**Upgrading Kured**

1. **Check the kured version,chart and APP version**

#helm ls -n kured

1. **Search the kured repo**

#helm search repo | grep -I kured

1. **Upgrade the kured**

#helm upgrade kured kured/kured -n kured

1. **After update Check the kured version,chart and APP version**

#helm ls -n kured

1. **Pod will restart after upgrade We can check Kured pod is ready or not**

#Kubectl get all -n kured

**Note**:-No Downtime require for this and no need to communicate to Application users for this.

++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**Upgrading ingress-nginx**

## With Helm[¶](https://kubernetes.github.io/ingress-nginx/deploy/upgrade/#with-helm)

If you installed ingress-nginx using the Helm command in the deployment docs so its name is ingress-nginx, you should be able to upgrade using

helm upgrade --reuse-values ingress-nginx ingress-nginx/ingress-nginx

## Without Helm[¶](https://kubernetes.github.io/ingress-nginx/deploy/upgrade/#without-helm)

To upgrade your ingress-nginx installation, it should be enough to change the version of the image in the controller Deployment.

I.e. if your deployment resource looks like (partial example):

kind: Deployment

metadata:

name: ingress-nginx-controller

namespace: ingress-nginx

spec:

replicas: 1

selector: ...

template:

metadata: ...

spec:

containers:

- name: ingress-nginx-controller

image: registry.k8s.io/ingress-nginx/controller:v1.0.4@sha256:545cff00370f28363dad31e3b59a94ba377854d3a11f18988f5f9e56841ef9ef

args: ...

simply change the v1.0.4 tag to the version you wish to upgrade to. The easiest way to do this is e.g. (do note you may need to change the name parameter according to your installation):

kubectl set image deployment/ingress-nginx-controller \

controller=registry.k8s.io/ingress-nginx/controller:v1.0.5@sha256:55a1fcda5b7657c372515fe402c3e39ad93aa59f6e4378e82acd99912fe6028d \

-n ingress-nginx

**How does it relate to the cluster patching?**

We need to check the compatibility of the ingress controllers with Kubernetes version. So we can upgrade this as per required version for particular Kubernetes Version.



**++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**Upgrading Secrets Store CSI Driver and Azure Key Vault Provider**

helm upgrade to the latest chart release in the repo will update the Azure Key Vault Provider and Secrets Store CSI Driver to the compatible versions

**helm upgrade csi-secrets-store secrets-store-csi-driver/secrets-store-csi-driver --namespace=NAMESPACE**

Set NAMESPACE to the same namespace where the driver was originally installed, (i.e. kube-system)

**Example:-**

* This updates the driver version to v0.0.14+
* This updates the provider version to 0.0.9+
* This updates the driver manifest to include the flag --grpc-supported-providers=azure to enable communication between driver and provider using gRPC

Run the following commands to confirm the images have been updated -

1. secrets-store container in secrets-store-csi-driver pod is running v0.0.14+

➜ kubectl get ds -l app**=**secrets-store-csi-driver -o jsonpath**=**'{range .items[\*]}{.spec.template.spec.containers[1].image}{"\n"}'

mcr.microsoft.com/k8s/csi/secrets-store/driver:v0.0.14

1. secrets-store container in the secrets-store-csi-driver pod contains the arg --grpc-supported-providers=azure

➜ kubectl get ds -l app**=**secrets-store-csi-driver -o jsonpath**=**'{range .items[\*]}{.spec.template.spec.containers[1].args}{"\n"}'

**[**"--debug=true","--endpoint=**$(**CSI\_ENDPOINT**)**","--nodeid=**$(**KUBE\_NODE\_NAME**)**","--provider-volume=/etc/kubernetes/secrets-store-csi-providers","--grpc-supported-providers=azure","--metrics-addr=:8080"**]**

1. csi-secrets-store-provider-azure pod is running 0.0.9+

➜ kubectl get ds -l app**=**csi-secrets-store-provider-azure -o jsonpath**=**'{range .items[\*]}{.spec.template.spec.containers[0].image}{"\n"}'

mcr.microsoft.com/oss/azure/secrets-store/provider-azure:0.0.9

**How and when are users informed?**

We can inform the Users of that namespace where there Provider and Secret driver are installed through Email 2 weeks Prior with 3 Reminders.

**Who has which responsibility in the process?**

AKS team (HACT) will take care this activity completely.

**How does the communication work?**

Will send communication via email.

**How does it relate to the cluster patching?**

We can check the Compatibility of drivers and Providers with AKS.

**++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**Helm Upgrade**

This command upgrades a release to a new version of a chart.

**helm upgrade [RELEASE] [CHART] [flags]**

Now We can check the version of Helm

**helm version –-template**

**How and when are users informed?**

No need to inform to user as no downtime require for this.

**Who has which responsibility in the process?**

AKS team will take care this activity**.**

**How does the communication work?**

No communication requires as no downtime need to take for this activity.

**How does it relate to the cluster patching?**

We can check the Helm Version Support Policy. When a new version of Helm is released, it is compiled against a particular minor version of Kubernetes. For example, Helm 3.0.0 interacts with Kubernetes using the Kubernetes 1.16.2 client, so it is compatible with Kubernetes 1.16.

For example, if you are using a version of Helm 3 that was compiled against the Kubernetes 1.17 client APIs, then it should be safe to use with Kubernetes 1.17, 1.16, 1.15, and 1.14. If you are using a version of Helm 2 that was compiled against the Kubernetes 1.16 client APIs, then it should be safe to use with Kubernetes 1.16 and 1.15.

| **Helm Version** | **Supported Kubernetes Versions** |
| --- | --- |
| 3.9.x | 1.24.x - 1.21.x |
| 3.8.x | 1.23.x - 1.20.x |
| 3.7.x | 1.22.x - 1.19.x |
| 3.6.x | 1.21.x - 1.18.x |
| 3.5.x | 1.20.x - 1.17.x |
| 3.4.x | 1.19.x - 1.16.x |
| 3.3.x | 1.18.x - 1.15.x |
| 3.2.x | 1.18.x - 1.15.x |

ss