Elasticsearch Installation & Setup With Helm

1. Add the Helm Repository

```
helm repo add elastic https://helm.elastic.co
```

2. Update Helm Repositories

```
helm repo update
```

3. Get the Default Helm `values.yaml` File

```
helm show values elastic/elasticsearch > values.yaml
```

sudo nano values.yaml (If you want to modify or change some settings)

4. Install Elasticsearch

```
helm install elasticsearch elastic/elasticsearch -f elasticse
arch-values.yaml --namespace elasticsearch --create-namespace
```

Display the pods with

```
kubectl get pod -n elasticsearch -w
```

The pods are in pending status and that's because we haven't bound them to a persistence volume claim

5- Create Storage Class

- mkdir -p ~/k8s-manifests/storage0
- cd ~/k8s-manifests/storage

- nano storageclass.yaml (Copy Paste Storage Class)

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: local-storage
provisioner: kubernetes.io/no-provisioner
volumeBindingMode: WaitForFirstConsumer
```

- Apply storage Class

```
kubectl apply -f storageclass.yaml
```

Check PVC it should be created by default

```
kubectl get pvc --all-namespaces
```

If the PVCs is not created you have to create them manually

- cd ~/k8s-manifests/pv
- nano elasticsearch-pv-0.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: elasticsearch-pvc
  namespace: elasticsearch   # Ensure this matches your names
pace
spec:
  accessModes:
    - ReadWriteOnce          # Choose the appropriate access
mode
  resources:
    requests:
      storage: 10Gi          # Adjust the storage size as nee
```

```
ded
  storageClassName: standard  # Adjust the storage class if n
eeded
```

- Apply the Yaml File

```
kubectl apply -f pvc.yaml
```

Create PV

- cd ~/k8s-manifests/pv
- nano elasticsearch-pv-0.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: elasticsearch-master-pv-0
spec:
  capacity:
    storage: 30Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: local-storage # Change this with the stor
age class name
  local:
    path: /mnt/data/elasticsearch-master-0 # This path has to
be add to the values.yaml
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
```

```
          - master-node  # Change this with your actual node
name
  claimRef:
    namespace: elasticsearch # Change this to match your name
space
    name: elasticsearch-master-elasticsearch-master-0 # Chang
e this to match the pod name of what PV is being created for
```

- Apply the PV class

```
kubectl apply -f elasticsearch-master-pv-0.yaml
```

- Create Directory which we specify for the PV and PVC to work

```
sudo mkdir -p /mnt/data/elasticsearch-master-0
sudo chown -R 1000:1000 /mnt/data/elasticsearch-master-0
sudo chown -R 1000:1000 /mnt/data/elasticsearch-master-0
```

- Check The bounding

```
kubectl get pv elasticsearch-master-pv-0
```

If Cant bound the status shows Released then Delete the PV

```
kubectl delete pv elasticsearch-master-pv-0
kubectl apply -f elasticsearch-master-pv-0.yaml
```

Repeate the steps to create PV for each pod

Modify the Values.yaml File

Under esConfig add the default corresponding container path of the path we added on the PV class

```
esConfig:
  elasticsearch.yml: |
    path.data: /usr/share/elasticsearch/data
   # coressponding mounted path inside a container of the phi
scal path we added on the PV class "/mnt/data/elasticsearch-m
aster-0"
```

Under the Resources section make sure the heap size is smaller than the total memory allocated to the container to avoid out-of-memory (OOM) errors.

```
esJavaOpts: "-Xmx2g -Xms2g" # Heap Size 2g
resources:
  requests:
    cpu: "1500m"
    memory: "3Gi"
  limits:
    cpu: "1500m"
    memory: "3Gi"
```

Add this section to grant permission for the Elasticsearch pod to access its storage class correctly

```
extraInitContainers:
  - name: fix-permissions
    image: busybox
    command: ['sh', '-c', 'chown -R 1000:1000 /usr/share/elas
ticsearch/data']
    securityContext:
      runAsUser: 0
    volumeMounts:
    - name: elasticsearch-master
      mountPath: /usr/share/elasticsearch/data
```

Add this section to Specify the local storage class to be used

```
volumeClaimTemplate:
  accessModes: [ "ReadWriteOnce" ]
  storageClassName: "local-storage"
  resources:
    requests:
      storage: 30Gi
```

Trun the antiAffinity to be soft to allow pods to run all at the same time

```
antiAffinity: "soft"
```

If you want to use upgraded version of docker image, simply change the version of the image

```
image: "docker.elastic.co/elasticsearch/elasticsearch"
imageTag: "8.15.0"
```

Save and exist

Apply the changes

```
helm install elasticsearch elastic/elasticsearch -f ./values-original.yaml --namespace elasticsearch
```

After this monitor the pods they should be ready and running

Troubleshooting

PV not bounded to PVC

If still getting PV & PVC issue try to patch the PVC to the storage class with

```
kubectl patch pvc elasticsearch-master-elasticsearch-master-0 --namespace=elasticsearch -p '{"spec": {"storageClassName": "local-storage"}}'
```

Repeat this for each pod

Flannel CrashLoopBackOff Issue (pod CIDR not assigned)

If Pod Status is CrashLoopBackOff

- Check Logs of the pod (If the error is)

```
E0825 06:36:39.232128       1 main.go:343] Error registering
network: failed to acquire lease: node "master-node" pod CIDR
not assigned
```

Solution

1. **Check PodCIDR Assignment**:

Run the following command to describe the node and check for the `PodCIDR` field:

```
kubectl describe node master-node
```

In the output, look for the **PodCIDR** field to see if it's missing or incorrectly assigned.

2. **Verify kube-apiserver Pod Network CIDR**:

Check the `--pod-network-cidr` setting in the Kubernetes API server manifest:

```
sudo cat /etc/kubernetes/manifests/kube-apiserver.yaml
```

Look for the `--pod-network-cidr` flag in the configuration. **If this flag is missing**, follow the steps below to reset and reinitialize the cluster.

3. **Reset and Reinitialize the Cluster** (if `--pod-network-cidr` is missing):

Run the following commands to reset and reinitialize the cluster with the appropriate Pod CIDR:

```
sudo kubeadm reset
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
```

The `10.244.0.0/16` CIDR is typically used for Flannel. If you are using a different CNI (Container Network Interface) plugin, adjust the CIDR accordingly.

4.
Set Up kubectl for the Admin User:
After reinitializing, configure
`kubectl` for the admin user:

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

5. Reapply the Flannel Network Plugin

Reapply the Flannel network plugin to ensure proper network setup:

```
kubectl apply -f https://raw.githubusercontent.com/coreos/fla
nnel/master/Documentation/kube-flannel.yml
```

6. **Verify the PodCIDR Assignment**:

After reapplying Flannel, verify that the **PodCIDR** is assigned to the node:

```
kubectl describe node master-node
```

You should see a `PodCIDR` field with the assigned range.

7. Check the Status of the Flannel Pod

Finally, verify that the Flannel pod is running correctly:

```
kubectl get pods --namespace=kube-flannel
```

Node is Bound but Pod is Still Pending (**FailedScheduling)**

If running the following command

```
kubectl describe pod elasticsearch-master-0 --namespace=defau
lt
```

Shows the following warning:

```
Warning  FailedScheduling  2m19s  default-scheduler  0/1 node
s are available: 1 node(s) had untolerated taint {node-role.k
ubernetes.io/control-plane: }.
```

This indicates that the **control-plane node** has a taint preventing the pod from being scheduled on that node.

Solution Steps:

1. **Check Node Status**:
   a. First, check the status of your nodes using the command:

   ```
   kubectl get nodes
   ```

   This will list all the nodes and their roles, along with any taints applied to them.

2. Remove the NoSchedule Taint on Control-Plane Node
   a. To allow pods to be scheduled on the control-plane (master) node, remove the `NoSchedule` taint:

   ```
   kubectl taint nodes master-node node-role.kubernetes.io/co
   ntrol-plane:NoSchedule-
   ```

   This command removes the `NoSchedule` taint from the control-plane node, allowing it to schedule non-critical pods.

3. **Add Tolerations to the `values.yaml` File**:
   If you want to allow Elasticsearch pods to be scheduled on tainted nodes (like the control-plane), you need to add
   tolerations to your Helm chart's `values.yaml` file

Add the following section to your `values.yaml` file:

```
tolerations:
  - key: "node-role.kubernetes.io/control-plane"
```

```
    operator: "Exists"
    effect: "NoSchedule"
```

This tells Kubernetes to tolerate the `NoSchedule` taint on the control-plane node and allow scheduling of Elasticsearch pods on that node.

Reapply the Helm Chart (if necessary)

```
helm upgrade elasticsearch elastic/elasticsearch -f values.ya
ml --namespace=default
```

Helm Upgrade Fails for Elasticsearch StatefulSet

When attempting to upgrade the Elasticsearch Helm chart using the following command:

```
helm upgrade elasticsearch elastic/elasticsearch -f ./values.
yaml --namespace elasticsearch
```

You encounter the error

```
Error: UPGRADE FAILED: cannot patch "elasticsearch-master" wi
th kind StatefulSet: StatefulSet.apps "elasticsearch-master"
is invalid: spec: Forbidden: updates to statefulset spec for
fields other than 'replicas', 'ordinals', 'template', 'update
Strategy', 'persistentVolumeClaimRetentionPolicy' and 'minRea
dySeconds' are forbidden
```

This error occurs because Kubernetes **StatefulSets** restrict changes to certain fields in their specification. Fields such as `volumeClaimTemplates` cannot be updated directly in an existing StatefulSet.

Solution

```
kubectl delete statefulset elasticsearch-master --namespace e
lasticsearch
```

```
helm upgrade --install elasticsearch elastic/elasticsearch -f
./values.yaml --namespace elasticsearch
```