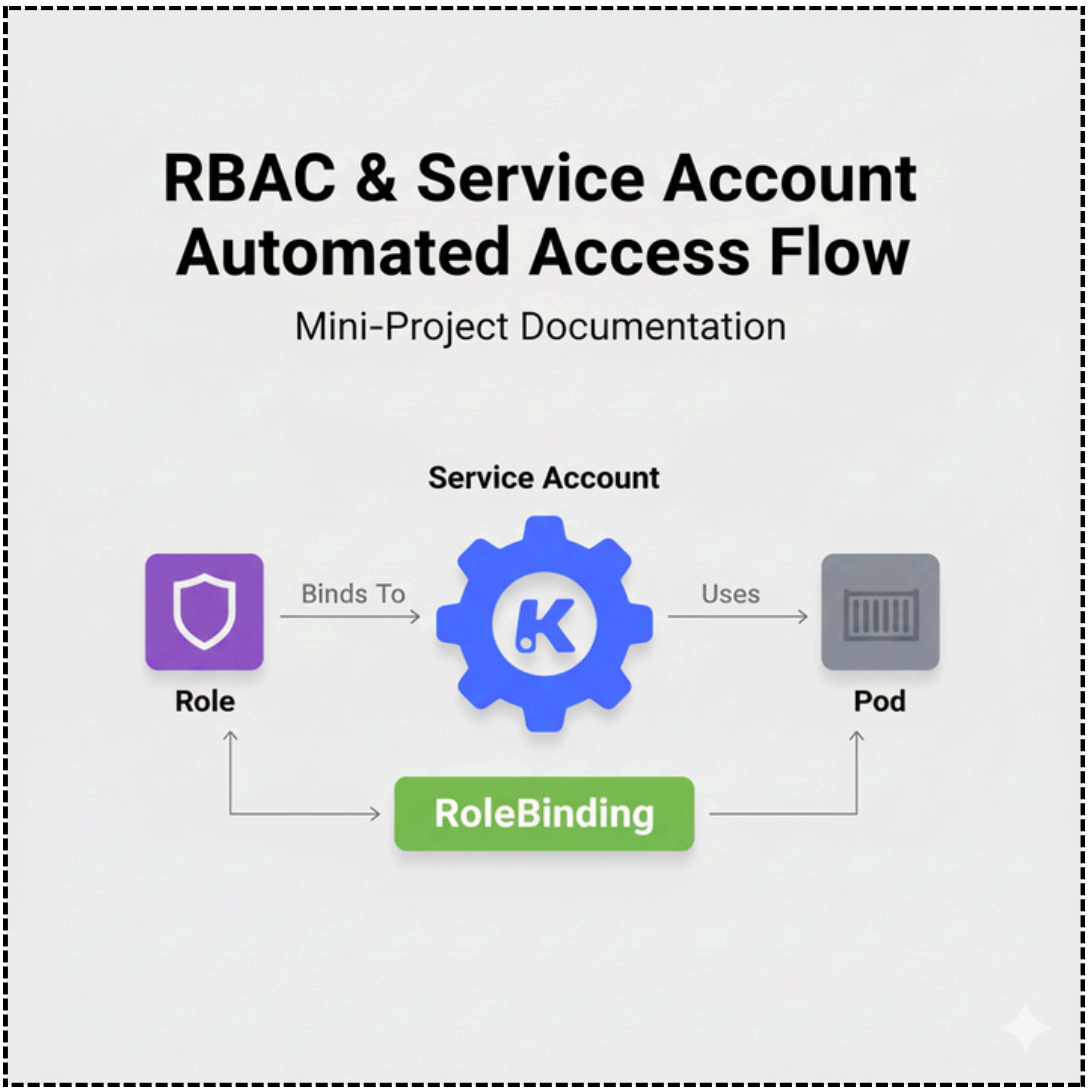
kubernetes

MINI PROJECT

RBAC & SERVICE ACCOUNT AUTOMATED ACCESS



kubernetes

🟦 Project Overview

This project demonstrates the creation of a secure and automated access mechanism to a Kubernetes cluster using **Service Accounts** and **Role-Based Access Control (RBAC).**

Instead of using the cluster administrator's credentials, a dedicated service account with limited privileges is created, and a custom KUBECONFIG file is generated for safe, token-based authentication.

This setup is ideal for enabling automation tools, CI/CD pipelines, or applications to interact securely with a Kubernetes cluster.

🎯 Objective

- To create a non-admin Kubernetes user using a Service Account.
- To apply RBAC policies to restrict access to specific resources (Pods only).
- To generate a custom kubeconfig file for secure, token-based cluster access.
- To validate the principle of least privilege by testing access permissions.

🧰 Environment Setup

- Platform: Killercoda Kubernetes Playground
- Kubernetes Version: v1.29+
- Tools Used: kubectl CLI
- Namespace: myexample

Concepts Involved

Concept	Description
Service Account	A Kubernetes identity used by applications or scripts instead of humans.
RBAC (Role-Based Access Control)	A security mechanism that defines who can do what within a cluster.
Role	Specifies a set of allowed actions (verbs) on specific resources.
RoleBinding	Associates a Role with a specific Service Account.
Kubeconfig File	A configuration file containing cluster, user, and authentication information.

kubernetes

⚙ Implementation Steps

Step 1: Create a Namespace

Isolate resources in a separate namespace to demonstrate scoped RBAC control.

```
controlplane:~$ kubectl create namespace myexample
namespace/myexample created
controlplane:~$ kubectl get namespaces
NAME                   STATUS   AGE
default                Active   22d
kube-node-lease        Active   22d
kube-public            Active   22d
kube-system            Active   22d
local-path-storage     Active   22d
myexample              Active   6s
controlplane:~$
```

Step 2: Create a Service Account

Create a new service account named project-sa inside the project namespace.

```
controlplane:~$ cat serviceaccount.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: myexample-sa
  namespace: myexample

controlplane:~$
```

Verify:
cmd:kubectl get sa -n myexample

```
controlplane:~$ kubectl apply -f serviceaccount.yaml
serviceaccount/myexample-sa created
controlplane:~$ kubectl -n myexample get sa
NAME           SECRETS   AGE
default        0         111s
myexample-sa   0         17s
```

Step 3: Create a Role with Limited Permissions

Define a role that allows read-only access to pods.

role.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
 namespace: myexample
 name: pod-manager
rules:
- apiGroups: [""]
 resources: ["pods"]
 verbs: ["get", "list", "watch",”Create”]

```
controlplane:~$ cat role.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: myexample
  name: pod-manager
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "list", "watch", "create"]

controlplane:~$
```

Apply:
cmd:kubectl apply -f role.yaml

```
controlplane:~$ kubectl apply -f role.yaml
role.rbac.authorization.k8s.io/pod-manager created
```

Verify:
cmd:kubectl -n myexample get role.

```
controlplane:~$ kubectl -n myexample get role
NAME          CREATED AT
pod-manager   2025-11-11T09:09:19Z
controlplane:~$
controlplane:~$
```

Step 4: Create a RoleBinding

Bind the role to the service account.

 # rolebinding.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-manager-binding
  namespace: myexample
subjects:
- kind: ServiceAccount
  name: myexample-sa
  namespace: myexample
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

```
controlplane:~$ cat rolebinding.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: pod-manager-binding
  namespace: myexample
subjects:
- kind: ServiceAccount
  name: myexample-sa
  namespace: myexample
roleRef:
  kind: Role
  name: pod-manager
  apiGroup: rbac.authorization.k8s.io

controlplane:~$
```

Apply:

cmd: kubectl apply -f rolebinding.yaml

```
controlplane:~$ kubectl apply -f rolebinding.yaml
rolebinding.rbac.authorization.k8s.io/pod-manager-binding created
```

Verify:

cmd:kubectl -n myexample get rolebinding.

```
controlplane:~$
controlplane:~$ kubectl -n myexample get rolebinding
NAME                  ROLE               AGE
pod-manager-binding   Role/pod-manager   47m
controlplane:~$
controlplane:~$
controlplane:~$
```

Step 5: Generate a Token for the Service Account

Since Kubernetes v1.24+, secrets are no longer auto-created.
So, use the following command:

cmd: kubectl create -n myexample token myexample-sa

```
controlplane:~$ kubectl -n myexample create token myexample-sa
eyJhbGciOiJSUzI1NiIsImtpZCI6InN0YXdpc2xNS3NVU3ZqZUlVU3ZRNG00WjdaVDlrX1d6QjVWNzJrN1RuNjQifQ.eyJhdWQ
G9jYWwiXSwiZXhwIjoxNzYyODU1ODYyLCJpYXQiOjE3NjI4NTIyNjIsImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx
DMzLWI3YmQtNWRlMGJmZmJiNmQzIiwia3ViZXJuZXRlcy5pbyI6eyJuYW1lc3BhY2UiOiJteWV4YW1wbGUiLCJzZXJ2aWNlYWN
jAtM2NkOS00ZDZjLWI1MDUtMzhhMjQzMjEwZGZhIn19LCJuYmYiOjE3NjI4NTIyNjIsInN1YiI6InN5c3RlbTpzZXJ2aWNlYWN
JW0WdCyiIOfpQHwCzwJNH8oxihy0Cuycr2Ffgj2CayxVsM_XtNEOVe6ZiToNFWCJ4amKXQV0EXVCcf6EM0AvBPIgrsLXjL9zA7
wzWltz4iduutwEMlT7TteNw_WgSM0-6PGrcrOv2e54FewtjxYMprFRH-5DZS8Ru7XHOj0OaAVr48saLAn6jwmzkwDdw3PDtNlL
uvzF9iemdJKsA
```

This token is used for authentication.

Step 6: Get Cluster Server URL

Fetch cluster details needed for kubeconfig creation.

```
controlplane:~$ kubectl cluster-info
Kubernetes control plane is running at https://172.30.1.2:6443
CoreDNS is running at https://172.30.1.2:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

Step7: Create Custom Kubeconfig

```
controlplane:~$ cat sa-kubeconfig.yaml
apiVersion: v1
kind: Config
clusters:
- name: kubernetes
  cluster:
    certificate-authority: /etc/kubernetes/pki/ca.crt
    server: https://172.30.1.2:6443   # Replace with output from Step 6
users:
- name: myexample-sa
  user:
    token: eyJhbGciOiJSUzI1NiIsImtpZCI6InN0YXdpc2xNS3NVU3ZqZUlVU3ZRNG00WjdaVDlrX1d6QjVWNzJrN1RuNjQifQ.eyJhdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLm
NsdXN0ZXIubG9jYWwiXSwiZXhwIjoxNzYyODU1ODYyLCJpYXQiOjE3NjI4NTIyNjIsImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwianRpIjoiOGEyNDc0Mz
ItZDA5NC00MDMzLWI3YmQtNWRlMGJmZmJiNmQzIiwia3ViZXJuZXRlcy5pbyI6eyJuYW1lc3BhY2UiOiJteWV4YW1wbGUiLCJzZXJ2aWNlYWNjb3VudCI6eyJuYW1lIjoibXlleGFtcGxlLXNhIiwidWlkIj
oiNWNjNjY4ZjAtM2NkOS00ZDZjLWI1MDUtMzhhMjQzMjEwZGZhIn19LCJuYmYiOjE3NjI4NTIyNjIsInN1YiI6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDpteWV4YW1wbGU6bXlleGFtcGxlLXNhIn0.CarRVC
kDMO5AdeuQsJW0WdCyiIOfpQHwCzwJNH8oxihy0Cuycr2Ffgj2CayxVsM_XtNEOVe6ZiToNFWCJ4amKXQV0EXVCcf6EM0AvBPIgrsLXjL9zA7vOOq5_yTG6T7s4ETMCuMmC2fjUx2bQoDACH9g5X75UnEfEQ
dSN8amErqzKwzWltz4iduutwEMlT7TteNw_WgSM0-6PGrcrOv2e54FewtjxYMprFRH-5DZS8Ru7XHOj0OaAVr48saLAn6jwmzkwDdw3PDtNlLPcFVzvzLAyrrBP3N0GhsUbJ18TToVjmkqfLs2YxHIk0O_-i
p1UqllI9WbYuvzF9iemdJKsA
contexts:
- name: myexample-context
  context:
    cluster: kubernetes
    namespace: myexample
    user: myexample-sa
current-context: myexample-context

controlplane:~$
```

Replace the token value with your real token

Save and exit

Step8:Test the Kubeconfig

Shows the current context being used from your custom kubeconfig file.

cmd:KUBECONFIG=/root/sa-kubeconfig.yaml kubectl config current-context

```
controlplane:~$ KUBECONFIG=/root/sa-kubeconfig.yaml kubectl config current-context
myexample-context
controlplane:~$
controlplane:~$
controlplane:~$
```

Lists Pods in the namespace

Lists all pods in the myexample namespace using your service account's kubeconfig.

cmd: KUBECONFIG=/root/sa-kubeconfig.yaml kubectl get pods -n myexample

```
controlplane:~$ KUBECONFIG=/root/sa-kubeconfig.yaml kubectl get pods -n myexample
No resources found in myexample namespace.
```

Tests whether the service account can create a new pod.

cmd: KUBECONFIG=/root/sa-kubeconfig.yaml kubectl run pod1 --image=httpd -n myexample

```
No resources found in myexample namespace.
controlplane:~$ KUBECONFIG=/root/sa-kubeconfig.yaml kubectl run pod1 --image=httpd -n myexample
pod/pod1 created
```

If it succeeds, your service account has correct create permissions.

Checks if a specific user/service account has permission to list pods in a namespace.

cmd: kubectl auth can-i list pods --as=system:serviceaccount:myexample:myexample-sa -n myexample

```
controlplane:~$ kubectl auth can-i list pods --as=system:serviceaccount:myexample:myexample-sa -n myexample
yes
controlplane:~$
controlplane:~$
controlplane:~$
```

Meaning this service account can list pods.

Checks whether the service account has permission to delete pods.

cmd: kubectl auth can-i delete pods --as=system:serviceaccount:myexample:myexample-sa -n myexample

```
controlplane:~$ kubectl auth can-i delete pods --as=system:serviceaccount:myexample:myexample-sa -n myexample
no
controlplane:~$
controlplane:~$
controlplane:~$
```

That confirms RBAC is working correctly, restricting unauthorized actions.

kubernetes

🧪 Verification Summary

Component	Name	Purpose	Status
Namespace	myexample	Resource isolation	✅
Service Account	myexample-sa	Non-admin identity	✅
Role	pod-manager	Read-only pod access	✅
RoleBinding	read-manager-binding	Links SA & Role	✅
Token	JWT	Authenticates the SA	✅
Custom Kubeconfig	myexample sa.kubeconfig	For secure access	✅

🧠 Key Learnings

Concept	Learning
RBAC Implementation	How to define Roles and RoleBindings in Kubernetes
Service Accounts	How non-human users can access the cluster securely
Security Practice	Avoid using admin credentials for automation
Least Privilege Principle	Grant only the minimum access required
Custom Kubeconfig Creation	Build kubeconfig manually using tokens and CA data

kubernetes

🏁 Project Outcome

✅ Successfully implemented RBAC and Service Account-based automated kubectl access

✅ Verified that the Service Account can only perform read-only operations on pods

✅ Ensured secure, token-based, non-admin cluster access

Thank you