

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA      Batch - 71

CS Practical 10

**Scenario :** 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.

Steps and Screenshots :

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 71  
CS Practical 10

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

Code :

```
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://60379ad1-b8cb-4933-bb03-b1bfa6df8a7e-bluemix.cloudantnosqldb.appdomain.cloud';
const apiKey = 'PTf_8CBfnhbiTsz59lJNaLpE93g7kFmxYXd09Ly4C6DP';

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");
});
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 71  
CS Practical 10

});

```
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 {
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 71  
CS Practical 10

```
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,
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 71  
CS Practical 10

```
    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 }
    });
  }
});
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 71  
CS Practical 10

```
});  
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", "index.js"]
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 71  
CS Practical 10

The screenshot shows a terminal window with a red box highlighting the Dockerfile content and another red box highlighting the build command and its output.

**Dockerfile Content:**

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

**Build Logs:**

```
ysl ~/Practical_10 ⌘ main !? @ v22.9.0 07:48 docker build -t yslcldnt:1.0 ./
[+] Building 29.0s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 168B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 110.53kB
=> [1/5] FROM docker.io/library/node:18-alpine@sha256:02376a266c84acbf45bd19440e08e48b1c8b98037417334046029ab585de03e2
=> CACHED [2/5] WORKDIR /app
=> [3/5] COPY package*.json .
=> [4/5] RUN npm install
=> [5/5] COPY index.js .
=> exporting to image
=> => exporting layers
=> => writing image sha256:e3b94dc33e761ffdc87f25497ba260124e6b4d60519358550d64d9def6b8dd
```

At the bottom of the terminal window, there is a status bar with various icons and text: Ln 11, Col 12, Spaces: 4, UTF-8, LF, Dockerfile, Go Live, Prettier.

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA      Batch - 71

CS Practical 10

## 2. Login to ibmcloud cli and push docker image to ibm container registry

The screenshot shows a terminal window within a code editor interface. The terminal is running on a Mac OS X system, as indicated by the dock icons at the bottom.

The terminal command history is as follows:

```
ysl ~/Practical_10 ⌘ main !? @ v22.9.0 08:38 docker tag yslclndnt:1.0 br.icr.io/yashlani-nmspc/yslclndnt:1.0
ysl ~/Practical_10 ⌘ main !? @ v22.9.0 08:39 docker push br.icr.io/yashlani-nmspc/yslclndnt:1.0
```

Output from the push command:

```
The push refers to repository [br.icr.io/yashlani-nmspc/yslclndnt]
941e6aae84d: Pushed
dee349e73253: Pushed
a264d39b4d6a: Pushed
22592a090fb1: Pushed
e2be10e97665: Pushed
06fd85419b65: Pushed
f58c462fa079: Pushed
63ca1fbdb43ae: Pushed
1.0: digest: sha256:8095c867df7cae5d0bebdf3ec597dcba65a76bde3acf9729dd6e8af9b242080e size: 1992
```

At the bottom of the terminal window, there are status indicators for battery level (4%), signal strength, and disk usage (51%). The status bar also shows the current file path (Q\_PRACTICAL\_10), file type (YAML), and line/character counts (Ln 15, Col 18).

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 71  
CS Practical 10

### 3. Create deployment and service files

Deployment:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: yashlanu-deployment-pten
spec:
  selector:
    matchLabels:
      app: yashlanu-pten
  replicas: 1
  template:
    metadata:
      labels:
        app: yashlanu-pten
  spec:
    containers:
      - name: nodecontainer
        image: br.icr.io/yashlani-nmspc/yslcldnt:1.0
    ports:
      - containerPort: 3000
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 71  
CS Practical 10

Service :

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: yashlanu-pten
  name: yashlani-service-pten
  namespace: default
spec:
  type: NodePort
  ports:
    - name: http
      protocol: TCP
      port: 3000
  selector:
    app: yashlanu-pten
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 71  
CS Practical 10

The screenshot shows the VS Code interface with the 'depl.yaml' file open in the editor. The code defines a Deployment named 'yashlanu-deployment-pten' with a single replica. It uses a selector with the label 'app: yashlanu-pten'. The container is named 'nodecontainer' and runs the image 'br.icr.io/yashlani-nmspc/yslcldnt:1.0' on port 3000.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: yashlanu-deployment-pten
spec:
  selector:
    matchLabels:
      app: yashlanu-pten
  replicas: 1
  template:
    metadata:
      labels:
        app: yashlanu-pten
    spec:
      containers:
        - name: nodecontainer
          image: br.icr.io/yashlani-nmspc/yslcldnt:1.0
          ports:
            - containerPort: 3000
```

In the terminal, the command `kubectl apply -f depl.yaml` is run, and the output shows the deployment created.

```
ysl ~/Practical_10 $ main !? @ v22.9.0 08:42
deployment.apps/yashlanu-deployment-pten created
```

The screenshot shows the VS Code interface with the 'srvc.yaml' file open in the editor. The code defines a Service named 'yashlanu-service-pten' with a NodePort type. It maps port 3000 to the application's port 3000. The selector matches the 'app: yashlanu-pten' label.

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: yashlanu-pten
  name: yashlanu-service-pten
  namespace: default
spec:
  type: NodePort
  ports:
    - name: http
      protocol: TCP
      port: 3000
      selector:
        app: yashlanu-pten
```

In the terminal, the command `kubectl apply -f srvc.yaml` is run, and the output shows the service created.

```
ysl ~/Practical_10 $ main !? @ v22.9.0 08:50
deployment.apps/yashlanu-deployment-pten created
service/yashlanu-service-pten created
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 71  
CS Practical 10

The screenshot shows a web browser window displaying the Kubernetes UI for a pod named 'yashlanu-deployment-pten-7fc54bddfc-nwwql'. The URL is <https://au-syd.containers.cloud.ibm.com/kubeproxy/clusters/cr3cp/cs0flnvbe4nbpg/service/#/pod/default/yashlanu-deployment-pten-7fc54bddfc-nwwql>. The browser's status bar indicates the battery level is 4%, the temperature is 62°C, and the time is 08:43 AM.

**Metadata**

Name	Namespace	Created	Age	UID
yashlanu-deployment-pten-7fc54bddfc-nwwql	default	Oct 14, 2024	a minute ago	8f03f89e-b85b-47a0-8c9d-5b03a47a415a

**Resource information**

Node	Status	IP	QoS Class	Restarts	Service Account
10.210.8.213	Running	172.30.10.151	BestEffort	0	default

**Conditions**

Type	Status	Last probe time	Last transition time	Reason	Message
PodReadyToStartContainers	True	-	a minute ago	-	-
Initialized	True	-	a minute ago	-	-
Ready	True	-	a minute ago	-	-

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA      Batch - 71

CS Practical 10

#### 4. Get IP and port and check connectivity

The screenshot shows a terminal window within a code editor interface. The terminal displays the following commands and their output:

```
./Practical_10 $ main !? @ v22.9.0 08:53 ibmcloud ks workers --cluster cr3cp7cs0flnvbe4nbpq
OK
ID          Public IP      Private IP    Flavor     State   Status  Zone  Version
kube-cr3cp7cs0flnvbe4nbpq-myclustersy-default-00000180 159.23.67.194  10.210.8.213 u3c.2x4.encrypted normal  Ready   syd01  1.30.3_1533*
```

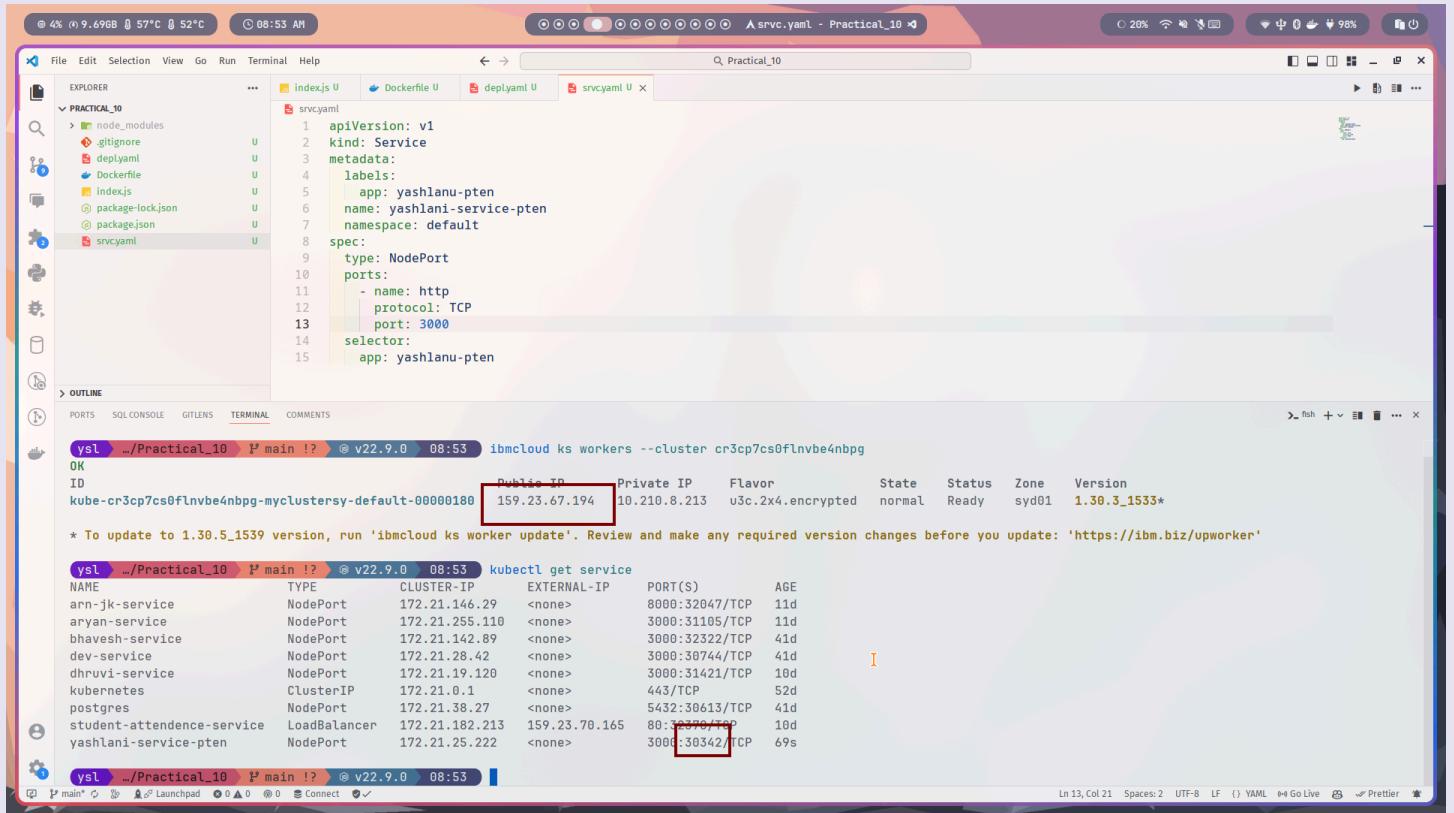
A red box highlights the Public IP address, 159.23.67.194.

\* To update to 1.30.5.1539 version, run 'ibmcloud ks worker update'. Review and make any required version changes before you update: '<https://ibm.biz/upworker>'

```
./Practical_10 $ main !? @ v22.9.0 08:53
```

At the bottom of the terminal, there is a status bar with various icons and text: Ln 13, Col 21 Spaces: 2 UTF-8 LF {} YAML Go Live Prettier.

**Name - Yash Lakhtariya**  
**Enrollment number - 21162101012**  
**Branch - CBA      Batch - 71**  
**CS Practical 10**



```

apiVersion: v1
kind: Service
metadata:
  labels:
    app: yashlanu-pten
    name: yashlanu-service-pten
    namespace: default
spec:
  type: NodePort
  ports:
    - name: http
      protocol: TCP
      port: 3000
      selector:
        app: yashlanu-pten

```

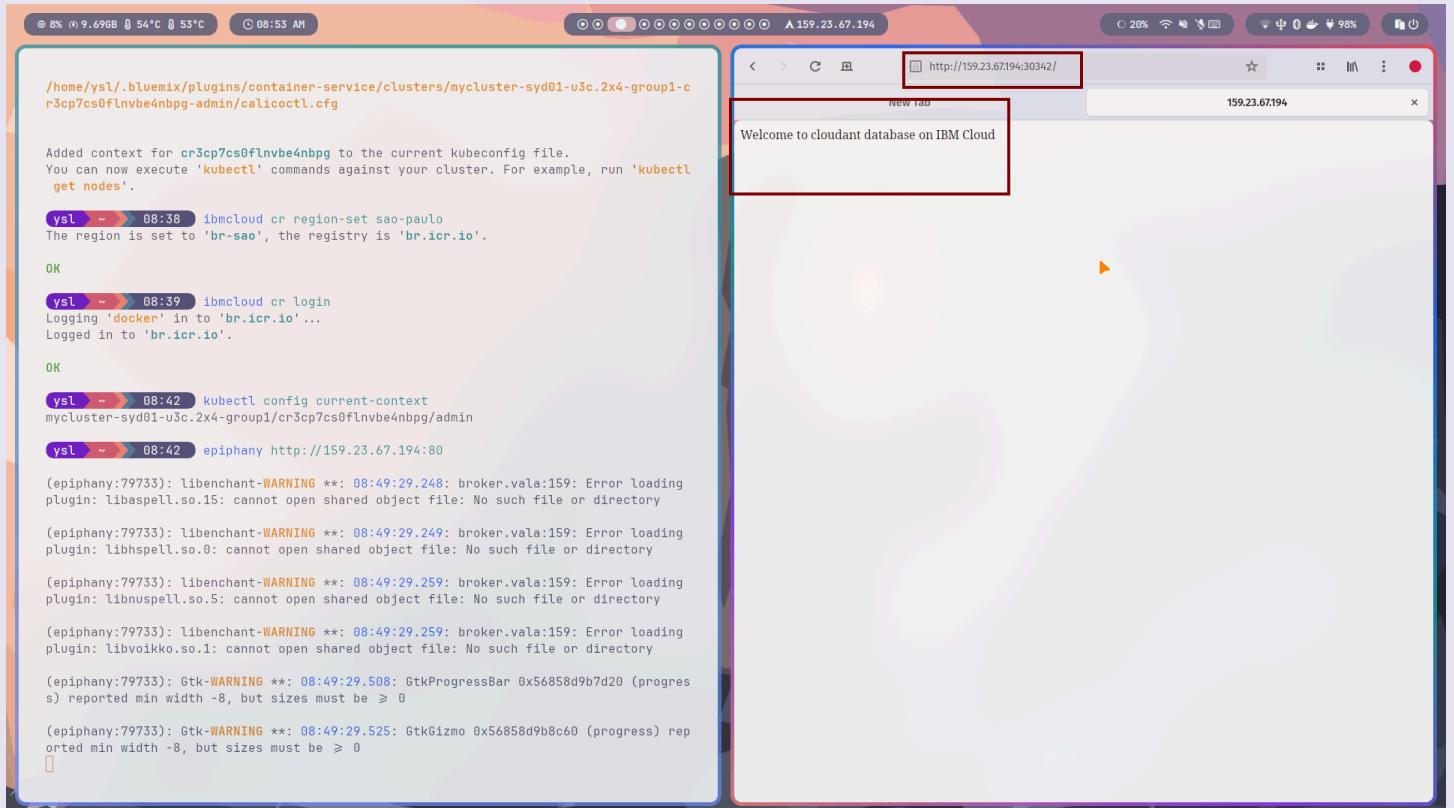
Terminal output:

```

yml _/Practical_10 M main !? @ v22.9.0 08:53 ibmcloud ks workers --cluster cr3cp7cs0flnvbe4nbpg
OK
ID          Public IP   Private IP   Flavor   State   Zone   Version
kube-cr3cp7cs0flnvbe4nbpg-myclustersy-default-00000108 159.23.67.194 10.210.8.213 u3c.2x4.encrypted normal  syd01  1.30.3_1533*
* To update to 1.30.5_1539 version, run 'ibmcloud ks worker update'. Review and make any required version changes before you update: 'https://ibm.biz/upworker'

yml _/Practical_10 M main !? @ v22.9.0 08:53 kubectl get service
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
arn-jk-service  NodePort  172.21.146.29 <none>        8080:32047/TCP 11d
aryan-service   NodePort  172.21.255.110 <none>       3000:31105/TCP 11d
bhavesh-service NodePort  172.21.142.89  <none>       3000:32322/TCP 41d
dev-service     NodePort  172.21.28.42   <none>       3000:30744/TCP 41d
dhrushi-service NodePort  172.21.19.120 <none>       3000:31421/TCP 10d
kubernetes      ClusterIP 172.21.0.1    <none>       443/TCP        52d
postgres        NodePort  172.21.38.27   <none>       5432:30613/TCP 41d
student-attendance-service LoadBalancer 172.21.182.213 159.23.70.165 80:30707/TCP 10d
yashlanu-service-pten NodePort  172.21.25.222 <none>       3000:30342/TCP 69s

```



```

/home/yasl/.bluemix/plugins/container-service/clusters/mycluster-syd01-u3c.2x4-group1-cr3cp7cs0flnvbe4nbpg-admin/calicoctl.cfg

Added context for cr3cp7cs0flnvbe4nbpg to the current kubeconfig file.
You can now execute 'kubectl' commands against your cluster. For example, run 'kubectl get nodes'.

yml ~ 08:38 ibmcloud cr region-set sao-paulo
The region is set to 'br-sao', the registry is 'br.icr.io'.

OK

yml ~ 08:39 ibmcloud cr login
Logging 'docker' in to 'br.icr.io'...
Logged in to 'br.icr.io'.

OK

yml ~ 08:42 kubectl config current-context
mycluster-syd01-u3c.2x4-group1/cr3cp7cs0flnvbe4nbpg/admin

yml ~ 08:42 epiphany http://159.23.67.194:80

(epiphany:79733): libenchant-WARNING **: 08:49:29.248: broker.vala:159: Error loading
plugin: libaspell.so.15: cannot open shared object file: No such file or directory

(epiphany:79733): libenchant-WARNING **: 08:49:29.249: broker.vala:159: Error loading
plugin: libhspell.so.0: cannot open shared object file: No such file or directory

(epiphany:79733): libenchant-WARNING **: 08:49:29.259: broker.vala:159: Error loading
plugin: libibuspell.so.0: cannot open shared object file: No such file or directory

(epiphany:79733): libenchant-WARNING **: 08:49:29.259: broker.vala:159: Error loading
plugin: libvoikko.so.1: cannot open shared object file: No such file or directory

(epiphany:79733): Gtk-WARNING **: 08:49:29.508: GtkProgressBar 0x56858d9b7d20 (progress
s) reported min width -8, but sizes must be ≥ 0

(epiphany:79733): Gtk-WARNING **: 08:49:29.525: GtkGizmo 0x56858d9b8c60 (progress) rep
orted min width -8, but sizes must be ≥ 0

```

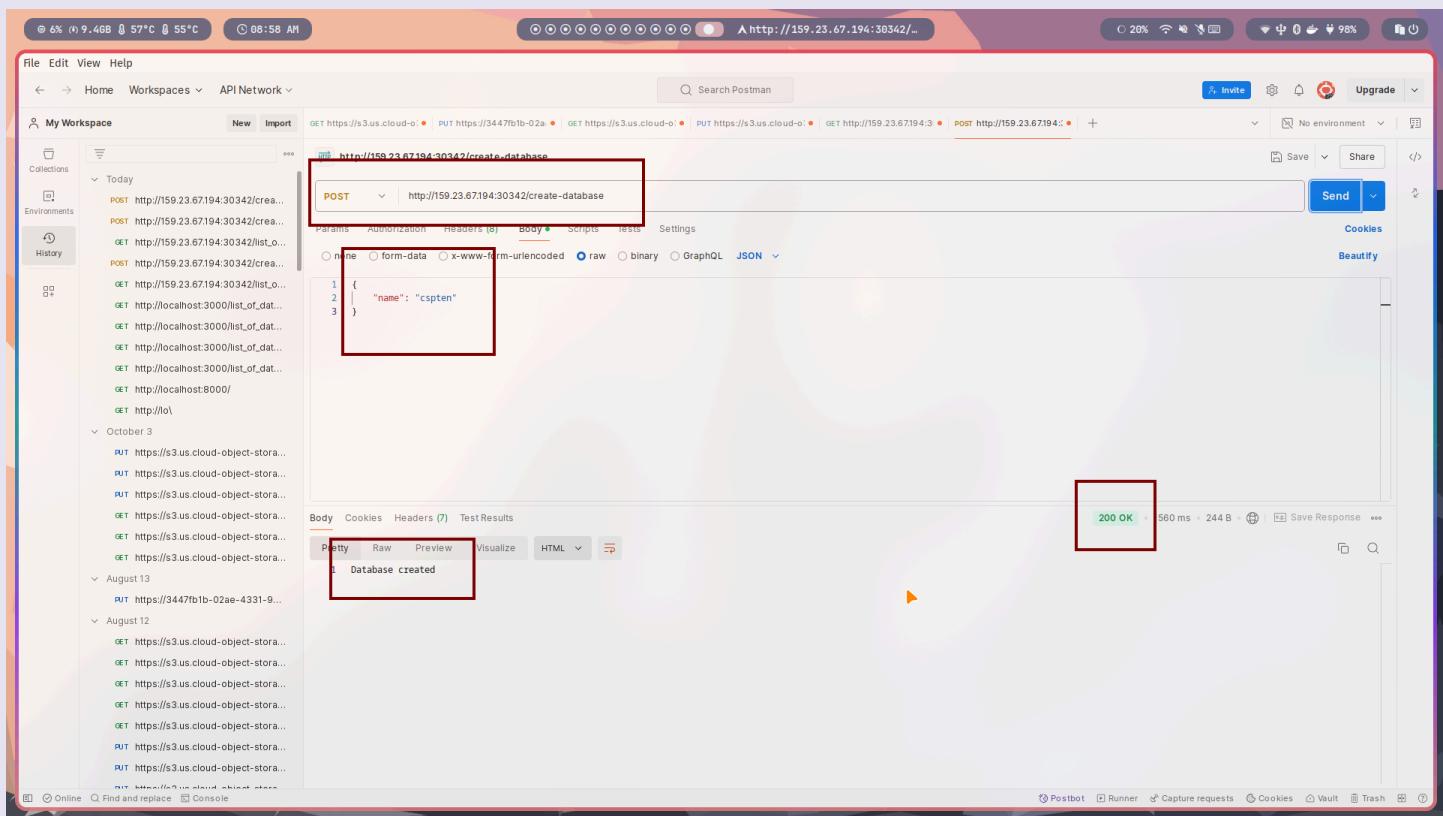
Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 71  
CS Practical 10

## 5. Check requests via Postman if the application accepts so

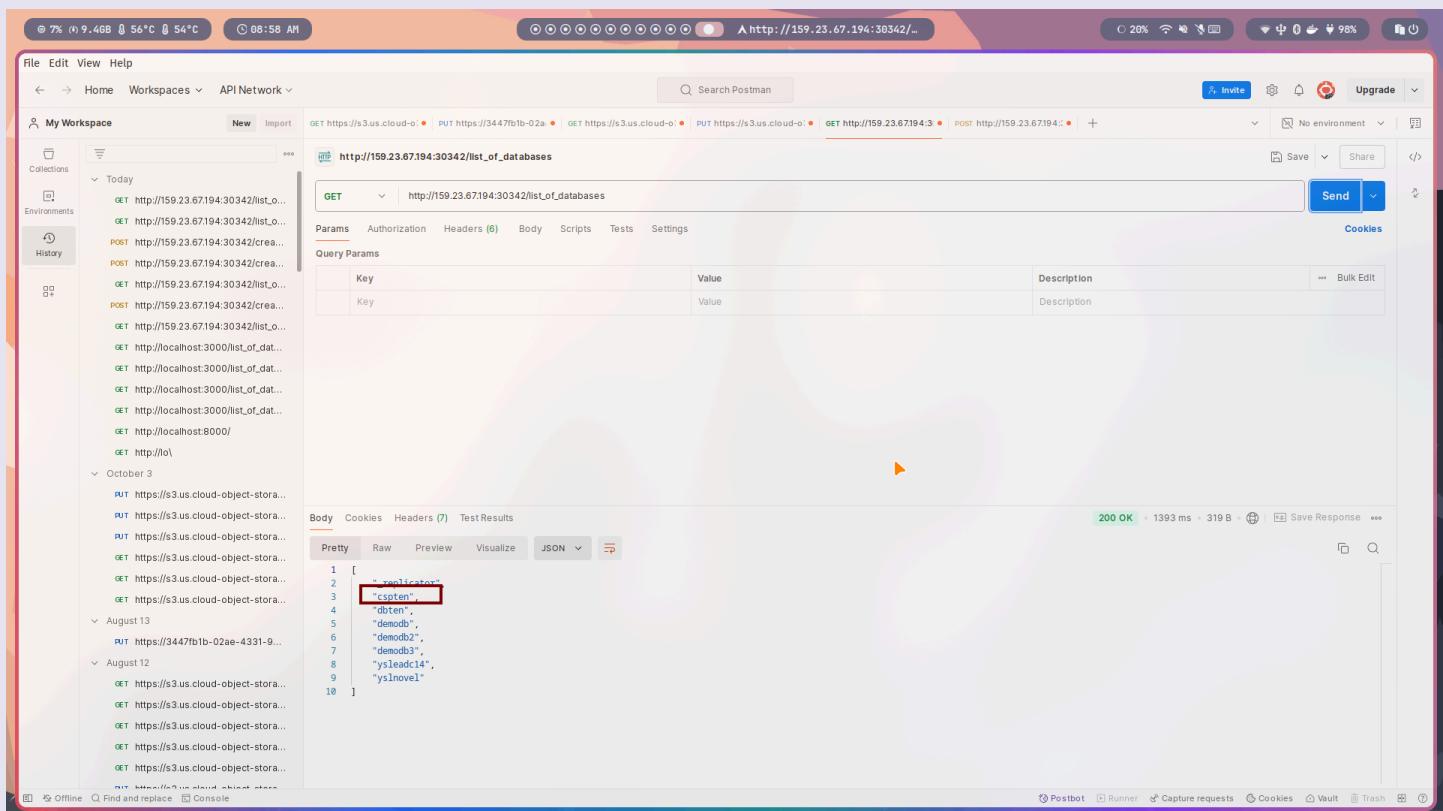
The screenshot shows the Postman application interface. A red box highlights a GET request to `http://159.23.67.194:30342/list_of_databases`. The response status is `200 OK`, and the response body is displayed in JSON format:

```
[{"_replicator", "dbten", "demodb", "demodb2", "demodb3", "ysledac14", "yslnovel"}]
```

**Name - Yash Lakhtariya**  
**Enrollment number - 21162101012**  
**Branch - CBA      Batch - 71**  
**CS Practical 10**



The screenshot shows the Postman application interface. A red box highlights the POST request to `http://159.23.67.194:30342/create-database`. The request body contains the JSON object `{"name": "csptn"}`. The response status is **200 OK** with a response time of 560 ms and a size of 244 B. The response body is `1 Database created`.



The screenshot shows the Postman application interface. A red box highlights the GET request to `http://159.23.67.194:30342/list_of_databases`. The response status is **200 OK** with a response time of 1393 ms and a size of 319 B. The response body is a JSON array containing database names: `[{"name": "realitv", "content": "content", "id": "1"}, {"name": "demodb1", "content": "content", "id": "2"}, {"name": "demodb2", "content": "content", "id": "3"}, {"name": "demodb3", "content": "content", "id": "4"}, {"name": "yslead14", "content": "content", "id": "5"}, {"name": "ys1novel", "content": "content", "id": "6"}]`.

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA      Batch - 71

CS Practical 10

## 6. Create global network policy to deny both egress and ingress traffic

The screenshot shows a terminal window with a light blue background and a dark blue header bar. The header bar includes system icons for battery level (2%), signal strength (9.46dB), temperature (52°C), and volume (49%). The time is 09:01 AM. The title bar says "A ysl@rog:~". The terminal window itself has a white background with a red border around the command and its output.

```
[ysl@rog ~]$ alias calicectl create -f <<EOF
apiVersion: projectcalico.org/v3
kind: GlobalNetworkPolicy
metadata:
  name: default-deny
spec:
  selector: projectcalico.org/namespace != "kube-system"
  types:
    - Ingress
    Egress
EOF
Successfully created 1 'GlobalNetworkPolicy' resource(s)
[ysl@rog ~]$
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 71  
CS Practical 10

## 7. Now the requests won't be accepted

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections, environments, and history. The main area displays a list of API requests. One specific request is highlighted with a red box: 'GET http://159.23.67.194:30342/list\_of\_databases'. Below this, a message in red text reads: 'goes on infinite loop, hence no allowing the requests after applying the global network policy'. At the bottom of the request details, there's a button labeled 'Send' with the placeholder text 'Click Send to get a response'. The status bar at the bottom of the browser window shows a message: 'goes on infinite loop, hence no allowing the requests after applying the global network policy'.