

## Install the KumoScale Provisioner

This page explains how to install the KumoScale Provisioner with your Kubernetes cluster. The KumoScale Provisioner Software is delivered as a binary container for Kubernetes deployment and requires specific configurations detailed on this page.

### •Install the KumoScale Provisioner

Step 1 Confirm your Kubernetes cluster meets requirements.  
Step 2 Install the KumoScale Provisioner

### Step 1. Confirm your Kubernetes cluster meets requirements

#### •Install the KumoScale Provisioner

Step 1 Confirm your Kubernetes cluster meets requirements.

Before you begin installing the KumoScale Provisioner, confirm

- Your Kubernetes cluster is installed and configured for high availability according to the best practices of the selected distribution as explained in the *Before you Begin* section of [Overview and Requirements for Installing KumoScale for Managed Mode with Kubernetes](#)
- You have completed Step 4. Create the KumoScale Provisioner Container Image defined in [Download KumoScale Software and Prepare for Installation](#).

You will need the following to complete the installation:

1. **If using a private registry, the Kubernetes Secret** storing the authentication credentials will be needed. This secret was created in [Step 3b of installing the KumoScale Operators](#).
2. **Provisioner version number.** You will use the Provisioner version number to specify the value of the **image tag** in the Provisioner CR described in the next section.
3. **Consul version number.** You will use the consul version number to specify the value of the **consul tag** in the Provisioner CR described in the next section.
4. **Token or OIDC Authentication information.** KIOXIA provides a token to use for authentication. This is useful for Proof of Concept (POC) or other non-production environments. For Production, you will want to use Open ID Connect (OIDC) authentication and will need to provide additional parameters when configuring the Provisioner.

### Step 2. Install the KumoScale Provisioner

#### •Install the KumoScale Provisioner

Step 2 Install the Provisioner

To install the KumoScale Provisioner you will use kubectl apply with ks-provisioner.yaml defined below.

1. Create the KumoScale Provisioner service by doing the following:

a) Copying the following code block into a file called **ks-provisioner-service.yaml**.

```
---
apiVersion: v1
kind: Service
metadata:
  name: ks-provisioner-service
  namespace: kumo-services
spec:
  selector:
    app: ks-provisioner
  ports:
    - name: https
      port: 30100
      targetPort: 8443
  type: LoadBalancer
---
apiVersion: v1
kind: Service
metadata:
  name: ks-consul-service
  namespace: kumo-services
spec:
  selector:
    app: ks-provisioner
  ports:
    - name: consul-http
      port: 8500
      protocol: TCP
      targetPort: 8500
    - name: consul-dns
      port: 8600
      protocol: TCP
      targetPort: 8600
```

```
---

apiVersion: v1
kind: ServiceAccount
metadata:
  name: ks-provisioner-sa
  namespace: kumo-services
```

b) Create the service:

```
kubect1 apply ks-provisioner-service.yaml
```

c) Verify the KumoScale Provisioner service was created with:

```
kubect1 -n kumo-services get svc
```

You should receive something like the following

NAME	TYPE	CLUSTER IP	EXTERNAL IP	PORT	AGE
ks-provisioner-service	LoadBalancer	###.###.###.###.	172.##.##.##	8443:31053/TCP	12s

The value of External IP in the above step is assigned by your load balancer and should be a known IP address that never changes. It You will use this IP address in step 2. If you prefer to add this IP to your DNS server, you can use the FQDN in future steps.

2. Create the KumoScale Provisioner by completing the following:

a) Copy the code block below into a file called **ks-provisioner.yaml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ks-provisioner-deployment
  namespace: kumo-services
spec:
  selector:
    matchLabels:
      app: ks-provisioner
  replicas: 1
  template:
    metadata:
      labels:
        app: ks-provisioner
    spec:
      serviceAccount: ks-provisioner-sa
      tolerations:
        - key: "node.kubernetes.io/unreachable"
          operator: "Exists"
          effect: "NoExecute"
          tolerationSeconds: 2
        - key: "node.kubernetes.io/not-ready"
          operator: "Exists"
          effect: "NoExecute"
          tolerationSeconds: 2
      # To run only on masters: uncomment and edit if needed
      # affinity:
      #   nodeAffinity:
      #     requiredDuringSchedulingIgnoredDuringExecution:
      #       nodeSelectorTerms:
      #         - matchExpressions:
      #           - key: node-role.kubernetes.io/master
      #             operator: Exists

      volumes:
        - name: localtime
          hostPath:
            path: /etc/localtime
      # To use custom tls secret: uncomment and edit
      # - name: jks
      #   secret:
      #     secretName: $SECRET
      # - name: keystore-volume
      #   emptyDir:
      #     medium: Memory

      imagePullSecrets:
        - name: <yoursecret>
      containers:
        - name: ks-provisioner
          #edit
          image: <yourlocalregistry>/provisioner:v3.22-1874
          args:
            - "--ip=$(PROVISIONER_IP)"
            - "--server.port=8443"
            - "--port=30100"
            - "--consul.ip=$(CONSUL_IP)"
            - "--spring.config.location=classpath:application.properties"
          # To use custom tls secret: uncomment
          # - --server.ssl.trust-store=/keystore/jksFile
          # - --server.ssl.trust-store-password=$(JKS_PASSWORD)
          # - --server.ssl.key-store=/keystore/jksFile
          # - --server.ssl.key-store-password=$(JKS_PASSWORD)
          # - --server.ssl.key-alias=$(JKS_KEY_ALIAS)
          env:
            - name: PROVISIONER_IP
              #edit
              value: <External IP>
            - name: CONSUL_IP
              value: 127.0.0.1
            - name: KUBE_NODE_NAME
              valueFrom:
                fieldRef:
                  fieldPath: spec.nodeName
          # To use custom tls secret: uncomment and edit
          # - name: JKS_PASSWORD
          #   valueFrom:
          #     secretKeyRef:
```

```
#           name: $SECRET
#           key: jksPassword
#       - name: JKS_KEY_ALIAS
#         valueFrom:
#           secretKeyRef:
#             name: $SECRET
#             key: jksKeyAlias

securityContext:
  runAsUser: 100
  runAsGroup: 101
  imagePullPolicy: "Always"
  volumeMounts:
    - name: localtime
      mountPath: /etc/localtime
#       To use custom tls secret: uncomment
#       - name: jks
#         mountPath: /provisioner
#         readOnly: true
#       - name: keystore-volume
#         mountPath: /keystore

ports:
  - containerPort: 8080
    name: healthz
    protocol: TCP
  livenessProbe:
    httpGet:
      path: /actuator/health/liveness
      port: healthz
      initialDelaySeconds: 25
      periodSeconds: 5
  readinessProbe:
    httpGet:
      path: /actuator/health/readiness
      port: healthz
      initialDelaySeconds: 25
      periodSeconds: 5
#       To use custom tls secret: uncomment
#       lifecycle:
#         postStart:
#           exec:
#             command:
#               - bash
#               - -c
#               - cat /provisioner/jksFile > /keystore/jksFile

- name: ks-consul
  image: <yourlocalregistry>/consul:1.8.4
  command:
    - '/bin/sh'
    - '-ec'
    - |
      consul agent -data-dir=/var/lib/consul -server -bootstrap-expect 1 -client 0.0.0.0
  env:
    - name: KUBE_NODE_NAME
      valueFrom:
        fieldRef:
          fieldPath: spec.nodeName
  volumeMounts:
    - name: localtime
      mountPath: /etc/localtime
```

b) Edit ks-provisioner.yaml with and replace

- <yoursecret> with your Kubernetes secret
- <External IP> with the value of the External IP from step 1c.
- <yourlocalregistry> with the location of your private registry.

```
<...>

imagePullSecrets:
  - name: <your secret>
containers:
  - name: ks-provisioner
  #edit
  image: <yourlocalregistry>/provisioner:v3.22-1874
  <...>
  env:
    - name: PROVISIONER_IP
      #edit
      value: <External IP>
  <...>
  - name: ks-consul
    image: <yourlocalregistry>/consul:1.8.4
```

c) Create the KumoScale Provisioner

```
kubectl apply ks-provisioner.yaml
```

d) Verify the KumoScale Provisioner was created with:

```
kubectl -n kumo-services get pods
```

You should receive something like

NAME	Status	Age
ks-provisioner-deployment-abcde	2/2 Running	0 23s

3. To use the KumoScale Cluster Manager CLI and set up storage nodes, you will need to set the Provisioner secret with the file **provisioner-secret.yaml**. An example file is in the directory **KumoScale Operator/deploy/crds** Below is a table showing the parameters and expected values for each.

Provisioner Secret Parameters	Description
url	Provisioner full URL . This should be the IP and port number a from step 4. ex. https://172.###.##.###:8443
token	Provisioner valid authorization token; applies to Local and LDAP authentication mode only. You may use the one provided by KIOXIA for testing or other non-production purposes.
authServerTokenUrl	URL for authorization server used for generating tokens; applies to OpenIDC authentication mode only. Openn IDC is recommended for Production systems.
provisionerClientID	Client ID of a client, which has a service account role of ADMIN at the provisioner resource; applies to OpenIDC authentication mode only.
provisionerClientSecret	The client secret; applies to OpenIDC authentication mode only.
ksClientID	Client ID of a client, which has a service account role of ADMIN at the KumoScale resource; not for a private cluster
ksClientSecret	The client secret; not for a private cluster
storagenodeClientID	The Client ID of a client. Applies to a private cluster in OpenIDC authentication mode only.
storagenodeClientSecret	The client secret. Applies to a private cluster in OpenIDC authentication mode only.

Apply the changes and create the Provisioner secret with:

```
kubect1 apply -f provisioner-secret.yaml
```

You can verify the secret was created with

```
kubect1 -n kube-system get secrets kumoscale-provisioner
```

*Next installation step:* [Install the KumoScale Cluster Manager CLI](#)

-