K21Academy
Learn Cloud from Experts

docker + kubernetes

Use Serviceaccount In Pod, Access API Inside Pod & Provide Acess to Serviceaccount using RBAC

[Edition 1]

[Last Update 201224]

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

Contents

1 INTRODUCTION

Service Account:

Service accounts are for processes, which run in pods. Service accounts are namespaced.

Configure Service Accounts for Pods

A service account provides an identity for processes that run in a Pod.

When you access the cluster using kubectl, you are authenticated by the apiserver as a particular User Account. Processes in containers inside pods can also contact the apiserver. When they do, they are authenticated as a particular Service Account (for example, default).

In this guide we have covered:

In this guide first we will create a serviceaccount then using that service account token we will try to access the API inside the pod. Afte this we will disable serviceaccount mount and then we will limit the serviceaccount access using RBAC.

2 DOCUMENTATION

2.1 Kubernetes Documentation

1. Kubernetes: access the API inside a Pod
   https://medium.com/@antoine_martin/kubernetes-access-the-api-inside-a-pod-eb49af8c8b06
2. Managing Service Accounts
   https://kubernetes.io/docs/reference/access-authn-authz/service-accounts-admin/
3. Configure Service Accounts for Pods
   https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/
4. Using RBAC Authorization
   https://kubernetes.io/docs/reference/access-authn-authz/rbac/

3 PRE-REQUSITE GUIDES

Ensure that you have completed following three activity guides (or you have an Ubuntu Server)

- Create account (Trial or Paid) on Azure Cloud.

Note**: Follow Activity Guide **Register_For_Azure_Cloud_Account_Accessing_Console_ed* *from portal*

Note**: Follow Activity Guide **Create_&_Connect_to_ubuntu_server_on_Azure_Cloud_ed* *from portal*

Note**: Follow Activity Guide **AG_Bootstrap_Kubernetes_Cluster_Using_Kubeadm_Guide_ed* *from portal*

4 USE SERVICEACCOUNT IN POD & ACCESS API INSIDE A POD

1. First we will create service account

```
$ kubectl create serviceaccount my-service-account
```

```
root@master:/home/ubuntu# kubectl create serviceaccount my-service-account
serviceaccount/my-service-account created
root@master:/home/ubuntu#
```

Service account create secret token which is used for API access.

```
$ kubectl get sa,secret
```

```
root@master:/home/ubuntu# kubectl get sa,secret
NAME                                 SECRETS   AGE
serviceaccount/default               1         2d1h
serviceaccount/my-service-account    1         2m52s

NAME                                      TYPE                                  DATA   AGE
secret/default-token-57pzl                kubernetes.io/service-account-token   3      2d1h
secret/my-service-account-token-fw8rb     kubernetes.io/service-account-token   3      2m52s
root@master:/home/ubuntu#
```

2. Describe secret token to check token.

```
$ kubectl describe
```

```
root@master:/home/ubuntu# kubectl describe secret/my-service-account-token-fw8rb
Name:         my-service-account-token-fw8rb
Namespace:    default
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: my-service-account
              kubernetes.io/service-account.uid: b00e586a-b4a8-44fa-b129-2ea1ef5f39d0

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:     1066 bytes
namespace:  7 bytes
token:      eyJhbGciOiJSUzI1NiIsImtpZCI6IjhQcXU0UkFnRUVXZDlJWmlVQzZ0a3FXUG9VN2loZVNGdnp3TEFlY21vTW8ifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJkZWZhdWx0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6Im15LXNlcnZpY2UtYWNjb3VudC10b2tlbiIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJteS1zZXJ2aWNlLWFjY291bnQiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiJiMDBlNTg2YS1iNGE4LTQ0ZmEtYjEyOS0yZWExZWY1ZjM5ZDAiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6ZGVmYXVsdDpteS1zZXJ2aWNlLWFjY291bnQifQ.f-KQI210oPTqTcpXyr5OBQeMy_Bww3aShdlQ-C8jf85d641gJi94eRiu79m2cjo7vOdp0YcTXEx0XvXM4e3_rhaCDQAy7FFQfXcsXYYUvUIRlykA9o8AeXu6hiyI84GpYh8Mq-Nz7iz2DupfJiqGe6GoQJlBAQdnEPSYiOyQgmewRsgW5m88jP1-e6Ik0_QpRShGoVq2LPzfyZsCLvLLZTKjDxjUZFMheUEoAwAVAjtVf94VKZpjpmoQO1jzVm_hKp8X9UZrg9-lvJp3XQO2opygyGbpaGjXK2OuQ4bkP4KXpWZfMIbUCijMx55X-9jAaTT779VNz0PBLiWZ0scxNA
root@master:/home/ubuntu#
```

Now using this token, we can now authenticate against the api server and do whatever the service account is allowed to do.

3. Create pod using this serviceaccount

```
$ kubectl run my-pod --image=nginx --dry-run=client -o yaml > my-pod.yaml
```

Using *--dry-run=client* this flag will not create pod then -o yaml will create yaml file then we will edit this to add service account in pod.

```
vim my-pod.yaml
```

Add serviceAccountName in pod definition file.

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: my-pod
  name: my-pod
spec:
 serviceAccountName: my-service-account
  containers:
  - image: nginx
    name: my-pod
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
~
~
~
```

4. Now we will create pod

```
$ kubectl create -f my-pod.yaml
```

```
root@master:/home/ubuntu# kubectl run my-pod --image=nginx --dry-run=client -o yaml > my-pod.yaml
root@master:/home/ubuntu# vim my-pod.yaml
root@master:/home/ubuntu# kubectl create -f my-pod.yaml
pod/my-pod created
root@master:/home/ubuntu#
```

5. ServiceAccount Admission Controller adds a *volumeSource* to each container of the pod mounted at */var/run/secrets/kubernetes.io/serviceaccount* which contains a token for API access.

   To check this

```
$ kubectl exec -it my-pod -- bash
# mount | grep sec
# cd /run/secrets/kubernetes.io/serviceaccount
# ls
# cat token
```

```
root@master:/home/ubuntu# kubectl exec -it my-pod -- bash
root@my-pod:/# mount | grep sec
tmpfs on /run/secrets/kubernetes.io/serviceaccount type tmpfs (ro,relatime)
root@my-pod:/# cd /run/secrets/kubernetes.io/serviceaccount
root@my-pod:/run/secrets/kubernetes.io/serviceaccount# ls
ca.crt  namespace  token
root@my-pod:/run/secrets/kubernetes.io/serviceaccount# cat token
eyJhbGciOiJSUzI1NiIsImtpZCI6IjhQcXU0UkFnRUVXZDlJWmlVQzZ0a3FXUG9VN2loZVNGdnp3TEFlY2lvTW8ifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJkZWZhdWx0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6Im15LXNlcnZpY2UtYWNjb3VudC10b2tlbiIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJteS1zZXJ2aWNlLWFjY291bnQiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiJiMDBlNTg2YS1iNGE4LTQ0ZmEtYjEyOS0yZWExZWY1ZjM5ZDAiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6ZGVmYXVsdDpteS1zZXJ2aWNlLWFjY291bnQifQ.f-KQIZ1OoPTqTcpXyr5OBQeMy_Bww3aShd1Q-C8jf85d641gJi94eRiu79m2cjo7vOdp0YcTXEx0XvXM4e3_rhaCDQAy7FFQfXcsXYYUvUIR1ykA9o8AeXu6hiyI84GpYh8Mq-Nz7iz2DupfJiqGe6GoQJlBAQdnEPSYiOyQgmewRsgW5m88jP1-e6Ik0_QpRShGoVq2LPzfyZsCLvLLZTKjDxjUZFMheUEoAwAVAjtVf94VKZpjpmoQO1jzVm_hKp8X9UZrg9-1vJp3XQO2opygyGbpaGjXK2OuQ4bkP4KXpWZfMIbUCijMx55X-9jAaTT779VNz0PBLiWZ0scxNAroot@my-pod:/run/secrets/kubernetes.io/serviceaccount#
```

Also you can check volume by describing the pod

```
kubectl describe pod my-pod
```

```
root@master:/home/ubuntu# kubectl describe pod my-pod
Name:         my-pod
Namespace:    default
Priority:     0
Node:         worker/10.0.0.7
Start Time:   Thu, 24 Dec 2020 08:09:51 +0000
Labels:       run=my-pod
Annotations:  <none>
Status:       Running
IP:           10.44.0.1
IPs:
  IP:  10.44.0.1
Containers:
  my-pod:
    Container ID:   docker://559aad6f19fcf1db767b99475764ab243592d87321d391cdcb9cc45ae0b1e475
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:4cf620a5c81390ee209398ecc18e5fb9dd0f5155cd82adcbae532fec94006fb9
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Thu, 24 Dec 2020 08:10:10 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from my-service-account-token-fw8rb (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  my-service-account-token-fw8rb:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  my-service-account-token-fw8rb
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
```

6. Accessing Api inside a pod using below command

```
# curl https://kubernetes -H "Authorization: Bearer $(cat
/var/run/secrets/kubernetes.io/serviceaccount/token)" --cacert
/var/run/secrets/kubernetes.io/serviceaccount/ca.crt
```

```
root@my-pod:/# curl https://kubernetes -H "Authorization: Bearer $(cat /var/run/secrets/kubernetes.io/serviceaccount/token)" --cacert /var/run/secrets/kub
ernetes.io/serviceaccount/ca.crt
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Failure",
  "message": "forbidden: User \"system:serviceaccount:default:my-service-account\" cannot get path \"/\"",
  "reason": "Forbidden",
  "details": {

  },
  "code": 403
}root@my-pod:/#
```

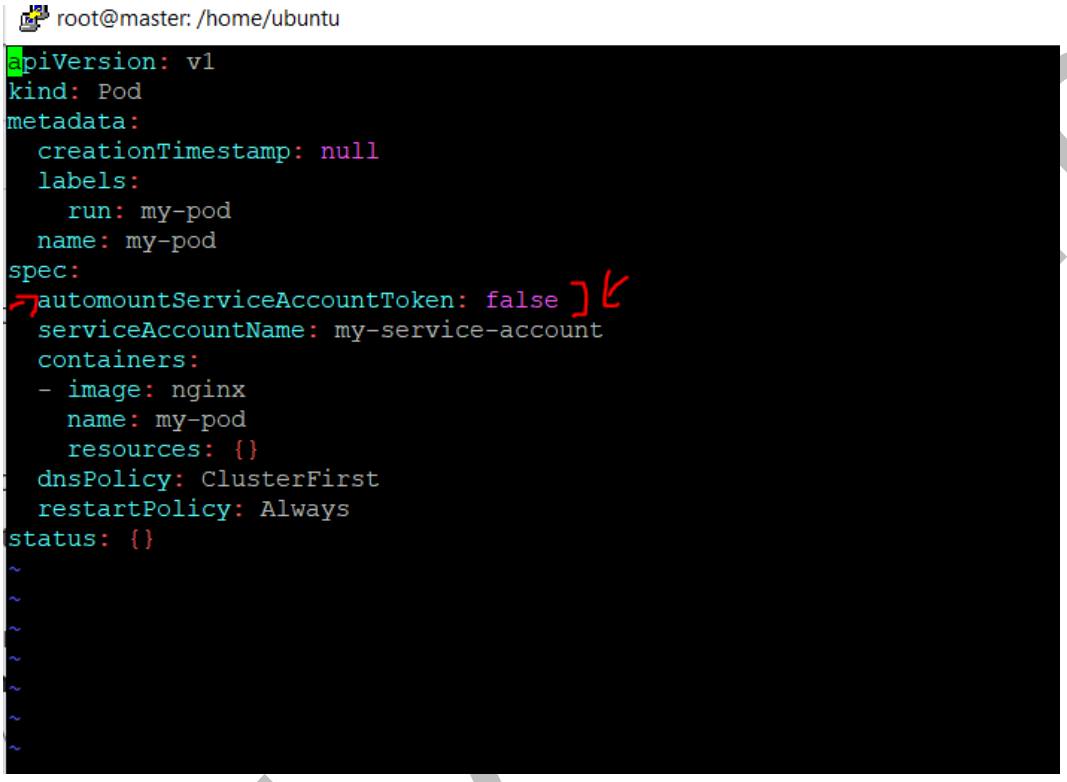
This Service account don't have access to the API, We can give access to secriveaccount using RBAC.

5 DISABLE SERVICEACCOUNT MOUNT

Note: you can opt out of automounting API credentials for a service account by setting automountServiceAccountToken: false on the service account or on a particular pod.

1. Add automountServiceAccountToken: false on pod definition file which we created earlier:

```
vim my-pod.yaml
```



2. Recreate the pod.

```
$ kubectl replace -f my-pod.yaml --force
```

```
root@master:/home/ubuntu# kubectl replace -f my-pod.yaml --force
pod "my-pod" deleted
pod/my-pod replaced
root@master:/home/ubuntu#
```

3. Check by describing the pod no volume available.

```
$ kubectl describe pod my-pod
```

```
root@master:/home/ubuntu# kubectl describe pod my-pod
Name:         my-pod
Namespace:    default
Priority:     0
Node:         worker/10.0.0.7
Start Time:   Thu, 24 Dec 2020 10:14:43 +0000
Labels:       run=my-pod
Annotations:  <none>
Status:       Running
IP:           10.44.0.1
IPs:
  IP:  10.44.0.1
Containers:
  my-pod:
    Container ID:   docker://f9467342c28731feb7041299be69309870ccc5e2eba59589c35053c88e327f3a
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:4cf620a5c81390ee209398ecc18e5fb9dd0f5155cd82adcbae532fec94006fb9
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Thu, 24 Dec 2020 10:14:44 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:         <none>
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:            <none>
QoS Class:          BestEffort
Node-Selectors:     <none>
Tolerations:        node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                    node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  78s   default-scheduler  Successfully assigned default/my-pod to worker
  Normal  Pulling    77s   kubelet            Pulling image "nginx"
```

6 PROVIDE ACCESS TO SERVICEACCOUNT USING RBAC

1. By default service account don't have any permission so we can give permission to service account using RBAC

   Privode default edit clusterrole access to Serviceaccount:

   ```
   $ kubectl create clusterrolebinding my-service-account-policy --clusterrole edit --serviceaccount=default:my-service-account
   ```

```
root@master:/home/ubuntu# kubectl create clusterrolebinding my-service-account-policy --clusterrole edit --serviceaccount=default:my-service-account
clusterrolebinding.rbac.authorization.k8s.io/my-service-account-policy created
root@master:/home/ubuntu#
```

2. Check access

   (check if can we delete or list pod in default namespace using service account my-service-account)

   ```
   $ kubectl auth can-i delete pods --as system:serviceaccount:default:my-service-account
   $ kubectl auth can-i list pods --as system:serviceaccount:default:my-service-account
   ```

   (check if can we delete or list pod in kube-system namespace using service account my-service-account)

   ```
   $ kubectl auth can-i list pod --as system:serviceaccount:kube-system:my-service-account
   ```

```
root@master:/home/ubuntu# kubectl auth can-i delete pods --as system:serviceaccount:default:my-service-account
yes
root@master:/home/ubuntu# kubectl auth can-i list pods --as system:serviceaccount:default:my-service-account
yes
root@master:/home/ubuntu# kubectl auth can-i list  --as system:serviceaccount:kube-system:my-service-account
error: you must specify two or three arguments: verb, resource, and optional resourceName
root@master:/home/ubuntu# kubectl auth can-i list pod --as system:serviceaccount:kube-system:my-service-account
no
root@master:/home/ubuntu#
```

6.1 Clean-up Resource

```
$ kubectl delete sa my-service-account
$ kubectl delete pod my-pod
$ kubectl delete clusterrolebinding my-service-account-policy
```

7 SUMMARY

In this guide we Covered:

- Use serviceaccount in pod & access API inside a pod
- Disable serviceaccount mount
- Provide access to service account using RBAC