**Deploy the Application Using the Kubernetes Dashboard.**

* **Creating a token and working on a dashboard.**

1. The Dashboard UI is not deployed by default. To deploy it, run the following command:

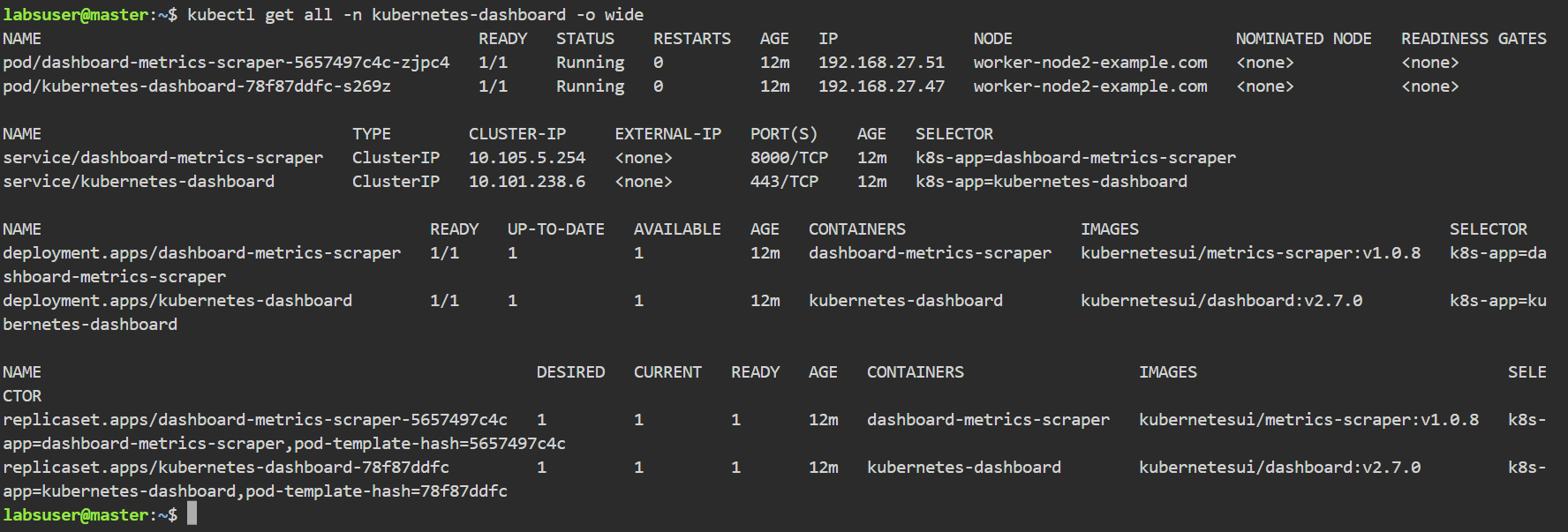
**kubectl apply -f** [**https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml**](https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml)

A screen shot of a computer program

Description automatically generated

1. To verify the dashboard resources, run the following command:

**kubectl get all -n kubernetes-dashboard -o wide**



1. Edit the service ‘kubernetes-dashboard’ and change the service type form ‘ClusterIP’ to ‘NodePort’. To edit the service, run the following command:

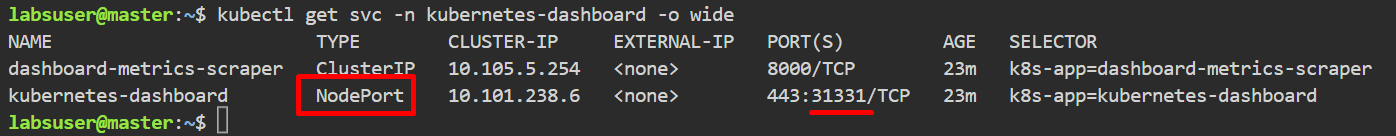
**kubectl edit svc -n kubernetes-dashboard kubernetes-dashboard**

A screenshot of a computer

Description automatically generated

1. To verify the changes done in point 3, run the following command:

**kubectl get svc -n kubernetes-dashboard -o wide**



Type is changed to **NodePort** and NodePort **31331** is being used.

1. To access Kubernetes Dashboard UI , open firefox on master node and open below url

“https://localhost:<<NodePort>>” 🡪 NodePort from point 4.

[**https://localhost:31331**](https://localhost:31331)

Click on Advanced -> Accept Risk and Continue and below page will be displayed.

Select the token and enter the token and sign in.

To generate the token follow below points.

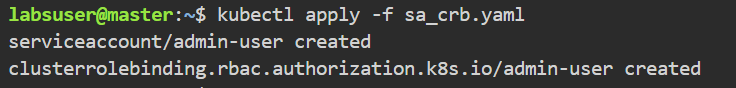
A screenshot of a computer

Description automatically generated

1. To generate the token, create a service account ‘admin-user’ in namespace ‘kubernetes-dashboard’ and perform clusterrolebinding with existing cluster role ‘cluster-admin’. Use attached sa\_crb.yaml file to create the resources.



**kubectl apply -f sa\_crb.yaml**



1. To Create the token for above service account, run following command:

**kubectl -n kubernetes-dashboard create token admin-user**

A black background with many small colored letters

Description automatically generated with medium confidence

1. Copy this token and enter in the Kubernetes dashboard login page and sign in into the dashboard. The dashboard is accessible

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* **Configure the NFS-server.**

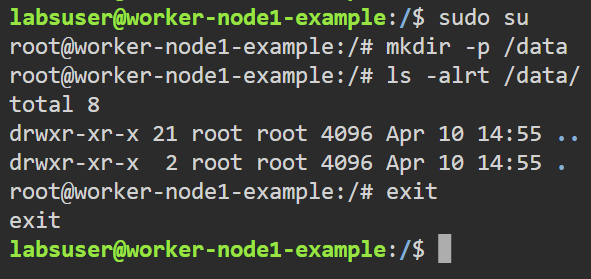
1. To create a directory on NFS server node which can be exported to NFS clients, run the below command on worker node 1.

**sudo su**

**mkdir -p /data**

**ls -alrt /data/**

**exit**



1. Install the NFS kernel server on the machine:

**sudo apt install nfs-kernel-server**

A screen shot of a computer

Description automatically generated

1. To specify which directories on the server should be exported to NFS clients and what level of access those clients should have, open the exports file in the /etc directory:

**sudo vi /etc/exports**



1. Inside the file, append the following code:

**/ data \*(rw,sync,no\_root\_squash)**:A screenshot of a computer program

Description automatically generated

1. To export all shared directories, you have defined in the /etc/exports file, use:

**sudo exportfs -rv**

A screen shot of a computer

Description automatically generated

1. Make the folder publicly accessible by changing its owner user and group:

**sudo chown nobody:nogroup /data/**



1. Assign full permissions to ensure everyone can read, write, and execute files in this directory:

**sudo chmod 777 /data/**



1. Restart the NFS kernel server to apply the changes:

**sudo systemctl restart nfs-kernel-server**



1. To get the IP of worker nodes running the nfs server run below command:

**ip a**

or run **kubectl get nodes -o wide** on master node.

A screenshot of a computer program

Description automatically generated

* **Setting up the NFS client side.**

**Note:** Perform these steps on each worker node intended for sharing.

1. Run the following command to install the NFS common package on worker2:

**sudo apt install nfs-common**

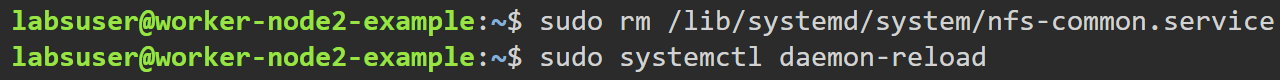
A screenshot of a computer program

Description automatically generated

Execute the following commands to refresh the NFS common service and verify its current status:  
  
 **sudo rm /lib/systemd/system/nfs-common.service  
 sudo systemctl daemon-reload**

**sudo systemctl restart nfs-common**

**sudo systemctl status nfs-common**



A screenshot of a computer program

Description automatically generated

* **Creating and verifying the PV and PVC.**

1. In master node create below wordpress-pv.yaml file to create a NFS PV.

The NFS server IP is the which we noted while installing NFS server on worker1.

apiVersion: v1

kind: PersistentVolume

metadata:

  name: mysql-pv

  labels:

    app: wordpress

    tier: mysql

spec:

  capacity:

    storage: 2Gi

  accessModes:

    - ReadWriteMany

  nfs:

    server: 172.31.26.41

    path: /data





**kubectl apply -f mysql-pv.yaml**

A black background with white text

Description automatically generated

1. To describe the persistent volume created in above step run below commands:

**kubectl get pv mysql-pv**

**kubectl describe pv mysql-pv**

A screenshot of a computer

Description automatically generated

1. Create below mysql-pvc.yaml file to create a persistent volume claim for pv created in above step:

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

  name: mysql-pvc

  labels:

    app: wordpress

spec:

  accessModes:

    - ReadWriteMany

  resources:

    requests:

      storage: 500Mi



**kubectl apply -f mysql-pvc.yaml**



A black background with white text

Description automatically generated

1. To describe the persistent volume claim created in above step run below commands:

**kubectl get pvc mysql-pvc**

**kubectl describe pvc mysql-pvc**

A screenshot of a computer screen

Description automatically generated

**NOTE: This mysql-pvc will be used in mysql deployment.**

1. To create PV and PVC for wordpress deployment use below yaml files.

apiVersion: v1

kind: PersistentVolume

metadata:

  name: wordpress-pv

  labels:

    type: hostpath

spec:

  capacity:

    storage: 2Gi

  accessModes:

    - ReadWriteMany

  persistentVolumeReclaimPolicy: Delete

  hostPath:

    type: DirectoryOrCreate

    path: "/opt/ "

---

kind: PersistentVolumeClaim

apiVersion: v1

metadata:

  name: wordpress-pvc

spec:

  accessModes:

    - ReadWriteMany

  resources:

    requests:

      storage: 500Mi





A black background with white text

Description automatically generated

1. To describe the PV and PVC created in above step, run the following command:

**kubectl describe pv wordpress-pv**

**kubectl describe pvc wordpress-pvc**

A screenshot of a computer program

Description automatically generated

* **Creating a secret for MySQL deployments secret data.**

1. Run the command below to encrypt your password with base64 encoding to make it more secure:

**echo -n 'admin' | base64**

A black background with white text

Description automatically generated

1. Create below mysql-secret.yaml file to create the secret to be used in mysql deployment.

apiVersion: v1

kind: Secret

metadata:

  name: mysql-secret

type: Opaque

data:

  password: YWRtaW4=





A close up of white text

Description automatically generated

1. To describe the secret created in above step, run the below command:

**kubectl get secret mysql-secret**

**kubectl describe secret mysql-secret**

A screen shot of a computer

Description automatically generated

* **Creating a configmap for WordPress deployment.**

1. Create below wordpress-config.yaml file to create the configmap to be used in wordpress deployment.

apiVersion: v1

kind: ConfigMap

metadata:

  name: wordpress-config

data:

  host: wordpress-mysql







1. To describe the configmap created in above step, run the below command:

**kubectl get configmap wordpress-configmap**

**kubectl describe configmap wordpress-configmap**

A screen shot of a computer

Description automatically generated

* **Creating and verifying the Deployments.**

1. To create the mysql deployment use the below mysql-deployment.yml file:

apiVersion:  apps/v1

kind: Deployment

metadata:

  name: wordpress-mysql

  labels:

    app: mysql-wordpress

spec:

  selector:

    matchLabels:

      app: mysql-wordpress

      tier: backend

  strategy:

    type: Recreate

  template:

    metadata:

      labels:

        app: mysql-wordpress

        tier: backend

    spec:

      containers:

      - image: mysql:5.6

        name: mysql

        env:

        - name: MYSQL\_ROOT\_PASSWORD

          valueFrom:

            secretKeyRef:

              name: mysql-secret

              key: password

        ports:

        - containerPort: 3306

          name: mysql

        volumeMounts:

        - name: mysql-persistent-storage

          mountPath: "/var/lib/mysql"

      volumes:

      - name: mysql-persistent-storage

        persistentVolumeClaim:

          claimName: mysql-pvc





A black background with white letters

Description automatically generated

1. To describe the configmap created in above step, run the below command:

**kubectl get deployment wordpress-mysql**

**kubectl describe deployment wordpress-mysql**

A black background with white text

Description automatically generated

A screenshot of a computer program

Description automatically generated

1. To create the wordpress deployment use the below wordpress-deployment.yml file:

apiVersion: apps/v1

kind: Deployment

metadata:

  name: wordpress

  labels:

    app: mysql-wordpress

spec:

  selector:

    matchLabels:

      app: mysql-wordpress

      tier: frontend

  strategy:

    type: Recreate

  template:

    metadata:

      labels:

        app: mysql-wordpress

        tier: frontend

    spec:

      containers:

      - image: wordpress

        name: wordpress

        env:

        - name: WORDPRESS\_DB\_HOST

          valueFrom:

            configMapKeyRef:

              name: wordpress-config

              key: host

        - name: WORDPRESS\_DB\_PASSWORD

          valueFrom:

            secretKeyRef:

              name: mysql-secret

              key: password

        ports:

        - containerPort: 80

          name: wordpress

        volumeMounts:

        - name: wordpress-persistent-storage

          mountPath: "/var/www/html"

      volumes:

      - name: wordpress-persistent-storage

        persistentVolumeClaim:

          claimName: wordpress-pvc



1. To describe the configmap created in above step, run the below command:

**kubectl get deployment wordpress**

**kubectl describe deployment wordpress**

A black background with white text

Description automatically generated

A screenshot of a computer program

Description automatically generated

* **Creating and verifying the service.**

1. Create below mysql-service.yaml file to create the service to expose the mysql deployment.

apiVersion: v1

kind: Service

metadata:

  name: wordpress-mysql

  labels:

    app: mysql-wordpress

spec:

  ports:

    - port: 3306

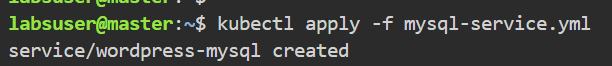
  selector:

    app: mysql-wordpress

    tier: backend

****





1. To describe the mysql service created in above step, run the below command:

**kubectl get svc wordpress-mysql**

**kubectl describe svc wordpress-mysql**

A screenshot of a computer

Description automatically generated

1. Create below wordpress-service.yaml file to create the service to expose the wordpress deployment. Type is NodePort so we can access the wordpress service on **http://localhost:nodeport**.

apiVersion: v1

kind: Service

metadata:

  name: wordpress

  labels:

    app: mysql-wordpress

spec:

  ports:

    - port: 80

  selector:

    app: mysql-wordpress

    tier: frontend

  type: NodePort







1. To describe the mysql service created in above step, run the below command:

**kubectl get svc wordpress**

**kubectl describe svc wordpress**

A screenshot of a computer

Description automatically generated

* **Access the WordPress application using the NodePort.**

1. Use the command below to get the Nodeport and access the WordPress application:

**kubectl get services**

A screenshot of a computer program

Description automatically generated

1. On the master node, open the Firefox browser, and then open the url http://localhost:<node\_port> to access the WordPress application

[**http://localhost:30175**](http://localhost:30175)

A screenshot of a computer

Description automatically generated

1. Once the WordPress application opens, choose the language as English and then click Continue , provide the website-related information, and proceed with the installation.

A screenshot of a computer

Description automatically generated

1. Once the installation is completed login into WordPress Application.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated