

Name: Patel Shivam S.

Enroll no: 21162101019

Sem: 7 Batch: 71 Branch: CBA

Sub: CS

Practical 10

AIM: 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.

SOLUTION:

➤ Login to ibmcloud and generate service credentials of cloudant.

The screenshot shows the 'Service credentials' section of the IBM Cloud Service Details interface. A red box highlights the credential entry for 'cs-prac10-shivam'. The credential details are as follows:

```
{
  "apikey": "6_EfrouL5x135FoT5gm781aSTI8kheQIfv0L9yGuUz",
  "host": "ab0b7dbc-af60-4512-a539-d65ed2b81993-bluemix.cloudantnosqldb.appdomain.cloud",
  "iam_apikey_description": "Auto-generated for key cni:v1:bluemix:public:cloudantnosqldb:eu-gb:a/931330a87d2847a583e6fecc1c4cea18:a9379721-6b4d-4b9d-a118-1bbcb6ef1ea:resource-key:53c60d273f-2907-40f2-aed-03193973dc9",
  "iam_apikey_expires": "2024-10-14T09:30:00Z",
  "iam_apikey_name": "cs-prac10-shivam",
  "iam_role_crn": "crn:v1:bluemix:public:iam::serviceRole:Manager",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam:identity:a/931330a87d2847a583e6fecc1c4cea18::serviceId:ServiceId-70d14e82-74c3-4cac-b0f2-559335669de",
  "password": "2805bed50eae53c60d2d706cdc954daf",
  "port": 443,
  "url": "https://apikey-v2-acs20d4fzeu5gtjtqg5m28e7ftio5rwebxme2lseese:2805bed50eae53c60d2d706cdc954daf@ab0b7dbc-af60-4512-a539-d65ed2b81993-bluemix.cloudantnosqldb.appdomain.cloud",
  "username": "apikey-v2-acs20d4fzeu5gtjtqg5m28e7ftio5rwebxme2lseese"
}
```

Give the permission

The screenshot shows the 'Permissions' section of the Cloudant Dashboard for the database 'cs-prac10-shivam'. A red box highlights the row for the API key 'apikey-v2-acs20d4fzeu5gtjtqg5m28e7ftio5rwebxme2lseese'. The permissions are set to checked for all roles: _admin, _reader, _writer, and _replicator. Below the table, a text input field contains the same API key, and a 'Grant Permissions' button is visible. A note at the bottom states: 'API keys can grant programmatic access to Cloudant databases. If you generate an API key, you can also manage that key's permissions.' A 'Generate API Key' button is also present.

push docker image to ibm container registry

The screenshot shows the VS Code interface. In the Explorer sidebar, there is a folder named 'PRAC10' containing files like 'index.js', 'Dockerfile', 'node_modules', 'package-lock.json', and 'package.json'. The 'Dockerfile' tab is selected, showing the following code:

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

In the bottom terminal window, the command `docker build -t prac10shivam:1.0 .` is being executed, and the output shows the build process:

```
[+] Building 11.5s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> [internal] transfer Dockerfile: 161B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> [internal] transfer context: 2B
=> [1/5] FROM docker.io/library/node:18-alpine@sha256:02376a266c84acbf45bd19440e08e48b1c8b98037417334046029ab585de03e2
=> [internal] load build context
=> [internal] transfer context: 106.38kB
=> 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 sha256:4ad25e06c4b68f441ea4ee17478232cc05eacb0299553356973b83d7024ac2f2
=> naming to docker.io/library/prac10shivam:1.0
```

The status bar at the bottom indicates the build was completed in 12s at 09:42:33 AM.

The screenshot shows the IBM Cloud Container Registry interface in a web browser. The left sidebar has sections for Overview, Quick start, Namespaces (selected), Repositories, Images, Trash, and Settings. The main area is titled 'Namespaces' and shows a table of namespaces:

Name	Resource group	Repository count	Image count	Retention policy
prac10shivam	default	1	1	Retain all images
Repository			1	Last updated 1 hour ago
br.icr.io/prac10shivam/prac10shivam				

Create deployment and service files.

```
deployment.yaml
service.yaml
```

```
kubectl apply -f deployment.yaml
deployment.apps/shivamm-app-prac10 configured
kubectl apply -f service.yaml
service/shivamm-service created
```

Workloads > Deployments > shivamm-app-prac10

Metadata

Name	Namespace	Created	Age	UID
shivamm-app-prac10	default	Oct 14, 2024	50 seconds ago	c7e1f1bb-2edf-4110-9d11-4dae0255e75e

Annotations

```
deployment.kubernetes.io/revision: 1
kubectl.kubernetes.io/last-applied-configuration
```

Resource information

Strategy	Min ready seconds	Revision history limit
RollingUpdate	0	10

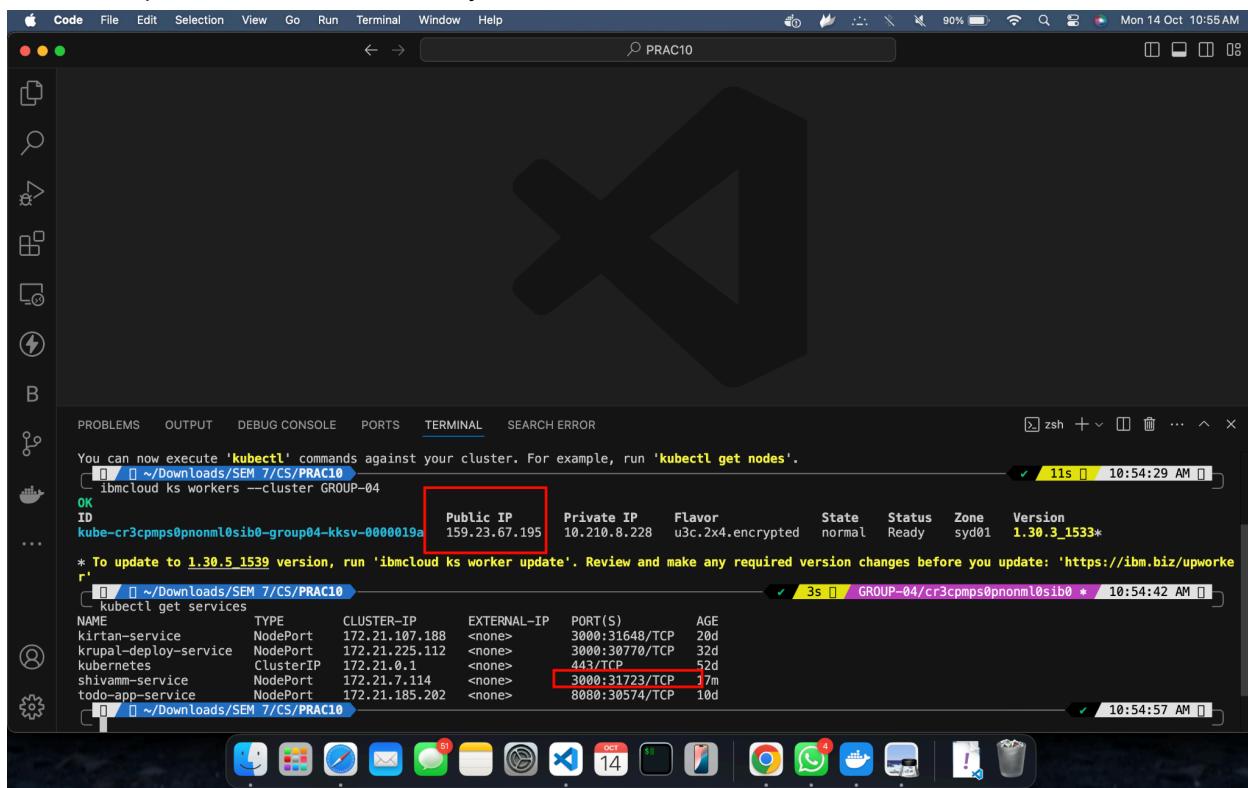
Selector

```
app: shivamm-app-prac10
```

Rolling update strategy

Max surge	Max unavailable
25%	25%

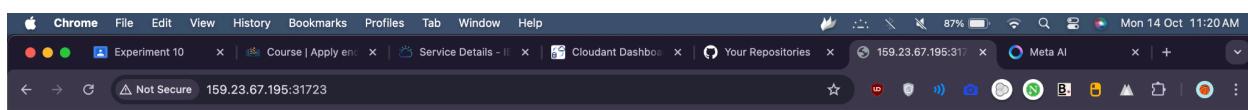
Get IP and port and check connectivity



```
You can now execute 'kubectl' commands against your cluster. For example, run 'kubectl get nodes'.
ibmcloud ks workers --cluster GROUP-04
OK
ID          Public IP      Private IP    Flavor   State  Status  Zone  Version
kube-cr3cpmps0pnomnl0sib0-group04-kksv-0000019a  159.23.67.195  10.210.8.228  u3c.2x4.encrypted  normal  Ready   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>'

```
ibmcloud ks workers --cluster GROUP-04
kubectl get services
NAME        TYPE      CLUSTER-IP     EXTERNAL-IP   PORT(S)      AGE
kirtan-service  NodePort  172.21.107.188 <none>        3000:31648/TCP  20d
krupal-deploy-service  NodePort  172.21.225.112 <none>        3000:30770/TCP  32d
kubernetes     ClusterIP  172.21.0.1    <none>        443/TCP      52d
shivamm-service  NodePort  172.21.7.114 <none>        3000:31723/TCP  7m
todo-app-service  NodePort  172.21.185.202 <none>        8080:30574/TCP  10d
```



Welcome to cloudant database on IBM Cloud

Check requests via Postman if the application accepts so

The screenshot shows the Postman application interface. A red box highlights the URL field which contains `http://159.23.67.195:31723/list_of_databases`. The response body is displayed in JSON format, showing a list of database names:

```
1  [
2    "_replicator",
3    "cspsten",
4    "dbten",
5    "demodb",
6    "demodb2",
7    "demodb3",
8    "ysleadc14",
9    "yslnovel"
10 ]
```

To see list of databases.

The screenshot shows the Postman application interface with a collection named "My Workspace". A red arrow points to the "shivam-database" entry in the response body, which was added after the initial list.

The response body is displayed in JSON format, showing a list of database names, including the newly added one:

```
1  [
2    "_replicator",
3    "cspsten",
4    "dbten",
5    "demodb",
6    "demodb2",
7    "demodb3",
8    "shivam-database", ←
9    "ysleadc14",
10   "yslnovel"
11 ]
```

To create a database.

The screenshot shows the Postman application interface. In the 'New Request' section, a POST request is being made to the URL `http://159.23.67.195:31723/create-database`. The request body contains the JSON object `{"name": "shivam-database"}`. The response status is 200 OK, and the response body is `Database created`.

The screenshot shows the Postman application interface. In the 'New Request' section, a GET request is being made to the URL `http://159.23.67.195:31723/list_of_databases`. The response status is 200 OK, and the response body is a JSON array containing database names: `["replicator", "cspsten", "dbten", "demode", "demode2", "demode3", "shivam-database", "yaleadcl4", "ylnovel"]`. A message `Sending request...` is visible in the response preview area.

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

The screenshot shows the VS Code interface with a dark theme. In the Explorer sidebar, there is a folder named 'PRAC10' containing files like 'node_modules', 'Dockerfile', and 'gnp.yaml'. The 'gnp.yaml' file is open in the editor and highlighted with a red box. Its content is as follows:

```
apiVersion: projectcalico.org/v3
kind: GlobalNetworkPolicy
metadata:
  name: default-deny
spec:
  selector: projectcalico.org/namespace != 'kube-system'
  types:
    - Ingress
    - Egress
```

In the bottom right corner of the code editor, there is a status bar showing 'Mon 14 Oct 11:28AM'. Below the editor, the terminal window shows two command-line sessions. The first session shows an error message about version mismatch:

```
calicoctl apply -f gnp.yaml
Version mismatch.
Client Version: v3.28.2
Cluster Version: 3.27.4
Use --allow-version-mismatch to override.
```

The second session shows the command being run again with the '--allow-version-mismatch' flag, resulting in a successful application:

```
calicoctl apply -f gnp.yaml --allow-version-mismatch
Successfully applied 1 'GlobalNetworkPolicy' resource(s)
```

The terminal also shows the time '11:27:46 AM' and '11:28:03 AM'.

The screenshot shows the Postman application window. On the left, the 'My Workspace' sidebar lists various environments and collections. In the center, a 'New Request' dialog is open for a collection named 'prac10-cs'. The request method is set to 'GET' and the URL is 'http://159.23.67.195:31723/list_of_databases'. The 'Headers' tab shows several environment variables defined:

Key	Value	Description
Key	Value	Description

Below the dialog, the 'Response' section displays an error message: 'Could not send request'. At the bottom, a status bar indicates 'Error: Request timed out' and 'View in Console'.

GlobalNetworkPolicy.yaml:

```
apiVersion: projectcalico.org/v3
kind: GlobalNetworkPolicy
metadata:
  name: default-deny
spec:
  selector: projectcalico.org/namespace != 'kube-system'
  types:
    - Ingress
    - Egress
```

Dockerfile:

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

Deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: shivamm-app-prac10
spec:
  selector:
    matchLabels:
      app: shivamm-app-prac10
  replicas: 1
  template:
    metadata:
      labels:
        app: shivamm-app-prac10
    spec:
      containers:
        - name: nodecontainer
```

```
# image: au.icr.io/shivamnamespace/newibmimageshivamprac4:2.0
image: br.icr.io/yashlani-nmspc/yslcldnt:1.0
ports:
- containerPort: 3000
```

service.yaml:

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: shivamm-app-prac10
  name: shivamm-service
  namespace: default
spec:
  type: NodePort
  ports:
  - name: http
    protocol: TCP
    port: 3000
  selector:
    app: shivamm-app-prac10
```