Kubernetes WG multi-tenancy Deep Dive

KubeCon Europe 2018
May 4, 2018

Kubernetes Security Profile

KubeCon Europe 2018
May 4, 2018

(presenter) David Oppenheimer <davidopp@google.com>, Software Engineer, Google
Work was done by Yisui Hu <yisuihu@google.com>, Software Engineer, Google

Security Profile

Improve usability of Kubernetes security/multi-tenancy features

Today, to operate a secure/multi-tenant cluster

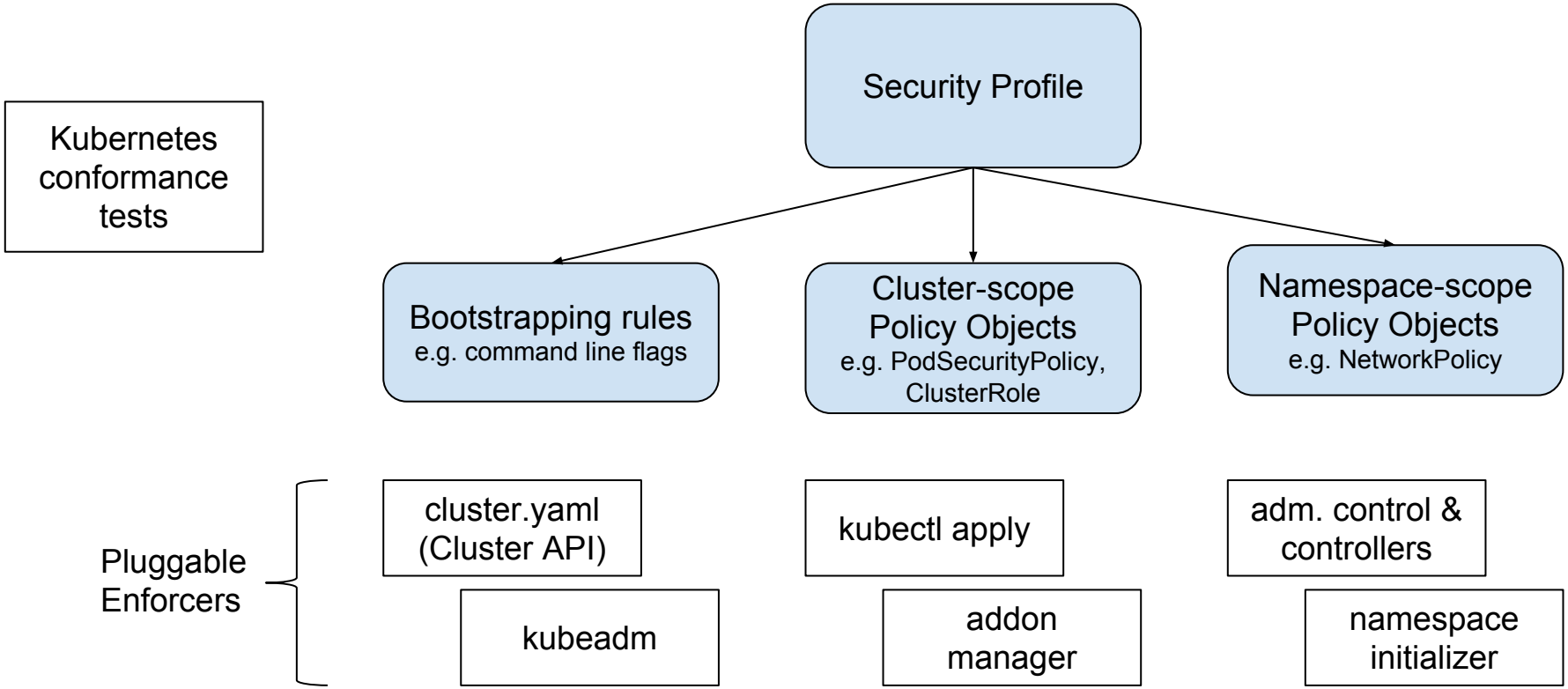
- cluster admins need to **understand security and Kubernetes** deeply
- policy **configuration is error-prone**, needs **tooling** to apply reproducibly
- policy configs must be **updated** as new Kubernetes features are added

Security Profile

Create a small menu of **versioned**, **community-curated policy profiles**, to enable **turnkey cluster creation** with desired **security and tenant isolation** (everywhere)

```
kubeadm init --security-profile=default-1.0-1
kubeadm init --security-profile=saas-multitenancy-1.0-1
kubeadm upgrade --security-profile=saas-multitenancy-1.0-1
```

Security Profile



Config As Code, API Design and You

rayc@google.com, Kubecon Europe 2018

Quick overview of config-as-code

- API objects represented as files
- Pipeline/Workflow
  a. Edit file, send for review, get approval/commit.
  b. Push to CI/CD staging repo
  c. Wait for tests to pass and the push to source-of-truth repo
  d. CI/CD invokes config management tool
  e. Config management tool calls platform APIs to reify changes to platform
  f. Optional: CI/CD may repeat steps c. thru e. Allowing slow rollout, deployment to staging then prod, etc.

Where things go wrong

- Errors at the last stage of the pipeline are difficult to deal with.
  - User sees CI/CD tool error wrapping a config management tool error wrapping the platform API error.
- Source repo now differs from running config.
  - These must be manually dealt with and many config management systems are not able to reconcile platform state with repo state.
  - Without diligence, these differences continue to exist onward.

Causes

Config management tools validation != platform API validation

- APIs and config management tools have to be in lock step which is hard.
- Pipelines that aim to test and validate separate the user from the actual deployment and reduce the ability to react quickly.

Approach

Goal: config management validation == platform API validation

- Platform API authors should make their validation code portable.
  - Release tools that can be run as pre-commit hooks.
  - These tools use the *same* validation code as the API so they rev together.
  - For complex validations offer dry-run/simulation modes that be run with lower privilege.
- Validation code should give very explicit errors on what is wrong with links to documentation describing requirements.
- Through this approach, human errors are caught before the initial commits.
- CI/CD pipeline is there to ensure pre-commit hooks have run.

Tenant Operator

Design Sketch: CRD

Lutz Behnke (lutz.behnke@haw-hamburg.de)
Christian Hüning (christianhuening@gmail.com)

Tenant Operator: Motivation

- Shared implementation for common functionality
  - Monitor resource use by tenant
  - Enforce limits (CPU, Mem, net Bandwidth, API Req/sec, # of declared objects)
- Partition concerns between common func and organisation specific
  - CRD represents leaf in arbitrary complex resource allocation schema, specific to each org.
  - Example: Current K8s-Gitlab-Integrator -> Gitlab specific implementation + TenantOp to map settings into actionable K8s configuration..
- Make opinionated behaviour explicit / configurable

Intended Mode of Operation

- External management system creates Tenant CRDs
  - That’s your Source of Truth (company IDM, Gitlab, Github etc.)
- Tenant Operator + TenantResourceQuota admission controller enforce resource limits.

Definition: Tenant

A Tenant consists of a set of namespaces in which any account with sufficient permissions may create Kubernetes objects. The number of these objects and their resources consumption are totaled over all namespaces of a tenant.

For each metric (CPU, mem, # of objects, etc.) a limit may be set either per namespace or for the total per tenant.

Note: Limits in specific namespaces are not part of the Tenant Operator as they can already be enforced (e.g. the LimitRanger).

Tenant CRD Contents

- Unique name of the tenant.
- A list of namespaces.
  - each namespace may have an external name
- Resource quotas for resources. These should at least include
  - Cores
  - Memory
  - API storage volumes
  - Requests per time unit
  - Number of objects
  - PodBandwidthResources
- Optionally it may contain a map of user names and roles
- Future: Time to live for object, until it is deleted.

Tenant Controller Docs:

Multi tenant kubernetes today on HUAWEI CLOUD

@kevin-wangzefeng

The way we use:

- HUAWEI COULD Container instance (serverless container) service **with k8s API exposed to end user**.
- Currently limited preview.

Tenant concept

- One namespace per tenant now
- no extra explicit tenant (API) definition in k8s.

Namespace

- Name generated by the platform (chosen solution).Or make the name historical unique, if allow end users to indicate name. (Imagine creating an avatar/character in a MMOG)
- Default quota set by CCI controller

API access

- k8s components talk to api server directly, end users and external clients talk to API-gateway
- RBAC and API rate limiting per tenant for external requests, enforced in API-gateway

Node

- Managed nodes, end user no access to the machine
- Secured container as container runtime

(continued)

Network

- Network API as CRD
- one network per namespace
- huawei container network implementation, l2 isolation
- Not blocking pod creation even network takes some time to be ready.

Storage

- Something alternative to emptydir introduced. Instead of evicting when exceeds limit, prevent from using more than claimed

Open issue: Pod up time, how to deal with upgrading

- kubelet inline upgrade, containers should no be restarted nor recreated
- os upgrade make node down. pod needs to be rrescheduled

Open issue: Root scope API access for end user

- CRD. Operator as plugin
- PV API. End user own the volumes in EVS, but cannot access correspond ing PV object.