

Task : Deploy Dockerized 3-Tier Architecture with Angular Frontend, Java Backend, and RDS on Kubernetes.

Prerequisites:

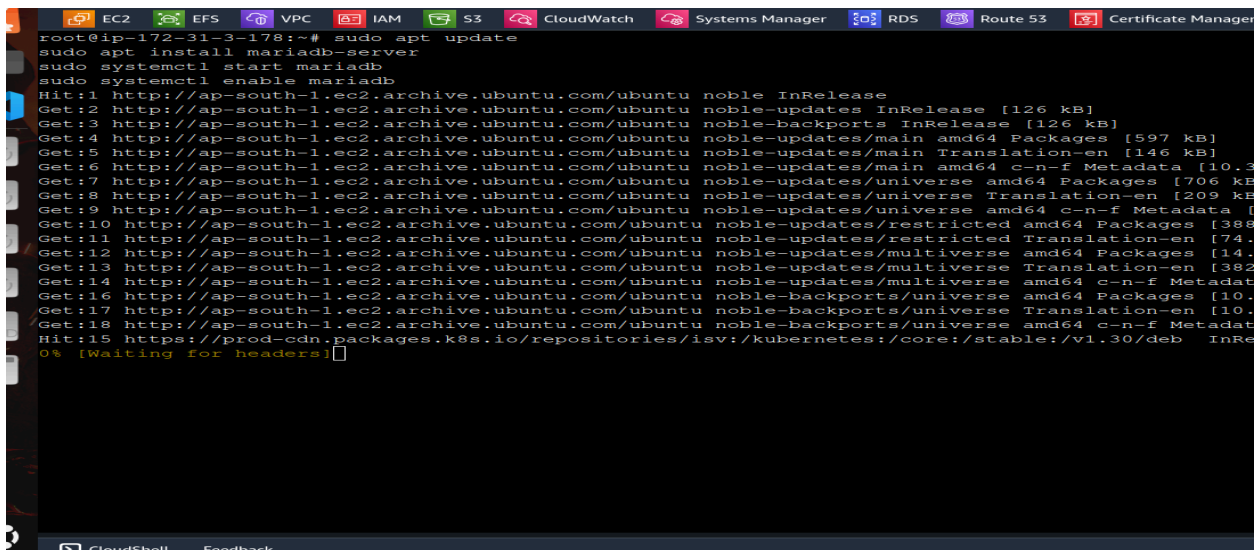
- Ensure Kubeadm, Kubelet, Kubectl, and Docker are installed on both master and worker nodes. Initialize the cluster on the master node with **kubeadm init** and join the worker nodes using **kubeadm join**.
- Reference link for installing and setup kubeadm
→ https://github.com/Gaurav1251/Devops_Tasks/tree/main/K8s/Kubeadm%20Installation%20and%20Configuration

Note: Perform the below steps only on master node.

Step 1: Install mariadb server on system (master node).

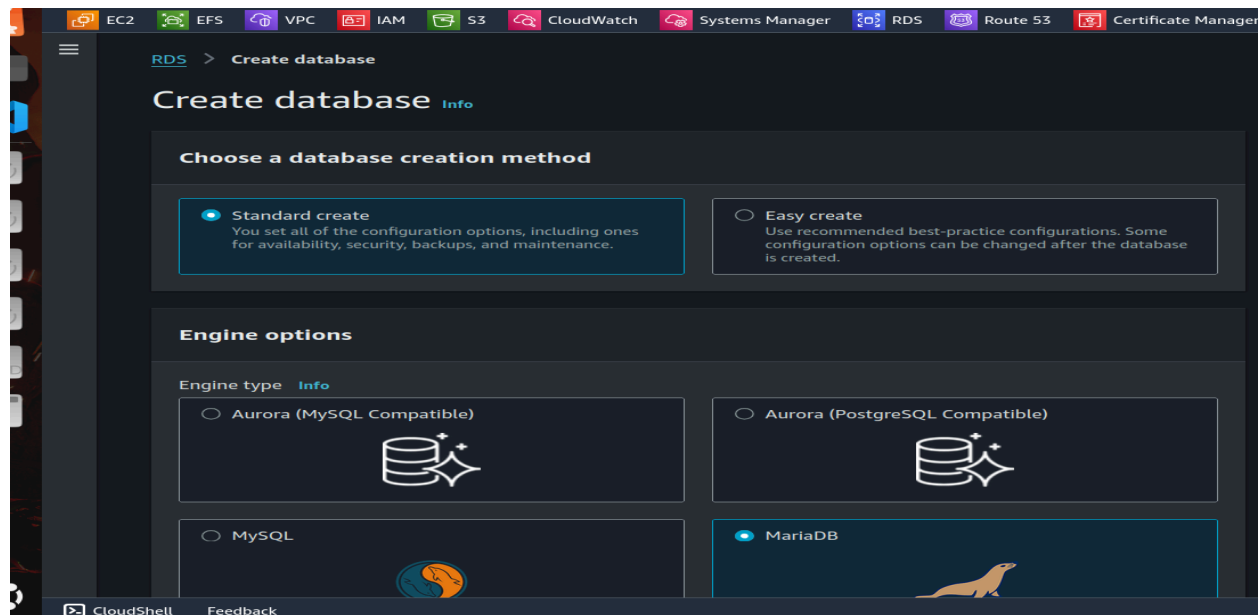
Commands

```
→sudo apt update
sudo apt install mariadb-server
sudo systemctl start mariadb
sudo systemctl enable mariadb
```



```
root@ip-172-31-3-178:~# sudo apt update
sudo apt install mariadb-server
sudo systemctl start mariadb
sudo systemctl enable mariadb
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [597 kB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [146 kB]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [10.3 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1705 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [209 kB]
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [10.3 kB]
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [388 kB]
Get:11 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [74.1 kB]
Get:12 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [14.1 kB]
Get:13 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [382 kB]
Get:14 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [10.3 kB]
Get:15 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [10.3 kB]
Get:16 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [10.3 kB]
Get:17 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 c-n-f Metadata [10.3 kB]
Hit:18 https://prod-cdn.packages.k8s.io/repositories/iso/v1/kubernetes:core/stable/v1.30/deb InRelease
0% [Waiting for headers]
```

Step 2: Create a RDS database with engine as mariadb.



Step 3: Now connect to the RDS database.

`mysql -h rds-endpoint -u admin -p`

1. After that create database springbackend in it .
2. After creating database exit and copy the springbackend.sql file to the database .

`mysql -h rds-endpoint -u admin -p springbackend < springbackend.sql`

```
root@ip-172-31-3-178:~# mysql -h database-1.c7kqi8k66m89.ap-south-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 57
Server version: 10.11.9-MariaDB-log managed by https://aws.amazon.com/rds/
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
MariaDB [(none)]> create database springbackend;
Query OK, 1 row affected (0.004 sec)

MariaDB [(none)]> exit
Bye
root@ip-172-31-3-178:~# ls
application.properties  backend-dp.yaml  backend-svc.yaml  frontend-dp.yaml  frontend-svc.yaml  snap  worker.service.ts
root@ip-172-31-3-178:~# nano springbackend.sql
root@ip-172-31-3-178:~# nano springbackend.sql
root@ip-172-31-3-178:~# mysql -h database-1.c7kqi8k66m89.ap-south-1.rds.amazonaws.com -u admin -p springbackend < springbackend.sql
Enter password:
root@ip-172-31-3-178:~#
```

Step 4: Now create application.properties which will be required to connect backend with our rds database.

Replace the rds-endpoint and username and password as per your configuration.

```
root@ip-172-31-3-178:~# nano application.properties
root@ip-172-31-3-178:~# cat application.properties
spring.datasource.url=jdbc:mysql://database-1.c7kqi8k66m89.ap-south-1.rds.amazonaws.com:3306/springbackend?useSSL=false
spring.datasource.username=admin
spring.datasource.password=12345678
server.port=8085

spring.jpa.generate-ddl=true
root@ip-172-31-3-178:~#
```

Step 5: Create Deployment and service yaml files for the backend pod.

```
root@ip-172-31-3-178:~# nano backend-dp.yaml
root@ip-172-31-3-178:~# cat backend-dp.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: backend-app
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: backend-app
    spec:
      containers:
      - name: backend-app
        image: gaurav1251/angularark8:ttbe
        ports:
        - name: backend
          containerPort: 8085
          protocol: TCP
root@ip-172-31-3-178:~#
```

```
root@ip-172-31-3-178:~# nano backend-svc.yaml
root@ip-172-31-3-178:~# cat backend-svc.yaml
apiVersion: v1
kind: Service
metadata:
  name: backend-service
spec:
  selector:
    app: backend-app
  ports:
  - name: backend
    targetPort: 8085
    port: 8085
    type: NodePort
root@ip-172-31-3-178:~#
```

We enter the container port =8085 because our backend image is exposed on 8085 port .

Also for the svc file we will be using NodePort because we want access the backend from frontend which cant be done by ClusterIP.

Step 6: Create deployment and svc file for the frontend.

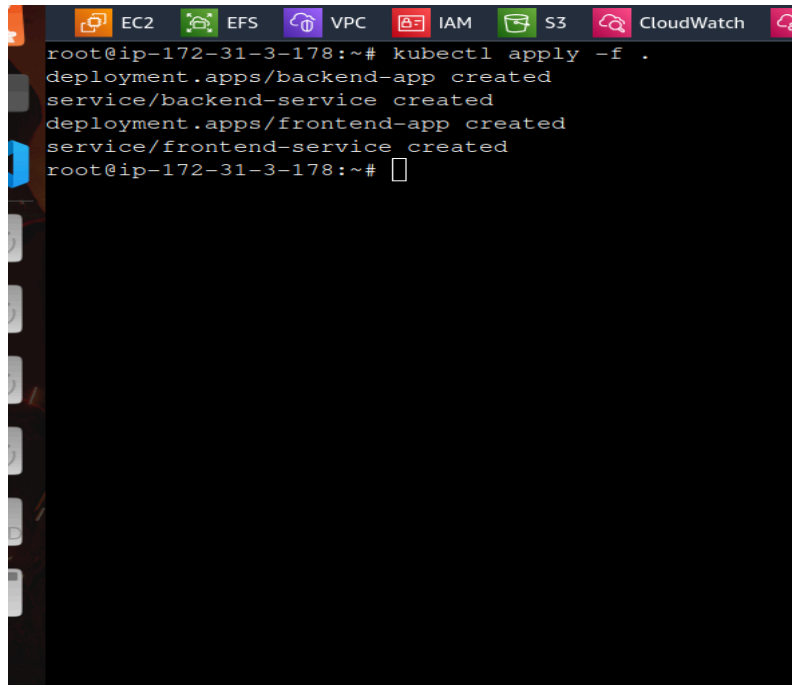
```
root@ip-172-31-3-178:~# nano frontend-dp.yaml
root@ip-172-31-3-178:~# cat frontend-dp.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: frontend-app
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: frontend-app
    spec:
      containers:
      - name: frontend-app
        image: gaurav1251/angularark8:ffe
        ports:
        - name: angular
          containerPort: 30010
          protocol: TCP
root@ip-172-31-3-178:~#
```

```
root@ip-172-31-3-178:~# nano frontend-svc.yaml
root@ip-172-31-3-178:~# cat frontend-svc.yaml
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  selector:
    app: frontend-app
  ports:
  - name: angular
    targetPort: 30010
    port: 30010
    type: NodePort
root@ip-172-31-3-178:~#
```

Here we gave 30010 as port because our frontend image is exposed to port 30010.

Step 7: Now apply the deployment and service files of the both backend and frontend.

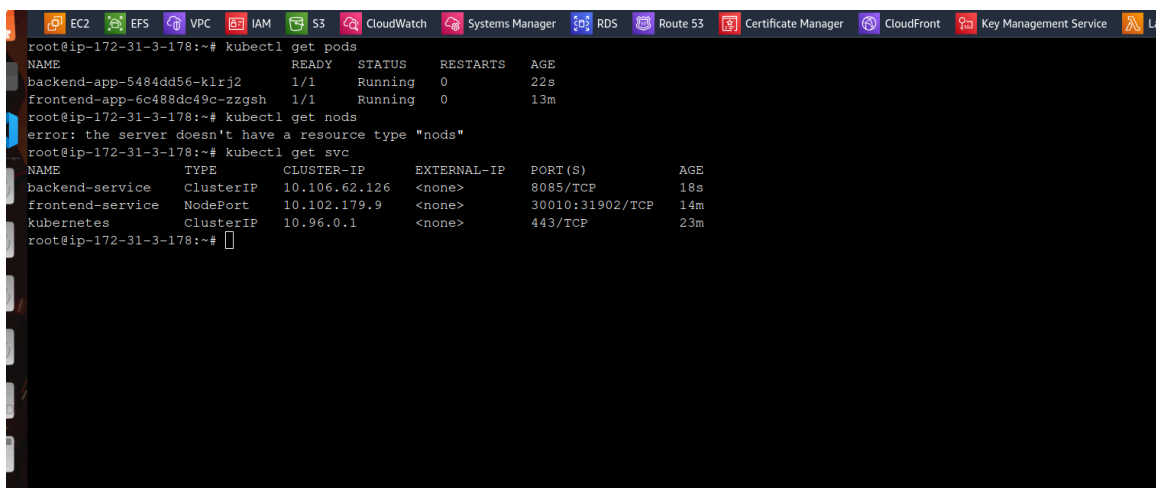
Kubectl apply -f .



```
root@ip-172-31-3-178:~# kubectl apply -f .
deployment.apps/backend-app created
service/backend-service created
deployment.apps/frontend-app created
service/frontend-service created
root@ip-172-31-3-178:~#
```

Check the pods and svc are running successfully.

kubectl get pods && kubectl get svc



```
root@ip-172-31-3-178:~# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
backend-app-5484dd56-klrj2          1/1     Running   0           22s
frontend-app-6c488dc49c-zzgsh       1/1     Running   0           13m
root@ip-172-31-3-178:~# kubectl get nodes
error: the server doesn't have a resource type "nodes"
root@ip-172-31-3-178:~# kubectl get svc
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP  PORT(S)          AGE
backend-service     ClusterIP   10.106.62.126   <none>       8085/TCP         18s
frontend-service    NodePort    10.102.179.9    <none>       30010:31902/TCP  14m
kubernetes           ClusterIP   10.96.0.1       <none>       443/TCP          23m
root@ip-172-31-3-178:~#
```

Step 8: Now make changes in the file worker.service.ts ,add the public ip of worker node and the port which was assigned by NodePort.

```
root@ip-172-31-3-178:~# kubectl get pods
NAME                                READY   STATUS    RESTARTS      AGE
backend-app-5484dd56-klrj2         1/1     Running   3 (3m24s ago)  4m54s
frontend-app-6c488dc49c-zzgsh     1/1     Running   0              18m
root@ip-172-31-3-178:~# kubectl logs backend-app-5484dd56-klrj2
root@ip-172-31-3-178:~# kubectl get svc
NAME                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
backend-service     NodePort    10.96.113.34  <none>         8085:31721/TCP   25s
frontend-service    NodePort    10.102.179.9  <none>         30010:31902/TCP  20m
kubernetes          ClusterIP   10.96.0.1     <none>         443/TCP          29m
root@ip-172-31-3-178:~# nano worker.service.ts
root@ip-172-31-3-178:~# nano worker.service.ts
root@ip-172-31-3-178:~# cat worker.service.ts
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { map } from 'rxjs/operators';
import { Worker } from '../models/worker';

@Injectable({
  providedIn: 'root'
})
export class WorkerService {

  private getUrl: string = "http://43.204.235.147:31721/api/v1/workers";
```

Step 9: Now as the both application.properties and worker.service.ts files are configured as per the essential values copy this files to there respected paths in the pods.

For backend pod

→ **kubectl cp /path/application.properties backend-pod_name:/path/**

```
root@ip-172-31-3-178:~# kubectl cp application.properties backend-app-5484dd56-klrj2:/app/src/main/resources/
```

For Frontend pod

→ `kubectl cp /path/worker.service.ts`
`frontend-pod_name:/path/`

```
root@ip-172-31-3-178:~# kubectl cp application.properties backend-app-5484dd56-klrj2:/app/src/main/resources/
```

Step 10: Now again check the pods are running successfully or not .

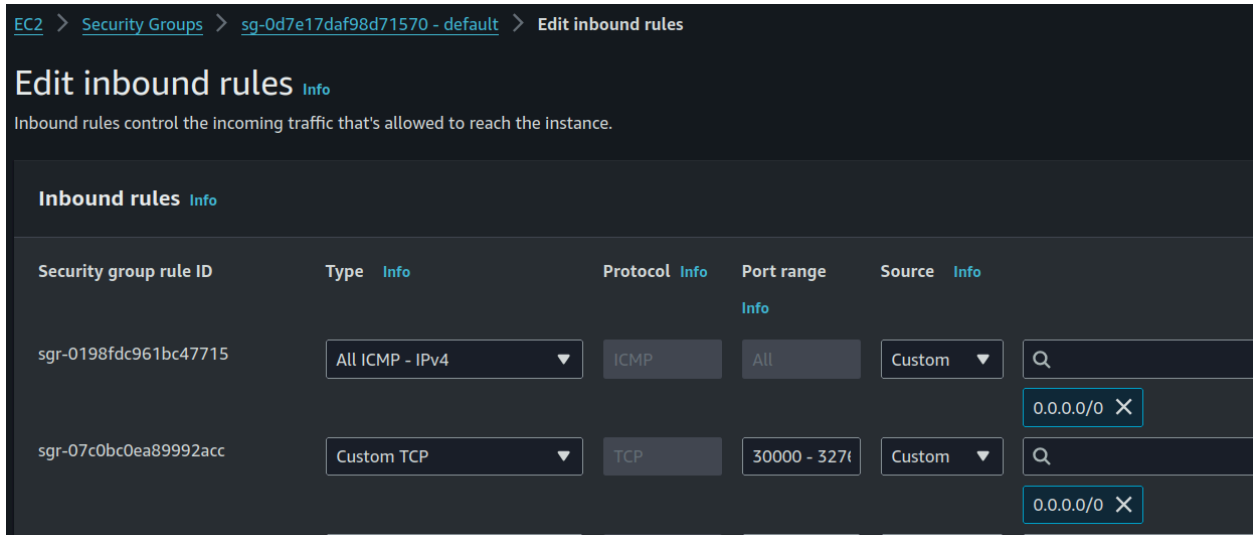
Kubectl get pods

After this check frontend service port number by →

kubectl get svc

```
root@ip-172-31-3-178:~# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
backend-app-5484dd56-klrj2          1/1     Running   3 (8m47s ago)  10m
frontend-app-6c488dc49c-zzgsh       1/1     Running   0           23m
root@ip-172-31-3-178:~# kubectl get svc
NAME                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
backend-service     NodePort    10.96.113.34 <none>        8085:31721/TCP   3m55s
frontend-service    NodePort    10.102.179.9  <none>        30010:31902/TCP  23m
kubernetes          ClusterIP   10.96.0.1     <none>        443/TCP          33m
root@ip-172-31-3-178:~#
```

After this add the port numbers 30000-32767 in the security Group of the worker node, which is the port range is used by Kubernetes for services of type **NodePort. These ports allow you to expose services on a node's IP address at a static port.**



Output : Paste the public ip of worker node and the port assigned by the NodePort to Frontend service in the web browser.

http://public_ip_of_worker-node:port/

