# Kubernetes Project :- Scalable, Decoupled WordPress MySQL project

08.07.2023 - 10.07.2023

—

Rakesh Kumar Jangid
jangidrakesh71@gmail.com
www.linkedin.com/rakeshkumarjangid
www.github.com/rakesh08061994
+91 9887211207

## Overview

"The goal of this project is to deploy a scalable and highly available WordPress application using Kubernetes. We will utilize MySQL for database storage and configure various components like Persistent Volumes, Persistent Volume Claims, ConfigMaps, Secrets, and load balancers to ensure a robust and reliable deployment."

Goals

1. **Project Goal:** Scalable, Decoupled WordPress Deployment with MySQL on Kubernetes
2. **Objective:** Deploy a highly scalable and resilient WordPress application on Kubernetes, utilizing the decoupling of WordPress and MySQL components for improved flexibility and scalability. Some more are following-
3. **Scalability:** Enable seamless horizontal scaling of the WordPress application by leveraging Kubernetes' auto-scaling capabilities. The architecture should support handling increasing user traffic and ensure optimal performance.

4. **Decoupled Architecture:** Implement a decoupled architecture that separates the WordPress application from the MySQL database. This design allows for independent scaling, upgrades, and maintenance of each component, improving overall system flexibility and reducing dependencies.
5. **Persistent Data Storage:** Configure Persistent Volumes (PVs) using NFS network storage to provide reliable and persistent storage for the MySQL database and WordPress application data. The solution should leverage NFS to ensure data persistence and enable access to the same storage from multiple pods and nodes.
6. **Dynamic Configuration Management:** Utilize Kubernetes ConfigMaps and Secrets to manage environment variables and configuration settings for the WordPress and MySQL containers. This approach simplifies configuration updates, enhances security by storing sensitive information securely, and promotes easy management of application settings.
7. **High Availability and Resiliency:** Employ pod scheduling techniques to ensure high availability and resiliency of the WordPress and MySQL components. Enable automated rescheduling of pods in case of failures or termination, and designate a backup node to store critical application data, minimizing potential downtime.
8. **Load Balancing:** Implement Network Load Balancers and Application Load Balancers to distribute incoming traffic across multiple instances of the WordPress application. This ensures even load distribution, optimal performance, and seamless scaling of the application.

9. **Role-Based Access Control (RBAC):** Implement RBAC to control access to Kubernetes resources. Create two users with restricted privileges and grant them namespace-level access through ClusterRoleBindings or RoleBindings, ensuring proper segregation of duties and securing the cluster.

---

## Architecture Overview:

The project architecture consists of the following components:

- ➔ **MySQL:** We will deploy a MySQL container using the ClusterIP service type for internal access within the cluster.
- ➔ **WordPress:** We will deploy the WordPress application using the LoadBalancer service type for external access. This will allow users to access the application through a specific port on each Kubernetes node. This will work externally by provisioning a cloud provider's load balancer.

- ➔ **Persistent Volumes (PV) and Persistent Volume Claims (PVC):** We will configure PV and PVC to provide persistent storage for MySQL and WordPress, ensuring data persistence even when pods are rescheduled or scaled.
- ➔ **ConfigMaps and Secrets:** We will utilize ConfigMaps to store configuration settings that can be consumed by the MySQL and WordPress containers. Secrets will be used to securely store sensitive data such as passwords and API keys.
- ➔ **Pod Scheduling:** We will employ pod scheduling techniques to ensure high availability by rescheduling pods if they fail or are terminated. Additionally, we will designate one node as a backup node to store critical application data.
- ➔ **Load Balancers:** We will set up a Load Balancer ex:-  Application Load Balancer to distribute traffic across the services and ensure optimal performance and scalability."

## Detailed Deployment Steps:

### ➔ Step-1 (Design your Cluster)

- **Instance Type:** Choose T2.Medium
- **Application and OS Images (Amazon Machine Image):** OS UBUNTU
- **Configure Storage:** 20 GB Instance Each
- Setup and configure Hostname & Hosts file on each instance
- Mount NFS on each instances node port with **`/etc/fstab`** entry
- Ping all machines and ensure all machines are connected properly.

### ➔ Step-2 (Deploy Docker and setup kubernetes on each machine )

**On your Instance machines run following commands to deploy kubernetes setup:**

```
# apt update
# git clone -b master https://github.com/rakesh08061994/Docker/
# cd Docker
# chmod a+x *
# ./kube-setup.sh
—- Follow instructions to setup kubernetes on each machine as per script instructions from kubeadm init on master node and kubeadm join  on slave nodes—---
$ kubectl get nodes -o wide
```

➔ Step-3 (Create two users on kubernetes cluster to give access to the namespace level, here two users are **"Anjali"** & **"Rakesh"**.)

```
 - on master node repeat for each user, here i am start with rakesh user-
root@master:~#  adduser rakesh
Adding user `rakesh' ...
Adding new group `rakesh' (1001) ...
Adding new user `rakesh' (1001) with group `rakesh' ...
Creating home directory `/home/rakesh' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for rakesh
Enter the new value, or press ENTER for the default
	Full Name []:
	Room Number []:
	Work Phone []:
	Home Phone []:
	Other []:
Is the information correct? [Y/n] y
```

---

```
root@master:~# su - rakesh
rakesh@master:~$ openssl genrsa -out rakesh.key 2048
rakesh@master:~$ ls
rakesh.key
rakesh@master:~$ openssl req -new -key rakesh.key -out rakesh.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
```

```
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:in
State or Province Name (full name) [Some-State]:raj
Locality Name (eg, city) []:jaipur
Organization Name (eg, company) [Internet Widgits Pty Ltd]:grras
Organizational Unit Name (eg, section) []:devops
Common Name (e.g. server FQDN or YOUR name) []:rakesh
Email Address []:jangidrakesh71@gmail.com
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password [ ]:@#User5
An optional company name [ ]:grras
rakesh@master:~$ ls
rakesh.csr  rakesh.key
```

---

```
rakesh@master:~$ sudo su -
root@master:~# mkdir rakesh-key
root@master:~# cd rakesh-key/
root@master:~/rakesh-key# cp /home/rakesh/rakesh.*   .
root@master:~/rakesh-key# ls
rakesh.csr  rakesh.key
root@master:~/rakesh-key# cp /etc/kubernetes/pki/ca.* .
root@master:~/rakesh-key# ls
ca.crt  ca.key  rakesh.csr  rakesh.key
root@master:~/rakesh-key#  openssl  x509  -req  -in  rakesh.csr  -CA  ca.crt  -CAkey  ca.key
-CAcreateserial -out rakesh.crt -days 365
Certificate request self-signature ok
subject=C = in, ST = raj, L = jaipur, O = grras, OU = devops, CN = rakesh, emailAddress =
jangidrakesh71@gmail.com
```

```
root@master:~/rakesh-key#  ls -al rakesh.crt
-rw-r--r-- 1 root root 1151 Jul 10 11:04 rakesh.crt
root@master:~/rakesh-key# chown rakesh rakesh.crt
root@master:~/rakesh-key# ls -al rakesh.crt
-rw-r--r-- 1 rakesh root 1151 Jul 10 11:04 rakesh.crt
root@master:~/rakesh-key# cp rakesh.* /home/rakesh/
```

```
root@master:~/rakesh-key# su - rakesh
rakesh@master:~$ ls
rakesh.crt  rakesh.csr  rakesh.key
rakesh@master:~$ sudo su -
root@master:~# pwd
/home/root
root@master:~# kubectl get nodes
NAME     STATUS   ROLES           AGE   VERSION
master   Ready    control-plane   9d    v1.27.3
node1    Ready    <none>          9d    v1.27.3
node2    Ready    <none>          9d    v1.27.3
node3    Ready    <none>          9d    v1.27.3
root@master:~# ls .kube/
cache  config
root@master:~# cd rakesh-key
root@master:~/rakesh-key# ls
ca.crt  ca.key  config  rakesh.crt  rakesh.csr  rakesh.key
root@master:~/rakesh-key# vi config
```

Note:- Configure and setup for rakesh user, using changing config file as per rakesh user

```
root@master:~/rakesh-key# vi config
apiVersion: v1
```

```
clusters:
- cluster:
Certificate-authority-data:   kFRc0ZBREFWTVJNd0VRWURWUVFERXdwcmRXSmGRHVnpNQjR
RFRJek1EY3dNVEEURZMU0xb1hEVE16TURZeU9EQTRNRFkxTTFvd0ZURVRNQkVHQTFVRQpB
eE1LYTNWaVpYSnVaWFJsY3pDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBRENDQV
FvQ2dnRUJBTGx1CmR6cFFhQ1lxVXd5UEs1OUl3VlI4cUd2REJmbEVLYU1yZkpOeFczNVp4eFp
CRlFCZ3ovSTQvOTJqRjkralRsWlEKU2trZW5aTHdYUUNtaHIyNWpEMWFsS1RrZ01la2IyYWEyQ
1hDK1AvSGJMbjlhSldtdmg5ZCsvbk96dXVPTG5jQQpyK1BBRC9PSS9QOUExRXd2cm9HMkV1
NDRGQ1ZubFRhNFFrTVQwTjVlZkRzK0pHNm5HZUJCSWdNRUhMR0RDZXJ6CmswaS9qTWI0b
TJLSFQ5elRIOEtETVU1bWVJSmVxYVFwZTE0WnBtRHRzWHNMbXRjcWtzV0xtRDNTQ2kvSytnd
FkKVVZ0RXlkRTAreVo2S1FGRWljVVhFRE03VGo4c0xrNjNaV0RxQW5IanVIZlg3ZThueUk1T213
aFgvcjlMSktUQwpiRllncmdycEp4WU41NVVMekYwQ0F3RUFBYU5aTUZjd0RnWURWUjBQQVF
IL0JBUURBZ0trTUE4R0ExVWRFd0VCCi93UUZNQU1CQWY4d0hRWURWUjBPQkJZRUZQeFhVc
W9XZTRiME9HS2RCSUhNeFNvaitvQnRNQlVHQTFVZEVRUU8KTUF5Q0NtdDFZbVZ5Ym1WMF
pYTXdEUVlKS29aSWh2Y05BUUVMQlFBRGdnRUJBQ0J1UFAwVWNXcU1zZ25vVTBpTgowb202
bzlvVW9MZ3lZT1ltTHhFS3VEaStpakNzOXBTUjFLdldrVUx1Uzh6WitQQ3dnaitBcktVdTJFeWk4
ajNlCmJacUZQRThjWGJaM1JIUUwzTGJJdlBWVUpQVVQrOU9nSStjUEgxamRaWWd6dGVJemh
2bzdVY3BHQUpvaVY5U2YKRkVmMzg3Qnd2emhnY2Y5Ync4dkpHN25IRWxoWENGNkdHZDJi
ZlNGZmFRdlRZUmhCaGFTclRMdEZKM3IycjhTZQpRNUh4VDhpaENMWFhuendpekxlZENLM0
0wYktETWhvQXhhOFNCVFlHU1gzYjNVWGV1eDlDcTBaVys0ZnhtSlBVCjNvYWRGRE5yd0xVU
mZ1U2tsekwyZHFPNGFMR1I2c2ZPQ2FxaExxnMDd2Qkh1MVc4bEhudURpSFphRC90RFZHdlY
KTStFPQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg==
    server: https://172.51.2.62:6443
  name: kubernetes
contexts:
- context:
    namespace: grras < ——----Namespace for rakesh----------->
    cluster: kubernetes
    user: rakesh
  name: rakesh@kubernetes
current-context: rakesh@kubernetes
kind: Config
preferences: {}
users:
- name: rakesh
  user:
    Client-certificate-data: < —---- `cat rakesh.crt | base64` result put here —------ >
    Client-key-data: < —-- `cat rakesh.key | base64` result put here —---------- >
```

---

```
root@master:~/rakesh-key# ls
ca.crt  ca.key  config  rakesh.crt  rakesh.csr  rakesh.key
root@master:~/rakesh-key# ls -al
total 36
drwxr-xr-x  2 root root 4096 Jul 10 11:46 .
drwx------ 10 root root 4096 Jul 10 11:46 ..
-rw-r--r--  1 root root 1099 Jul 10 11:02 ca.crt
-rw-------  1 root root 1675 Jul 10 11:02 ca.key
-rw-------  1 root root 5658 Jul 10 11:46 config
-rw-r--r--  1 root root 1151 Jul 10 11:20 rakesh.crt
-rw-r--r--  1 root root 1110 Jul 10 11:20 rakesh.csr
-rw-------  1 root root 1704 Jul 10 11:20 rakesh.key
root@master:~/rakesh-key# chown rakesh:rakesh config
root@master:~/rakesh-key# ls -al
total 36
drwxr-xr-x  2 root   root   4096 Jul 10 11:46 .
drwx------ 10 root   root   4096 Jul 10 11:46 ..
-rw-r--r--  1 root   root   1099 Jul 10 11:02 ca.crt
-rw-------  1 root   root   1675 Jul 10 11:02 ca.key
-rw-------  1 rakesh rakesh 5658 Jul 10 11:46 config
-rw-r--r--  1 root   root   1151 Jul 10 11:20 rakesh.crt
-rw-r--r--  1 root   root   1110 Jul 10 11:20 rakesh.csr
-rw-------  1 root   root   1704 Jul 10 11:20 rakesh.key
root@master:~/rakesh-key# cp -a config /home/rakesh/
root@master:~/rakesh-key# su - rakesh
rakesh@master:~$ ls
config  rakesh.crt  rakesh.csr  rakesh.key
rakesh@master:~$ ls -al
```

```
total 44
drwxr-x--- 2 rakesh rakesh 4096 Jul 10 11:48 .
drwxr-xr-x 4 root   root   4096 Jul 10 10:48 ..
-rw------- 1 rakesh rakesh   92 Jul 10 10:59 .bash_history
-rw-r--r-- 1 rakesh rakesh  220 Jul 10 10:48 .bash_logout
-rw-r--r-- 1 rakesh rakesh 3771 Jul 10 10:48 .bashrc
-rw-r--r-- 1 rakesh rakesh  807 Jul 10 10:48 .profile
-rw------- 1 rakesh rakesh 5658 Jul 10 11:46 config
-rw-r--r-- 1 rakesh root   1151 Jul 10 11:04 rakesh.crt
-rw-rw-r-- 1 rakesh rakesh 1110 Jul 10 10:56 rakesh.csr
-rw------- 1 rakesh rakesh 1704 Jul 10 10:52 rakesh.key
rakesh@master:~$ mkdir .kube
rakesh@master:~$ cp -a config .kube/
rakesh@master:~$ tree -al  .kube/
.kube/
└── config

0 directories, 1 file
```

---

Note:- User rakesh is created, authenticated and authorized with kubernetes master CA certifications successfully. Do the same with user anjali. Now we will create a ClusterRole and bind with ClusterRoleBinding and a namespace grras.

---

```
rakesh@master:~$ kubectl get nodes
Error from server (Forbidden): nodes is forbidden: User "rakesh" cannot list resource "nodes" in API group "" at the cluster scope
rakesh@master:~$ sudo su -
```

---

```
root@master:~# vi rakesh-role.yml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: rakesh-role
  namespace: grras
rules:
- apiGroups: ["", "extensions", "apps", "batch"]
  resources: ["*"]
  verbs: ["*"]

root@master:~# kubectl create -f rakesh-role.yml
role.rbac.authorization.k8s.io/rakesh-role created
```

---

Note: This Role grants the "rakesh" user full control (verbs: ["*"]) over all resources (resources: ["*"]) in the grras namespace.

---

Note:- We already set up the NFS mount on /mnt directory and created PV and PVC with slow and fast subclass. So Check PV and PVC status and mount point

```
root@master:/mnt# kubectl get pv
NAME   CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM        STORAGECLASS
REASON   AGE
pv01   5Gi        RWX            Recycle          Bound    grras/pvc02   slow             4d17h
pv02   5Gi        RWX            Recycle          Bound    grras/pvc04   fast            4d17h
pv03   5Gi        RWX            Recycle          Bound    grras/pvc03   slow             4d17h
pv04   5Gi        RWX            Recycle          Bound    grras/pvc01   slow             4d17h
pv05   5Gi        RWX            Recycle          Bound    grras/pvc05   fast            4d17h
root@master:/mnt# kubectl get pvc
NAME    STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
pvc01   Bound    pv04     5Gi        RWX            slow           4d17h
pvc02   Bound    pv01     5Gi        RWX            slow           4d16h
pvc03   Bound    pv03     5Gi        RWX            slow           4d16h
pvc04   Bound    pv02     5Gi        RWX            fast          4d16h
pvc05   Bound    pv05     5Gi        RWX            fast          4d16h
root@master:/mnt# tree /mnt/
/mnt/
```

```
├── nfs_mount1
├── nfs_mount2
├── nfs_mount3
├── nfs_mount4
│   ├── #ib_16384_0.dblwr
│   ├── #ib_16384_1.dblwr
│   ├── #innodb_redo  [error opening dir]
│   ├── #innodb_temp  [error opening dir]
│   ├── auto.cnf
│   ├── binlog.000001
│   ├── binlog.000002
│   ├── binlog.index
│   ├── ca-key.pem
│   ├── ca.pem
│   ├── client-cert.pem
│   ├── client-key.pem
│   ├── ib_buffer_pool
│   ├── ibdata1
│   ├── ibtmp1
│   ├── mysql  [error opening dir]
│   ├── mysql.ibd
│   ├── mysql.sock -> /var/run/mysqld/mysqld.sock
│   ├── performance_schema  [error opening dir]
│   ├── private_key.pem
│   ├── public_key.pem
│   ├── server-cert.pem
│   ├── server-key.pem
│   ├── sys  [error opening dir]
│   ├── undo_001
│   └── undo_002
└── nfs_mount5
```

```
10 directories, 21 files
```

---

```
root@master:~# vi rakesh-role-binding.yml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: rakesh-rolebinding
  namespace: grras
subjects:
- kind: User
  name: rakesh
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: rakesh-role
  apiGroup: rbac.authorization.k8s.io
root@master:~# kubectl create -f rakesh-role-binding.yml
rolebinding.rbac.authorization.k8s.io/rakesh-rolebinding created
```

---

```
root@master:~# su - rakesh
Password:
rakesh@master:~$ kubectl get nodes
Error from server (Forbidden): nodes is forbidden: User "rakesh" cannot list resource "nodes" in API group "" at the cluster scope
rakesh@master:~$ kubectl get pods
NAME                            READY   STATUS    RESTARTS       AGE
nginx-deploy-b5bcccfd4-cftpx   1/1     Running   1 (2d4h ago)   2d4h
nginx-deploy-b5bcccfd4-cxl86   1/1     Running   1 (2d4h ago)   2d4h
nginx-deploy-b5bcccfd4-kn4n8   1/1     Running   1 (2d4h ago)   2d4h
nginx-deploy-b5bcccfd4-nsj8l   1/1     Running   1 (2d4h ago)   2d4h
```

```
nginx-deploy-b5bcccfd4-xnqrb   1/1     Running   1 (2d4h ago)   2d4h
```

---

```
~~~~~ CONGRATULATIONS~~~~~~ User Rakesh is able to access grras namespace
rakesh@master:~$ kubectl delete deployments.apps nginx-deploy
deployment.apps "nginx-deploy" deleted
```

## ➔ Step4 (Create And Setup Mysql Deployment, ConfigMap/Secrets and ClusterIP Service)

```
rakesh@master:~$ vi sql-configmap1.yml
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql-configmap1
data:
  database-name: mysqldba
rakesh@master:~$ kubectl create -f sql-configmap1.yml
ConfigMap "mysql-configmap1" created
rakesh@master:~$ kubectl get cm
rakesh@master:~$ kubectl describe cm mysql-configmap1
```

---

```
rakesh@master:~$ vi mysql-secret1.yml
apiVersion: v1
kind: Secret
metadata:
  name: mysql-secret1
type: Opaque
data:
  root-password: <base64-encoded-root-password>
  username: <base64-encoded-username>
  password: <base64-encoded-password>
```

```
rakesh@master:~$ kubectl create -f mysql-secret1.yml
Secret "mysql-secret1" created
rakesh@master:~$ kubectl get secret
rakesh@master:~$ kubectl describe secret mysql-secret1
```

---

```
rakesh@master:~$ vi mysql-deployment1.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment1
spec:
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - name: mysql
        image: mysql:latest
        env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-secret1
              key: root-password
        - name: MYSQL_USER
          valueFrom:
            secretKeyRef:
```

```
          name: mysql-secret1
          key: username
    - name: MYSQL_PASSWORD
      valueFrom:
        secretKeyRef:
          name: mysql-secret1
          key: password
    - name: MYSQL_DATABASE
      valueFrom:
        configMapKeyRef:
          name: mysql-configmap1
          key: database-name
    ports:
    - containerPort: 3306
```

```
rakesh@master:~$ kubectl create -f  mysql-deployment1.yml
deployment.apps "mysql-deployment1" created
rakesh@master:~$ kubectl get pods
rakesh@master:~$ kubectl describe deployment.apps/mysql-deployment1
```

```
rakesh@master:~/kubernetes_project$ pwd
/home/rakesh/kubernetes_project
rakesh@master:~/kubernetes_project$ ls
mysql-configmap1.yml  mysql-deployment1.yml  mysql-secret1.yml
rakesh@master:~/kubernetes_project$ kubectl get pods
NAME                                READY   STATUS    RESTARTS      AGE
mysql-deployment1-5fdd5c87c4-nsx6z   1/1     Running   1 (79s ago)   97s
```

```
rakesh@master:~/kubernetes_project$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
```

```
mysql-deployment-5fdd5c87c4-d6dk6   1/1     Running   0          113s
rakesh@master:~/kubernetes_project$ kubectl exec -it mysql-deployment-5fdd5c87c4-\
d6dk6 -- bash
bash-4.4# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.33 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| mysqldba           |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql> exit
Bye
```

```
rakesh@master:~/kubernetes_project$ Vi mysql-expose.yml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  name: mysql-expose
spec:
  ports:
  - port: 3306
    protocol: TCP
    targetPort: 3306
  selector:
    app: mysql
  type: ClusterIP
status:
  loadBalancer: {}
rakesh@master:~/kubernetes_project$ kubectl create -f mysql-expose.yml
rakesh@master:~/kubernetes_project$ kubectl get svc

NAME           TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
mysql-expose   ClusterIP   10.99.79.235   <none>        3306/TCP   38s
rakesh@master:~/kubernetes_project$
```

NOTE: You can check mysql outside with in cluster

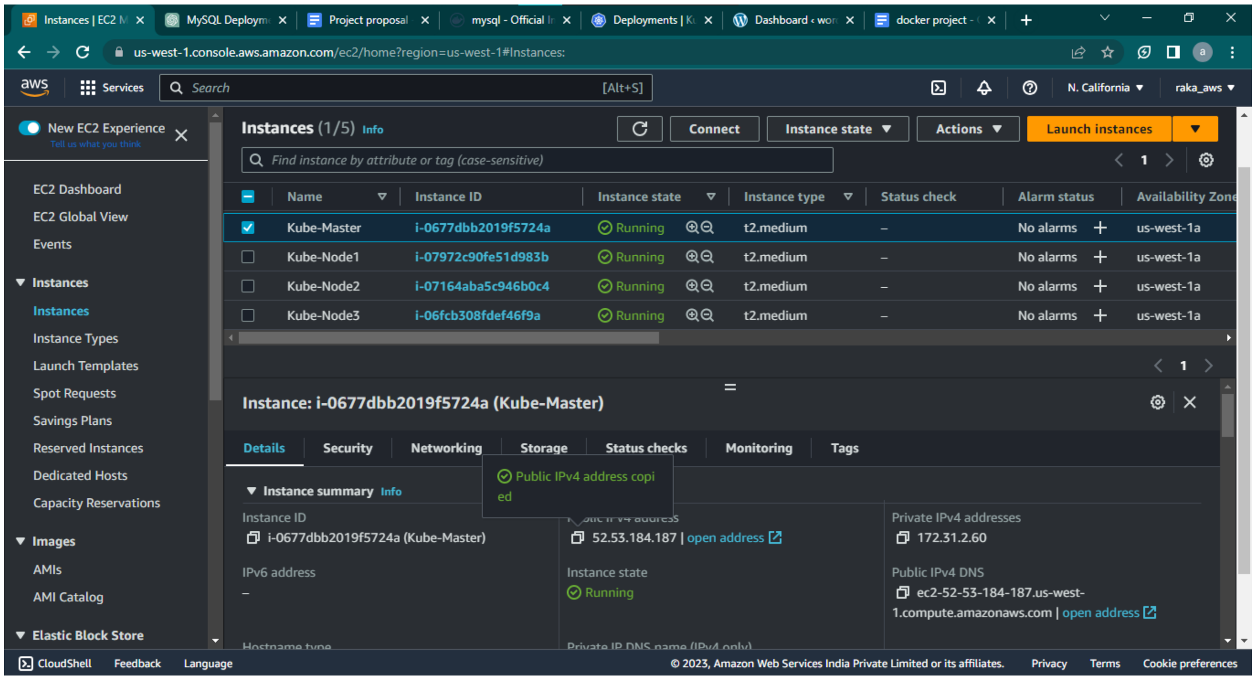
For this you have to require install mysql-server package and then run this command

```
rakesh@master:~/kubernetes_project$ sudo apt-get install mysql-server
rakesh@master:~/kubernetes_project$ kubectl get svc
NAME           TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
```

```
mysql-expose   ClusterIP   10.99.79.235   <none>        3306/TCP   4m49s
rakesh@master:~/kubernetes_project$ mysql -h 10.99.79.235  -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.33 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| mysqldba           |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.01 sec)
```

➔ Step5 (Create And Setup Wordpress Deployment, ConfigMap/Secrets and LoadBalancer Service. link with mysql deployment using environment variable)

```
rakesh@master:~/kubernetes_project$ vi wordpress-deployment.yml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: wordpress-deployment
spec:
 selector:
  matchLabels:
   app: wordpress
 template:
  metadata:
   labels:
    app: wordpress
  Spec:
   replicas: 5
   containers:
   - name: wordpress
    image: wordpress:latest
    ports:
    - containerPort: 80
    env:
    - name: WORDPRESS_DB_HOST
     value: mysql-expose  —-------------------> Check This before launch
    - name: WORDPRESS_DB_USER
     valueFrom:
      secretKeyRef:
       name: mysql-secret1
       key: username —-------------------> Check This before launch
    - name: WORDPRESS_DB_PASSWORD
     valueFrom:
          secretKeyRef:
            name: mysql-secret1
            key: password  —----------------> Check This before launch
      - name: WORDPRESS_DB_NAME
        valueFrom:
          configMapKeyRef:
            name: mysql-configmap1
            key: database-name  —-------------------> Check This before launch
      volumeMounts:
      - name: wordpress-persistent-storage
        mountPath: /var/www/html
    volumes:
    - name: wordpress-persistent-storage
      persistentVolumeClaim:
        claimName: pvc02  —-------------------> Check This before launch

rakesh@master:~/kubernetes_project$ kubectl create -f wordpress-deployment1.yml
deployment.apps/wordpress-deployment1 created
rakesh@master:~/kubernetes_project$ kubectl get pods
NAME                                     READY   STATUS    RESTARTS   AGE
mysql-deployment1-5fdd5c87c4-5x8j8       1/1     Running   0          8m28s
wordpress-deployment1-775c87c79c-llj5h   1/1     Running   0          6m31s
rakesh@master:~/kubernetes_project$
```



Instances (1/5) Info

| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone |
|---|---|---|---|---|---|---|
| Kube-Master | i-0677dbb2019f5724a | Running | t2.medium | – | No alarms | us-west-1a |
| Kube-Node1 | i-07972c90fe51d983b | Running | t2.medium | – | No alarms | us-west-1a |
| Kube-Node2 | i-07164aba5c946b0c4 | Running | t2.medium | – | No alarms | us-west-1a |
| Kube-Node3 | i-06fcb308fdef46f9a | Running | t2.medium | – | No alarms | us-west-1a |

Instance: i-0677dbb2019f5724a (Kube-Master)

Instance summary Info

Public IPv4 address copied

Instance ID: i-0677dbb2019f5724a (Kube-Master)
Public IPv4 address: 52.53.184.187 | open address
Private IPv4 addresses: 172.31.2.60
IPv6 address: –
Instance state: Running
Public IPv4 DNS: ec2-52-53-184-187.us-west-1.compute.amazonaws.com | open address
Hostname type
Private IP DNS name (IPv4 only)

<——----------------------------**Wordpress Customization Needed–**—------------------------------------------->

## → Step-6 ( Expose Wordpress Deployment using LoadBalancer Internally )

```
rakesh@master:~$ kubectl expose deployment wordpress-deployment1 --port 80 --type
LoadBalancer  --name=wordpress-expose1

service/wordpress-expose1 exposed

rakesh@master:~$
```

## → Note:- ( Important Key-Points  From Step1 to Step5 )

—-------------~~~~~~~~~~*******************************************~~~~~~~~~~------------—

*Please Make sure you have mentioned the correct value of ConfigMap, Secrets, HASH values for secrets, PV, PVC, deployment Expose services, hostname of Mysql service. My personal recommendation is to check and test and then move further. Make your own notes.*

—-------------~~~~~~~~~~*************************************~~~~~~~~~--------------

Now we will do the same with User “Anjali”

Create User Anjali and assign RBAC, Authorization and Authentication access

Create ClusterRole and ClusterRoleBinding to access “Gip” Namespace

Create Mysql, Wordpress, ConfigMap, Secrets, PV, PVC, Volumes, Quota etc. and launch “Wordpress-2”

## ➔ Step-7 ( Repeat Steps 4 & 5  with different user anjali)

- We will create one more wordpress + mysql application controlled by User Anjali and namespace Gip.

```
anjali@master:~$ vi sql-configmap2.yml
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql-configmap2
data:
  database-name: mysqldbb
anjali@master:~$ kubectl create -f sql-configmap2.yml
ConfigMap "mysql-configmap2" created
anjali@master:~$ kubectl get cm
anjali@master:~$ kubectl describe cm mysql-configmap2
```

---

```
anjali@master:~$ vi mysql-secret2.yml
apiVersion: v1
kind: Secret
metadata:
```

```
  name: mysql-secret2
type: Opaque
data:
  root-password: <base64-encoded-root-password>
  username: <base64-encoded-username>
  password: <base64-encoded-password>
anjali@master:~$ kubectl create -f mysql-secret2.yml
Secret "mysql-secret2" created
anjali@master:~$ kubectl get secret
anjali@master:~$ kubectl describe secret mysql-secret2
```

---

```
anjali@master:~$ vi mysql-deployment2.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment2
spec:
  selector:
    matchLabels:
      app: mysql-2
  template:
    metadata:
      labels:
        app: mysql-2
    spec:
      containers:
      - name: mysql
        image: mysql:latest
        env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
```

```
      secretKeyRef:
        name: mysql-secret2
        key: root-password
  - name: MYSQL_USER
    valueFrom:
      secretKeyRef:
        name: mysql-secret2
        key: username
  - name: MYSQL_PASSWORD
    valueFrom:
      secretKeyRef:
        name: mysql-secret2
        key: password
  - name: MYSQL_DATABASE
    valueFrom:
      configMapKeyRef:
        name: mysql-configmap2
        key: database-name
  ports:
  - containerPort: 3306
```

---

```
anjali@master:~$ kubectl create -f  mysql-deployment2.yml
deployment.apps "mysql-deployment2" created
anjali@master:~$ kubectl get pods
anjali@master:~$ kubectl describe deployment.apps/mysql-deployment2
```

---

```
anjali@master:~/kubernetes_project$ pwd
/home/anjali/kubernetes_project
anjali@master:~/kubernetes_project$ ls
mysql-configmap2.yml  mysql-deployment2.yml  mysql-secret2.yml
anjali@master:~/kubernetes_project$ kubectl get pods
```

```
NAME                          READY   STATUS    RESTARTS     AGE
mysql-deployment2-5fdd5c87c4-nsx6z   1/1     Running   1 (79s ago)   97s
```

---

```
anjali@master:~/kubernetes_project$ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
mysql-deployment2-5fdd5c87c4-d6dk6   1/1     Running   0          113s
anjali@master:~/kubernetes_project$ kubectl exec -it mysql-deployment2-5fdd5c87c4-\
d6dk6 -- bash
bash-4.4# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.33 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| mysqldbb           |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql> exit
Bye
```

```
anjali@master:~/kubernetes_project$ Vi mysql-expose2.yml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  name: mysql-expose2
spec:
  ports:
  - port: 3306
    protocol: TCP
    targetPort: 3306
  selector:
    app: mysql-2
  type: ClusterIP
status:
  loadBalancer: {}
anjali@master:~/kubernetes_project$ kubectl create -f mysql-expose2.yml
anjali@master:~/kubernetes_project$ kubectl get svc

NAME          TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
mysql-expose2   ClusterIP   10.97.89.135   <none>        3306/TCP   48s
anjali@master:~/kubernetes_project$
```

NOTE: You can check mysql outside with in cluster

For this you have to require install mysql-server package and then run this command

```
anjali@master:~/kubernetes_project$ kubectl get svc
NAME          TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
mysql-expose2   ClusterIP   10.97.89.135   <none>        3306/TCP   3m20s
anjali@master:~/kubernetes_project$ mysql -h 10.97.89.135  -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.33 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| mysqldbb           |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.01 sec)
```

➔ Step8 (Create And Setup one more Wordpress Deployment, ConfigMap/Secrets and LoadBalancer Service. link with mysql deployment using environment variable in gip namespace)

```
anjali@master:~/kubernetes_project$ vi wordpress-deployment2.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-deployment2
spec:
  selector:
    matchLabels:
      app: wordpress2
  template:
    metadata:
      labels:
        app: wordpress2
    Spec:
      replicas: 5
      containers:
      - name: wordpress
        image: wordpress:latest
        ports:
        - containerPort: 80
        env:
        - name: WORDPRESS_DB_HOST
          value: mysql-expose2  —--------------------> Check This before launch
        - name: WORDPRESS_DB_USER
          valueFrom:
```

```
      secretKeyRef:
        name: mysql-secret2
        key: username —-------------------> Check This before launch
  - name: WORDPRESS_DB_PASSWORD
    valueFrom:
      secretKeyRef:
        name: mysql-secret2
        key: password  —----------------> Check This before launch
  - name: WORDPRESS_DB_NAME
    valueFrom:
      configMapKeyRef:
        name: mysql-configmap2
        key: database-name  —-------------------> Check This before launch
  volumeMounts:
  - name: wordpress-persistent-storage
    mountPath: /var/www/html
volumes:
- name: wordpress-persistent-storage
  persistentVolumeClaim:
    claimName: pvc06  —-------------------> Check This before launch

anjali@master:~/kubernetes_project$ kubectl create -f wordpress-deployment2.yml
deployment.apps/wordpress-deployment2 created
anjali@master:~/kubernetes_project$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
mysql-deployment2-5fdd5c87c4-5x8j8       1/1     Running   0          8m28s
wordpress-deployment2-775c87c79c-llj5h   1/1     Running   0          6m31s
anjali@master:~/kubernetes_project$
```

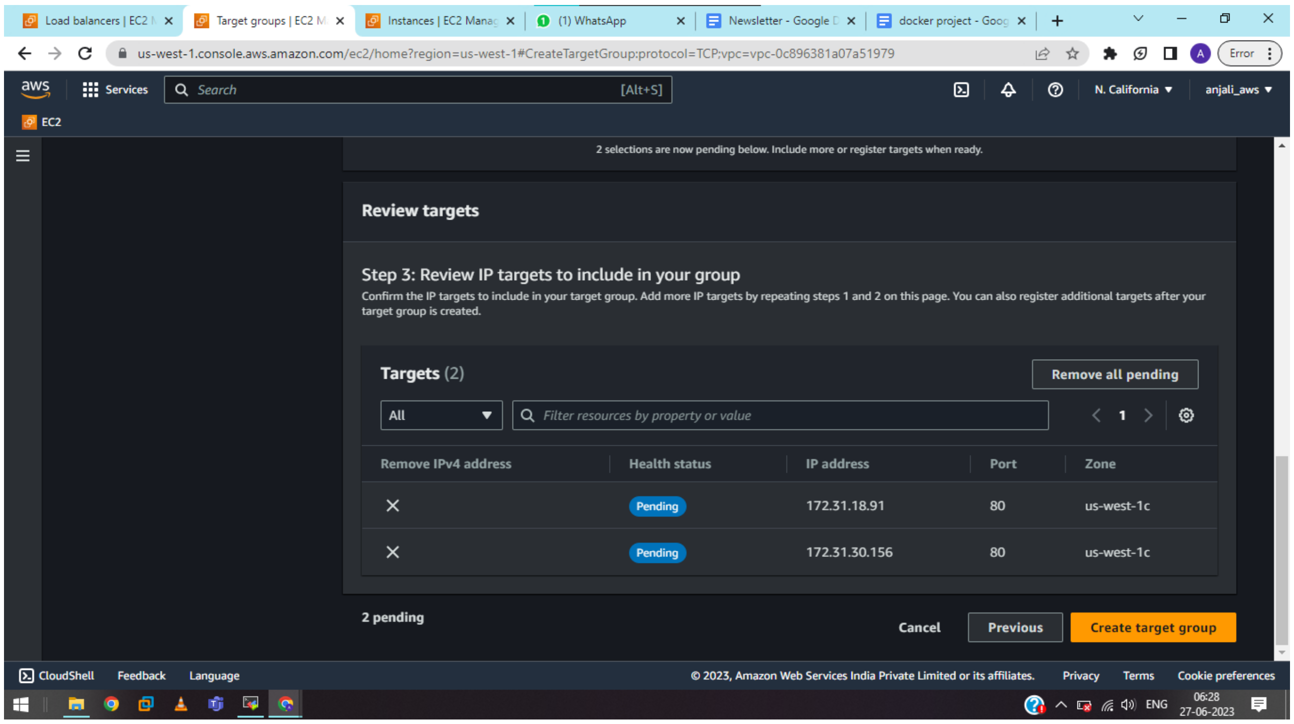
## ➔ Step-9 ( Expose Wordpress Deployment using LoadBalancer Internally )

```
anjali@master:~$ kubectl expose deployment wordpress-deployment2 --port 80 --type LoadBalancer --name=wordpress-expose2

service/wordpress-expose2 exposed

anjali@master:~$
```

## ➔ Step-10 (Configure an AWS NLB to distribute the load requests over Two LoadBalancer Services. )

- We have deployed two applications App1 & App2 among two different namespaces Gip & Grras with two different users Anjali and Rakesh. And make its outer-cluster available using the LoadBalancer Exposing service . Now we will create an AWS NLB Loadbalancer to distribute traffic among LoadBalancer Applications Services.

## ➔ Step-11 Configure AWS NLB:



us-west-1.console.aws.amazon.com/ec2/home?region=us-west-1#CreateTargetGroup:protocol=TCP:vpc=vpc-0c896381a07a51979

2 selections are now pending below. Include more or register targets when ready.

**Review targets**

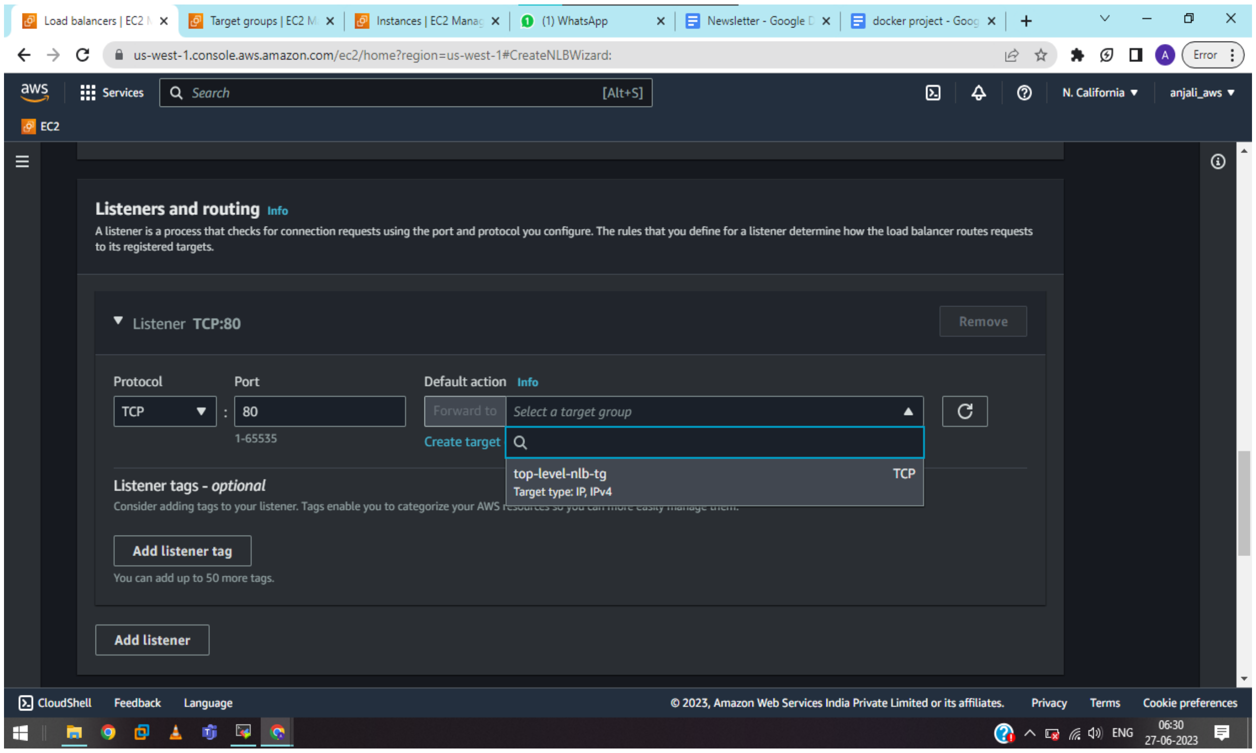
**Step 3: Review IP targets to include in your group**

Confirm the IP targets to include in your target group. Add more IP targets by repeating steps 1 and 2 on this page. You can also register additional targets after your target group is created.

**Targets** (2)

| Remove IPv4 address | Health status | IP address | Port | Zone |
|---|---|---|---|---|
| ✕ | Pending | 172.31.18.91 | 80 | us-west-1c |
| ✕ | Pending | 172.31.30.156 | 80 | us-west-1c |

2 pending


Listeners and routing Info

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

Listener TCP:80

Protocol: TCP

Port: 80 (1-65535)

Default action: Forward to — Select a target group

top-level-nlb-tg — TCP — Target type: IP, IPv4

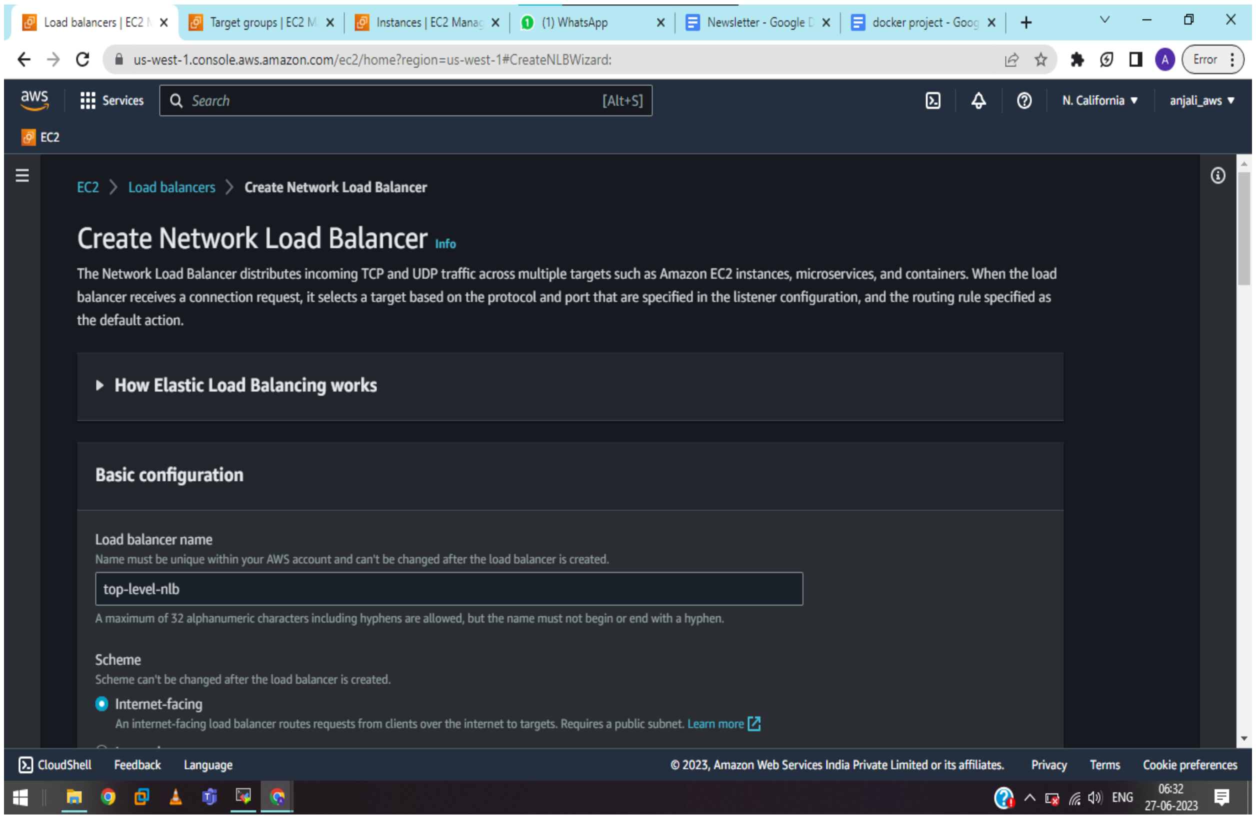
Listener tags - *optional*

Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

Add listener tag

You can add up to 50 more tags.

Add listener


EC2 > Load balancers > Create Network Load Balancer

Create Network Load Balancer Info

The Network Load Balancer distributes incoming TCP and UDP traffic across multiple targets such as Amazon EC2 instances, microservices, and containers. When the load balancer receives a connection request, it selects a target based on the protocol and port that are specified in the listener configuration, and the routing rule specified as the default action.

How Elastic Load Balancing works

Basic configuration

Load balancer name

Name must be unique within your AWS account and can't be changed after the load balancer is created.

top-level-nlb

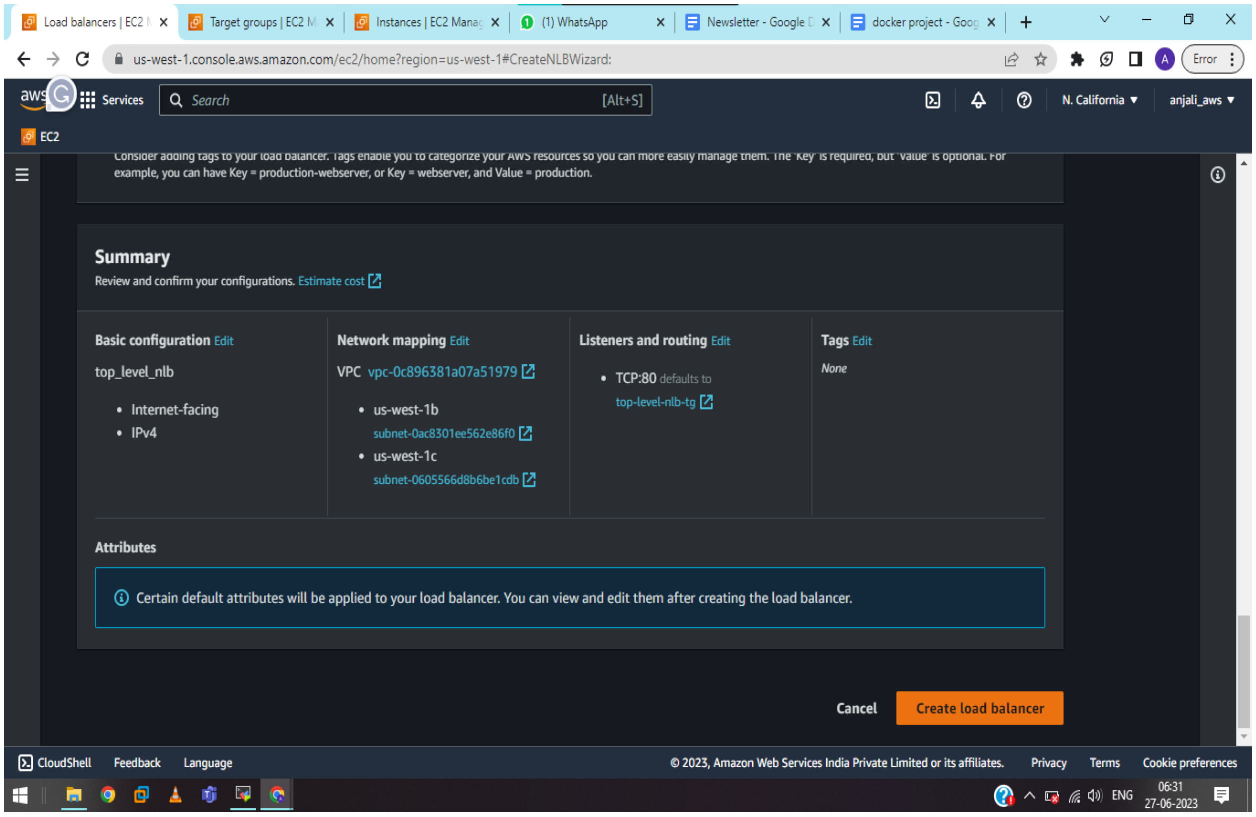
A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Scheme

Scheme can't be changed after the load balancer is created.

● Internet-facing

An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. Learn more



Consider adding tags to your load balancer. Tags enable you to categorize your AWS resources so you can more easily manage them. The 'Key' is required, but 'Value' is optional. For example, you can have Key = production-webserver, or Key = webserver, and Value = production.

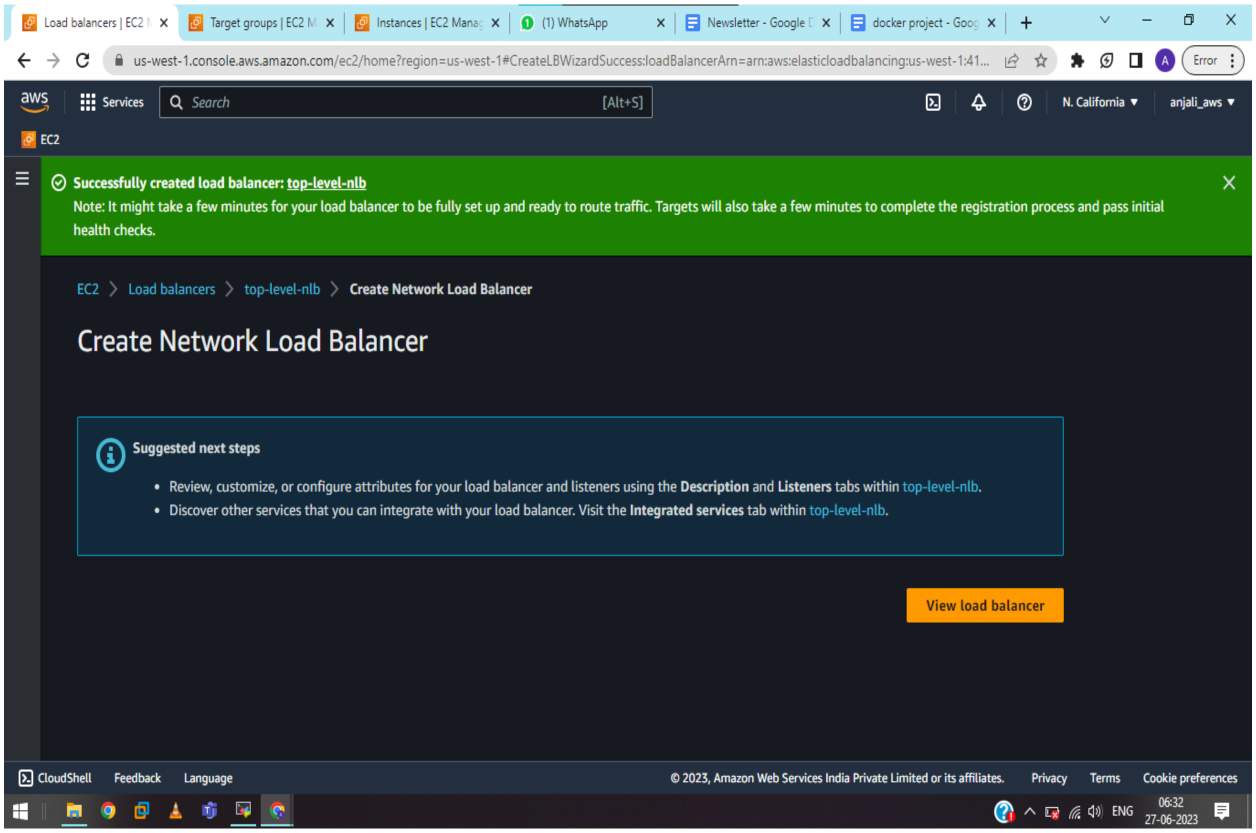
**Summary**

Review and confirm your configurations. Estimate cost

| Basic configuration Edit | Network mapping Edit | Listeners and routing Edit | Tags Edit |
|---|---|---|---|
| top_level_nlb<br>• Internet-facing<br>• IPv4 | VPC vpc-0c896381a07a51979<br>• us-west-1b subnet-0ac8301ee562e86f0<br>• us-west-1c subnet-0605566d8b6be1cdb | • TCP:80 defaults to top-level-nlb-tg | None |

**Attributes**

Certain default attributes will be applied to your load balancer. You can view and edit them after creating the load balancer.



**Successfully created load balancer: top-level-nlb**

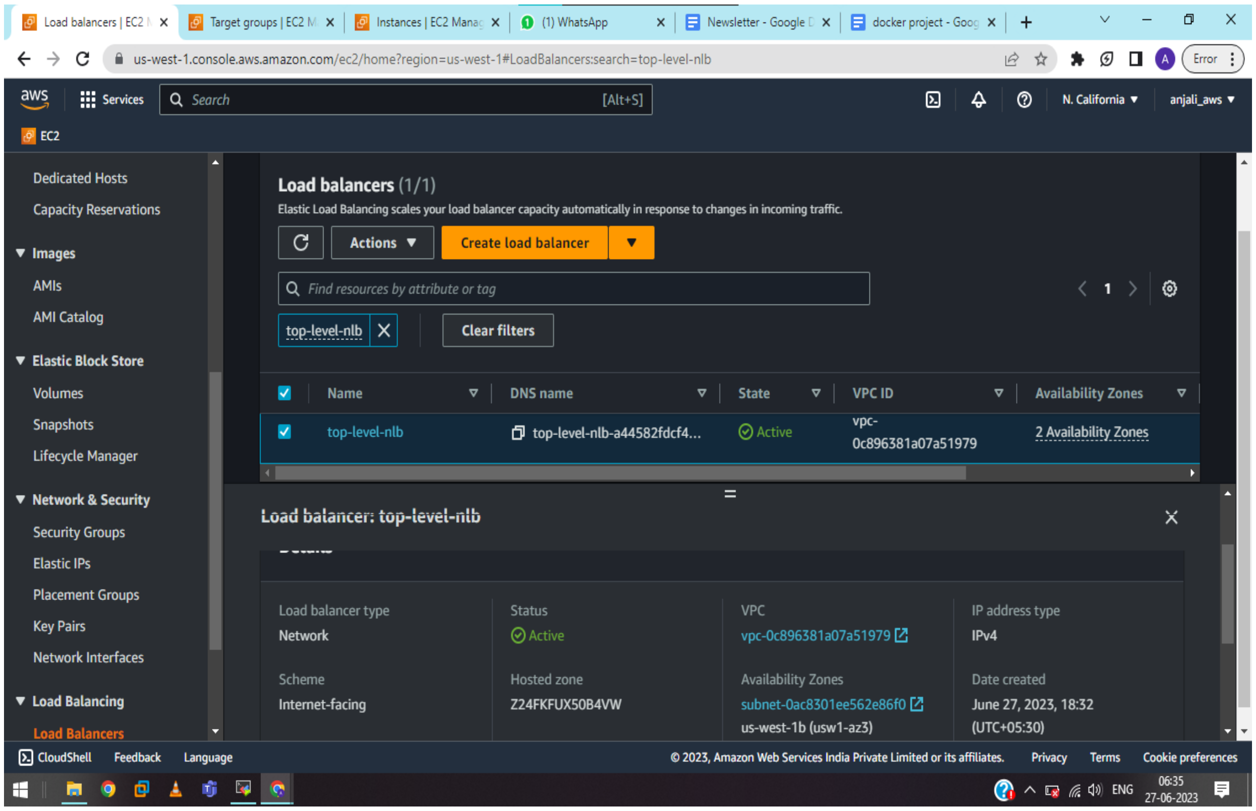
Note: It might take a few minutes for your load balancer to be fully set up and ready to route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks.

EC2 > Load balancers > top-level-nlb > Create Network Load Balancer

**Create Network Load Balancer**

**Suggested next steps**

- Review, customize, or configure attributes for your load balancer and listeners using the **Description** and **Listeners** tabs within top-level-nlb.
- Discover other services that you can integrate with your load balancer. Visit the **Integrated services** tab within top-level-nlb.



Load balancers | EC2 · Target groups | EC2 M · Instances | EC2 Manag · (1) WhatsApp · Newsletter - Google D · docker project - Goog

us-west-1.console.aws.amazon.com/ec2/home?region=us-west-1#LoadBalancers:search=top-level-nlb

**Load balancers** (1/1)

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

| Name | DNS name | State | VPC ID | Availability Zones |
|---|---|---|---|---|
| top-level-nlb | top-level-nlb-a44582fdcf4... | Active | vpc-0c896381a07a51979 | 2 Availability Zones |

**Load balancer: top-level-nlb**

| Load balancer type | Status | VPC | IP address type |
|---|---|---|---|
| Network | Active | vpc-0c896381a07a51979 | IPv4 |
| **Scheme** | **Hosted zone** | **Availability Zones** | **Date created** |
| Internet-facing | Z24FKFUX50B4VW | subnet-0ac8301ee562e86f0 us-west-1b (usw1-az3) | June 27, 2023, 18:32 (UTC+05:30) |

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

ENG 06:35 27-06-2023

## ➔ Step-12 (Access App1 & App2 over NLB DNS address:)

On browser
[http://us-west-1.console.aws.amazon.com/console/home?region=us-west-1/app1](http://us-west-1.console.aws.amazon.com/console/home?region=us-west-1/app1)
[http://us-west-1.console.aws.amazon.com/console/home?region=us-west-1/app1](http://us-west-1.console.aws.amazon.com/console/home?region=us-west-1/app1)