Multi-tenancy in Kubernetes

KubeCon Europe 2018
May 4, 2018

David Oppenheimer <davidopp@google.com>
Software Engineer, Google

What is multi-tenancy?

Providing isolation between tenants within a cluster.

Why **within** a cluster?

Alternative: cluster per tenant

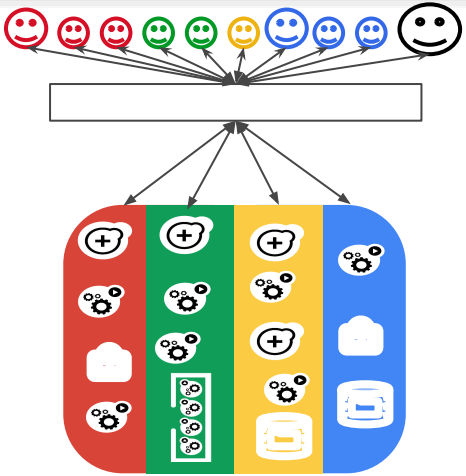
- get isolation for “free” !

But...

- need tools to manage 100s/1000s/... clusters
- pay control plane cost per tenant
- resource fragmentation
- need to create a cluster for each new tenant

=> intra-cluster multi-tenancy

Multi-tenancy use cases



Control plane (API server) isolation

Container isolation

Enterprise

Users all from same organization

Tenant == namespace == team/department

Cluster admin

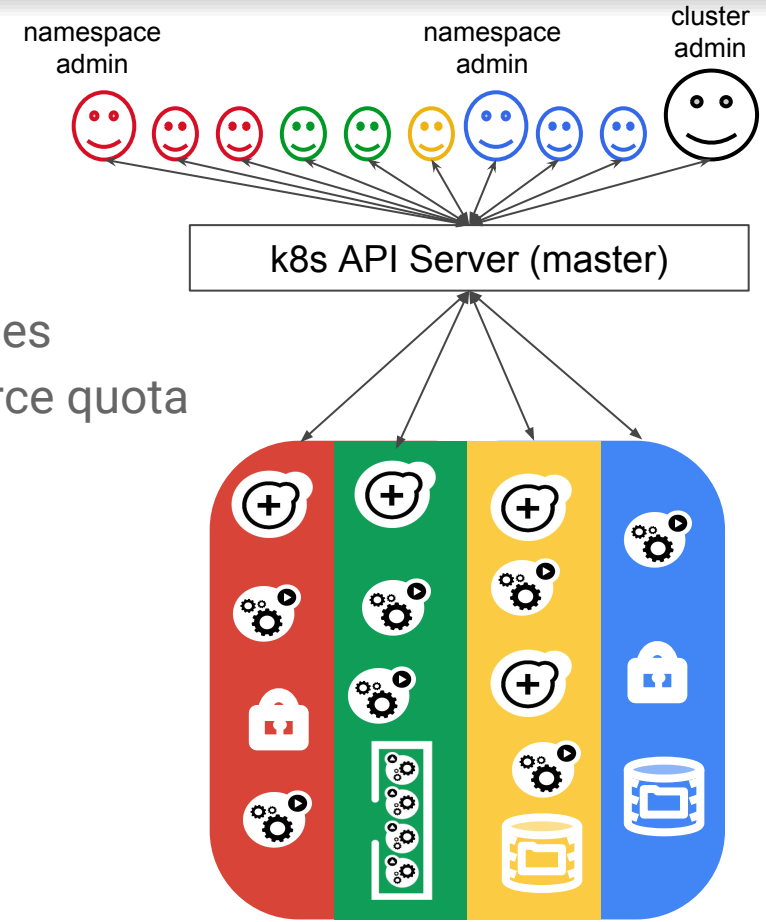
- CRUD any objects, including creating namespaces
- manage policies, e.g. set per-namespace resource quota based on internal budgets
- assign namespace admins

Namespace admin

- decide who has access to the namespace

Users

- CRUD non-policy objects in their namespace(s)



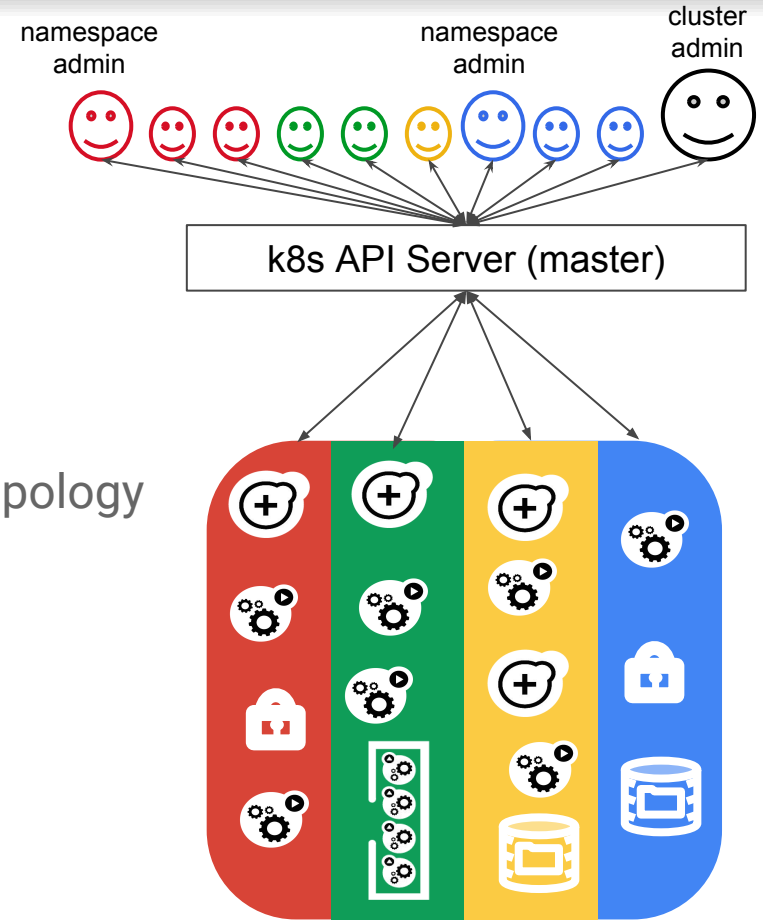
Enterprise

Many different apps, semi-trusted

May be OK with vanilla container isolation
May want to restrict container capabilities

Allow inter-pod communication depending on app topology

- self-contained within namespace
- split across namespaces
- communicate outside the organization



Kubernetes as a Service (KaaS) / Platform as a Service (PaaS)

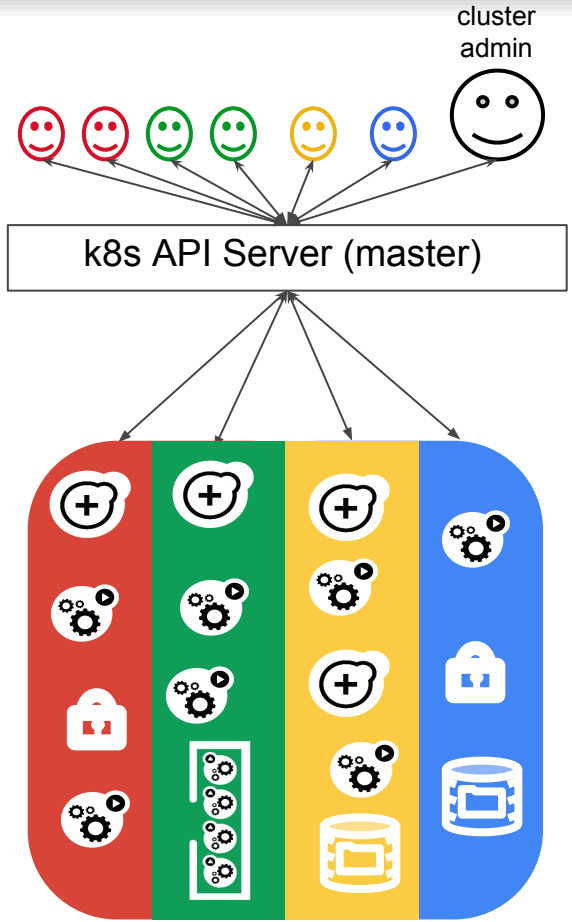
SaaS where the app is hosted Kubernetes
(+ extensions like producer-supplied CRDs/controllers)

Like enterprise, but **untrusted users running untrusted code**

Tenant == a consumer + their API objects

User can create namespaces and CRUD non-policy objects within their namespace(s)

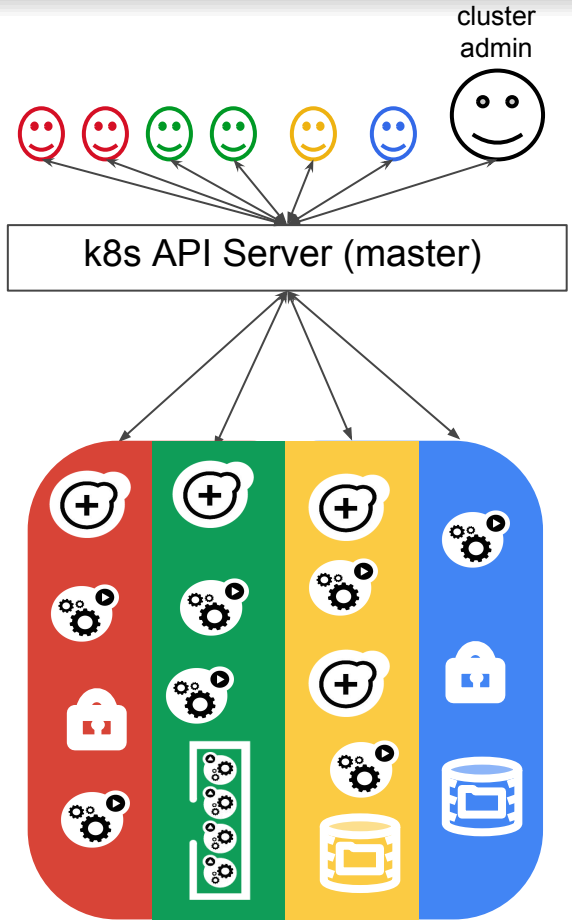
Resource quota based on how much the user paid



Kubernetes as a Service (KaaS) / Platform as a Service (PaaS)

Untrusted code => sandbox pods or sole-tenant nodes

Strong network isolation between namespaces belonging to different tenants



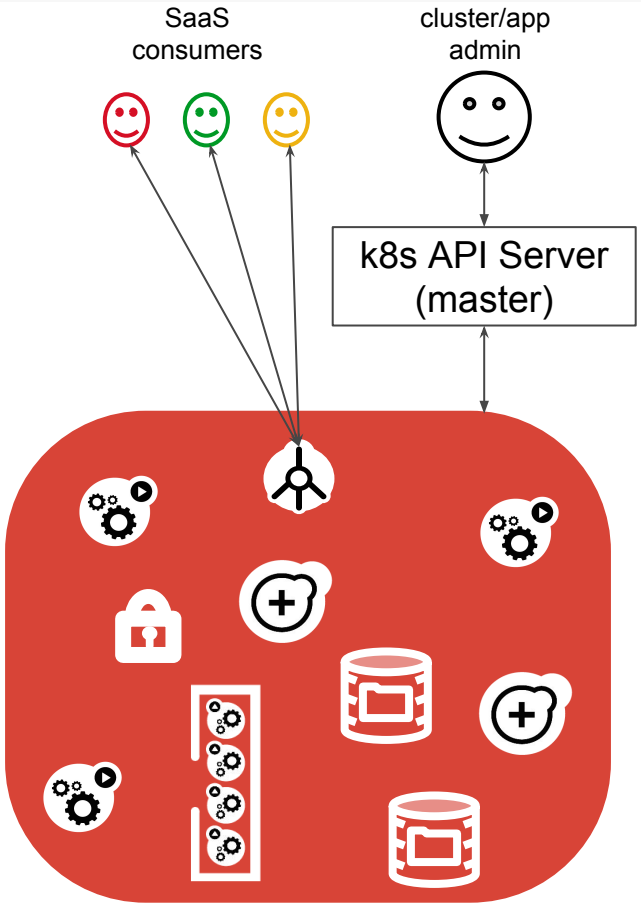
Software as a Service (SaaS): multi-tenant app

Single instance of application

Consumer interacts only with application

- cluster and API are hidden implementation details
- API server only accessible by cluster/app admin

Tenant is internal to app, opaque to Kubernetes



Software as a Service (SaaS): single-tenant app

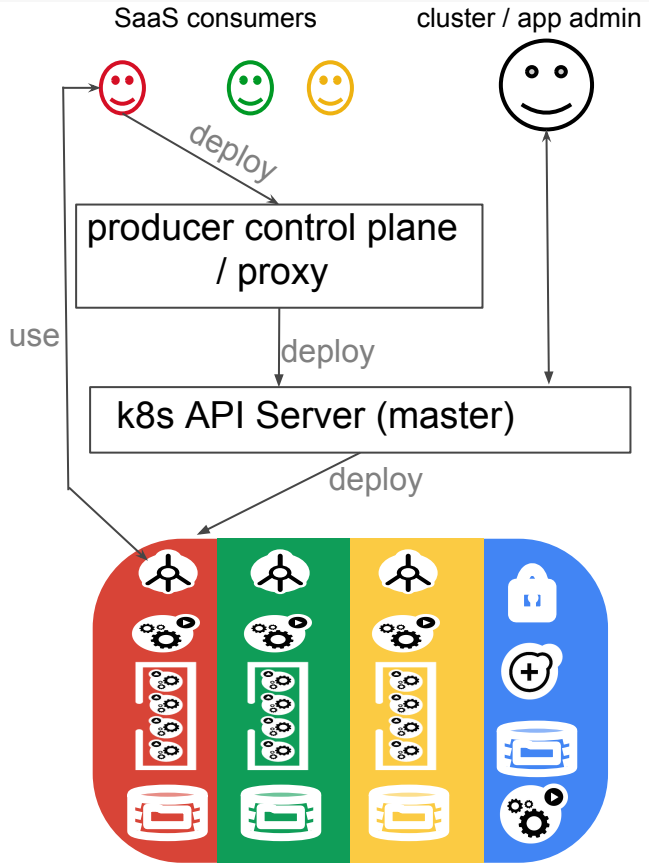
Consumer interacts only with application

Each consumer has their own app instance

Tenant == one application instance

API server accessible by

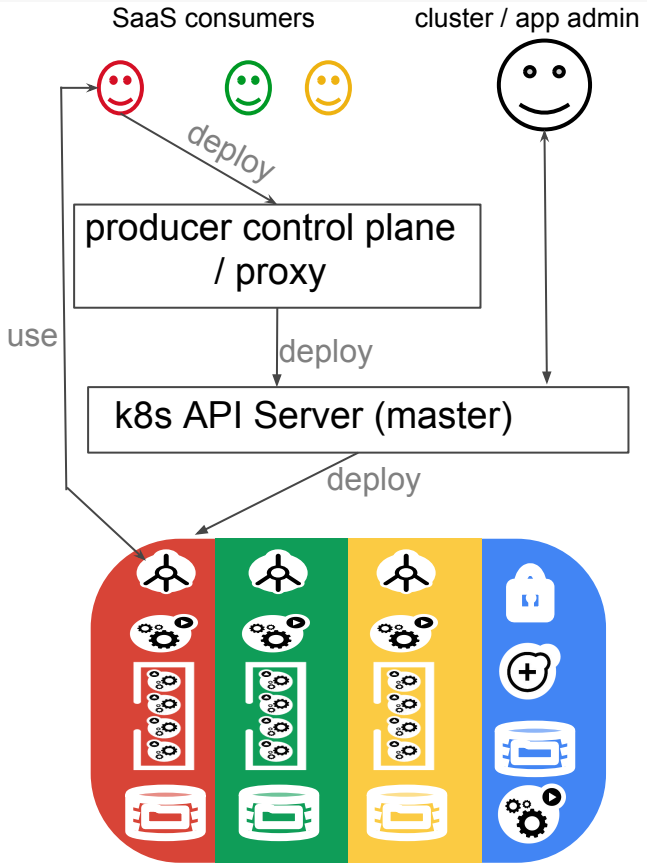
- cluster/app admin
- service control plane/proxy



Software as a Service (SaaS): single-tenant app

Code semi-trusted, may include untrusted (plugins)
=> sandbox pods or sole-tenant nodes

Pod communication: within namespace + to shared infra



Recap

Control plane

- who can access API server, what they can do
- resource consumption quotas

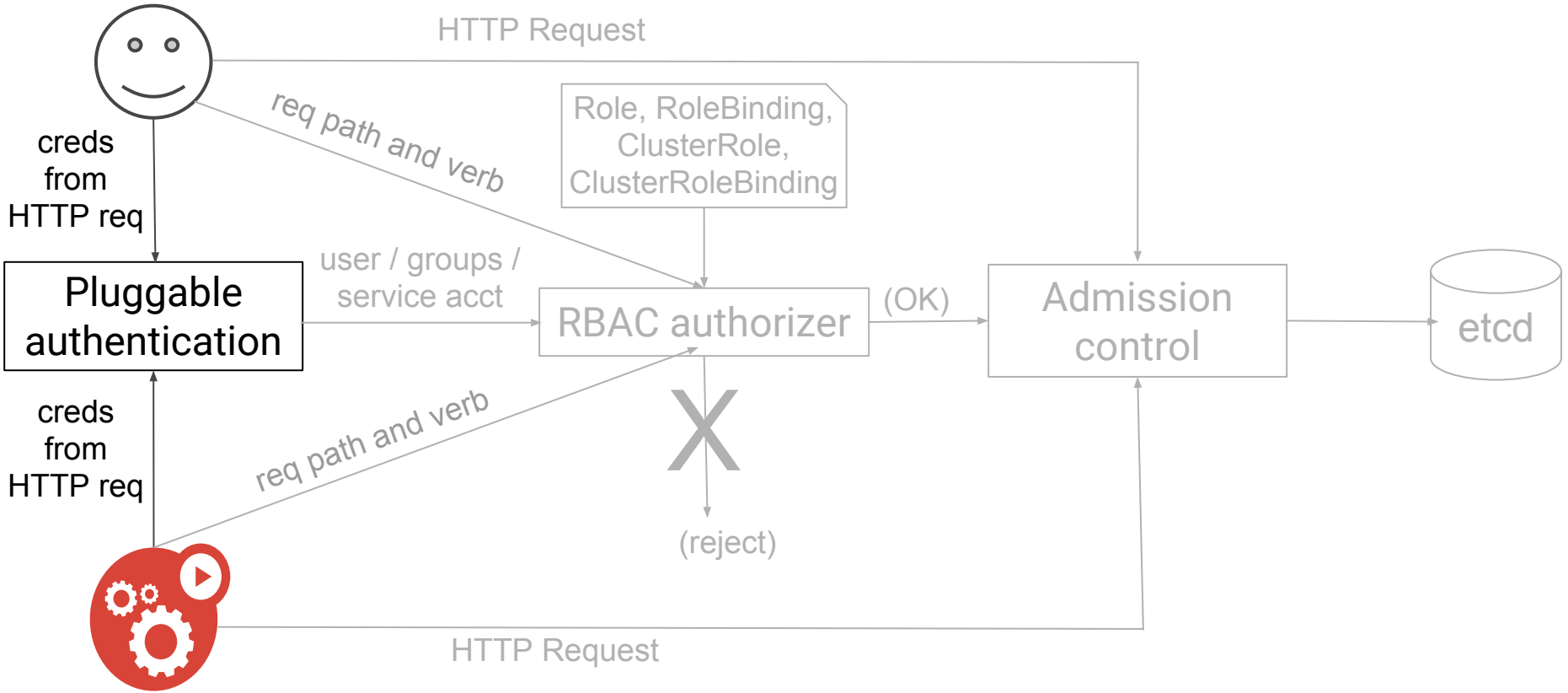
Container/network isolation

- what code executing in container can do
- which pods can share nodes with one another
- communication between containers and with outside world

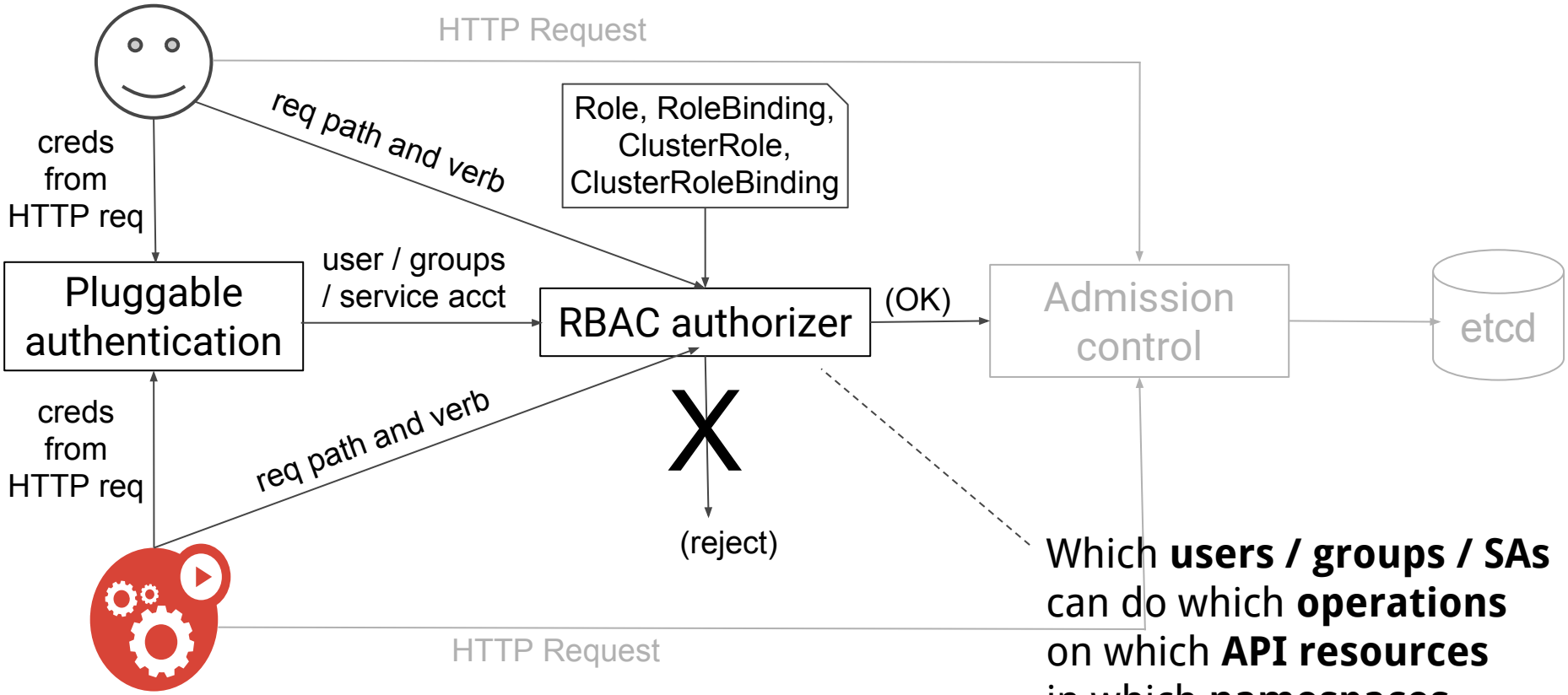
Multi-tenancy features

Multi-tenancy features: auth-related

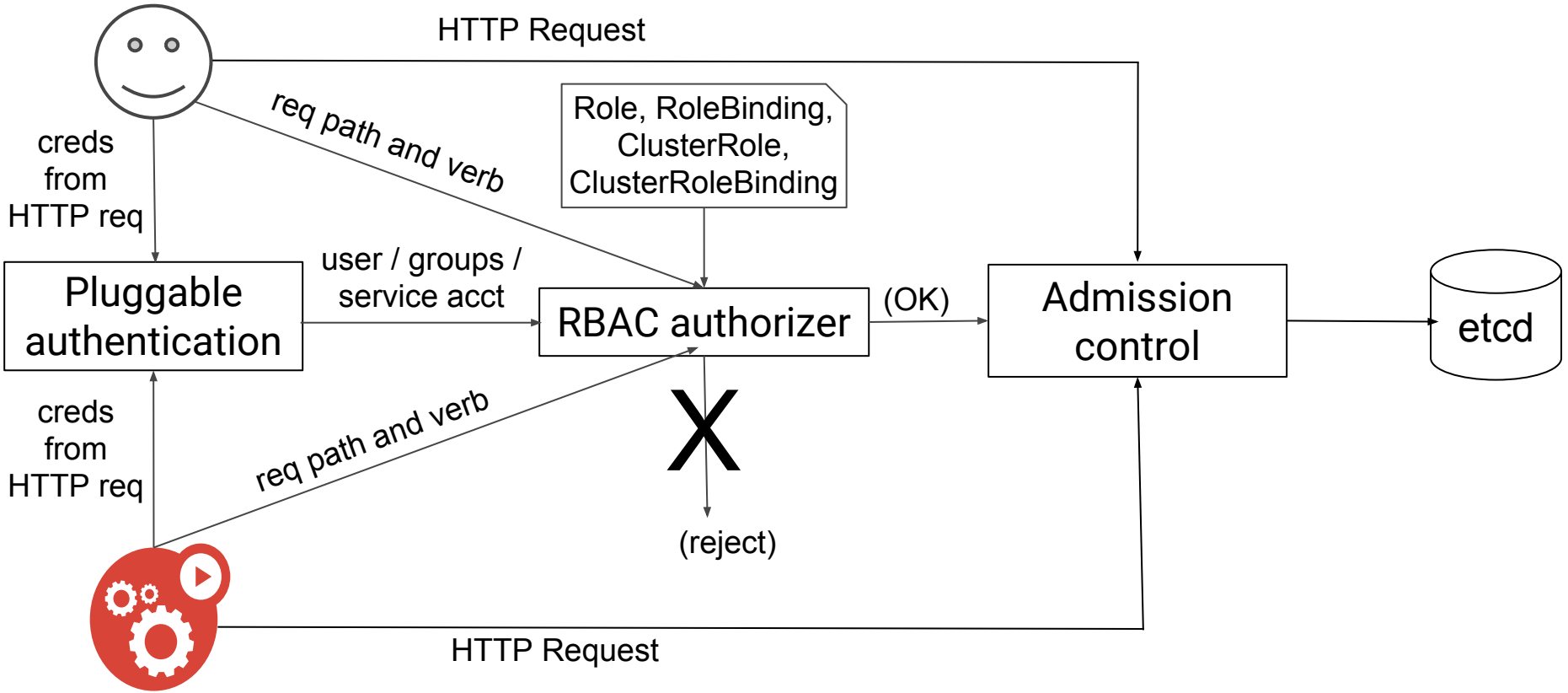
Multi-tenancy features: authentication and authorization



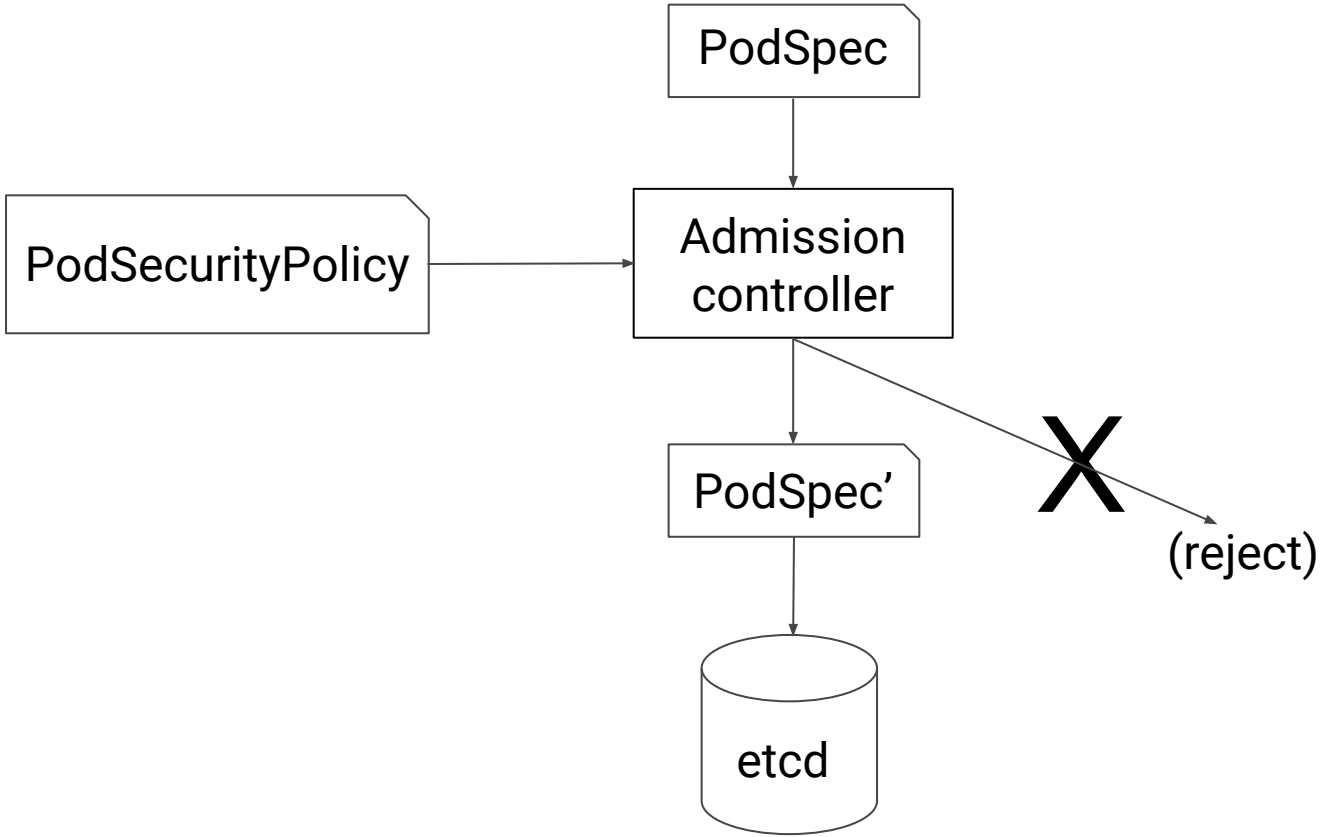
Multi-tenancy features: authentication and authorization



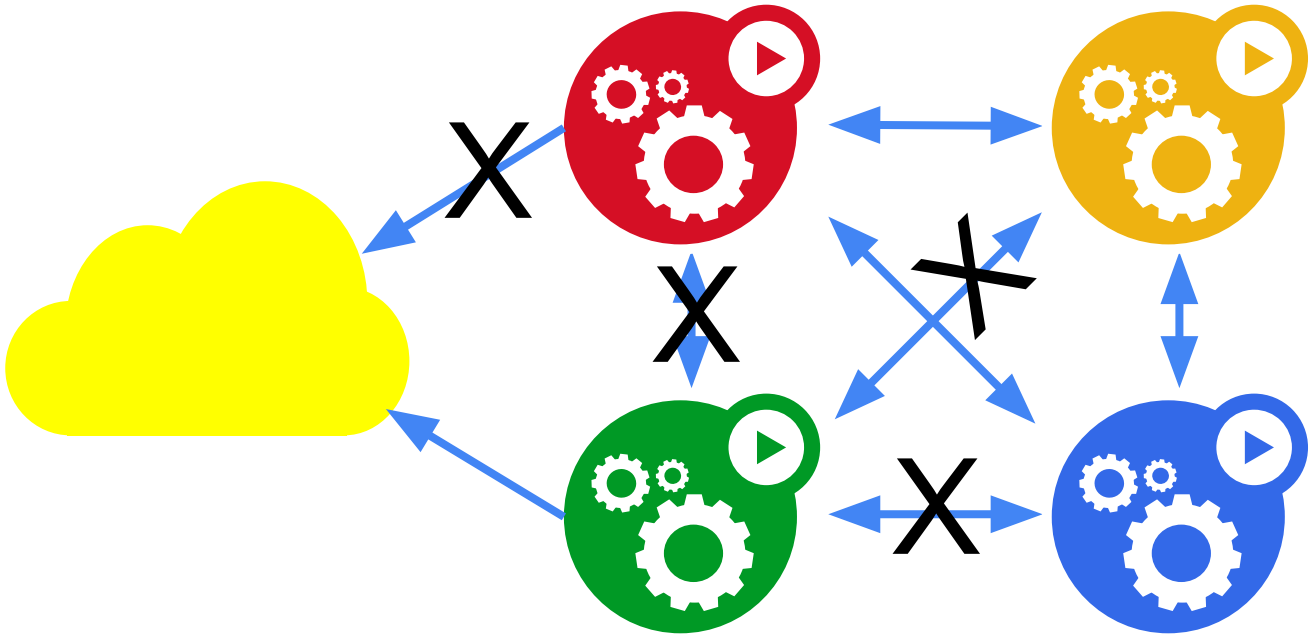
Multi-tenancy features: authentication and authorization



Multi-tenancy features: PodSecurityPolicy



Multi-tenancy features: NetworkPolicy



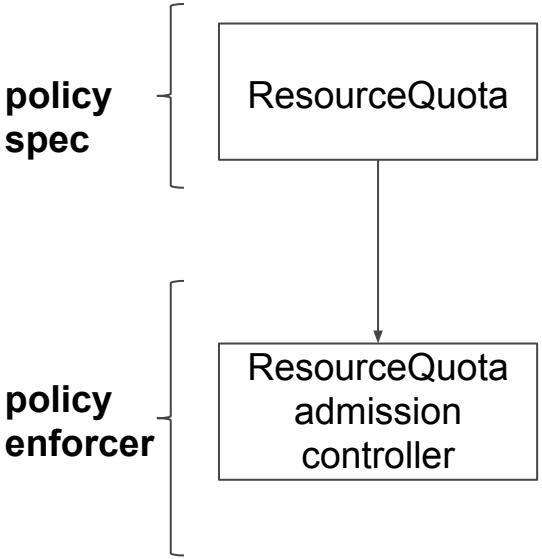
Multi-tenancy features: scheduling-related

Multi-tenancy features: scheduling-related

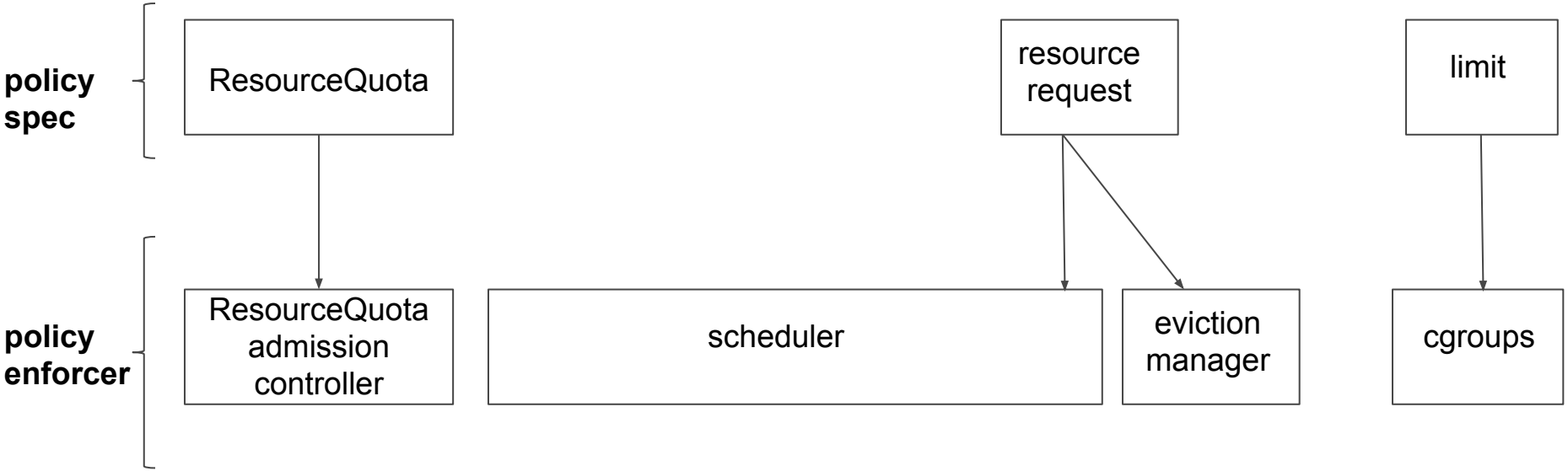
policy spec

policy enforcer

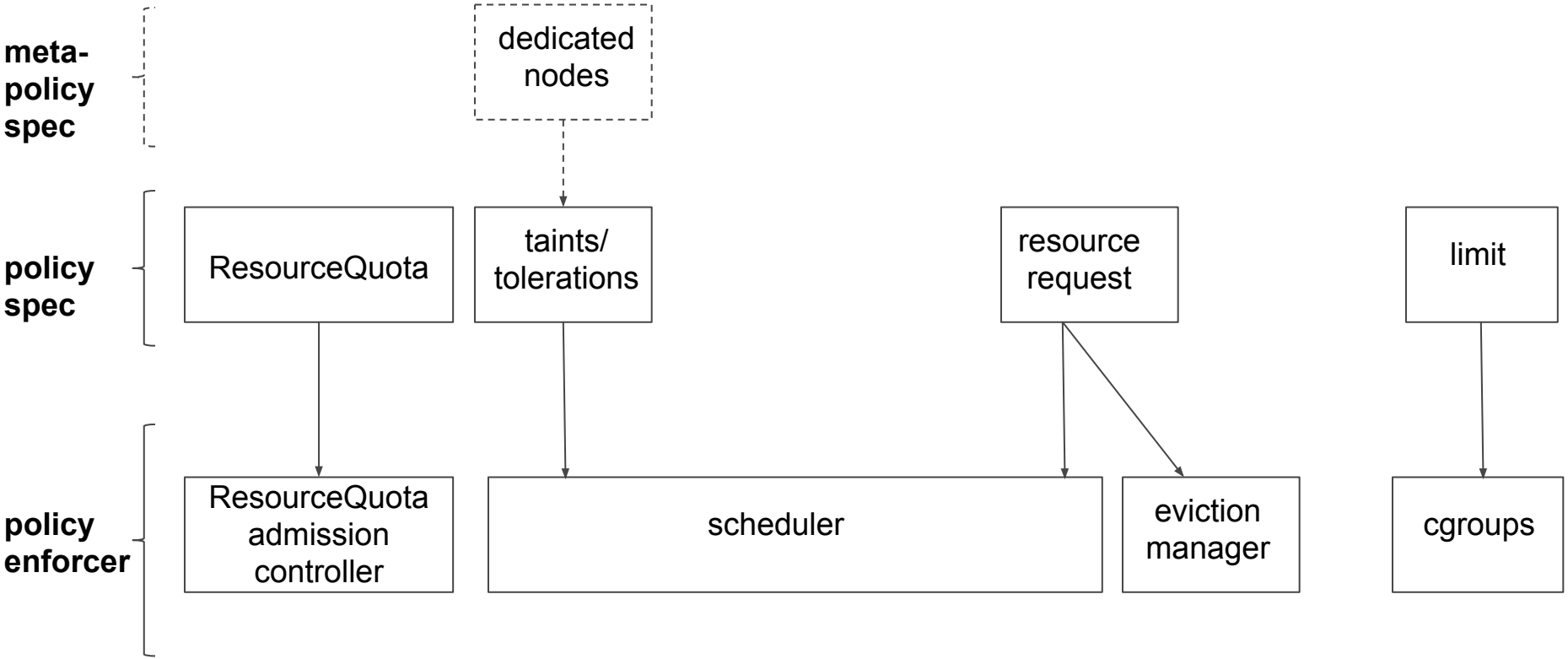
Multi-tenancy features: scheduling-related



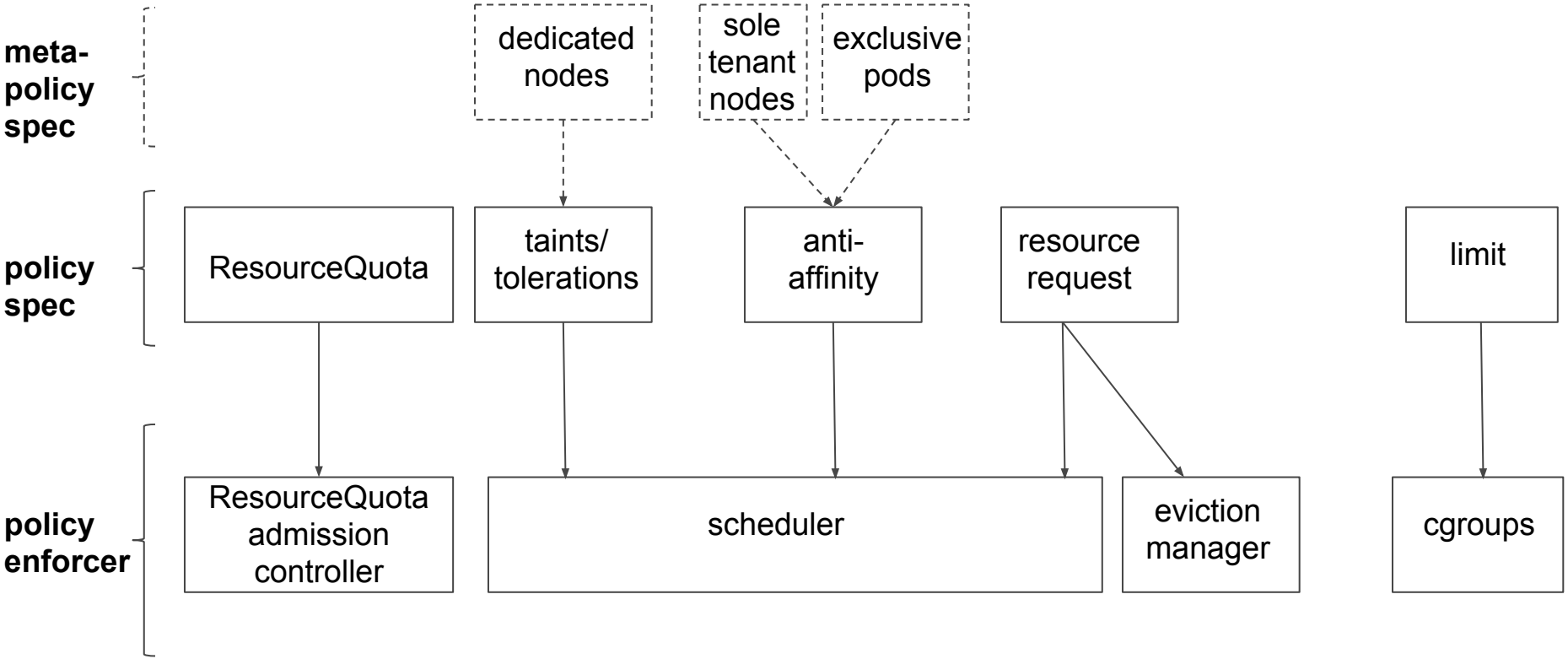
Multi-tenancy features: scheduling-related



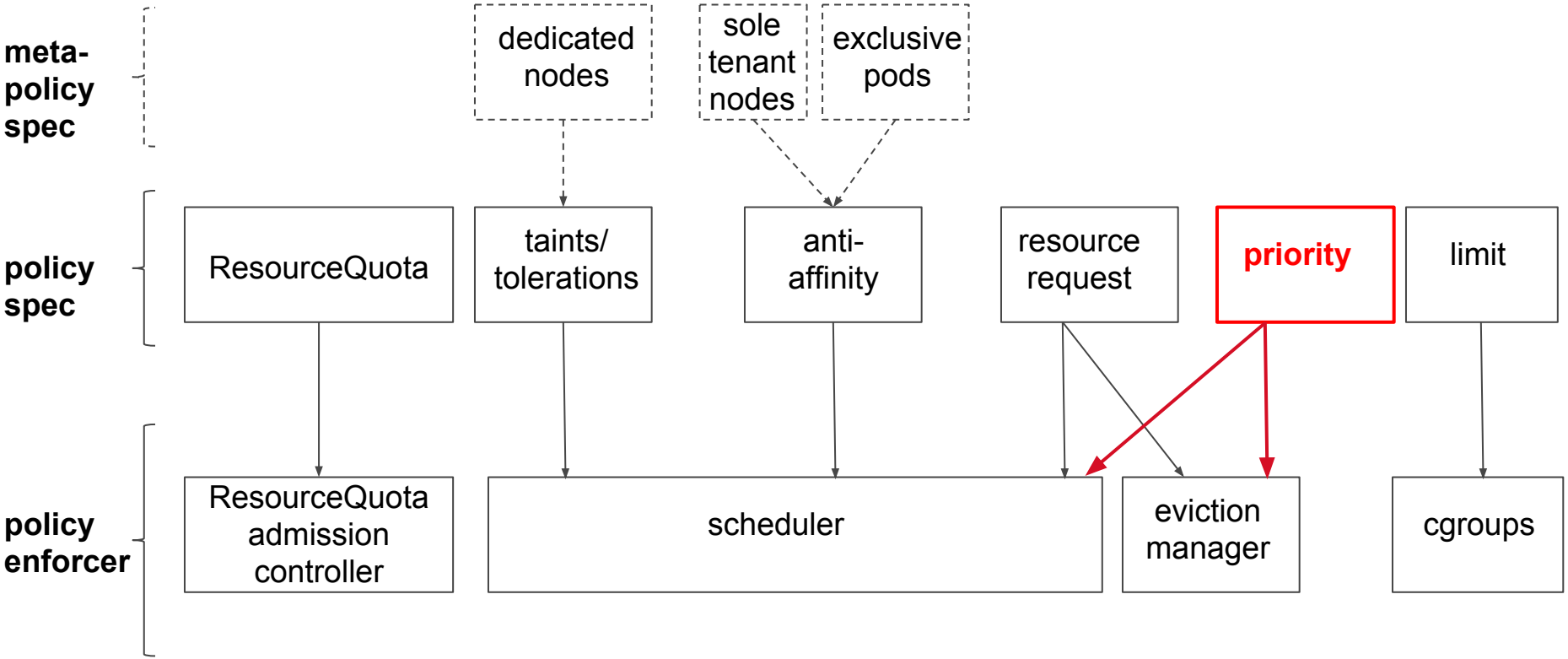
Multi-tenancy features: scheduling-related



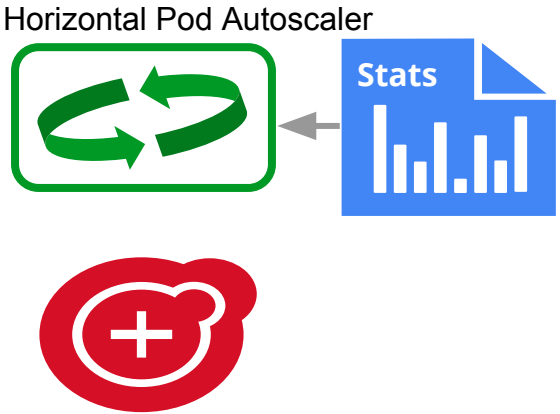
Multi-tenancy features: scheduling-related

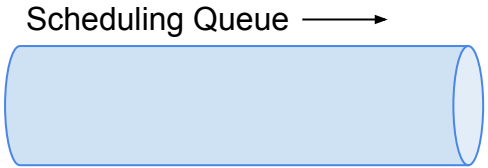


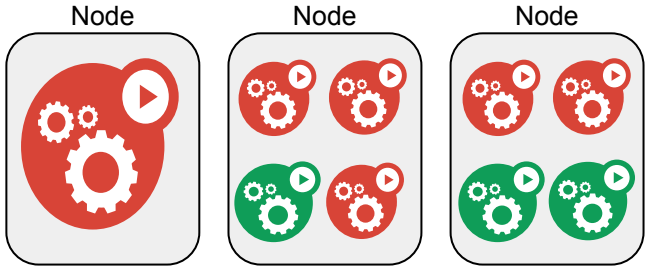
Multi-tenancy features: scheduling-related

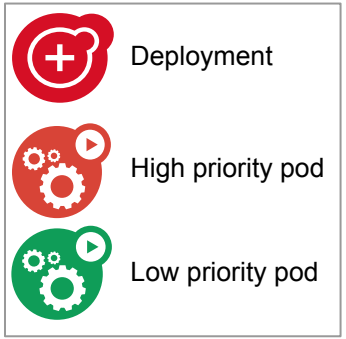


Priority and preemption (alpha)

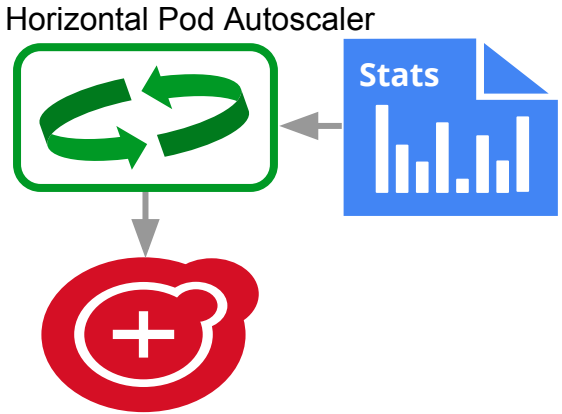


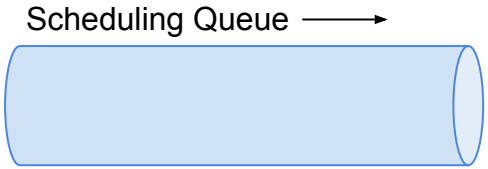


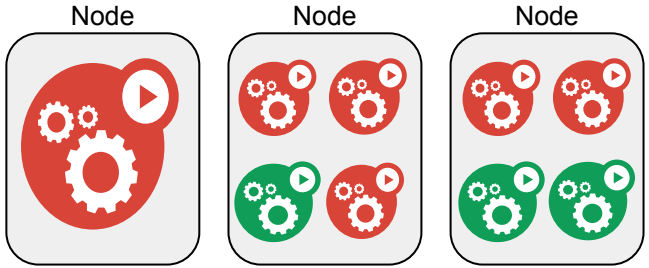


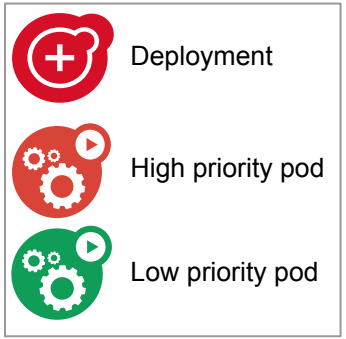


Priority and preemption (alpha)

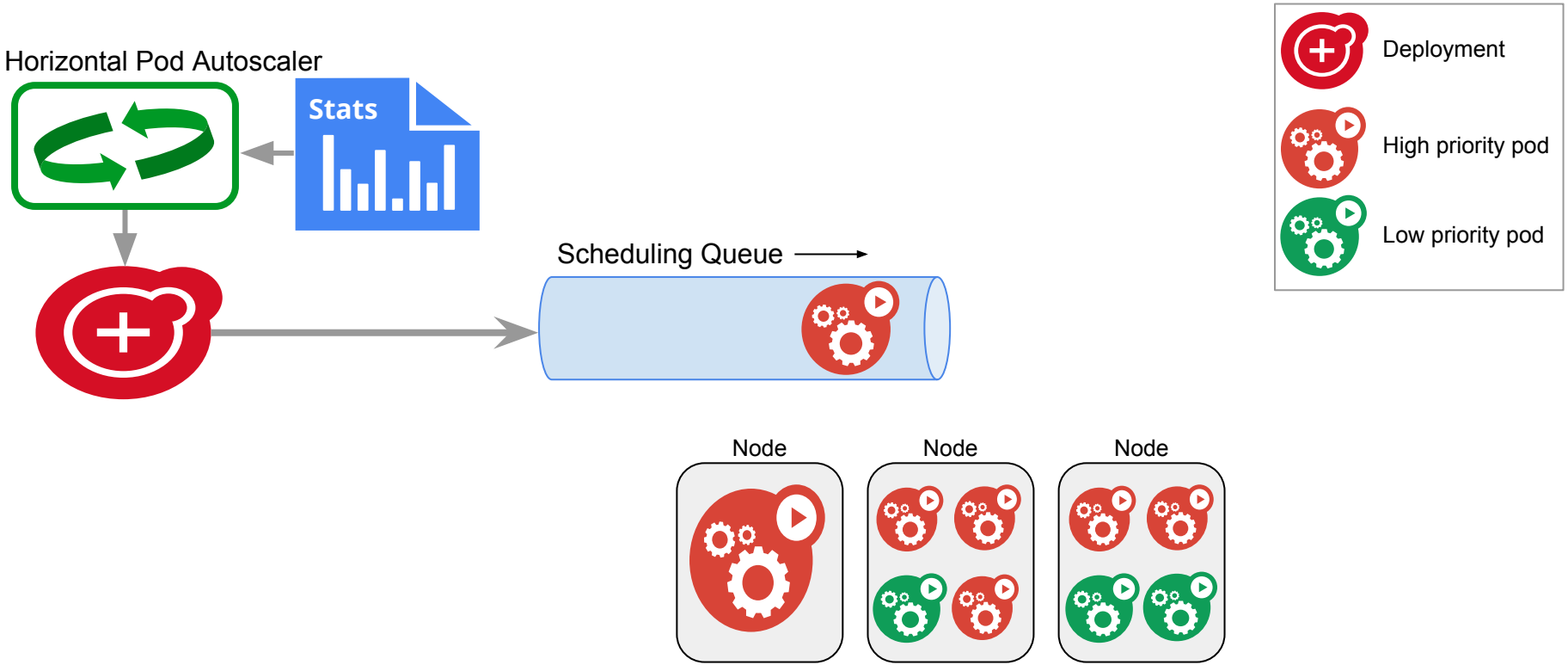




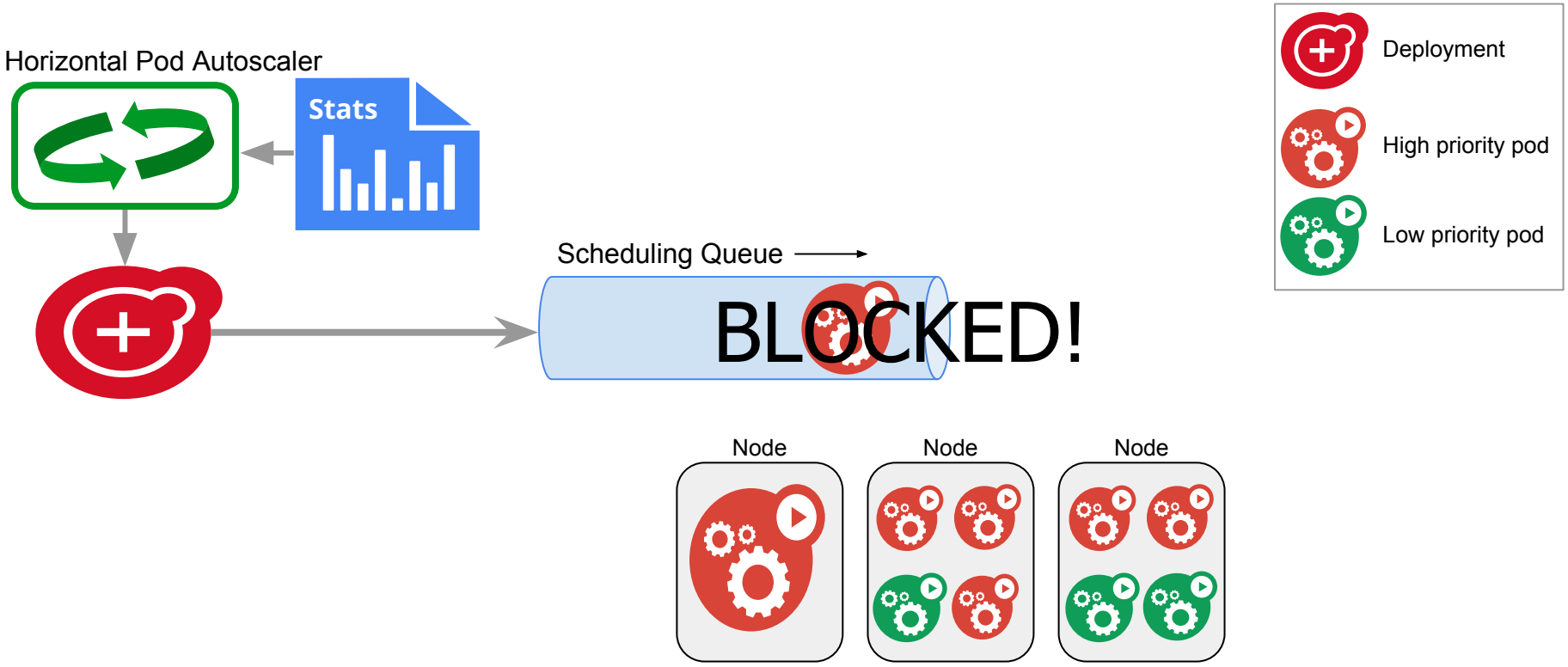




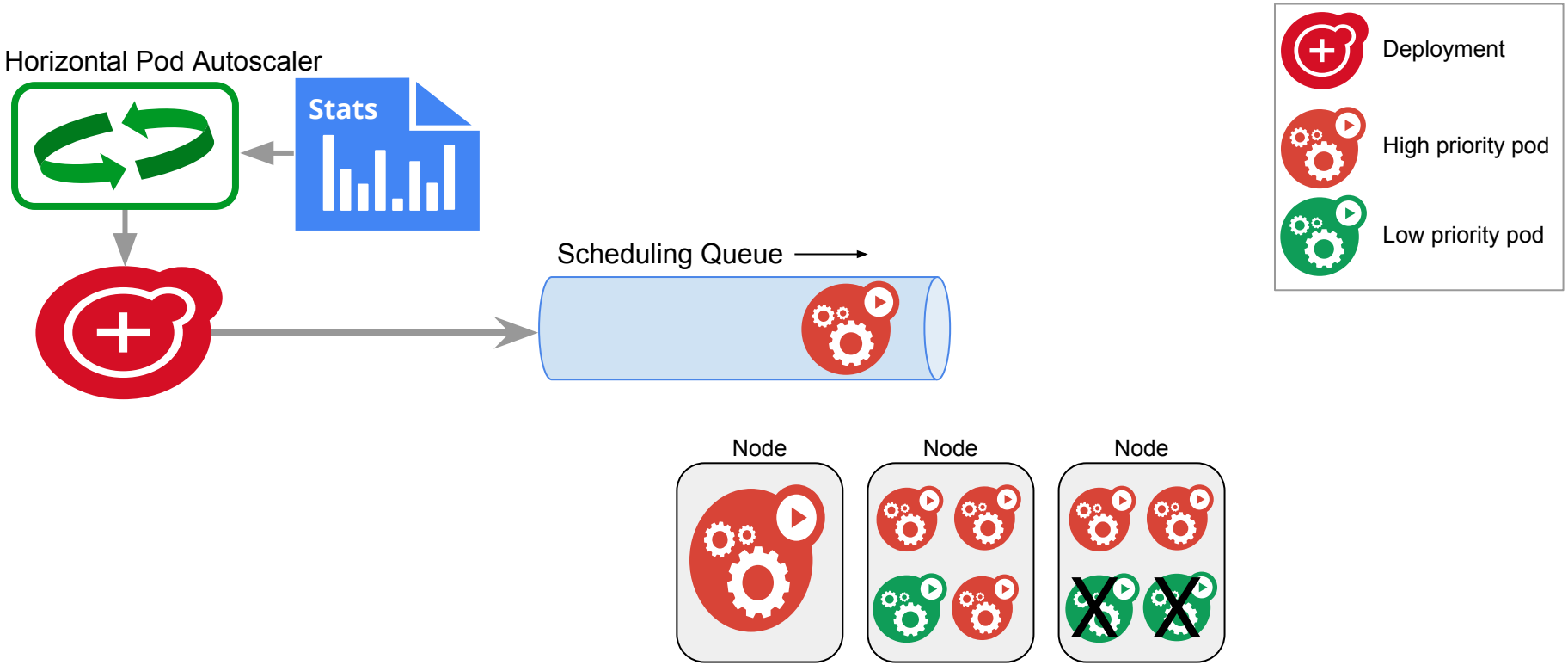
Priority and preemption (alpha)



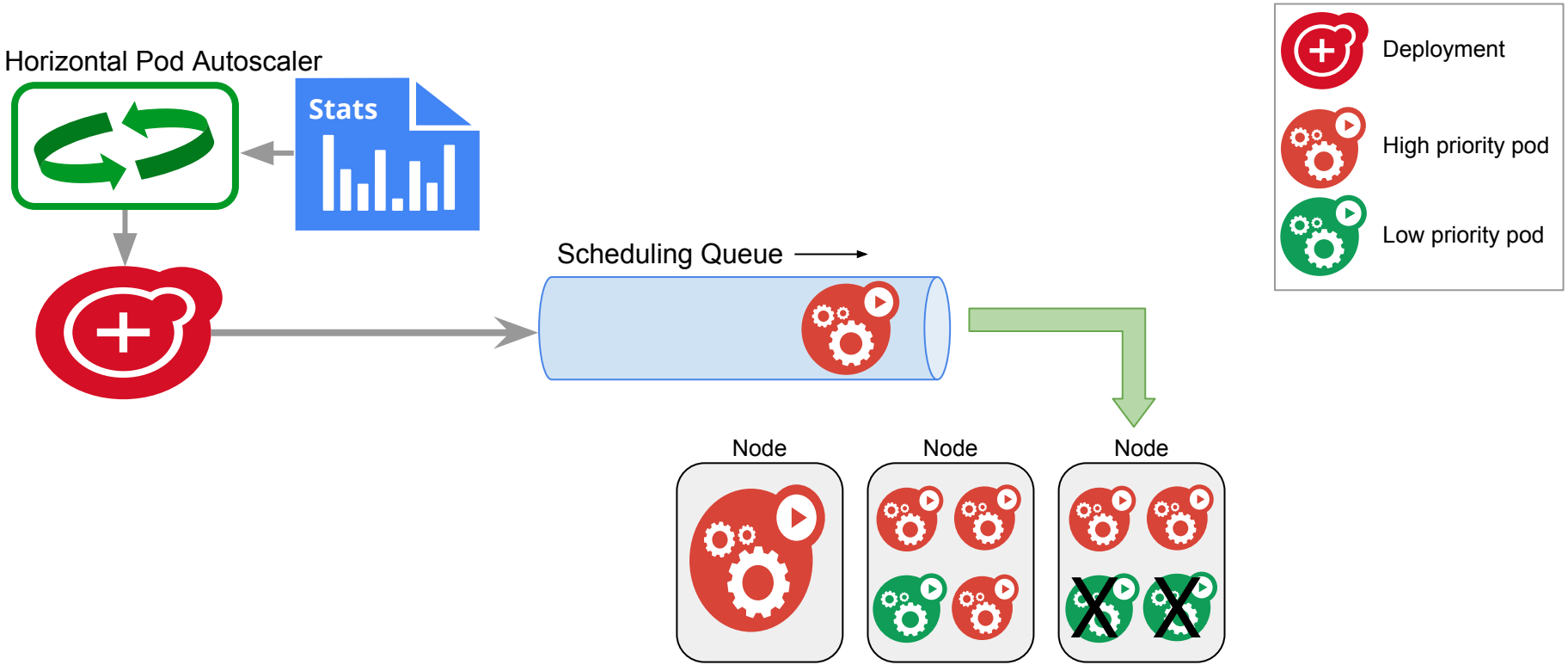
Priority and preemption (alpha)



Priority and preemption (alpha)



Priority and preemption (alpha)



Priority and preemption (future: when beta)

In a multi-tenant system, won't everyone just use the highest priority?

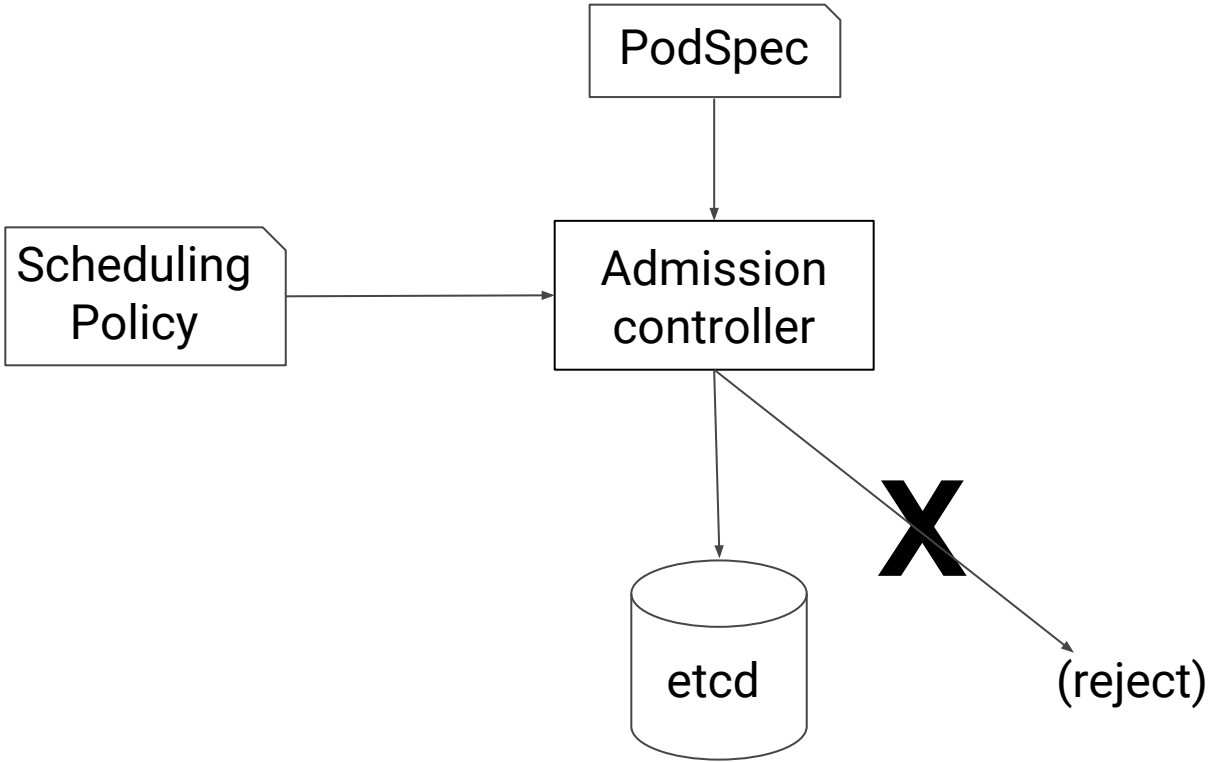
No!

ResourceQuota subdivided by priority

- "namespace X gets Y amount of RAM at priority Z"

Work in Progress: Policy

SchedulingPolicy



Security Profile

Improve usability of Kubernetes security/multi-tenancy features

Today, to operate a secure/multi-tenant cluster

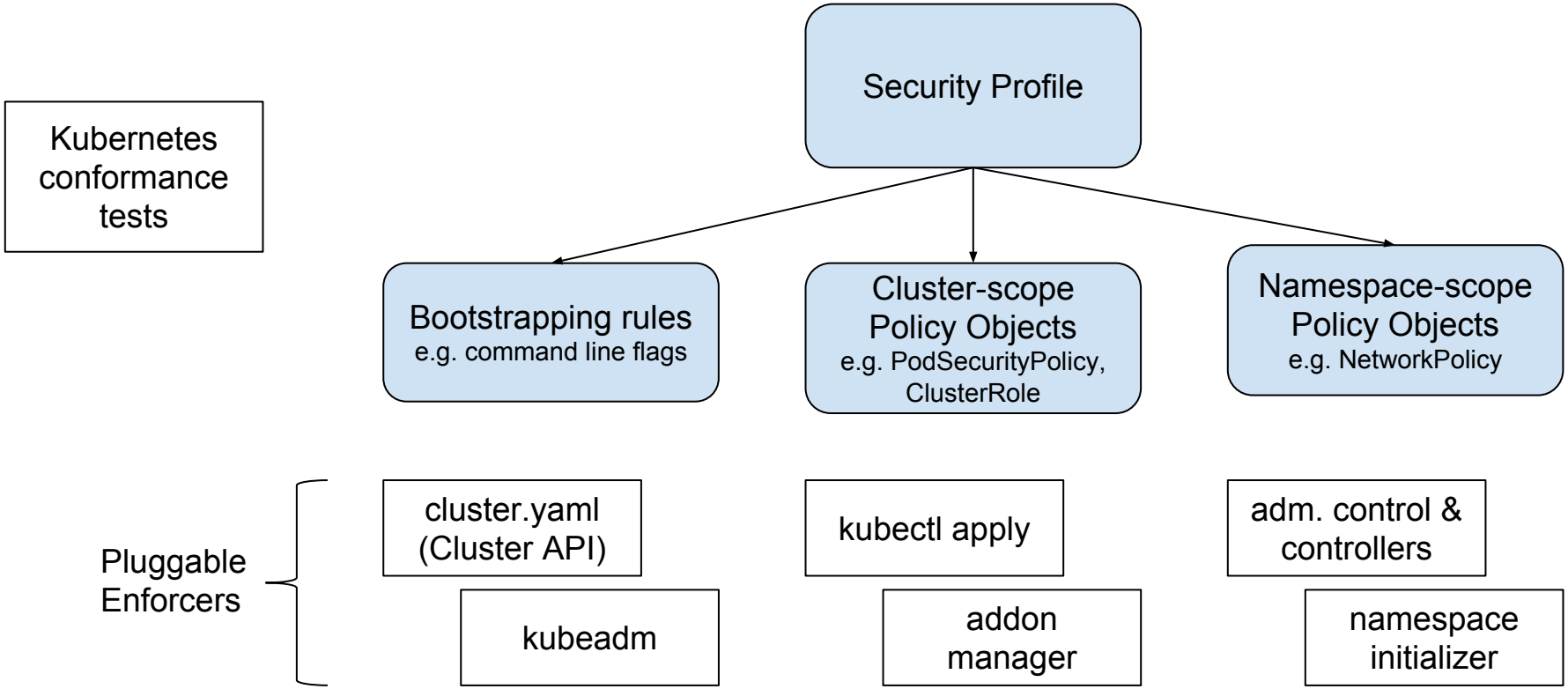
- cluster admins need to **understand security and Kubernetes** deeply
- policy **configuration is error-prone**, needs **tooling** to apply reproducibly
- policy configs must be **updated** as new Kubernetes features are added

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

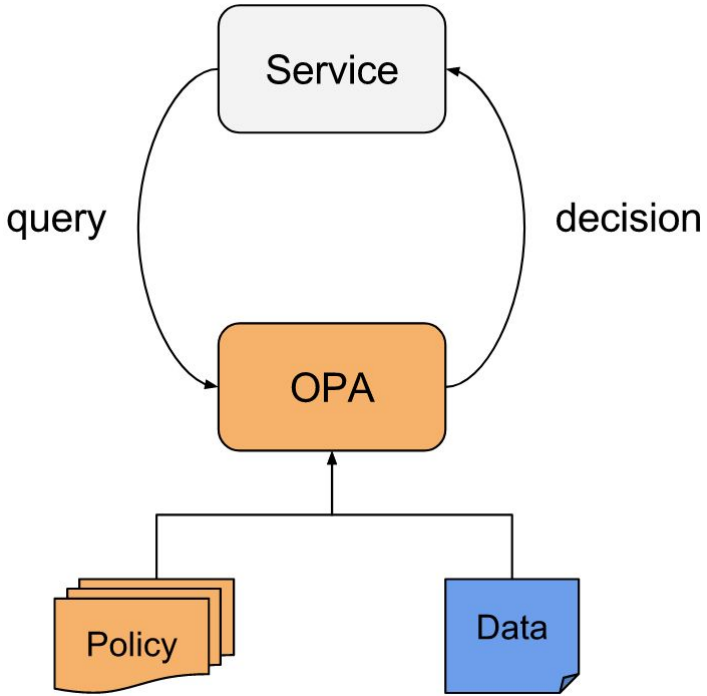


Open Policy Agent (OPA)

Policy Engine

- flexible, declarative language for expressing policies
- Kubernetes or application queries OPA to make policy decisions
  - e.g. authorization

Policy Working Group is exploring OPA integration with Kubenetes



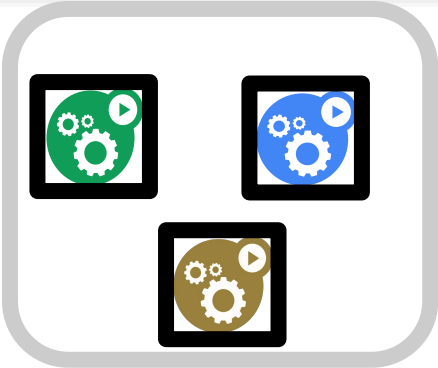
from http://www.openpolicyagent.org, (c) 2017 Open Policy Agent contributors

Work in Progress: Non-policy

Work-in-progress: non-policy

Sandbox pods

- VM-strength isolation between pods
- prevent code from escaping pod boundary
- examples: Kata Containers, gVisor



Control plane robustness

- Tenants can’t DoS each other by monopolizing shared control plane resources (API server, scheduler, controllers, etc.)
- Example: EventRateLimit admission controller

Container identity

Wrap up

Summary

Many multi-tenancy scenarios, e.g. enterprise, SaaS, KaaS/PaaS

Kubernetes multi-tenancy support is adequate today for many use cases

- **auth-related:** pluggable authn, RBAC, PodSecurityPolicy, NetworkPolicy
- **scheduling:** ResourceQuota, taints/tolerations, anti-affinity, request/limit, priority

Intra-cluster multi-tenancy is probably the only realistic choice with large # tenants

Ongoing work will improve isolation and make multi-tenancy features more usable

- **policy-related:** SchedulingPolicy, SecurityProfiles, OPA
- **non-policy-related:** sandbox pods, control plane robustness, container identity

How to get involved

SIG Auth

- https://groups.google.com/forum/#!forum/kubernetes-sig-auth
- meets alternating Wed @ 11:00 AM PT

Multitenancy Working Group

- http://groups.google.com/forum/#!forum/kubernetes-wg-multitenancy
- meets alternating Wed @ 11:00 AM PT

Policy Working Group

- https://groups.google.com/forum/#!forum/kubernetes-wg-policy
- meets every Thu @ 5:00 PM PT

Container Identity Working Group

- https://groups.google.com/forum/#!forum/kubernetes-wg-container-identity
- meets alternating Fri @ 8:00 AM PT