
KubeCon
CloudNativeCon
North America 2019

Admission Webhooks: Configuration and Debugging Best Practices

Haowei Cai (@roycaihw), Google

About me

Haowei Cai (@roycaihw)

Software Engineer for Google Cloud. He is an active contributor for Kubernetes SIG API Machinery.

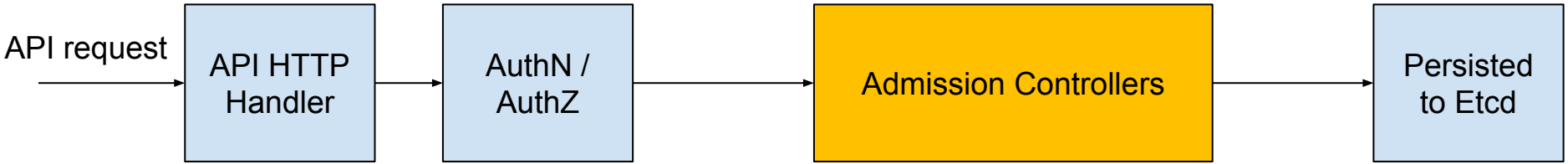
Agenda

- **What** are Admission Webhooks?
- **How to configure** my admission webhooks following the **best practices**?
- **How to debug** my admission webhooks?
- **Demo**
- **Key Takeaways**

What are Admission Webhooks?

Admission

- Admission Controllers
  - `kube-apiserver --enable-admission-plugins=NamespaceLifecycle,LimitRanger ...`

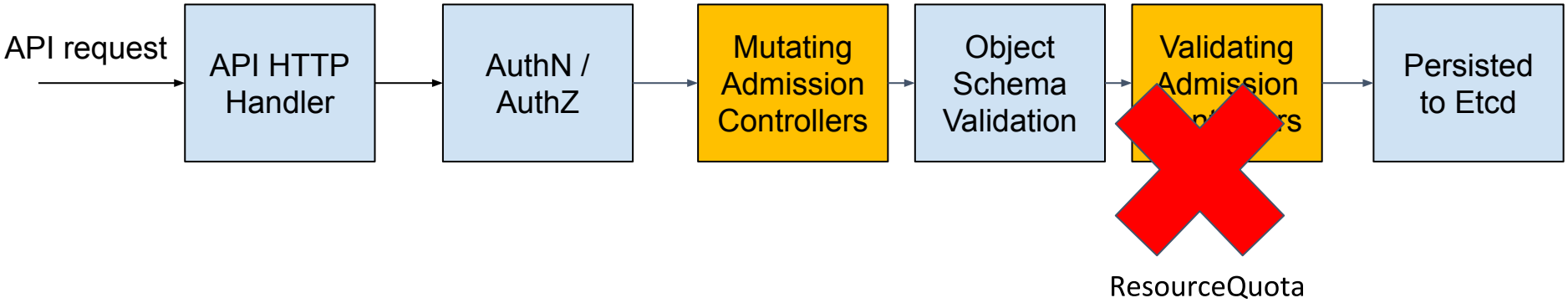


Admission

- Admission Controllers
  - Mutating
  - Validating

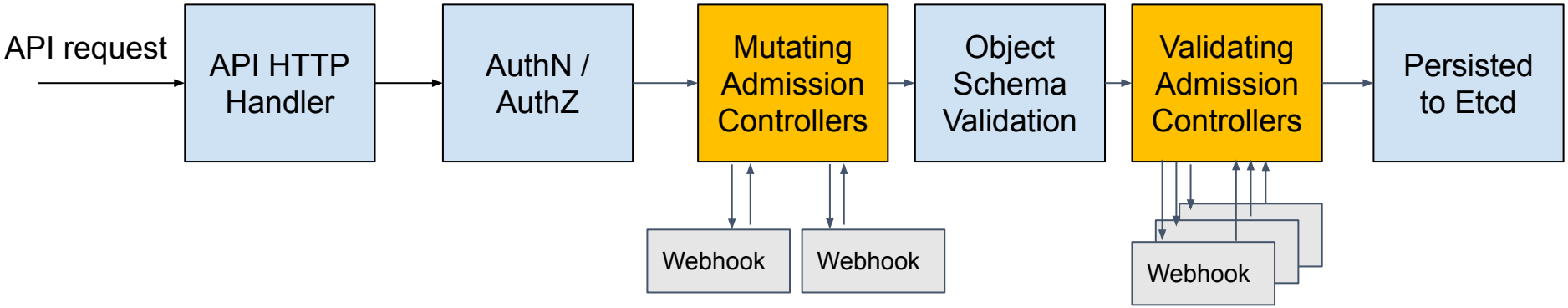
Admission

- Admission Controllers
  - Mutating
  - Validating



Admission Webhooks

- Admission Controllers
- Dynamic Admission Controllers:
  - **MutatingAdmissionWebhooks**
  - **ValidatingAdmissionWebhooks**



Why do I need admission webhooks?

- What built-in admission controllers do:
  - Security
  - Governance
  - Configuration management
  - etc..


KubeCon | CloudNativeCon

North America 2019

How to configure my admission webhooks?

Configuration fields


```yaml
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  name: "pod-policy.example.com"
webhooks:
- name: "pod-policy.example.com"
  rules:
  - apiGroups:   [""]
    apiVersions: ["v1"]
    operations:  ["CREATE"]
    resources:   ["pods"]
    scope:       "Namespaced"
  clientConfig:
    service:
      namespace: "example-namespace"
      name: "example-service"
    caBundle: "Ci0tLS0tQk...<base64-encoded PEM bundle containing the CA that signed the webhook's serving certificate>...tLS0K"
  admissionReviewVersions: ["v1", "v1beta1"]
  sideEffects: None
  timeoutSeconds: 5
```


KubeCon | CloudNativeCon

North America 2019

Configuration best practices

Best practices

- Idempotence
- Intercepting all versions of an object
- Availability
- Guaranteeing the final state of the object is seen
- Side effects
- Avoiding operating on the kube-system namespace

1. Idempotence

- Why did Kubernetes API Server call my mutating webhook **twice for the same request**?

1. Idempotence

- Why did Kubernetes API Server call my mutating webhook **twice**?
  - **Ordering** is hard
    - Some other admission controller may mutate the object
    - The decision can be **different** depending on the order

1. Idempotence

- Why did Kubernetes API Server call my mutating webhook **twice**?
  - **Ordering** is hard
    - Some other admission controller may mutate the object
    - The decision can be **different** depending on the order

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      command: [ "echo", "SUCCESS" ]
```

Inject Sidecar →

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      command: [ "echo", "SUCCESS" ]
    - name: sidecar
      image: $SIDECAR_IMAGE_NAME
      command: [ "echo", "SIDECAR" ]
```

1. Idempotence

- Why did Kubernetes API Server call my mutating webhook **twice**?
  - **Ordering** is hard
    - Some other admission controller may mutate the object
    - The decision can be **different** depending on the order

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      command: [ "echo", "SUCCESS" ]
```

Inject Sidecar →

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      command: [ "echo", "SUCCESS" ]
    - name: sidecar
      image: $SIDECAR_IMAGE_NAME
      command: [ "echo", "SIDECAR" ]
```

Always Pull Images →

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      imagePullPolicy: Always
      command: [ "echo", "SUCCESS" ]
    - name: sidecar
      image: $SIDECAR_IMAGE_NAME
      imagePullPolicy: Always
      command: [ "echo", "SIDECAR" ]
```

1. Idempotence

- Why did Kubernetes API Server call my mutating webhook **twice**?
  - **Ordering** is hard
    - Some other admission controller may mutate the object
    - The decision can be **different** depending on the order

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      command: [ "echo", "SUCCESS" ]
```

Always Pull Images →

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      imagePullPolicy: Always
      command: [ "echo", "SUCCESS" ]
```

1. Idempotence

- Why did Kubernetes API Server call my mutating webhook **twice**?
  - **Ordering** is hard
    - Some other admission controller may mutate the object
    - The decision can be **different** depending on the order

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      command: [ "echo", "SUCCESS" ]
```

Always Pull Images →

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      imagePullPolicy: Always
      command: [ "echo", "SUCCESS" ]
```

Inject Sidecar →

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      imagePullPolicy: Always
      command: [ "echo", "SUCCESS" ]
    - name: sidecar
      image: $SIDECAR_IMAGE_NAME
      command: [ "echo", "SIDECAR" ]
```

1. Idempotence

- Why did Kubernetes API Server call my mutating webhook **twice**?
  - **Ordering** is hard
    - Some other admission controller may mutate the object
    - The decision can be **different** depending on the order

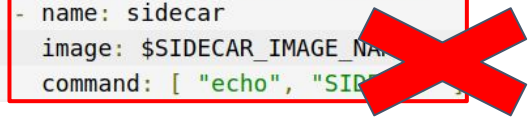
```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      command: [ "echo", "SUCCESS" ]
```

Always Pull Images →

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      imagePullPolicy: Always
      command: [ "echo", "SUCCESS" ]
```

Inject Sidecar →

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      imagePullPolicy: Always
      command: [ "echo", "SUCCESS" ]
    - name: sidecar
      image: $SIDECAR_IMAGE_NA
      command: [ "echo", "SID
```



1. Idempotence

- Why did Kubernetes API Server call my mutating webhook **twice**?
  - **Ordering** is hard
    - Some other admission controller may mutate the object
    - The decision can be **different** depending on the order
  - **Best effort re-invocation** to make sure everyone see the latest state -> reinvocationPolicy: IfNeeded

Inject Sidecar →

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      imagePullPolicy: Always
      command: [ "echo", "SUCCESS" ]
    - name: sidecar
      image: $SIDECAR_IMAGE_NAME
      command: [ "echo", "SIDECAR" ]
```

Always Pull Images →

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      imagePullPolicy: Always
      command: [ "echo", "SUCCESS" ]
    - name: sidecar
      image: $SIDECAR_IMAGE_NAME
      imagePullPolicy: Always
      command: [ "echo", "SIDECAR" ]
```

1. Idempotence

- **Best practice: be idempotent**
  - (From Wikipedia:) *Idempotence is the property of certain operations in mathematics and computer science whereby they can be applied multiple times without changing the result beyond the initial application.*

Example of idempotent webhook


```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      command: [ "echo", "SUCCESS" ]
```

Always Pull Images →

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      imagePullPolicy: Always
      command: [ "echo", "SUCCESS" ]
```

Example of idempotent webhook


```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      command: [ "echo", "SUCCESS" ]
```

Always Pull Images →

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      imagePullPolicy: Always
      command: [ "echo", "SUCCESS" ]
```

Always Pull Images →

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      imagePullPolicy: Always
      command: [ "echo", "SUCCESS" ]
```

Example of non-idempotent webhook


```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      command: [ "echo", "SUCCESS" ]
```

Inject Sidecar →

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      command: [ "echo", "SUCCESS" ]
    - name: sidecar-19700101-000000
      image: $SIDECAR_IMAGE_NAME
      command: [ "echo", "SIDECAR" ]
```

Example of non-idempotent webhook


```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      command: [ "echo", "SUCCESS" ]
```

Inject Sidecar →

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      command: [ "echo", "SUCCESS" ]
    - name: sidecar-19700101-000000
      image: $SIDECAR_IMAGE_NAME
      command: [ "echo", "SIDECAR" ]
```

Inject Sidecar →

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: $MY_IMAGE_NAME
      command: [ "echo", "SUCCESS" ]
    - name: sidecar-19700101-000000
      image: $SIDECAR_IMAGE_NAME
      command: [ "echo", "SIDECAR" ]
    - name: sidecar-19700101-000050
      image: $SIDECAR_IMAGE_NAME
      command: [ "echo", "SIDECAR" ]
```

2. Intercepting all versions of an object

- **Deployment** API:
  - extensions/v1beta1
  - apps/v1beta1
  - apps/v1beta2
  - apps/v1

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
...
webhooks:
- name: my-webhook.example.com

  rules:
  - operations: ["CREATE","UPDATE","DELETE"]
    apiGroups: ["apps"]
    apiVersions: ["v1"]
    resources: ["deployments"]
    scope: "Namespaced"
  ...
```

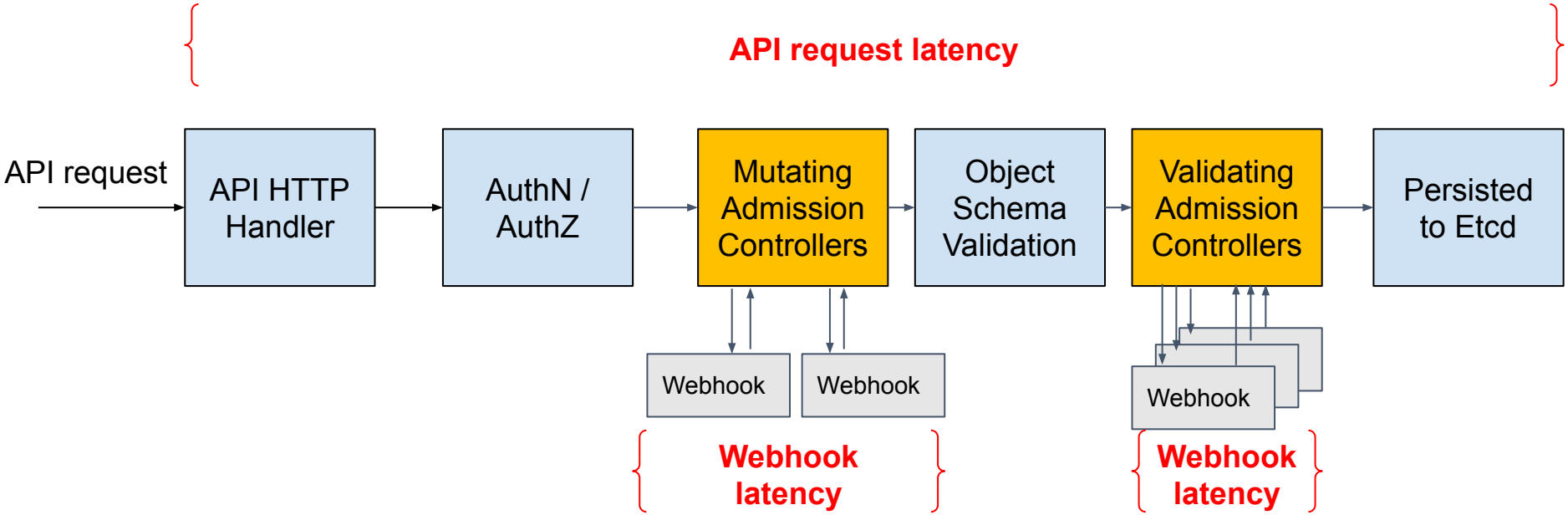
2. Intercepting all versions of an object

- **Deployment** API:
  - extensions/v1beta1
  - apps/v1beta1
  - apps/v1beta2
  - apps/v1

```yaml
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
...
webhooks:
- name: my-webhook.example.com
  matchPolicy: Equivalent
  rules:
  - operations: ["CREATE","UPDATE","DELETE"]
    apiGroups: ["apps"]
    apiVersions: ["v1"]
    resources: ["deployments"]
    scope: "Namespaced"
  ...
```

3. Availability

- Time calling webhook builds-up time completing API requests



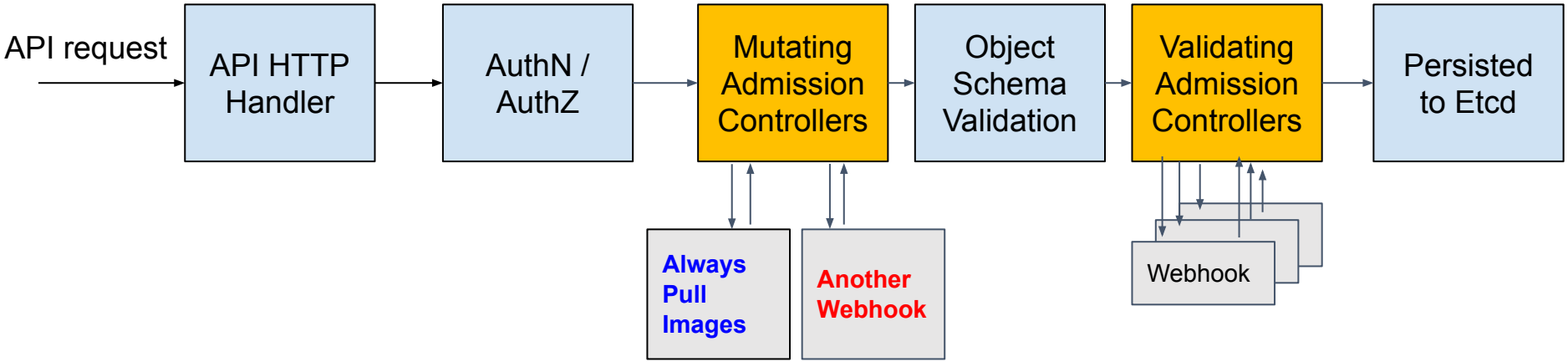
3. Availability

- Time calling webhook builds-up time completing API requests

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
...
webhooks:
- name: my-webhook.example.com
  timeoutSeconds: 2
  ...
```

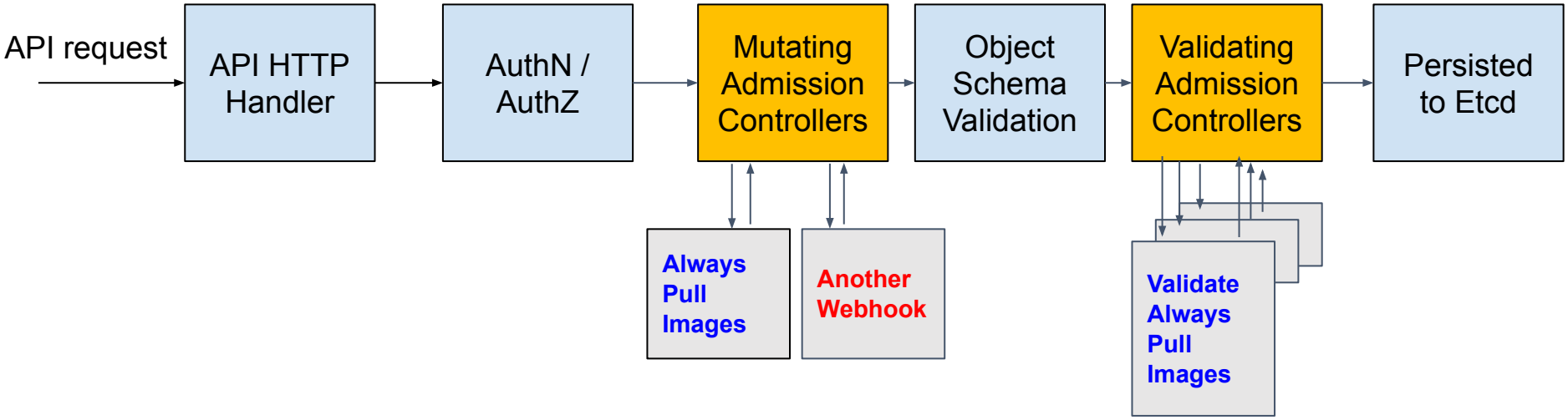
4. Guaranteeing the final state of the object is seen

- If you use a mutating webhook to enforce security policy, make sure to use a validating webhook to ensure that.



4. Guaranteeing the final state of the object is seen

- If you use a mutating webhook to enforce security policy, make sure to use a validating webhook to ensure that.



5. Side effects

- Mutate related resources as part of request processing. E.g.
  - Incrementing quota usage

- Best practices:
  - 1. avoid side effects if possible
  - 2. have a reconciliation mechanism (e.g. a controller) in case the request didn’t make through
  - 3. don’t trigger the side effect in dry run

5. Side effects

- Best practices:
  - 3. don’t trigger the side effect in dry run

```
{
  "apiVersion": "admission.k8s.io/v1",
  "kind": "AdmissionReview",
  "request": {
    ...
    "dryRun": true
  }
}
```

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
...
webhooks:
- name: my-webhook.example.com
  sideEffects: NoneOnDryRun
  ...
```

6. Avoiding operating on the kube-system namespace

- Safety (system-critical components)
  - kube-apiserver post-start hooks
  - Control plane components
    - Service accounts
    - kube-dns
- Efficiency

```
apiVersion: admissionregistration.k8s.io/v1
kind: MutatingWebhookConfiguration
...
webhooks:
- name: my-webhook.example.com
  namespaceSelector:
    matchExpressions:
    - key: runlevel
      operator: NotIn
      values: ["0","1"]
  rules:
  - operations: ["CREATE"]
    apiGroups: ["*"]
    apiVersions: ["*"]
    resources: ["*"]
    scope: "Namespaced"
  ...
```


KubeCon | CloudNativeCon

North America 2019

How to debug my admission webhooks?

Types of webhook failure/rejection

Failure category	Valid webhook rejection		Error calling webhook			apiserver internal error
What happened	403 webhook forbids	500 webhook internal error	Timeout	Connection failure	Malformed webhook response	apiserver internal error
End-user see HTTP status	**403**	500	500	500	500	500

Types of webhook failure/rejection

Failure category	Valid webhook rejection		Error calling webhook			Error calling webhook **failure Policy: Ignore**	apiserver internal error
What happened	403 webhook forbids	500 webhook internal error	Timeout	Connection failure	Malformed webhook response	Timeout/Connection/ Malformed response	apiserver internal error
End-user see HTTP status	**403**	500	500	500	500	**No error**	500

Metrics

- kube-apiserver /metrics endpoint

- **Apiserver_admission_webhook_admission_duration_seconds**
  - Histogram metrics
- **Apiserver_admission_webhook_rejection_count**
  - Counter metrics
    - Name
    - Operation
    - Type
    - Error type
    - Rejection code

Metrics


```
# HELP apiserver_admission_webhook_rejection_count [ALPHA] Admission webhook rejection count, identified by name and broken out for each admission type (validating or admit) and operation. Additional labels specify an error type (calling_webhook_error or apiserver_internal_error if an error occurred; no_error otherwise) and optionally a non-zero rejection code if the webhook rejects the request with an HTTP status code (honored by the apiserver when the code is greater or equal to 400). Codes greater than 600 are truncated to 600, to keep the metrics cardinality bounded.
# TYPE apiserver_admission_webhook_rejection_count counter
apiserver_admission_webhook_rejection_count{error_type="calling_webhook_error",name="allow-configmap-with-delay-webhook.k8s.io",operation="CREATE",rejection_code="0",type="validating"} 1
apiserver_admission_webhook_rejection_count{error_type="calling_webhook_error",name="deny-unwanted-pod-container-name-and-label.k8s.io",operation="CREATE",rejection_code="0",type="validating"} 1
apiserver_admission_webhook_rejection_count{error_type="calling_webhook_error",name="fail-closed.k8s.io",operation="CREATE",rejection_code="0",type="validating"} 1
apiserver_admission_webhook_rejection_count{error_type="no_error",name="deny-attaching-pod.k8s.io",operation="CONNECT",rejection_code="400",type="validating"} 1
apiserver_admission_webhook_rejection_count{error_type="no_error",name="deny-crd-with-unwanted-label.k8s.io",operation="CREATE",rejection_code="400",type="validating"} 1
apiserver_admission_webhook_rejection_count{error_type="no_error",name="deny-unwanted-configmap-data.k8s.io",operation="CREATE",rejection_code="400",type="validating"} 13
```

Besides monitoring

- Why my mutating webhook seems **not working**?
  - E.g. *setting a field* seems to be ignored by the API server
- Why my mutating webhook **worked but in an unexpected way**?
  - Bug in my webhook backend?
  - Bug in API server?
  - Something else mutated the request after my webhook?
  - etc.

Audit Logging

- **Auditing** https://kubernetes.io/docs/tasks/debug-application-cluster/audit/
  - what happened?
  - when did it happen?
  - who initiated it?
  - on what did it happen?
  - where was it observed?
  - from where was it initiated?
  - to where was it going?

Audit Logging


```yaml
apiVersion: audit.k8s.io/v1 # This is required.
kind: Policy
# Don't generate audit events for all requests in RequestReceived stage.
omitStages:
  - "RequestReceived"
rules:
  # Log pod changes at RequestResponse level
  - level: RequestResponse
    resources:
    - group: ""
      # Resource "pods" doesn't match requests to any subresource of pods,
      # which is consistent with the RBAC policy.
      resources: ["pods"]
  # Log "pods/log", "pods/status" at Metadata level
  - level: Metadata
    resources:
    - group: ""
      resources: ["pods/log", "pods/status"]
```

```
{
  "kind": "Event",
  "apiVersion": "audit.k8s.io/v1",
  "level": "Metadata",
  "auditID": "0672151f-0254-4567-ac42-527a054a5e38",
  "stage": "RequestReceived",
  "requestURI": "/apis/events.k8s.io/v1beta1/namespaces/kube-system/events",
  "verb": "create",
  "user": {
    "username": "system:kube-scheduler",
    "groups": [
      "system:authenticated"
    ]
  },
  "sourceIPs": [
    "::1"
  ],
  "userAgent": "kube-scheduler/v1.18.0 (linux/amd64) kubernetes/ac2a1b7/scheduler",
  "objectRef": {
    "resource": "events",
    "namespace": "kube-system",
    "apiGroup": "events.k8s.io",
    "apiVersion": "v1beta1"
  },
  "requestReceivedTimestamp": "2019-11-17T00:33:56.980210Z",
  "stageTimestamp": "2019-11-17T00:33:56.980210Z"
}
```

Audit Record for Mutating Webhooks

- Annotations
  - Key-value pairs
    - “mutation.webhook.admission.k8s.io/round_{}_index_{}”
    - “patch.webhook.admission.k8s.io/round_{}_index_{}”

Audit Record for Mutating Webhooks

- Annotations
  - Key-value pairs
    - “mutation.webhook.admission.k8s.io/round_{}_index_{}”
    - “patch.webhook.admission.k8s.io/round_{}_index_{}”

```
# the audit event recorded
{
    "kind": "Event",
    "apiVersion": "audit.k8s.io/v1",
    "annotations": {
        "mutation.webhook.admission.k8s.io/round_1_index_2": "{\"configuration\":\"my-mutating-webhook-
        # other annotations
        ...
    }
    # other fields
    ...
}
```

Patch occurrence

- **Key:** mutation.webhook.admission.k8s.io/round_{}_index_{}”
  - E.g. round_0_index_1
  - Recorded at **Metadata** audit level or higher
- **Value**:

```
# the annotation value deserialized
{
    "configuration": "my-mutating-webhook-configuration.example.com",
    "webhook": "my-webhook.example.com",
    "mutated": false
}
```

Patch mutation

- **Key:** patch.webhook.admission.k8s.io/round_{}_index_{}”
  - E.g. round_1_index_3
  - Recorded at **Request** audit level or higher
- **Value**:

```
# the annotation value deserialized
{
    "configuration": "my-other-mutating-webhook-configuration.example.com",
    "webhook": "my-webhook-always-mutate.example.com",
    "patchType": "JSONPatch",
    "patch": [
        {
            "op": "add",
            "path": "/data/mutation-stage",
            "value": "yes"
        }
```

```
  "requestReceivedTimestamp": "2019-11-17T00:47:07.511785Z",
  "stageTimestamp": "2019-11-17T00:47:07.527925Z",
  "annotations": {
    "authorization.k8s.io/decision": "allow",
    "authorization.k8s.io/reason": "",
    "mutation.webhook.admission.k8s.io/round_0_index_0": "{\"configuration\":\"webhook-7130-0\",\"webhook\":\"adding-configmap-data-stag
e-1.k8s.io\",\"mutated\":true}",
    "mutation.webhook.admission.k8s.io/round_0_index_10": "{\"configuration\":\"webhook-7130-5\",\"webhook\":\"adding-configmap-data-sta
ge-1.k8s.io\",\"mutated\":false}",
    "mutation.webhook.admission.k8s.io/round_0_index_12": "{\"configuration\":\"webhook-7130-6\",\"webhook\":\"adding-configmap-data-sta
ge-1.k8s.io\",\"mutated\":false}",
    "mutation.webhook.admission.k8s.io/round_0_index_14": "{\"configuration\":\"webhook-7130-7\",\"webhook\":\"adding-configmap-data-sta
ge-1.k8s.io\",\"mutated\":false}",
    "mutation.webhook.admission.k8s.io/round_0_index_16": "{\"configuration\":\"webhook-7130-8\",\"webhook\":\"adding-configmap-data-sta
ge-1.k8s.io\",\"mutated\":false}",
    "mutation.webhook.admission.k8s.io/round_0_index_18": "{\"configuration\":\"webhook-7130-9\",\"webhook\":\"adding-configmap-data-sta
ge-1.k8s.io\",\"mutated\":false}",
    "mutation.webhook.admission.k8s.io/round_0_index_2": "{\"configuration\":\"webhook-7130-1\",\"webhook\":\"adding-configmap-data-stag
e-1.k8s.io\",\"mutated\":true}",
    "mutation.webhook.admission.k8s.io/round_0_index_4": "{\"configuration\":\"webhook-7130-2\",\"webhook\":\"adding-configmap-data-stag
e-1.k8s.io\",\"mutated\":false}",
    "mutation.webhook.admission.k8s.io/round_0_index_6": "{\"configuration\":\"webhook-7130-3\",\"webhook\":\"adding-configmap-data-stag
e-1.k8s.io\",\"mutated\":false}",
    "mutation.webhook.admission.k8s.io/round_0_index_8": "{\"configuration\":\"webhook-7130-4\",\"webhook\":\"adding-configmap-data-stag
e-1.k8s.io\",\"mutated\":false}",
    "patch.webhook.admission.k8s.io/round_0_index_0": "{\"configuration\":\"webhook-7130-0\",\"webhook\":\"adding-configmap-data-stage-1
.k8s.io\",\"patch\":[{\"op\":\"add\",\"path\":\"/data/mutation-stage-1\",\"value\":\"yes\"}],\"patchType\":\"JSONPatch\"}",
    "patch.webhook.admission.k8s.io/round_0_index_10": "{\"configuration\":\"webhook-7130-5\",\"webhook\":\"adding-configmap-data-stage-
1.k8s.io\",\"patch\":[{\"op\":\"add\",\"path\":\"/data/mutation-stage-2\",\"value\":\"yes\"}],\"patchType\":\"JSONPatch\"}",
    "patch.webhook.admission.k8s.io/round_0_index_12": "{\"configuration\":\"webhook-7130-6\",\"webhook\":\"adding-configmap-data-stage-
1.k8s.io\",\"patch\":[{\"op\":\"add\",\"path\":\"/data/mutation-stage-2\",\"value\":\"yes\"}],\"patchType\":\"JSONPatch\"}",
    "patch.webhook.admission.k8s.io/round_0_index_14": "{\"configuration\":\"webhook-7130-7\",\"webhook\":\"adding-configmap-data-stage-
1.k8s.io\",\"patch\":[{\"op\":\"add\",\"path\":\"/data/mutation-stage-2\",\"value\":\"yes\"}],\"patchType\":\"JSONPatch\"}",
    "patch.webhook.admission.k8s.io/round_0_index_16": "{\"configuration\":\"webhook-7130-8\",\"webhook\":\"adding-configmap-data-stage-
1.k8s.io\",\"patch\":[{\"op\":\"add\",\"path\":\"/data/mutation-stage-2\",\"value\":\"yes\"}],\"patchType\":\"JSONPatch\"}",
    "patch.webhook.admission.k8s.io/round_0_index_18": "{\"configuration\":\"webhook-7130-9\",\"webhook\":\"adding-configmap-data-stage-
1.k8s.io\",\"patch\":[{\"op\":\"add\",\"path\":\"/data/mutation-stage-2\",\"value\":\"yes\"}],\"patchType\":\"JSONPatch\"}",
```

Debug the webhook backend

- Have good logging for the backend
  - What AdmissionReview it got
  - What AdmissionResponse it responded

Demo

```
$ kubectl apply -f my-pod.yaml
The Pod "my-pod" is invalid: spec.containers[2].name: Duplicate value: "my-sidecar"

# A new webhook "the-other-webhook" suddenly breaks my "inject-sidecar" webhook
```

```
$ kubectl apply -f my-pod.yaml
The Pod "my-pod" is invalid: spec.containers[2].name: Duplicate value: "my-sidecar"

# A new webhook "the-other-webhook" suddenly breaks my "inject-sidecar" webhook
$ kubectl get mutatingwebhookconfigurations
NAME                  CREATED AT
inject-sidecar        2019-11-12T00:10:46Z
the-other-webhook     2019-11-19T06:13:36Z
```

```
$ kubectl apply -f my-pod.yaml
The Pod "my-pod" is invalid: spec.containers[2].name: Duplicate value: "my-sidecar"

# A new webhook "the-other-webhook" suddenly breaks my "inject-sidecar" webhook
$ kubectl get mutatingwebhookconfigurations
NAME                  CREATED AT
inject-sidecar        2019-11-12T00:10:46Z
the-other-webhook     2019-11-19T06:13:36Z

# I configured the audit level to be Request for my local cluster
$ grep "cat <<EOF > /tmp/kube-audit-policy-file" -A 6 hack/local-up-cluster.sh
      cat <<EOF > /tmp/kube-audit-policy-file
# Log all requests at the Metadata level.
apiVersion: audit.k8s.io/v1
kind: Policy
rules:
- level: Request
EOF
```

```
$ kubectl apply -f my-pod.yaml
The Pod "my-pod" is invalid: spec.containers[2].name: Duplicate value: "my-sidecar"

# A new webhook "the-other-webhook" suddenly breaks my "inject-sidecar" webhook
$ kubectl get mutatingwebhookconfigurations
NAME                  CREATED AT
inject-sidecar        2019-11-12T00:10:46Z
the-other-webhook     2019-11-19T06:13:36Z

# I configured the audit level to be Request for my local cluster
$ grep "cat <<EOF > /tmp/kube-audit-policy-file" -A 6 hack/local-up-cluster.sh
      cat <<EOF > /tmp/kube-audit-policy-file
# Log all requests at the Metadata level.
apiVersion: audit.k8s.io/v1
kind: Policy
rules:
- level: Request
EOF

# retrieve the audit log for `kubectl apply -f my-pod.yaml`
$ grep "/api/v1/namespaces/default/pods" /tmp/kube-apiserver-audit.log | tail -n 1 | jq
```

```
piserver-audit.log | tail -n 1 | jq
{
  "kind": "Event",
  "apiVersion": "audit.k8s.io/v1",
  "level": "Request",
  "auditID": "90c24842-62d7-4cdd-8aee-db60ef59f08c",
  "stage": "ResponseComplete",
  "requestURI": "/api/v1/namespaces/default/pods",
  "verb": "create",
  "user": {
    "username": "system:admin",
    "groups": [
      "system:masters",
      "system:authenticated"
    ]
  },
  "sourceIPs": [
    "::1"
  ],
  "userAgent": "kubectl/v1.18.0 (linux/amd64) kubernetes/ac2a1b7",
  "objectRef": {
    "resource": "pods",
    "namespace": "default",
    "name": "my-pod",
    "apiVersion": "v1"
  },
```


"responseStatus": {
"metadata": {},
"status": "Failure",
"reason": "Invalid",
"code": 422
},

```
  "requestObject": {
    "kind": "Pod",
    "apiVersion": "v1",
    "metadata": {
      "name": "my-pod",
      "namespace": "default",
      "creationTimestamp": null,
      "annotations": {
        "kubectl.kubernetes.io/last-applied-configuration": "{\"apiVersion\":\"v1\
:{},\"name\":\"my-pod\",\"namespace\":\"default\"},\"spec\":{\"containers\":[{\"co
ihw/demo-hook-sidecar:latest\",\"name\":\"my-container\"}]}}\n"
      }
    },
    "spec": {
      "containers": [
        {
          "name": "my-container",
          "image": "roycaihw/demo-hook-sidecar:latest",
          "command": [
            "echo",
            "SUCCESS"
          ],
          "resources": {},
          "terminationMessagePath": "/dev/termination-log",
          "terminationMessagePolicy": "File",
          "imagePullPolicy": "Always"
        }
      ],
      "restartPolicy": "Always",
```

```
  },
  "requestReceivedTimestamp": "2019-11-19T06:13:56.681766Z",
  "stageTimestamp": "2019-11-19T06:13:56.702766Z",
  "annotations": {
    "authorization.k8s.io/decision": "allow",
    "authorization.k8s.io/reason": "",
    "mutation.webhook.admission.k8s.io/round_0_index_0": "{\"configuration\":\"inject-sidecar-webhook\",\"webhook\":\"inject-webhook.webhook-demo-2.svc\",\"mutated\":true}",
    "mutation.webhook.admission.k8s.io/round_0_index_1": "{\"configuration\":\"the-other-webhook\",\"webhook\":\"the-other-webhook.webhook-demo.svc\",\"mutated\":true}",
    "mutation.webhook.admission.k8s.io/round_1_index_0": "{\"configuration\":\"inject-sidecar-webhook\",\"webhook\":\"inject-webhook.webhook-demo-2.svc\",\"mutated\":true}",
    "patch.webhook.admission.k8s.io/round_0_index_0": "{\"configuration\":\"inject-sidecar-webhook\",\"webhook\":\"inject-webhook.webhook-demo-2.svc\",\"patch\":[{\"op\":\"add\",\"path\":\"/spec/containers/-\",\"value\":{\"name\":\"my-sidecar\",\"image\":\"my-sidecar-image:latest\",\"command\":[\"echo\",\"SIDECAR\"],\"resources\":{}}}],\"patchType\":\"JSONPatch\"}",
    "patch.webhook.admission.k8s.io/round_0_index_1": "{\"configuration\":\"the-other-webhook\",\"webhook\":\"the-other-webhook.webhook-demo.svc\",\"patch\":[{\"op\":\"add\",\"path\":\"/spec/securityContext/runAsNonRoot\",\"value\":true},{\"op\":\"add\",\"path\":\"/spec/securityContext/runAsUser\",\"value\":1234}],\"patchType\":\"JSONPatch\"}",
    "patch.webhook.admission.k8s.io/round_1_index_0": "{\"configuration\":\"inject-sidecar-webhook\",\"webhook\":\"inject-webhook.webhook-demo-2.svc\",\"patch\":[{\"op\":\"add\",\"path\":\"/spec/containers/-\",\"value\":{\"name\":\"my-sidecar\",\"image\":\"my-sidecar-image:latest\",\"command\":[\"echo\",\"SIDECAR\"],\"resources\":{}}}],\"patchType\":\"JSONPatch\"}"
  }
}
```

Key Takeaways

- Configure your admission webhooks following the best practices
- Use metrics and audit logging to monitor and debug your webhooks

Thank you!
Q&A

Reference

- https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/
- https://kubernetes.io/docs/reference/access-authn-authz/extensible-admission-controllers
- https://kubernetes.io/blog/2019/03/21/a-guide-to-kubernetes-admission-controllers/
- https://kubernetes.io/docs/tasks/debug-application-cluster/audit/


KubeCon
CloudNativeCon
North America 2019