

Troubleshooting – Kubernetes PodSecurityPolicies



Before starting – Check if common pitfalls are avoided:

- admission controller active
- PSP, (Cluster-)Role and (Cluster-)RoleBinding applied
- Pods started *after* enabling admission controller, applying PSP, (Cluster-)Role and (Cluster-)RoleBinding
- correct apiGroup for (Cluster-)Role, depending on k8s version
 - < 1.16: apiGroups: [extensions]
 - >= 1.16: apiGroups: [policy]

START

```
$ kubectl get pod
```

Is the pod there?

yes

no

Pod managed by Deployment?

yes

no

```
$ kubectl describe replicaset | grep <deployment>
```

```
$ kubectl describe <sts|ds|cron|job>
```

Evaluate warnings in events

Service Account not authorized for any PSP?

yes

no

Authorize ServiceAccount of Pods to use PSP via RBAC

STOP

Error could be due to settings in SecurityContext that violate PSP, e.g.

- user or group ID that is not in range of PSP.
- explicitly added capability not allowed by PSP.

Fix by

- getting rid of securityContext setting
- using other (trusted) image
- adapting PSP/using less restrictive PSP (last resort)

STOP

```
$ kubectl get pod <pod> \
-o jsonpath='{.metadata.annotations.kubernetes.io/psp}'
```

Expected PSP?

yes

no

```
$ kubectl get pod <pod> \
-o jsonpath='{.spec.serviceAccountName}'
```

Expected ServiceAccount?

yes

no

Set ServiceAccount in Pod

STOP

```
$ kubectl auth can-i use <psp> \
--as=system:serviceaccount:<namespace>:<serviceAccount>
```

Can use?

yes

no

ServiceAccount seems to be authorized for other PSPs. Read docs on policy order to find a solution.

STOP

Check (Cluster-)Role/(Cluster-)RoleBinding:

- Typos
- Role → RoleBinding
- ServiceAccount
- Namespace
- wrong apiGroup for k8s version

Hints:

- Query all roles of ServiceAccount (kubectl plugin)


```
kubectl rbac-lookup <serviceAccount>
```
- Show who is authorized for psp (kubectl plugin)


```
kubectl who-can use psp <psp>
```

STOP

Pod Status CrashLoopBackOff?

yes

no

```
$ kubectl logs <pod>
```

Error could be due to be missing capability or not running as root. Evaluate logs and ...

Pod Status CreateContainerConfigError?

yes

no

```
$ kubectl describe <pod>
```

Error could be due to Rule runAsNonRoot and running as root. Evaluate warnings in events and ...

There should be no problem

STOP

... fix security settings via

- securityContext (set UID, add/drop capabilities, etc.)
- volumeMounts (when using read-only filesystem)
- container image, e.g. via
 - USER instruction in Dockerfile or
 - adapting image so it does not need capabilities or has proper file permissions
 - adapting PSP/using less restrictive PSP (last resort)

STOP

Hints for testing if settings are effective:

- Query capabilities of container:


```
capsh --decode="$(kubectl exec <pod> cat /proc/1/status | grep CapBnd | cut -d':' -f2)"
```
- Check if seccomp active ("2" means active)


```
kubectl exec <pod> cat /proc/1/status | grep Seccomp
```
- Check if privilege escalation is possible:


```
kubectl exec <pod> cat /proc/1/status | grep NoNewPrivs
```
- Simple test for read-only filesystem (expected result -touch: cannot touch 'a': Read-only file system)


```
kubectl exec <pod> touch a
```
- Query user and group of container:


```
kubectl exec <pod> id
```