SECURING KUBERNETES

I. Service Account

Create User

Create Service Account

- Each namespace has a default ServiceAccount, named default

```
kubectl get sa --all-namespaces | grep default
```

```
default          default                           1        5m49s
kube-node-lease   default                           1        5m49s
kube-public      default                           1        5m49s
kube-system      default                           1        5m49s
```

- Let’s inspect the ServiceAccount named default of the default namespace

```
kubectl get sa default -o yaml
```

```
apiVersion: v1
kind: ServiceAccount
metadata:
  creationTimestamp: "2020-07-27T00:24:14Z"
  name: default
  namespace: default
  resourceVersion: "254"
  selfLink: /api/v1/namespaces/default/serviceaccounts/default
  uid: 0468392f-1efa-442e-8583-3d96a8df7666
secrets:
- name: default-token-grng7
```

- We can see here that a Secret is provided to this ServiceAccount.

```
kubectl get secret default-token-grng7 -o yaml
```

```

apiVersion: v1
data:
  ca.crt: <>
  namespace: ZGVmYXVsdA==
  token: <>
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: default
    kubernetes.io/service-account.uid: 0468392f-1efa-442e-8583-3d96a8df7666
  creationTimestamp: "2020-07-27T00:24:14Z"
  name: default-token-grng7
  namespace: default
  resourceVersion: "251"
  selfLink: /api/v1/namespaces/default/secrets/default-token-grng7
  uid: eb46dd28-0e97-48fd-9ac3-48d91f6be4db
type: kubernetes.io/service-account-token
```

- Decode base64

```
echo $(kubectl get secret default-token-grng7 --template={{.data.token}} | base64 --decode )
```

- Check how service account apply to a pod

```
https://github.com/hungtran84/k8s-cka/blob/master/d1_managing_cluster/03_RBAC/pod-noserviceaccount.yaml
kubectl apply -f pod-noserviceaccount.yaml

kubectl get po/pod-default -o yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    cni.projectcalico.org/podIP: 10.48.0.3/32
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"name":"pod-default","namespace":"default"},"spec":{"containers":[{"command":["sleep","10000"],"image":"alpine:3.9","name":"alpine"}]}}
    kubernetes.io/limit-ranger: 'LimitRanger plugin set: cpu request for container
      alpine'
  creationTimestamp: "2020-07-27T00:34:33Z"
  name: pod-default
  namespace: default
  resourceVersion: "3729"
  selfLink: /api/v1/namespaces/default/pods/pod-default
  uid: 380d797b-ce6d-465e-b4eb-9cb804ba6eaf
spec:
  containers:
  - command:
    - sleep
    - "10000"
    image: alpine:3.9
    imagePullPolicy: IfNotPresent
    name: alpine
    resources:
      requests:
        cpu: 100m
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: default-token-grng7
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: gke-cluster-1-default-pool-f4ac809e-sx18
  priority: 0
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  serviceAccount: default
  serviceAccountName: default
  terminationGracePeriodSeconds: 30
  tolerations:
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 300
  - effect: NoExecute
```

```
    key: node.kubernetes.io/unreachable
    operator: Exists
    tolerationSeconds: 300
  volumes:
  - name: default-token-grng7
    secret:
      defaultMode: 420
      secretName: default-token-grng7
status:
  conditions:
  - lastProbeTime: null
    lastTransitionTime: "2020-07-27T00:34:34Z"
    status: "True"
    type: Initialized
  - lastProbeTime: null
    lastTransitionTime: "2020-07-27T00:34:36Z"
    status: "True"
    type: Ready
  - lastProbeTime: null
    lastTransitionTime: "2020-07-27T00:34:36Z"
    status: "True"
    type: ContainersReady
  - lastProbeTime: null
    lastTransitionTime: "2020-07-27T00:34:33Z"
    status: "True"
    type: PodScheduled
  containerStatuses:
  - containerID:
docker://19f7df29614f0d9326441ea1028a51fd9632928322688bd77cac53908fb7358e
    image: alpine:3.9
    imageID: docker-
pullable://alpine@sha256:65b3a80ebe7471beecbc090c5b2cdd0aafeaefa0715f8f12e40
dc918a3a70e32
    lastState: {}
    name: alpine
    ready: true
    restartCount: 0
    started: true
    state:
      running:
        startedAt: "2020-07-27T00:34:35Z"
  hostIP: 10.148.0.14
  phase: Running
  podIP: 10.48.0.3
  podIPs:
  - ip: 10.48.0.3
  qosClass: Burstable
  startTime: "2020-07-27T00:34:34Z"
```

- Important things to note here:
  * The serviceAccountName key is set with the name of the default ServiceAccount.
  * The information of the ServiceAccount is mounted inside the container of the Pod, through the usage of volume, in /var/run/secrets/kubernetes.io/serviceaccount

- Anonymous call of the API server

```
kubectl exec -ti pod-default -- sh
apk add --update curl
curl https://kubernetes/api --insecure
```

```
/ # curl https://kubernetes/api --insecure
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Failure",
  "message": "forbidden: User \"system:anonymous\" cannot get path \"/api\"",
  "reason": "Forbidden",
  "details": {

  },
  "code": 403
}/
```

- Call using the ServiceAccount token

```
TOKEN=$(cat /run/secrets/kubernetes.io/serviceaccount/token)
curl -H "Authorization: Bearer $TOKEN" https://kubernetes/api/v1/ --insecure
```

```
{
```

```
"kind": "APIResourceList",
"groupVersion": "v1",
"resources": [
  {
    "name": "bindings",
    "singularName": "",
    "namespaced": true,
    "kind": "Binding",
    "verbs": [
      "create"
    ]
  },
  {
    "name": "componentstatuses",
    "singularName": "",
    "namespaced": false,
    "kind": "ComponentStatus",
    "verbs": [
      "get",
      "list"
    ],
    "shortNames": [
      "cs"
    ]
  },
  {
    "name": "configmaps",
    "singularName": "",
    "namespaced": true,
    "kind": "ConfigMap",
    "verbs": [
      "create",
      "delete",
      "deletecollection",
      "get",
      "list",
      "patch",
      "update",
      "watch"
    ],
    "shortNames": [
      "cm"
    ],
    "storageVersionHash": "qFsyl6wFWjQ="
    },
    {
      "name": "endpoints",
      "singularName": "",
      "namespaced": true,
      "kind": "Endpoints",
      "verbs": [
        "create",
        "delete",
        "deletecollection",
        "get",
        "list",
        "patch",
        "update",
        "watch"
      ],
      "shortNames": [
        "ep"
      ],
      "storageVersionHash": "fWeeMqaN/OA="
    },
    {
      "name": "events",
      "singularName": "",
      "namespaced": true,
      "kind": "Event",
      "verbs": [
        "create",
        "delete",
        "deletecollection",
        "get",
        "list",
        "patch",
        "update",
        "watch"
      ],
      "shortNames": [
        "ev"
      ],
      "storageVersionHash": "r2yiGXH7wu8="
    },
    {
      "name": "limitranges",
      "singularName": "",
    "namespaced": true,
    "kind": "LimitRange",
    "verbs": [
      "create",
      "delete",
      "deletecollection",
      "get",
      "list",
      "patch",
      "update",
      "watch"
    ],
    "shortNames": [
      "limits"
    ],
    "storageVersionHash": "EBKMFVe6cwo="
  },
  {
    "name": "namespaces",
    "singularName": "",
    "namespaced": false,
    "kind": "Namespace",
    "verbs": [
      "create",
      "delete",
      "get",
      "list",
      "patch",
      "update",
      "watch"
    ],
    "shortNames": [
      "ns"
    ],
    "storageVersionHash": "Q3oi5N2YM8M="
  },
  {
    "name": "namespaces/finalize",
    "singularName": "",
    "namespaced": false,
    "kind": "Namespace",
    "verbs": [
      "update"
    ]
    },
    {
      "name": "namespaces/status",
      "singularName": "",
      "namespaced": false,
      "kind": "Namespace",
      "verbs": [
        "get",
        "patch",
        "update"
      ]
    },
    {
      "name": "nodes",
      "singularName": "",
      "namespaced": false,
      "kind": "Node",
      "verbs": [
        "create",
        "delete",
        "deletecollection",
        "get",
        "list",
        "patch",
        "update",
        "watch"
      ],
      "shortNames": [
        "no"
      ],
      "storageVersionHash": "XwShjMxG9Fs="
    },
    {
      "name": "nodes/proxy",
      "singularName": "",
      "namespaced": false,
      "kind": "NodeProxyOptions",
      "verbs": [
        "create",
        "delete",
        "get",
        "patch",
        "update"
      ]
```

```
   },
   {
    "name": "nodes/status",
    "singularName": "",
    "namespaced": false,
    "kind": "Node",
    "verbs": [
     "get",
     "patch",
     "update"
    ]
   },
   {
    "name": "persistentvolumeclaims",
    "singularName": "",
    "namespaced": true,
    "kind": "PersistentVolumeClaim",
    "verbs": [
     "create",
     "delete",
     "deletecollection",
     "get",
     "list",
     "patch",
     "update",
     "watch"
    ],
    "shortNames": [
     "pvc"
    ],
    "storageVersionHash": "QWTyNDq0dC4="
   },
   {
    "name": "persistentvolumeclaims/status",
    "singularName": "",
    "namespaced": true,
    "kind": "PersistentVolumeClaim",
    "verbs": [
     "get",
     "patch",
     "update"
    ]
   },
   {
    "name": "persistentvolumes",
    "singularName": "",
    "namespaced": false,
    "kind": "PersistentVolume",
    "verbs": [
      "create",
      "delete",
      "deletecollection",
      "get",
      "list",
      "patch",
      "update",
      "watch"
    ],
    "shortNames": [
      "pv"
    ],
    "storageVersionHash": "HN/zwEC+JgM="
  },
  {
    "name": "persistentvolumes/status",
    "singularName": "",
    "namespaced": false,
    "kind": "PersistentVolume",
    "verbs": [
      "get",
      "patch",
      "update"
    ]
  },
  {
    "name": "pods",
    "singularName": "",
    "namespaced": true,
    "kind": "Pod",
    "verbs": [
      "create",
      "delete",
      "deletecollection",
      "get",
      "list",
      "patch",
      "update",
      "watch"
```

```
   ],
   "shortNames": [
     "po"
   ],
   "categories": [
     "all"
   ],
   "storageVersionHash": "xPOwRZ+Yhw8="
  },
  {
   "name": "pods/attach",
   "singularName": "",
   "namespaced": true,
   "kind": "PodAttachOptions",
   "verbs": [
     "create",
     "get"
   ]
  },
  {
   "name": "pods/binding",
   "singularName": "",
   "namespaced": true,
   "kind": "Binding",
   "verbs": [
     "create"
   ]
  },
  {
   "name": "pods/eviction",
   "singularName": "",
   "namespaced": true,
   "group": "policy",
   "version": "v1beta1",
   "kind": "Eviction",
   "verbs": [
     "create"
   ]
  },
  {
   "name": "pods/exec",
   "singularName": "",
   "namespaced": true,
   "kind": "PodExecOptions",
   "verbs": [
     "create",
     "get"
   ]
  },
  {
   "name": "pods/log",
   "singularName": "",
   "namespaced": true,
   "kind": "Pod",
   "verbs": [
     "get"
   ]
  },
  {
   "name": "pods/portforward",
   "singularName": "",
   "namespaced": true,
   "kind": "PodPortForwardOptions",
   "verbs": [
     "create",
     "get"
   ]
  },
  {
   "name": "pods/proxy",
   "singularName": "",
   "namespaced": true,
   "kind": "PodProxyOptions",
   "verbs": [
     "create",
     "delete",
     "get",
     "patch",
     "update"
   ]
  },
  {
   "name": "pods/status",
   "singularName": "",
   "namespaced": true,
   "kind": "Pod",
   "verbs": [
     "get",
     "patch",
     "update"
   ]
  },
  {
   "name": "podtemplates",
   "singularName": "",
   "namespaced": true,
   "kind": "PodTemplate",
   "verbs": [
     "create",
     "delete",
     "deletecollection",
     "get",
     "list",
     "patch",
     "update",
     "watch"
   ],
   "storageVersionHash": "LIXB2x4IFpk="
  },
  {
   "name": "replicationcontrollers",
   "singularName": "",
   "namespaced": true,
   "kind": "ReplicationController",
   "verbs": [
     "create",
     "delete",
     "deletecollection",
     "get",
     "list",
     "patch",
     "update",
     "watch"
   ],
   "shortNames": [
     "rc"
   ],
   "categories": [
     "all"
   ],
   "storageVersionHash": "Jond2If31h0="
  },
  {
   "name": "replicationcontrollers/scale",
   "singularName": "",
   "namespaced": true,
   "group": "autoscaling",
   "version": "v1",
   "kind": "Scale",
   "verbs": [
     "get",
     "patch",
     "update"
   ]
  },
  {
   "name": "replicationcontrollers/status",
   "singularName": "",
   "namespaced": true,
   "kind": "ReplicationController",
   "verbs": [
     "get",
     "patch",
     "update"
   ]
  },
  {
   "name": "resourcequotas",
   "singularName": "",
   "namespaced": true,
   "kind": "ResourceQuota",
   "verbs": [
     "create",
     "delete",
     "deletecollection",
     "get",
     "list",
     "patch",
     "update",
     "watch"
   ],
   "shortNames": [
     "quota"
   ],
   "storageVersionHash": "8uhSgffRX6w="
  },
  {
   "name": "resourcequotas/status",
   "singularName": "",
   "namespaced": true,
   "kind": "ResourceQuota",
   "verbs": [
     "get",
     "patch",
     "update"
   ]
  },
  {
   "name": "secrets",
   "singularName": "",
   "namespaced": true,
   "kind": "Secret",
   "verbs": [
     "create",
     "delete",
     "deletecollection",
     "get",
     "list",
     "patch",
     "update",
     "watch"
   ],
   "storageVersionHash": "S6u1pOWzb84="
  },
  {
   "name": "serviceaccounts",
   "singularName": "",
   "namespaced": true,
   "kind": "ServiceAccount",
   "verbs": [
     "create",
     "delete",
     "deletecollection",
     "get",
     "list",
     "patch",
     "update",
     "watch"
   ],
   "shortNames": [
      "sa"
    ],
    "storageVersionHash": "pbx9ZvyFpBE="
  },
  {
    "name": "serviceaccounts/token",
    "singularName": "",
    "namespaced": true,
    "group": "authentication.k8s.io",
    "version": "v1",
    "kind": "TokenRequest",
    "verbs": [
      "create"
    ]
  },
  {
    "name": "services",
    "singularName": "",
    "namespaced": true,
    "kind": "Service",
    "verbs": [
      "create",
      "delete",
      "get",
      "list",
      "patch",
      "update",
      "watch"
    ],
    "shortNames": [
      "svc"
    ],
    "categories": [
      "all"
    ],
    "storageVersionHash": "0/CO1lhkEBI="
  },
  {
    "name": "services/proxy",
    "singularName": "",
    "namespaced": true,
    "kind": "ServiceProxyOptions",
    "verbs": [
      "create",
        "delete",
        "get",
        "patch",
        "update"
      ]
    },
    {
      "name": "services/status",
      "singularName": "",
      "namespaced": true,
      "kind": "Service",
      "verbs": [
        "get",
        "patch",
        "update"
      ]
    }
  ]
```

- Let’s now try something more ambitious, and use this token to list all the Pods within the default namespace

```
curl -H "Authorization: Bearer $TOKEN"
https://kubernetes/api/v1/namespaces/default/pods/ --insecure
```

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

  },
  "status": "Failure",
  "message": "pods is forbidden: User \"system:serviceaccount:default:default\"
cannot list resource \"pods\" in API group \"\" in the namespace \"default\"",
  "reason": "Forbidden",
  "details": {
    "kind": "pods"
  },
  "code": 403
}/
```

2. Pod with service account:

```
# Create service  account:
Kubect create serviceaccount demo-sa
#Create pod with sa:
https://github.com/hungtran84/k8s-
cka/blob/master/d1_managing_cluster/03_RBAC/pod-with-sa.yaml
kubectl apply –f pod-with-sa.yaml
apiVersion: v1
kind: Pod
metadata:
 name: pod-demo-sa
spec:
 serviceAccountName: demo-sa
 containers:
 - name: alpine
   image: alpine:3.9
   command:
    - "sleep"
    - "10000"



#get the pod with sa:
Kubectl describe pod pod-demo-sa
```

3. Create User:

```
# Create user demouser with role permission developer only on the pod
#Generate the ssl certificate for demoUser
openssl genrsa -out demoUser.key 2048

openssl req -new -key demoUser.key -out demoUser.csr

#Get the csr file content
CSR=$(cat demoUser.csr | base64 | tr -d "\n")
```

```
Create CSR request
cat <<EOF | kubectl apply -f -
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: demoUser
spec:
  groups:
  - system:authenticated
  request: $CSR
  usages:
  - client auth
EOF



# list of csr:
kubectl get csr
#Approve the demoUser CSR
kubectl certificate approve demoUser

#get the certification:
kubectl get csr demoUser -o jsonpath='{.status.certificate}'| base64 -d >
demoUser.crt

#Create the role for developer
kubectl create role developer --verb=create --verb=get --verb=list --verb=update --
verb=delete --resource=pods

#Create rolebinding for developer
kubectl create rolebinding developer-binding-demoUser --role=developer --
user=demoUser

#Test with the demo user
k get pod --as demoUser
k get deployment --as demoUser



#Add new user to the kubeconfig (options):
```

```
kubectl config set-credentials demoUser --client-key=demoUser.key --client-certificate=demoUser.crt --embed-certs=true

kubectl config set-context demoUser --cluster=kubernetes --user=demoUser
kubectl config use-context demoUser
```

4. Pulling a Container from a Private Container Registry

```
#Create secrets
kubectl create secret docker-registry private-reg-cred \
    --docker-server=https://index.docker.io/v1/ \
    --docker-username=$DOCKERACC \
    --docker-password=$PASSWORD \
    --docker-email=$EMAIL


secret/private-reg-cred created


kubectl apply -f deployment-private-registry.yaml
deployment.apps/hello-world-private-registry created

kubectl get pods hello-world

Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-r5klh:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-r5klh
    Optional:    false
QoS Class:       Guaranteed
Node-Selectors:  <none>
```

```
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type    Reason     Age   From                                       Message
  ----    ------     ----  ----                                       -------
  Normal  Scheduled  8s    default-scheduler                          Successfully
assigned default/hello-world-private-registry-76f8ddc944-p2jsv to gke-cluster-1-
default-pool-990b49f7-bzft
  Normal  Pulling    7s    kubelet, gke-cluster-1-default-pool-990b49f7-bzft  Pulling
image "votiethuy/hello:latest"
  Normal  Pulled     4s    kubelet, gke-cluster-1-default-pool-990b49f7-bzft
Successfully pulled image "votiethuy/hello:latest"
  Normal  Created    4s    kubelet, gke-cluster-1-default-pool-990b49f7-bzft  Created
container hello-world
  Normal  Started    4s    kubelet, gke-cluster-1-default-pool-990b49f7-bzft  Started
container hello-world
```