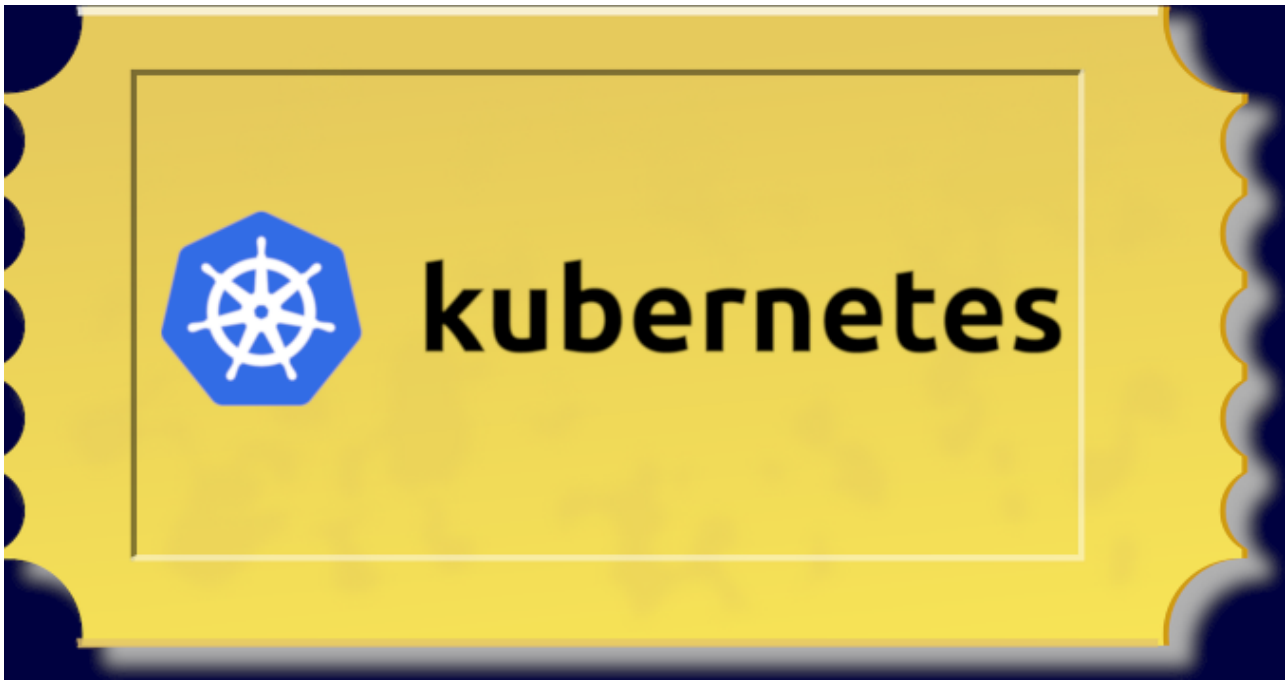
KUBERNETES / SECURITY / TECHNOLOGY

Kubernetes Access Control: Exploring Service Accounts

16 Aug 2019 3:00am, by Janakiram MSV



This is the last part of a tutorial series on Kubernetes access control. Having explored the key concepts related to authentication and authorization, we will take a closer look at service accounts.

Kubernetes has the notion of users and service account to access resources. A user is associated with a key and certificate to authenticate API requests. Any request originated outside of the cluster is authenticated using one of the configured schemes. The most common technique to authenticate requests is through X.509 certificates. Refer to the tutorial on Kubernetes authentication on creating and associating certificates with users.

It's important to recall that Kubernetes doesn't maintain a database or profiles of users and passwords. Instead, it expects it to be managed outside of the cluster. Through the concept of authentication modules, Kubernetes can delegate authentication to a 3rd party like OpenID or Active Directory.

Every Kubernetes installation has a service account called default that is associated with every running pod. Similarly, to enable pods to make calls to the internal API Server endpoint, there is a ClusterIP service called Kubernetes. This combination makes it possible for internal processes to call the API endpoint.

```
kubectl get serviceAccounts
```

```
NAME      SECRETS   AGE
default   1         122m
```

```
kubectl get svc
```

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1            443/TCP   123m
```

Notice that the service account is pointing to a secret that is mounted inside every pod. This secret contains the token expected by the API Server.

```
kubectl get secret
```

```
NAME                  TYPE                                  DATA   AGE
default-token-4rpmv   kubernetes.io/service-account-token   3      123m
```

Things get clear when we actually schedule a pod and access it. We will launch a pod that is based on BusyBox with curl command.

```
kubectl run -i --tty --rm curl-tns --image=radial/busyboxplus:curl
```

```
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version.
If you don't see a command prompt, try pressing enter.
[ root@curl-tns-56c6d54585-6v2xp:/ ]$
```

While we are within the BusyBox shell, let's try to hit the API Server endpoint.

```
[ root@curl-tns-56c6d54585-6v2xp:/ ]$ curl https://kubernetes:8443/api
```

This will not yield any result since the request lacks the authentication token. Let's find out how to retrieve the token that can be embedded in the HTTP header.

```
[ root@curl-tns-56c6d54585-6v2xp:/tmp/secrets/kubernetes.io/serviceaccount ]$ ls
ca.crt     namespace  token
```

Let's set a few environment variables to simplify the curl command.

```
CA_CERT=/var/run/secrets/kubernetes.io/serviceaccount/ca.crt
TOKEN=$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)
NAMESPACE=$(cat /var/run/secrets/kubernetes.io/serviceaccount/namespace)
```

The below curl command requests the list of services running in the default namespace. Let's see if we get a response from the API Server.

```
CERT -H "Authorization: Bearer $TOKEN" "https://kubernetes/api/v1/namespaces/$NAMESPACE/services/"
```

```
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Failure",
  "message": "services is forbidden: User \"system:serviceaccount:default:default\" cannot list re
  "reason": "Forbidden",
  "details": {
    "kind": "services"
  },
  "code": 403
}
```

Surprisingly, the default service account doesn't have enough permissions to retrieve the services running in the same namespace.

Remember that Kubernetes follows the convention of closed-to-open which means that by default no user or service account has any permissions.

In order to fulfill this request, we need to create a role binding associating the default service account with an appropriate role. This workflow is similar to how we bound a role to Bob that gave him permission to list pods.

```
  --clusterrole=view \
  --serviceaccount=default:default \
  --namespace=default
```

```
rolebinding.rbac.authorization.k8s.io/default-view created
```

The above command associated the default service account with the cluster role view that enables the pod to list the resources.

If you are curious to see all the available cluster roles, run the command, *kubectl get clusterroles*.

Let's launch the BusyBox pod again and hit the API Server.

```
kubectl run -i --tty --rm curl-tns --image=radial/busyboxplus:curl
```

```
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version.
If you don't see a command prompt, try pressing enter.
[ root@curl-tns-56c6d54585-2cx44:/ ]$
```

```
CA_CERT=/var/run/secrets/kubernetes.io/serviceaccount/ca.crt
TOKEN=$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)
NAMESPACE=$(cat /var/run/secrets/kubernetes.io/serviceaccount/namespace)
```

```
curl --cacert $CA_CERT -H "Authorization: Bearer $TOKEN" "https://kubernetes/api/v1/namespaces/$NA
```

```
{
  "kind": "ServiceList",
  "apiVersion": "v1",
  "metadata": {
    "selfLink": "/api/v1/namespaces/default/services/",
    "resourceVersion": "11076"
  },
  "items": [
    {
      "metadata": {
        "name": "kubernetes",
        "namespace": "default",
        "selfLink": "/api/v1/namespaces/default/services/kubernetes",
        "uid": "b715a117-6be1-4de0-8830-45bddcda7O1c",
        "resourceVersion": "151",
        "creationTimestamp": "2019-08-13T09:45:27Z",
```

```
      },
      "spec": {
        "ports": [
          {
            "name": "https",
            "protocol": "TCP",
            "port": 443,
            "targetPort": 8443
          }
        ],
        "clusterIP": "10.96.0.1",
        "type": "ClusterIP",
        "sessionAffinity": "None"
      },
      "status": {
        "loadBalancer": {

        }
      }
    }
  ]
}
```

Feel free to create additional bindings for the default service account to check how RBAC extends to pods.

This concludes the series on Kubernetes access control where we discussed the essential building blocks of authentication, authorization, and service accounts.

Janakiram MSV's Webinar series, "Machine Intelligence and Modern Infrastructure (MI2)" offers informative and insightful sessions covering cutting-edge technologies. Sign up for the upcoming MI2 webinar at http://mi2.live.

TUTORIAL

Subscribe

We don't sell or share your email. By continuing, you agree to our Terms of Use and Privacy Policy.

RELATED STORIES

CLOUD SERVICES / MACHINE LEARNING

Tutorial: Create Training and Inferencing Pipelines with Azure ML Designer

15 May 2020 8:31am, by Janakiram MSV

DEVELOPMENT

How to Set up the HTTP Git Server for Private Projects

14 May 2020 10:00am, by Jack Wallen

Please stay on topic and be respectful of others. Review our Terms of Use.

SPONSORED FEED

sonatype

DevSecOps Delivered: Nexus IQ Google Chrome Extension
MAY 22, 2020

GitLab

GitLab CEO Shadow program takeaways and lessons learned
MAY 21, 2020

DIAMANTI

Maximize Your Returns on AI/ML Investment
MAY 21, 2020

PRISMA BY PALO ALTO NETWORKS

FLEXWORK
MAY 21, 2020

redislabs HOME OF REDIS

How Redis Enterprise Powers TransNexus' Microservices Architecture to Help Fight Robocallers
MAY 21, 2020

NetApp

We're Helping Companies Affected by COVID-19 with Free Cloud Storage for 90

Cloud Computing: Choose a Multi-Cloud Strategy or Fly Solo
MAY 21, 2020

circleci

Prerequisites for evolutionary architectures
MAY 21, 2020

THE LINUX FOUNDATION

EdgeX Foundry Hits Major Milestone with 5 Million+ Container Downloads and a New Release that Simplifies Deployment for AI, Data Analytics and Digital Transformation
MAY 21, 2020

Lightbend

Lightbend Podcast: Scaling Akka Cluster to 10000 nodes with Rapid
MAY 21, 2020

cloudbees

CloudBees SDM in Action: How to Identify Unreviewed Pull-Requests Blocking Jira Issues
MAY 20, 2020

SENTRY

Dogfooding Chronicles: Tracing the path from “It’s Slow” to “What’s Slow”
MAY 20, 2020

SAP

UI5 Web Components version rc.7 is here!
MAY 20, 2020

MAY 20, 2020

portworx

How to take a backup of Kubernetes applications
MAY 20, 2020

ASPEN MESH

Announcing Aspen Mesh 1.5
MAY 20, 2020

HashiCorp

Connect Terraform Workspaces with Run Triggers: New Tutorial
MAY 20, 2020

New Relic.

Reduce MTTR With Recent Innovations in New Relic Logs
MAY 20, 2020

influxdata

How to Run a Time Series Database on Azure
MAY 20, 2020

HAPROXY

HAProxy Process Management
MAY 20, 2020

SALTSTACK.

The State of “XOps”
MAY 19, 2020

aws

DATASTAX

Cloud-Native Architecture Puts the Power Back in Developers' Hands
MAY 19, 2020

TRIGGERMESH

TriggerMesh EveryBridge, Event Flows from Any App to Any Service
MAY 19, 2020

APPDYNAMICS

Introducing Shifting Left: A New IT Content Series from AppDynamics
MAY 19, 2020

mongoDB.

How to Build a Customer Success Team in Six Steps: My MongoDB Journey
MAY 19, 2020

CLOUD FOUNDRY

Bi-weekly Round-Up: Technical + Ecosystem Updates from Cloud Foundry | 5.19.20
MAY 19, 2020

pagerduty

Driving Real-Time ChatOps With PagerDuty and Microsoft Teams by Mya King
MAY 19, 2020

accurics

Breach Path Prediction: Getting Ahead of Cloud Breaches
MAY 18, 2020

MAY 14, 2020

snyk

JavaScript frameworks security report 2019
MAY 06, 2020

packet

Introducing the Edge Metal Private Beta
MAY 04, 2020

CITRIX

Troubleshoot cloud-native apps with Citrix ADM Distributed Tracing
APRIL 29, 2020

NS1.

Enhance Network Visibility and Control with NS1 Enterprise DDI and Cisco Umbrella
APRIL 27, 2020

vmware

Managing Kubernetes at enterprise scale: A closer look at Tanzu Mission Control
MARCH 19, 2020

mongoDB.

Getting storage engines ready for fast storage devices
MARCH 16, 2020

ARCHITECTURE

Cloud Native

Containers