KubeCon | CloudNativeCon

North America 2018

SIG-Auth Deep Dive

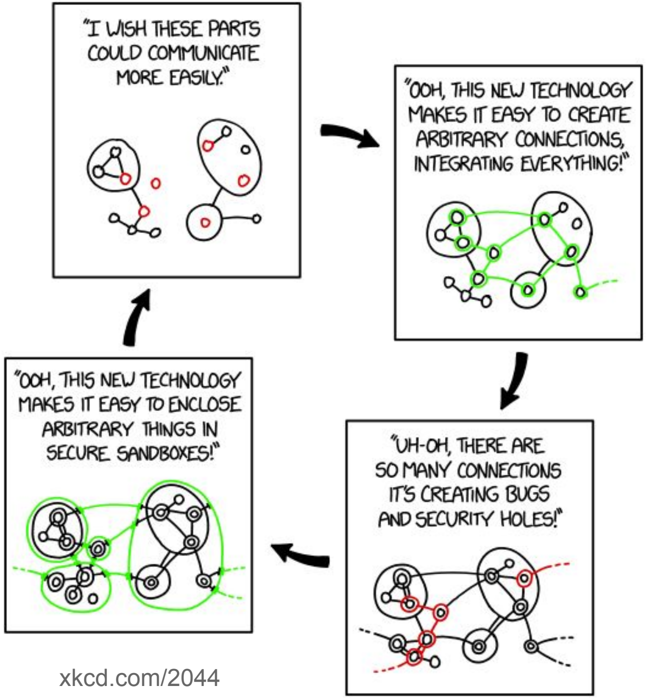
Tim Allclair, Mike Danese, Jordan Liggitt

Add-on Auth

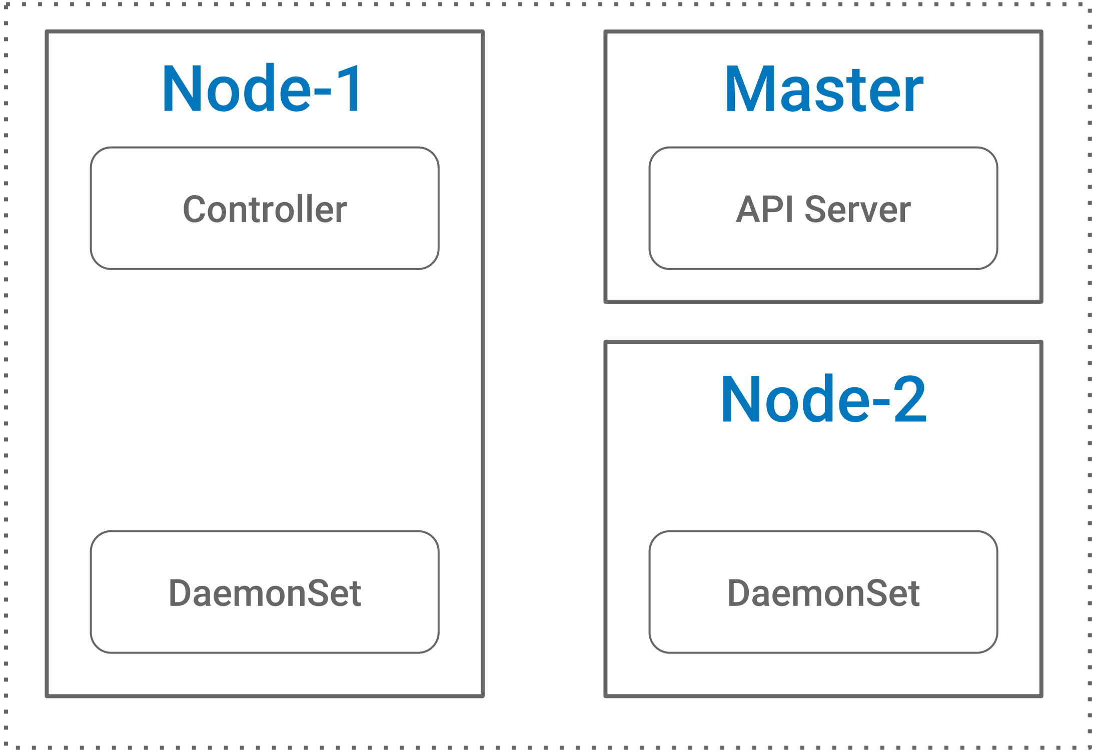
kubernetes/kubernetes/#62747

Examples

- Local volume provisioning
- Device plugins
- Device metrics
- CRI Streaming server


"I WISH THESE PARTS COULD COMMUNICATE MORE EASILY."
"OOH, THIS NEW TECHNOLOGY MAKES IT EASY TO CREATE ARBITRARY CONNECTIONS, INTEGRATING EVERYTHING!"
"UH-OH, THERE ARE SO MANY CONNECTIONS IT'S CREATING BUGS AND SECURITY HOLES!"
"OOH, THIS NEW TECHNOLOGY MAKES IT EASY TO ENCLOSE ARBITRARY THINGS IN SECURE SANDBOXES!"

xkcd.com/2044



Node-1

Controller

DaemonSet

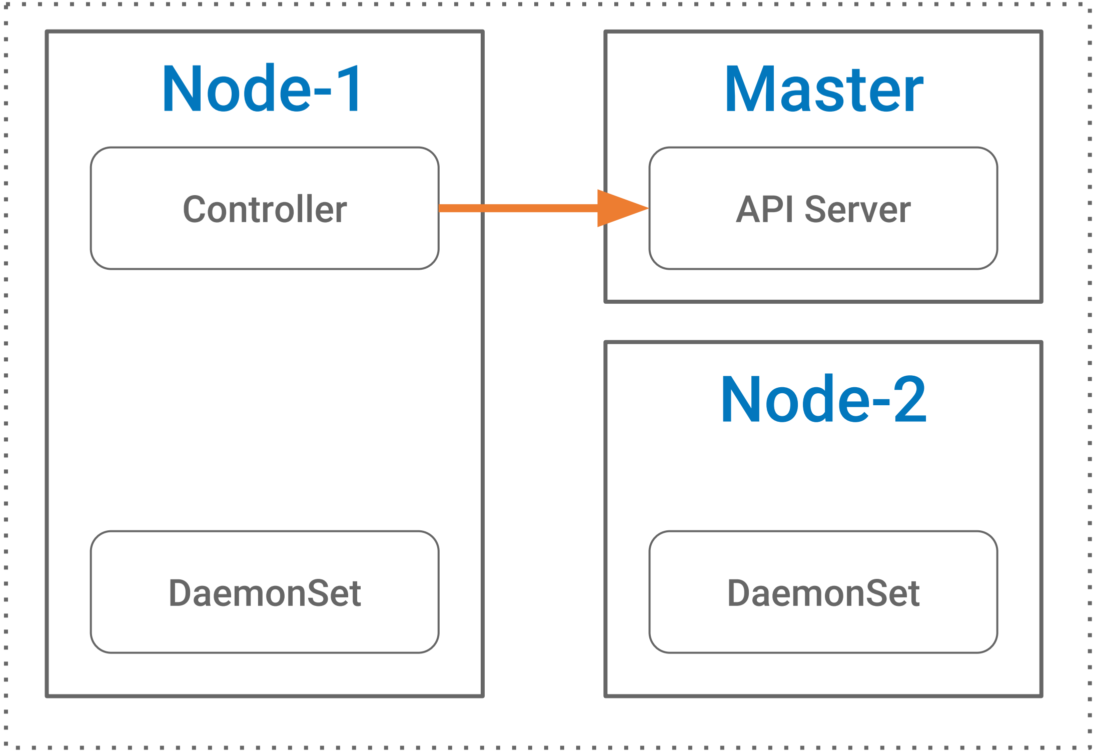
Master

API Server

Node-2

DaemonSet

SIG-Auth Deep Dive

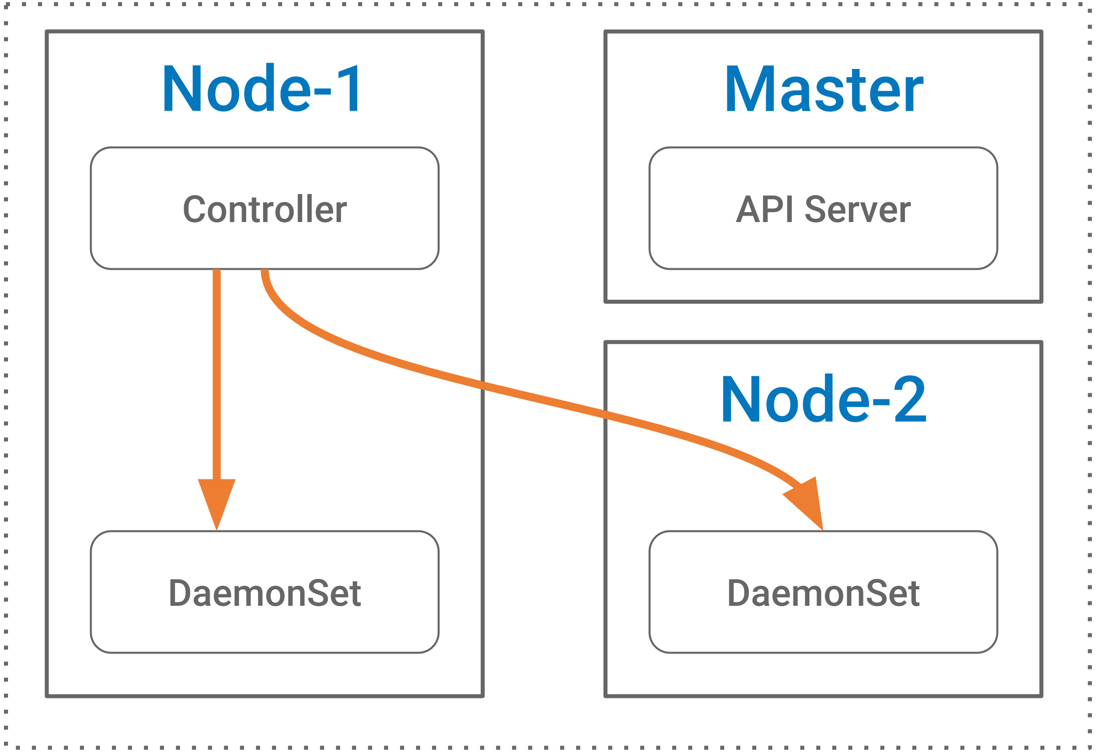


Server Auth'n
Cluster CA (automounted)

Client Auth'n
Service Accounts

Auth'z
Mostly RBAC

Done!

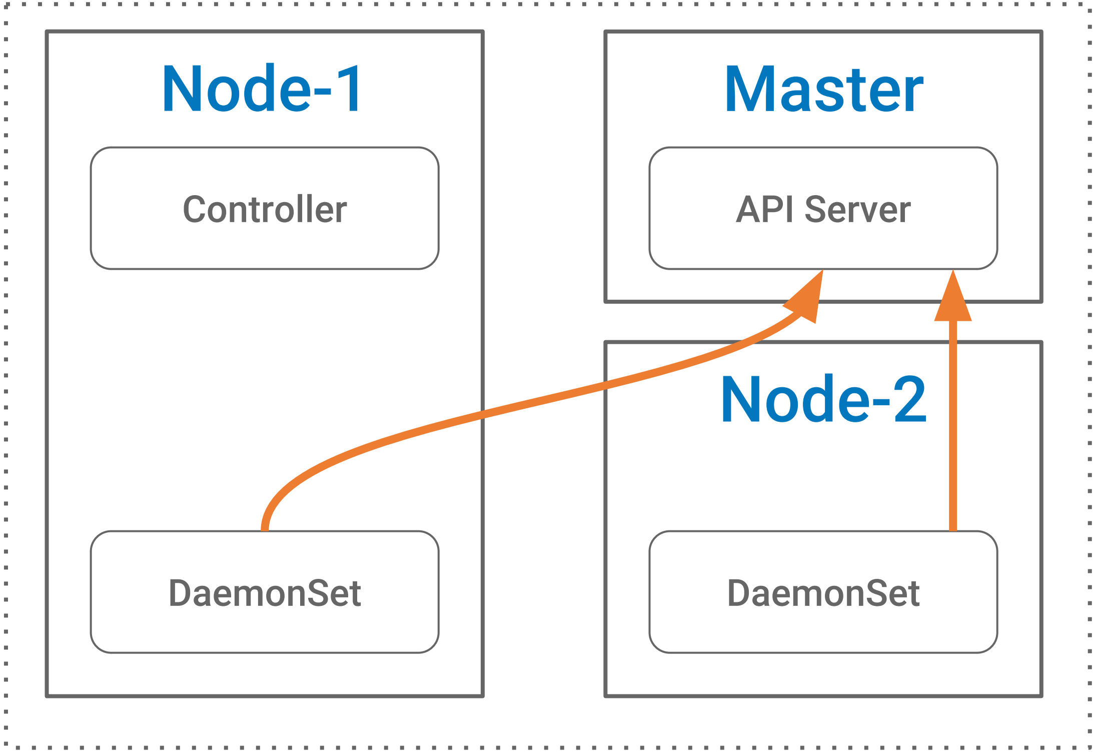


Server Auth'n
Auto-approved in-cluster per-pod certs?

Client Auth'n
Service Accounts + TokenReview

Auth'z
RBAC + SubjectAccessReview

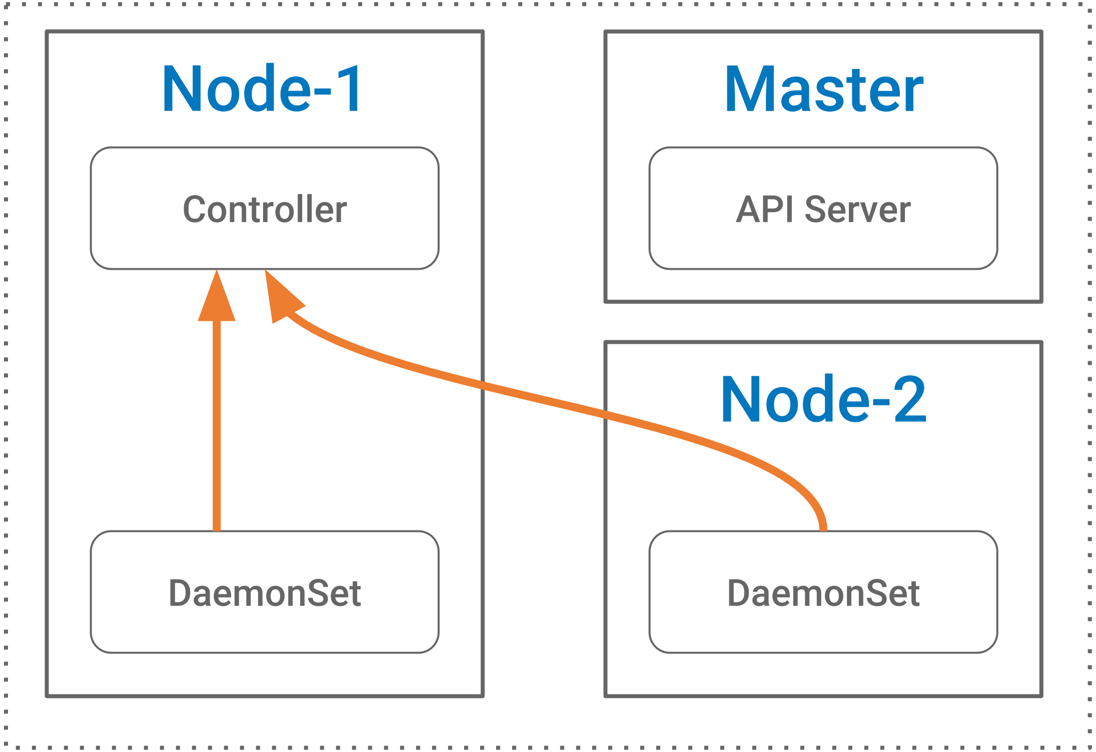
SIG-Auth Deep Dive



Server Auth'n
Cluster CA (automounted)

Client Auth'n
Enhanced! Service Accounts

Auth'z
Maybe:
NodeAuthorizer applied to DaemonSet pods



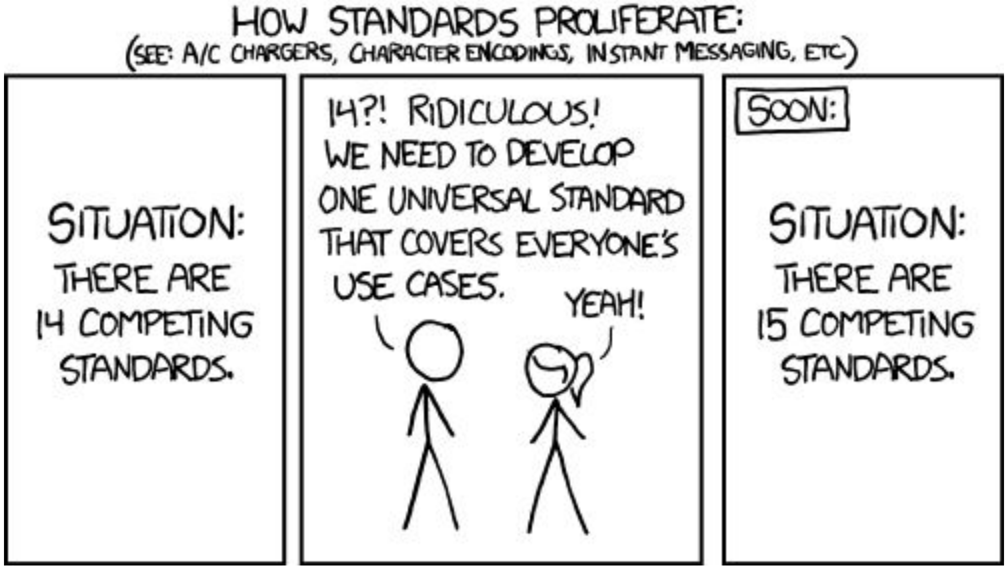
Server Auth'n
Auto-approved in-cluster service certs?

Client Auth'n
Enhanced! Service Accounts + TokenReview

Auth'z
Maybe:
NodeAuthorizer + SubjectAccesReview

Common approach for delegated pod admission & policy

kubernetes/kubernetes/#60001


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

https://xkcd.com/927/

PodSecurityPolicy - checked against the pods service account OR the creating user

NetworkPolicy - Namespaced; PodSelector determines the pods to apply to

ImagePolicy - delegates to an external webhook. Review includes image, annotations, and namespace

LimitRanger, ResourceQuota - namespace singleton

Toleration & NodeSelector restrictions - namespace singleton, defined on the namespace object

Apply policy at the namespace level

- most widely used approach right
- consistent with authorization (create granted at the namespace level)
- can't be applied more granularly in a namespace, and managing policy across namespaces needs to be handled.

Apply policy on the pod's service account

- Counter-intuitive
- Not really more secure than namespace level
- PodSecurityPolicy conflates 2 approaches and weakens security

Applied to requesting user - check policy when a create {ReplicaSet/Controller, Job, Deployment, DaemonSet, StatefulSet, ...} request is made

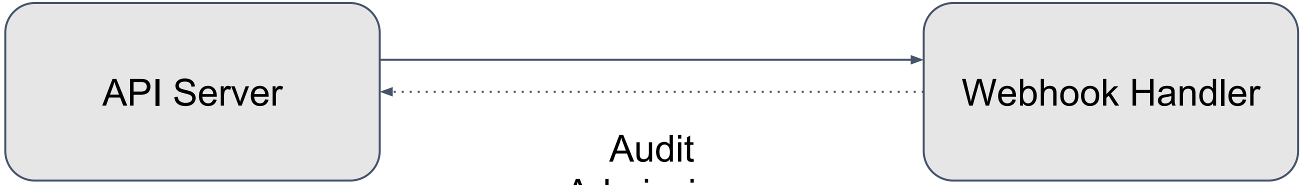
- How does it handle delegation to controllers?
- How does it handle CRDs and 3rd party controllers?
- What about mutating admission that acts on pods?
- Doesn't work for stateful policies (e.g. ResourceQuota)

Other areas of inconsistency

- Composability & conflict resolution
  (especially with mutation, or mixed allow & deny)
- Domain specific (scheduling policy) vs. resource specific (pod restriction)
- Default allow vs. default deny; whitelist vs. blacklist
- How to handle mutations
- Policy scope: namespaced or cluster-level

API server authentication to webhooks

kubernetes/kubernetes/#70815



Audit
Admission
Conversion
...

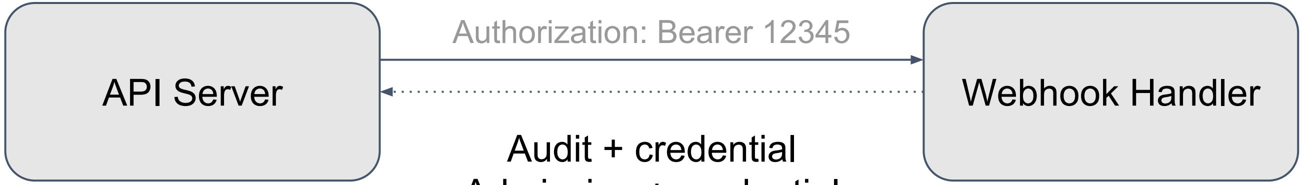
Why do API servers need to authenticate to webhooks at all?

- Webhooks accepting data need to know if the data should be trusted
  - Audit webhooks
  - Admission webhooks that take external actions

- Webhooks returning data need to know if the recipient is authorized
  - Admission webhooks that modify incoming objects

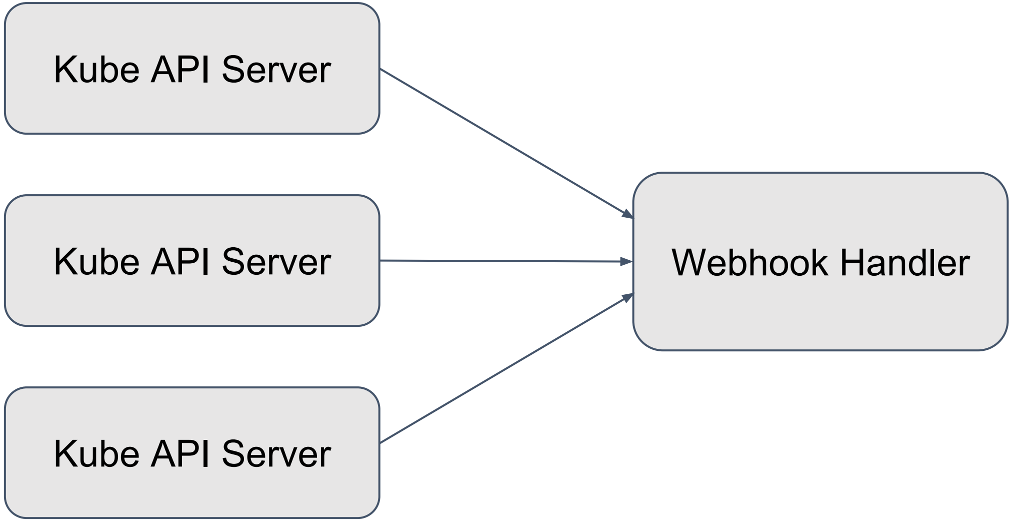
- Webhooks doing expensive work should only do it for the right callers

Simplest approach: add credentials to the webhook registration object

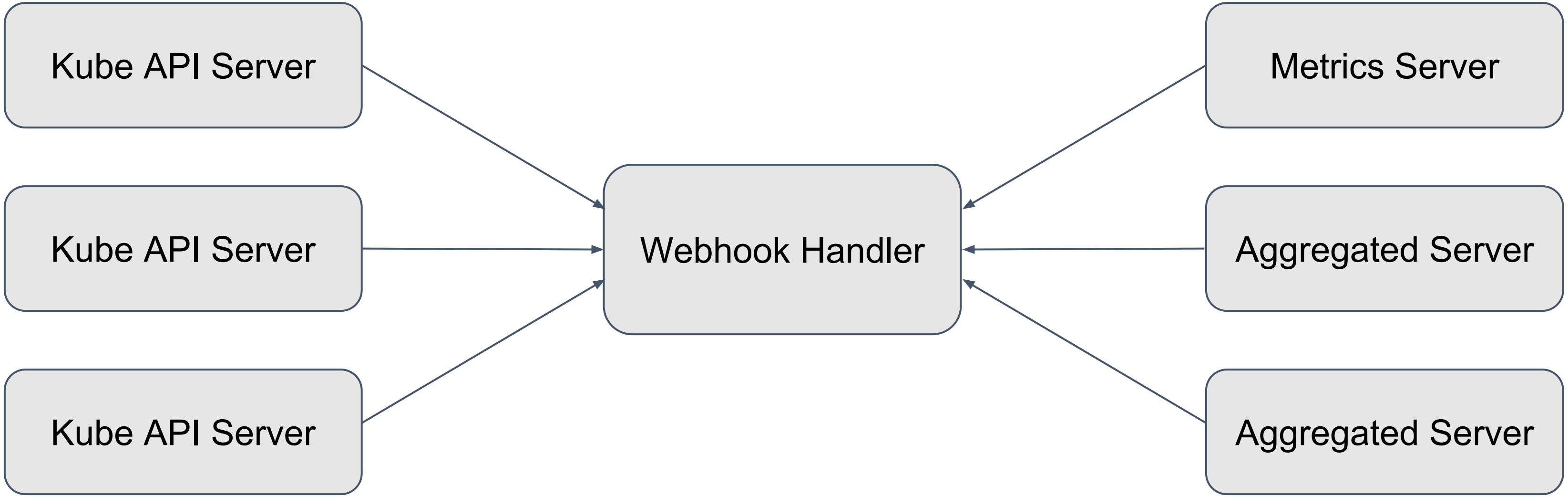


Audit + credential
Admission + credential
Conversion + credential
...

Problem 1: no ability to distinguish between API servers



Problem 2: assumes uniform permissions among API servers



Kube API Server

Kube API Server

Kube API Server

Webhook Handler

Metrics Server

Aggregated Server

Aggregated Server

Ideas

	Kubernetes-aware webhook	Kubernetes-unaware webhook
Uniform identity	Shared credential	Shared credential
Per-caller identity	TokenRequest Per-caller, per-webhook	???

Bringing the Certificates API to GA

kubernetes/kubernetes#69836

- API shape/issues
  - Requires requesters to know all the info about the end certificate.
  - Use for higher-level requests (i.e. profiles).
  - Requested certificate attributes split unthoughtful between encoded CSR and fields in request spec which create difference in semantics.
- Approval flow/issues
  - Cannot limit or add components to request (limit or add SANs, usages, etc)
- Signing flow/issues
  - Method for multiple signers to interact (or approver to indicate what signer should be used)
- Guarantees on issued certificates
  - No (current) guarantee all requested extensions/SANs are issued
  - No (current) guarantee issued client certificates will be accepted as API client certs