WG Overview
Motivation (from Brian G)

- Kubernetes The Policy Framework
- Policies impose permissions, quotas, constraints, requirements, defaults, etc. on other resources
- What patterns should we adopt going forward?
  - Built in vs extensions
  - Extension using DSLs vs APIs
  - Domain-specific (scheduling policy) vs resource-specific (pod restriction)
  - Conventions across policy types: whitelists, blacklists, profiles, defaults, etc.
  - Cluster-level vs namespace-level
  - Policies vs component flags
- How do we provide policy defaults?
Motivation (from ourselves)

● Policy are needed and designed all over the place in Kubernetes.
● Policy description are domain specific in nature:
  ○ Not only in the sense Brian G meant (Kubernetes’ domain), but also in a larger context of usage (audit, security, storage, network, AI...), vertical adoption (finance, telco, pharma,...), languages, ...
  ○ Usually out of scope for WG description.
● Policy semantic and control mechanism is universal
  ○ Policy semantic: the underlying description of the policy description.
  ○ Policy control mechanism: life cycle of policy itself, and life cycle of elements defined in policy.
Motivation (Policy is needed in many places outside k8s)
WG Work Items
Policy WG Work Items Overview

● Running list of interested items
  ○ Multi-tenancy: https://github.com/kubernetes-sigs/multi-tenancy
  ○ Gatekeeper: https://docs.google.com/document/d/1A1-Q-1OMw3QODs1wT6eqfLTaqcGmgzAJAjJihiO3T48/edit#heading=h.rosd3aktk pys
  ○ PodSecurityPolicy Migration: https://github.com/kubernetes/enhancements/issues/5

● New Area Exploration
  ○ Policy as type system
  ○ Policy formal verification

● Case Studies
WG Running List 2019 - Multi-Tenancy Policy

Minimal Base version workflow:
Kubectl create -f newtenant.yaml

Kind:
tenant
Spec:
Tenant_name
Admin_contact
<NamespaceTemplateList>

Advanced Full version workflow (WIP):
Kubectl create -f newtenant.yaml

Kind:
tenantrequest
Spec:
tenantTemplateInstance
Self-service Namespace Creation
  ○ “kubectl create ns” by tenant admins without going over an indirect way through Tenant CRD and Tenant CRD controller.

Cluster-scoped Resources
  ○ the tenant admins may have permissions to create cluster scoped resources like PodSecurityPolicy

In a nutshell, help solving the CR population problem in the context of multi-tenancy
WG Running List 2019 - Multi-Tenancy Policy

- **Proposal**: Policy Engine -> Policy Compiler -> Tenant Policy object -> Resource Population (ns, podsec, network, rbac, ....)
- **Example**: OPA -> Gatekeeper (Tenant Policy Object -> Resource Population) -> General Kubernetes Cluster
- **Problem**: how to define the constraint for a population (when do we hit a wall and stop)
OPA Gatekeeper v3.0

- Validating admission.
  - Audit. CICD.
- Policy templates (Rego) and instances
- Policies stored in CRDs
- Audit results stored on policy CRDs
- Full architecture
- Google, Microsoft, Redhat, CBA, Styra
- "Gatekeeper" donated by Replicated
- Built with kubebuilder
v0.11: Native Integrations: WebAssembly progress

- WebAssembly (Wasm) is an instruction format for virtual machines
  - Provides a safe/efficient/portable runtime for policy evaluation
  - Goal: enable library embeddings of OPA policies in any language/runtime

- v0.10 added experimental Wasm stage to OPA

- v0.11 expands the fragment of Rego supported by the Wasm stage
  - All types of rules (ordered/unordered, default, partial sets/objects) now supported

- Example: open-policy-agent/contrib/wasm (CDN example)

openpolicyagent.org
PodSecurityPolicy Migration

- Explore the possibility of using Gatekeeper for PSP

```go
// PodSecurityPolicy specifies the policy enforced.
type PodSecurityPolicySpec struct {
    // Privileged determines if a pod can request to be run as privileged.
    Privileged bool `json:"privileged,omitempty"
    // Capabilities is a list of capabilities that can be added.
    Capabilities []api.Capability `json:"capabilities,omitempty"
    // Volumes allows and disallows the use of different types of volume plugins.
    Volumes VolumeSecurityPolicy `json:"volumes,omitempty"
    // HostNetwork determines if the policy allows the use of HostNetwork in the pod spec.
    HostNetwork bool `json:"hostNetwork,omitempty"
    // HostPorts determines which host port ranges are allowed to be exposed.
    HostPorts []HostPortRange `json:"hostPorts,omitempty"
    // HostPID determines if the policy allows the use of HostPID in the pod spec.
    HostPID bool `json:"hostPID,omitempty"
    // HostIPC determines if the policy allows the use of HostIPC in the pod spec.
    HostIPC bool `json:"hostIPC,omitempty"
    // SELinuxContext is the strategy that will dictate the allowable labels that may be set.
    SELinuxContext SELinuxContextStrategyOptions `json:"selinuxContext,omitempty"
    // RunAsUser is the strategy that will dictate the allowable RunAsUser values that may be set.
    RunAsUser RunAsUserStrategyOptions `json:"runAsUser,omitempty"

    // The users who have permissions to use this policy
    Users []string `json:"users,omitempty"
    // The groups that have permission to use this policy
    Groups []string `json:"groups,omitempty"
}
```
WG New Area Exploration - policy formal verification

- Background Knowledge
  - SMT can be thought of as a form of the constraint satisfaction problem and thus a certain formalized approach to constraint programming
  - an SMT instance is a formula in first-order logic, where some function and predicate symbols have additional interpretations, and SMT is the problem of determining whether such a formula is satisfiable
  - A predicate is a binary-valued function of non-binary variables. Example predicates include linear inequalities (e.g. $3x + 2y - z \geq 4$) or equalities involving uninterpreted terms and function symbols (e.g. $f(f(u, v), v) = f(u, v)$)
$$(\sin(x))^3 = \cos(\log(y) \cdot x) \lor b \lor -x^2 \geq 2.3y) \land (-b \lor y < -34.4 \lor \exp(x) > \frac{y}{x})$$

where

$$b \in \mathbb{B}, x, y \in \mathbb{R}$$
WG New Area Exploration - policy formal verification

Construct a policy symbolic graph for each kubernetes domain

networking

Multi-tenancy

Security
WG New Area Exploration - policy formal verification

● Starting with use case for “Privilege Escalation”, requirements from operator LCM, multitenancy, Istio,…
● Collaboration involving AWS, Styra, and many others in the community
● Keep an eye on the slack channel or ping us via email (zhipengh512@gmail, evb@redhat.com) if you are also interested
Together, these concepts enable us to define a policy in a way that is consistent and automatable.
WG New Area Exploration - policy as type system

Proposed long term vision:

1- **Strong type system for Kubernetes resources**

- Better specifications and validation with a formal type system
- Algebraic types:
  - Allows you to define more complex resource types (e.g. “pod”+”configmap”, union types)
  - Compositional transformations and admission chains
WG New Area Exploration - policy as type system

Proposed long term vision:

2- Policy Hooks at key points

- Lifecycle: Admission, deletion
- Network traffic in and pod of pods
- Pod start up and down
- API calls - webhook not quite enough
Proposed long term vision:

3- Capabilities

- Pod “leases”
- Delegation, access control
Proposed long term vision:

4- Kubernetes as the “now”

- Flattened view with explicit consistency bands
- Pipeline of transformations to facilitate managing clusters
- Favor “compiled” over runtime interpretation
- **Everything in namespaces**
  - Doesn’t have to be the same “namespace”
    - e.g. “organization” concept over users
    - Needs to be - every resource is in one and only one namespace (or zone or class or whatever)
- **Immutable labels or annotations**
  - Keep context, allow chains of validations
- **Improved ownership**
  - Cross-namespace
  - “Object pairs” or other way to easily tie lifecycles together
CNCF Wide Collaboration
Semantic + Control = Architecture
Contact and Contribute
WG Facts

- Feel free to join the weekly meeting or leave a note on the meeting minute doc (https://goo.gl/auTfy2) if you have more interesting topics or projects could be used for case studies!
- Find us at #wg-policy on slack, propose any new interesting idea like we talked here for futuristic open source study!
- Add label wg-policy for your KEPS if it is policy related!
Thank you!

Q & A