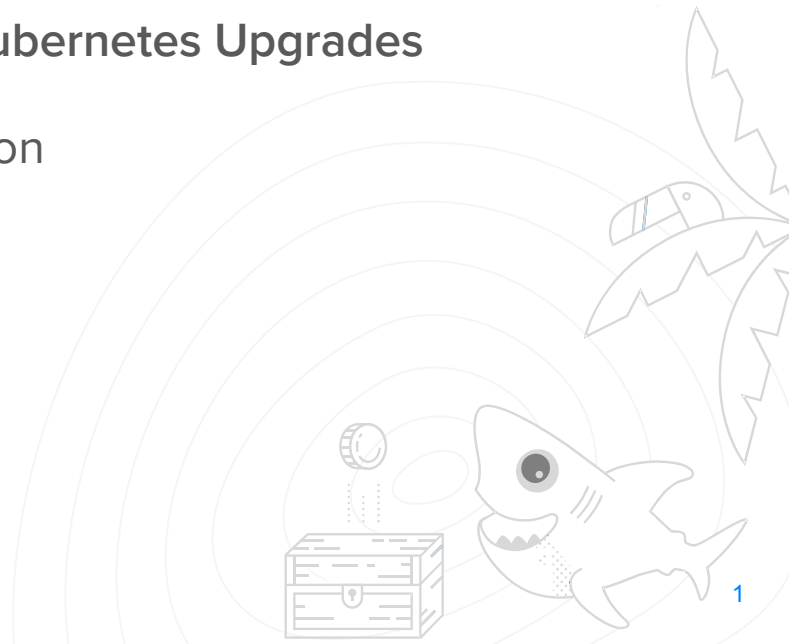




# 20,000 Upgrades Later

Lessons From a Year of Managed Kubernetes Upgrades

Adam Wolfe Gordon  
DigitalOcean





# This Talk Started One(ish) Year Ago...

Me, in Barcelona



DO, in Barcelona





# Generally Available?

With the help of our customers, we've been working hard on enhancements to our Kubernetes service. Most notably, we're pleased to introduce a free, integrated monitoring service that automatically provides insights and alerts for your clusters. In addition, DigitalOcean Kubernetes now supports the [latest Kubernetes release, 1.14](#), which introduced 31 enhancements to the container orchestration platform. Now you can also schedule [automatic patch version upgrades](#).

1.14 to 1.14 **UPGRADES!**

Finally, because the service is now Generally Available, you can now spin up clusters in each city where we have a data center: New York, San Francisco, Amsterdam, London, Frankfurt, Bangalore, and Toronto.

## AVAILABLE UPGRADES

This cluster is up to date: 1.17.5-do.0

Upgrades will not be applied automatically. [Enable automatic upgrades](#)



[do.co/doks](https://do.co/doks)



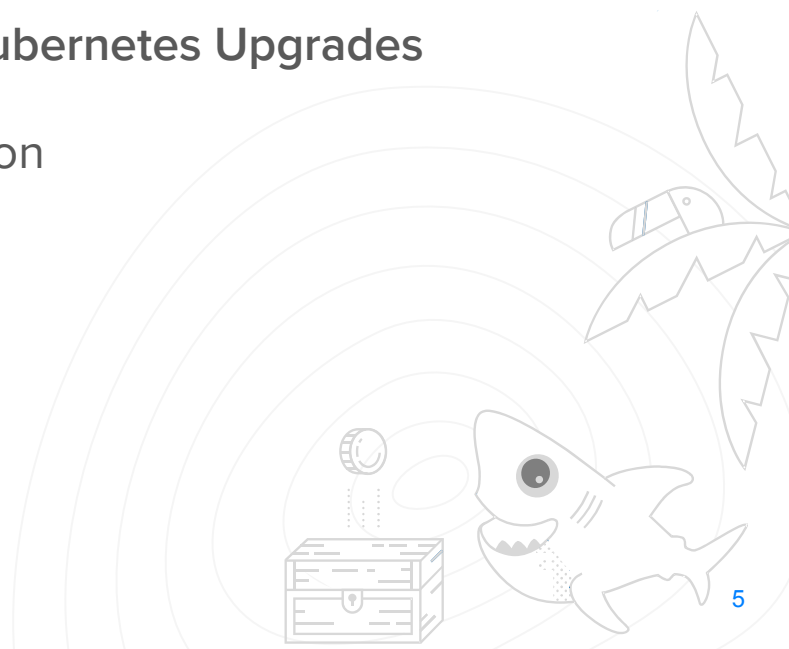
[@maybeawg](https://twitter.com/maybeawg)



# ~~20,000~~ <sup>35,000</sup> Upgrades Later

Lessons From a Year of Managed Kubernetes Upgrades

Adam Wolfe Gordon  
DigitalOcean





## Disclaimers!

- Lessons from **our** upgrade process.
  - You might upgrade differently!
- Upgrades of **our** customers' clusters.
  - Your workloads might be different!





# How to Upgrade Kubernetes

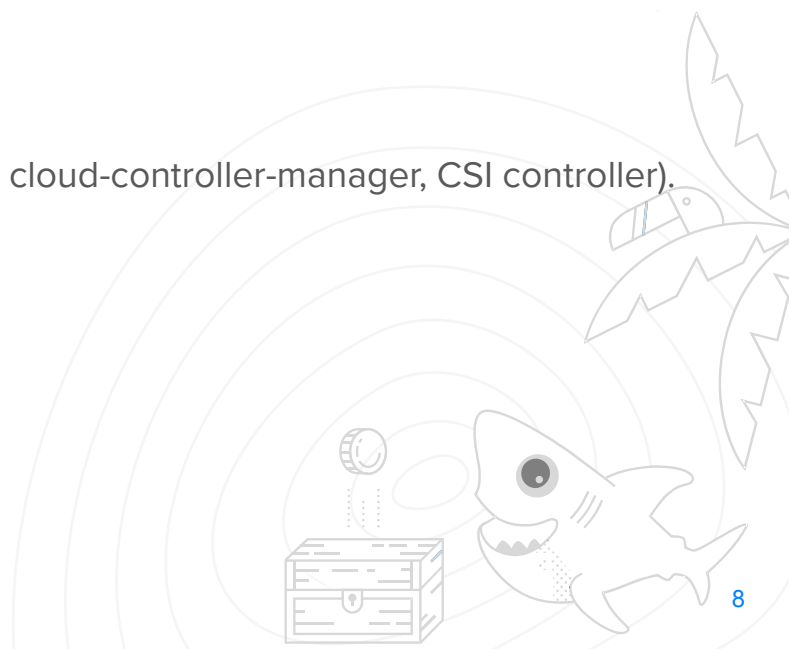
1. Upgrade the control plane.
2. Upgrade the worker nodes.
3. ???
4. Profit!





# How to Upgrade Kubernetes

1. Upgrade the control plane.
  - a. Update any resources that aren't supported in the target version.
  - b. Upgrade etcd (if needed).
  - c. Upgrade kube-apiserver.
  - d. Upgrade kube-controller-manager.
  - e. Upgrade kube-scheduler.
  - f. Upgrade your CNI plugin (if needed).
  - g. Upgrade provider-specific components (e.g. cloud-controller-manager, CSI controller).
  - h. Upgrade kubelet and kubectl.
2. Upgrade the worker nodes.
  - a. Cordon and drain a worker node.
  - b. Update kubelet configuration (if needed).
  - c. Upgrade the kubelet.
  - d. Uncordon the node.
  - e. Repeat for each node in the cluster.





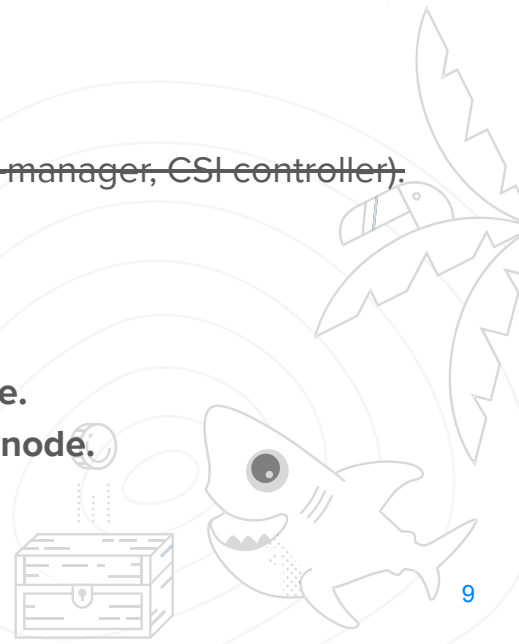
# Shortcut: Upgrade via Node Replacement

## 1. Upgrade the control plane.

- a. Update any resources that aren't supported in the target version.
  - ~~b. Upgrade etcd (if needed).~~
  - ~~c. Upgrade kube-apiserver.~~
  - ~~d. Upgrade kube-controller-manager.~~
  - ~~e. Upgrade kube-scheduler.~~
  - ~~f. Upgrade your CNI plugin (if needed).~~
  - ~~g. Upgrade provider specific components (e.g. cloud-controller-manager, CSI controller).~~
  - ~~h. Upgrade kubelet and kubectl.~~
- b. Destroy the original control plane node.**  
**c. Provision a new control plane node.**

## 2. Upgrade the worker nodes.

- a. Cordon and drain a worker node.
  - ~~b. Update kubelet configuration (if needed).~~
  - ~~c. Upgrade the kubelet.~~
  - ~~d. Uncordon the node.~~
  - e. Repeat for each node in the cluster.
- b. Destroy the node.**  
**c. Provision a new node.**

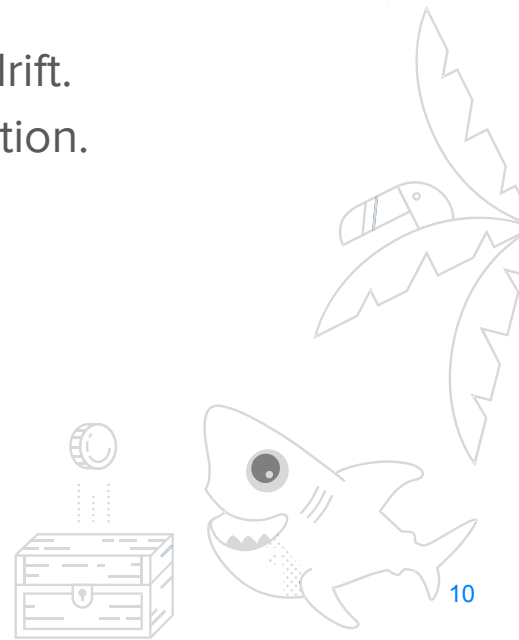






## Advantages of Node Replacement

- Clean slate - no chance for configuration drift.
- Fewer steps to manage - good for automation.
- Same process works for all release types.
  - (Mostly)





Things We Got Right

# Upgrades via Node Replacement



[do.co/doks](https://do.co/doks)



[@maybeawg](https://twitter.com/maybeawg)



## Problems Ch-ch-changes

- Custom node configuration is reset.
- Node names change.
- Node IPs change.
- Node labels and taints lost.





## Lessons for Operators Managing Change

- Re-use node names and IPs if possible.
- Retain labels or provide a good alternative.
- Retain taints or provide a good alternative.
- Provide simple ingress/load balancing.





## Lessons for Developers Tolerating Change

- Use Kubernetes to do node customization.
  - DaemonSets
  - Init containers
- Don't use node names for scheduling.
- Use provider-supported label/taint settings.
- Use provider-supported load balancing.





Things We Got Wrong

**Break Before Make**



[do.co/doks](https://do.co/doks)



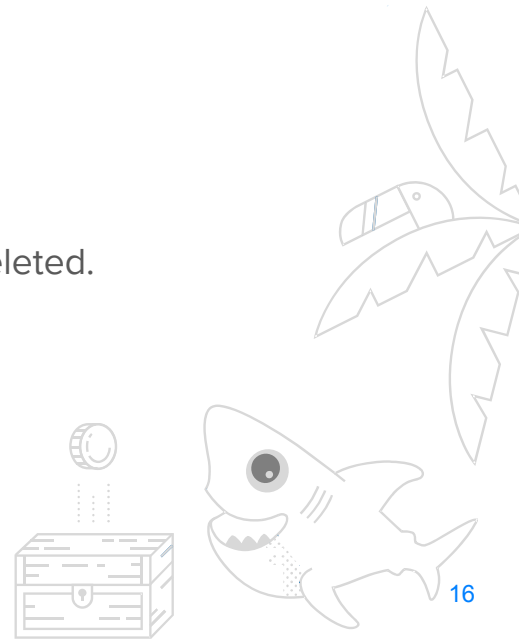
[@maybeawg](https://twitter.com/maybeawg)



## Problems

### Drain to Nowhere

- Insufficient capacity to drain nodes.
- Downtime in single-node clusters.
- Extra churn for workloads.
  - Might be drained to a node that's about to be deleted.

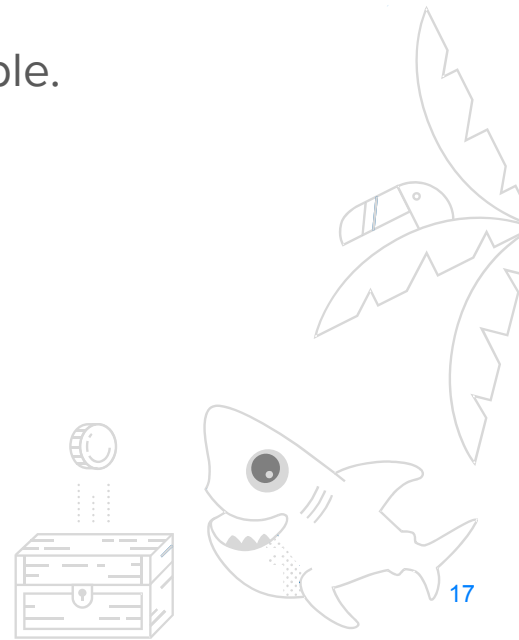




## Lessons for Operators

### Drainage Capacity

- Add nodes before deleting nodes if possible.
- Consider reserving capacity.







## Lessons for Developers Expect to be Drained

- Leave capacity for a node to be drained.





Things We Got Wrong

# Replacing Nodes One by One



[do.co/doks](https://do.co/doks)



[@maybeawg](https://twitter.com/maybeawg)



## Problems

### Ants Go Marching

- Replacing nodes one-by-one is slow.
- Workloads can get stuck draining
  - Making replacement even slower.
- Upgrades need to be expedient.





## Lessons for Operators

### Rapid Replacement

- Drain and replace multiple nodes at once.
  - This usually requires make-before-break.
- Set reasonable drain timeouts.





# Lessons for Developers

## Unclog Your Drains

- Make sure your workloads can be evicted.
  - Safely: Use PodDisruptionBudgets.
  - Quickly: Respond to signals.
- Test this!





Things We Got Wrong (but felt so right)

# Minor Version Upgrades are Easy



[do.co/doks](https://do.co/doks)

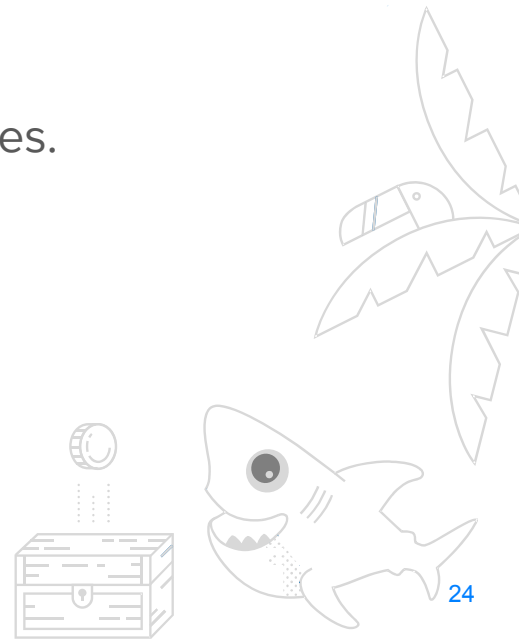


[@maybeawg](https://twitter.com/maybeawg)



## Lessons for Operators Don't Worry, Be Happy

- Minor version upgrades aren't that scary.
- Try to use the same process for all upgrades.





Things We Got Right

# Disabling Alpha Features



[do.co/doks](https://do.co/doks)



[@maybeawg](https://twitter.com/maybeawg)





## Lessons for Operators

### Wait for Beta

- Alpha features are disabled by default.
- Alpha features are likely to change/break.
- Beta features are less likely to change.
- Consider the upgrade tradeoff.





## Lessons for Developers

### Alpha as a Last Resort

- Avoid using alpha features if possible.
- Read release notes before upgrading.





Common Problems

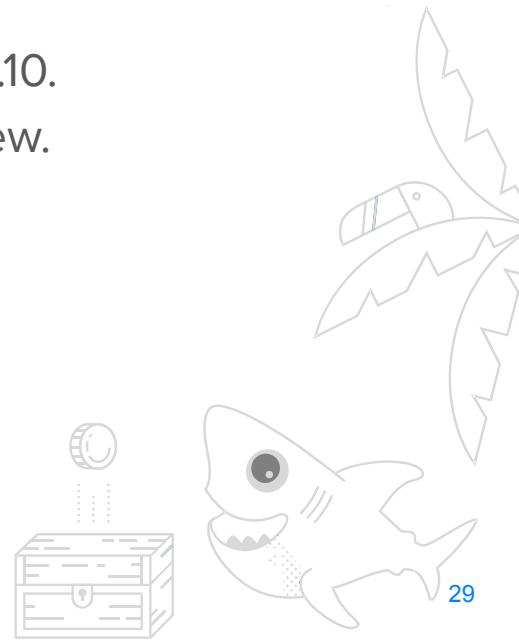
# Container Storage Interface (CSI)





## CSI Problems Beta

- CSI was promoted to beta in Kubernetes 1.10.
- Supporting components were relatively new.
- CSI drivers were relatively new.
- Out-of-sync state.
- Far fewer problems in recent releases.





## CSI Problems Driver Names

- In early CSI specs, `com.example.csi`.
- In later CSI specs, `csi.example.com`.
- The name is immutable in Kubernetes!
- Solution: detect and persist old naming.

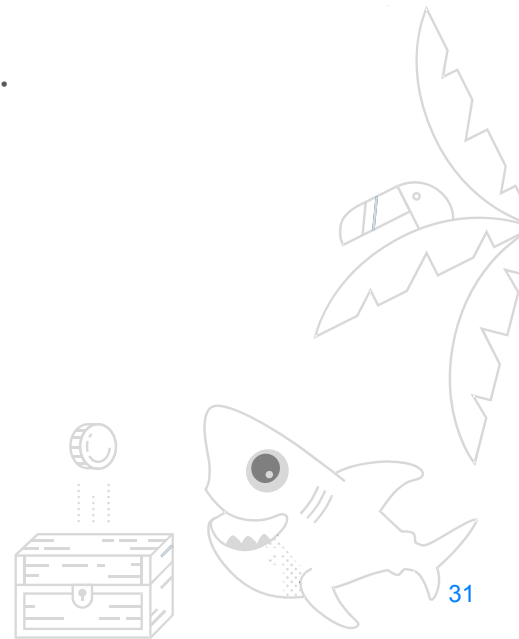




## Lessons

### Beware the CSI

- If you're using CSI, carefully test upgrades.
- Watch for workloads that get stuck.
- Use Kubernetes 1.14+ if possible.





Common Problems

# Admission Control Webhooks



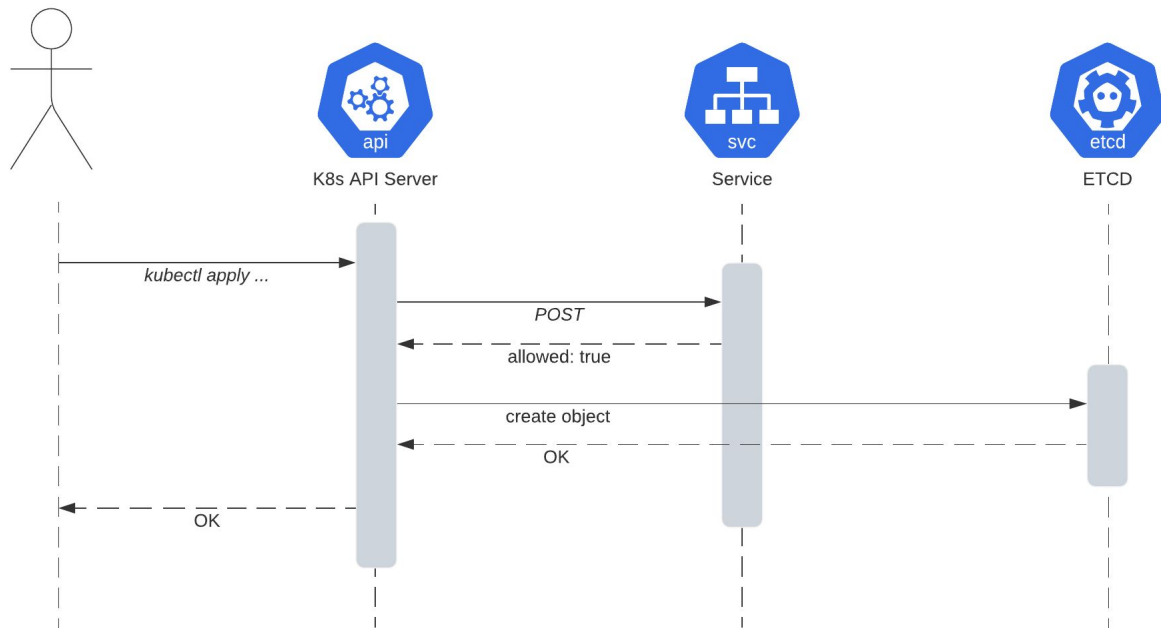
[do.co/doks](https://do.co/doks)



[@maybeawg](https://twitter.com/maybeawg)



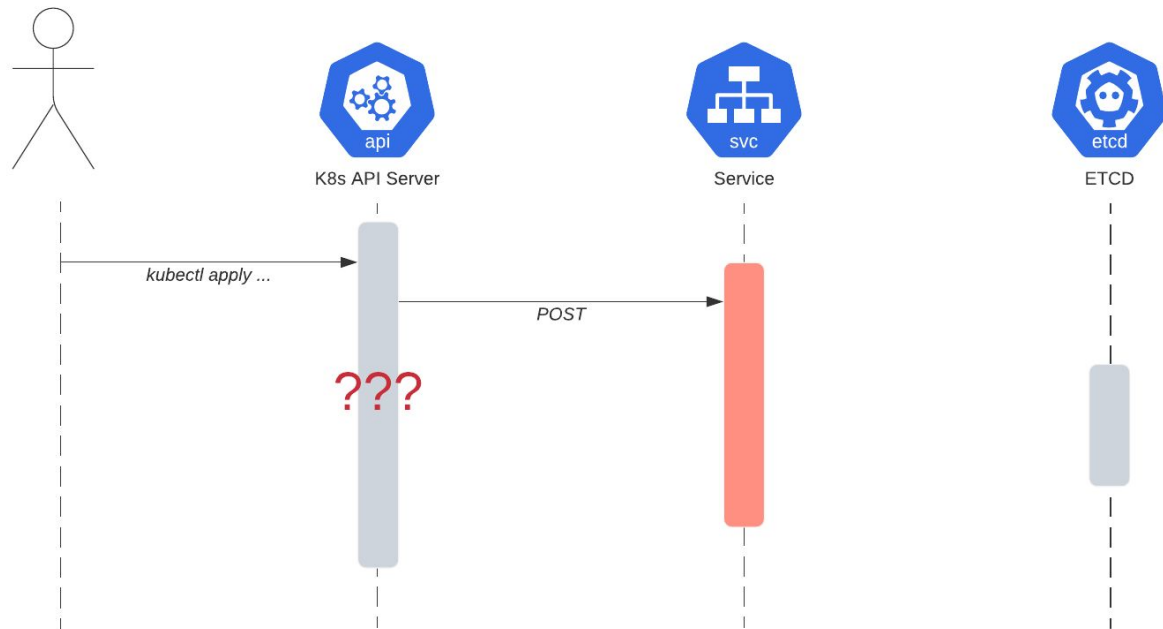
# Admission Control Webhooks Overview







# Admission Control Webhooks Overview





# Admission Control Webhooks Overview

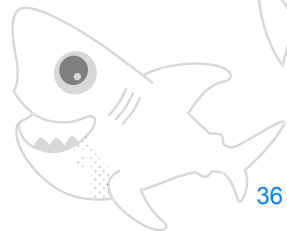
```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  name: webhook.example.com
webhooks:
- name: webhook.example.com
  rules:
  - apiGroups:  [""]
    apiVersions: ["v1"]
    operations:  ["CREATE"]
    resources:   ["pods"]
    scope:       "Namespaced"
  clientConfig:
    service:
      namespace: "webhook-namespace"
      name: "webhook-service"
  admissionReviewVersions: ["v1", "v1beta1"]
  sideEffects: None
  timeoutSeconds: 30
  failurePolicy: Ignore
```





## Admission Control Webhooks Trouble for Upgrades

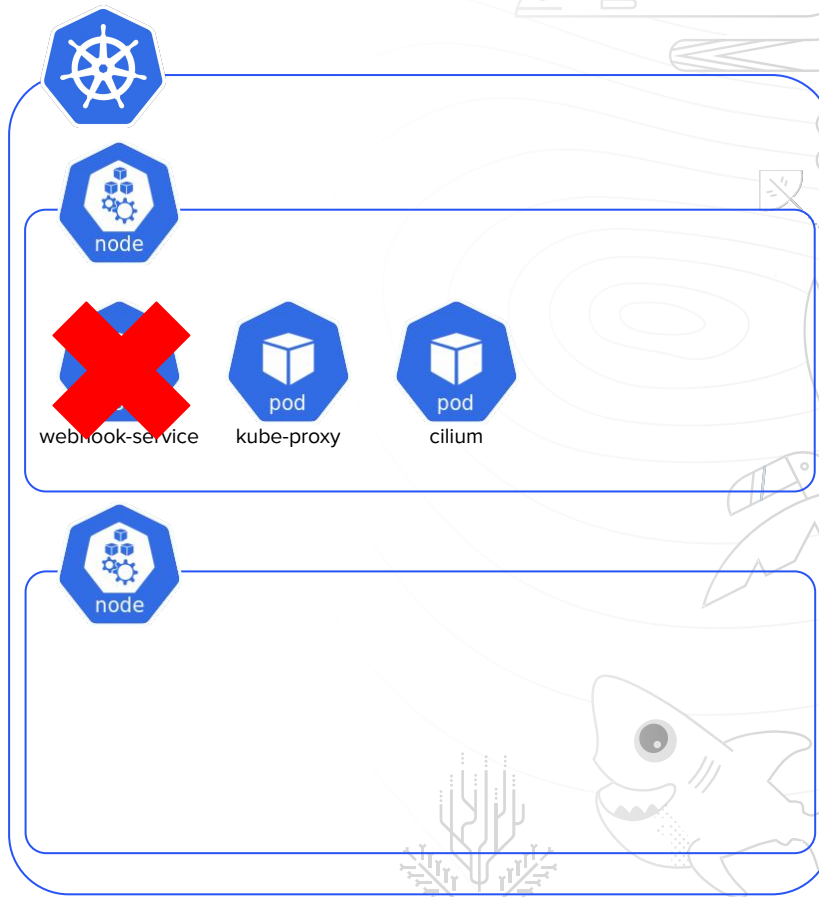
- Upgrades update system components.
- Some of these components run as workloads.
  - Usually in the kube-system namespace.
- Webhooks can prevent these updates.
- Webhooks can also affect their own services!





# Admission Control Webhooks: Problems

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  name: webhook.example.com
webhooks:
- name: webhook.example.com
  rules:
  - apiGroups:  [""]
    apiVersions: ["v1"]
    operations:  ["CREATE"]
    resources:   ["pods"]
    scope:       "Namespaced"
  clientConfig:
    service:
      namespace: "webhook-namespace"
      name: "webhook-service"
  admissionReviewVersions: ["v1", "v1beta1"]
  sideEffects: None
  timeoutSeconds: 30
  failurePolicy: Fail
```





# Admission Control Webhooks: Solutions

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  name: webhook.example.com
webhooks:
- name: webhook.example.com
  rules:
  - apiGroups:  [""]
    apiVersions: ["v1"]
    operations:  ["CREATE"]
    resources:   ["pods"]
    scope:       "Namespaced"
  clientConfig:
    service:
      namespace: "webhook-namespace"
      name: "webhook-service"
  admissionReviewVersions: ["v1", "v1beta1"]
  sideEffects: None
  timeoutSeconds: 30
  failurePolicy: Ignore
```



[do.co/doks](https://do.co/doks)



[@maybeawg](https://twitter.com/maybeawg)





# Admission Control Webhooks: Solutions

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  name: webhook.example.com
webhooks:
- name: webhook.example.com
  rules:
  - apiGroups:  [""]
    apiVersions: ["v1"]
    operations:  ["CREATE"]
    resources:   ["pods"]
    scope:       "Namespaced"
  clientConfig:
    service:
      namespace: "webhook-namespace"
      name: "webhook-service"
  admissionReviewVersions: ["v1", "v1beta1"]
  sideEffects: None
  timeoutSeconds: 5
  failurePolicy: Ignore
```



[do.co/doks](https://do.co/doks)



[@maybeawg](https://twitter.com/maybeawg)





# Admission Control Webhooks: Solutions

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  name: webhook.example.com
webhooks:
- name: webhook.example.com
  namespaceSelector: ←
    matchExpressions:
    - key: system-critical
      operator: DoesNotExist
...
clientConfig:
  service:
    namespace: "webhook-namespace"
    name: "webhook-service"
  admissionReviewVersions: ["v1", "v1beta1"]
  sideEffects: None
  timeoutSeconds: 5
  failurePolicy: Fail
```

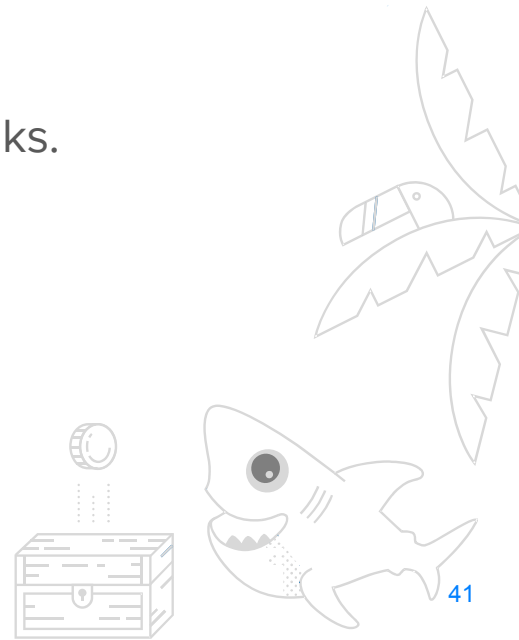




## Lessons for Operators

### Webhooks are Trouble

- Check webhook config before upgrading.
- Consider a mutating webhook for webhooks.



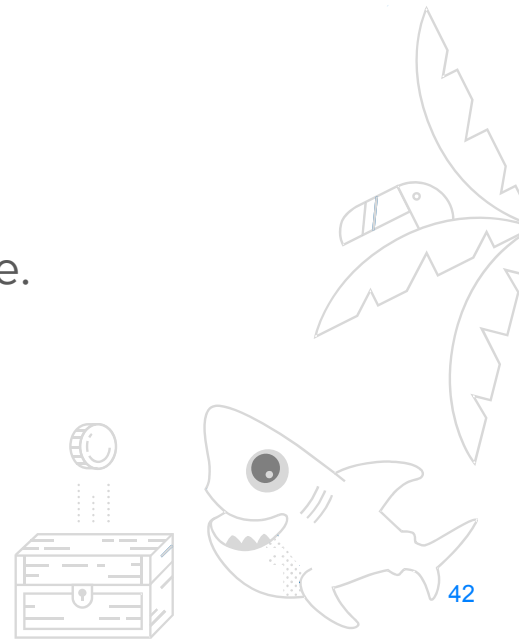




## Lessons for Developers

### Be Careful with Webhooks

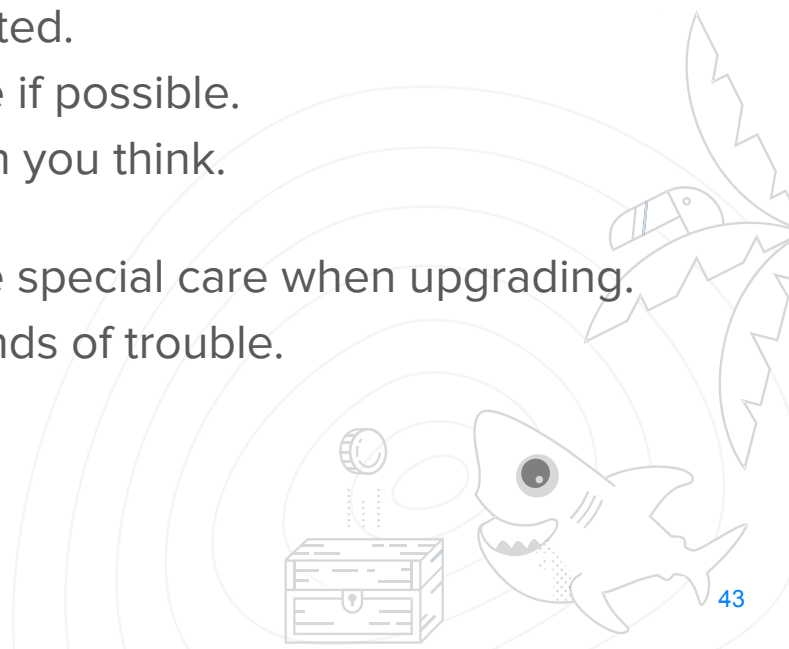
- Set failurePolicy to Ignore if possible.
- Set timeout much lower than 30 seconds.
- Exclude kube-system.
- Exclude the webhook service's namespace.
  - Or run the webhook service outside the cluster.
- Exclude any other critical namespaces.





# Wrap Up

- Consider upgrading via node replacement.
  - Retain node names and IP addresses if you can.
  - Workloads should assume that nodes will go away.
  - Create new nodes before destroying old ones, if possible.
- Make sure your workloads can be evicted.
- Upgrade more than one node at a time if possible.
- Minor version upgrades are easier than you think.
  - Especially if you avoid alpha features.
- CSI is just now becoming mature - take special care when upgrading.
- Admission control webhooks are all kinds of trouble.
  - Check your targets.
  - Check your failure policies.
  - Check your timeouts.



# Questions?

Adam Wolfe Gordon  
awg@do.co

