de.NBI - Cloud Usermeeting 2021

Introduction to Kubernetes III: RBAC/Security/Network policies

Sebastian Beyvers
sebastian.beyvers@computational.bio.uni-giessen.de
Marius Dieckmann
marius.dieckmann@computational.bio.uni-giessen.de

22 November 2021

Authorization in k8s

- Two types of accounts:
    - User
    - Serviceaccount
- Serviceaccounts are used in pods
    - Main use: Spawning new Pods
    - Can be mounted and used from programs in pods
- Authorization via RBAC (Role Based Access Control)
    - Defines ”Roles” that describe permissions for resources
    - Roles are bound to accounts (”subjects”)

Roles

- Roles define rights
- Two "Scopes":
  - Role: namespaced
  - ClusterRole: cluster scoped
- Uses verbs (get, list, create,...) and resources (pods, deployments,...) to define permission
- The assignment of a Role to a user/serviceaccounts is called Rolebinding (RB)
  - Rolebindings assign a role to collection of users, in a namespace
  - Cluster scoped rolebindings are called "ClusterRoleBindings"
  - ClusterRoles can be used in regular Rolebindings
    - Will only grant rights in the specified namespace

Privileges

- Users can create/update roles
  - Requires the same permissions (namespaced)
  - Or requires the ”escalate” permission (namespaced) [1.12+]
- Users can apply rolebindings
  - Requires the same permissions (namespaced)
  - Or requires the ”bind” permission (namespaced)
- K8s has some predefined ClusterRoles (admin, edit, view)
  - Can be used as default
  - Can also be used by users for namespaced permissions

Pod Security Policy (PSP)

- Controls Pod security aspects ("permissions of pods")
- Cluster scoped resource
- Can be used in (Cluster)Roles like any other resource
- Pod security settings are validated againt all available psp
- One of two important security measures in k8s (admin and user)

PSP - Settings

- Privileged
  - host device access
- Hostpath(s)
  - paths that can be mounted from the host
- Privilege escalation
  - uid/gid change permitted?
- RunAs[User,Group]
  - Forces the container to run as a specific user $\rightarrow$ avoid running container as root

Network policies I

- By default all Pods can communicate to each other
- Network policies restrict this network access
    - defaults are often applied by Admin
- Implemented in the network plugin
    - paths that can be mounted from the host
- Pod level firewall
- Introduces only very low latency
    - Sometimes in badly designed microservice application

Network policies II

- Rules are applied based on labels
  - e.g pod or/and namespace labels
- Rules can apply to:
  - Ingress
  - Egress
- Rules can be based on:
  - Labels
  - ipBlocks
  - DNS-Names
  - Ports
  - ...

Network policies – Best practices

- Use them
  - Check the default policy / behavior
- Be careful with label selectors
  - Namespace labels should only be handled by Admins
- Secure and limit access to all non-public APIs
  - gRPC
  - REST
  - Thrift
  - ...
- Avoid stacking multiple network policies
  - Hard to understand the behavior

Questions?