
Kyverno

Policies are a contract

Engin Diri
@_ediri

PM: We don’t need policies on our kubernetes cluster

The prod cluster 5min later:

12:06 PM · Jun 4, 2021 · Twitter for iPhone

10 Retweets **2** Quote Tweets **22** Likes

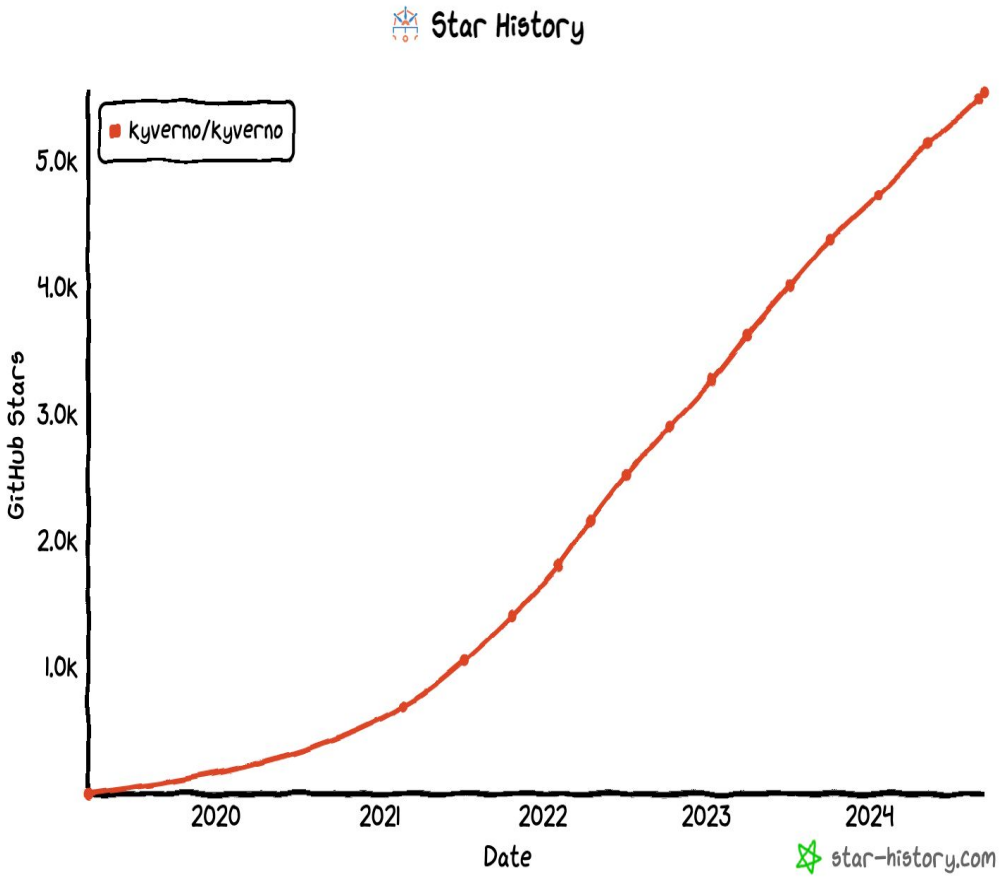
https://twitter.com/_ediri/status/1400891699581308930

Kyverno is a cloud native policy engine

A CNCF project created and maintained by Nirmata

- **Eliminate misconfigurations**
- **Prevent vs. detect**
- **Automate security**

3.2B image pulls
5.5K GitHub Stars
300+ contributors
3000+ Slack members



Why Kyverno?

Kyverno simplifies K8s policy management!

1. Make K8s policies easy to write and manage
2. Make policy results easy to process
3. Validate (audit or enforce), Mutate, and Generate
4. Support all Kubernetes types including Custom Resources
5. Use Kubernetes patterns and practices
   e.g. labels and selectors, annotations, events, ownerReferences, pod controllers, etc.

Kyverno History

2019 Created by Nirmata

2020 Donated to the CNCF

2022 Moved to Incubation

2023 Applied for Graduation

Security is a piece of cake with Kyverno. Kyverno helped us to implement proper security for different kind of clients (medical/telecommunication/trading...).

It solves problems like security enforcement, container image verification, distribution of imagePullSecrets and many more.

CNCF Policy Engines: Kyverno or OPA?

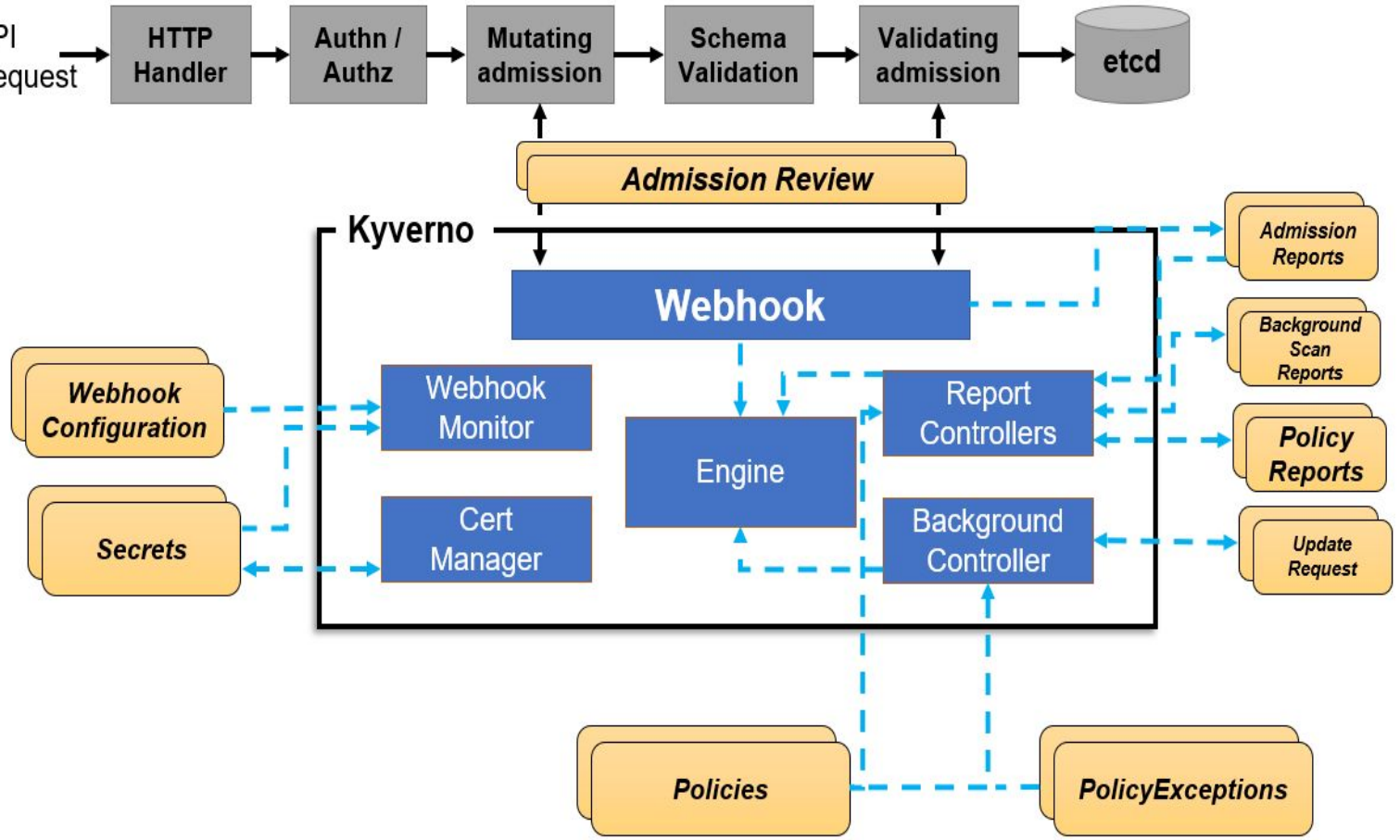
Features	Kyverno	OPA/Gatekeeper
Low-code Policies	Y	N (Rego)
Resource Validation	Y	Y
Resource Mutation	Y	Limited
Resource Generation	Y	N
Policy Exceptions	Y	N
Policy Reports	Y	N
Validation of non-Kubernetes resources	Y	Y
Integrated Supply Chain Security	Y	N

Key Use Cases

- Pod Security
- Workload Security
- Best practices
- Multi-tenancy and Isolation
- Resource Management
- Cost governance
- Software supply chain security

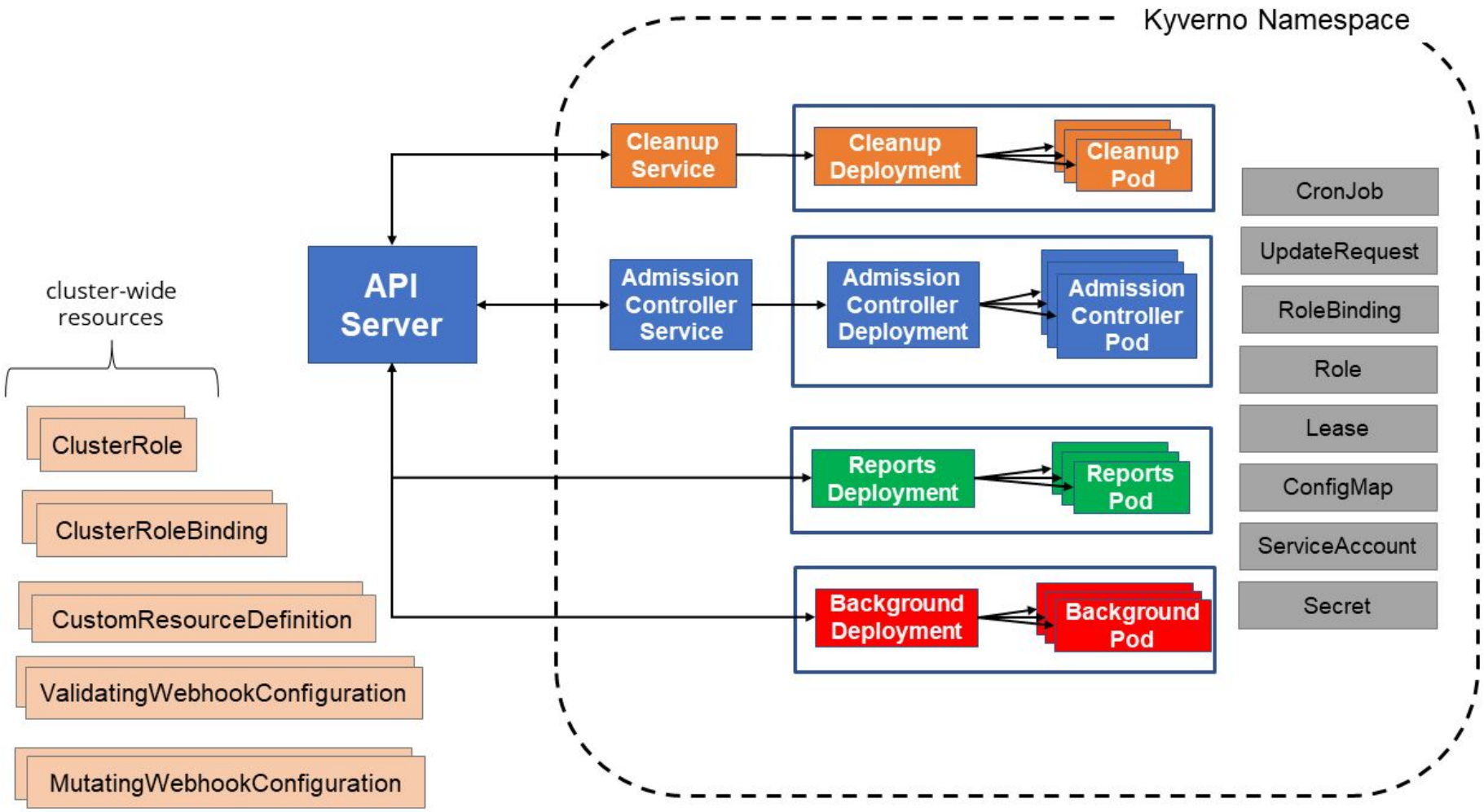
Kyverno Architecture

- Admission Controller
- Command Line Interface
- Background Scanner

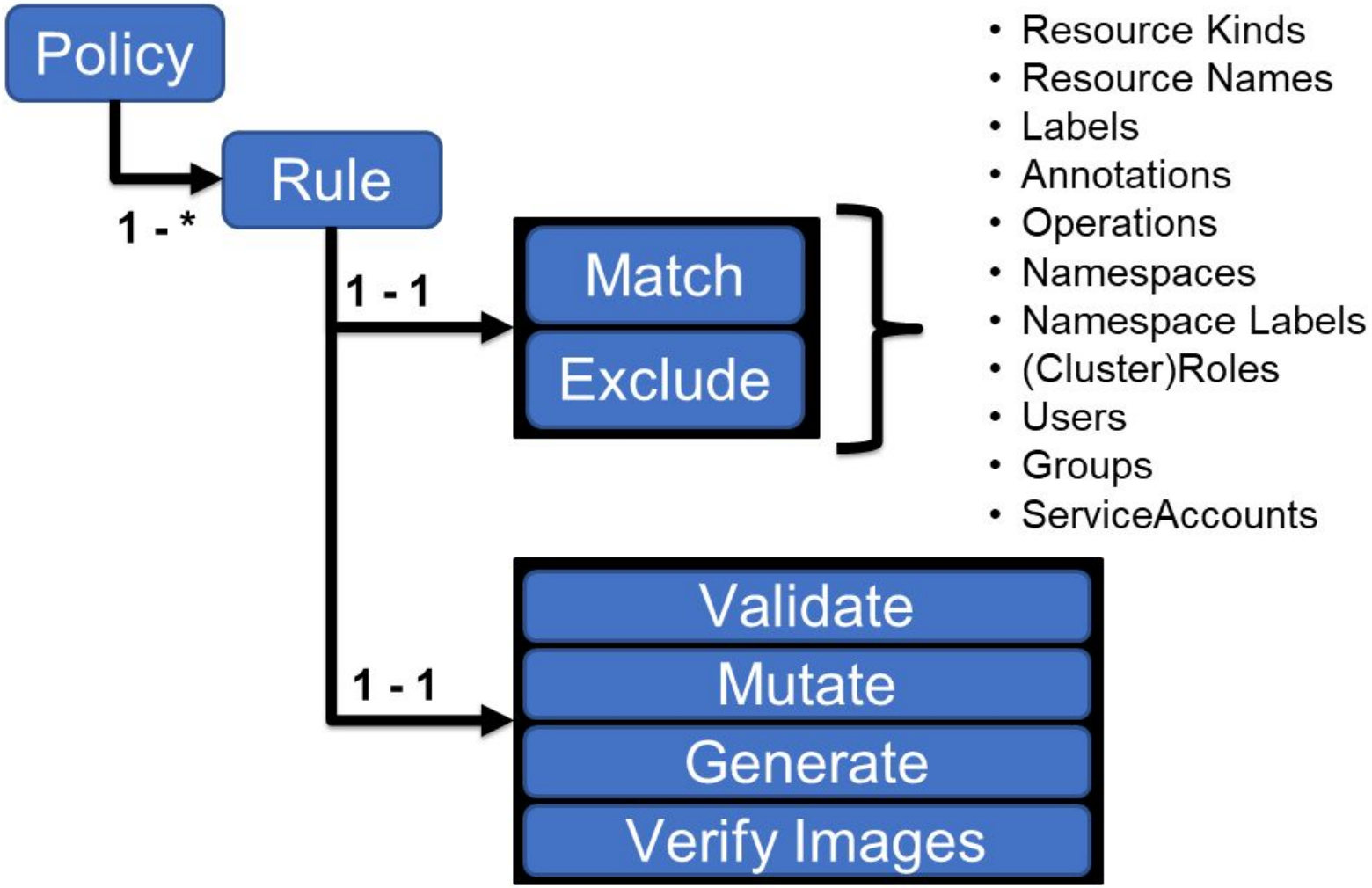


Deployment architecture (Kyverno 1.10+)

Increased scalability with service decomposition



Policy Anatomy



A Kyverno Policy

```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: require-labels
spec:
  validationFailureAction: enforce
  rules:
  - name: check-for-labels
    match:
      resources:
        kinds:
        - Pod
    validate:
      message: "label 'app.kubernetes.io/name' is required"
      pattern:
        metadata:
          labels:
            app.kubernetes.io/name: "?*"
```

Validate Policy

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

best_practices > ! disallow_latest_tag.yaml > ...

io.kyverno.v1.ClusterPolicy (v1@clusterpolicy.json) | You, last month | 1 author (You)

```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: disallow-latest-tag
spec:
  validationFailureAction: Enforce
  background: true
  rules:
  - name: validate-image-tag
    match:
      resources:
        kinds:
        - Pod
    validate:
      message: "An image tag is required; latest is not allowed"
      pattern:
        spec:
          containers:
          - image: "!*:latest & *:*"
```

Mutate Policy

- JSON Patch (RFC 6902)
  - Use for precise updates
- StrategicMergePatch
  - Use for describing intent
  - Anchors for conditional logic
    - “If-then-else”
    - “if-not-defined”

```yaml
mutate:
  patches:
  - path: "/spec/template/spec/initContainers/0/"
    op: add
    value:
      - image: "nirmata.io/kube-vault-client:v2"
        name: "init-secrets"
```

```yaml
mutate:
  overlay:
    subsets:
    - ports:
      - (name): "secure*"
        port: 6443
```

```yaml
mutate:
  overlay:
    subsets:
    - ports:
      +(port): 6443
```

Generate Policy

- Triggers when a new resource is created or based on label and metadata changes
- Useful in creating defaults for a namespace
- Clones existing resources or copies in-line data
- Can optionally keep data in-sync across namespaces

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

Image Verification Policy

- Native Sigstore and Notary support
- Match images using wildcards
- Verify multiple signatures
- Optional signature registry
- Verify Attestations

verify_images > check-images.yaml > {} spec > [ ] rules > {} 0 > [ ] verifyImages

```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: check-image
spec:
  validationFailureAction: enforce
  background: false
  rules:
    - name: check-image
      match:
        resources:
          kinds:
            - Pod
      verifyImages:
      - image: "ghcr.io/kyverno/test-verify-image:*"
        key: |-
          - ---BEGIN PUBLIC KEY-----
          MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE8nXRh950
          IZbRj8Ra/N9sbqOPZrfM5/KAQN0/KjHcorm/J5yctVd7
          iEcnessRQjU917hmKO6JWVGHpDguIyakZA==
          - ---END PUBLIC KEY--- -
```

Cleanup policies

- Delete resources based on flexible match/exclude and conditions
- Run checks periodically using Cron schedule format

```yaml
apiVersion: kyverno.io/v2alpha1
kind: ClusterCleanupPolicy
metadata:
  name: clean-bare-pods
  annotations:
    pod-policies.kyverno.io/autogen-controllers: none
spec:
  match:
    any:
    - resources:
        kinds:
          - Pod
  conditions:
    all:
    - key: "{{ target.metadata.ownerReferences[] || `[]` }}"
      operator: Equals
      value: []
  schedule: "0/1 * * * *"
```

PolicyException

- Decouple exceptions from policies
- Fine-grained exclusions
- Can be combined with other features like TTL and signing

```yaml
apiVersion: kyverno.io/v2alpha1
kind: PolicyException
metadata:
  name: allow-insecure-pod
  namespace: kyverno
spec:
  exceptions:
    - policyName: disallow-capabilities-strict
      ruleNames:
        - require-drop-all
    - policyName: disallow-privilege-escalation
      ruleNames:
        - privilege-escalation
    - policyName: require-run-as-nonroot
      ruleNames:
        - run-as-non-root
    - policyName: restrict-seccomp-strict
      ruleNames:
        - check-seccomp-strict
  match:
    any:
      - resources:
          namespaces:
            - test
```

Policy Report

- Manage reports as K8s resources
- Common API now part of WG-Policy
- KEP to promote to Kubernetes SIG
- Several producers and consumers

```
~ ➜ kubectl get polr -A -o yaml | grep "result: fail" -A 10 -B 2 | more
    message: 'validation failure: Containers must drop `ALL` capabilities.'
    policy: disallow-capabilities-strict
    result: fail
    rule: require-drop-all
    scored: true
    severity: medium
    source: kyverno
    timestamp:
      nanos: 0
      seconds: 1707951292
  - category: Pod Security Standards (Baseline)
    message: validation rule 'host-namespaces' passed.
    policy: disallow-host-namespaces
```

Additional Features

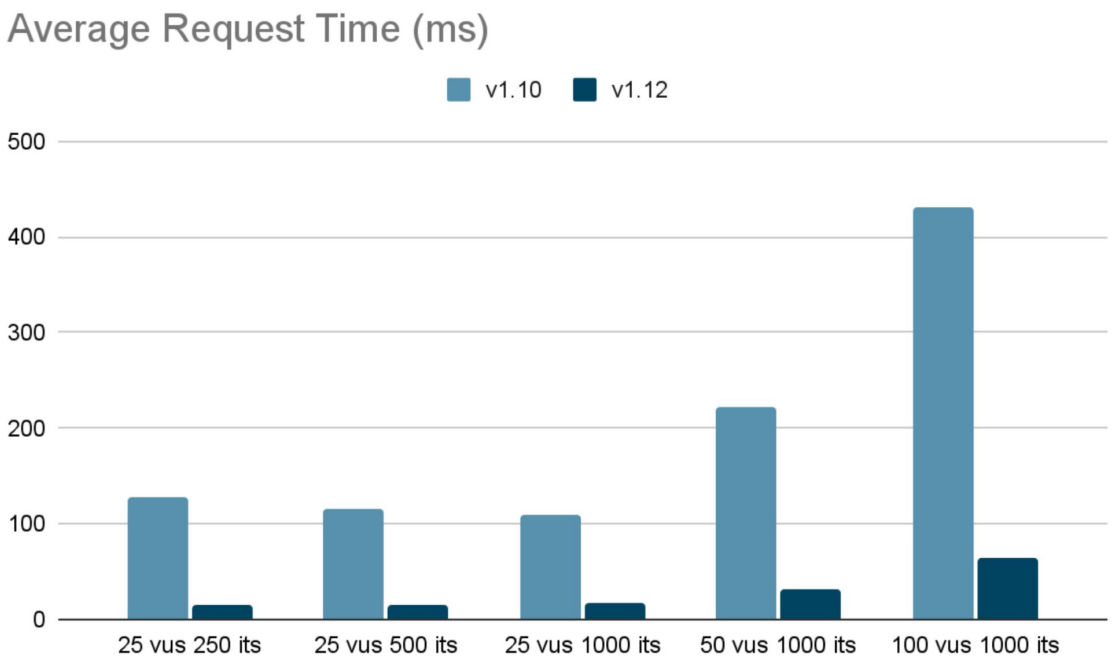
- Built-in variables (resource, images, users, …)
- Custom context variables from API lookups, OCI registries, ConfigMap, or admission payloads
- JMESPath support for complex logic with custom functions for X.509 certs, regex, time, etc.
- OCI Registry integrations for manifests and configuration validation
- Mutate existing workloads

Additional Features (2)

- Cleanup using TTL annotations
- YAML Signing
- CEL support
- ValidationAdmissionPolicy lifecycle management
- Unit testing with Kyverno CLI
- Declarative e2e tests with Kyverno Chainsaw

1.12 Major Features

- PolicyReport API Aggregation
- Global Context with flexible caching
- Fine-grained webhook management
- Kyverno JSON integration with CLI
- Performance improvements



1.13 Major Features

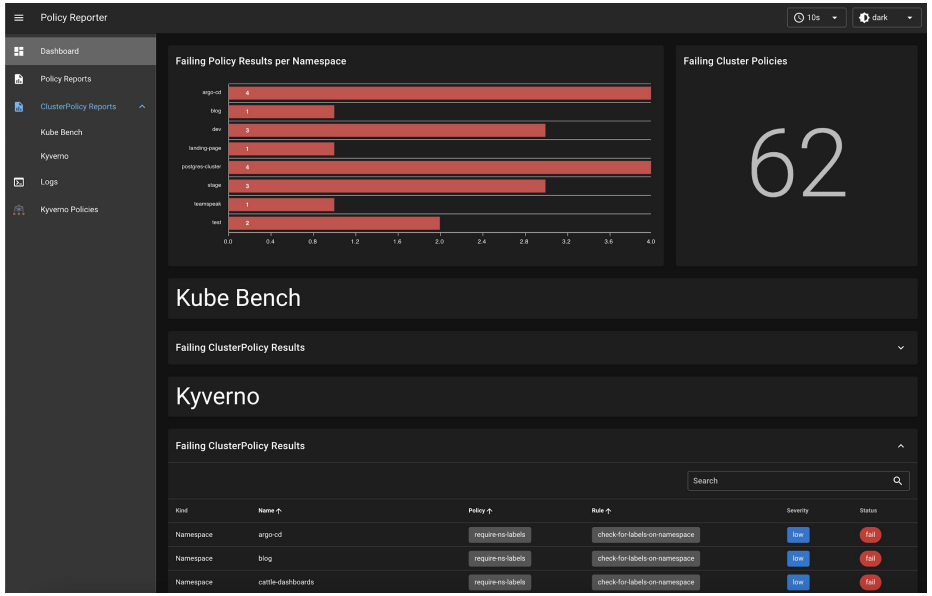
- Sigstore Bundle Verification
- Exceptions for Validating Admission Policies
- Assertion Trees in Validation Rules
- Warnings for Policy Violations
- Generate Foreach
- Improved ArgoCD integration
- PolicyException and CleanupPolicy promoted to GA
- Shallow evaluation of variables
- Security Hardening

https://kyverno.io/blog/2024/10/30/announcing-kyverno-release-1.13/

Kyverno Evolution

K8s policies

Policy
Report API



Policy Reporter

Kyverno JSON

apply Kyverno Policies Anywhere

Kyverno Chainsaw

declarative e2e K8s tests

Kyverno “top 10” Features

1. Start with no code; use JMESPath, CEL for complex logic
2. Validate, Mutate, Generate, Cleanup resources
3. Integrated image verification with Cosign and Notary
4. API calls and extensions
5. Integrated unit and e2e test tools
6. CLI to apply policies off-cluster
7. Kubernetes native policy reporting
8. Kubernetes native policy exceptions
9. Fine-grained automated webhook management
10. Auto-Gen ValidatingAdmissionPolicy resources

Kyverno Use Cases

SecOps

- Pod security
- Workload security
- Granular RBAC
- Workload isolation
- Image signing & verification
- Workload identity

DevOps

- Self-service Kubernetes environments
- Self-service infrastructure (IaC)
- Resource governance and cleanup
- Label/Annotation management
- Naming conventions
- Event driven automation
- Custom CA management
- Time-bound policies

FinOps

- Quota Management
- Pod Requests and limits
- Team and app labels
- Scaling limits
- Scheduled resources
- QoS management
- Auto-scalers

Thank-You!

https://kyverno.io