



SCD-Assignment-03

Prepared by: Abad Naseer

Submitted to: Laiba Imran

Submission Date

12 December 2024

Comprehensive Report on Deployment and Management of a Multi-Episode Containerized Application

Assignment Details:

This report documents the deployment and management process for a multi-episode containerized application. The application consists of five services, each with a frontend and backend, developed as separate modules. The deployment was carried out in both Docker and Kubernetes environments as per the assignment's requirements.

Part 1: Individual Deployment (80 Points)

Task Overview

1. Deploying Individual Modules Locally

- Each service's frontend and backend modules were containerized and run locally using Docker.
- Modules were made accessible and responsive to HTTP requests.

Dockerfile Details

- Each module has its own `Dockerfile` for frontend and backend.

Episode-01:

- **Backend Dockerfile:** Located at `C:\Users\DELL\Downloads\i201815_A3\episode-01\backend\Dockerfile`.

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell - backend + - - - ^ x
PS C:\Users\DELL\Downloads\i201815_A3\episode-01\backend> docker build -t episode-01-backend .
>>
[+] Building 123.5s (12/12) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 517B
=> [internal] load metadata for docker.io/library/node:18
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/6] FROM docker.io/library/node:18@sha256:b57ae84fe7880a23b389f8260d726b784010ed470c2ee26d4e2cddb955d25b12
=> => resolve docker.io/library/node:18@sha256:b57ae84fe7880a23b389f8260d726b784010ed470c2ee26d4e2cddb955d25b12
=> => sha256:b57ae84fe7880a23b389f8260d726b784010ed470c2ee26d4e2cddb955d25b12 6.41kB / 6.41kB
=> => sha256:9371b74049c4bb6422e1f25c93f564ce9524a647fc951a35f0945efe3e96d4d 2.49kB / 2.49kB
=> => sha256:b2da2cb649e9f22d68ecc32c89d6833e6dc1705f0162c3f8a64df5772a478884 6.39kB / 6.39kB
=> => sha256:551df7f94f9c131f2fec0e8063142411365f0a1c88b935b9fac22be91af227e0 64.39MB / 64.39MB
=> => sha256:5bd71677db44bb63b94de61b6f1f95d5540b4ba2d6a8a6bc4d19f422b25e0c2b 23.87MB / 23.87MB
=> => sha256:fdff894e782a221820acf469d425b802be26aedb5e5d26ea80a650ff6a974d488 48.50MB / 48.50MB
=> => sha256:ce82e98d553dd62ca6a12bebf8e83992ae9f9ae2748275e74b66a68cc094f868b 211.31MB / 211.31MB
=> => sha256:6399a464889d3eae2913051cb98c35d0b6bfa20ec77d6b3a04617d4a298a2a56 3.32kB / 3.32kB
=> => sha256:a3c94c84d15dfc1c2c202acca56d7327f541d62c10f9bc1dfb013a618aebd5f1 45.70MB / 45.70MB
=> => sha256:2cd8c50fd8ca9ed98f596afc5d92d00b4492b7b069d2d339a6ed8682fc568961 1.25MB / 1.25MB
=> => sha256:247468edfd9afcf43bf96caab52a1d979edd5eb13afcaf570c1513f4a35fa43f 446B / 446B
=> => extracting sha256:fdff894e782a221820acf469d425b802be26aedb5e5d26ea80a650ff6a974d488
=> => extracting sha256:5bd71677db44bb63b94de61b6f1f95d5540b4ba2d6a8a6bc4d19f422b25e0c2b
=> => extracting sha256:551df7f94f9c131f2fec0e8063142411365f0a1c88b935b9fac22be91af227e0
=> => extracting sha256:ce82e98d553dd62ca6a12bebf8e83992ae9f9ae2748275e74b66a68cc094f868b
=> => extracting sha256:6399a464889d3eae2913051cb98c35d0b6bfa20ec77d6b3a04617d4a298a2a56
=> => extracting sha256:a3c94c84d15dfc1c2c202acca56d7327f541d62c10f9bc1dfb013a618aebd5f1
=> => extracting sha256:2cd8c50fd8ca9ed98f596afc5d92d00b4492b7b069d2d339a6ed8682fc568961
=> => extracting sha256:247468edfd9afcf43bf96caab52a1d979edd5eb13afcaf570c1513f4a35fa43f
=> [internal] load build context
=> => transferring context: 1.44kB

```

Now below is the command to run docker container for the backend image:

```

View a summary of image vulnerabilities and recommendations: docker run -d -p 8080:8080 --name episode-01-backend episode-01-backend
>> C:\Users\DELL\Downloads\i201815_A3\episode-01\backend>
fe0fcee67376ad0647f40fc4e14c12b8627d8f338a1b4af6117baaefd87e1f93
PS C:\Users\DELL\Downloads\i201815_A3\episode-01\backend> docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        NAMES
fe0fcee67376   episode-01-backend                 "docker-entrypoint.s..." 24 seconds ago Up 8 seconds  episode-01-backend
7027825f2fd    gcr.io/k8s-minikube/kicbase:v0.0.45 "/usr/local/bin/entr..." 7 days ago     Exited (255) 54 minutes ago 127.0.0.1:32772->22/tcp, 12
7.0.0.1:32771->2376/tcp, 127.0.0.1:32770->5000/tcp, 127.0.0.1:32769->8443/tcp, 127.0.0.1:32768->32443/tcp minikube
PS C:\Users\DELL\Downloads\i201815_A3\episode-01\backend>

```

- o **Frontend Dockerfile:** Located at C:\Users\DELL\Downloads\i201815_A3\episode-01\fronted\Dockerfile.

```

PS C:\Users\DELL\Downloads\i201815_A3\episode-01\fronted> docker build -t episode-01-frontend .
>>
[+] Building 98.8s (13/15)
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 762B
=> [internal] load metadata for docker.io/library/nginx:alpine
=> [internal] load metadata for docker.io/library/node:18
=> [auth] library/nginx:pull token for registry-1.docker.io
[+] Building 98.8s (13/15)
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 762B
=> [internal] load metadata for docker.io/library/nginx:alpine
=> [internal] load metadata for docker.io/library/node:18
=> [auth] library/nginx:pull token for registry-1.docker.io
[+] Building 98.9s (13/15)
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 762B
=> [internal] load metadata for docker.io/library/nginx:alpine
=> [internal] load metadata for docker.io/library/node:18
=> [auth] library/nginx:pull token for registry-1.docker.io
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 762B
=> [internal] load metadata for docker.io/library/nginx:alpine
=> [internal] load metadata for docker.io/library/node:18
=> [auth] library/nginx:pull token for registry-1.docker.io
=> [internal] load metadata for docker.io/library/nginx:alpine
=> [internal] load metadata for docker.io/library/node:18
=> [auth] library/nginx:pull token for registry-1.docker.io

```

Here is the screenshot of the frontend service running in a container:

```

PS C:\Users\DELL\Downloads\i201815_A3\episode-01\fronted> docker run -d -p 8001:8001 --name episode-01-frontend episode-01-frontend
bf2e9b75e22646eacd87861846abfe8d7c375efa59856086b688c6e1dcc3e1c0
PS C:\Users\DELL\Downloads\i201815_A3\episode-01\fronted> docker ps -a

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
bf2e9b75e226	episode-01-frontend	"/docker-entrypoint..."	9 seconds ago	Up 8 seconds	80/tcp, 0.0.0.0:8001->8001/tcp
fe0fcee67376	episode-01-backend	"docker-entrypoint.s..."	25 minutes ago	Up 25 minutes	0.0.0.0:8080->8080/tcp
7027825f2f2d	gcr.io/k8s-minikube/kicbase:v0.0.45	"/usr/local/bin/entr..."	7 days ago	Exited (255) About an hour ago	127.0.0.1:32772->22/tcp, 127.0.0.1:32771->2376/tcp, 127.0.0.1:32770->5000/tcp, 127.0.0.1:32769->8443/tcp, 127.0.0.1:32768->32443/tcp

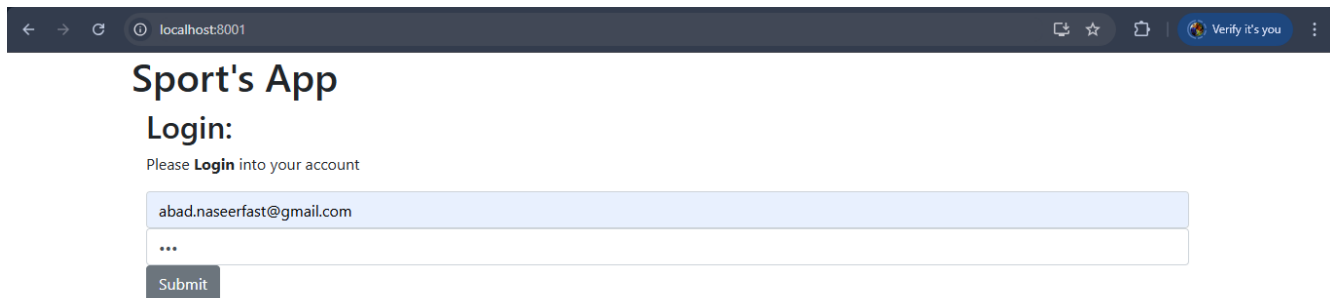
But docker-compose will handle the mongodb part as well by making the communication b/w the frontend, backend and the mongodb for each service. Here is the completed image and running container of the service (episode-01).

```

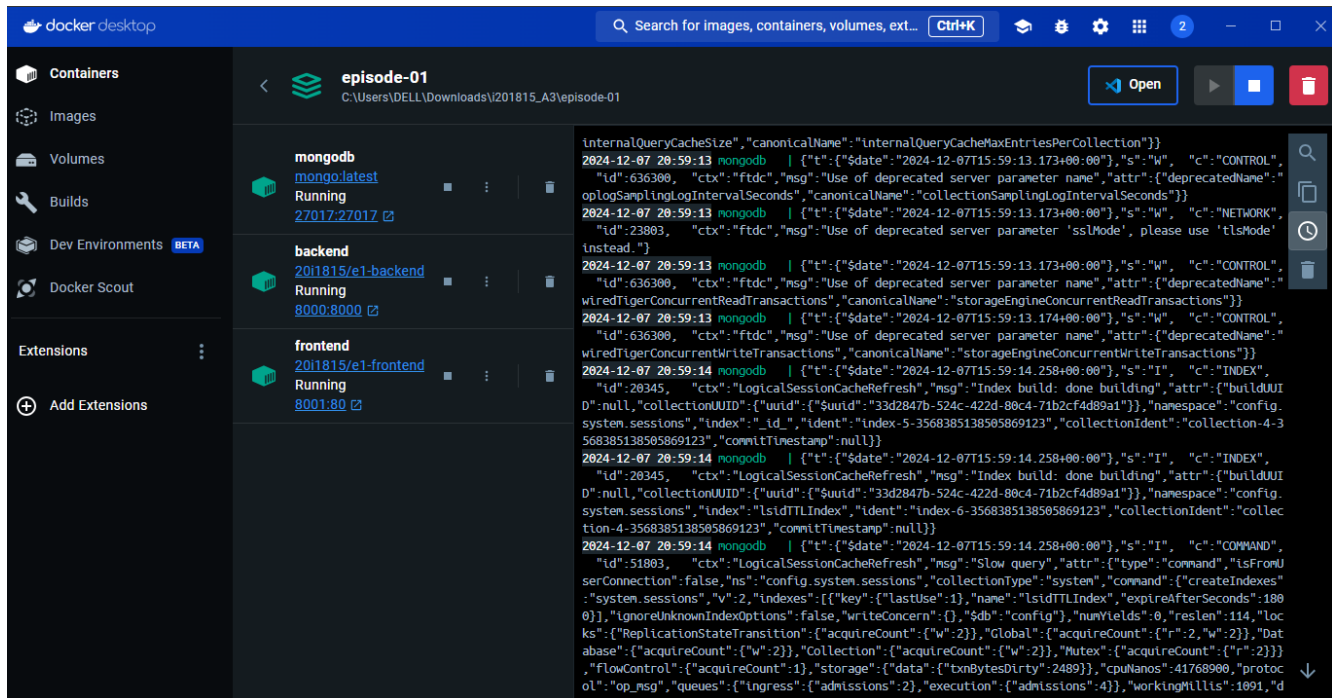
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\DELL\Downloads\i201815_A3\episode-01> docker-compose up --build
>>
[+] Building 1.4s (11/11)                                docker:default
[+] Building 30.8s (25/25) FINISHED                      docker:default
=> [backend internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 517B 0.0s
=> [backend internal] load metadata for docker.io/library/node:18 1.0s
=> [backend internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [backend internal] load build context 0.1s
=> => transferring context: 981B 0.0s
=> [backend 1/6] FROM docker.io/library/node:18@sha256:b57ae84fe7880a23b389f8260d726b784010ed470c2ee26d4e2cbdb955d25b12 0.0s
=> CACHED [backend 2/6] WORKDIR /app 0.0s
=> CACHED [backend 3/6] COPY package.json package-lock.json ./ 0.0s
=> CACHED [backend 4/6] RUN npm install 0.0s
=> CACHED [backend 5/6] COPY . . 0.0s
=> CACHED [backend 6/6] WORKDIR /app/src 0.0s
=> [backend] exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:3e3cf27bafbea002a51ad6c3b2c1b9aa959732165829364b11bdaf2195fedd2 0.0s
=> => naming to docker.io/library/episode-01-backend 0.0s
=> [frontend internal] load build definition from Dockerfile 0.1s

```

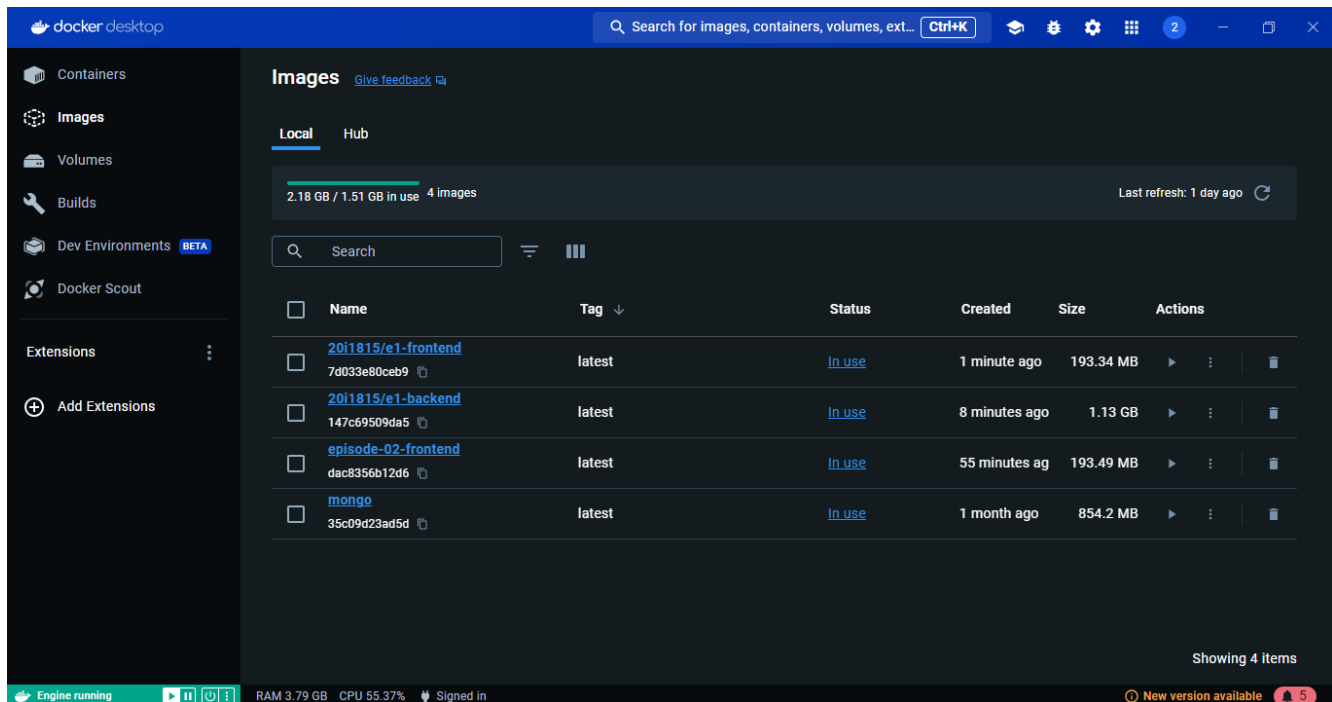
And we can access these routes on localhost:



And here is the docker desktop containers for episode-01 service running:



And here are the images:



Episode-02:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\DELL\Downloads\i201815_A3\episode-02> docker-compose up --build
>>
[+] Building 0.0s (0/0)  docker:default
[+] Building 7.3s (12/12)
[+] Building 12.4s (27/27) FINISHED
=> [backend internal] load build definition from Dockerfile
=> => transferring dockerfile: 517B
=> [backend internal] load metadata for docker.io/library/node:18
=> [backend auth] library/node:pull token for registry-1.docker.io
=> [backend internal] load .dockerignore
=> => transferring context: 2B
=> [backend 1/6] FROM docker.io/library/node:18@sha256:b57ae84fe7880a23b389f8260d726b784010ed470c2ee26d4e2cbdb955d25b12
=> [backend internal] load build context
=> => transferring context: 981B
=> CACHED [backend 2/6] WORKDIR /app
=> CACHED [backend 3/6] COPY package.json package-lock.json ./
=> CACHED [backend 4/6] RUN npm install
=> CACHED [backend 5/6] COPY . .
=> CACHED [backend 6/6] WORKDIR /app/src
=> [backend] exporting to image
=> => exporting layers
=> => writing image sha256:d3b71999c54f6f6b8e3c105798753ac9111d1c36f54267988c5e2d9dfb9cf996
=> => naming to docker.io/library/episode-02-backend
=> [frontend internal] load build definition from Dockerfile
=> => transferring dockerfile: 319B
=> [frontend internal] load metadata for docker.io/library/nginx:latest
=> [frontend internal] load metadata for docker.io/library/node:18-alpine
=> [frontend auth] library/nginx:pull token for registry-1.docker.io

```

And here is the docker desktop log that shows the episode is running perfectly:

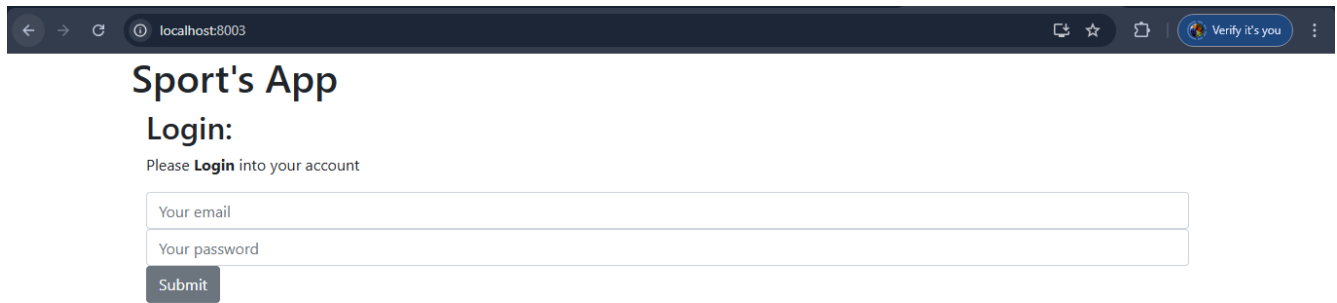
The screenshot shows the Docker Desktop interface with the 'episode-02' container stack. The stack consists of three containers: 'mongodb1' (mongo:latest), 'backend1' (episode-02-backend), and 'frontend1' (episode-02-frontend). The logs for 'frontend1' show successful connections to 'mongodb1' and 'listening on 8002'.

```

2024-12-07 20:09:43 frontend1 | 172.18.0.1 - - [07/Dec/2024:15:09:43 +0000] "GET /static/js/2.1bd72835.chunk.js HTTP/1.1" 200 286631 "http://localhost:8003/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36" "-"
2024-12-07 20:09:44 frontend1 | 172.18.0.1 - - [07/Dec/2024:15:09:44 +0000] "GET /favicon.ico HTTP/1.1" 200 3150 "http://localhost:8003/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36" "-"
2024-12-07 20:09:44 frontend1 | 172.18.0.1 - - [07/Dec/2024:15:09:44 +0000] "GET /manifest.json HTTP/1.1" 200 492 "http://localhost:8003/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36" "-"
2024-12-07 20:09:44 frontend1 | 172.18.0.1 - - [07/Dec/2024:15:09:44 +0000] "GET /logo192.png HTTP/1.1" 200 5347 "http://localhost:8003/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36" "-"
2024-12-07 20:07:40 backend1 | MongoDB connected successfully!
2024-12-07 20:07:40 backend1 | Listening on 8002
2024-12-07 20:05:51 mongodb1 | {"t":{"$date":"2024-12-07T15:05:51.165+00:00"},"s":"I", "c":"CONTR OL", "id":"23285", "ctx":"main","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"
2024-12-07 20:05:51 mongodb1 | {"t":{"$date":"2024-12-07T15:05:51.169+00:00"},"s":"I", "c":"CONTR OL", "id":"5945603", "ctx":"main","msg":"Multi threading initialized"}
2024-12-07 20:05:51 mongodb1 | {"t":{"$date":"2024-12-07T15:05:51.189+00:00"},"s":"I", "c":"NETWO RK", "id":"4648601", "ctx":"main","msg":"Implicit TCP FastOpen unavailable. If TCP FastOpen is requir ed, set at least one of the related parameters","attr":{"relatedParameters":["tcpFastOpenServer","tc pFastOpenClient","tcpFastOpenQueueSize"]}}
2024-12-07 20:05:51 mongodb1 | {"t":{"$date":"2024-12-07T15:05:51.195+00:00"},"s":"I", "c":"NETWO RK", "id":"4915701", "ctx":"main","msg":"Initialized wire specification","attr":{"spec":{"incomingExt ernalClient":{"minWireVersion":0,"maxWireVersion":25},"incomingInternalClient":{"minWireVersion":0, "maxWireVersion":25},"outgoing":{"minWireVersion":6,"maxWireVersion":25},"isInternalClient":true}}}
2024-12-07 20:05:51 mongodb1 | {"t":{"$date":"2024-12-07T15:05:51.206+00:00"},"s":"I", "c":"TENAN T_M", "id":"7091600", "ctx":"main","msg":"Starting TenantMigrationAccessBlockerRegistry"}

```

And when accessed on browser, it shows the view:



Sport's App

Login:

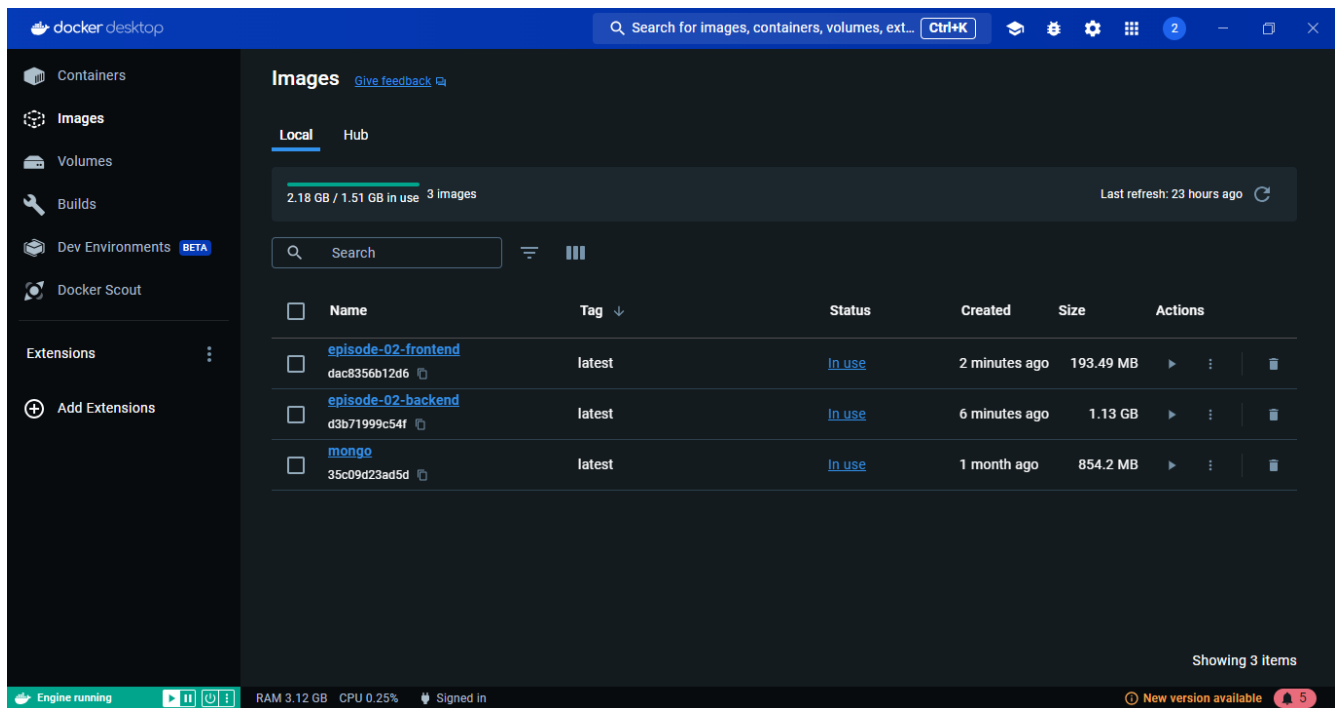
Please **Login** into your account

Your email

Your password

Submit

And here are the images for episodes-02:



docker desktop

Search for images, containers, volumes, ext... **Ctrl+K**

Containers

Images

Volumes

Builds

Dev Environments **BETA**

Docker Scout

Extensions

Add Extensions

Images [Give feedback](#)

Local Hub

2.18 GB / 1.51 GB in use 3 images Last refresh: 23 hours ago

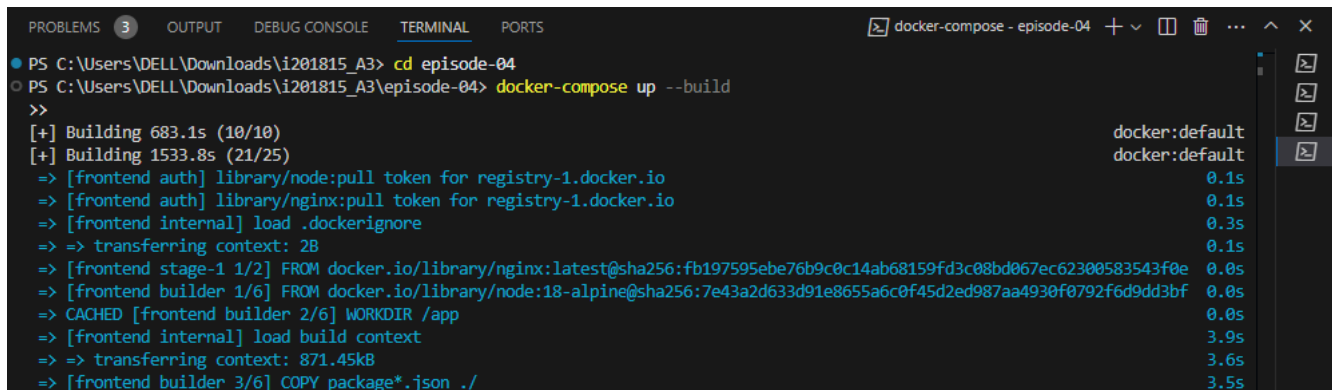
Search

Name	Tag	Status	Created	Size	Actions
episode-02-frontend dac8356b12d6	latest	In use	2 minutes ago	193.49 MB	▶ ⋮ 🗑️
episode-02-backend d3b71999c54f	latest	In use	6 minutes ago	1.13 GB	▶ ⋮ 🗑️
mongo 35c09d23ad5d	latest	In use	1 month ago	854.2 MB	▶ ⋮ 🗑️

Showing 3 items

Engine running RAM 3.12 GB CPU 0.25% Signed in New version available 5

Now here we will find the episode-03:

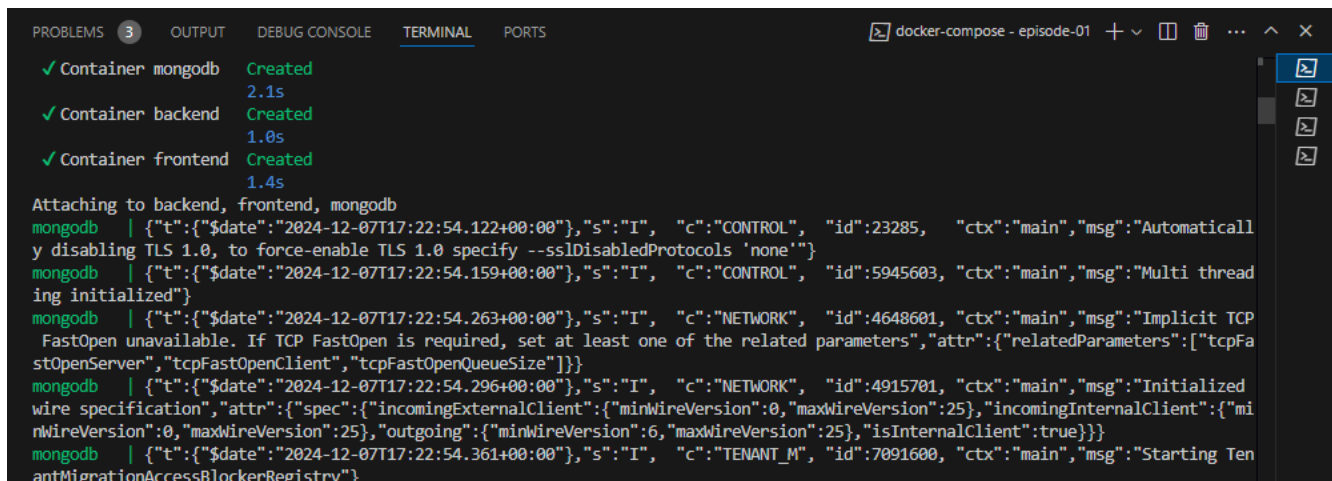


```

PS C:\Users\DELL\Downloads\i201815_A3> cd episode-04
PS C:\Users\DELL\Downloads\i201815_A3\episode-04> docker-compose up --build
>>
[+] Building 683.1s (10/10)                                docker:default
[+] Building 1533.8s (21/25)                                docker:default
=> [frontend auth] library/node:pull token for registry-1.docker.io 0.1s
=> [frontend auth] library/nginx:pull token for registry-1.docker.io 0.1s
=> [frontend internal] load .dockerignore 0.3s
=> => transferring context: 2B 0.1s
=> [frontend stage-1 1/2] FROM docker.io/library/nginx:latest@sha256:fb197595ebe76b9c0c14ab68159fd3c08bd067ec62300583543f0e 0.0s
=> [frontend builder 1/6] FROM docker.io/library/node:18-alpine@sha256:7e43a2d633d91e8655a6c0f45d2ed987aa4930f0792f6d9dd3bf 0.0s
=> CACHED [frontend builder 2/6] WORKDIR /app 0.0s
=> [frontend internal] load build context 3.9s
=> => transferring context: 871.45kB 3.6s
=> [frontend builder 3/6] COPY package*.json ./ 3.5s

```

Here you will find image creation and docker container building for episode-05:



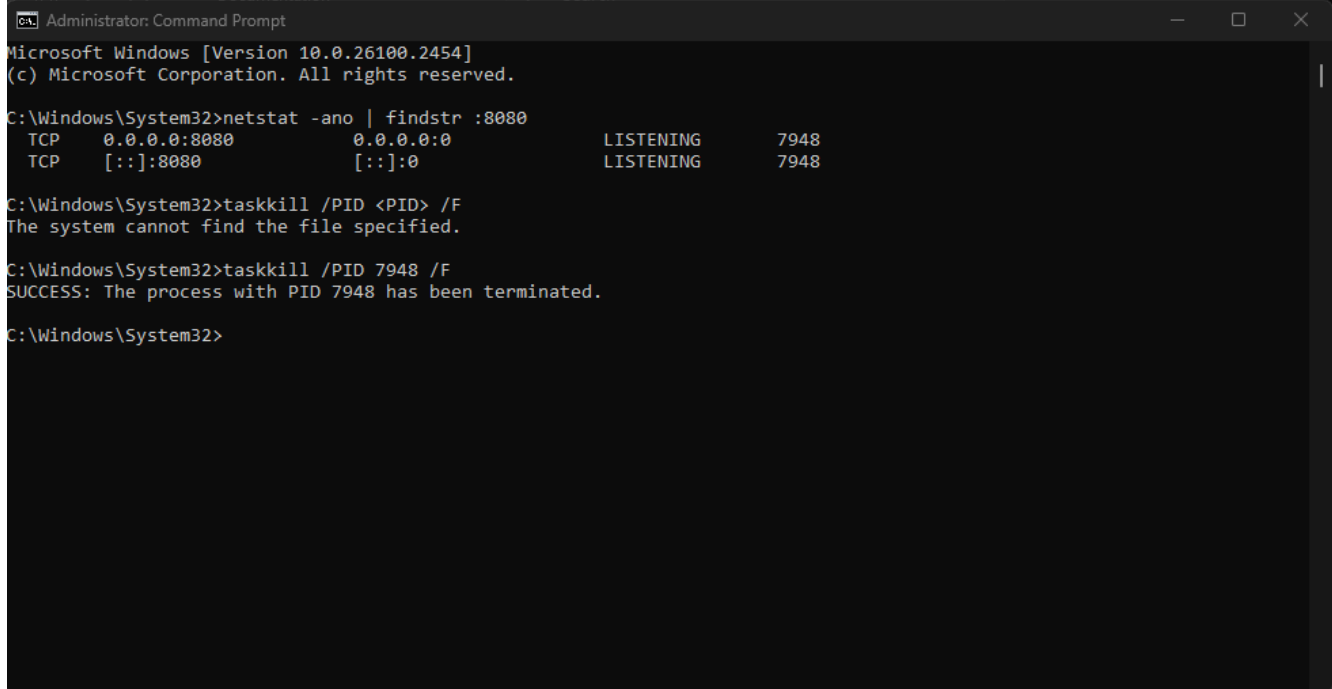
```

✓ Container mongodb Created 2.1s
✓ Container backend Created 1.0s
✓ Container frontend Created 1.4s
Attaching to backend, frontend, mongodb
mongodb | {"t":{"$date":"2024-12-07T17:22:54.122+00:00"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"main","msg":"Automaticall
y disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
mongodb | {"t":{"$date":"2024-12-07T17:22:54.159+00:00"},"s":"I", "c":"CONTROL", "id":5945603, "ctx":"main","msg":"Multi thread
ing initialized"}
mongodb | {"t":{"$date":"2024-12-07T17:22:54.263+00:00"},"s":"I", "c":"NETWORK", "id":4648601, "ctx":"main","msg":"Implicit TCP
FastOpen unavailable. If TCP FastOpen is required, set at least one of the related parameters","attr":{"relatedParameters":{"tcpFa
stOpenServer","tcpFastOpenClient","tcpFastOpenQueueSize"}}}
mongodb | {"t":{"$date":"2024-12-07T17:22:54.296+00:00"},"s":"I", "c":"NETWORK", "id":4915701, "ctx":"main","msg":"Initialized
wire specification","attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":25},"incomingInternalClient":{"mi
nWireVersion":0,"maxWireVersion":25},"outgoing":{"minWireVersion":6,"maxWireVersion":25},"isInternalClient":true}}}
mongodb | {"t":{"$date":"2024-12-07T17:22:54.361+00:00"},"s":"I", "c":"TENANT_M", "id":7091600, "ctx":"main","msg":"Starting Ten
antMigrationAccessBlockerRegistry"}

```

Issues and Resolutions

- Issue: Port conflicts during initial runs.
 - Resolution: Adjusted the ports in the `docker run` command.



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.26100.2454]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>netstat -ano | findstr :8080
  TCP    0.0.0.0:8080      0.0.0.0:0        LISTENING       7948
  TCP    [::]:8080        [::]:0           LISTENING       7948

C:\Windows\System32>taskkill /PID <PID> /F
The system cannot find the file specified.

C:\Windows\System32>taskkill /PID 7948 /F
SUCCESS: The process with PID 7948 has been terminated.

C:\Windows\System32>
```

- Issue: Dependency errors during `npm install`.
 - Resolution: Updated `package.json` and re-installed dependencies. Error giving me building dependencies so I added this in `package.json` to resolve (episode-01-frontend): `"build": "react-scripts --openssl-legacy-provider build"`,

Part 2: Kubernetes Deployment (50 Points)

Task Overview

1. **Setting Up a Local Kubernetes Cluster**
 - Minikube was used to set up the cluster.

```
minikube start
```

The screenshot shows the Docker Desktop application window. In the background, the Docker Desktop interface is visible with the 'General' tab selected. In the foreground, a 'Command Prompt' window is open, displaying the output of the 'minikube start' command. The output shows the process of starting the minikube cluster, including pulling the base image, creating the docker container, and configuring Kubernetes components. The command prompt shows the user is at the C:\Users\DELL> prompt.

```

C:\Users\DELL>minikube start
W1207 21:07:18.864428 19256 main.go:291] Unable to resolve the current Docker CLI context "default": context "default"
: context not found: open C:\Users\DELL\.docker\contexts\meta\37a8eec1ce19687d132fe29051dca629d164e2c4958ba141d5f4133a33
f0688f\meta.json: The system cannot find the path specified.
* minikube v1.34.0 on Microsoft Windows 11 Pro 10.0.26100.2454 Build 26100.2454
* Using the docker driver based on existing profile
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.45 ...
* docker "minikube" container is missing, will recreate.
* Creating docker container (CPUs=2, Memory=2200MB) ...
! Failing to connect to https://registry.k8s.io/ from both inside the minikube container and host machine
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking
/proxy/
* Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
- Generating certificates and keys ...
- Booting up control plane ...
- Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
- Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

C:\Users\DELL>

```

2. Namespace Configuration

- Namespaces were created for development and production environments.

- ```
kubectl create namespace development
```

```
kubectl create namespace production
```

The screenshot shows a 'Command Prompt' window with the output of several 'kubectl' commands. The first command, 'kubectl create namespace development', results in an error because '3.kubectl' is not recognized. The second command, 'kubectl create namespace development', successfully creates the 'development' namespace. The third command, 'kubectl create namespace production', successfully creates the 'production' namespace. The prompt is currently at C:\Users\DELL>.

```

* minikube v1.34.0 on Microsoft Windows 11 Pro 10.0.26100.2454 Build 26100.2454
* Using the docker driver based on existing profile
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.45 ...
* docker "minikube" container is missing, will recreate.
* Creating docker container (CPUs=2, Memory=2200MB) ...
! Failing to connect to https://registry.k8s.io/ from both inside the minikube container and host machine
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking
/proxy/
* Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
- Generating certificates and keys ...
- Booting up control plane ...
- Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
- Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

C:\Users\DELL>3.kubectl create namespace development
'3.kubectl' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\DELL>kubectl create namespace development
namespace/development created

C:\Users\DELL>kubectl create namespace production
namespace/production created

C:\Users\DELL>

```

## 4. Module Containerization

- Each module's dependencies were encapsulated in their respective containers.

## **Part 3: Integration and Final Deployment (20 Points)**

### **● Kubectl Deployment:**

#### **○ Episode 1:**

Kubectl apply -f k8

#### **○ Episode 2:**

Kubectl apply -f k8

#### **○ Episode 3:**

Kubectl apply -f k8

#### **○ Episode 4:**

Kubectl apply -f k8

#### **○ Episode 5:**

Kubectl apply -f k8

Kubectl apply -f k8

Kubectl apply -f k8

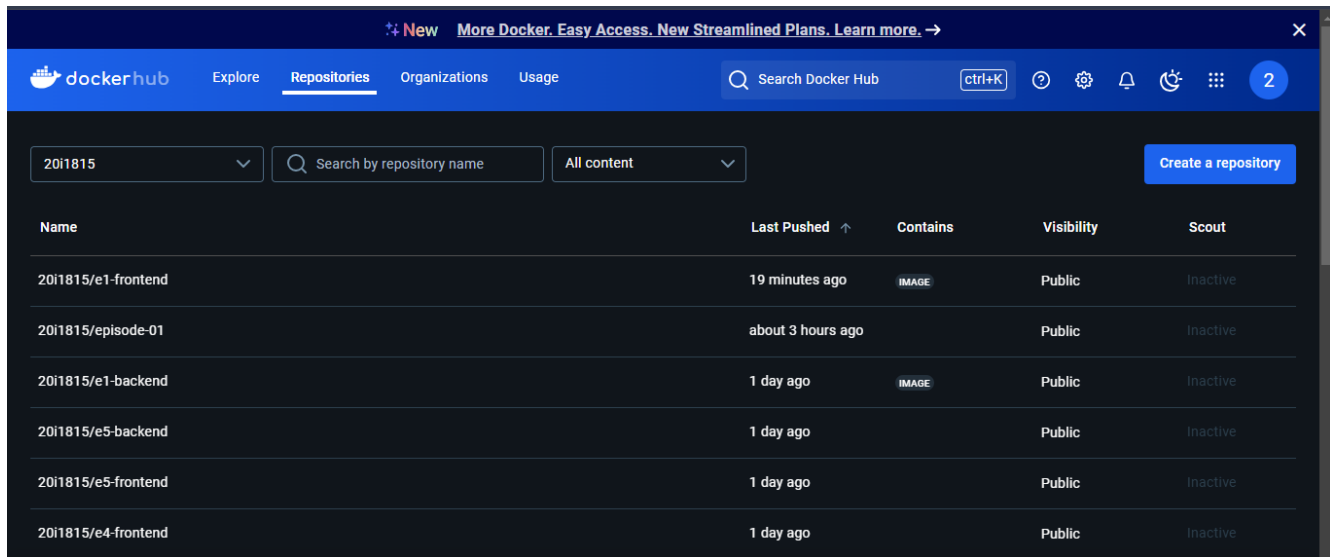
### **● Status:**

```
PS C:\Users\DELL\Downloads\i201815_A3\episode-01> kubectl get pods
NAME READY STATUS RESTARTS AGE
backend-75b98c4ccc-tphb5 0/1 CrashLoopBackOff 13 (73s ago) 57m
frontend-5b7cdd5fb7-7j66t 0/1 ImagePullBackOff 0 57m
mongodb-7bbb88dc8b-jwcjs 1/1 Running 1 (13m ago) 50m
```

### **● Scaling:**

```
PS C:\Users\DELL\Downloads\i201815_A3\episode-01> kubectl scale deployment frontend --replicas=3
deployment.apps/frontend scaled
```

### **● Docker hub Images:**

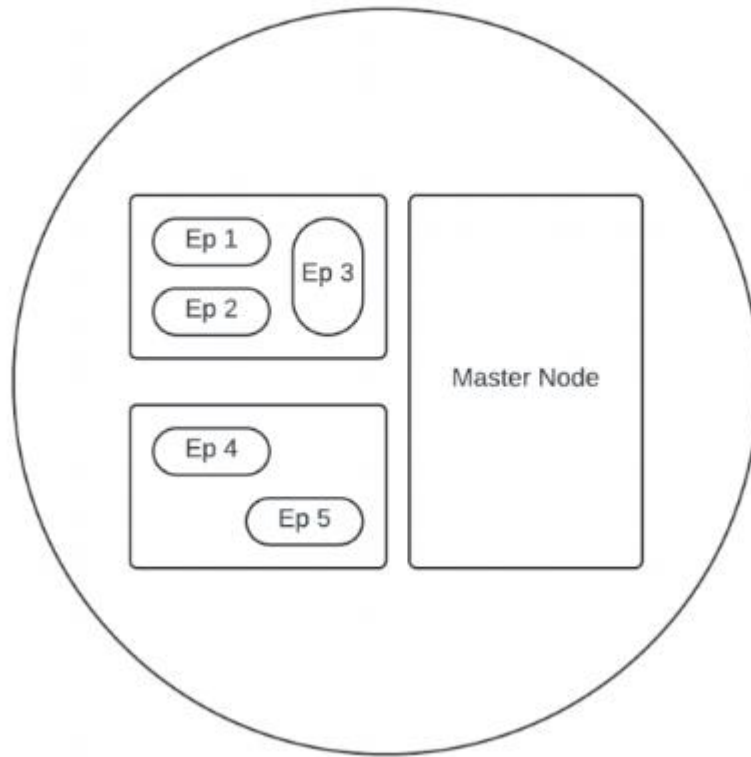


- **Monitoring:**

minikube -p assignment addons enable metrics-server minikube dashboard --profile=assignment

- **Kubernetes Architecture:**

- **Kubernetes Architecture:**



Thank You!