0/24

Addresses
InternalIP: 192.168.49.2
Hostname: minikube

System information

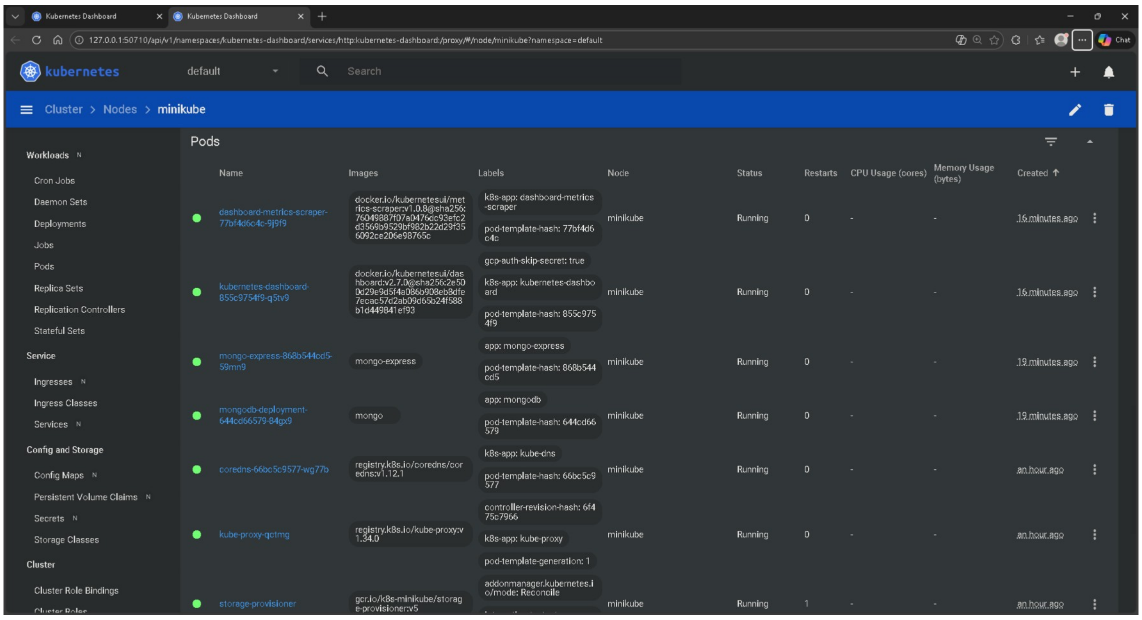
Machine ID: ce58a23c82fa44bfaa2c1dd8e42519b7
System UUID: ce58a23c82fa44bfaa2c1dd8e42519b7
Boot ID: b1880764-f610-4dd3-a31b-1037a607b21c


Allocation

	CPU		Memory		Pods
	6.3% Requests	0.0% Limits	2.2% Requests	2.2% Limits	10.0% Allocation
	Cores: 0.75	Cores: 0	MiB: 170	MiB: 170	Pods: 11

Conditions

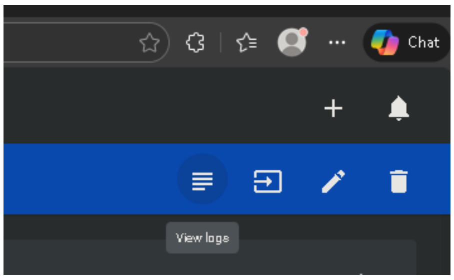
Type	Status	Last probe time	Last transition time	Reason	Message
MemoryPressure	False	a minute ago	an hour ago	KubeletHasSufficientMemory	kubelet has sufficient memory available
DiskPressure	False	a minute ago	an hour ago	KubeletHasNoDiskPressure	kubelet has no disk pressure
PIDPressure	False	a minute ago	an hour ago	KubeletHasSufficientPID	kubelet has sufficient PID available
Ready	True	a minute ago	an hour ago	KubeletReady	kubelet is posting ready status

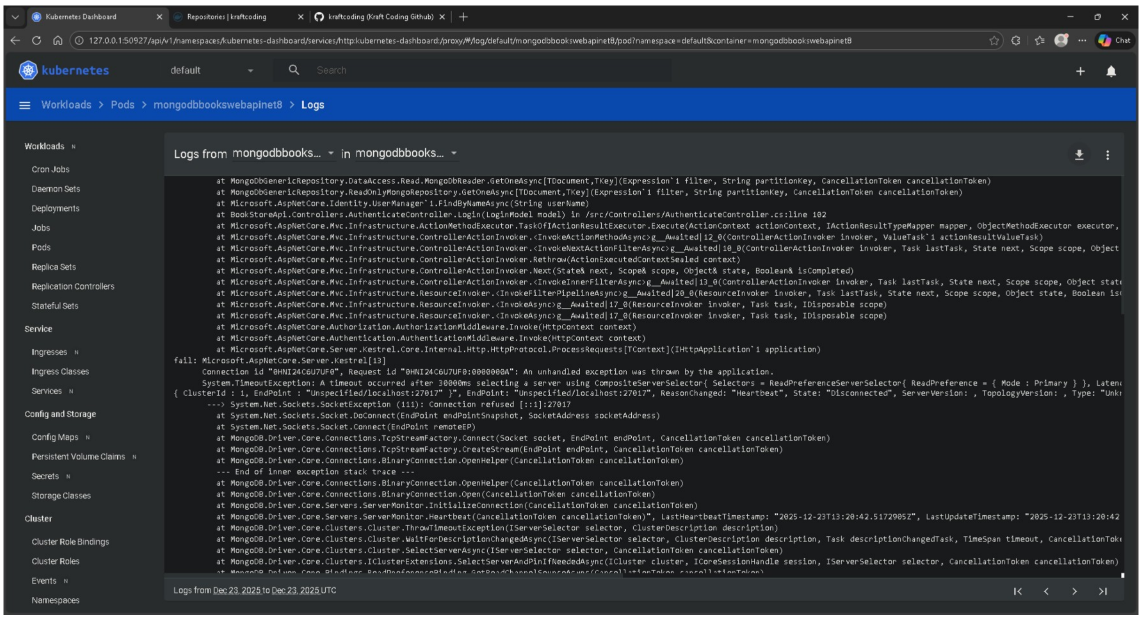

Pods

Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created
dashboard-metrics-scraper-77bf4d6c4c-9j9f9	docker.io/kubernetesui/metrics-scraper:v1.0.8@sha256:76049887f07a0476dc93efc2d3569b9529bf982b22d29f356092ce206e98765c	k8s-app: dashboard-metrics-scraper; pod-template-hash: 77bf4d6c4c	minikube	Running	0	-	-	16 minutes ago
kubernetes-dashboard-855c9754f9-q5tv9	docker.io/kubernetesui/dashboard:v2.7.0@sha256:2e500d29e9d5f4a086b908eb8dfe7ecac57d2ab09d65b24f588b1d449841ef93	gcp-auth-skip-secret: true; k8s-app: kubernetes-dashboard; pod-template-hash: 855c9754f9	minikube	Running	0	-	-	16 minutes ago
mongo-express-868b544cd5-59mn9	mongo-express	app: mongo-express; pod-template-hash: 868b544cd5	minikube	Running	0	-	-	19 minutes ago
mongodb-deployment-644cd66579-84gx9	mongo	app: mongodb; pod-template-hash: 644cd66579	minikube	Running	0	-	-	19 minutes ago
coredns-66bc5c9577-wg77b	registry.k8s.io/coredns/coredns:v1.12.1	k8s-app: kube-dns; pod-template-hash: 66bc5c9577	minikube	Running	0	-	-	an hour ago
kube-proxy-qctmg	registry.k8s.io/kube-proxy:v1.34.0	controller-revision-hash: 6f475c7966; k8s-app: kube-proxy; pod-template-generation: 1	minikube	Running	0	-	-	an hour ago
storage-provisioner	gcr.io/k8s-minikube/storage-provisioner:v5	addonmanager.kubernetes.io/mode: Reconcile	minikube	Running	1	-	-	an hour ago

Logs

For example if we would need to debug a pod, he can access to it's logs by click in "view logs button".





```
         at MongoDbGenericRepository.DataAccess.Read.MongoDbReader.GetOneAsync[TDocument,TKey](Expression`1 filter, String partitionKey, CancellationToken cancellationToken)
         at MongoDbGenericRepository.ReadOnlyMongoRepository.GetOneAsync[TDocument,TKey](Expression`1 filter, String partitionKey, CancellationToken cancellationToken)
         at Microsoft.AspNetCore.Identity.UserManager`1.FindByNameAsync(String userName)
         at BookStoreApi.Controllers.AuthenticateController.Login(LoginModel model) in /src/Controllers/AuthenticateController.cs:line 102
         at Microsoft.AspNetCore.Mvc.Infrastructure.ActionMethodExecutor.TaskOfIActionResultExecutor.Execute(ActionContext actionContext, IActionResultTypeMapper mapper, ObjectMethodExecutor executor,
         at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.<InvokeActionMethodAsync>g__Awaited|12_0(ControllerActionInvoker invoker, ValueTask`1 actionResultValueTask)
         at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.<InvokeNextActionFilterAsync>g__Awaited|10_0(ControllerActionInvoker invoker, Task lastTask, State next, Scope scope, Object
         at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.Rethrow(ActionExecutedContextSealed context)
         at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.Next(State& next, Scope& scope, Object& state, Boolean& isCompleted)
         at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.<InvokeInnerFilterAsync>g__Awaited|13_0(ControllerActionInvoker invoker, Task lastTask, State next, Scope scope, Object state
         at Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.<InvokeFilterPipelineAsync>g__Awaited|20_0(ResourceInvoker invoker, Task lastTask, State next, Scope scope, Object state, Boolean is
         at Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.<InvokeAsync>g__Awaited|17_0(ResourceInvoker invoker, Task task, IDisposable scope)
         at Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.<InvokeAsync>g__Awaited|17_0(ResourceInvoker invoker, Task task, IDisposable scope)
         at Microsoft.AspNetCore.Authorization.AuthorizationMiddleware.Invoke(HttpContext context)
         at Microsoft.AspNetCore.Authentication.AuthenticationMiddleware.Invoke(HttpContext context)
         at Microsoft.AspNetCore.Server.Kestrel.Core.Internal.Http.HttpProtocol.ProcessRequests[TContext](IHttpApplication`1 application)
fail: Microsoft.AspNetCore.Server.Kestrel[13]
      Connection id "0HNI24C6U7UF0", Request id "0HNI24C6U7UF0:0000000A": An unhandled exception was thrown by the application.
      System.TimeoutException: A timeout occurred after 30000ms selecting a server using CompositeServerSelector{ Selectors = ReadPreferenceServerSelector{ ReadPreference = { Mode : Primary } }, Latenc
{ ClusterId : 1, EndPoint : "Unspecified/localhost:27017" }", EndPoint: "Unspecified/localhost:27017", ReasonChanged: "Heartbeat", State: "Disconnected", ServerVersion: , TopologyVersion: , Type: "Unk
       ---> System.Net.Sockets.SocketException (111): Connection refused [::1]:27017
         at System.Net.Sockets.Socket.DoConnect(EndPoint endPointSnapshot, SocketAddress socketAddress)
         at System.Net.Sockets.Socket.Connect(EndPoint remoteEP)
         at MongoDB.Driver.Core.Connections.TcpStreamFactory.Connect(Socket socket, EndPoint endPoint, CancellationToken cancellationToken)
         at MongoDB.Driver.Core.Connections.TcpStreamFactory.CreateStream(EndPoint endPoint, CancellationToken cancellationToken)
         at MongoDB.Driver.Core.Connections.BinaryConnection.OpenHelper(CancellationToken cancellationToken)
         --- End of inner exception stack trace ---
         at MongoDB.Driver.Core.Connections.BinaryConnection.OpenHelper(CancellationToken cancellationToken)
         at MongoDB.Driver.Core.Connections.BinaryConnection.Open(CancellationToken cancellationToken)
         at MongoDB.Driver.Core.Servers.ServerMonitor.InitializeConnection(CancellationToken cancellationToken)
         at MongoDB.Driver.Core.Servers.ServerMonitor.Heartbeat(CancellationToken cancellationToken)", LastHeartbeatTimestamp: "2025-12-23T13:20:42.5172905Z", LastUpdateTimestamp: "2025-12-23T13:20:42
         at MongoDB.Driver.Core.Clusters.Cluster.ThrowTimeoutException(IServerSelector selector, ClusterDescription description)
         at MongoDB.Driver.Core.Clusters.Cluster.WaitForDescriptionChangedAsync(IServerSelector selector, ClusterDescription description, Task descriptionChangedTask, TimeSpan timeout, CancellationToke
         at MongoDB.Driver.Core.Clusters.Cluster.SelectServerAsync(IServerSelector selector, CancellationToken cancellationToken)
         at MongoDB.Driver.Core.Clusters.IClusterExtensions.SelectServerAndPinIfNeededAsync(ICluster cluster, ICoreSessionHandle session, IServerSelector selector, CancellationToken cancellationToken)
         at MongoDB.Driver.Core.Bindings.ReadPreferenceBinding.GetReadChannelSourceAsync(CancellationToken cancellationToken)
```

Logs from Dec 23, 2025 to Dec 23, 2025 UTC

Deployment

When building modern, stateless applications, containerizing application’s components is the first step in deploying and scaling on distributed platforms. By using Docker in development, you can containerize your application by:

- Extracting necessary configuration information from your code.
- Offloading your application’s state.
- Packaging your application for repeated use.

To run your services on a distributed platform like Kubernetes, you will need to translate your service definitions to Kubernetes objects to specify how your container images should run. This will allow you to scale your application with resiliency.

For a development environment we will need:

- Docker
- Minikube
- Kubectl

The deployment procedure is the next one:

1. First of all we need Docker running to the launch Minikube
2. Once both are running we can deploy MongoDB components
3. Then we install / launch the Web API

Annexes

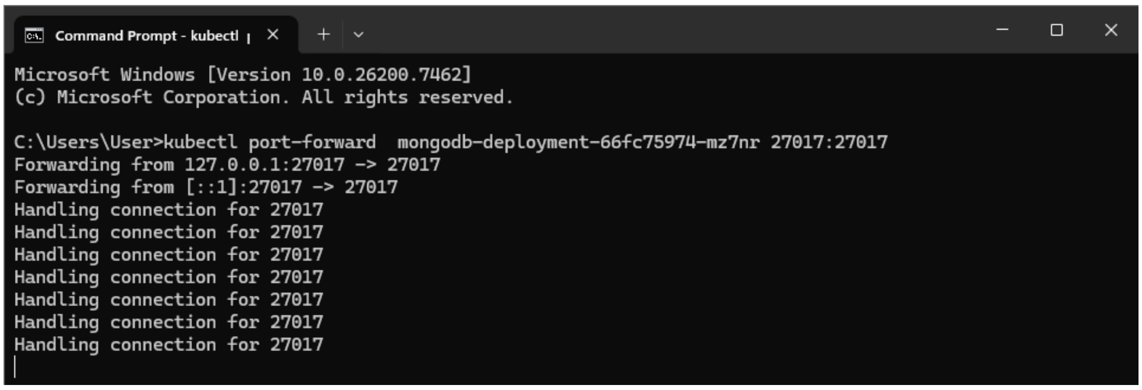
Docker Compose

Docker-compose is the right tool for you to make containers communication to each other. Docker is a software platform that allows you to build, test, and deploy applications quickly. Docker packages software into standardized units called containers that have everything the software needs to run including libraries, system tools, code, and runtime.

Forward Service ports

Due to the MongoDB instance is running in Kubernetes it is needed to forward ports to debug with visual studio.

- kubectl port-forward mongodb-deployment-66fc75974-mz7nr 27017:27017



```
Microsoft Windows [Version 10.0.26200.7462]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User>kubectl port-forward  mongodb-deployment-66fc75974-mz7nr 27017:27017
Forwarding from 127.0.0.1:27017 -> 27017
Forwarding from [::1]:27017 -> 27017
Handling connection for 27017
Handling connection for 27017
Handling connection for 27017
Handling connection for 27017
Handling connection for 27017
Handling connection for 27017
Handling connection for 27017
```