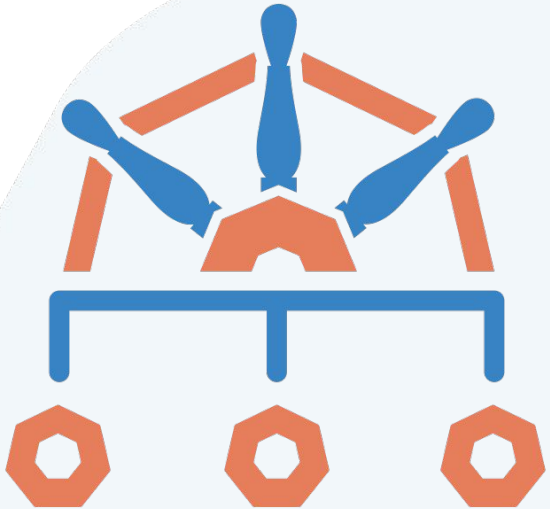
How to keep your clusters safe and healthy!

Kubernetes Native Policy Management

https://kyverno.io



nirmata

Real world problems

1. Validating Kubernetes configurations
2. Managing pod security
3. Generating default configurations
4. Applying fine grained access controls

Validating Configurations

The Problem

- Kubernetes configurations are complex
- A majority of security problems are caused by misconfigurations
- Some portions are environment specific
- Some portions are not developer concerns but critical for operations
- External configuration management tools (e.g. Helm, Kustomize) etc. cannot ensure best-practices and compliance for configurations

Sample Best Practice Configuration Checks

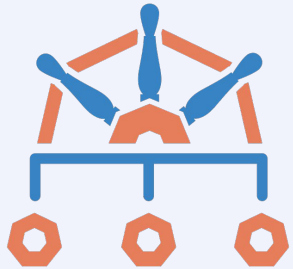
1. Configure readiness and liveness probes
2. Configure resource quotas
3. Do not use mutable (latest) image tags
4. Restrict image registries
5. Configure pod security

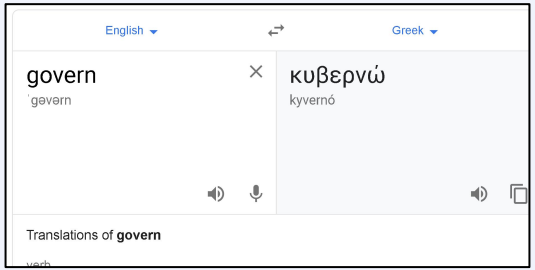
A Solution - Policy Management?

- Kubernetes admission controllers provide a way to inspect and validate, or mutate, API requests
- A policy manager can allow writing desired rules that are validated or enforced
- General purpose policy managers are not Kubernetes native and require learning a new language
- Writing custom admission controllers is non-trivial

Introducing Kyverno!

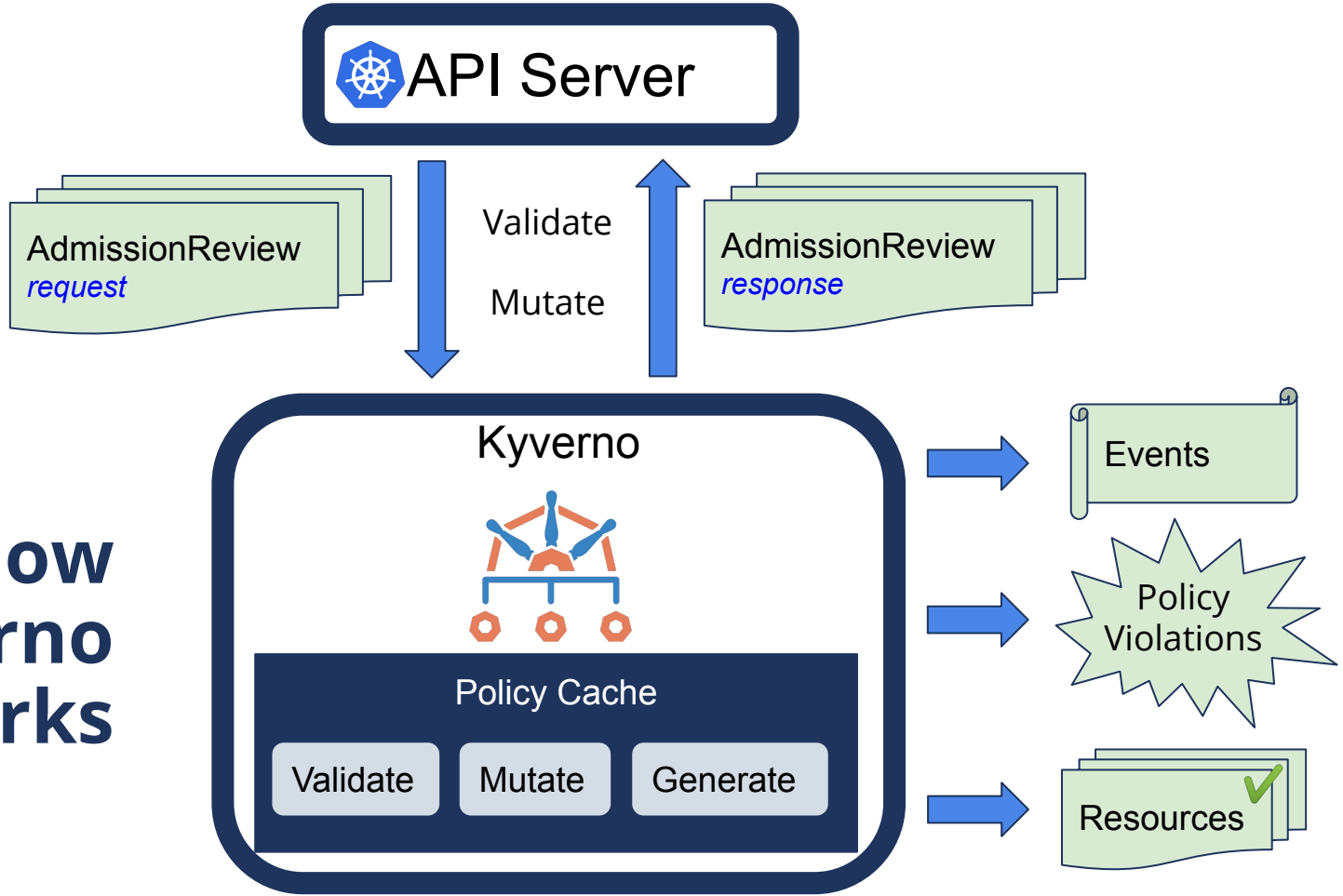
- An OSS policy engine for Kubernetes
- Uses Kubernetes resources, patterns, idioms. Writing policies and managing results is familiar to K8s users
- Kyverno blocks configurations that do not match a policy, or can generate policy violations (audit mode)
- Kyverno scans existing configurations and reports violations in your cluster.





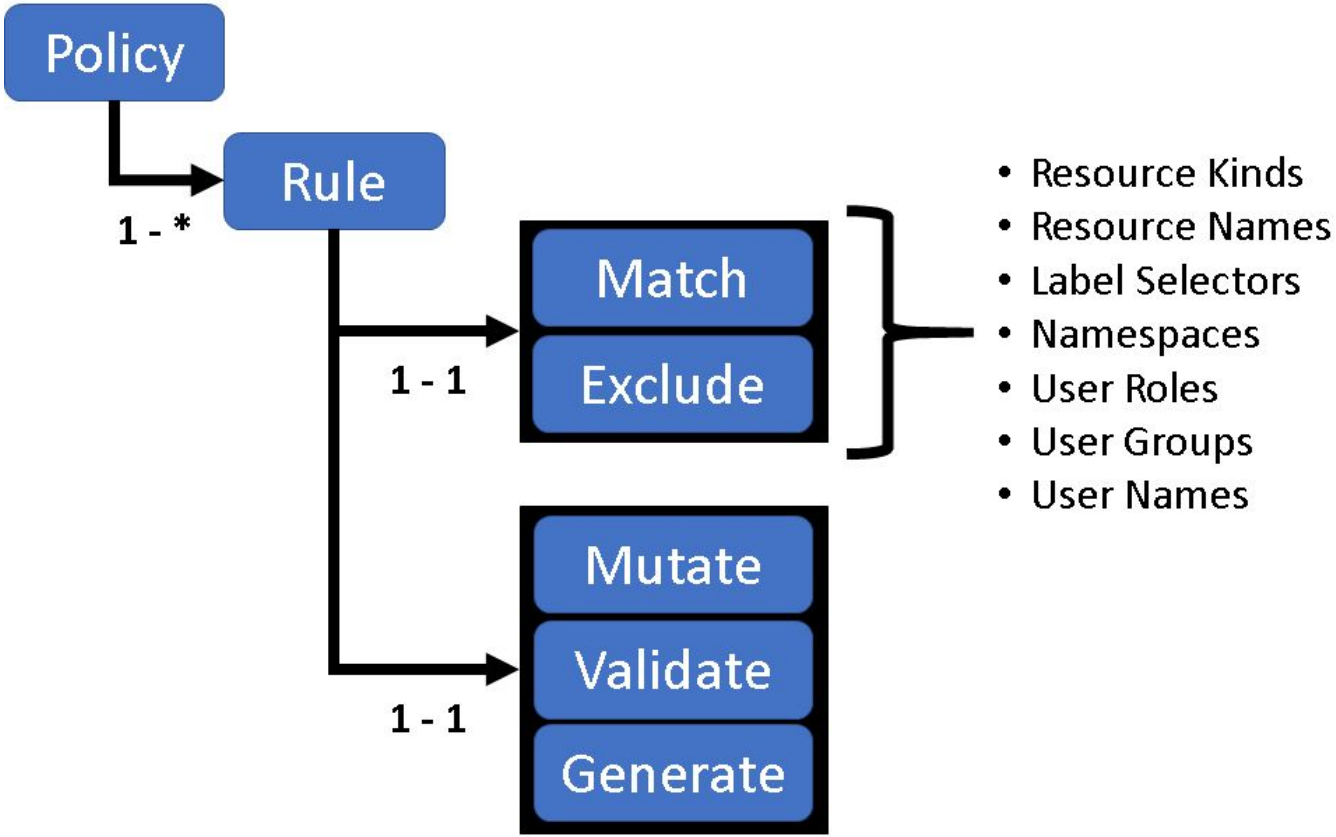
nirmata

How Kyverno works



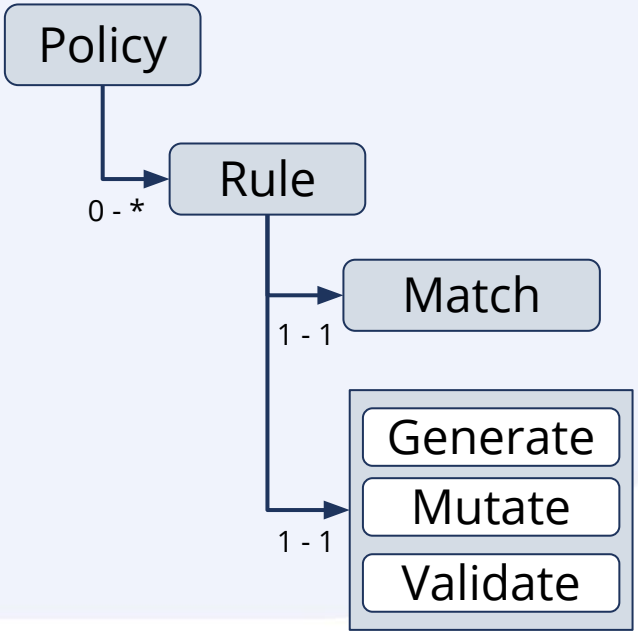
nirmata

Kyverno Policies



nirmata

A Kyverno Policy




```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: disallow-root-user
spec:
  rules:
  - name: validate-runAsNonRoot
    match:
      resources:
        kinds:
        - Pod
    validate:
      message: "Running as root user is not allowed. Set runAsNonRoot to true"
      anyPattern:
      - spec:
          securityContext:
            runAsNonRoot: true
      - spec:
          containers:
          - securityContext:
              runAsNonRoot: true
```


nirmata

Kyverno Validate Policy

- Overlays with patterns specify desired state
- Matches all defined fields
- Patterns
  - * : zero or more
  - ? : any one
- Operators
  - >, <, >=, <=, !, | (or)

```yaml
validate:
  message: "The label app is required"
  pattern:
    spec:
      template:
        metadata:
          labels:
            app: "?*"
```

OPA or Kyverno?

Policy to require read-only filesystem

```
package k8spspreadonlyrootfilesystem

violation[{"msg": msg, "details": {}}] {
    c := input_containers[_]
    input_read_only_root_fs(c)
    msg := sprintf("only read-only root filesystem container is allowed: %v", [c.name])
}

input_read_only_root_fs(c) {
    not has_field(c, "securityContext")
}
input_read_only_root_fs(c) {
    has_field(c, "securityContext")
    not has_field(c.securityContext, "readOnlyRootFilesystem")
}

input_containers[c] {
    c := input.review.object.spec.containers[_]
}
input_containers[c] {
    c := input.review.object.spec.initContainers[_]
}

# has_field returns whether an object has a field
has_field(object, field) = true {
    object[field]
}
```

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: require-ro-rootfs
spec:
  rules:
  - name: validate-readOnlyRootFilesystem
    match:
      resources:
        kinds:
        - Pod
    validate:
      message: "Root filesystem must be read-only"
      pattern:
        spec:
          containers:
          - securityContext:
              readOnlyRootFilesystem: true
```

Demo

Validating Configurations

Managing Pod Security

The Problem

- Pod Security is important
- Pod Security Policies (PSPs) are difficult:
  - Authorization model
  - Roll-out (all or nothing)
  - Beta. Scheduled to be deprecated in 1.22 in favor of Standardized Pod Security Profiles.

for a full discussion on PSPs see: SIG Auth Deep Dive KubeCon Video

Standardized Pod Security Profiles

- ***Privileged*** - Unrestricted policy, providing the widest possible level of permissions. This policy allows for known privilege escalations.
- ***Baseline/Default*** - Minimally restrictive policy while preventing known privilege escalations. Allows the default (minimally specified) pod configuration.
- ***Restricted*** - Heavily restricted policy, following current pod hardening best practices.

Kyverno Pod Security Policies

1. Run as non-root user
2. Disable privileged containers and disallow privilege escalation
3. Require Read-only root filesystem
4. Disallow use of host filesystem
5. Disallow hostNetwork and hostPort
6. Disallow hostPID and hostIPC
7. Configure Linux Capabilities
8. Limit Kernel parameter access

https://github.com/nirmata/kyverno/blob/master/samples/README.md

Demo

Pod Security

Auto-generating Pod Controller Policies

Problem:

- Operators want policies to apply to all pods independent of which controller creates them - hence a best practice is to write policies at a pod level
- ***But***, pods are typically created by pod controllers. Hence error reporting at the pod controller (e.g. Deployment) level provides a better user experience as pod-level errors are not visible to users

Auto-generating Pod Controller Policies

Solution:

- Kyverno automatically generates policy rules for pod controllers from rules written on pods
- Enabled by default
- Managed by an annotation `pod-policies.kyverno.io/autogen-controllers`:

none	Disable auto-generation
all	Enable auto-generation for Deployments, StatefulSets, Jobs, CronJobs
<name>,<name>	Enable auto-generation for each named controller

Generating Configurations

Problem

- When an object is created, we may need other default configuration to be created
- Example: When a namespace is created:
  - Create role and role bindings for namespace owner
  - Create default network policy
  - Create default quotas
  - …

Solution - Kyverno Generate Policies

- Triggers when a new resource is created
- Useful in creating defaults
- Can clones existing resources or copy in-line data

```
generate:
  kind: NetworkPolicy
  name: deny-all-traffic
  data:
    spec:
    podSelector:
      matchLabels: {}
      matchExpressions: []
    policyTypes: []
    metadata:
      labels:
        policyname: "default"
```

Demo

Generate Configurations

Fine Grained Access Controls

Problem

- Kubernetes RBAC is powerful but manages access at the API object and operation level only
- Sometimes we need more...e.g.
  - *Block deletes of objects tagged with a label for all users, except for cluster admins*

Solution - Kyverno deny rules

- Prevent deletes of resources generated by Kyverno...

  ...except by the cluster-admin and Kyverno

```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: multi-tenancy
spec:
  validationFailureAction: enforce
  background: false
  rules:
    - name: block-deletes-for-kyverno-resources
      match:
        resources:
          selector:
            matchLabels:
              app.kubernetes.io/managed-by: kyverno
      exclude:
        clusterRoles:
          - cluster-admin
      validate:
        message: >
          Deleting {{request.oldObject.kind}}/{{request.oldObject.metadata.name}}
          is not allowed
        deny:
          conditions:
            - key: "{{request.operation}}"
              operator: Equals
              value: "DELETE"
```

Demo

Fine Grained Access Controls

Summary

Takeaways

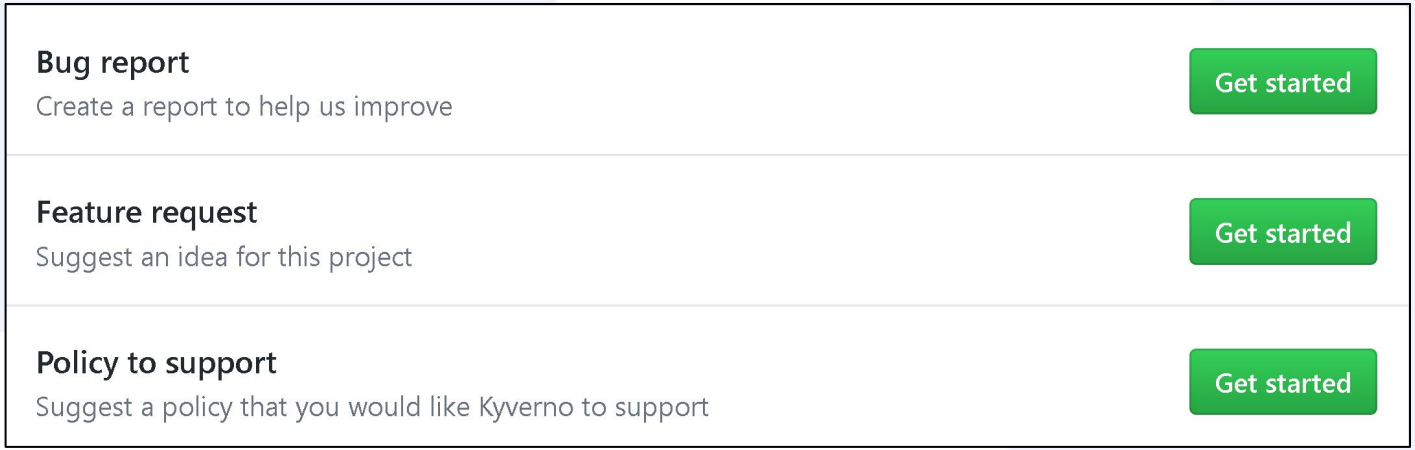
1. Policies are useful in managing Kubernetes configurations at scale
2. Kyverno is a policy engine built for Kubernetes
3. Kyverno can validate (audit or enforce), mutate, and generate configurations
4. Kyverno supports best practices for pod security and isolation
5. Kyverno is easy to use! Install Kyverno in your clusters, try the best practice policies, and give us feedback!

What's next? Major features on our roadmap

1. "Generate-and-Synchronize"
2. Query any existing resources e.g. ConfigMaps
3. Additional operators e.g. regex

How you can contribute?

- Install and use Kyverno - https://github.com/nirmata/kyverno
- Get involved - Kubernetes Slack channel #kyverno



nirmata

Thank-You!

https://kyverno.io