

From Fragile to Resilient: Validating Admission Policies Strengthen Kubernetes

Cloud Native Rejekts NA

November 10th 2024



Hi 🙋,

I'm **Marcus Noble**!

I'm a platform engineer at  **Giant Swarm** working on release engineering, CI/CD infrastructure and provider-independent Kubernetes development.

6+ years experience running Kubernetes in production environments.

Fediverse at: **@Marcus@k8s.social**
Everywhere else: **MarcusNoble.com**



Dynamic Admission Control

**ValidatingAdmission
Webhook**

**MutatingAdmission
Webhook**

Purpose / Use Cases

Defaulting

- Injecting `imagePullSecrets` dynamically when pods are created
- Injecting sidecars into pods
- Injecting proxy environment variables into pods

- Require a `PodDisruptionBudget`
- Enforce a standard set of labels / annotations on all resources
- Replace all image registries with an in-house container image proxy / cache

Best Practices

Policy Enforcement

- Prevent using `latest` image tag
- Require all pods to have resource limits set
- Block the use of deprecated Kubernetes APIs (e.g. `batch/v1beta1`)

- Block nodes joining the cluster with known CVEs based on the kernel version (e.g. CVE-2022-0185)
- Inject Log4Shell mitigation env var into all pods (CVE-2021-44228)
- Block binding to the cluster-admin role

Problem Mitigation

Webhooks in Kubernetes are

✨ **POWERFUL** ✨

But with that power comes

😱 **RISK** 😱



*Taken from my talk about this
at Rejekts in Chicago*

Cloud Native
Rejekts [RM'23]

NOVEMBER 4-5 • CHICAGO

Webhooks - what's the worst that could happen?



Marcus Noble
Platform Engineer
Grant Swarm

Stay Connected to the
Cloud Native Rejekts Community

📧 @Rejektsio 📧 @Rejektsio@phachyderm.io

REJEKTS2023
cloud-native.rejekts.io

KUBERNETES

**Wouldn't it be great if we
had a safer alternative?**

Yes! Yes, it would!

Introducing Validating Admission Policies

Status: Alpha in v1.26, Beta in v1.28, GA in v1.30

Introduced in: [KEP-3488](#)

A declarative, in-process alternative to validating admission webhooks.

Introducing Validating Admission Policies

Status: Alpha in v1.26, Beta in v1.28, GA in v1.30

Introduced in: [KEP-3488](#)

A declarative, in-process alternative to validating admission webhooks.

Kubernetes manifests 

Introducing Validating Admission Policies

Status: Alpha in v1.26, Beta in v1.28, GA in v1.30

Introduced in: [KEP-3488](#)

 *api-server*

A declarative, in-process alternative to validating admission webhooks.

Kubernetes manifests 

Introducing Validating Admission Policies

Status: Alpha in v1.26, Beta in v1.28, GA in v1.30

Introduced in: [KEP-3488](#)

 *api-server*

A declarative, in-process alternative to validating admission webhooks.

Kubernetes manifests 

 *More on this shortly*

Uses the Common Expression Language (CEL) for the policy language.

Introducing Validating Admission Policies

Status: Alpha in v1.26, Beta in v1.28, GA in v1.30

Introduced in: [KEP-3488](#)

 *api-server*

A declarative, in-process alternative to validating admission webhooks.

Kubernetes manifests 

 *More on this shortly*

Uses the Common Expression Language (CEL) for the policy language.

Consists of two main resources:

- `ValidatingAdmissionPolicy` describes the policy logic
- `ValidatingAdmissionPolicyBinding` links the above policy to the resources it applies to

Brief introduction to CEL

- Uses a similar syntax to the expressions in C-based languages, e.g.

```
self.minReplicas <= self.replicas &&  
self.replicas <= self.maxReplicas
```

- Designed to be embedded into other applications with a focus on “one-liners” of code.
- Small number of built in functions (e.g. `split`, `has`)
- Functions expanded through custom libraries (Kubernetes includes several of these)
- Already used by [Kyverno](#), [Tekton](#), etc.
- Has a concept of “cost” per operation.

This can be calculated ahead of running the expression and if needed, block its execution if too expensive.

References:

- <https://kubernetes.io/docs/reference/using-api/cel/>
- <https://github.com/google/cel-go>
- <https://playcel.undistro.io/> (CEL playground)

Example

Blocking the use of the 'latest' image tag

prevent-latest-vap.yaml

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
metadata:
  name: "prevent-latest-image-tag"
spec:
  failurePolicy: Fail
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE", "UPDATE"]
        resources: ["deployments", "daemonsets", "statefulsets"]
  validations:
    - message: "The use of the 'latest' tag is not allowed"
      expression: |
        object.spec.template.spec.containers.all(container,
          !container.image.endsWith(":latest") && container.image.contains(":")
        ) &&
        (
          !has(object.spec.template.spec.initContainers) ||
          object.spec.template.spec.initContainers.all(container,
            !container.image.endsWith(":latest") && container.image.contains(":")
          )
        )
    )
```

Example

Blocking the use of the 'latest' image tag

prevent-latest-vap.yaml

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
metadata:
  name: "prevent-latest-image-tag"
spec:
  failurePolicy: Fail
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE", "UPDATE"]
        resources: ["deployments", "daemonsets", "statefulsets"]
  validations:
    - message: "The use of the 'latest' tag is not allowed"
      expression: |
        object.spec.template.spec.containers.all(container,
          !container.image.endsWith(":latest") && container.image.contains(":")
        ) &&
        (
          !has(object.spec.template.spec.initContainers) ||
          object.spec.template.spec.initContainers.all(container,
            !container.image.endsWith(":latest") && container.image.contains(":")
          )
        )
    )
```

The name of our policy.
We'll reference this in our
**ValidatingAdmission
PolicyBinding**

Example

Blocking the use of the 'latest' image tag

prevent-latest-vap.yaml

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
metadata:
  name: "prevent-latest-image-tag"
spec:
  failurePolicy: Fail
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE", "UPDATE"]
        resources: ["deployments", "daemonsets", "statefulsets"]
  validations:
    - message: "The use of the 'latest' tag is not allowed"
      expression: |
        object.spec.template.spec.containers.all(container,
          !container.image.endsWith(":latest") && container.image.contains(":")
        ) &&
        (
          !has(object.spec.template.spec.initContainers) ||
          object.spec.template.spec.initContainers.all(container,
            !container.image.endsWith(":latest") && container.image.contains(":")
          )
        )
    )
```

We want our policy to block any incoming requests that don't meet this policies requirements (default).

The alternative is **Ignore** if you want to disable enforcement of this policy.

Example

Blocking the use of the 'latest' image tag

prevent-latest-vap.yaml

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
metadata:
  name: "prevent-latest-image-tag"
spec:
  failurePolicy: Fail
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE", "UPDATE"]
        resources: ["deployments", "daemonsets", "statefulsets"]
  validations:
    - message: "The use of the 'latest' tag is not allowed"
      expression: |
        object.spec.template.spec.containers.all(container,
          !container.image.endsWith(":latest") && container.image.contains(":")
        ) &&
        (
          !has(object.spec.template.spec.initContainers) ||
          object.spec.template.spec.initContainers.all(container,
            !container.image.endsWith(":latest") && container.image.contains(":")
          )
        )
      )
```



We define what resources this policy applies to.

Multiple resource types can be defined here but if they don't have the same general API the CEL expression will quickly become very complex.

Example

Blocking the use of the 'latest' image tag

prevent-latest-vap.yaml

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
metadata:
  name: "prevent-latest-image-tag"
spec:
  failurePolicy: Fail
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE", "UPDATE"]
        resources: ["deployments", "daemonsets", "statefulsets"]
  validations:
    - message: "The use of the 'latest' tag is not allowed"
      expression: |
        object.spec.template.spec.containers.all(container,
          !container.image.endsWith(":latest") && container.image.contains(":")
        ) &&
        (
          !has(object.spec.template.spec.initContainers) ||
          object.spec.template.spec.initContainers.all(container,
            !container.image.endsWith(":latest") && container.image.contains(":")
          )
        )
    )
```

We'll add a human-friendly message to be shown when the API request has been blocked by this policy.

If we don't include this, a generic message that include the whole expression is shown instead.

Example

Blocking the use of the 'latest' image tag

prevent-latest-vap.yaml

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
metadata:
  name: "prevent-latest-image-tag"
spec:
  failurePolicy: Fail
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE", "UPDATE"]
        resources: ["deployments", "daemonsets", "statefulsets"]
  validations:
    - message: "The use of the 'latest' tag is not allowed"
      expression: |
        object.spec.template.spec.containers.all(container,
          !container.image.endsWith(":latest") && container.image.contains(":")
        ) &&
        (
          !has(object.spec.template.spec.initContainers) ||
          object.spec.template.spec.initContainers.all(container,
            !container.image.endsWith(":latest") && container.image.contains(":")
          )
        )
      )
```

Finally we have our expression.

This is the expression of **allowed** resources, not those to block.

(I keep getting caught out by this 😊)

Example

Blocking the use of the 'latest' image tag

prevent-latest-vap.yaml

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
metadata:
  name: "prevent-latest-image-tag"
spec:
  failurePolicy: Fail
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE", "UPDATE"]
        resources: ["deployments", "daemonsets", "statefulsets"]
  validations:
    - message: "The use of the 'latest' tag is not allowed"
      expression: |
        object.spec.template.spec.containers.all(container,
          !container.image.endsWith(":latest") && container.image.contains(":")
        ) &&
        (
          !has(object.spec.template.spec.initContainers) ||
          object.spec.template.spec.initContainers.all(container,
            !container.image.endsWith(":latest") && container.image.contains(":")
          )
        )
      )
```

object is the
incoming resource
from the API call.

Example

Blocking the use of the 'latest' image tag

prevent-latest-vap.yaml

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
metadata:
  name: "prevent-latest-image-tag"
spec:
  failurePolicy: Fail
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE", "UPDATE"]
        resources: ["deployments", "daemonsets", "statefulsets"]
  validations:
    - message: "The use of the 'latest' tag is not allowed"
      expression: |
        object.spec.template.spec.containers.all(container,
          !container.image.endsWith(":latest") && container.image.contains(":")
        ) &&
        (
          !has(object.spec.template.spec.initContainers) ||
          object.spec.template.spec.initContainers.all(container,
            !container.image.endsWith(":latest") && container.image.contains(":")
          )
        )
      )
```

First we check all containers don't have the latest tag (or no tag specified).

Example

Blocking the use of the 'latest' image tag

prevent-latest-vap.yaml

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
metadata:
  name: "prevent-latest-image-tag"
spec:
  failurePolicy: Fail
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE", "UPDATE"]
        resources: ["deployments", "daemonsets", "statefulsets"]
  validations:
    - message: "The use of the 'latest' tag is not allowed"
      expression: |
        object.spec.template.spec.containers.all(container,
          !container.image.endsWith(":latest") && container.image.contains(":")
        ) &&
        (
          !has(object.spec.template.spec.initContainers) ||
          object.spec.template.spec.initContainers.all(container,
            !container.image.endsWith(":latest") && container.image.contains(":")
          )
        )
      )
```

Then we check if this resource defines initContainers and if so we check they also don't use latest.

Example

Blocking the use of the 'latest' image tag

Our policy does nothing until we bind it to some conditions.

prevent-latest-binding.yaml

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicyBinding
metadata:
  name: "prevent-latest-image-tag"
spec:
  policyName: "prevent-latest-image-tag"
  validationActions: [Deny]
  matchResources: {}
```

Example

Blocking the use of the 'latest' image tag

The name of the policy we have just created.

prevent-latest-binding.yaml

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicyBinding
metadata:
  name: "prevent-latest-image-tag"
spec:
  policyName: "prevent-latest-image-tag"
  validationActions: [Deny]
  matchResources: {}
```

Example

Blocking the use of the 'latest' image tag

The action to take when a policy isn't met.

Available options are:
Deny, **Warn** & **Audit**

prevent-latest-binding.yaml

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicyBinding
metadata:
  name: "prevent-latest-image-tag"
spec:
  policyName: "prevent-latest-image-tag"
  validationActions: [Deny]
  matchResources: {}
```

Note: this is an array as you can specify both warn and audit together

Example

Blocking the use of the 'latest' image tag

We're not defining any filtering so this policy applies cluster-wide.

We could limit our policy to specific namespaces or labels, for example.

prevent-latest-binding.yaml

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicyBinding
metadata:
  name: "prevent-latest-image-tag"
spec:
  policyName: "prevent-latest-image-tag"
  validationActions: [Deny]
  matchResources: {}
```

Example

Blocking the use of the 'latest' image tag

We're not defining any filtering so this policy applies cluster-wide.

We could limit our policy to specific namespaces or labels, for example.

prevent-latest-binding.yaml

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicyBinding
metadata:
  name: "prevent-latest-image-tag"
spec:
  policyName: "prevent-latest-image-tag"
  validationActions: [Deny]
  matchResources:
    namespaceSelector:
      matchLabels:
        environment: prod
```



Example

Blocking the use of the 'latest' image tag

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-blocked
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
```

```
# kubectl apply -f deployment.yaml
```

The deployments "nginx-blocked" is invalid: : ValidatingAdmissionPolicy 'prevent-latest-image-tag' with binding 'prevent-latest-image-tag' denied request: The use of the 'latest' tag is not allowed

Our friendly message ↪



Blocked!



More advanced features

More advanced features

- **More context values** - Along with `object` you also have `oldObject` & `request` you can use in your expressions

```
matchConditions:
```

```
- name: "exclude-kubelet-requests"
```

```
  expression: "!("system:nodes" in request.userInfo.groups)"
```

More advanced features

- **More context values** - Along with `object` you also have `oldObject` & `request` you can use in your expressions
- **Parameters** - make your policies configurable by allowing parameter resources to be applied by the binding

policy.yaml

```
apiVersion: admissionregistration.k8s...
kind: ValidatingAdmissionPolicy
metadata:
  name: "param-example"
spec:
  paramKind:
    apiVersion: rules.example.com/v1
    kind: ReplicaLimit
  validations:
    - expression: object.spec.replicas <=
      params.maxReplicas
```

policy-binding.yaml

```
apiVersion: admissionregistration.k8s...
kind: ValidatingAdmissionPolicyBinding
metadata:
  name: "param-example"
spec:
  policyName: "param-example"
  paramRef:
    name: "my-parameters"
    namespace: "default"
```

params.yaml

```
apiVersion: rules.example.com/v1
kind: ReplicaLimit
metadata:
  name: "my-parameters"
  namespace: "default"
maxReplicas: 3
```

More advanced features

- **More context values** - Along with `object` you also have `oldObject` & `request` you can use in your expressions
- **Parameters** - make your policies configurable by allowing parameter resources to be applied by the binding
- **Variables** - reusable CEL expressions to simplify your validation expressions.

```
spec:
  variables:    ↪ Must be a valid CEL identifier (e.g. no '-' )
    - name: teamLabel
      expression: "'team' in object.spec.metadata.labels ? object.spec.metadata.labels['team'] : 'no-team'"
  validations:
    - expression: variables.teamLabel == 'rel-eng'
```

More advanced features

- **More context values** - Along with `object` you also have `oldObject` & `request` you can use in your expressions
- **Parameters** - make your policies configurable by allowing parameter resources to be applied by the binding
- **Variables** - reusable CEL expressions to simplify your validation expressions.
- **Audit annotations** - Add extra metadata to the audit logs, dynamic values using CEL

```
auditAnnotations:  
- key: "replica-count"  
  valueExpression: |  
    "Deployment spec.replicas set to " + string(object.spec.replicas)"
```

```
{  
  "kind": "Event",  
  "apiVersion": "audit.k8s.io/v1",  
  "annotations": {  
    "demo-policy.example.com/replica-count":  
      "Deployment spec.replicas set to 128"  
    ...  
  }  
}
```


More advanced features

- **More context values** - Along with `object` you also have `oldObject` & `request` you can use in your expressions
- **Parameters** - make your policies configurable by allowing parameter resources to be applied by the binding
- **Variables** - reusable CEL expressions to simplify your validation expressions.
- **Audit annotations** - Add extra metadata to the audit logs, dynamic values using CEL
- **Message expressions** - Leverage some CEL in your message to have dynamic validation messages

```
messageExpression: "object.spec.replicas must be no greater than ' + string(params.maxReplicas)"
```

**So that's validation covered,
what about mutating?**

Well...

KEP-3962 - Mutating Admission Policies

Status: Currently targeting v1.32 for *alpha*

Work in progress implementation in [#127134](#)

- Follows the same idea as Validating Admission Policies.
- Introduces `MutatingAdmissionPolicy` and `MutatingAdmissionPolicyBinding`
- Supports two patch strategies: `ApplyConfiguration` and `JSONPatch`
- Phase 1 will only support **adding** and **updating** of values (unsetting values planned for phase 2 but will need to get creative as CEL doesn't natively support such an operation)
- Not yet finalized or available for testing (unless you want to build from source)

Example

Injecting proxy values as env vars

proxy-vars-map.yaml

```
apiVersion: admissionregistration.k8s.io/v1alpha1
kind: MutatingAdmissionPolicy
metadata:
  name: "proxy-values"
spec:
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE"]
        resources: ["pods"]
  mutations:
    - patchType: "ApplyConfiguration"
      mutation: >
        Object{
          spec: Object.spec{
            containers: object.spec.containers.map(c,
              Object.spec.containers.item{
                name: c.name,
                env: [
                  Object.spec.containers.env{
                    name: "HTTP_PROXY",
                    value: "http://proxy.proxy.svc:3128"
                  } + c.env
                ]
              }
            )
          }
        }
      }
```

Example

Injecting proxy values as env vars

Same as ValidatingAdmissionPolicies →

proxy-vars-map.yaml

```
apiVersion: admissionregistration.k8s.io/v1alpha1
kind: MutatingAdmissionPolicy
metadata:
  name: "proxy-values"
spec:
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE"]
        resources: ["pods"]
  mutations:
    - patchType: "ApplyConfiguration"
      mutation: >
        Object{
          spec: Object.spec{
            containers: object.spec.containers.map(c,
              Object.spec.containers.item{
                name: c.name,
                env: [
                  Object.spec.containers.env{
                    name: "HTTP_PROXY",
                    value: "http://proxy.proxy.svc:3128"
                  } + c.env
                ]
              }
            )
          }
        }
      }
```

Example

Injecting proxy values as env vars

Replace **validations** with **mutations**



proxy-vars-map.yaml

```
apiVersion: admissionregistration.k8s.io/v1alpha1
kind: MutatingAdmissionPolicy
metadata:
  name: "proxy-values"
spec:
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE"]
        resources: ["pods"]
  mutations:
    - patchType: "ApplyConfiguration"
      mutation: >
        Object{
          spec: Object.spec{
            containers: object.spec.containers.map(c,
              Object.spec.containers.item{
                name: c.name,
                env: [
                  Object.spec.containers.env{
                    name: "HTTP_PROXY",
                    value: "http://proxy.proxy.svc:3128"
                  } + c.env
                ]
              }
            )
          }
        }
      }
```

Example

Injecting proxy values as env vars

Indicates the patch strategy

Possible values:

- `ApplyConfiguration`
- `JSONPatch`

proxy-vars-map.yaml

```
apiVersion: admissionregistration.k8s.io/v1alpha1
kind: MutatingAdmissionPolicy
metadata:
  name: "proxy-values"
spec:
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE"]
        resources: ["pods"]
  mutations:
    - patchType: "ApplyConfiguration"
      mutation: >
        Object{
          spec: Object.spec{
            containers: object.spec.containers.map(c,
              Object.spec.containers.item{
                name: c.name,
                env: [
                  Object.spec.containers.env{
                    name: "HTTP_PROXY",
                    value: "http://proxy.proxy.svc:3128"
                  } + c.env
                ]
              }
            )
          }
        }
      }
```

Example

Injecting proxy values as env vars

Introduction of named types

The Object refers to the type of the incoming resource (Pod in this example).

This *may* change in later releases to something like `v1.Pod.spec` for example.

This has memory consumption implications.

proxy-vars-map.yaml

```
apiVersion: admissionregistration.k8s.io/v1alpha1
kind: MutatingAdmissionPolicy
metadata:
  name: "proxy-values"
spec:
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE"]
        resources: ["pods"]
  mutations:
    - patchType: "ApplyConfiguration"
      mutation: >
        Object{
          spec: Object.spec{
            containers: object.spec.containers.map(c,
              Object.spec.containers.item{
                name: c.name,
                env: [
                  Object.spec.containers.env{
                    name: "HTTP_PROXY",
                    value: "http://proxy.proxy.svc:3128"
                  } + c.env
                ]
              }
            )
          }
        }
      }
```


Example

Injecting proxy values as env vars

Merge our proxy vars with any existing vars in all containers.

Note: We're missing `initContainers` and `ephemeralContainers` here due to limited space.

proxy-vars-map.yaml

```
apiVersion: admissionregistration.k8s.io/v1alpha1
kind: MutatingAdmissionPolicy
metadata:
  name: "proxy-values"
spec:
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE"]
        resources: ["pods"]
  mutations:
    - patchType: "ApplyConfiguration"
      mutation: >
        Object{
          spec: Object.spec{
            containers: object.spec.containers.map(c,
              Object.spec.containers.item{
                name: c.name,
                env: [
                  Object.spec.containers.env{
                    name: "HTTP_PROXY",
                    value: "http://proxy.proxy.svc:3128"
                  } + c.env
                ]
              }
            )
          }
        }
      }
```

*I'm not 100%
sure about
this type*

Example

Injecting proxy values as env vars

proxy-vars-binding.yaml

```
apiVersion: admissionregistration.k8s.io/v1beta1
kind: MutatingAdmissionPolicyBinding
metadata:
  name: "proxy-values"
spec:
  policyName: "proxy-values"
  matchResources: {}
```

Nothing too surprising about our binding resource. All properties are the same as Validating except for the lack of a `validationActions` property.

What about the future beyond that?



- Generative policies
- Resource lookup
- Policy exceptions
- More abstractions (e.g. Kyverno already working on this with VAP)

Summary

- Time to start replacing those risky webhooks with in-process policies.
- ValidatingAdmissionPolicies generally available for use from **v1.30** for safe validation logic.
- Keep an eye on KEP-3962 for the status of MutatingAdmissionPolicies. Currently targeting **v1.32** for *alpha release*.
- More abstractions, generative policies and api lookups hopefully coming in the future.

Wrap-up

Slides and resources available at:

<https://go-get.link/rejektsna24>



Thoughts, comments and feedback:



feedback@marcusnoble.co.uk



<https://k8s.social/@Marcus>

Thank you

