Ingress-Controller and StatefulSet Resource

[Edition 1]

[Last Update 200911]

For any issues/help contact : ***support@k21academy.com***

Contents

1 INTRODUCTION

Ingress-Controller

In order for the Ingress resource to work, the cluster must have an ingress controller running.

Unlike other types of controllers which run as part of the kube-controller-manager binary, Ingress controllers are not started automatically with a cluster. Use this page to choose the ingress controller implementation that best fits your cluster.

StatefulSets

It is the workload API object used to manage stateful applications.

Manages the deployment and scaling of a set of Pods, *and provides guarantees about the ordering and uniqueness* of these Pods.

Like a Deployment, a StatefulSet manages Pods that are based on an identical container spec. Unlike a Deployment, a StatefulSet maintains a sticky identity for each of their Pods. These pods are created from the same spec, but are not interchangeable: each has a persistent identifier that it maintains across any rescheduling.

This guide Covers:

- Advanced Routing with Ingress-Controller
- Deploying and Managing a StatefulSet Resource

2 DOCUMENTATION

2.1 Kubernetes Documentation

1. Ingress Controllers
   https://kubernetes.io/docs/concepts/services-networking/ingress-controllers
2. StatefulSets
   https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

2.2 Linux Commands and VIM Commands

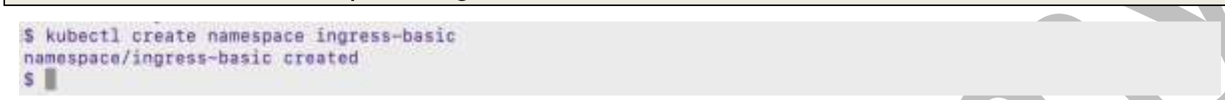
1. Basic Linux Commands
   https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners
   https://www.hostinger.in/tutorials/linux-commands
2. Basic VIM Commands
   https://coderwall.com/p/adv71w/basic-vim-commands-for-getting-started
3. Popular VIM Commands
   https://www.keycdn.com/blog/vim-commands

3 ADVANCED ROUTING WITH INGRESS-CONTROLLER

3.1 Deploying NGINX Ingress Controller using helm chart

1. Create a namespace for your ingress resources

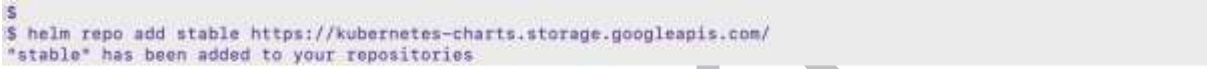
```
$ kubectl create namespace ingress-basic
```



```
$ kubectl create namespace ingress-basic
namespace/ingress-basic created
$ 
```

2. Add the official stable repository

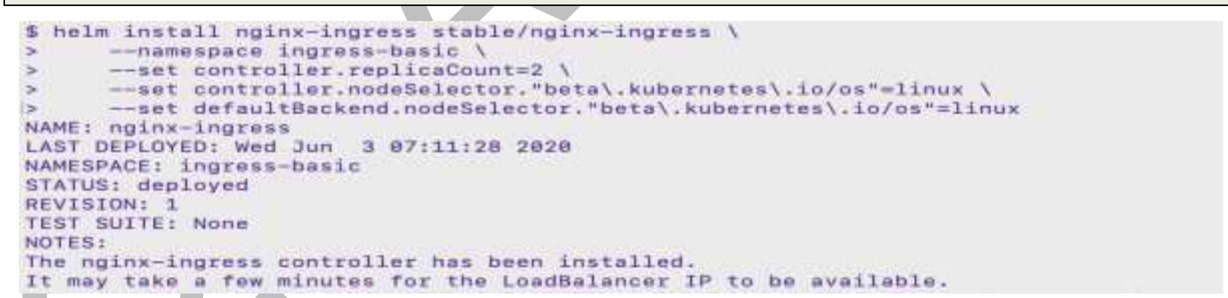
```
$ helm repo add stable https://kubernetes-charts.storage.googleapis.com/
```



```
$
$ helm repo add stable https://kubernetes-charts.storage.googleapis.com/
"stable" has been added to your repositories
```

3. Use Helm to deploy an NGINX ingress controller

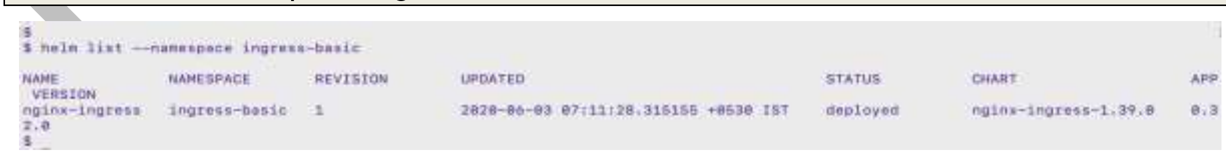
```
$ helm install nginx-ingress stable/nginx-ingress \
    --namespace ingress-basic \
    --set controller.replicaCount=2 \
    --set controller.nodeSelector."beta\.kubernetes\.io/os"=linux \
    --set defaultBackend.nodeSelector."beta\.kubernetes\.io/os"=linux
```



```
$ helm install nginx-ingress stable/nginx-ingress \
>     --namespace ingress-basic \
>     --set controller.replicaCount=2 \
>     --set controller.nodeSelector."beta\.kubernetes\.io/os"=linux \
>     --set defaultBackend.nodeSelector."beta\.kubernetes\.io/os"=linux
NAME: nginx-ingress
LAST DEPLOYED: Wed Jun  3 07:11:28 2020
NAMESPACE: ingress-basic
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The nginx-ingress controller has been installed.
It may take a few minutes for the LoadBalancer IP to be available.
```

4. Verify the helm chart is installed

```
$ helm list --namespace ingress-basic
```

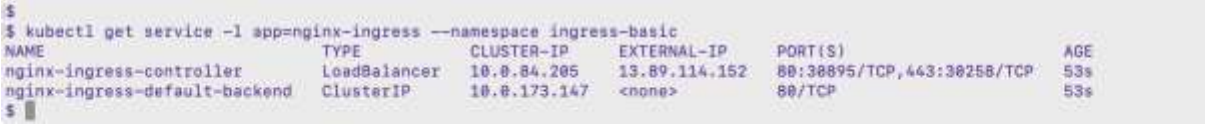


```
$
$ helm list --namespace ingress-basic

NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART                   APP
 VERSION
nginx-ingress   ingress-basic   1               2020-06-03 07:11:28.316155 +0530 IST    deployed        nginx-ingress-1.39.0    0.3
2.0
$ _
```

5. Verify that the load balancer service is created for the NGINX ingress controller and a dynamic public IP address is assigned to it

```
$ kubectl get service -l app=nginx-ingress --namespace ingress-basic
```

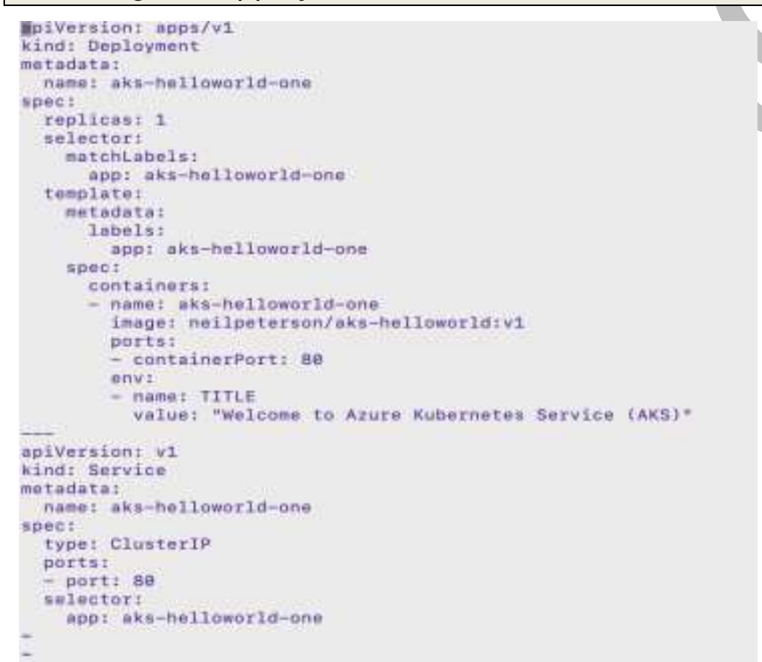


```
$
$ kubectl get service -l app=nginx-ingress --namespace ingress-basic
NAME                           TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                       AGE
nginx-ingress-controller       LoadBalancer   10.0.84.205    13.89.114.152   80:30895/TCP,443:30258/TCP   53s
nginx-ingress-default-backend  ClusterIP      10.0.173.147   <none>          80/TCP                        53s
$ 
```

3.2 Creating simple demo applications

1. View the content of ingress-app1.yaml file and see the definition of first application and its service in the file

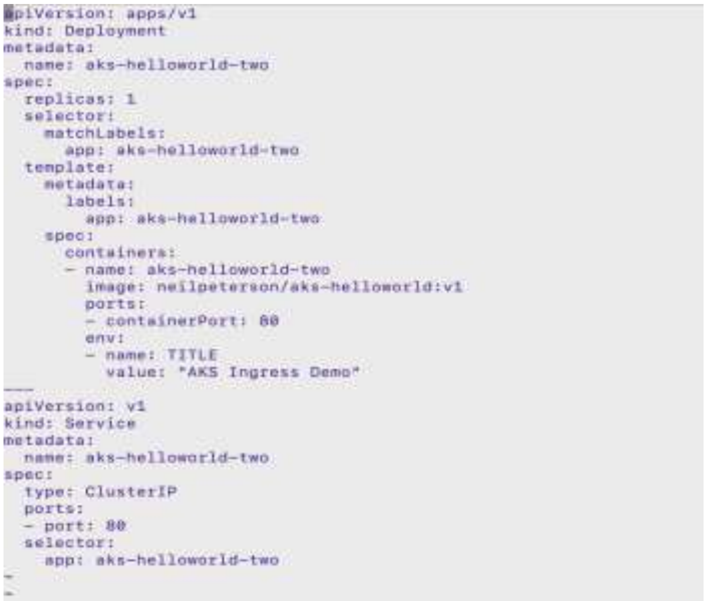
```
$ vim ingress-app1.yaml
```



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: aks-helloworld-one
spec:
  replicas: 1
  selector:
    matchLabels:
      app: aks-helloworld-one
  template:
    metadata:
      labels:
        app: aks-helloworld-one
    spec:
      containers:
      - name: aks-helloworld-one
        image: neilpeterson/aks-helloworld:v1
        ports:
        - containerPort: 80
        env:
        - name: TITLE
          value: "Welcome to Azure Kubernetes Service (AKS)"
---
apiVersion: v1
kind: Service
metadata:
  name: aks-helloworld-one
spec:
  type: ClusterIP
  ports:
  - port: 80
  selector:
    app: aks-helloworld-one
~
~
```

2. View the content of ingress-app2.yaml file and see the definition of second application and its service in the file

```
$ vim ingress-app2.yaml
```

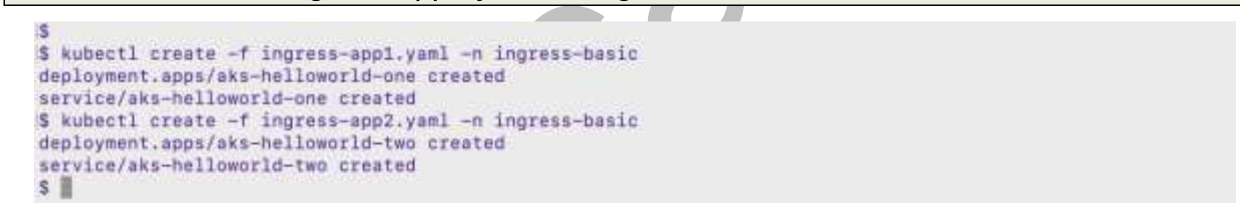


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: aks-helloworld-two
spec:
  replicas: 1
  selector:
    matchLabels:
      app: aks-helloworld-two
  template:
    metadata:
      labels:
        app: aks-helloworld-two
    spec:
      containers:
      - name: aks-helloworld-two
        image: neilpeterson/aks-helloworld:v1
        ports:
        - containerPort: 80
        env:
        - name: TITLE
          value: "AKS Ingress Demo"
---
apiVersion: v1
kind: Service
metadata:
  name: aks-helloworld-two
spec:
  type: ClusterIP
  ports:
  - port: 80
  selector:
    app: aks-helloworld-two
~
~
```

3. Create the deployment and services resources from both the files created above:

```
$ kubectl create -f ingress-app1.yaml -n ingress-basic

$ kubectl create -f ingress-app2.yaml -n ingress-basic
```

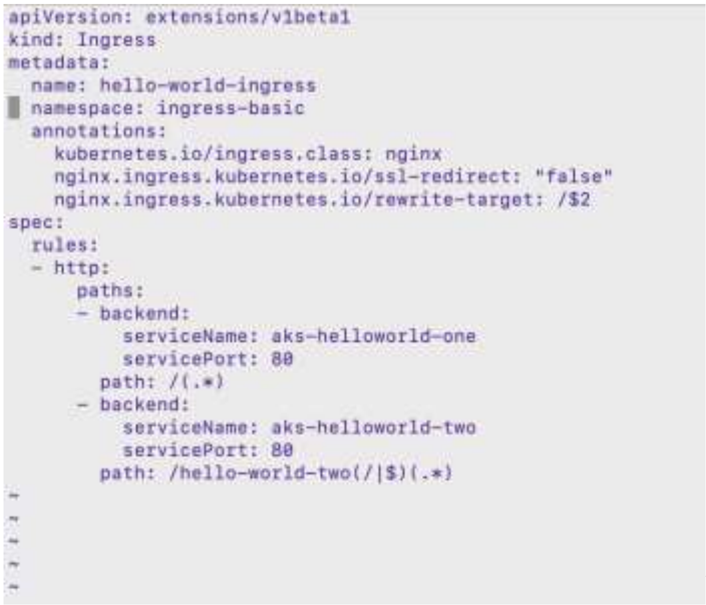


```
$
$ kubectl create -f ingress-app1.yaml -n ingress-basic
deployment.apps/aks-helloworld-one created
service/aks-helloworld-one created
$ kubectl create -f ingress-app2.yaml -n ingress-basic
deployment.apps/aks-helloworld-two created
service/aks-helloworld-two created
$ 
```

3.3 Create Ingress Route to route traffic to both the running applications

1. View the ingress-route.yaml file and see the rules defined in the file to route the traffic to both the applications

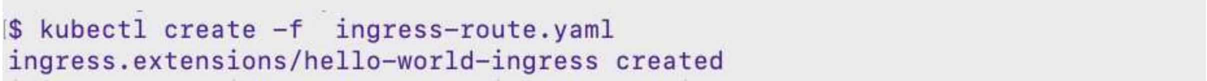
```
$ vim ingress-route.yaml
```



```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: hello-world-ingress
  namespace: ingress-basic
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/ssl-redirect: "false"
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  rules:
  - http:
      paths:
      - backend:
          serviceName: aks-helloworld-one
          servicePort: 80
        path: /(.*)
      - backend:
          serviceName: aks-helloworld-two
          servicePort: 80
        path: /hello-world-two(/|$)(.*)
~
~
~
~
~
```

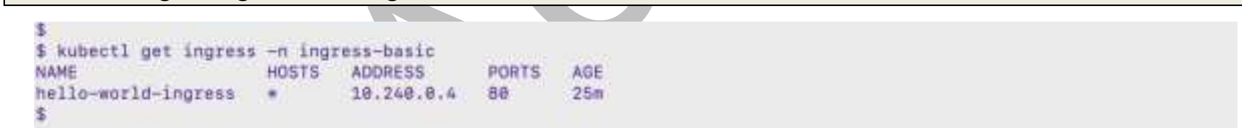
2. Create the ingress resource from ingress-route.yaml and verify using kubectl get command

```
$ kubectl create -f  ingress-route.yaml -n ingress-basic
```



```
$ kubectl create -f  ingress-route.yaml
ingress.extensions/hello-world-ingress created
```

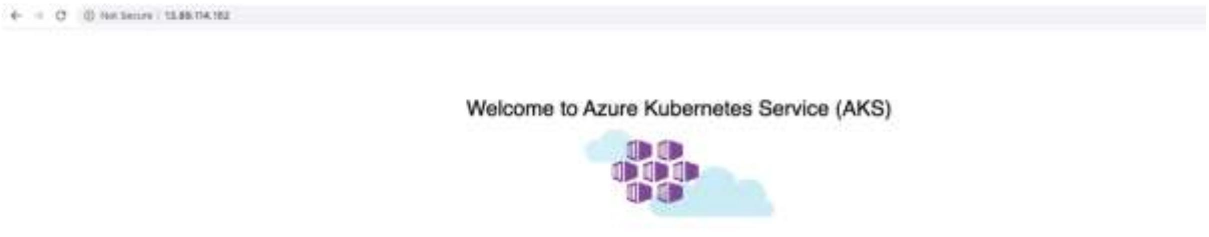
```
$ kubectl get ingress -n ingress-basic
```



```
$
$ kubectl get ingress -n ingress-basic
NAME                  HOSTS   ADDRESS      PORTS   AGE
hello-world-ingress   *       10.240.0.4   80      25m
$
```

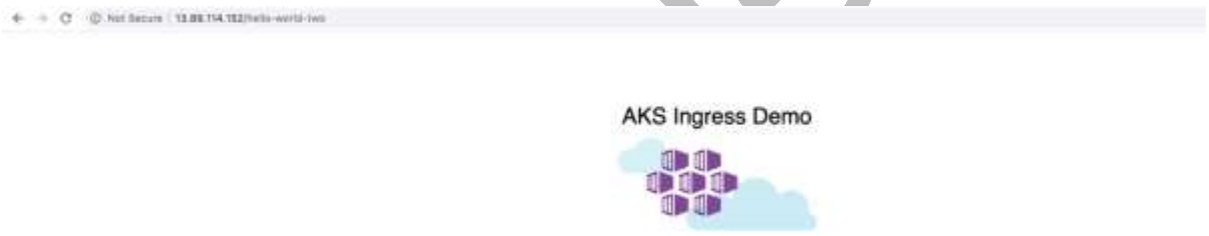
3.4 Testing the ingress controller routes correctly to both the application

1. Open a web browser to the IP address of your NGINX ingress controller, *EXTERNAL_IP*. The first demo application should be displayed in the web browser,


Not Secure | 13.88.114.182

Welcome to Azure Kubernetes Service (AKS)

2. Open a web browser to the IP address of your NGINX ingress controller with /hello-world-two path, EXTERNAL_IP /hello-world-two path. The second demo application should be displayed in the web browser,


Not Secure | 13.88.114.182/hello-world-two

AKS Ingress Demo

3.5 Clean up resources created in this lab exercise

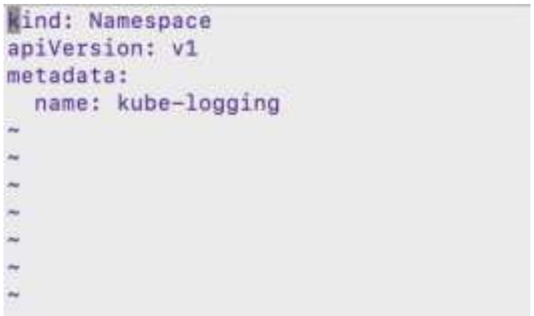
```
$ helm uninstall nginx-ingress --namespace ingress-basic
$ kubectl delete namespace ingress-basic
```

4 DEPLOYING AND MANAGING A STATEFULSET RESOURCE

4.1 Creating Logging namespace

1. Viewing the contents of namespace.yaml file to create kube-logging namespace

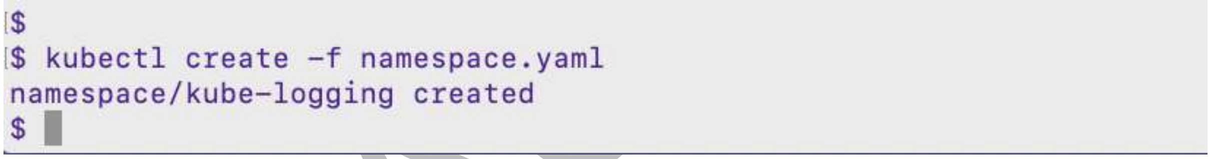
```
$ vim namespace.yaml
```



```
kind: Namespace
apiVersion: v1
metadata:
  name: kube-logging
~
~
~
~
~
~
~
```

2. Creating namespace from above file

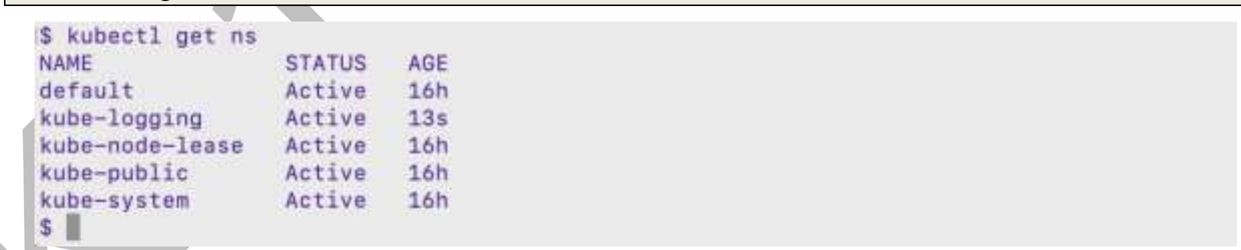
```
$ kubectl create -f namespace.yaml
```



```
$
$ kubectl create -f namespace.yaml
namespace/kube-logging created
$ 
```

3. Confirm that the Namespace was successfully created by listing all the namespace present in the cluster

```
$ kubectl get ns
```

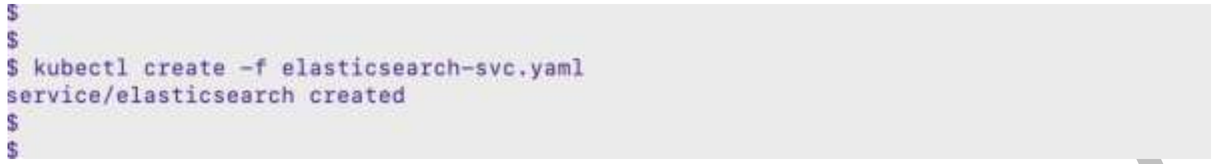


```
$ kubectl get ns
NAME              STATUS   AGE
default           Active   16h
kube-logging      Active   13s
kube-node-lease   Active   16h
kube-public       Active   16h
kube-system       Active   16h
$ 
```

4.2 Setting up Elasticsearch application

1. Create the Elasticsearch StatefulSet using elasticsearch-stfullset.yaml file. Run through the content and create the resource

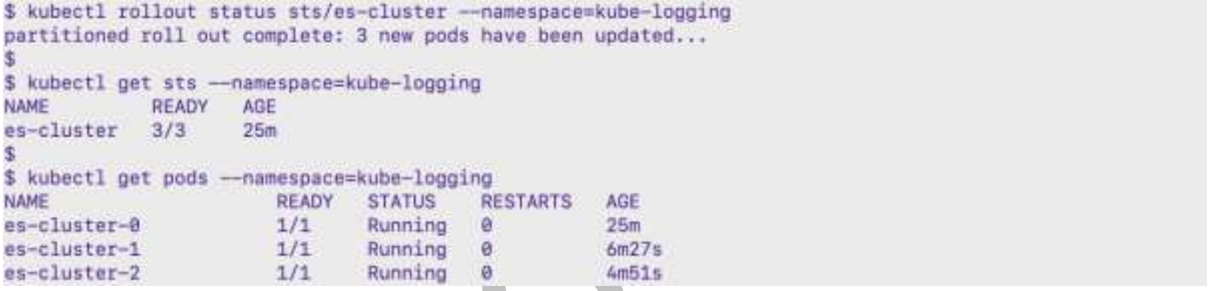
```
$ vim elasticsearch-stfullset.yaml
$ kubectl create -f elasticsearch-stfullset.yaml
```



```
$
$
$ kubectl create -f elasticsearch-svc.yaml
service/elasticsearch created
$
$
```

2. Verify the creation of StatefulSet Elasticsearch pods. monitor the StatefulSet as it is rolled out using kubectl rollout status

```
$ kubectl rollout status sts/es-cluster --namespace=kube-logging
$ kubectl get sts --namespace=kube-logging
$ kubectl get pods --namespace=kube-logging
```



```
$ kubectl rollout status sts/es-cluster --namespace=kube-logging
partitioned roll out complete: 3 new pods have been updated...
$
$ kubectl get sts --namespace=kube-logging
NAME         READY   AGE
es-cluster   3/3     25m
$
$ kubectl get pods --namespace=kube-logging
NAME                        READY   STATUS    RESTARTS   AGE
es-cluster-0                1/1     Running   0          25m
es-cluster-1                1/1     Running   0          6m27s
es-cluster-2                1/1     Running   0          4m51s
```

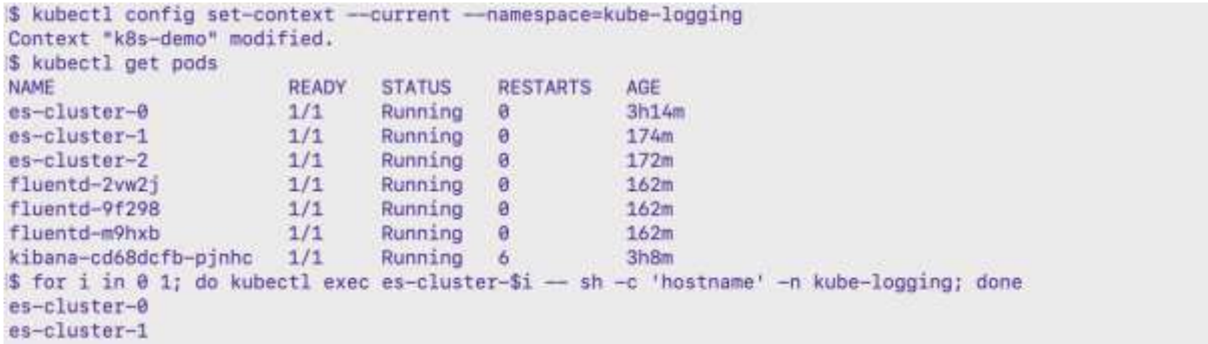
4.3 Pods in a StatefulSet

1. Pods in a StatefulSet have a unique ordinal index and a stable network identity.

   Each Pod has a stable hostname based on its ordinal index. Use kubectl exec to execute the hostname command in each Pod. Let's examine the pods

```
$ kubectl config set-context --current --namespace=kube-logging
$ kubectl get pods
```

```
for i in 0 1 2; do kubectl exec es-cluster-$i -- sh -c 'hostname'; done
```

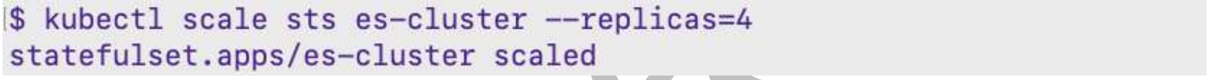


```
$ kubectl config set-context --current --namespace=kube-logging
Context "k8s-demo" modified.
$ kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
es-cluster-0             1/1     Running   0          3h14m
es-cluster-1             1/1     Running   0          174m
es-cluster-2             1/1     Running   0          172m
fluentd-2vw2j            1/1     Running   0          162m
fluentd-9f298            1/1     Running   0          162m
fluentd-m9hxb            1/1     Running   0          162m
kibana-cd68dcfb-pjnhc    1/1     Running   6          3h8m
$ for i in 0 1; do kubectl exec es-cluster-$i -- sh -c 'hostname' -n kube-logging; done
es-cluster-0
es-cluster-1
```

4.4 Scaling up and down a Statefulset object

1. Scaling up the replicas from 3 to 4 for sts es-cluster. The StatefulSet controller scales the number of replicas.

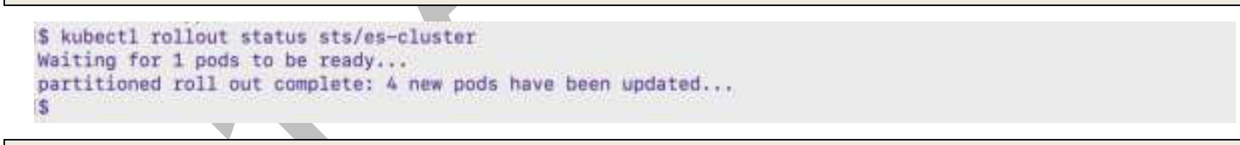
```
$ kubectl scale sts es-cluster --replicas=4
```



```
$ kubectl scale sts es-cluster --replicas=4
statefulset.apps/es-cluster scaled
```

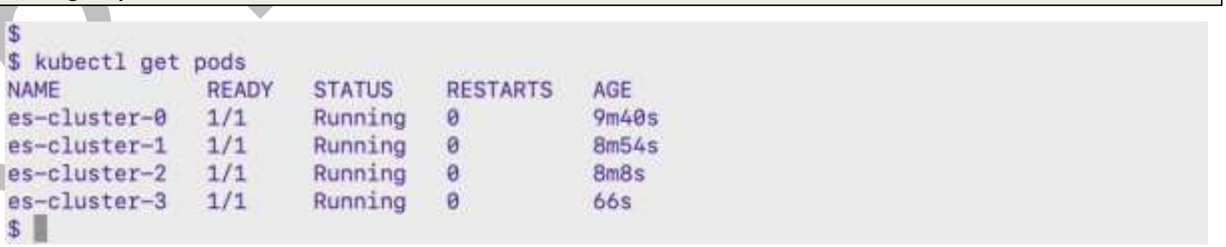
2. The StatefulSet controller creates each Pod sequentially with respect to its ordinal index, and it waits for each Pod's predecessor to be Running and Ready before launching the subsequent Pod

```
$ kubectl rollout status sts/es-cluster
```



```
$ kubectl rollout status sts/es-cluster
Waiting for 1 pods to be ready...
partitioned roll out complete: 4 new pods have been updated...
$
```

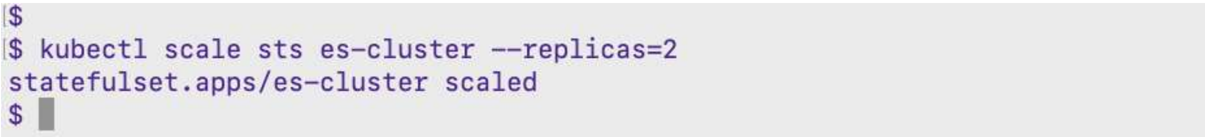
```
$ kubectl get pods
```



```
$
$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
es-cluster-0   1/1     Running   0          9m40s
es-cluster-1   1/1     Running   0          8m54s
es-cluster-2   1/1     Running   0          8m8s
es-cluster-3   1/1     Running   0          66s
$
```

3. Scaling down the replicas from 4 to 2 for sts es-cluster. The StatefulSet controller scales the number of replicas.

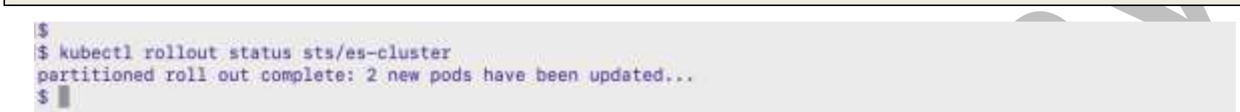
```
$ kubectl scale sts es-cluster --replicas=2
```



```
$
$ kubectl scale sts es-cluster --replicas=2
statefulset.apps/es-cluster scaled
$
```

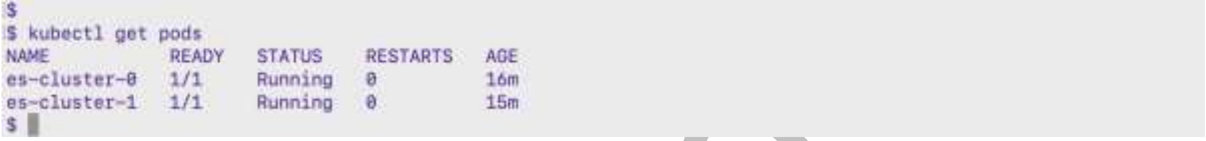
4. The controller deletes one Pod at a time, in reverse order with respect to its ordinal index, and it waits for each to completely shut down before deleting the next.

```
$ kubectl rollout status sts/es-cluster
```



```
$
$ kubectl rollout status sts/es-cluster
partitioned roll out complete: 2 new pods have been updated...
$
```

```
$ kubectl get pods
```



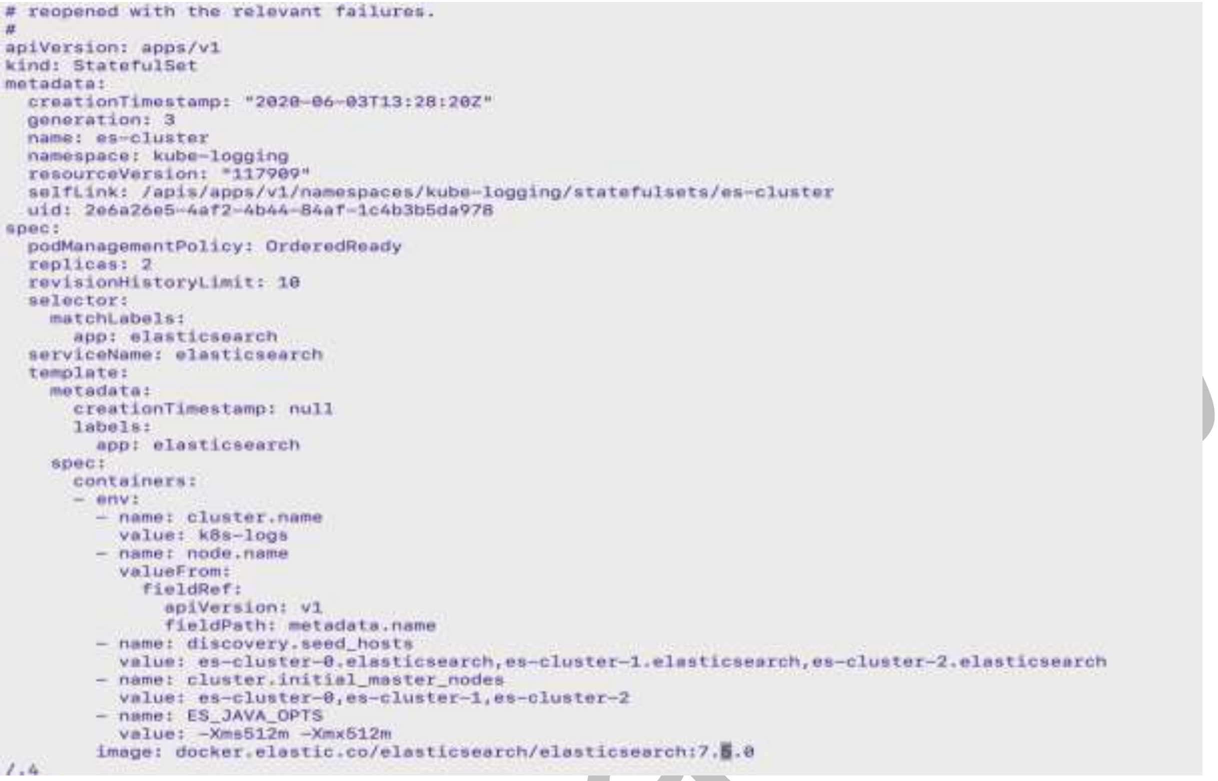
```
$
$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
es-cluster-0   1/1     Running   0          16m
es-cluster-1   1/1     Running   0          15m
$
```

4.5 Rolling update StatefulSets

1. The RollingUpdate update strategy will update all Pods in a StatefulSet, in reverse ordinal order, while respecting the StatefulSet guarantees.

2. Edit the StatefulSet to update the new image version of Elasticsearch elasticsearch:7.5.0

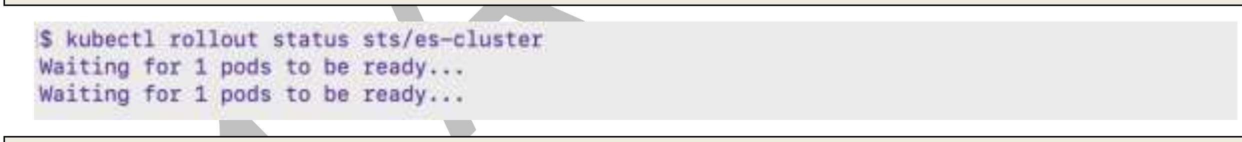
```
$ kubectl edit sts es-cluster
```



```
# reopened with the relevant failures.
#
apiVersion: apps/v1
kind: StatefulSet
metadata:
  creationTimestamp: "2020-06-03T13:28:20Z"
  generation: 3
  name: es-cluster
  namespace: kube-logging
  resourceVersion: "117909"
  selfLink: /apis/apps/v1/namespaces/kube-logging/statefulsets/es-cluster
  uid: 2e6a26e5-4af2-4b44-84af-1c4b3b5da978
spec:
  podManagementPolicy: OrderedReady
  replicas: 2
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: elasticsearch
  serviceName: elasticsearch
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: elasticsearch
    spec:
      containers:
      - env:
        - name: cluster.name
          value: k8s-logs
        - name: node.name
          valueFrom:
            fieldRef:
              apiVersion: v1
              fieldPath: metadata.name
        - name: discovery.seed_hosts
          value: es-cluster-0.elasticsearch,es-cluster-1.elasticsearch,es-cluster-2.elasticsearch
        - name: cluster.initial_master_nodes
          value: es-cluster-0,es-cluster-1,es-cluster-2
        - name: ES_JAVA_OPTS
          value: -Xms512m -Xmx512m
        image: docker.elastic.co/elasticsearch/elasticsearch:7.5.0
/.4
```

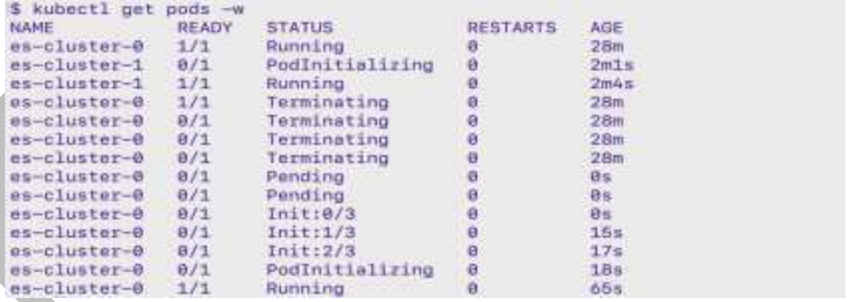
3. Verify the updation of StatefulSet Elasticsearch pods. Monitor the StatefulSet as it is rolled out using kubectl rollout status

```
$ kubectl rollout status sts/es-cluster
```



```
$ kubectl rollout status sts/es-cluster
Waiting for 1 pods to be ready...
Waiting for 1 pods to be ready...
```

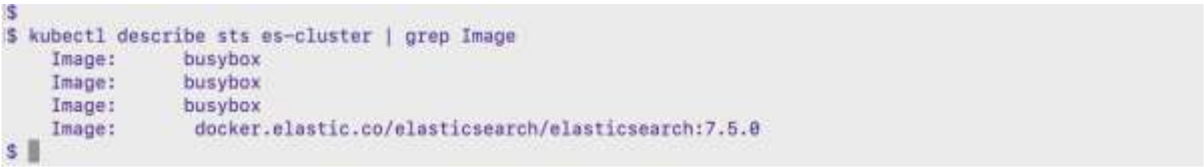
```
$ kubectl get pods -w
```



```
$ kubectl get pods -w
NAME           READY   STATUS            RESTARTS   AGE
es-cluster-0   1/1     Running           0          28m
es-cluster-1   0/1     PodInitializing   0          2m1s
es-cluster-1   1/1     Running           0          2m4s
es-cluster-0   1/1     Terminating       0          28m
es-cluster-0   0/1     Terminating       0          28m
es-cluster-0   0/1     Terminating       0          28m
es-cluster-0   0/1     Terminating       0          28m
es-cluster-0   0/1     Pending           0          0s
es-cluster-0   0/1     Pending           0          0s
es-cluster-0   0/1     Init:0/3          0          0s
es-cluster-0   0/1     Init:1/3          0          15s
es-cluster-0   0/1     Init:2/3          0          17s
es-cluster-0   0/1     PodInitializing   0          18s
es-cluster-0   1/1     Running           0          65s
```

4. Verify the image version with describe command

```
$ kubectl describe sts es-cluster | grep Image
```



```
$
$ kubectl describe sts es-cluster | grep Image
    Image:      busybox
    Image:      busybox
    Image:      busybox
    Image:       docker.elastic.co/elasticsearch/elasticsearch:7.5.0
$
```

4.6 Clean Up resources created the lab exercise

```
$ kubectl delete ns kube-logging
$ kubectl config set-context --current --namespace=default
```

5 SUMMARY

In this guide we Covered:

- Advanced Routing with Ingress-Controller
- Deploying and Managing a StatefulSet Resource