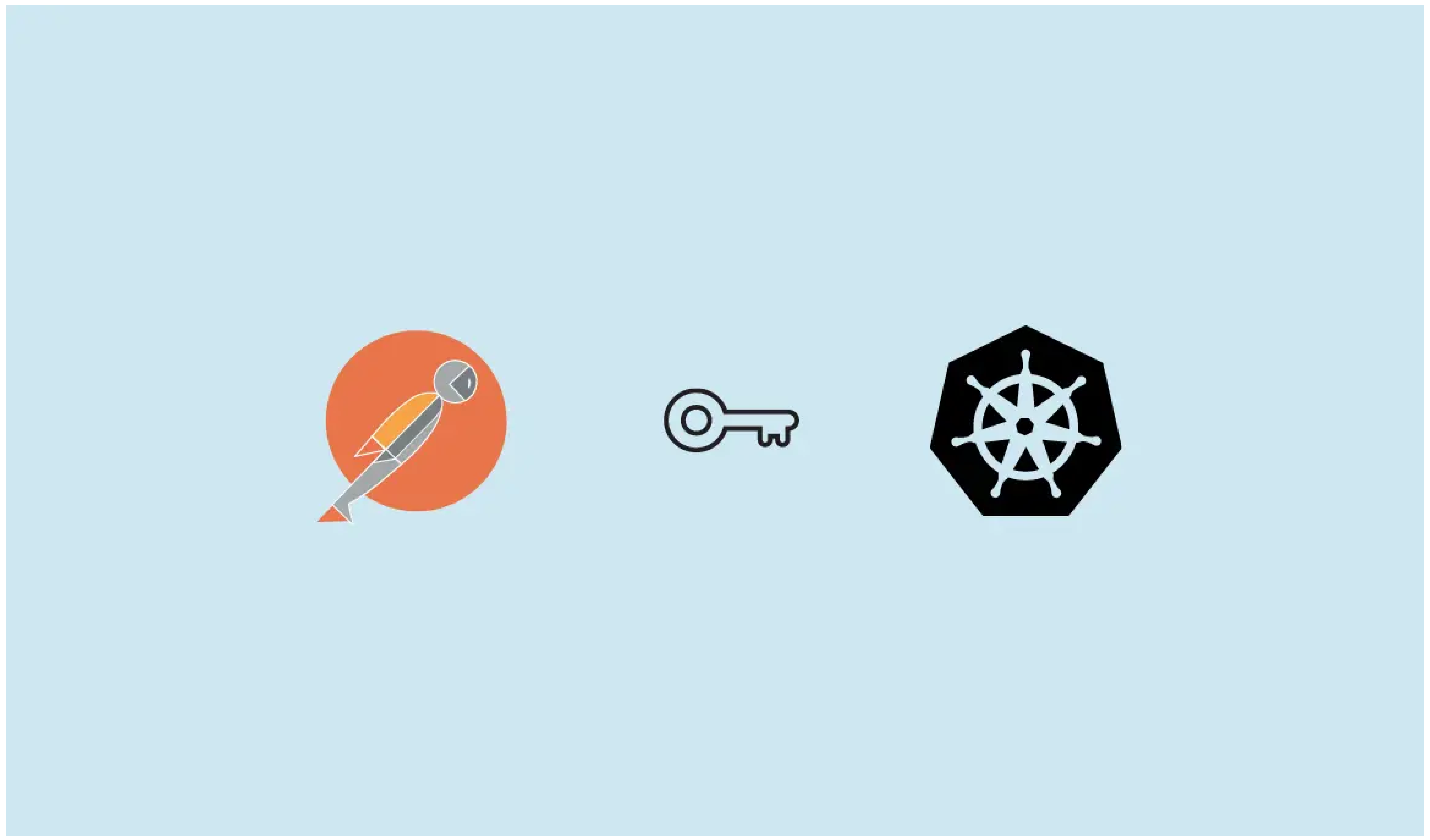
K — KUBERNETES

How To Create Kubernetes Service Account For API Access

by **Bibin Wilson** · June 5, 2021



This tutorial will guide you through the process of creating the service account, role, and role binding to have API access to the kubernetes cluster

The best and recommended way to allow API access to the Kubernetes cluster is through service accounts following the **principle of least privilege** (PoLP).

Use Cases

Following are the example use cases of Kubernetes service account for external API access.

1. Allowing third-party monitoring tools to access Kubernetes data

Now, why would you need this access?

Lets take an example of Prometheus monitoring stack.

Prometheus needs read access to cluster API to get information from metrics server, read pods, etc.

When you deploy Prometheus, you add cluster read permissions to the default service account where the Prometheus pods are deployed. This way, Prometheus pods get read access to cluster resources.

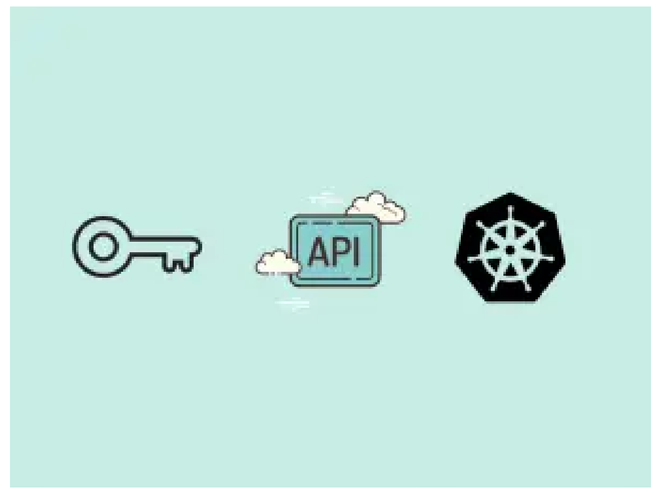
Setup Kubernetes API Access Using Service Account

Follow the steps given below for setting up the API access using the service account.

> ***Note:*** *If you are using GKE on Google Cloud, you might need to run the following two commands to add cluster-admin access to your user account for creating roles and role-bindings with your gcloud user.*

```
ACCOUNT=$(gcloud info --format='value(config.account)')
kubectl create clusterrolebinding owner-cluster-admin-binding \
    --clusterrole cluster-admin \
    --user $ACCOUNT
```

ALSO READ



K — KUBERNETES

How to Create kubernetes Role for Service Account

by Bibin Wilson · June 1, 2021

Step 1: Create service account in a namespace

Create a `devops-tools` namespace.

```
kubectl create namespace devops-tools
```

Create a service account named " `api-service-account` " in `devops-tools` namespace

```
kubectl create serviceaccount api-service-account -n devops-tools
```

or use the following manifest.

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: ServiceAccount
metadata:
  name: api-service-account
  namespace: devops-tools
EOF
```

Step 2: Create a Cluster Role

Assuming that the service account needs access to the entire cluster resources, we will create a cluster role with a list of allowed access.

Create a clusterRole named `api-cluster-role` with the following manifest file.

```
cat <<EOF | kubectl apply -f -
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: api-cluster-role
  namespace: devops-tools
rules:
  - apiGroups:
        - ""
        - apps
        - autoscaling
        - batch
        - extensions
        - policy
        - rbac.authorization.k8s.io
    resources:
      - pods
      - componentstatuses
      - configmaps
      - daemonsets
      - endpoints
      - horizontalpodautoscalers
      - ingress
      - jobs
      - limitranges
      - namespaces
      - nodes
      - pods
      - persistentvolumes
      - persistentvolumeclaims
      - resourcequotas
      - replicasets
      - replicationcontrollers
      - serviceaccounts
      - services
    verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
EOF
```

The above YAML declaration has a `ClusterRole` with full access to all cluster resources and a role binding to " `api-service-account` ".

It is not recommended to create a service account with all cluster component access without any requirement.

To get the list of available API resources execute the following command.

```
kubectl api-resources
```

Step 3: Create a ClusterRole Binding

Now that we have the ClusterRole and service account, it needs to be mapped together.

Bind the `cluster-api-role` to `api-service-account` using a `RoleBinding`

```
cat <<EOF | kubectl apply -f -
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: api-cluster-role-binding
subjects:
- namespace: devops-tools
  kind: ServiceAccount
  name: api-service-account
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: api-cluster-role
EOF
```

Using kubectl

To validate the clusterrole binding, we can use `can-i` commands to validate the API access assuming a service account in a specific namespace.

For example, the following command checks if the `api-service-account` in the `devops-tools` namespace can list the pods.

```
kubectl auth can-i get pods --as=system:serviceaccount:devops-tools:api-service-
account
```

Here is another example, to check if the service account has permissions to delete deployments.

```
kubectl auth can-i delete deployments --as=system:serviceaccount:devops-
tools:api-service-account
```

Step 5: Validate Service Account Access Using API call

To use a service account with an HTTP call, you need to have the token associated with the service account.

First, get the secret name associated with the `api-service-account`

```
kubectl get serviceaccount api-service-account  -
o=jsonpath='{.secrets[0].name}' -n devops-tools
```

Use the secret name to get the base64 decoded token. It will be used as a bearer token in the API call.

```
kubectl get secrets  <service-account-token-name>  -o=jsonpath='{.data.token}'
-n devops-tools | base64 -D
```

For example,

```
kubectl get secrets  api-service-account-token-pgtrr  -
o=jsonpath='{.data.token}' -n devops-tools | base64 -D
```

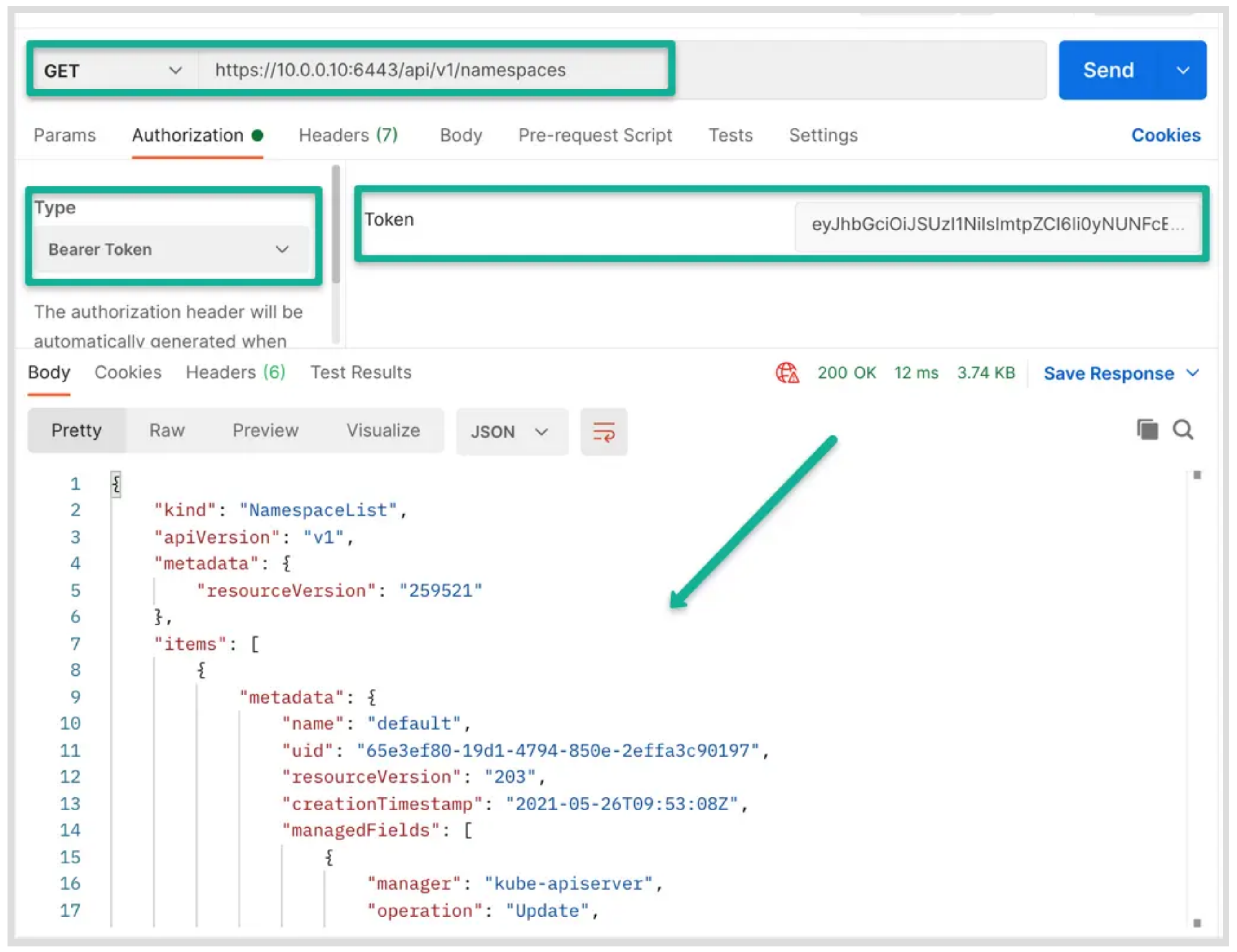
```
kubectl get endpoints | grep kubernetes
```

Now that you have the cluster endpoint and the service account token, you can test the API connectivity using the CURL or the postman app.

For example, list all the namespaces in the cluster using curl. Use the token after `Authorization: Bearer` section.

```
curl -k  https://35.226.193.217/api/v1/namespaces -H "Authorization: Bearer
eyJhbGcisdfsdfsdfiJ9.eyJpc3MiOisdfsdfVhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNj
ezrRXeLS8SLOae4DuOGGGbInSg_gIo6oO7bLHhCixWOBJNOA5gzrLVioof_kHDR8gH5crrsWoR-
GSSsdfgsdfg6fA_LDOqdxzqMC0WlXt6tgHfrwIHerPPvkI6NWLyCqX9tn_akpcihd-
bL6GwOKlph17l_ND710FnTkE7kBfdXtQWWxaPPe06UEmoKK9t-
0gsOCBxJxViwhHkvwqetr987q9enkadfgd_2cY_CA"
```

If can also try that same API call in postman.



https://10.0.0.10:6443/api/v1/namespaces

Type: Bearer Token

Token: eyJhbGciOiJSUzI1NiIsImtpZCI6Ii0yNUNFcE...

The authorization header will be automatically generated when

200 OK 12 ms 3.74 KB

```
{
    "kind": "NamespaceList",
    "apiVersion": "v1",
    "metadata": {
        "resourceVersion": "259521"
    },
    "items": [
        {
            "metadata": {
                "name": "default",
                "uid": "65e3ef80-19d1-4794-850e-2effa3c90197",
                "resourceVersion": "203",
                "creationTimestamp": "2021-05-26T09:53:08Z",
                "managedFields": [
                    {
                        "manager": "kube-apiserver",
                        "operation": "Update",
```

The ClusterRole we created can be attached to pods/deployments as well.

You can also use the token to login to the Kubernetes dashboard.

Conclusion

When using Kubernetes service account for API access from third party applications, ensure you add only required roles to the service account.

access to the clusterRole.

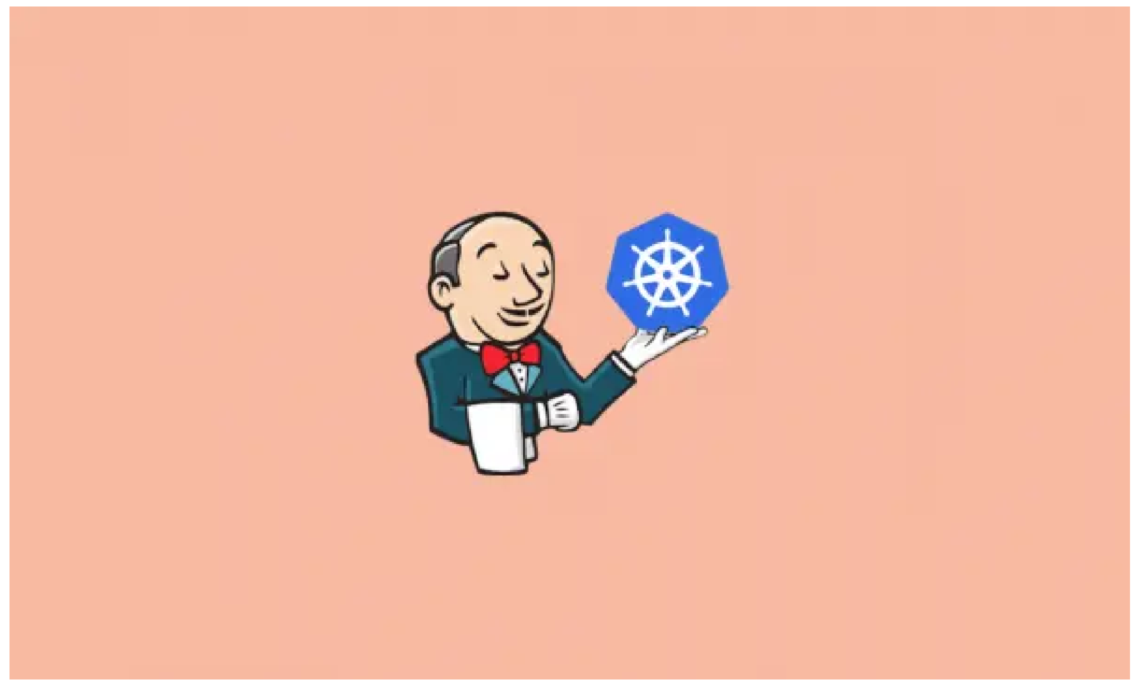
Also, use a secret management tool like Hashicorp vault to store, retrieve, and share secret tokens.

Bibin Wilson

An author, blogger and DevOps practitioner. In spare time, he loves to try out the latest open source technologies. He works as an Associate Technical Architect

VIEW COMMENTS (1)

YOU MAY ALSO LIKE



C — CI/CD

How To Setup Jenkins On Kubernetes Cluster – Beginners Guide

by **Bibin Wilson** · May 9, 2021

Hosting Jenkins on a Kubernetes cluster is beneficial for Kubernetes-based deployments and dynamic container-based scalable Jenkins agents. In...

H — HOW TO

How To Setup Grafana On Kubernetes

by **Bibin Wilson** · April 23, 2021

Grafana is an open-source lightweight dashboard tool. It can be integrated with many data sources like Prometheus, AWS...

K — KUBERNETES

Google Kubernetes Engine

by **Bibin Wilson** · June 17, 2021

In this blog, you will learn how to setup Persistent Volume For the GKE Kubernetes cluster. If you…

K — KUBERNETES

How to Setup Kubernetes Cluster on Vagrant VMs

by **Bibin Wilson** · May 19, 2021

In this post, I have covered the step-by-step guide to setup Kubernetes cluster on Vagrant. It is a…

C — COMMON

Kubernetes Deployment Tutorial For Beginners

by **devopscube** · November 29, 2018

This Kubernetes deployment tutorial guide will explain the key concepts in a Kubernetes YAML specification with an Nginx…

K — KUBERNETES

CKS Exam Study Guide: Resources to Pass Certified Kubernetes Security Specialist

by **Bibin Wilson** · June 23, 2021

In this Certified Kubernetes Security Specialist (CKS) Exam study guide, I have listed all the resources you can…