DevOps Assignment

Multi-Container Application Deployment with Docker Compose and Kubernetes

Here is the repo :

- Backend: https://github.com/Anand-1432/Techdome-backend
- Frontend: https://github.com/Anand-1432/Techdome-frontend

Task:

- Build a multi-container application with at least 3 containers (e.g., frontend, backend, database).
- Use Docker Compose to define the application and its dependencies.
- Deploy the application to a local Kubernetes cluster using Minikube or through docker.
- Demonstrate the application functionality and explain your deployment strategy.

Requires knowledge of Docker Compose, Kubernetes deployments, and container networking.

Tools: Docker, Docker Compose, Minikube, Kubernetes

Deliverables:

- Docker Compose file: A properly formatted Docker Compose file defining the application architecture and its dependencies.
- Kubernetes deployment manifests: YAML files defining the deployment of each container in the Kubernetes cluster (Optional).
- Documentation: A document explaining the application architecture, deployment strategy, and instructions for building, deploying, and managing the application.
- Demonstration: A recording or screenshot showcasing the application's functionality and providing a detailed explanation of the deployment approach

Bonus points:

- Automate infrastructure scaling based on load or resource utilization Terraform .
- Implement unit tests for your Terraform code and automation scripts.
- Demonstrate a rollback strategy for infrastructure changes.

These tasks can be completed using free tools and resources, making them accessible for everyone. **Remember to document your approach, solutions implemented, and**

challenges faced. Upload your completed work to a public GitHub repository and be prepared to discuss your solutions. Good luck!

DevOps Assignment
By
A Dinesh Kumar Reddy

To complete the Multi-Container Application Deployment using Docker Compose and Kubernetes, I will outline the necessary steps, provide the required files (Docker Compose, Kubernetes manifests, and documentation), and walk you through the dependencies.

Prerequisites: Tools to Install

Docker: Install Docker to build, ship, and run containerized applications.

Docker Compose: To define and run multi-container Docker applications.

Minikube: Local Kubernetes cluster to test and run the application.

Kubectl: Command-line tool for interacting with Kubernetes clusters.

Step-1: Creation of EC2 and Login into the server

- Created the server with Ubuntu Operating system
- Configured all the details like instance type, Security Groups, Key pair, Storage and all the details that are needed
- Log in to the server by using the SSH key

Step-2: Installing GIT and cloning of the Repositories to the Server

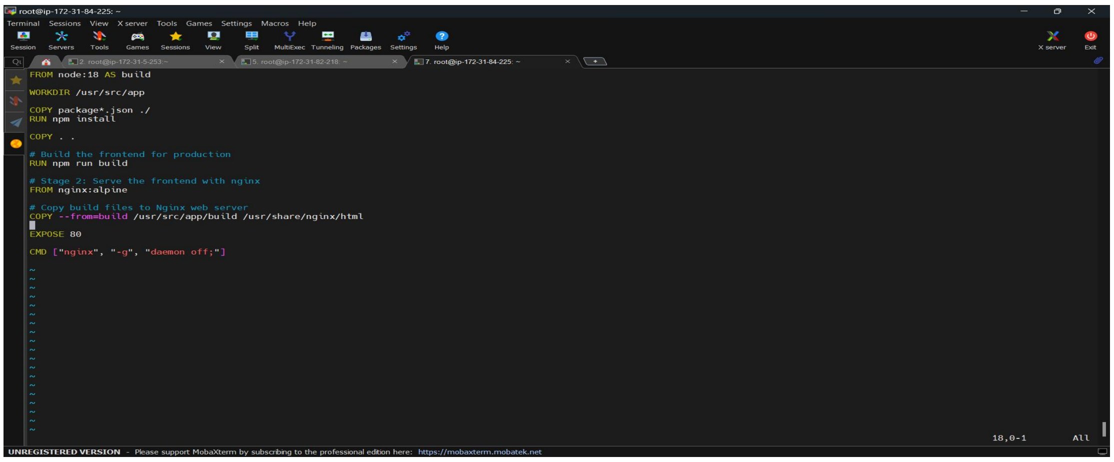
- apt install git -y
- Git clone https://github.com/Anand-1432/Techdome-frontend.git
- Git clone https://github.com/Anand-1432/Techdome-backend.git

Step-3: Creating Dockerfile for Frontend

- Created the Dockerfile for frontend to build the images in the Techdone-frontend directory
- Builted the custome image by using the following command

  - **Docker build -d <image_name>:<tag> .**

    - . Refers to the current locaiton



```
FROM node:18 AS build

WORKDIR /usr/src/app

COPY package*.json ./
RUN npm install

COPY . .

# Build the frontend for production
RUN npm run build

# Stage 2: Serve the frontend with nginx
FROM nginx:alpine

# Copy build files to Nginx web server
COPY --from=build /usr/src/app/build /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

root@ip-172-31-84-225: ~
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
2. root@ip-172-31-5-253:~
5. root@ip-172-31-82-218: ~
7. root@ip-172-31-84-225: ~
18,0-1 All
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

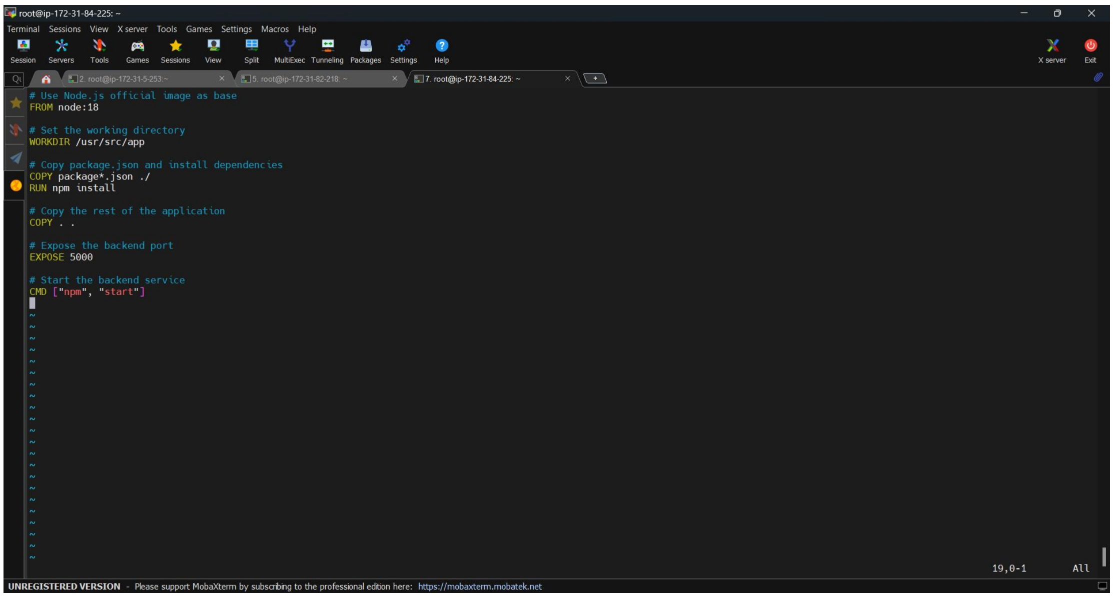
\

Step-4: Creating Dockerfile for Backend

- ✧ Created the Dockerfile for baclend to build the images in the Techdone-backend directory
- ✧ Builted the custome image by using the following command

  - ➢ **Docker build -d <image_name>:<tag> .**

    - ◆ . Refers to the current locaiton

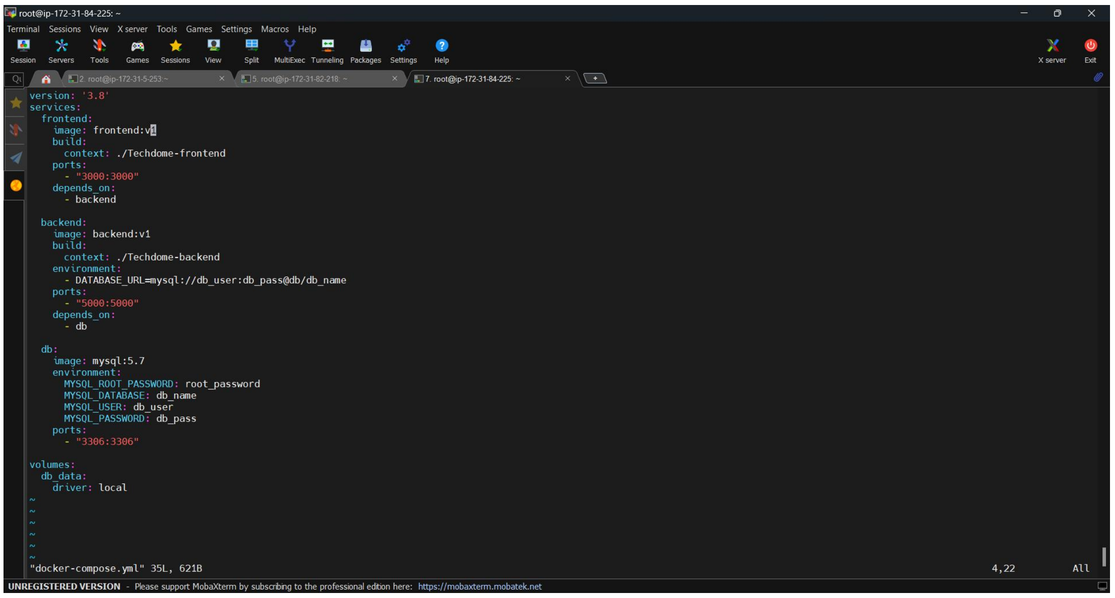


```
# Use Node.js official image as base
FROM node:18

# Set the working directory
WORKDIR /usr/src/app

# Copy package.json and install dependencies
COPY package*.json ./
RUN npm install

# Copy the rest of the application
COPY . .

# Expose the backend port
EXPOSE 5000

# Start the backend service
CMD ["npm", "start"]
```

root@ip-172-31-84-225: ~
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
2. root@ip-172-31-5-253:~
5. root@ip-172-31-82-218: ~
7. root@ip-172-31-84-225: ~
19,0-1 All
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

Step-5: Creating the Docker Compose File

- ✧ Create a docker-compose.yml in your root directory that defines our frontend, backend, and database services.

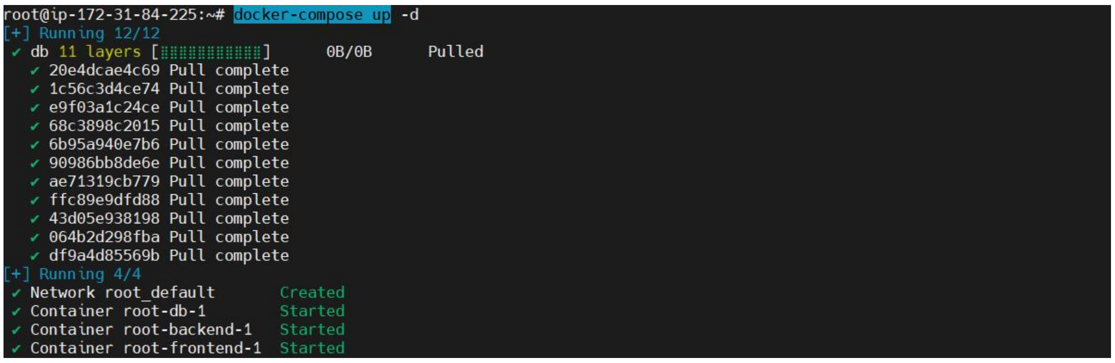


```
version: '3.8'
services:
  frontend:
    image: frontend:v1
    build:
      context: ./Techdome-frontend
    ports:
      - "3000:3000"
    depends_on:
      - backend

  backend:
    image: backend:v1
    build:
      context: ./Techdome-backend
    environment:
      - DATABASE_URL=mysql://db_user:db_pass@db/db_name
    ports:
      - "5000:5000"
    depends_on:
      - db

  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: root_password
      MYSQL_DATABASE: db_name
      MYSQL_USER: db_user
      MYSQL_PASSWORD: db_pass
    ports:
      - "3306:3306"

volumes:
  db_data:
    driver: local
~
~
~
~
~
~
"docker-compose.yml" 35L, 621B                                                    4,22          All
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

Step-6: Build the Docker-compose file

✧ Build the docker-compose file using the command **docker-compose up -d**



```
root@ip-172-31-84-225:~# docker-compose up -d
[+] Running 12/12
 ✔ db 11 layers [⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled
   ✔ 20e4dcae4c69 Pull complete
   ✔ 1c56c3d4ce74 Pull complete
   ✔ e9f03a1c24ce Pull complete
   ✔ 68c3898c2015 Pull complete
   ✔ 6b95a940e7b6 Pull complete
   ✔ 90986bb8de6e Pull complete
   ✔ ae71319cb779 Pull complete
   ✔ ffc89e9dfd88 Pull complete
   ✔ 43d05e938198 Pull complete
   ✔ 064b2d298fba Pull complete
   ✔ df9a4d85569b Pull complete
[+] Running 4/4
 ✔ Network root_default        Created
 ✔ Container root-db-1         Started
 ✔ Container root-backend-1    Started
 ✔ Container root-frontend-1   Started
```

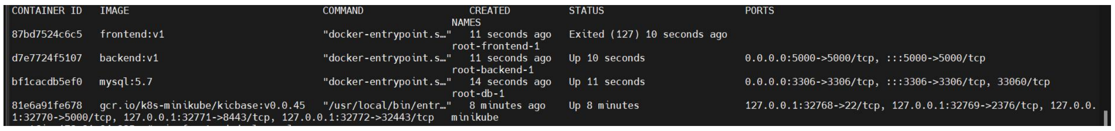
✧ After running the above command the container will be created. We can check it with the below command

➢ **Docker ps**

◆ It will get only the running containers present in the server

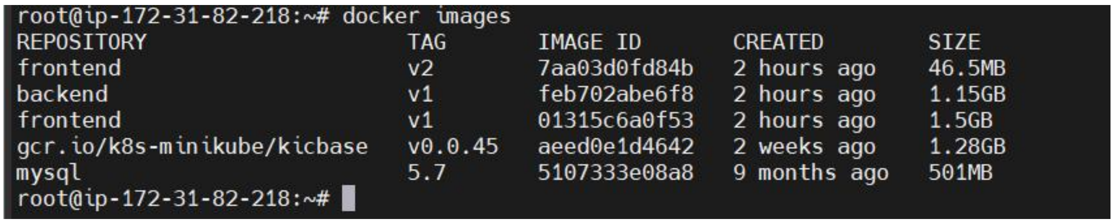
➢ **Docker ps -a**

◆ It will get all the containers including the stop containers



```
CONTAINER ID   IMAGE                                  COMMAND                  CREATED          STATUS                      PORTS
                                                                              NAMES
87bd7524c6c5   frontend:v1                            "docker-entrypoint.s…"   11 seconds ago   Exited (127) 10 seconds ago
                                                                              root-frontend-1
d7e7724f5107   backend:v1                             "docker-entrypoint.s…"   11 seconds ago   Up 10 seconds               0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
                                                                              root-backend-1
bf1cacdb5ef0   mysql:5.7                              "docker-entrypoint.s…"   14 seconds ago   Up 11 seconds               0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp
                                                                              root-db-1
81e6a91fe678   gcr.io/k8s-minikube/kicbase:v0.0.45    "/usr/local/bin/entr…"   8 minutes ago    Up 8 minutes                127.0.0.1:32768->22/tcp, 127.0.0.1:32769->2376/tcp, 127.0.0.
1:32770->5000/tcp, 127.0.0.1:32771->8443/tcp, 127.0.0.1:32772->32443/tcp   minikube
```

✧ To list all the images present in the server, we use the command docker images



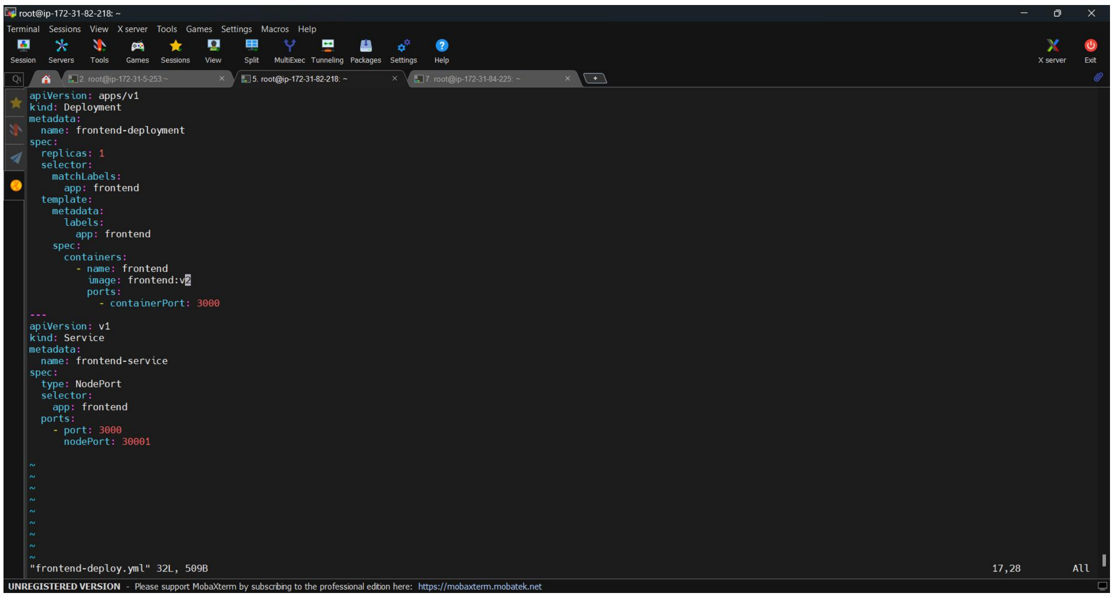
```
root@ip-172-31-82-218:~# docker images
REPOSITORY                     TAG       IMAGE ID       CREATED        SIZE
frontend                       v2        7aa03d0fd84b   2 hours ago    46.5MB
backend                        v1        feb702abe6f8   2 hours ago    1.15GB
frontend                       v1        01315c6a0f53   2 hours ago    1.5GB
gcr.io/k8s-minikube/kicbase    v0.0.45   aeed0e1d4642   2 weeks ago    1.28GB
mysql                          5.7       5107333e08a8   9 months ago   501MB
root@ip-172-31-82-218:~#
```

Step-7: Accessing the application

- ✧ We can access the application from the browser by using the public ip along the port number of the application
  - Frontend: http://localhost:3000
  - Backend: http://localhost:5000

Step-8: Creation of K8s Deployment Manifest files

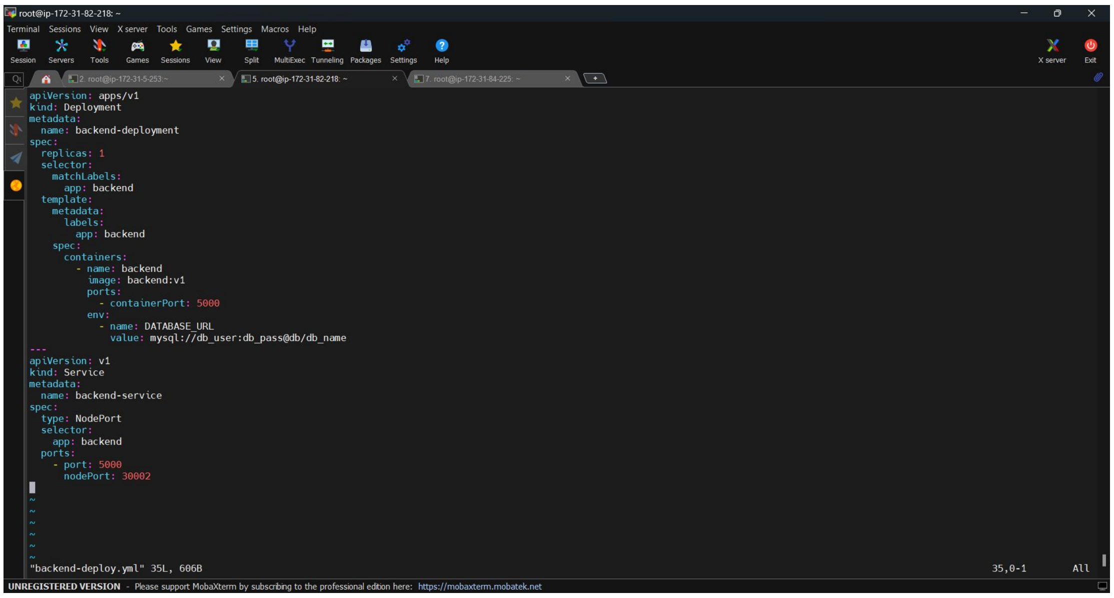
- ✧ We create 3 different Deployment YAML files along with the service node to expose our application
- ✧ **Frontend-deployment.yml file**


root@ip-172-31-82-218: ~
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
2. root@ip-172-31-5-253:~
5. root@ip-172-31-82-218: ~
7. root@ip-172-31-84-225: ~

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
        - name: frontend
          image: frontend:v2
          ports:
            - containerPort: 3000
---
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  type: NodePort
  selector:
    app: frontend
  ports:
    - port: 3000
      nodePort: 30001

~
~
~
~
~
~
~
~
~
"frontend-deploy.yml" 32L, 509B                                          17,28        All
```

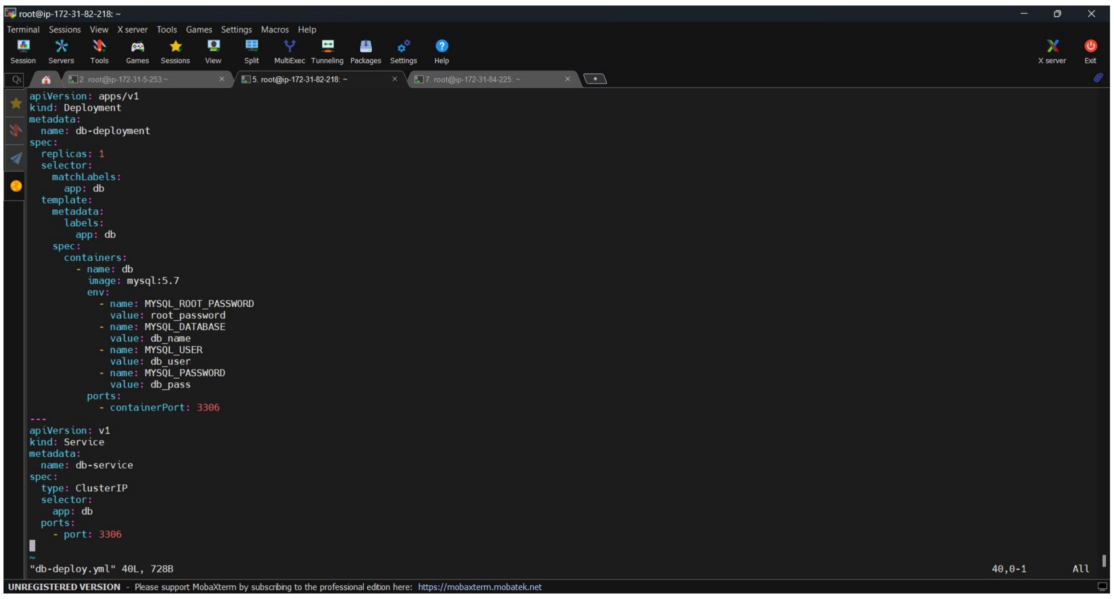
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

- ✧ **Backend-deployment.yml file**



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
        - name: backend
          image: backend:v1
          ports:
            - containerPort: 5000
          env:
            - name: DATABASE_URL
              value: mysql://db_user:db_pass@db/db_name
---
apiVersion: v1
kind: Service
metadata:
  name: backend-service
spec:
  type: NodePort
  selector:
    app: backend
  ports:
    - port: 5000
      nodePort: 30002

~
~
~
~
~
~
"backend-deploy.yml" 35L, 606B                                              35,0-1        All
```

✧ Database-deployment.yml file



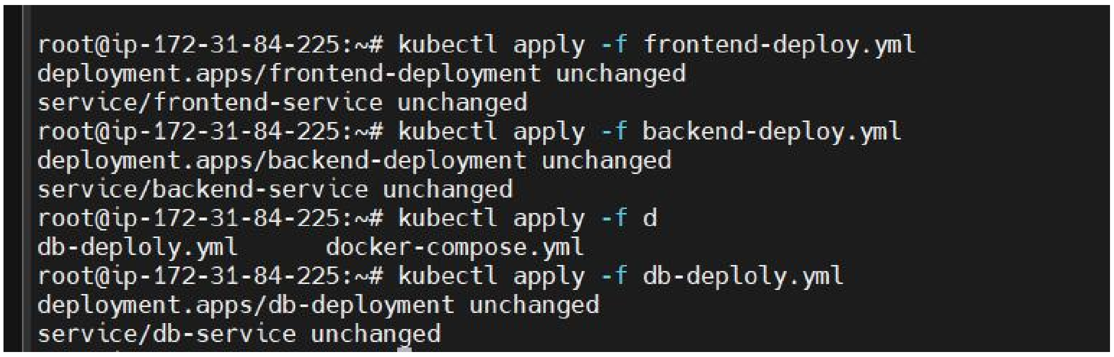
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: db-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: db
  template:
    metadata:
      labels:
        app: db
    spec:
      containers:
        - name: db
          image: mysql:5.7
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: root_password
            - name: MYSQL_DATABASE
              value: db_name
            - name: MYSQL_USER
              value: db_user
            - name: MYSQL_PASSWORD
              value: db_pass
          ports:
            - containerPort: 3306
---
apiVersion: v1
kind: Service
metadata:
  name: db-service
spec:
  type: ClusterIP
  selector:
    app: db
  ports:
    - port: 3306

"db-deploy.yml" 40L, 728B                                                   40,0-1        All
```

Step-9: Start the Minikube

✧ Start the minikube by the command **Minikube start**

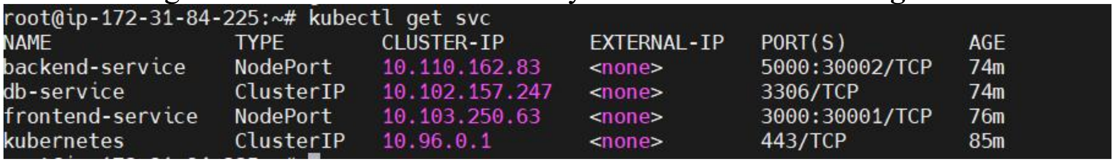
Step-10: Deploy to Kubernetes Using Minikube

✧ We deploy the manifest files to the kubernetes cluster by using the command
➢ **Kubectl apply -f <manifest_file_name>**


```
root@ip-172-31-84-225:~# kubectl apply -f frontend-deploy.yml
deployment.apps/frontend-deployment unchanged
service/frontend-service unchanged
root@ip-172-31-84-225:~# kubectl apply -f backend-deploy.yml
deployment.apps/backend-deployment unchanged
service/backend-service unchanged
root@ip-172-31-84-225:~# kubectl apply -f d
db-deploly.yml      docker-compose.yml
root@ip-172-31-84-225:~# kubectl apply -f db-deploly.yml
deployment.apps/db-deployment unchanged
service/db-service unchanged
```

Step-11: Getting the list of services

✧ We can get the list of services created by the command **Kubectl get svc**


```
root@ip-172-31-84-225:~# kubectl get svc
NAME               TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
backend-service    NodePort    10.110.162.83    <none>        5000:30002/TCP   74m
db-service         ClusterIP   10.102.157.247   <none>        3306/TCP         74m
frontend-service   NodePort    10.103.250.63    <none>        3000:30001/TCP   76m
kubernetes         ClusterIP   10.96.0.1        <none>        443/TCP          85m
```

Application Architecture

- **Frontend**: A React application served using Nginx.
- **Backend**: A Node.js/Express application that interacts with a MySQL database.
- **Database**: MySQL 5.7 instance.

Deployment Strategy

Docker Compose:

- Three containers (frontend, backend, database) are defined and run in a single Docker network.
- Dependencies are managed via depends_on in Docker Compose.

Kubernetes (Minikube):

- The application is divided into deployments for each service (frontend, backend, database).
- Kubernetes manages the replication, health, and scaling of these services.
- Each service is exposed via a NodePort to access them externally