Security Contexts

Security Contexts allow you to restrict the privilege available to a pod's containers.

These can be applied at either a pod level or a container level.

If one of these items is set at both the pod level and in a container, the container's value overrides the pod's value.

Reference:

https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#podsecuritycontext-v1-core

https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#securitycontext-v1-core

Pod Security Contexts

Here are some of the items you can currently set at the pod level:

- `runAsUser` – user (UID) that the container's first process will start as.
- `runAsGroup` – group (GID) that the container's first process will start as.
- `runAsNonRoot` – if true, the pod's containers will not start if their images specify running as root.
- `seccompProfile` – seccomp settings that must be used by the containers in this pod.
- `supplementalGroups` – a list of groups added to the container's first process

Container Security Contexts

Here are some of the items you can currently set at the container level:

- `allowPrivilegeEscalation` – deactivates Set-UID, Set-GID, filesystem root capabilities & anything that allows process to have more privilege than their parent.
- `capabilities` – a list of root capabilities to add and/or drop
- `privileged` – if true, root in the container is root on the host
- `runAsUser` – user (UID) that the container's first process will start as.
- `runAsGroup` – group (GID) that the container's first process will start as.
- `runAsNonRoot` – if true, the pod's containers will not start if their images specify running as root.
- `seccompProfile` – seccomp settings that must be used by the containers in this pod.
- `supplementalGroups` – a list of groups added to the container's first process

Pod Security Context

Items specified in the pod's securityContext section apply to all containers, but can be overriden by setting the same item in a container.

Container Security Context

Items specified in a specific container can override the same item specified in the pod's securityContext section.

```
apiVersion: v1
kind: Pod
metadata:
  name: pod
spec:
  securityContext:
    runAsNonRoot:  true
  containers:
  - name: ctr-1
    image: nginx
    securityContext:
      allowPrivilegeEscalation: false
      capabilities:
        add: ["CAP_NET_BIND_SERVICE"]
        drop: ["ALL"]
      runAsUser:  www-data
      runAsGroup: www-data
```

Mandating Security Contexts

Admission controllers like Pod Security Standards, Gatekeeper, Kyverno and Pod Security Policies allow you to require pods being created to have security contexts, with specific constraints on the values.

- Validating admission controllers can express a requirement, causing a pod to be rejected at the time of creation or edit.
- Mutating admission controllers can change pods that are being created, to add or alter these values.
- An admissions controller can do mutation first, then validation.