Name: Tushar Panchal

En.No: 21162101014

Sub: CS(Cloud Security)

Branch: CBA

Batch:71

---------------------------PRACTICAL 10----------------------------

You are working for a company that uses IBM Cloud to store critical data in a Cloudant database. Your team has developed an API-based application that performs CRUD operations on the Cloudant database, and this application is now ready for deployment in a Kubernetes environment.

As part of the security team, your task is to ensure that the application adheres to security best practices, including limiting network traffic for the pods.

Task:

- **Deploy the existing API-based application on a Kubernetes cluster.**
- **Configure a network policy that blocks all egress traffic from the pod.**

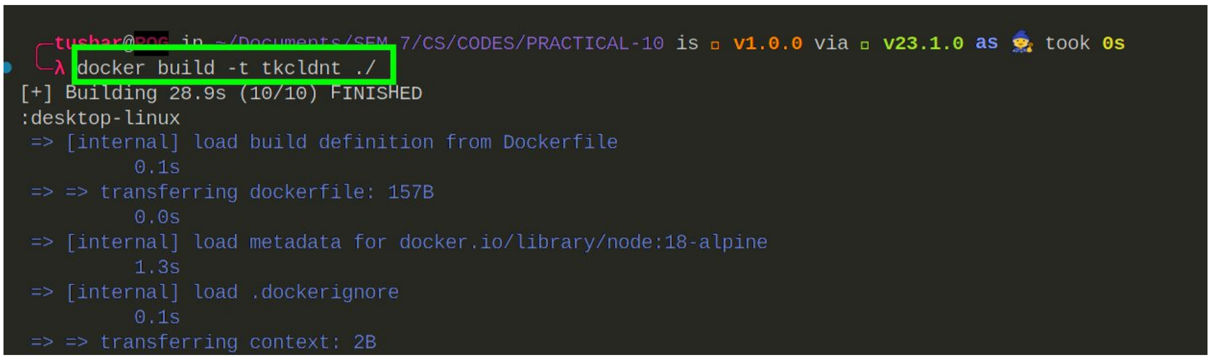
1. Create nodejs application and dockerfile to connect to cloudant database.

App.js:

```
const express = require('express');
const { CloudantV1 } = require('@ibm-cloud/cloudant');
const { IamAuthenticator } = require('ibm-cloud-sdk-core');
const bodyParser = require('body-parser');

let PORT = process.env.PORT || 3000;

const url = 'https://apikey-v2-
26aj3ozdvpr2f1cmuwbxky11mhbiubctlpomomvyprjj:531f952562337579bc0b3ef8cfea1
4bc@ed643cff-2d52-40c5-b33d-71ad7ff9e885-
bluemix.cloudantnosqldb.appdomain.cloud';
const apiKey = 'jQEQoKHvrKYVArJDcUXBl-IWtA90bnJHy0QfhkJuuuC8';

const authenticator = new IamAuthenticator({ apikey: apiKey });
const cloudant = CloudantV1.newInstance({ authenticator });
cloudant.setServiceUrl(url);

const app = express();
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());

app.get('/', function (req, res) {
  res.send("Welcome to cloudant database on IBM Cloud");
});

app.get('/list_of_databases', async function (req, res) {
  try {
    const response = await cloudant.getAllDbs();
    res.send(response.result);
  } catch (err) {
    res.send(err);
  }
});

app.post('/create-database', async (req, res) => {
  const name = req.body.name;
  try {
    await cloudant.putDatabase({ db: name });
    res.send("Database created");
  } catch (err) {
    res.send(err);
  }
});

app.post('/insert-document', async function (req, res) {
  const { db, id, name, address, phone, age } = req.body;
  try {
    const response = await cloudant.postDocument({
      db,
      document: { _id: id, name, address, phone, age }
    });
    res.send(response.result);
  } catch (err) {
    res.send(err);
  }
});

app.post('/insert-bulk/:database_name', async function (req, res) {
  const database_name = req.params.database_name;
  const students = req.body.docs.map(doc => ({
    _id: doc.id,
    name: doc.name,
    address: doc.address,
    phone: doc.phone,
    age: doc.age
  }));

  try {
    await cloudant.postBulkDocs({
      db: database_name,
      bulkDocs: { docs: students }
    });
    res.send('Inserted all documents');
  } catch (err) {
    res.send(err);
  }
});

app.delete('/delete-document', async function (req, res) {
  const { db, id, rev } = req.body;
  try {
    await cloudant.deleteDocument({ db, docId: id, rev });
    res.send('Document deleted');
  } catch (err) {
    res.send(err);
  }
});

app.put('/update-document', async function (req, res) {
  const { db, id, rev, name, address, phone, age } = req.body;
  try {
    const response = await cloudant.postDocument({
      db,
      document: { _id: id, _rev: rev, name, address, phone, age }
    });
    res.send(response.result);
  } catch (err) {
    res.send(err);
  }
});

app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

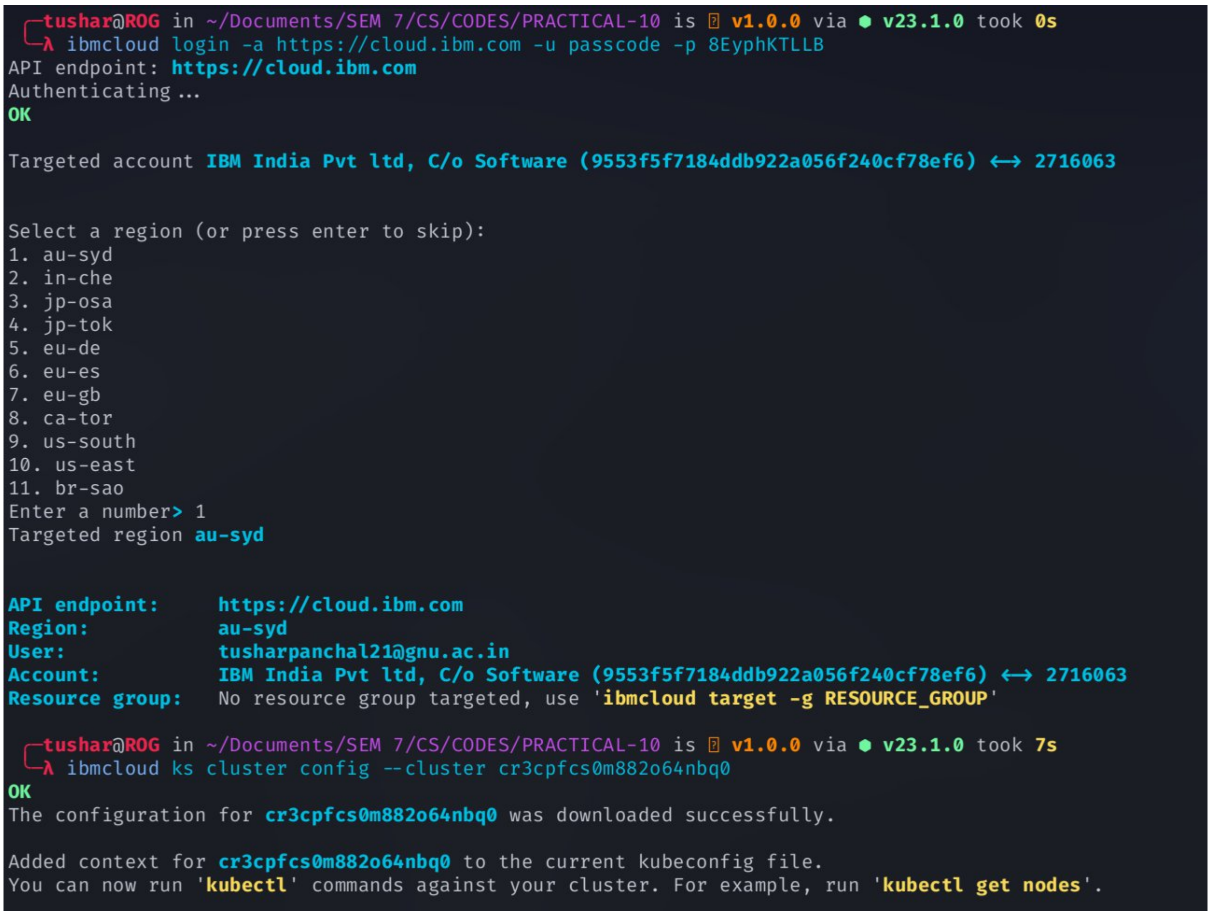
Dockerfile:

```
FROM node:18-alpine
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY index.js .
EXPOSE 3000
CMD ["node", "app.js"]
```

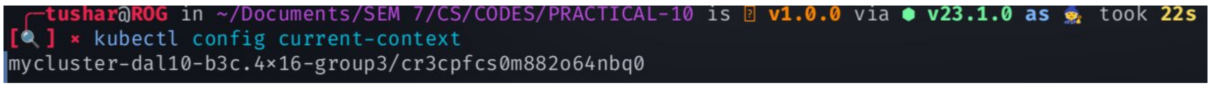


```
╭─tushar@[redacted] in ~/Documents/SEM 7/CS/CODES/PRACTICAL-10 is ◻ v1.0.0 via ◻ v23.1.0 as 🧙 took 0s
╰─λ docker build -t tkcldnt ./
[+] Building 28.9s (10/10) FINISHED
:desktop-linux
 => [internal] load build definition from Dockerfile
          0.1s
 => => transferring dockerfile: 157B
          0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine
          1.3s
 => [internal] load .dockerignore
          0.1s
 => => transferring context: 2B
```

2. Login to ibmcloud cli and push docker image to ibm container registry and than config context

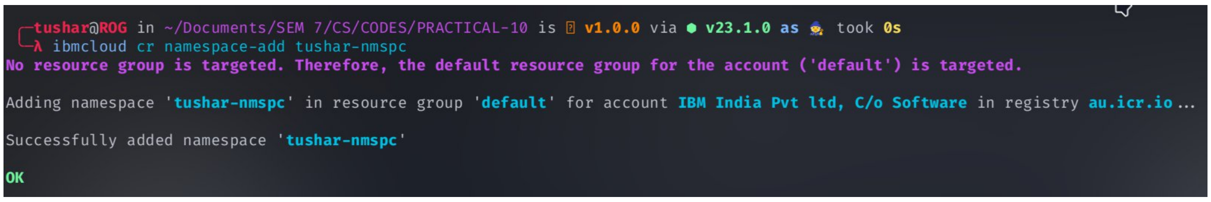


```
╭─tushar@ROG in ~/Documents/SEM 7/CS/CODES/PRACTICAL-10 is ▯ v1.0.0 via ● v23.1.0 took 0s
╰─λ ibmcloud login -a https://cloud.ibm.com -u passcode -p 8EyphKTLLB
API endpoint: https://cloud.ibm.com
Authenticating ...
OK

Targeted account IBM India Pvt ltd, C/o Software (9553f5f7184ddb922a056f240cf78ef6) ⟷ 2716063


Select a region (or press enter to skip):
1. au-syd
2. in-che
3. jp-osa
4. jp-tok
5. eu-de
6. eu-es
7. eu-gb
8. ca-tor
9. us-south
10. us-east
11. br-sao
Enter a number> 1
Targeted region au-syd


API endpoint:     https://cloud.ibm.com
Region:           au-syd
User:             tusharpanchal21@gnu.ac.in
Account:          IBM India Pvt ltd, C/o Software (9553f5f7184ddb922a056f240cf78ef6) ⟷ 2716063
Resource group:   No resource group targeted, use 'ibmcloud target -g RESOURCE_GROUP'

╭─tushar@ROG in ~/Documents/SEM 7/CS/CODES/PRACTICAL-10 is ▯ v1.0.0 via ● v23.1.0 took 7s
╰─λ ibmcloud ks cluster config --cluster cr3cpfcs0m882o64nbq0
OK
The configuration for cr3cpfcs0m882o64nbq0 was downloaded successfully.

Added context for cr3cpfcs0m882o64nbq0 to the current kubeconfig file.
You can now run 'kubectl' commands against your cluster. For example, run 'kubectl get nodes'.
```



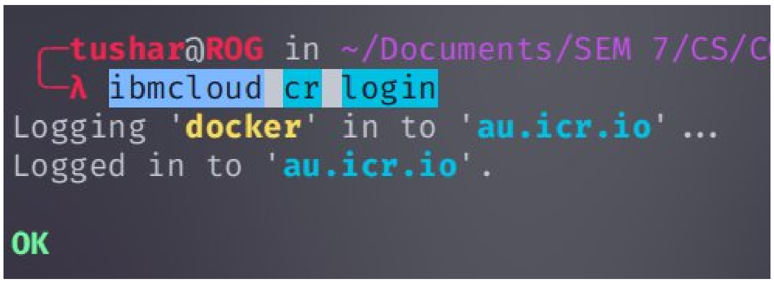
```
 tushar@ROG in ~/Documents/SEM 7/CS/CODES/PRACTICAL-10 is ▯ v1.0.0 via ● v23.1.0 as 🧙 took 22s
[🔍] × kubectl config current-context
mycluster-dal10-b3c.4x16-group3/cr3cpfcs0m882o64nbq0
```

Now add namespace of yours



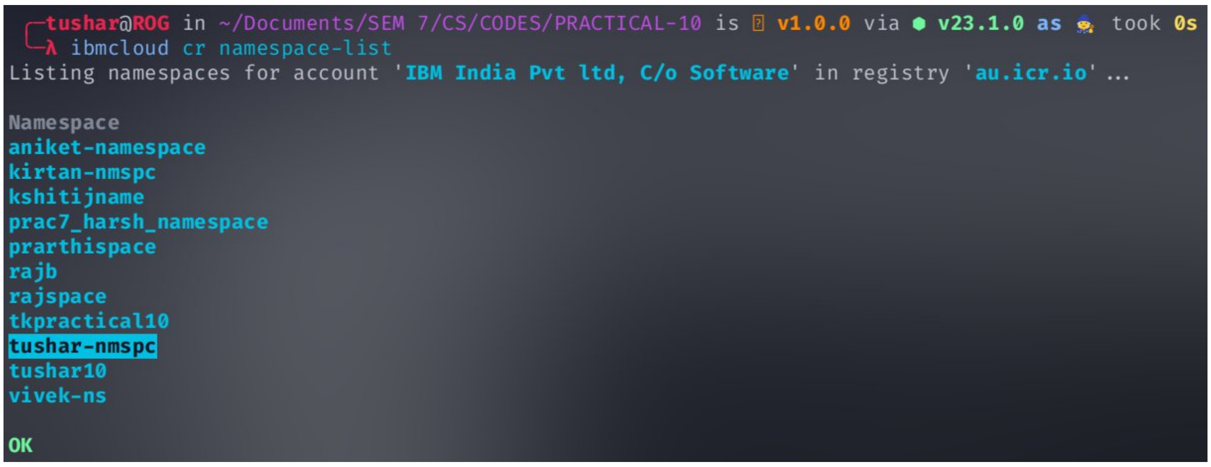
```
╭─tushar@ROG in ~/Documents/SEM 7/CS/CODES/PRACTICAL-10 is ▯ v1.0.0 via ● v23.1.0 as 🧙 took 0s
╰─λ ibmcloud cr namespace-add tushar-nmspc
No resource group is targeted. Therefore, the default resource group for the account ('default') is targeted.

Adding namespace 'tushar-nmspc' in resource group 'default' for account IBM India Pvt ltd, C/o Software in registry au.icr.io...

Successfully added namespace 'tushar-nmspc'

OK
```

Now login into Docker using ibmcloud cr login



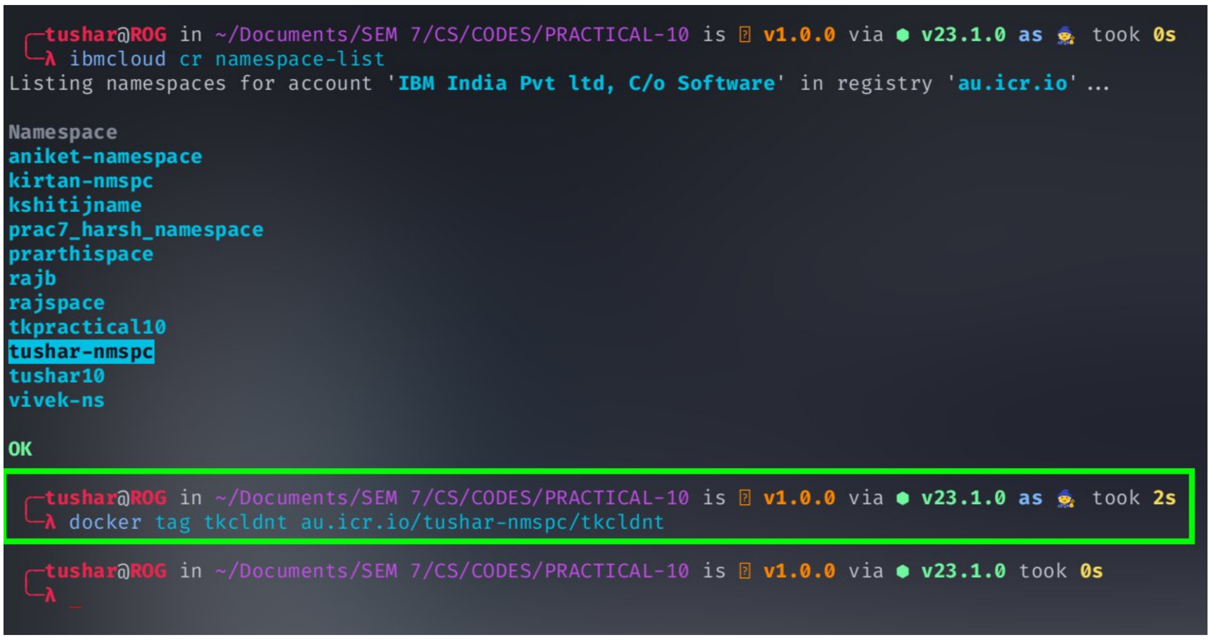
```
╭─tushar@ROG in ~/Documents/SEM 7/CS/C
╰─λ ibmcloud cr login
Logging 'docker' in to 'au.icr.io'...
Logged in to 'au.icr.io'.

OK
```

Now check namespce list available using below command



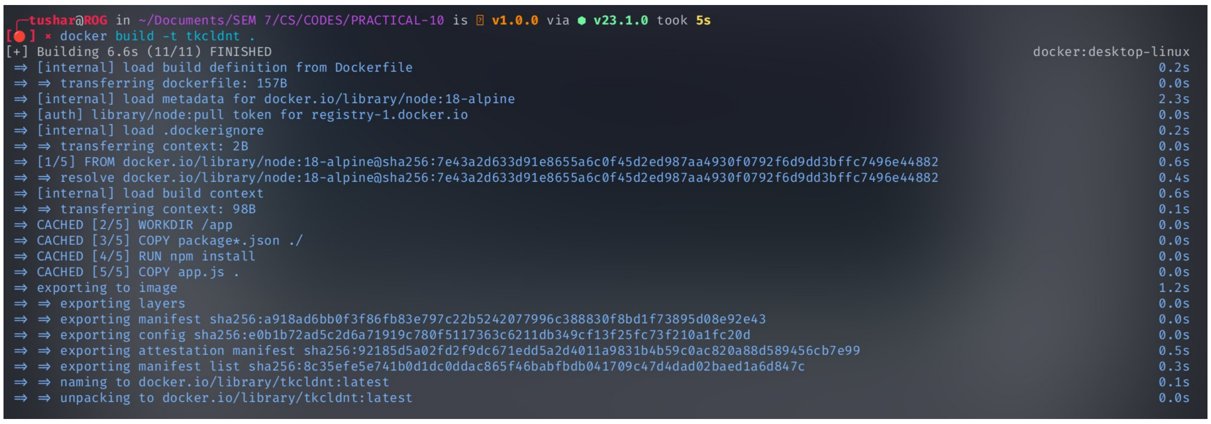
```
┌─tushar@ROG in ~/Documents/SEM 7/CS/CODES/PRACTICAL-10 is  v1.0.0 via ● v23.1.0 as 🧙 took 0s
└─λ ibmcloud cr namespace-list
Listing namespaces for account 'IBM India Pvt ltd, C/o Software' in registry 'au.icr.io'...

Namespace
aniket-namespace
kirtan-nmspc
kshitijname
prac7_harsh_namespace
prarthispace
rajb
rajspace
tkpractical10
tushar-nmspc
tushar10
vivek-ns

OK
```

Tag the image with your namespace



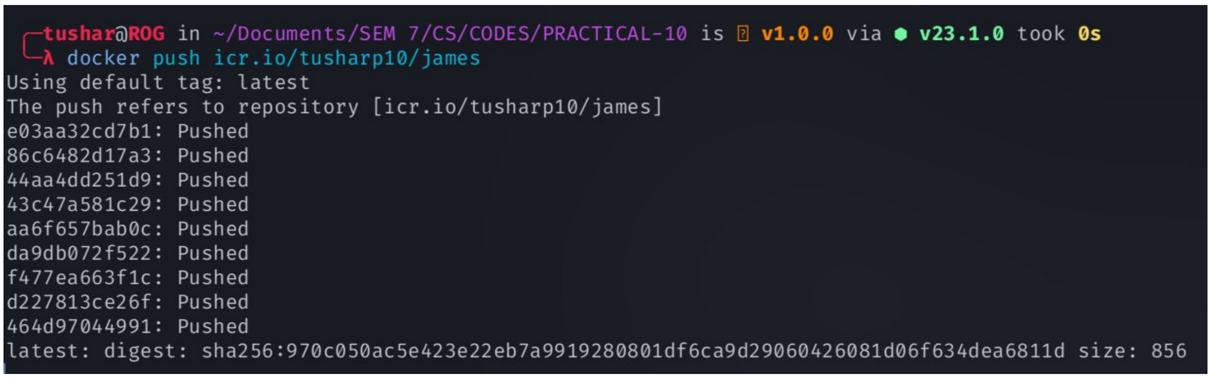
```
┌─tushar@ROG in ~/Documents/SEM 7/CS/CODES/PRACTICAL-10 is  v1.0.0 via ● v23.1.0 as 🧙 took 0s
└─λ ibmcloud cr namespace-list
Listing namespaces for account 'IBM India Pvt ltd, C/o Software' in registry 'au.icr.io'...

Namespace
aniket-namespace
kirtan-nmspc
kshitijname
prac7_harsh_namespace
prarthispace
rajb
rajspace
tkpractical10
tushar-nmspc
tushar10
vivek-ns

OK

┌─tushar@ROG in ~/Documents/SEM 7/CS/CODES/PRACTICAL-10 is  v1.0.0 via ● v23.1.0 as 🧙 took 2s
└─λ docker tag tkcldnt au.icr.io/tushar-nmspc/tkcldnt

┌─tushar@ROG in ~/Documents/SEM 7/CS/CODES/PRACTICAL-10 is  v1.0.0 via ● v23.1.0 took 0s
└─λ _
```

Then build the image again



```
┌─tushar@ROG in ~/Documents/SEM 7/CS/CODES/PRACTICAL-10 is  v1.0.0 via ● v23.1.0 took 5s
[●] × docker build -t tkcldnt .
[+] Building 6.6s (11/11) FINISHED                                                                          docker:desktop-linux
 ⇒ [internal] load build definition from Dockerfile                                                                        0.2s
 ⇒ ⇒ transferring dockerfile: 157B                                                                                        0.0s
 ⇒ [internal] load metadata for docker.io/library/node:18-alpine                                                          2.3s
 ⇒ [auth] library/node:pull token for registry-1.docker.io                                                                0.0s
 ⇒ [internal] load .dockerignore                                                                                          0.2s
 ⇒ ⇒ transferring context: 2B                                                                                             0.0s
 ⇒ [1/5] FROM docker.io/library/node:18-alpine@sha256:7e43a2d633d91e8655a6c0f45d2ed987aa4930f0792f6d9dd3bffc7496e44882     0.6s
 ⇒ ⇒ resolve docker.io/library/node:18-alpine@sha256:7e43a2d633d91e8655a6c0f45d2ed987aa4930f0792f6d9dd3bffc7496e44882      0.4s
 ⇒ [internal] load build context                                                                                          0.6s
 ⇒ ⇒ transferring context: 98B                                                                                            0.1s
 ⇒ CACHED [2/5] WORKDIR /app                                                                                              0.0s
 ⇒ CACHED [3/5] COPY package*.json ./                                                                                     0.0s
 ⇒ CACHED [4/5] RUN npm install                                                                                           0.0s
 ⇒ CACHED [5/5] COPY app.js .                                                                                             0.0s
 ⇒ exporting to image                                                                                                     1.2s
 ⇒ ⇒ exporting layers                                                                                                     0.0s
 ⇒ ⇒ exporting manifest sha256:a918ad6bb0f3f86fb83e797c22b5242077996c388830f8bd1f73895d08e92e43                           0.0s
 ⇒ ⇒ exporting config sha256:e0b1b72ad5c2d6a71919c780f5117363c6211db349cf13f25fc73f210a1fc20d                             0.0s
 ⇒ ⇒ exporting attestation manifest sha256:92185d5a02fd2f9dc671edd5a2d4011a9831b4b59c0ac820a88d589456cb7e99               0.5s
 ⇒ ⇒ exporting manifest list sha256:8c35efe5e741b0d1dc0ddac865f46babfbdb041709c47d4dad02baed1a6d847c                      0.3s
 ⇒ ⇒ naming to docker.io/library/tkcldnt:latest                                                                           0.1s
 ⇒ ⇒ unpacking to docker.io/library/tkcldnt:latest                                                                        0.0s
```

Then push the image



```
┌─tushar@ROG in ~/Documents/SEM 7/CS/CODES/PRACTICAL-10 is 📦 v1.0.0 via ● v23.1.0 took 0s
└─λ docker push icr.io/tusharp10/james
Using default tag: latest
The push refers to repository [icr.io/tusharp10/james]
e03aa32cd7b1: Pushed
86c6482d17a3: Pushed
44aa4dd251d9: Pushed
43c47a581c29: Pushed
aa6f657bab0c: Pushed
da9db072f522: Pushed
f477ea663f1c: Pushed
d227813ce26f: Pushed
464d97044991: Pushed
latest: digest: sha256:970c050ac5e423e22eb7a9919280801df6ca9d29060426081d06f634dea6811d size: 856
```

Then create the deployment.yaml and service.yaml

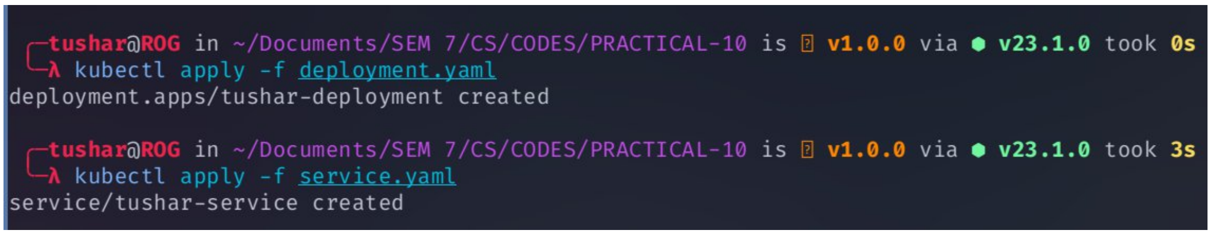
Deployment.yaml:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tushar-deployment
  labels:
    app: james-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: james-app
  template:
    metadata:
      labels:
        app: james-app
    spec:
      containers:
        - name: james-container
          image: icr.io/tusharp10/james:latest  # Updated to your image
          ports:
            - containerPort: 3000
          resources:
            requests:
              memory: "256Mi"
              cpu: "250m"
            limits:
              memory: "512Mi"
              cpu: "500m"
```

Service.yaml:

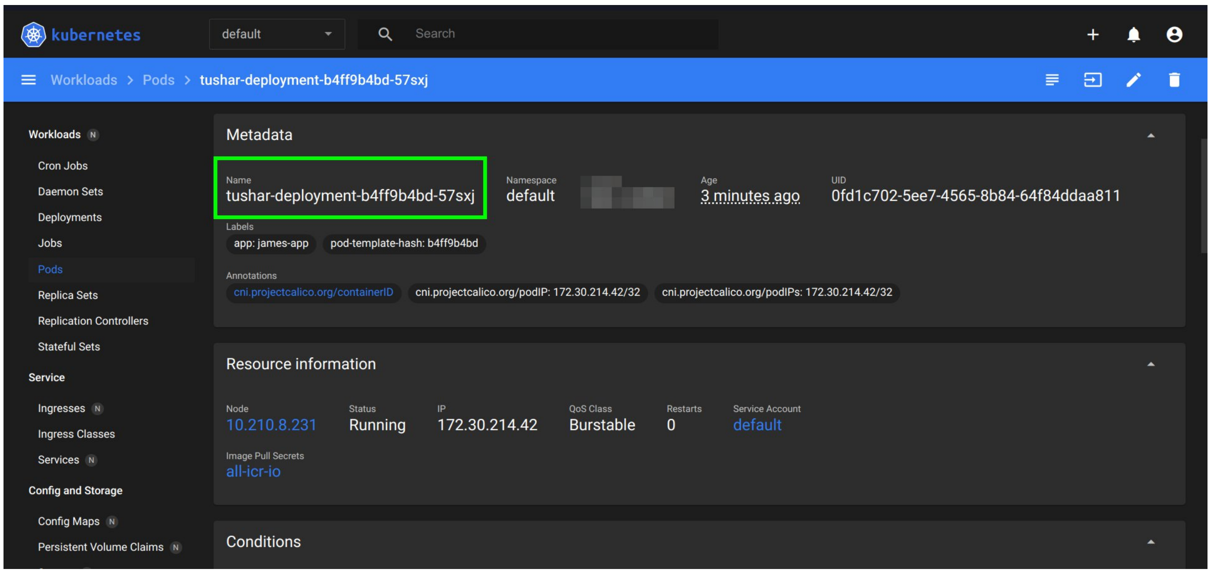
```
apiVersion: v1
kind: Service
metadata:
  name: tushar-service
  namespace: default
  labels:
    app: james-app
spec:
  type: NodePort
  ports:
    - name: http
      protocol: TCP
      port: 3000         # External port for the service
      targetPort: 3000   # Port exposed by the container
  selector:
    app: james-app
```

Now apply this both yaml files with below command



```
┌─tushar@ROG in ~/Documents/SEM 7/CS/CODES/PRACTICAL-10 is 📦 v1.0.0 via ⬢ v23.1.0 took 0s
└─λ kubectl apply -f deployment.yaml
deployment.apps/tushar-deployment created

┌─tushar@ROG in ~/Documents/SEM 7/CS/CODES/PRACTICAL-10 is 📦 v1.0.0 via ⬢ v23.1.0 took 3s
└─λ kubectl apply -f service.yaml
service/tushar-service created
```

As you see in kubernetes cluster dashborad that deplyod



kubernetes default Search

Workloads > Pods > tushar-deployment-b4ff9b4bd-57sxj

Workloads N
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Service
Ingresses N
Ingress Classes
Services N
Config and Storage
Config Maps N
Persistent Volume Claims N

Metadata

Name	Namespace		Age	UID
tushar-deployment-b4ff9b4bd-57sxj	default	[redacted]	3 minutes ago	0fd1c702-5ee7-4565-8b84-64f84ddaa811

Labels
app: james-app
pod-template-hash: b4ff9b4bd

Annotations
cni.projectcalico.org/containerID
cni.projectcalico.org/podIP: 172.30.214.42/32
cni.projectcalico.org/podIPs: 172.30.214.42/32

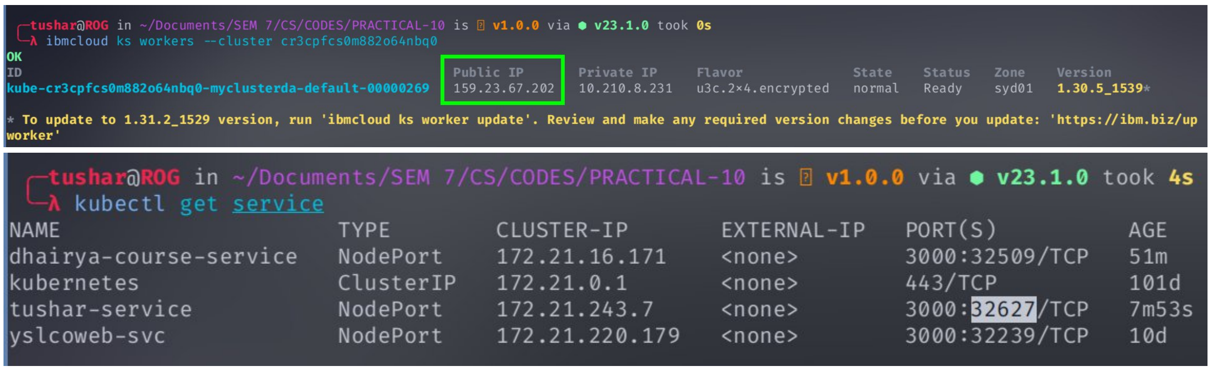
Resource information

Node	Status	IP	QoS Class	Restarts	Service Account
10.210.8.231	Running	172.30.214.42	Burstable	0	default

Image Pull Secrets
all-icr-io

Conditions

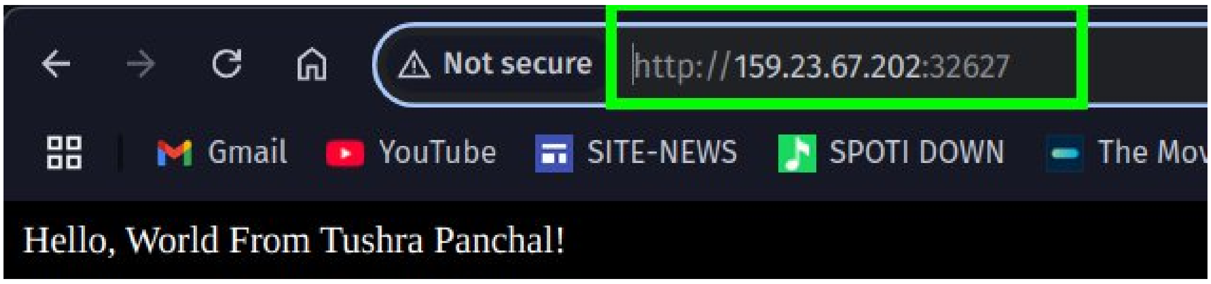
Now we will gather ip and port to ensure it is working..



```
╭─tushar@ROG in ~/Documents/SEM 7/CS/CODES/PRACTICAL-10 is 📦 v1.0.0 via ● v23.1.0 took 0s
╰─λ ibmcloud ks workers --cluster cr3cpfcs0m882o64nbq0
OK
ID                                                       Public IP       Private IP    Flavor               State    Status   Zone    Version
kube-cr3cpfcs0m882o64nbq0-myclusterda-default-00000269   159.23.67.202   10.210.8.231  u3c.2x4.encrypted    normal   Ready    syd01   1.30.5_1539*

* To update to 1.31.2_1529 version, run 'ibmcloud ks worker update'. Review and make any required version changes before you update: 'https://ibm.biz/up
worker'
```

```
╭─tushar@ROG in ~/Documents/SEM 7/CS/CODES/PRACTICAL-10 is 📦 v1.0.0 via ● v23.1.0 took 4s
╰─λ kubectl get service
NAME                    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
dhairya-course-service  NodePort    172.21.16.171   <none>        3000:32509/TCP   51m
kubernetes              ClusterIP   172.21.0.1      <none>        443/TCP          101d
tushar-service          NodePort    172.21.243.7    <none>        3000:32627/TCP   7m53s
yslcoweb-svc            NodePort    172.21.220.179  <none>        3000:32239/TCP   10d
```

As you can see below we can succefully access

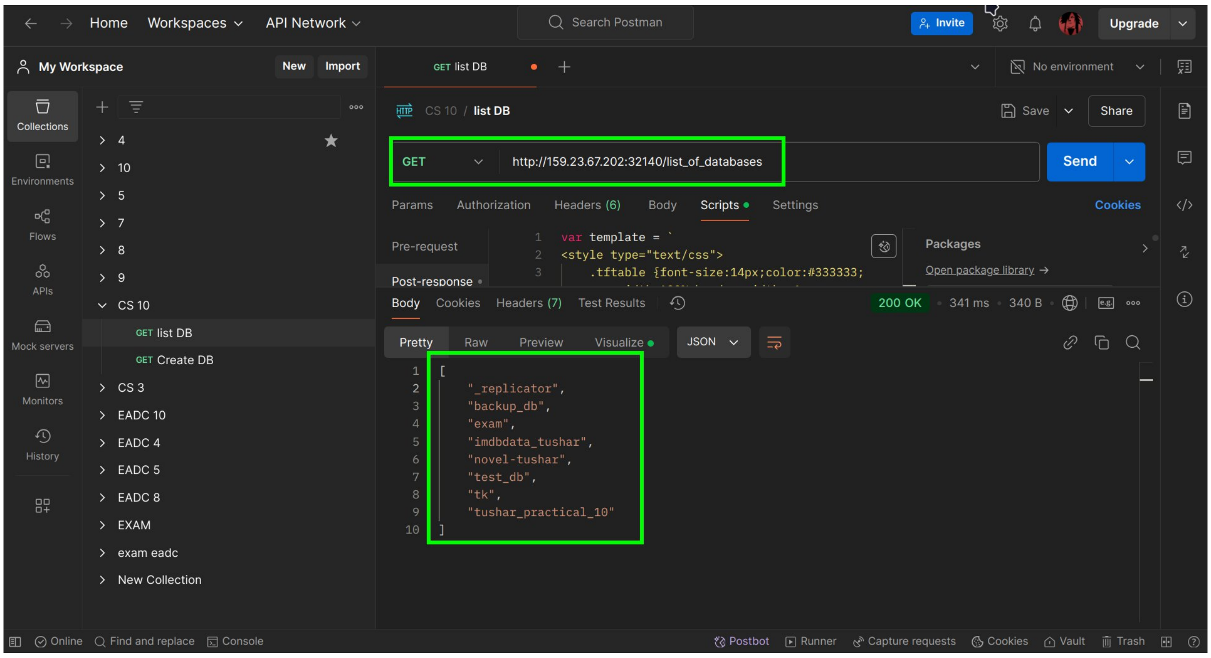


Not secure http://159.23.67.202:32627

Hello, World From Tushra Panchal!

Now test the postman request to list and create database:

List Databases:

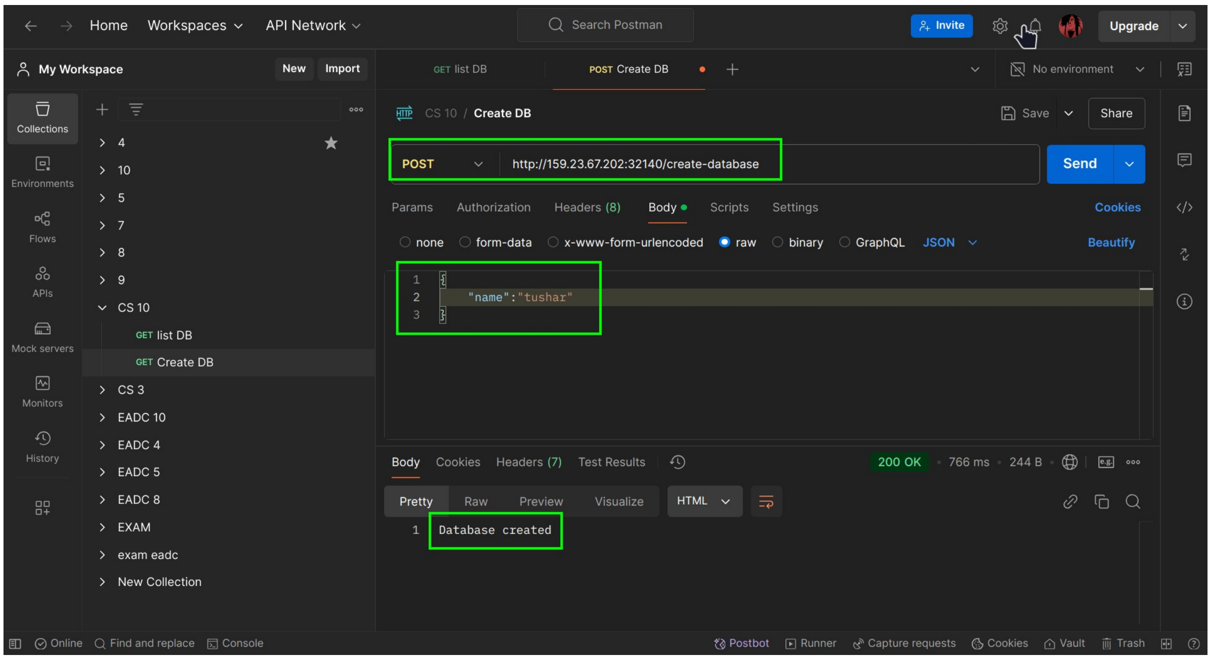


GET http://159.23.67.202:32140/list_of_databases

```
[
    "_replicator",
    "backup_db",
    "exam",
    "imdbdata_tushar",
    "novel-tushar",
    "test_db",
    "tk",
    "tushar_practical_10"
]
```

200 OK 341 ms 340 B

Create Databse:



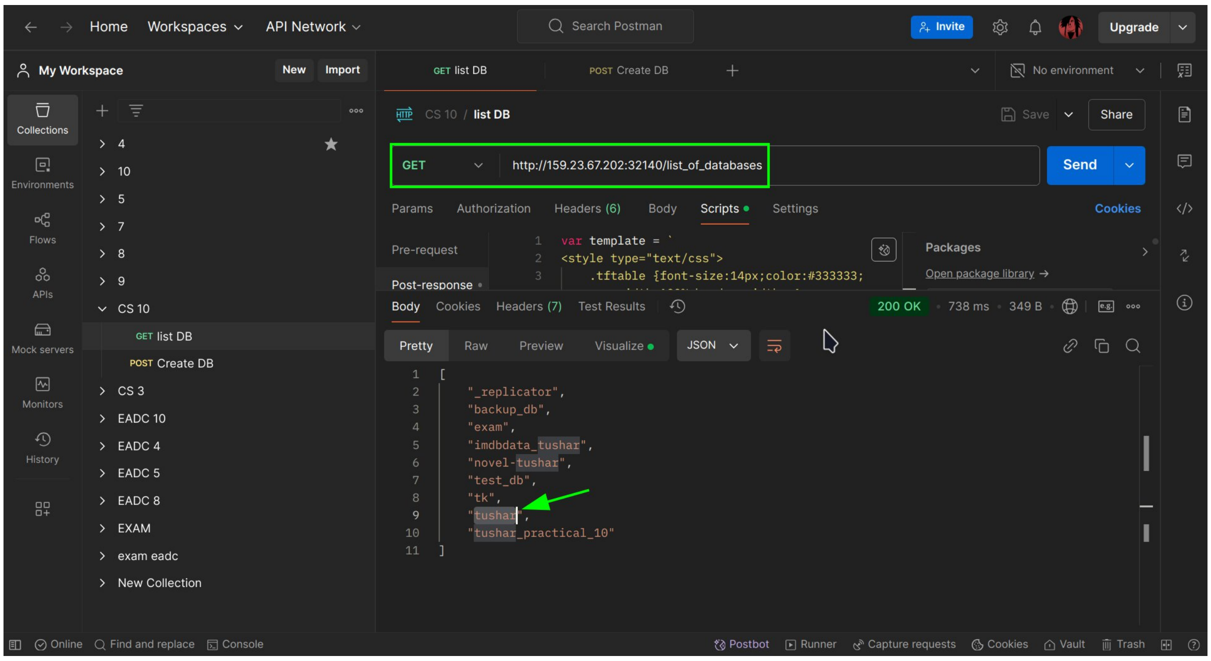
POST http://159.23.67.202:32140/create-database

```
{
    "name":"tushar"
}
```

200 OK 766 ms 244 B

```
Database created
```

As you can see below DB created successfully



CS 10 / list DB

GET http://159.23.67.202:32140/list_of_databases

```
var template = `
<style type="text/css">
    .tftable {font-size:14px;color:#333333;
```

200 OK • 738 ms • 349 B

```
[
    "_replicator",
    "backup_db",
    "exam",
    "imdbdata_tushar",
    "novel-tushar",
    "test_db",
    "tk",
    "tushar",
    "tushar_practical_10"
]
```