
KubeCon
CloudNativeCon
North America 2019


# Making the Most Out of Kubernetes Audit Logs

*Robert Boll* *@roboll_*
*Laurent Bernaille* *@lbernail*

# Datadog

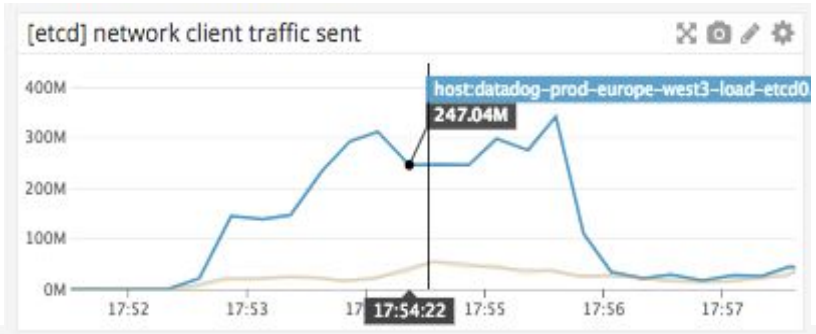


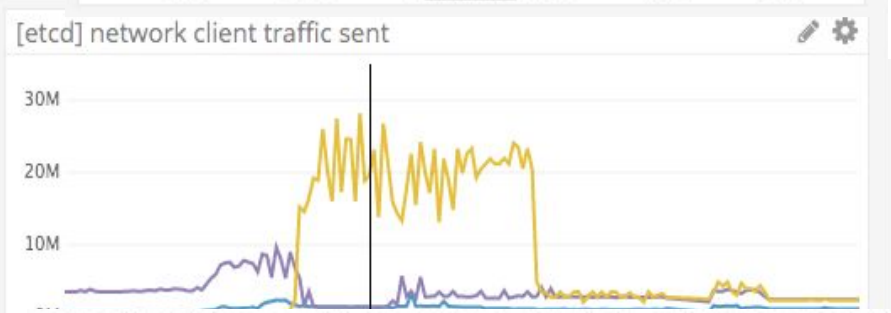
Monitoring service
Over 350 integrations
Over 1,200 employees
Over 8,000 customers
Runs on millions of hosts
Trillions of data points per day

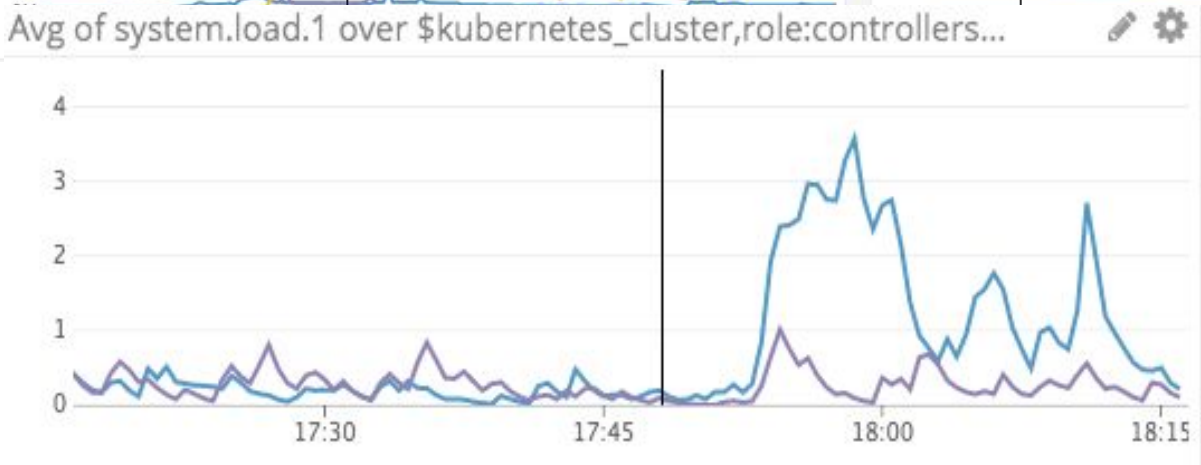
10000s hosts in our infra
10s of Kubernetes clusters
Clusters from 50 to 3000 nodes
Multi-cloud
Very fast growth

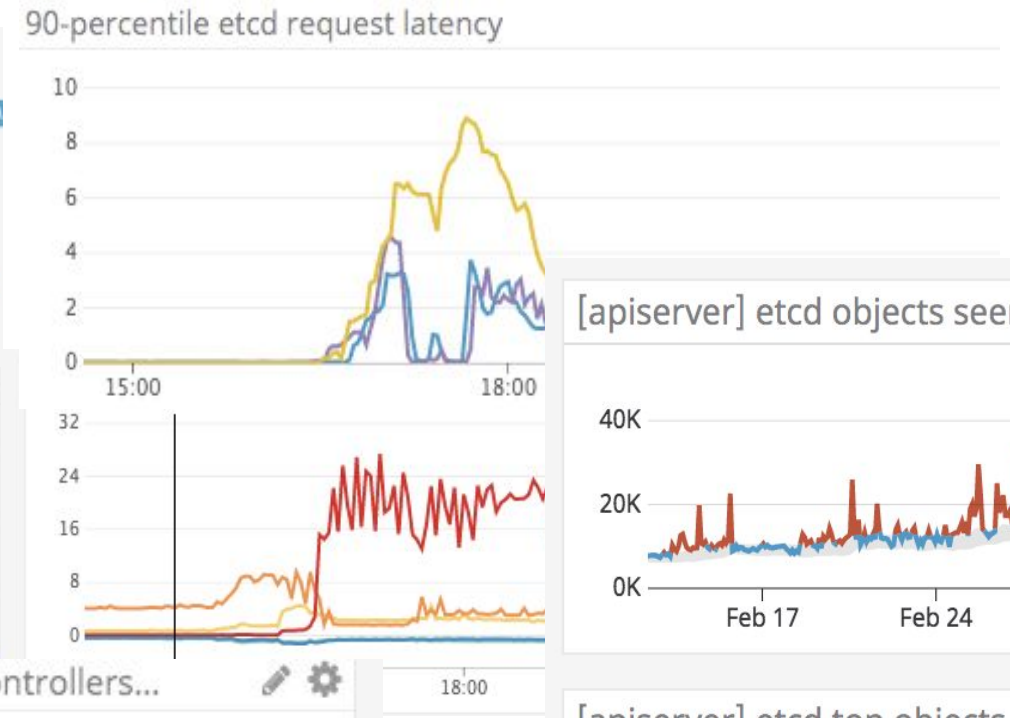
# Understanding what happens can be hard





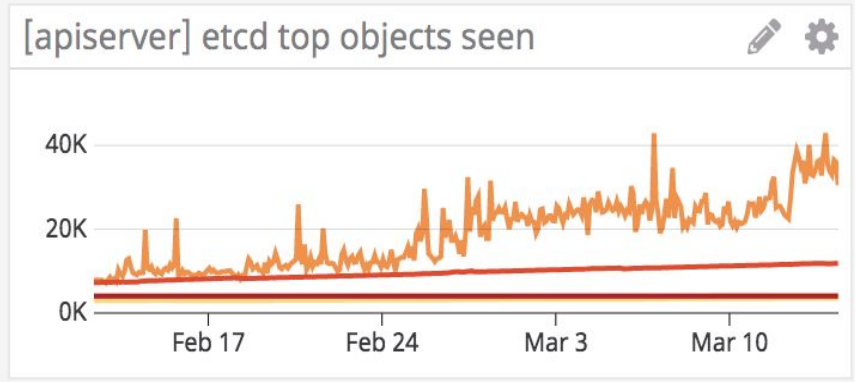









# Outline

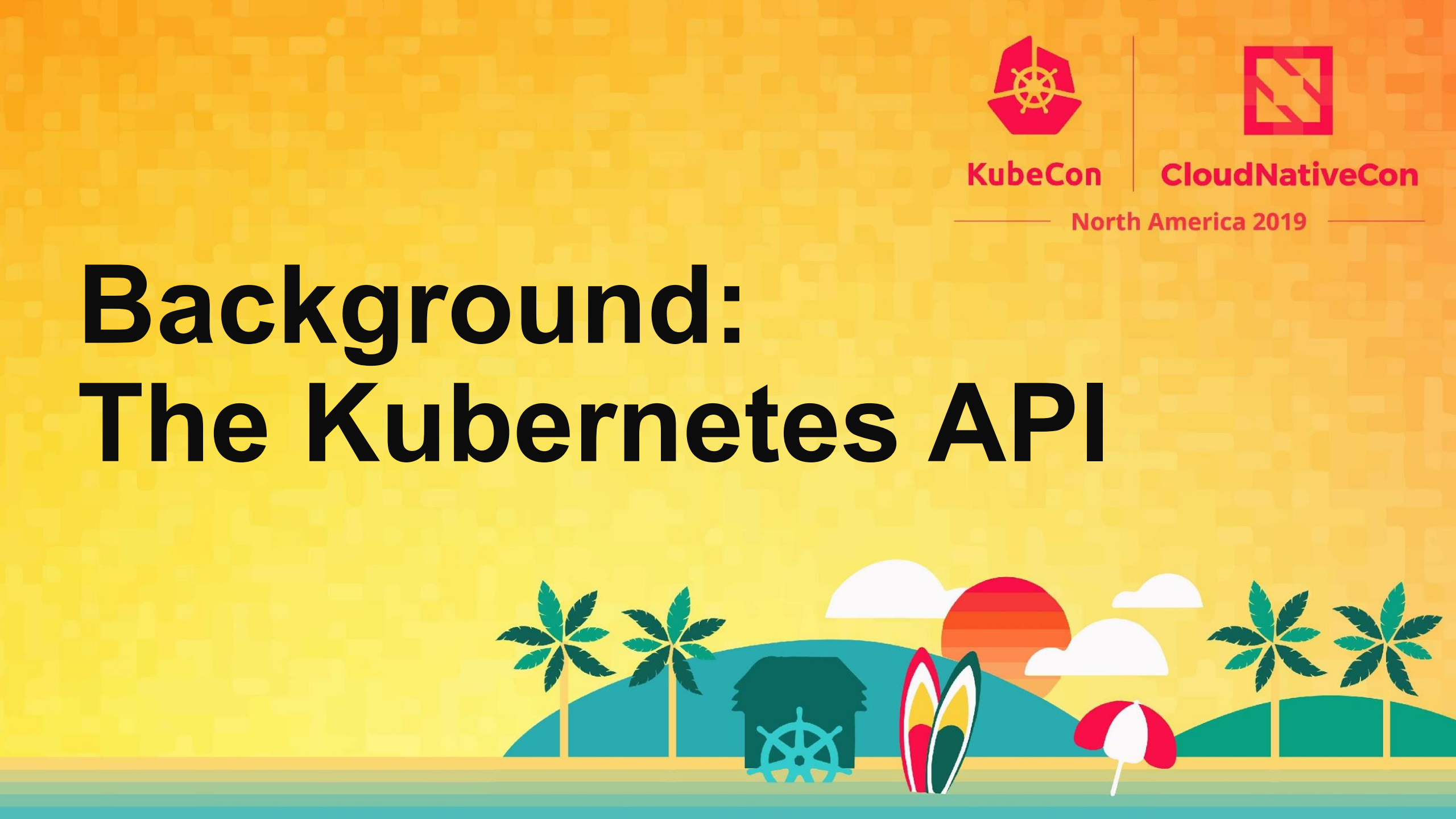


1. Background: The Kubernetes API
2. Audit Logs
3. Configuring Audit Logs
4. 10000 foot view for a large cluster
5. Understanding Kubernetes Internals
6. Troubleshooting examples

# Outline

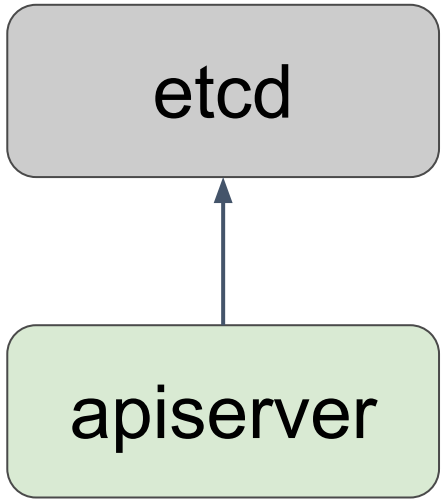


1. **Background: The Kubernetes API**
2. Audit Logs
3. Configuring Audit Logs
4. 10000 foot view for a large cluster
5. Understanding Kubernetes Internals
6. Troubleshooting examples


KubeCon
CloudNativeCon
North America 2019
Background:
The Kubernetes API

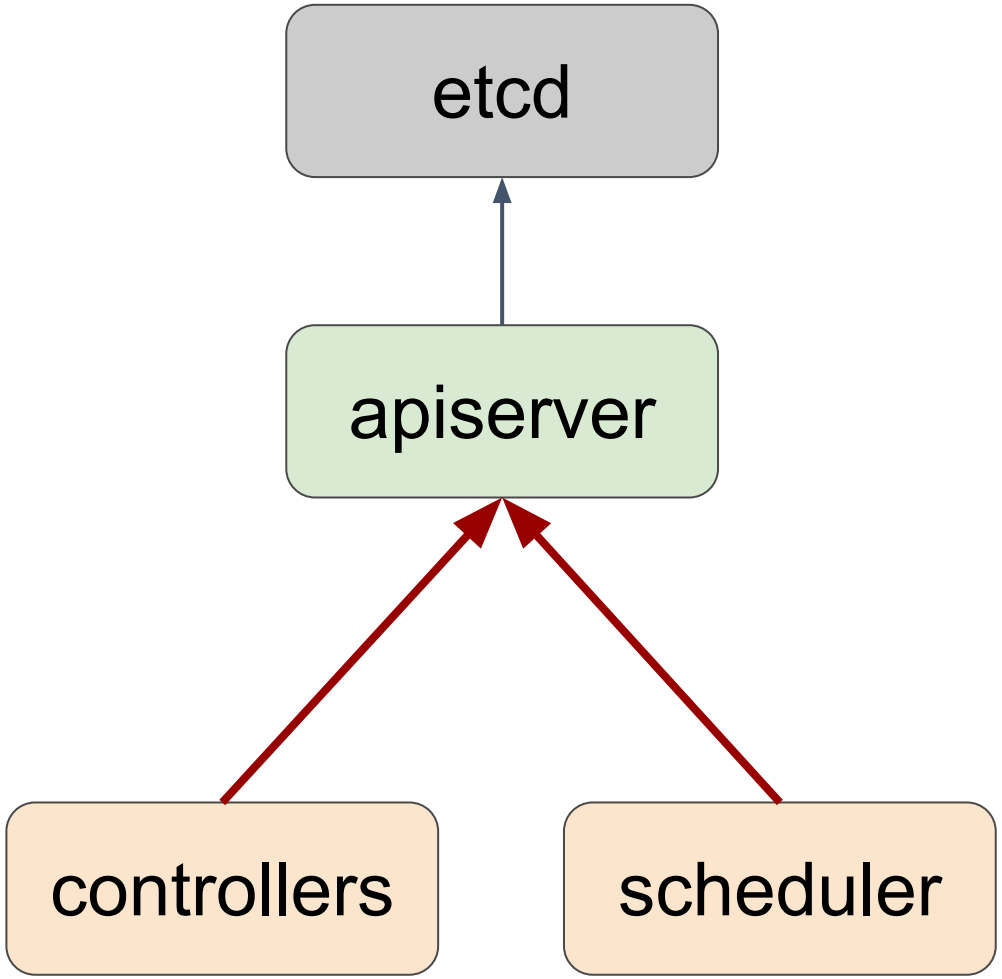
# Calls to the apiservers





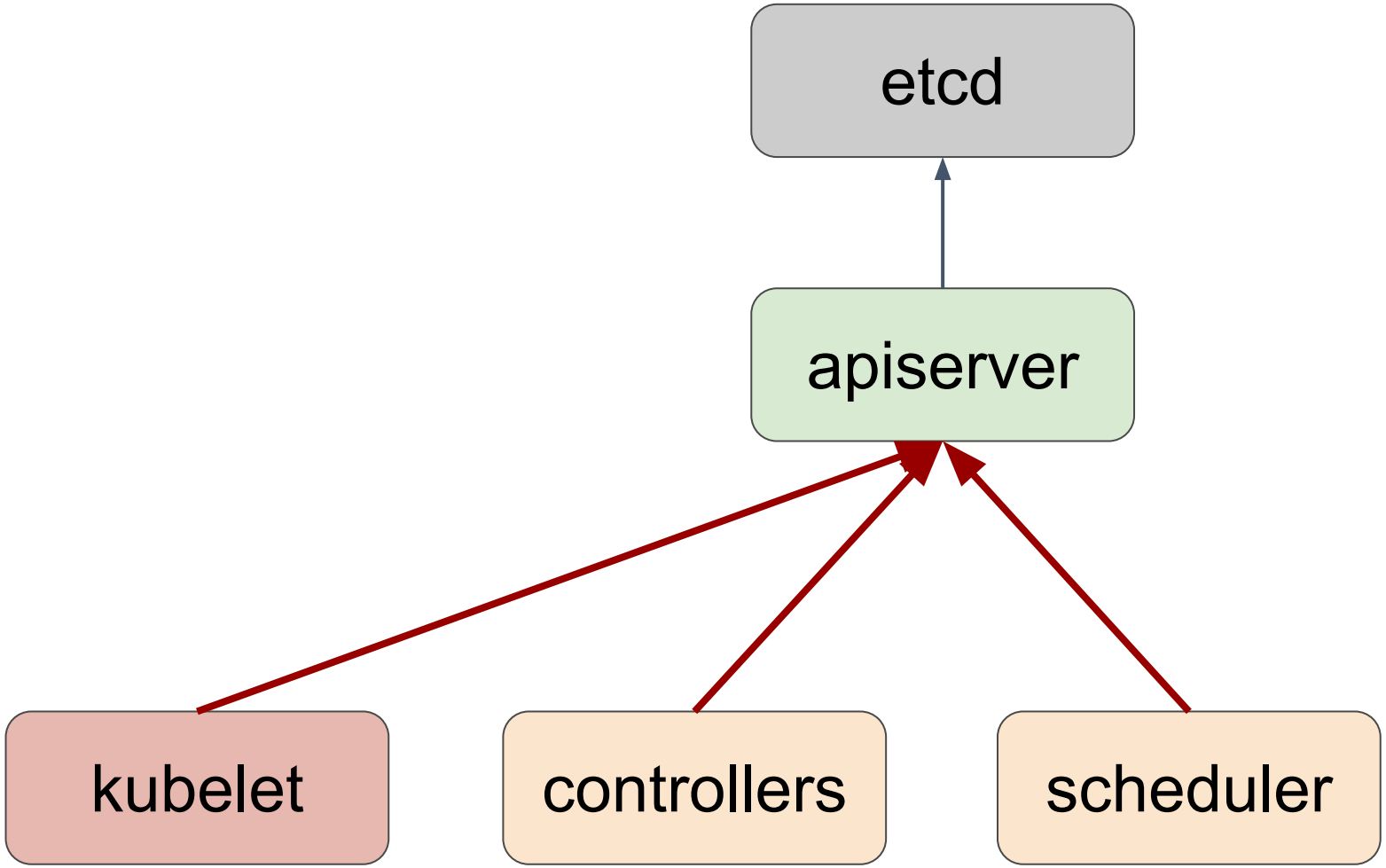
# Control plane





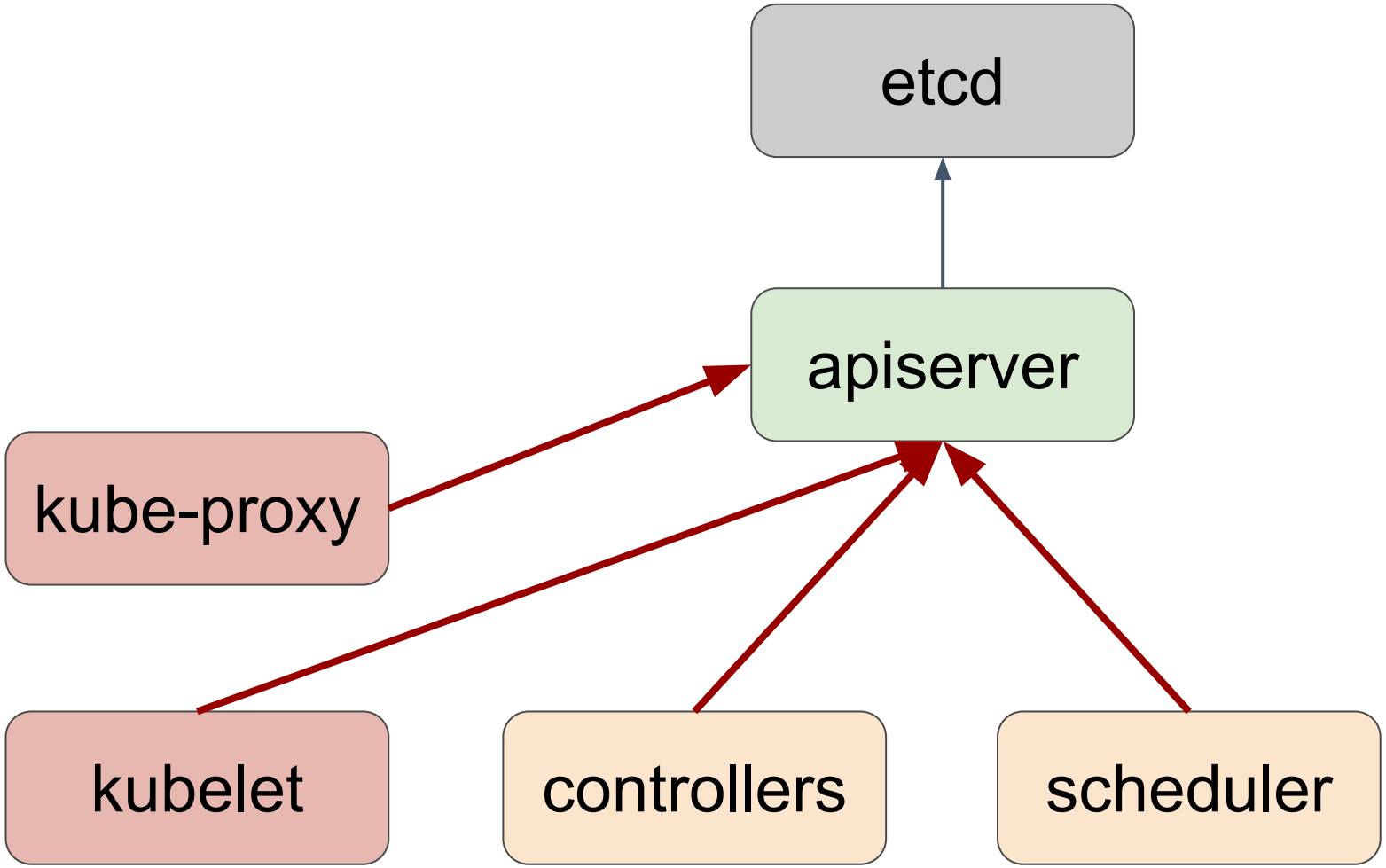
# Kubelet





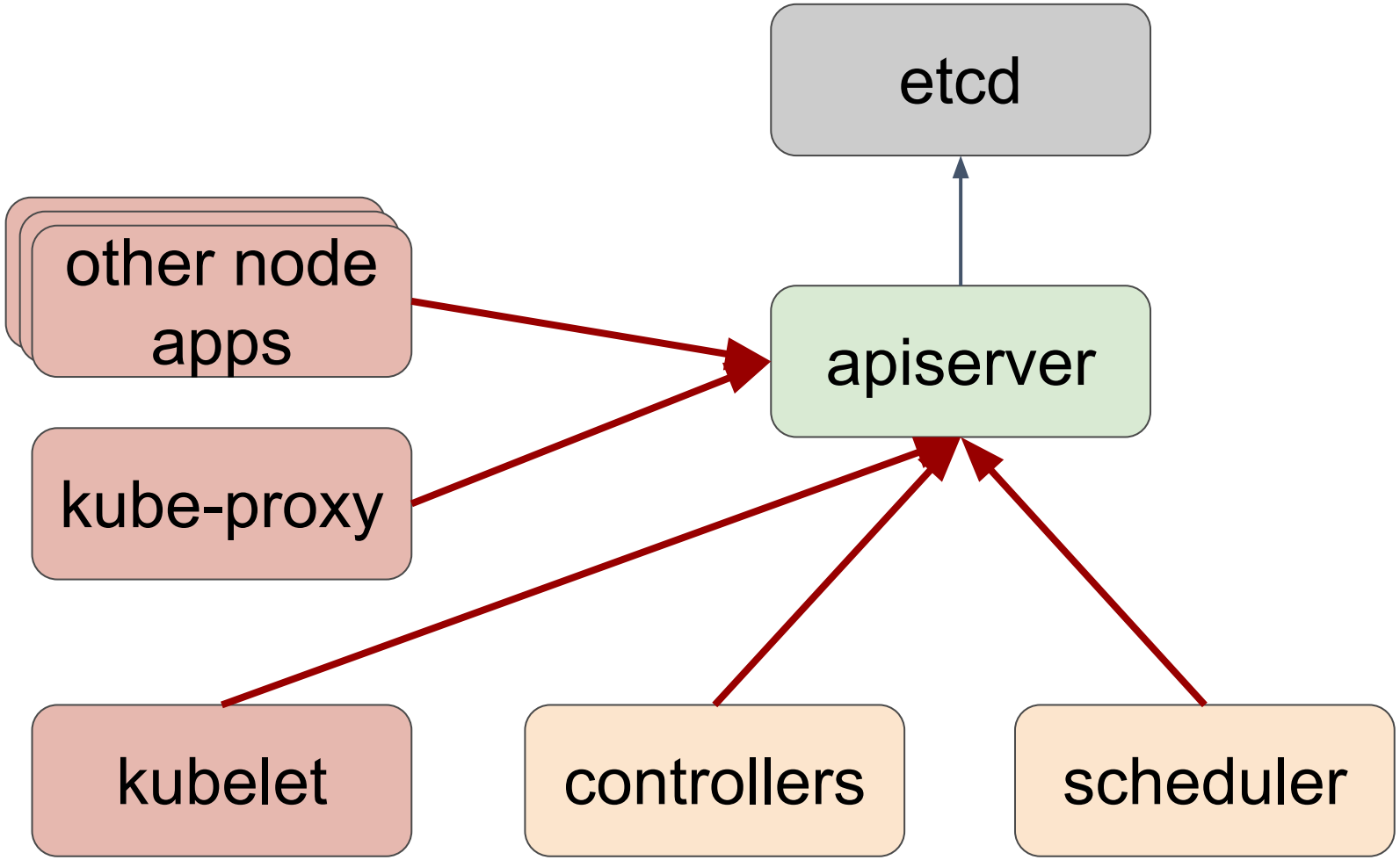
# DaemonSet: kube-proxy





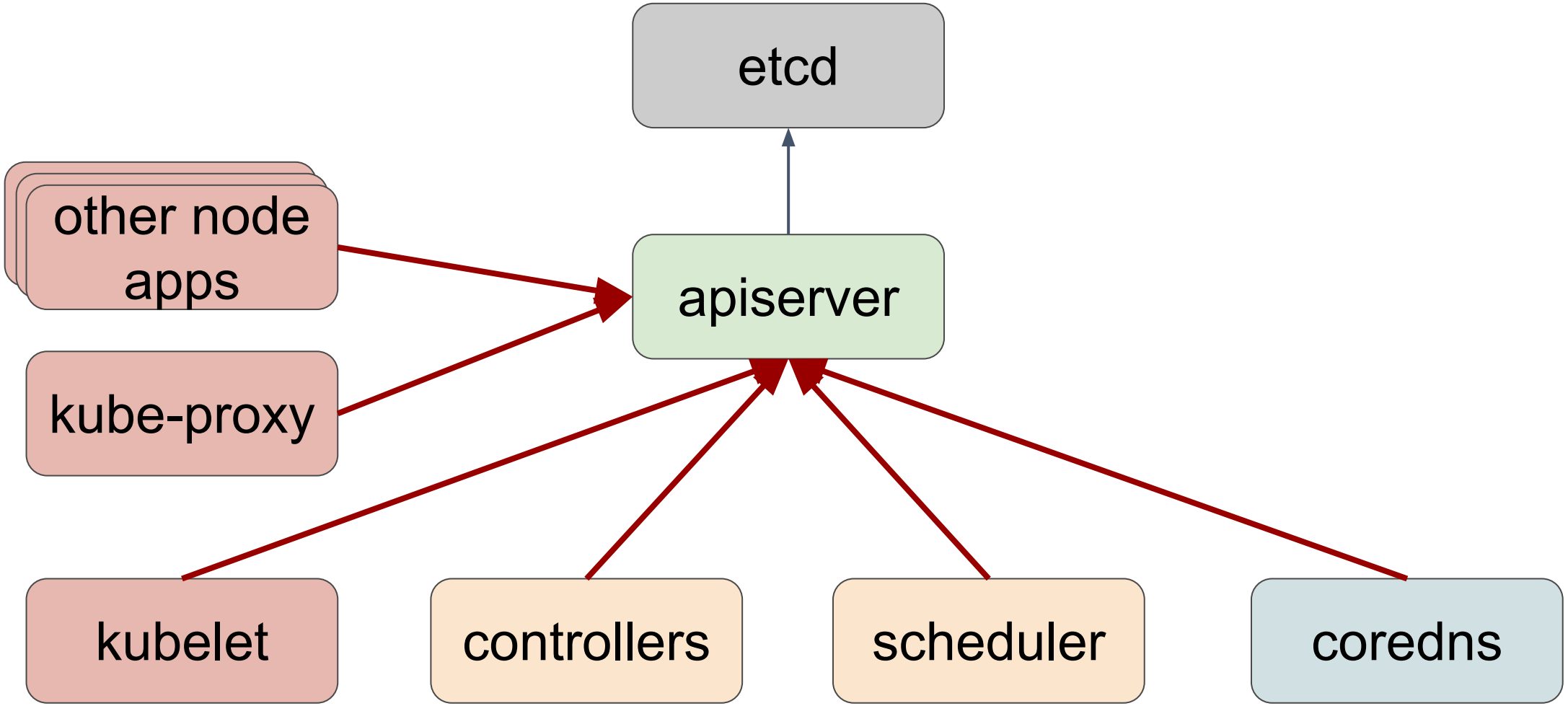
# Other DaemonSets (cni, etc.)





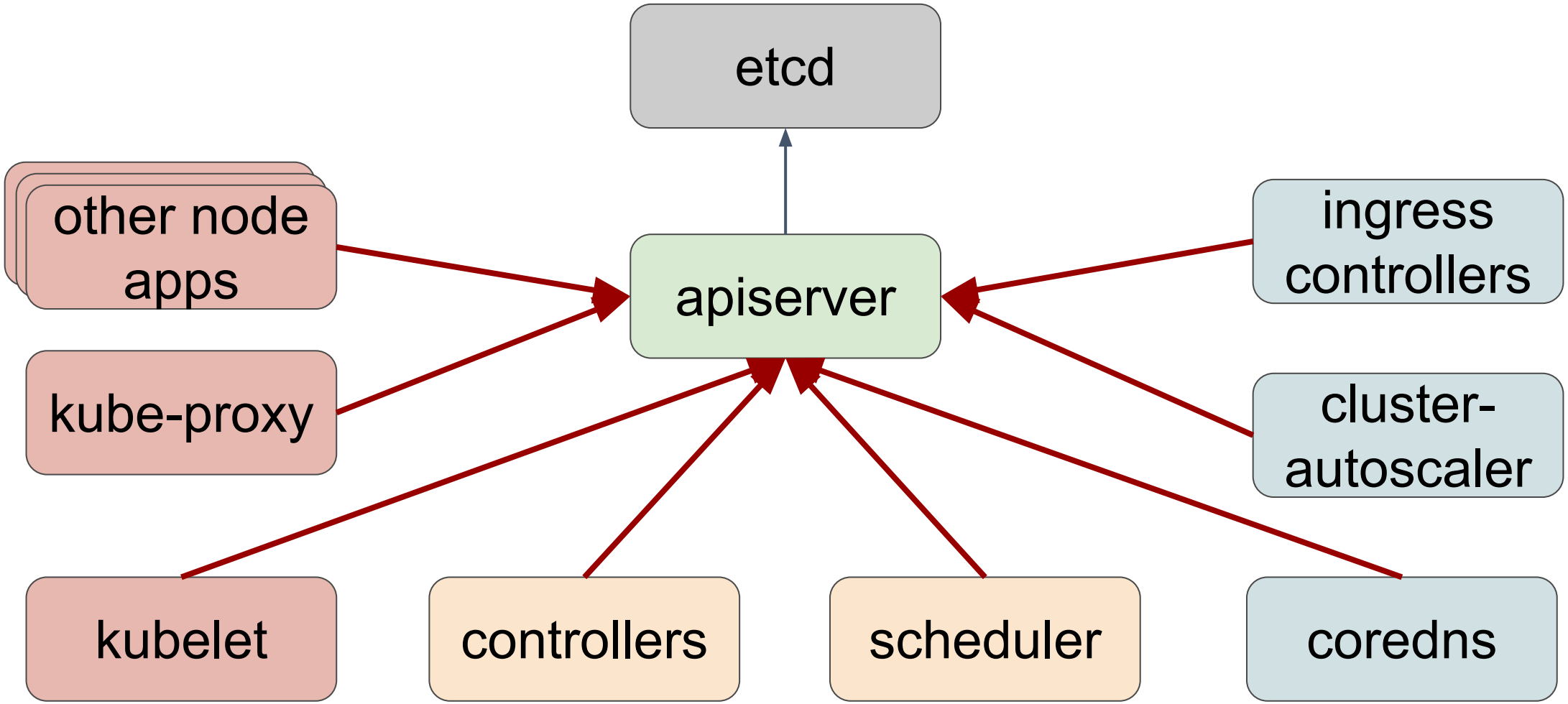
# Cluster services: DNS





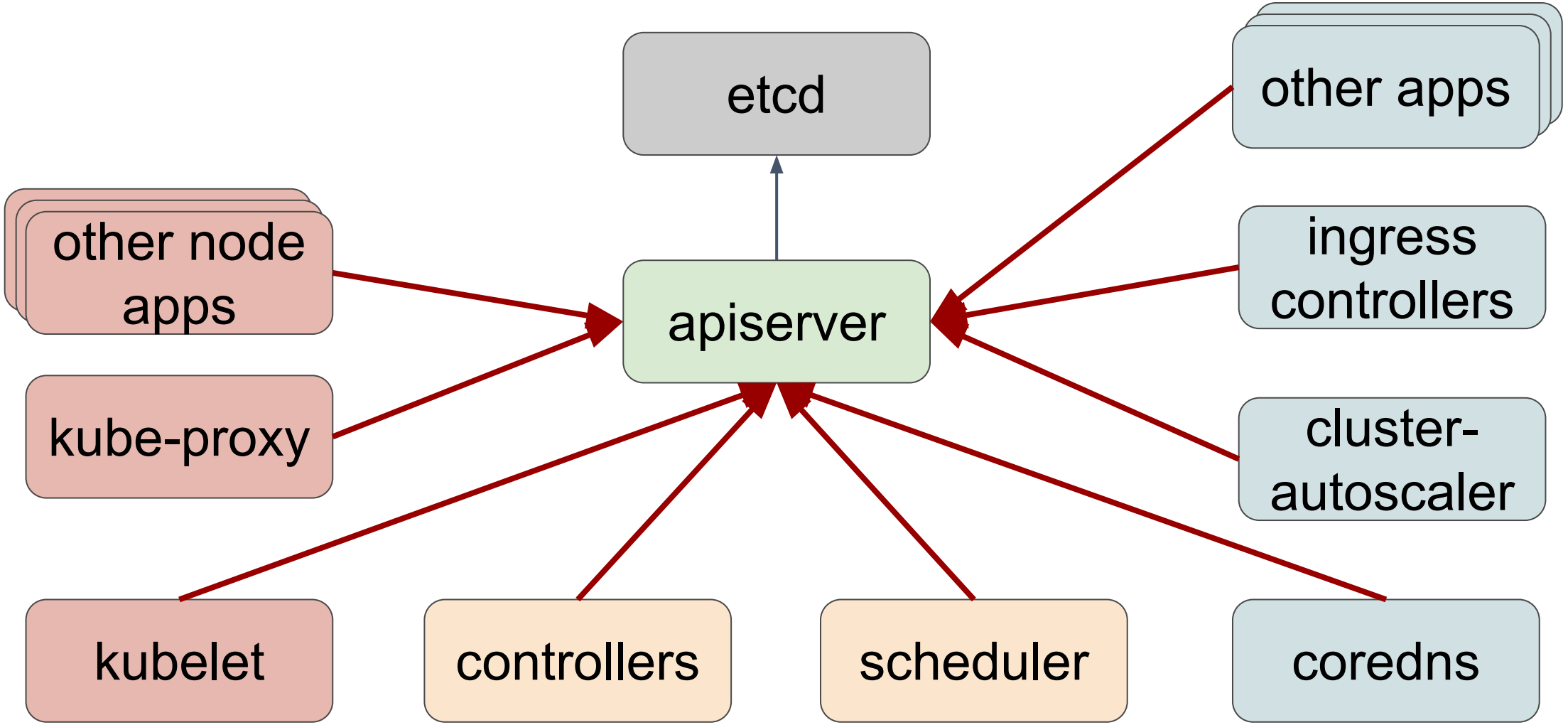
# Other cluster services





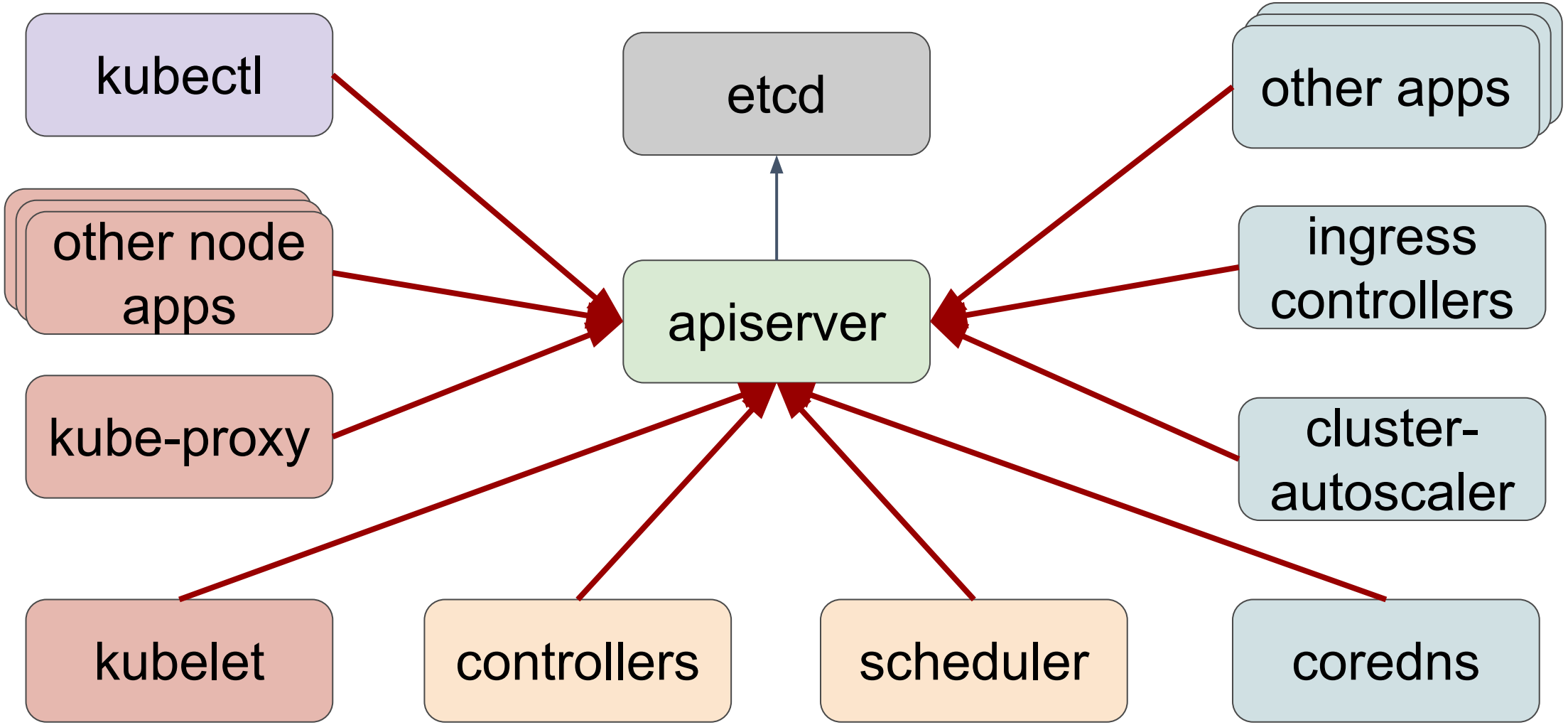
# Probably several other applications





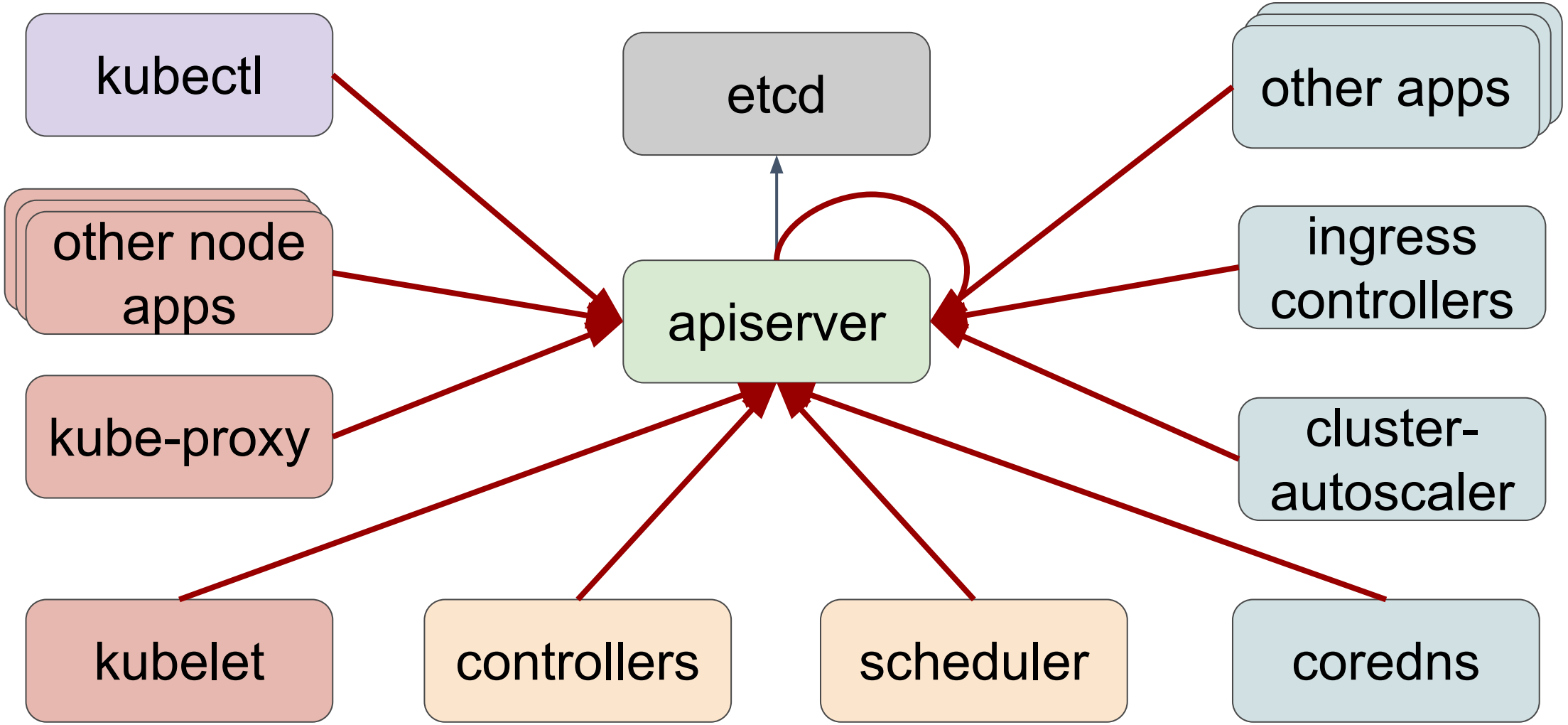
# And users, of course





# And, surprise, the apiserver itself



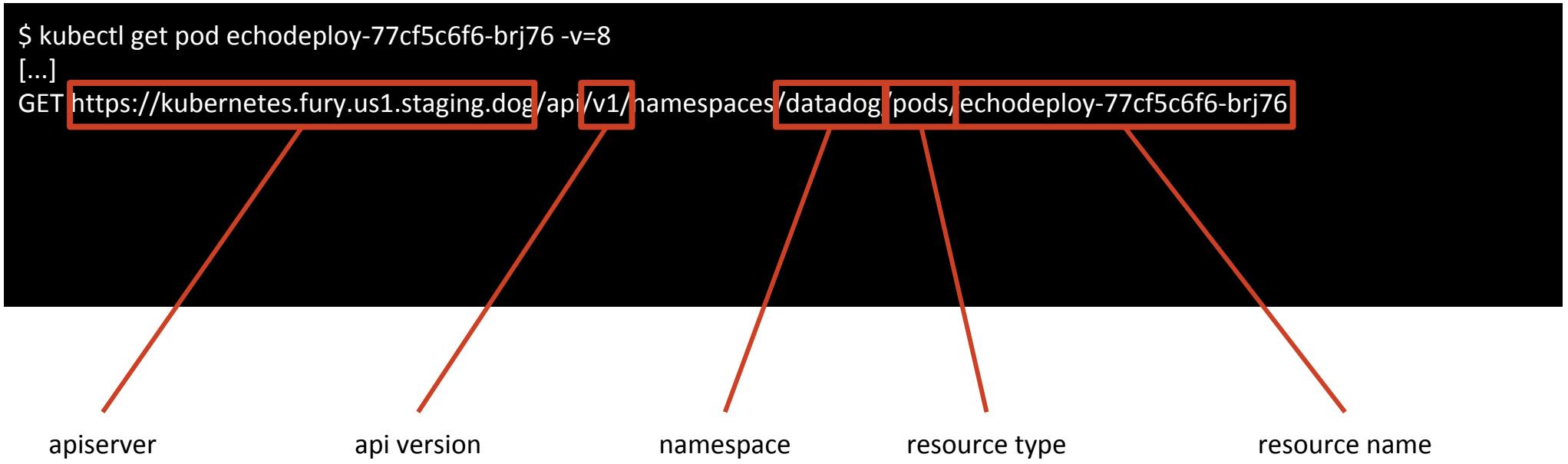


# What happens when you kubectl?


```
$ kubectl get pod echodeploy-77cf5c6f6-brj76 -v=8
[...]
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods/echodeploy-77cf5c6f6-brj76
```

# Let's look at details





# A few more GET examples



```
$ kubectl get pod echodeploy-77cf5c6f6-brj76 -v=8
[...]
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods/echodeploy-77cf5c6f6-brj76


$ kubectl get pods
[...]
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods?limit=500
```

List (paginated)

# A few more GET examples



```
$ kubectl get pod echodeploy-77cf5c6f6-brj76 -v=8
[...]
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods/echodeploy-77cf5c6f6-brj76


$ kubectl get pods
[...]
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods?limit=500


$ kubectl get pods --watch=true -v=8
[...]

GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods?limit=500
[...]
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods?resourceVersion=282725545&watch=true
```

List & Watch

# Describe resource



```
kubectl describe pod echodeploy-77cf5c6f6-5wmw9 -v=8
[...]
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods/echodeploy-77cf5c6f6-5wmw9
[...]
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/events?
        fieldSelector=involvedObject.name=echodeploy-77cf5c6f6-5wmw9,
                      involvedObject.namespace=datadog,
                      involvedObject.uid=770b3a5e-0631-11ea-bc60-12d7306f3c0c
[...]
ResponseBody
{
  "kind": "EventList",
   "items": [
    {
     "involvedObject": { "kind": "Pod", "namespace": "datadog", "name": "echodeploy-77cf5c6f6-5wmw9"},
     "reason": "Scheduled",
     "message": "Successfully assigned echodeploy-77cf5c6f6-5wmw9 to ip-10-128-205-156.ec2.internal",
     "source": {
       "component": "default-scheduler"
     },
    }
  ]
}
```

# Describe resource



```
kubectl describe pod echodeploy-77cf5c6f6-5wmw9 -v=8
[...]
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods/echodeploy-77cf5c6f6-5wmw9
[...]
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/events?
        fieldSelector=involvedObject.name=echodeploy-77cf5c6f6-5wmw9,
                      involvedObject.namespace=datadog,
                      involvedObject.uid=770b3a5e-0631-11ea-bc60-12d7306f3c0c
[...]
ResponseBody
{
  "kind": "EventList",
   "items": [
    {
      "involvedObject": { "kind": "Pod", "namespace": "datadog", "name": "echodeploy-77cf5c6f6-5wmw9"},
      "reason": "Scheduled",
      "message": "Successfully assigned echodeploy-77cf5c6f6-5wmw9 to ip-10-128-205-156.ec2.internal",
      "source": {
        "component": "default-scheduler"
      },
    }
  ]
}
```

Get resource

# Describe resource



```
kubectl describe pod echodeploy-77cf5c6f6-5wmw9 -v=8
[...]
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods/echodeploy-77cf5c6f6-5wmw9
[...]
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/events?
        fieldSelector=involvedObject.name=echodeploy-77cf5c6f6-5wmw9,
                      involvedObject.namespace=datadog,
                      involvedObject.uid=770b3a5e-0631-11ea-bc60-12d7306f3c0c
[...]
ResponseBody
{
  "kind": "EventList",
   "items": [
    {
      "involvedObject": { "kind": "Pod", "namespace": "datadog", "name": "echodeploy-77cf5c6f6-5wmw9"},
      "reason": "Scheduled",
      "message": "Successfully assigned echodeploy-77cf5c6f6-5wmw9 to ip-10-128-205-156.ec2.internal",
      "source": {
        "component": "default-scheduler"
      },
    }
  ]
}
```

Get resource

Get **events** associated with resource

# A few other examples



```
$ kubectl delete pod echodeploy-77cf5c6f6-brj76 -v=8
[...]
DELETE https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods/echodeploy-77cf5c6f6-brj76
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods?fieldSelector=metadata.name%3Dechodeploy-77cf5c6f6-brj76
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods?fieldSelector=metadata.name%3Dechodeploy-77cf5c6f6-brj76&
    resourceVersion=282733788&watch=true
```

Delete + List & Watch

# A few other examples



```
$ kubectl delete pod echodeploy-77cf5c6f6-brj76 -v=8
[...]
DELETE https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods/echodeploy-77cf5c6f6-brj76
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods?fieldSelector=metadata.name%3Dechodeploy-77cf5c6f6-brj76
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods?fieldSelector=metadata.name%3Dechodeploy-77cf5c6f6-brj76&
      resourceVersion=282733788&watch=true


$ kubectl create deployment test --image=busybox -v=8
Request Body:
{"apiVersion":"apps/v1","kind":"Deployment","metadata":{"creationTimestamp":null,"labels":{"app":"test"},"name":"test"},"spec":{"replicas":1,"s
elector":{"matchLabels":{"app":"test"}},"strategy":{},"template":{"metadata":{"creationTimestamp":null,"labels":{"app":"test"}},"spec":{"container
s":[{"image":"busybox","name":"busybox","resources":{}}]}},"status":{}}

POST https://kubernetes.fury.us1.staging.dog/apis/apps/v1/namespaces/datadog/deployments
```

Minimal deployment spec

POST call

# A few other examples



```
$ kubectl delete pod echodeploy-77cf5c6f6-brj76 -v=8
[...]
DELETE https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods/echodeploy-77cf5c6f6-brj76
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods?fieldSelector=metadata.name%3Dechodeploy-77cf5c6f6-brj76
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods?fieldSelector=metadata.name%3Dechodeploy-77cf5c6f6-brj76&
      resourceVersion=282733788&watch=true


$ kubectl create deployment test --image=busybox -v=8
Request Body:
{"apiVersion":"apps/v1","kind":"Deployment","metadata":{"creationTimestamp":null,"labels":{"app":"test"},"name":"test"},"spec":{"replicas":1,"s
elector":{"matchLabels":{"app":"test"}},"strategy":{},"template":{"metadata":{"creationTimestamp":null,"labels":{"app":"test"}},"spec":{"container
s":[{"image":"busybox","name":"busybox","resources":{}}]}},"status":{}}

POST https://kubernetes.fury.us1.staging.dog/apis/apps/v1/namespaces/datadog/deployments


$ kubectl scale deploy test --replicas=2 -v=8
GET https://kubernetes.fury.us1.staging.dog/apis/extensions/v1beta1/namespaces/datadog/deployments/test

Request Body: {"spec":{"replicas":2}}
PATCH https://kubernetes.fury.us1.staging.dog/apis/extensions/v1beta1/namespaces/datadog/deployments/test/scale
```

GET current

PATCH body +call

# Takeaways



- A lot of components are making calls
  - Control plane: controllers, scheduler
  - Node daemons: kubelet, kube-proxy
  - Other controllers: autoscaler, ingress
- “Simple” user ops translate to **many** API calls

How can we understand what is going on?

# Outline



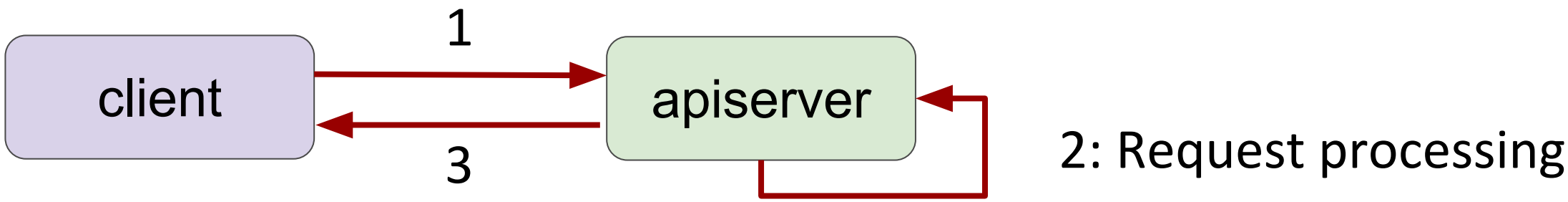
1. Background: The Kubernetes API
2. **Audit Logs**
3. Configuring Audit Logs
4. 10000 foot view for a large cluster
5. Understanding Kubernetes Internals
6. Troubleshooting examples


# Audit Logs

# What are Audit Logs?



- **Rich Structured json logs** output by the apiserver
- **Configurable Verbosity** for each resource
- Logging can happen at **different processing stages**




1: Apiserver receives request, Stage: RequestReceived
2: Apiserver processes request
3: Apiserver answers, Stage: ResponseComplete/ResponseStarted

# Content of Audit Logs



- What happened?
- Who initiated it?
- Why was it authorized?
- When did it happen?
- From where?
- Depending on verbosity, Request/Response

# GET example from earlier


```
$ kubectl get pod echodeploy-77cf5c6f6-5wmw9 -v=8
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods/echodeploy-77cf5c6f6-5wamw9
```


```
{
  apiVersion: audit.k8s.io/v1beta1
  auditID: 0fe92a67-248e-461e-b5ec-af6e90f81c49
  http {
    status_code: 200
  }
  kind: Event
  level: Metadata
  metadata {
    creationTimestamp: 2019-11-19T13:49:30Z
  }
  objectRef {
    apiVersion: v1
    name: echodeploy-77cf5c6f6-5wmw9
    namespace: datadog
    resource: pods
  }
  requestReceivedTimestamp: 2019-11-19T13:49:30.882458Z
  requestURI: /api/v1/namespaces/datadog/pods/echodeploy-77cf5c6f6-5wmw9
  responseStatus {
    code: 200
  }
  sourceIPs [
    10.128.215.38
  ]
  stage: ResponseComplete
  stageTimestamp: 2019-11-19T13:49:30.885163Z
  timestamp: 2019-11-19T13:49:30Z
  user {
    groups [
      datadoghq.com
      system:authenticated
    ]
    username: laurent.bernaille@datadoghq.com
  }
  verb: get
}
```


Structured JSON log
A lot of information

# What happened?



```
$ kubectl get pod echodeploy-77cf5c6f6-5wmw9 -v=8
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods/echodeploy-77cf5c6f6-5wmw9
```

```
requestURI      /api/v1/namespaces/datadog/pods/echodeploy-77cf5c6f6-5wmw9
verb            get
objectRef {
    apiVersion  v1
    name        echodeploy-77cf5c6f6-5wmw9
    namespace   datadog
    resource    pods
}
responseStatus {
    code  200
}
```

# Who initiated it?



```
$ kubectl get pod echodeploy-77cf5c6f6-5wmw9 -v=8
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods/echodeploy-77cf5c6f6-5wamw9
```


```
user {
  groups [
    datadoghq.com
    system:authenticated
  ]
  username  laurent.bernaille@datadoghq.com
}
```


User was **laurent.bernaille@datadoghq.com** and mapped to groups

- **datadoghq.com**
- **system:authenticated**

# Why was it authorized?



```
$ kubectl get pod echodeploy-77cf5c6f6-5wmw9 -v=8
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods/echodeploy-77cf5c6f6-5wamw9
```

```
authorization {
  k8s {
      io/decision  allow
      io/reason    RBAC: allowed by ClusterRoleBinding "datadoghq:cluster-admin-binding" of ClusterRole
                   "datadoghq:cluster-user" to Group "datadoghq.com"
  }
}
```

It was authorized because group **datadoghq.com** is bound to role **datadoghq:cluster-user** by ClusterRoleBinding **datadoghq:cluster-admin-binding** (and this role has the required permissions)

# When, and from where?



```
$ kubectl get pod echodeploy-77cf5c6f6-5wmw9 -v=8
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods/echodeploy-77cf5c6f6-5wamw9
```


requestReceivedTimestamp 2019-11-13T20:33:26.757736Z
stage ResponseComplete
stageTimestamp 2019-11-13T20:33:26.771303Z

sourceIPs [
10.1 74
]


Request received at **20:33:26.757**
Response completed at **20:33:26:771**
Duration: **14ms**
From IP: **10.X.Y.74**

# Another GET call



```
$ kubectl get pods  -v=8
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods?limit=500
```

```
requestURI              /api/v1/namespaces/datadog/pods?limit=500
verb                    list
objectRef {
    apiVersion    v1
    namespace     datadog
    resource      pods
}
```

**GET is mapped to different verbs (get/list)**

# Watches


```
$ kubectl get pods  -v=8 -w
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods?limit=500
GET https://kubernetes.fury.us1.staging.dog/api/v1/namespaces/datadog/pods?resourceVersion=288656279&watch=true
```

```
objectRef {
    apiVersion    v1
    namespace     datadog
    resource      pods
  }
  requestReceivedTimestamp  2019-11-19T14:17:06.162487Z
  requestURI                /api/v1/namespaces/datadog/pods?limit=500
+ responseStatus {...}
+ sourceIPs [...]
  stage                     ResponseComplete
  stageTimestamp            2019-11-19T14:17:06.165555Z
  timestamp                 2019-11-19T14:17:06Z
+ user {...}
  verb                      list
```

**Call 1 : list**
stage: ResponseComplete

```
objectRef {
    apiVersion    v1
    namespace     datadog
    resource      pods
  }
  requestReceivedTimestamp  2019-11-19T14:17:06.301721Z
  requestURI                /api/v1/namespaces/datadog/pods?resourceVersion=288656279&watch=true
+ responseStatus {...}
+ sourceIPs [...]
  stage                     ResponseStarted
  stageTimestamp            2019-11-19T14:17:06.302044Z
  timestamp                 2019-11-19T14:17:06Z
+ user {...}
  verb                      watch
```

**Call 2 : watch**
get + watch parameters
136ms later
stage: ResponseStarted

@lbernail @roboll_

# Create call



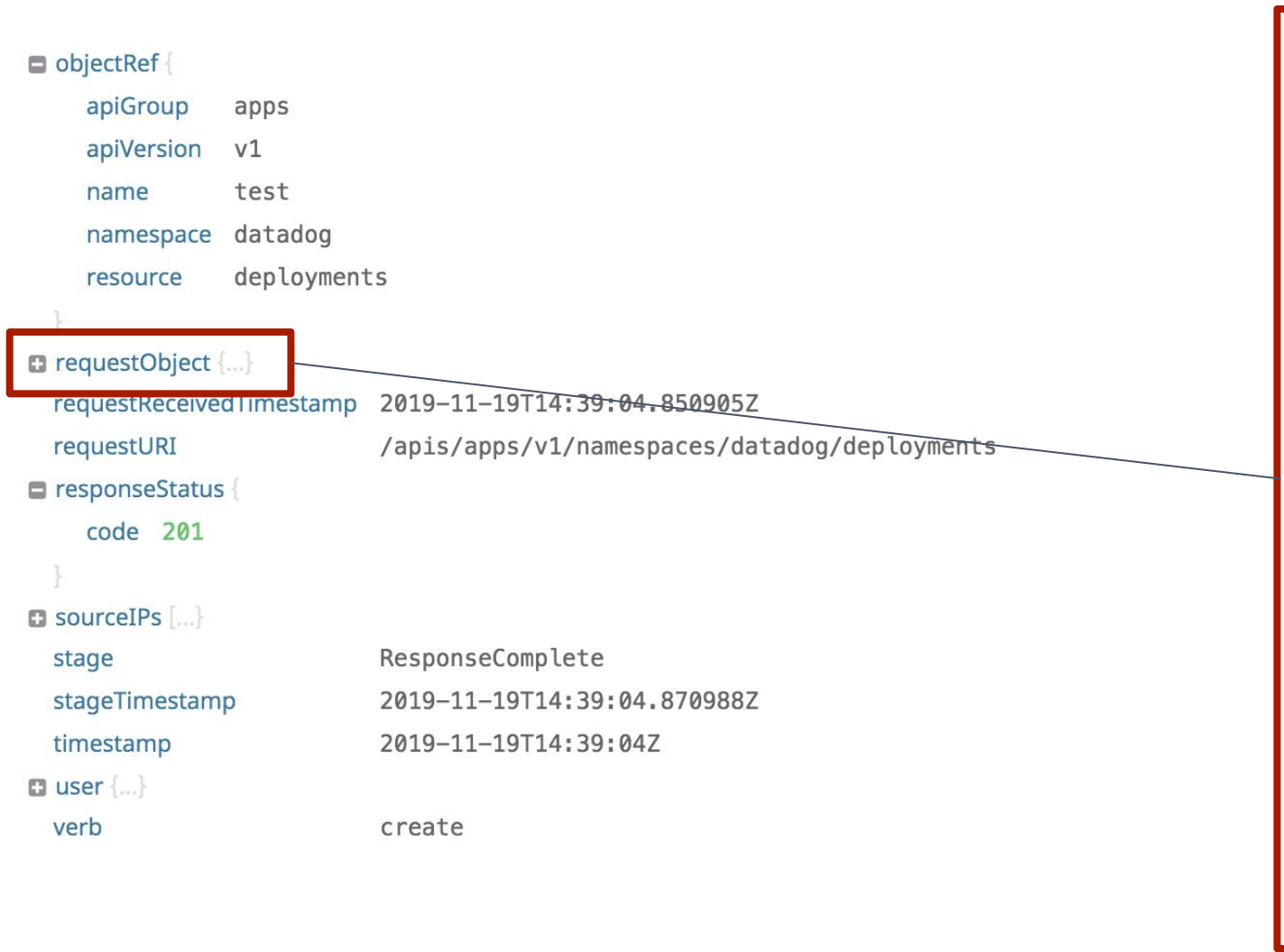
```
$ kubectl create deployment test --image=busybox -v=8
POST https://kubernetes.fury.us1.staging.dog/apis/apps/v1/namespaces/datadog/deployments
```

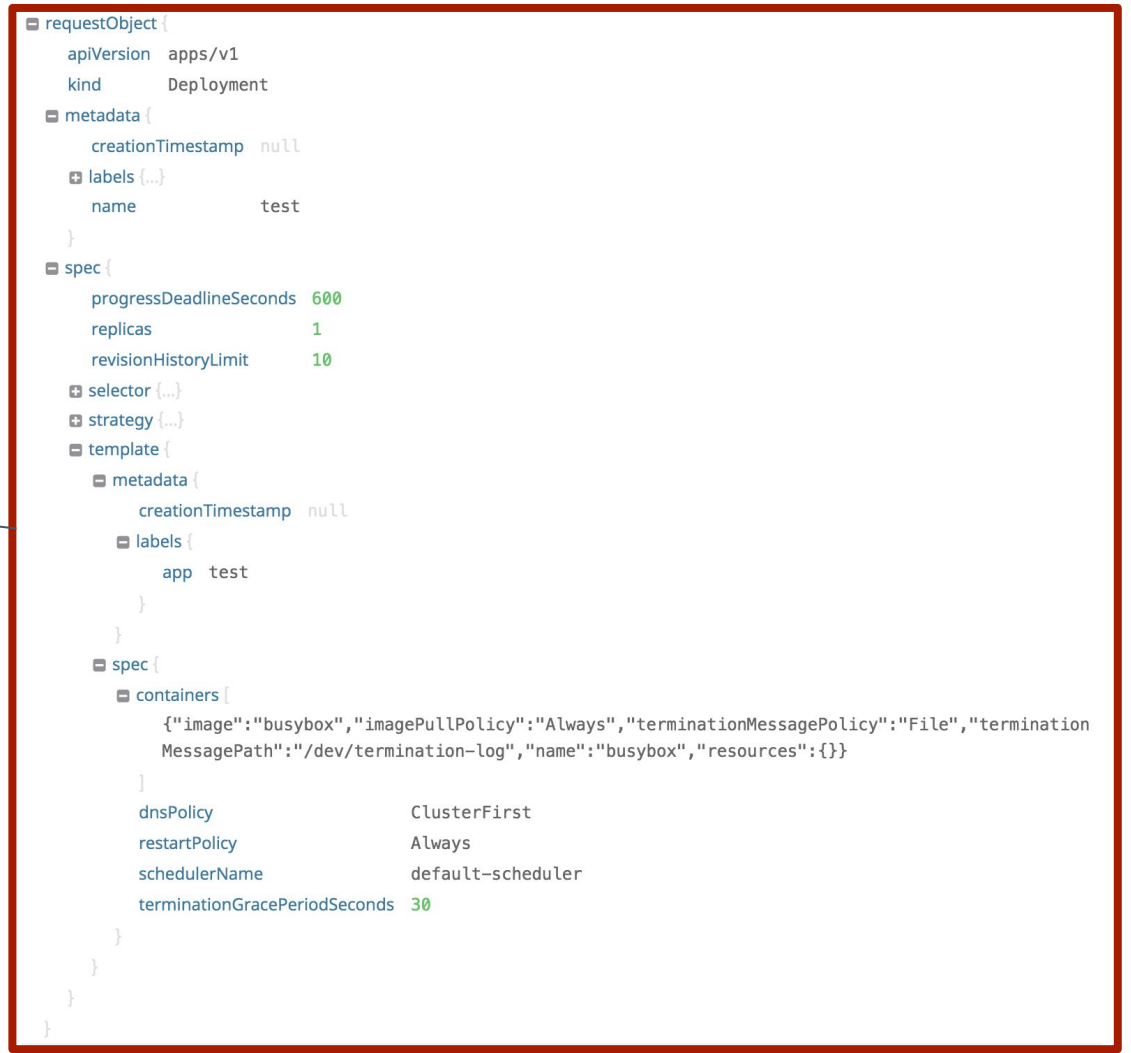
```
objectRef {
    apiGroup     apps
    apiVersion   v1
    name         test
    namespace    datadog
    resource     deployments
}
requestObject {...}
requestReceivedTimestamp  2019-11-19T14:39:04.850905Z
requestURI                /apis/apps/v1/namespaces/datadog/deployments
responseStatus {
    code  201
}
sourceIPs [...]
stage                     ResponseComplete
stageTimestamp            2019-11-19T14:39:04.870988Z
timestamp                 2019-11-19T14:39:04Z
user {...}
verb                      create
```

# Create call



```
$ kubectl create deployment test --image=busybox -v=8
POST https://kubernetes.fury.us1.staging.dog/apis/apps/v1/namespaces/datadog/deployments
```



# Takeaways



Audit logs contain information on all API calls

- What happened?
- Who initiated it?
- Why was it authorized?
- When did it happen?
- From where?
- Depending on verbosity, Request/Response

OK, how do I get them?

# Outline



1. Background: The Kubernetes API
2. Audit Logs
3. **Configuring Audit Logs**
4. 10000 foot view for a large cluster
5. Understanding Kubernetes Internals
6. Troubleshooting examples


KubeCon
CloudNativeCon
North America 2019
Configuring Audit Logs

# Apiserver configuration



Minimum configuration

```
kube-apiserver
  [...]
  --audit-log-path=/var/log/kubernetes/apiserver/audit.log

  --audit-policy-file=/etc/kubernetes/audit-policies/policy.yaml
```

Where to store them

What to collect

Advanced

- Rotation parameters (max size, backup options)
- Alternative backend (webhook)
- Batching mode

# Audit policy: what to log?



```
apiVersion: audit.k8s.io/v1 kind: Policy

rules:
  # Log pod changes at RequestResponse level
  - level: RequestResponse
    omitStages:
     - "RequestReceived"
    resources:
    - group: "" # core API group
      resources: ["pods"]
    verbs: ["create", "patch"", "update", "delete"]

  # Log "pods/log", "pods/status" at Metadata level
  - level: Metadata
    omitStages:
     - "RequestReceived"
    resources:
    - group: ""
        resources: ["pods/log", "pods/status"]
```

Set of rules

# Audit policy: what to log?



```
apiVersion: audit.k8s.io/v1 kind: Policy

rules:

 # Log pod changes at RequestResponse level
 - level: RequestResponse
   omitStages:
    - "RequestReceived"
   resources:
   - group: "" # core API group
     resources: ["pods"]
   verbs: ["create", "patch"", "update", "delete"]


 # Log "pods/log", "pods/status" at Metadata level
 - level: Metadata
   omitStages:
    - "RequestReceived"
   resources:
   - group: ""
      resources: ["pods/log", "pods/status"]
```

Rules match api call

- api group / version
- resource
- verbs

> Similar to RBAC

# Audit policy: when to log?



```
apiVersion: audit.k8s.io/v1 kind: Policy

rules:

 # Log pod changes at RequestResponse level
 - level: RequestResponse
   omitStages:
    - "RequestReceived"
   resources:
   - group: "" # core API group
     resources: ["pods"]
   verbs: ["create", "patch"", "update", "delete"]


 # Log "pods/log", "pods/status" at Metadata level
 - level: Metadata
   omitStages:
    - "RequestReceived"
   resources:
   - group: ""
      resources: ["pods/log", "pods/status"]
```

For matching API calls

- Which verbosity?
- When? (stage)

# Gotchas



```
apiVersion: audit.k8s.io/v1 kind: Policy

rules:

 # Log pod changes at RequestResponse level
 - level: RequestResponse
   omitStages:
    - "RequestReceived"
   resources:
   - group: "" # core API group
     resources: ["pods"]
   verbs: ["create", "patch"", "update", "delete"]


 # Log "pods/log", "pods/status" at Metadata level
 - level: Metadata
   omitStages:
    - "RequestReceived"
   resources:
   - group: ""
       resources: ["pods/log", "pods/status"]
```

Rules are evaluated in order
First matching rule sets level

Request/RequestResponse
 > contain payload data
Careful with security implications
 ex: tokenreviews calls

group: “” means core API only
Don’t forget to add

- 3rd party apiservices
- your apiservices

# Recommendations



- Ignore **RequestReceived** stage
- Use at least **Metadata** level for almost everything
  - Possibly ignore healthz, metrics
- Use **Request/Response** level for critical resource/verbs
  - Very valuable for retroactive debugging
  - Careful for large/sensitive request/response bodies

- Very complete example in GKE
  https://github.com/kubernetes/kubernetes/blob/master/cluster/gce/gci/configure-helper.sh
- Documentation
  https://kubernetes.io/docs/tasks/debug-application-cluster/audit/#audit-policy

# Takeaways



- Getting audit logs is “simple”: 2 flags
- Getting policies right is harder
- You will get **a lot** of logs
- Requires a solution to analyze them

Let’s look at an overview on a real large cluster

# Outline



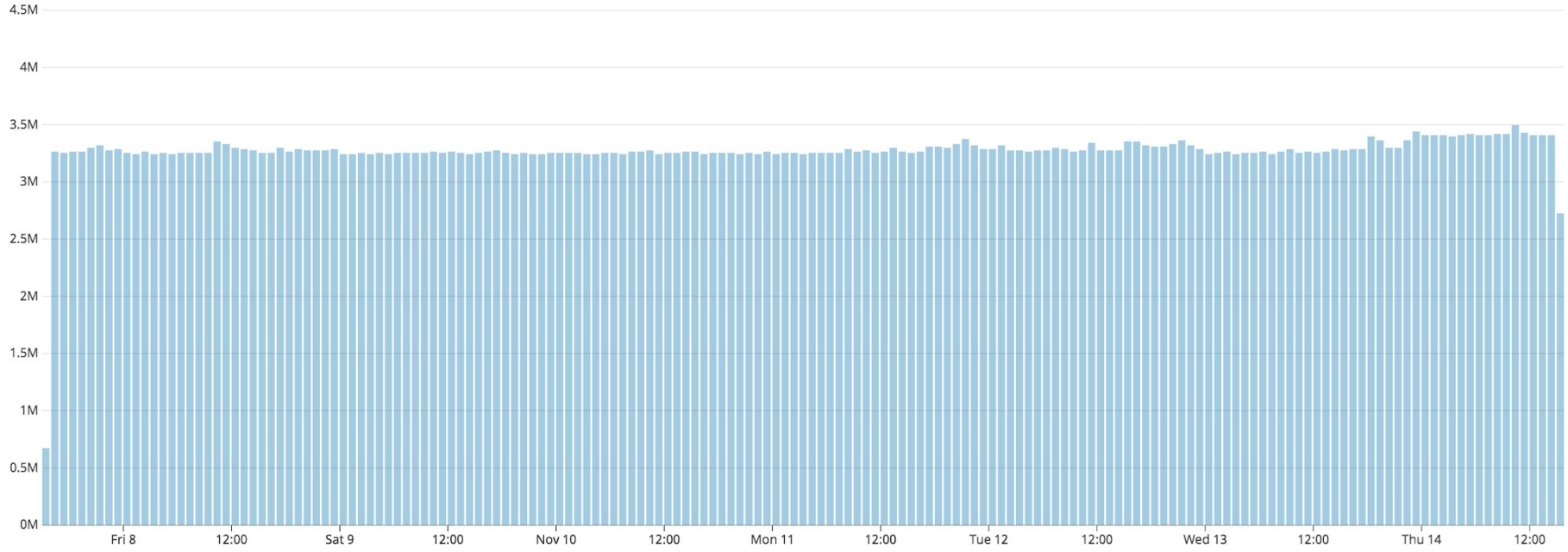
1. Background: The Kubernetes API
2. Audit Logs
3. Configuring Audit Logs
4. **10000 foot view for a large cluster**
5. Understanding Kubernetes Internals
6. Troubleshooting examples


KubeCon
CloudNativeCon
North America 2019
10000 foot view for a large cluster

# Total number of API calls



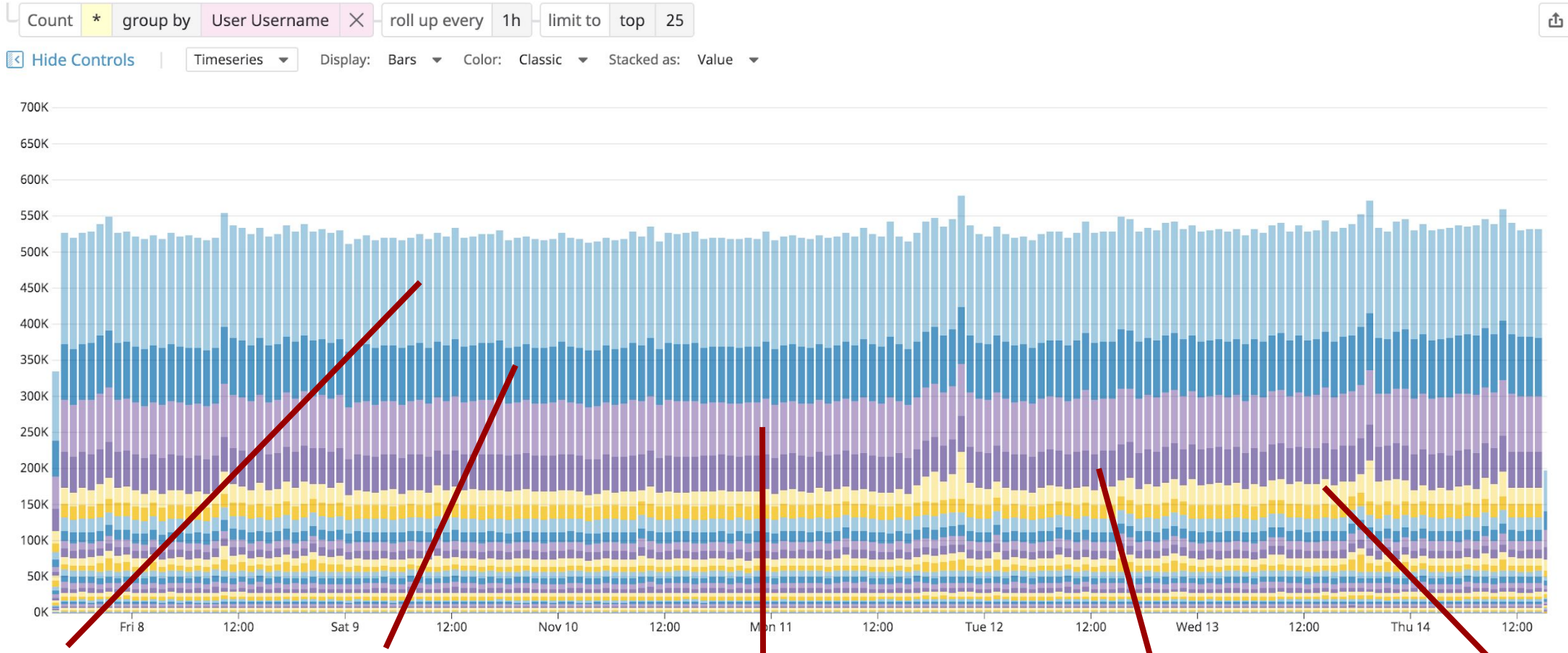
Number of audit logs per hour




**~900 calls/second on this 2500 nodes cluster**

# Top API users?






apiserver: 40 rps

kube-proxy: 20 rps

local-volume-provisioner:20 rps

cronjob-controller: 15 rps

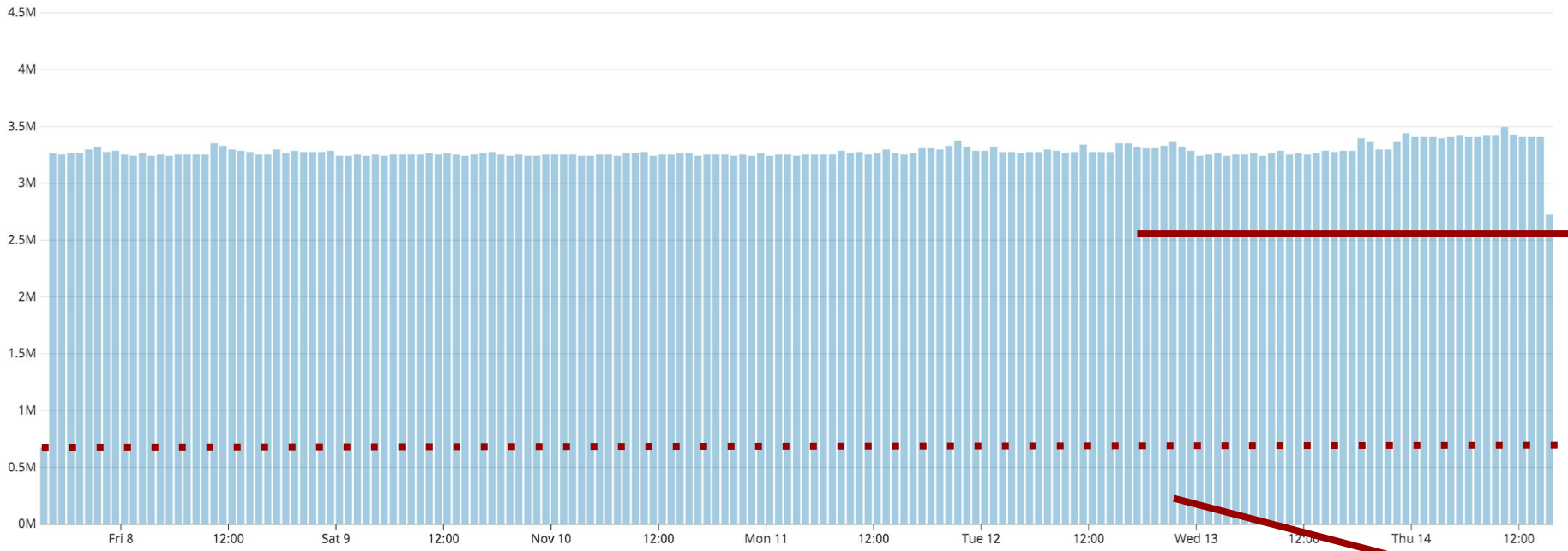
spinnaker: 5-10 rps

@lbernail @roboll_

# Top list, missing “small” users



Total number of API calls




total: ~900rps

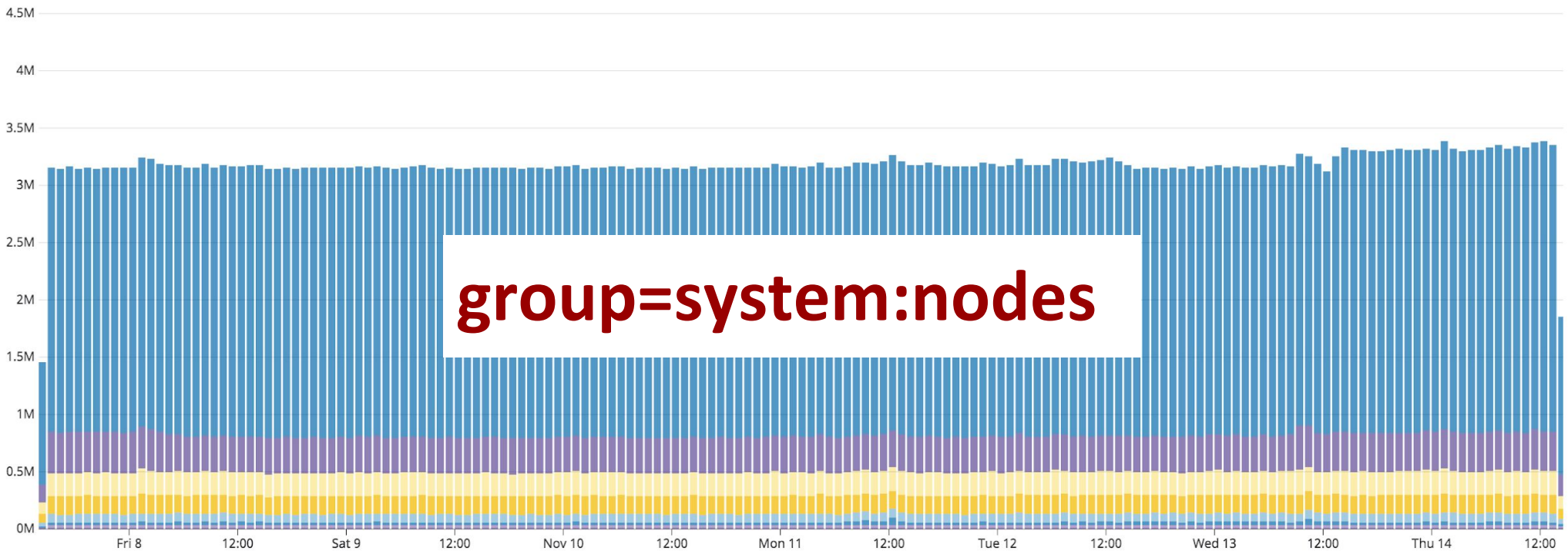
Top 25: 150 rps

**900 calls/second on this 2500 nodes cluster**
**What is doing ~80% of API calls?**

# Grouping by users is not helping



Calls by user group



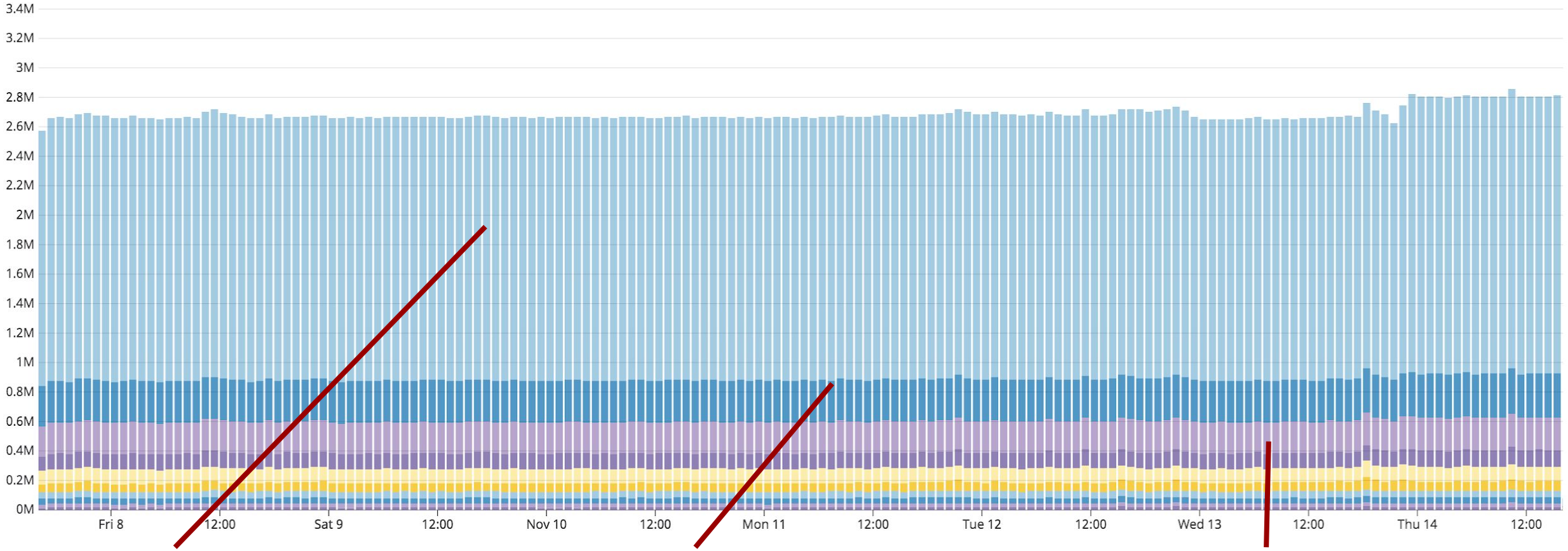

Calls from users in group “system:nodes”: 750 rps (~80% of API calls)

In this 2500-nodes cluster, this means **0.3 rps per node**!

# Why is “system:nodes” so high?



Calls from group “system:nodes” **by resource targeted**




nodes: 500 rps

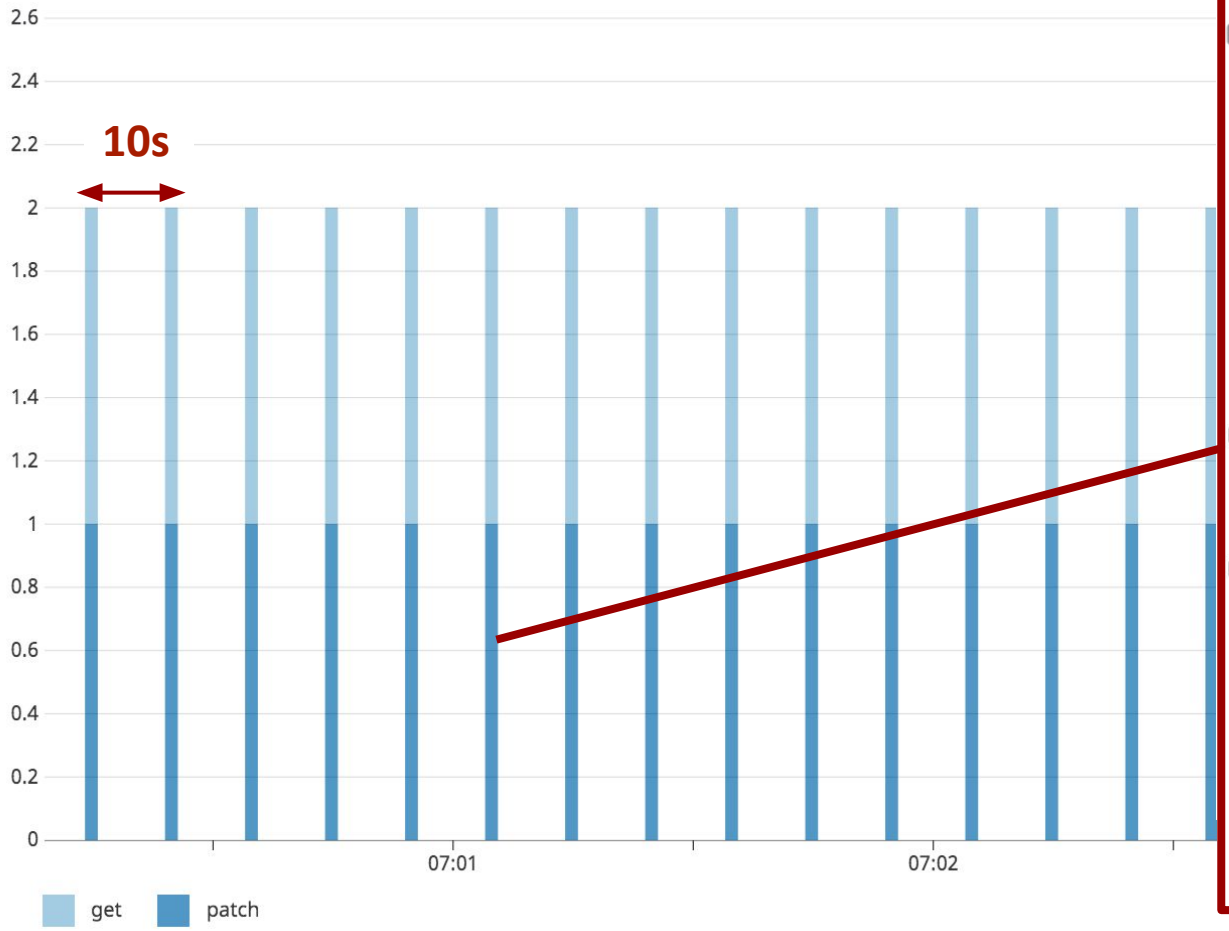
configmaps: 75 rps

secrets: 60 rps

@lbernail @roboll_

# Verbs on “node” for a kubelet





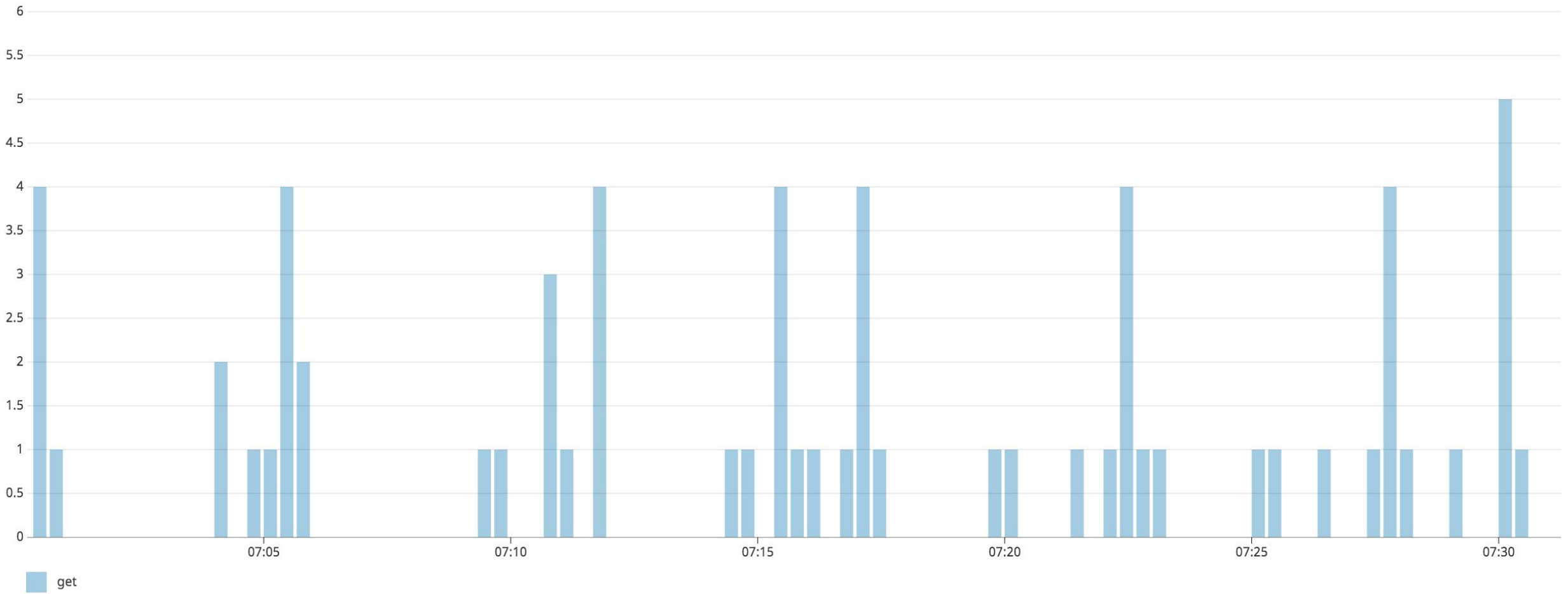


```
user {
  groups [
      system:nodes,
      system:authenticated
  ]
  username  system:node:eu1-prod-dog-app3-k8s-nodepool-vm-c602a136b569f8c0-b0j7
}
kind                    Event
level                   Metadata
metadata {
  creationTimestamp  2019-11-19T15:03:04Z
}
objectRef {
  apiVersion    v1
  name          eu1-prod-dog-app3-k8s-nodepool-vm-c602a136b569f8c0-b0j7
  resource      nodes
  subresource   status
}
verb                    patch
```


Each node update its status every 10s

# Verbs on “configmaps”

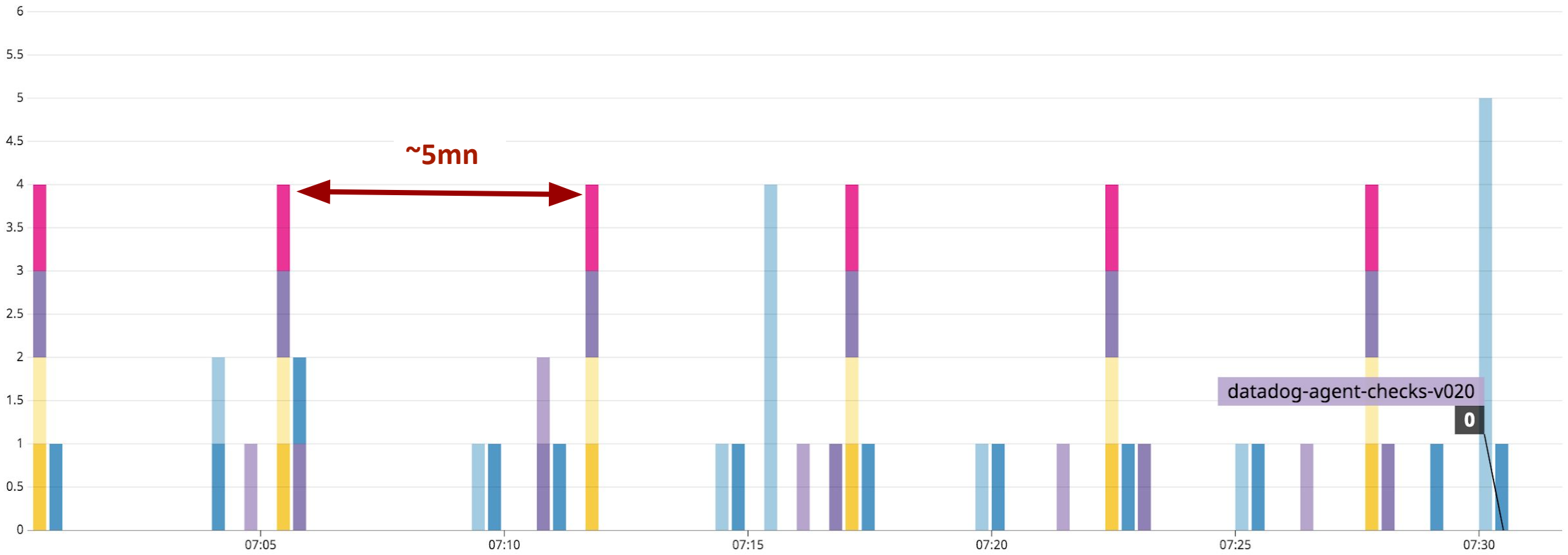





Only GETs
Regularity but not clear pattern

# Group by resource name



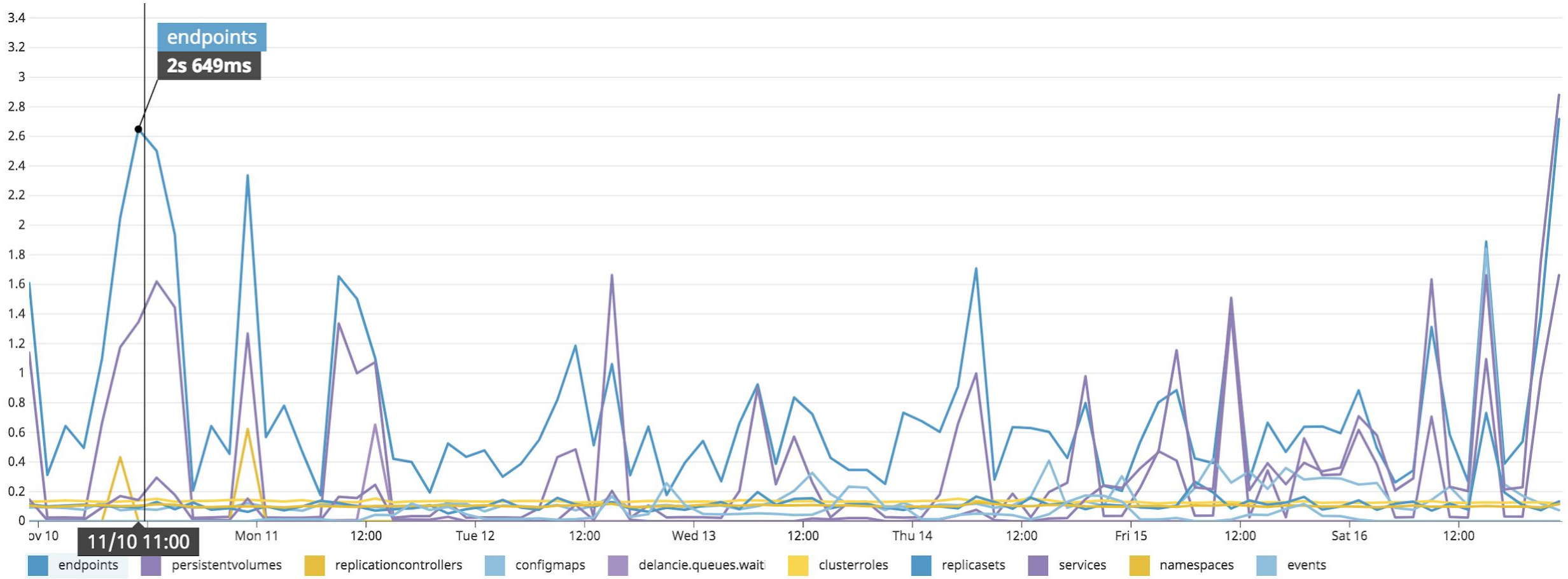



Each configmap is refreshed every ~5mn => GET call
Similar for secrets

# List latency by resource



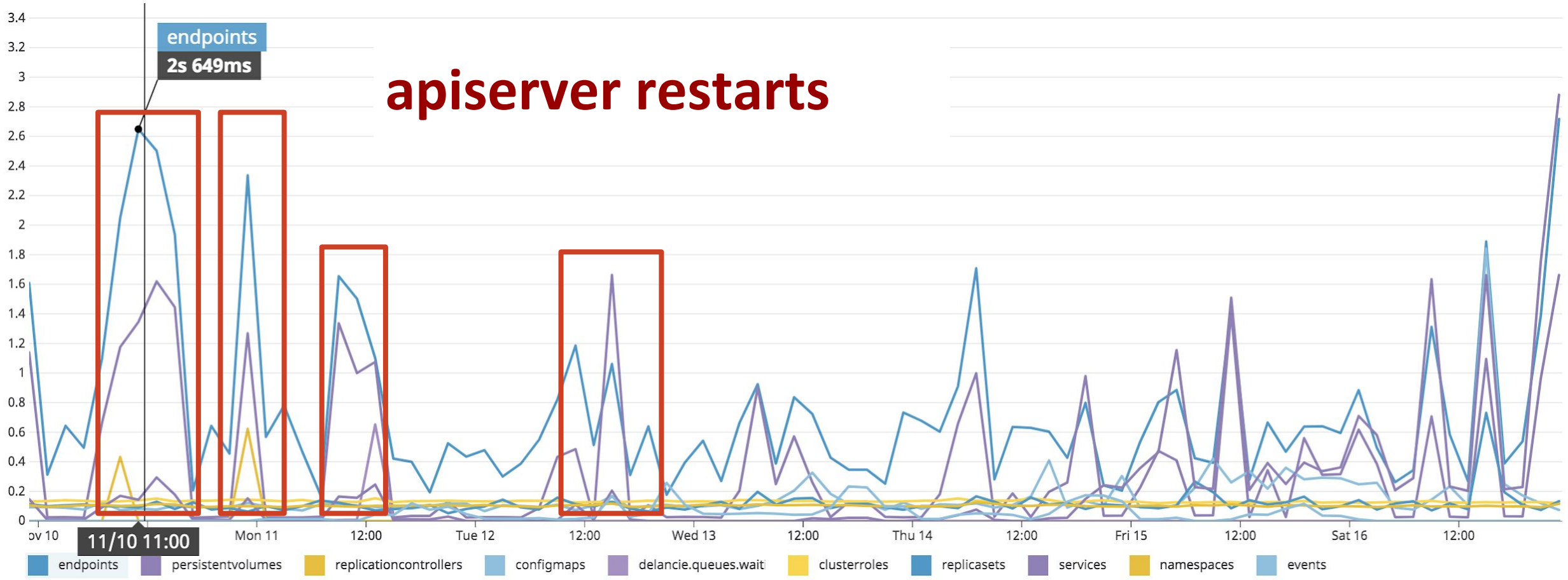
Latency for list by resource



# List latency by resource



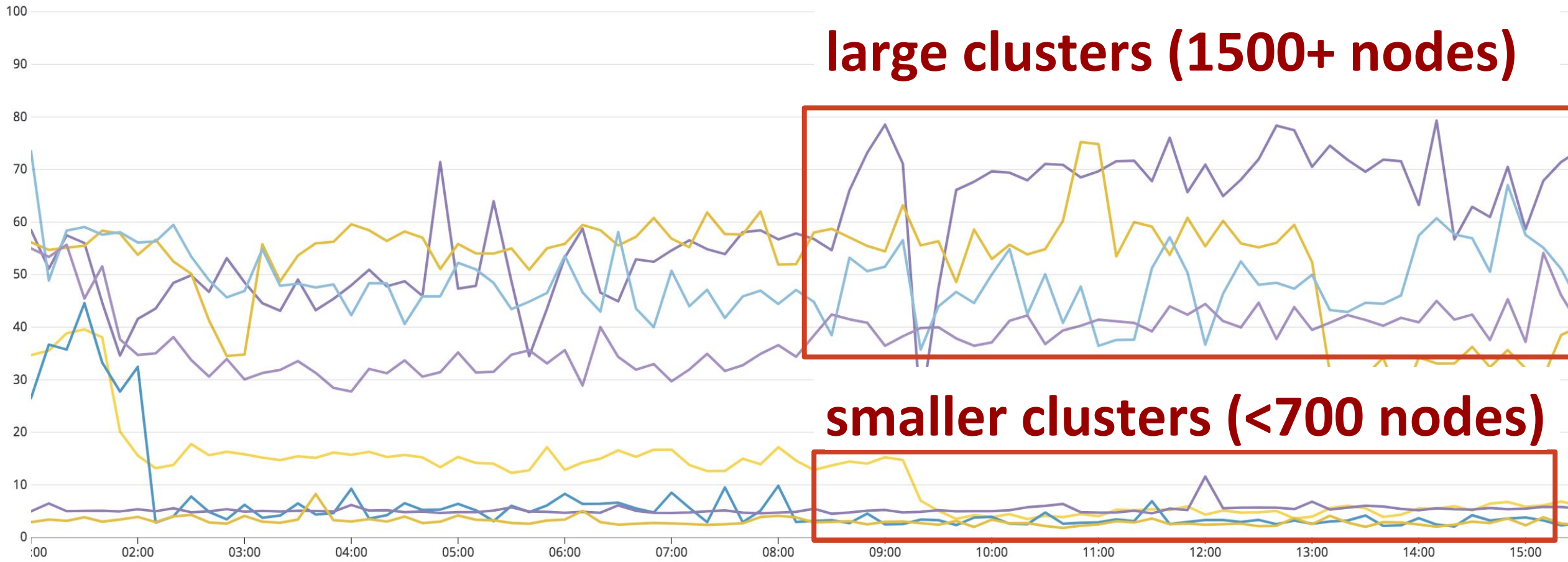
Latency for list by resource



# Compare cluster performances



Latency for get pod by cluster (ms)




@lbernail @roboll_

# Takeaways



- Biggest users are the ones running on each node
  - kubelet, daemonsets (kube-proxy)
  - A lot of effort upstream to reduce their load
  - Be extra careful of damonsets doing API calls
- Audit logs structure allow to filter and slice & dice
- Audit logs are verbose (1000 logs/s in our example)

# Outline



1. Background: The Kubernetes API
2. Audit Logs
3. Configuring Audit Logs
4. 10000 foot view for a large cluster
5. **Understanding Kubernetes Internals**
6. Troubleshooting examples


KubeCon
CloudNativeCon
North America 2019
Understanding
Kubernetes Internals

# Creating a simple deployment



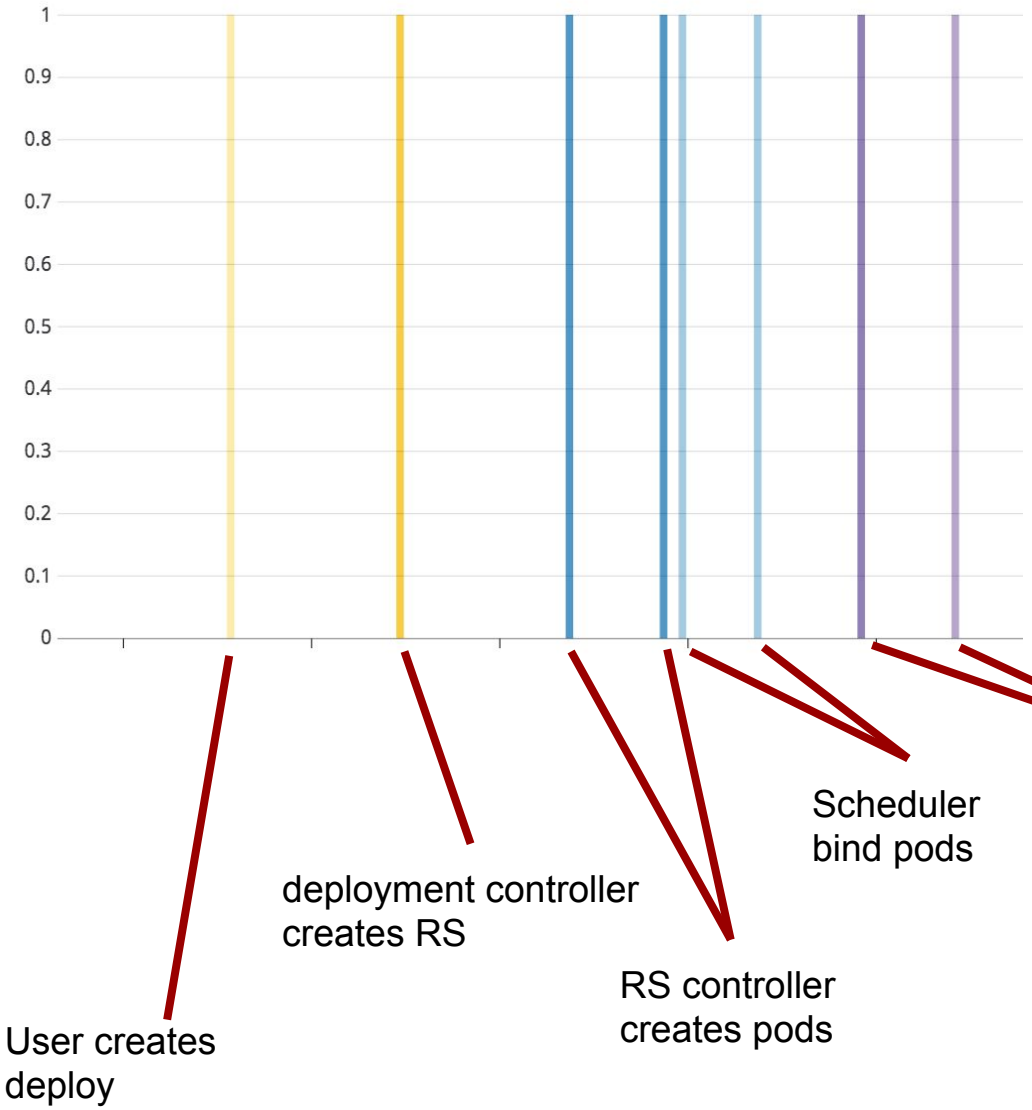
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx

spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx

    spec:
      containers:
      - name: nginx
        image: nginx
```

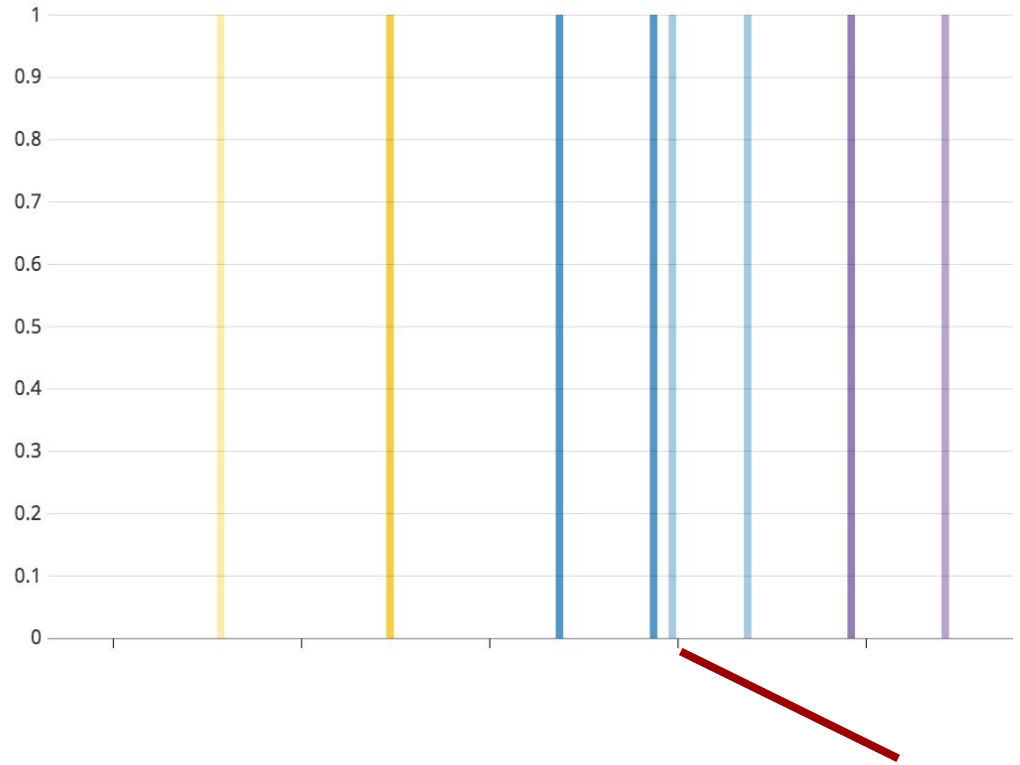
# Sequence




1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
User creates deploy
deployment controller creates RS
RS controller creates pods
Scheduler bind pods
Kubelets Update pod status

# Sequence






**Scheduler**
bind pods

**Create**

```
http {
    method       create
    status_code  201
  url_details {
      path  /api/v1/namespaces/demo/pods/nginx-68c5f9f877-29qsb/binding
  }
}
```

**Scheduler call**

**Binding**
For nginx pod

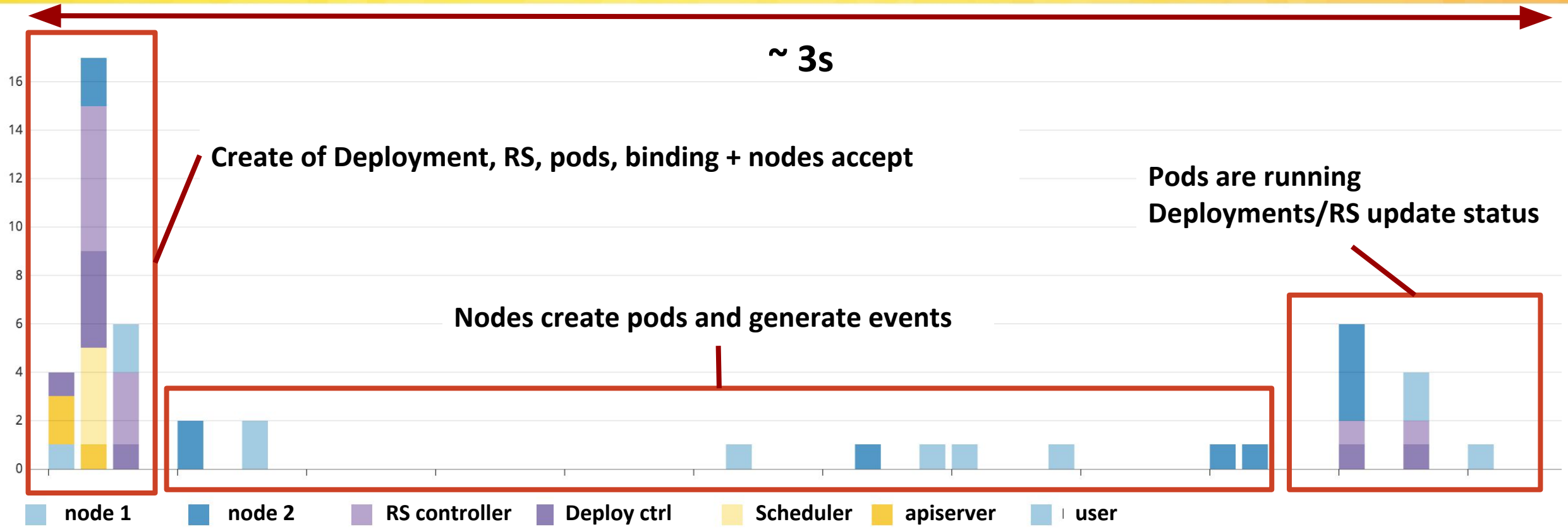
```
objectRef {
    apiVersion   v1
    name         nginx-68c5f9f877-29qsb
    namespace    demo
    resource     pods
    subresource  binding
    uid          43a0691b-07a8-11ea-8101-12d7306f3c0c
}
requestObject {
    apiVersion  v1
    kind        Binding
  metadata {
      creationTimestamp  null
      name               nginx-68c5f9f877-29qsb
      namespace          demo
      uid                43a0691b-07a8-11ea-8101-12d7306f3c0c
  }
```

**To node**
ip-10-x-y-123

```
  target {
      kind   Node
      name   ip-10-        -123.ec2.internal
  }
}
```

# Actually a lot more






## Additions

Apiserver verifies Quotas
Components also get/list
Creation of events + Update of status fields
Complete node workflow

# Node API calls




```
Nov 15 14:03:07.904 > {"duration":4000000,"objectRef":{"resource":"pods","subresource":"status","namespace":"demo","name":"nginx-68c5f9f877-d7q
Nov 15 14:03:07.900 > {"duration":2000000,"objectRef":{"resource":"pods","namespace":"demo","name":"nginx-68c5f9f877-d7qqg","apiVersion":"v1"},'
Nov 15 14:03:07.393 > {"duration":5000000,"objectRef":{"resource":"events","namespace":"demo","name":"nginx-68c5f9f877-d7qqg.15d757ab26df72e4",'
Nov 15 14:03:07.248 > {"duration":4000000,"objectRef":{"resource":"events","namespace":"demo","name":"nginx-68c5f9f877-d7qqg.15d757ab1e1587b1",'
Nov 15 14:03:07.162 > {"duration":4000000,"objectRef":{"resource":"events","namespace":"demo","name":"nginx-68c5f9f877-d7qqg.15d757ab19061b3d",'
Nov 15 14:03:06.897 > {"duration":3000000,"objectRef":{"resource":"events","namespace":"demo","name":"nginx-68c5f9f877-d7qqg.15d757ab09445624",'
Nov 15 14:03:06.111 > {"duration":4000000,"objectRef":{"resource":"events","namespace":"demo","name":"nginx-68c5f9f877-d7qqg.15d757aada77ac19",'
Nov 15 14:03:06.100 > {"duration":2000000,"objectRef":{"resource":"secrets","namespace":"demo","name":"default-token-dqmzw","apiVersion":"v1"},
Nov 15 14:03:05.908 > {"duration":3000000,"objectRef":{"resource":"pods","subresource":"status","namespace":"demo","name":"nginx-68c5f9f877-d7q
Nov 15 14:03:05.900 > {"duration":6000000,"objectRef":{"resource":"pods","namespace":"demo","name":"nginx-68c5f9f877-d7qqg","apiVersion":"v1"},
```


## Initial calls

Get pod
Update pod/status: ContainerCreating
Get service account token

@lbernail @roboll_

# Node API calls




```
Nov 15 14:03:07.904 > {"duration":4000000,"objectRef":{"resource":"pods","subresource":"status","namespace":"demo","name":"nginx-68c5f9f877-d7q
Nov 15 14:03:07.900 > {"duration":2000000,"objectRef":{"resource":"pods","namespace":"demo","name":"nginx-68c5f9f877-d7qqg","apiVersion":"v1"},'
Nov 15 14:03:07.393 > {"duration":5000000,"objectRef":{"resource":"events","namespace":"demo","name":"nginx-68c5f9f877-d7qqg.15d757ab26df72e4",'
Nov 15 14:03:07.248 > {"duration":4000000,"objectRef":{"resource":"events","namespace":"demo","name":"nginx-68c5f9f877-d7qqg.15d757ab1e1587b1",'
Nov 15 14:03:07.162 > {"duration":4000000,"objectRef":{"resource":"events","namespace":"demo","name":"nginx-68c5f9f877-d7qqg.15d757ab19061b3d",'
Nov 15 14:03:06.897 > {"duration":3000000,"objectRef":{"resource":"events","namespace":"demo","name":"nginx-68c5f9f877-d7qqg.15d757ab09445624",'
Nov 15 14:03:06.111 > {"duration":4000000,"objectRef":{"resource":"events","namespace":"demo","name":"nginx-68c5f9f877-d7qqg.15d757aada77ac19",'
Nov 15 14:03:06.100 > {"duration":2000000,"objectRef":{"resource":"secrets","namespace":"demo","name":"default-token-dqmzw","apiVersion":"v1"},'
Nov 15 14:03:05.908 > {"duration":3000000,"objectRef":{"resource":"pods","subresource":"status","namespace":"demo","name":"nginx-68c5f9f877-d7q
Nov 15 14:03:05.900 > {"duration":6000000,"objectRef":{"resource":"pods","namespace":"demo","name":"nginx-68c5f9f877-d7qqg","apiVersion":"v1"},'
```


## Create events to show progression

MountVolume.SetUp succeeded for volume "default-token-dqmzw"
pulling image "nginx"
Successfully pulled image "nginx"
Created container
Started container

@lbernail @roboll_

# Node API calls


## Finalize pod creation

Get pod

Update container status to “Running”

# Takeaways



- A lot of interactions between kube components
- Audit logs give a great understanding of this!
- Events are spiky and get generate a lot of logs
- Events have a 1h default TTL, but stay in audit logs

Let’s identify some problems using audit logs

# Outline



1. Background: The Kubernetes API
2. Audit Logs
3. Configuring Audit Logs
4. 10000 foot view for a large cluster
5. Understanding Kubernetes Internals
6. **Troubleshooting examples**


# Troubleshooting examples

# Troubleshooting



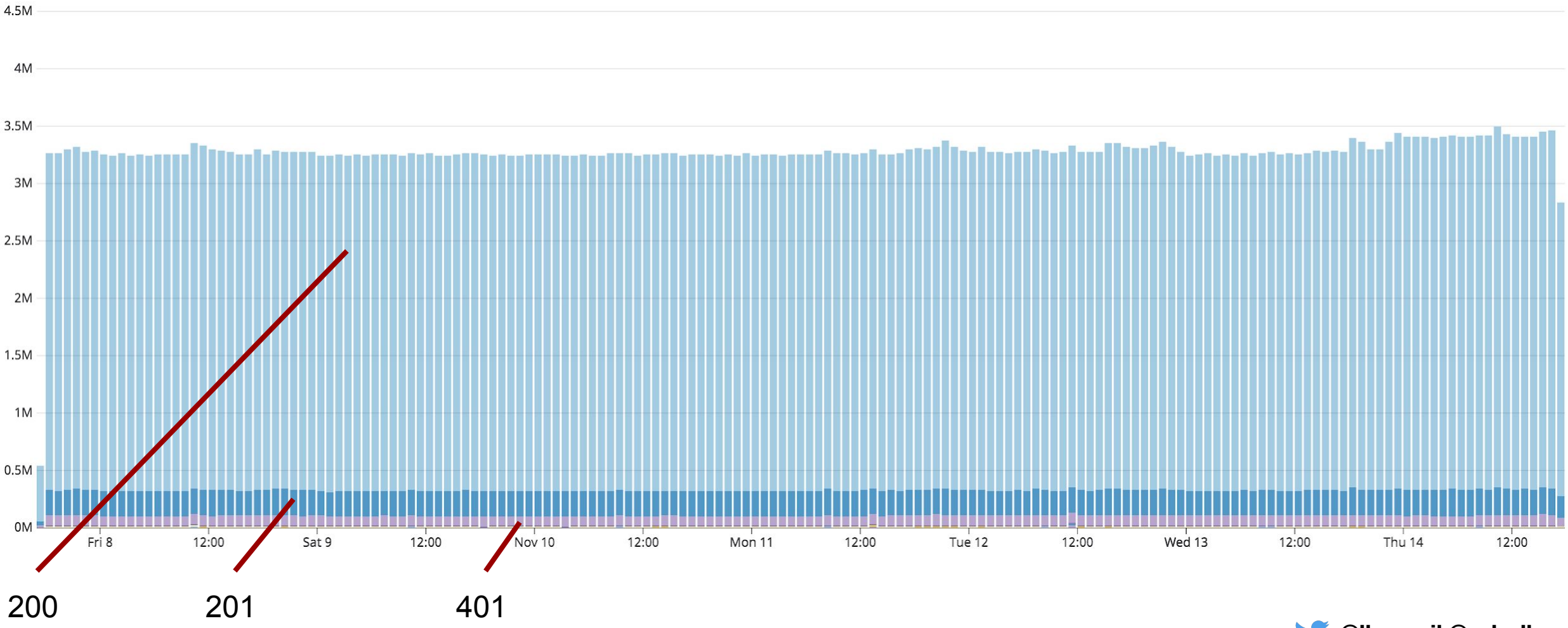
- Understand what happened
  - “Why was a resource deleted?”
- Debug performance regressions/improve performances
  - “Which application is responsible for so many calls?”

- Also, identify issues by looking at HTTP status codes

# Status codes



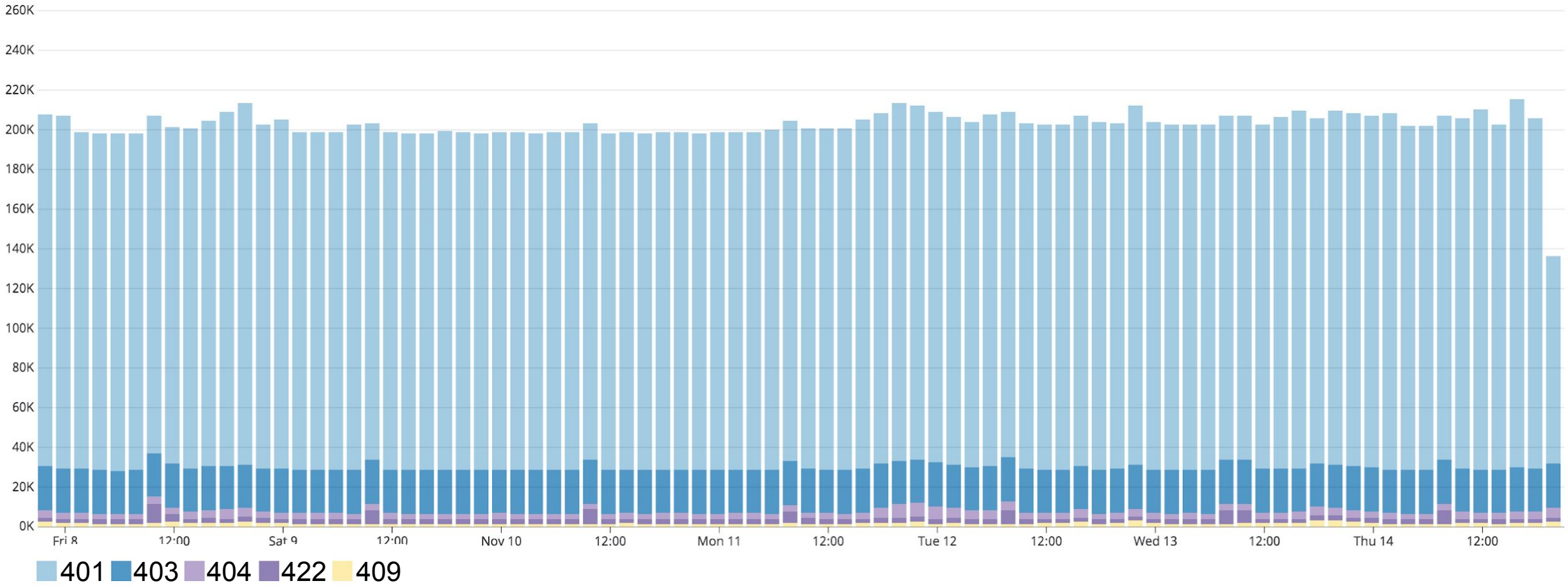
Calls by status code



# 4xx only



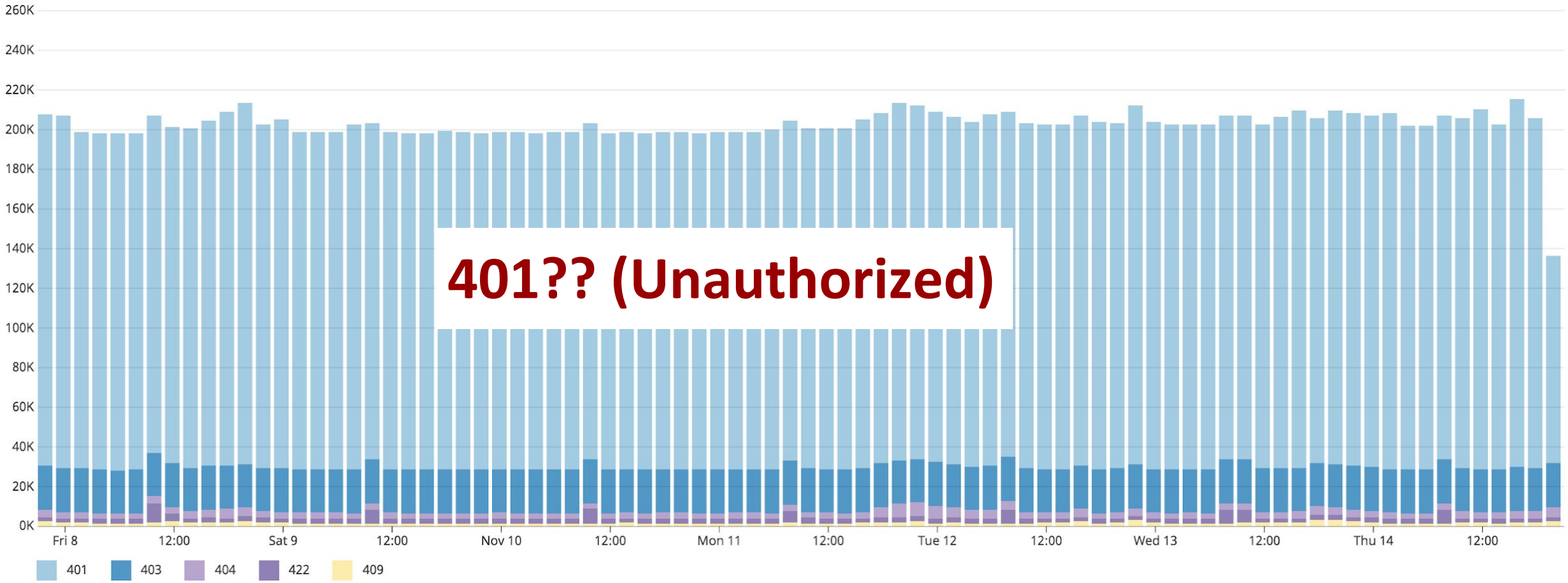
4xx by status code



# 4xx only



4xx by status code

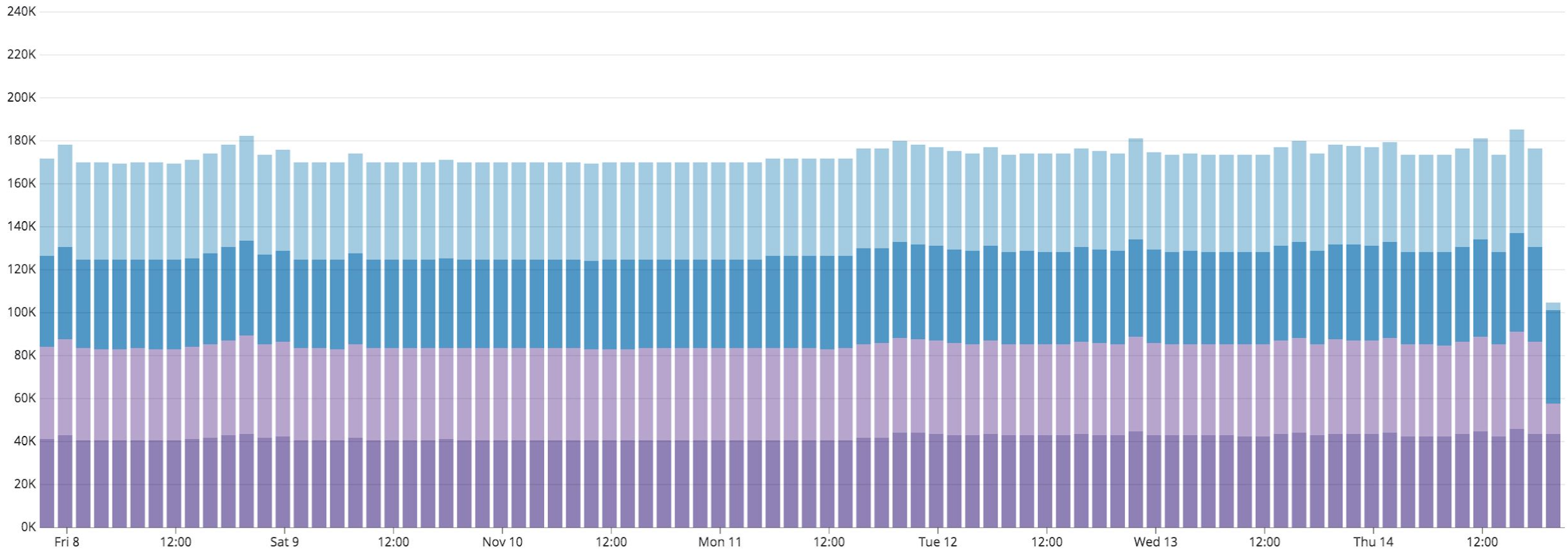



**401?? (Unauthorized)**

# Analyzing 401s



401s by source IP

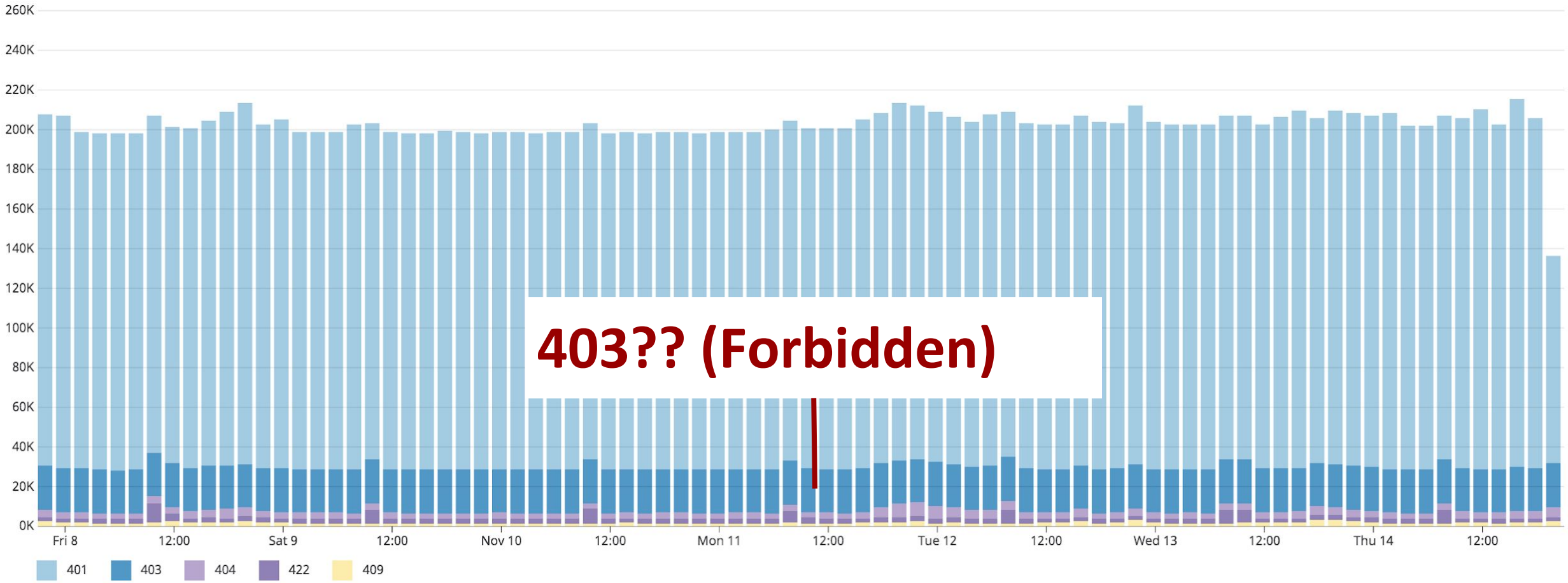



4 nodes only, turns out they had expired certificates

# 4xx only



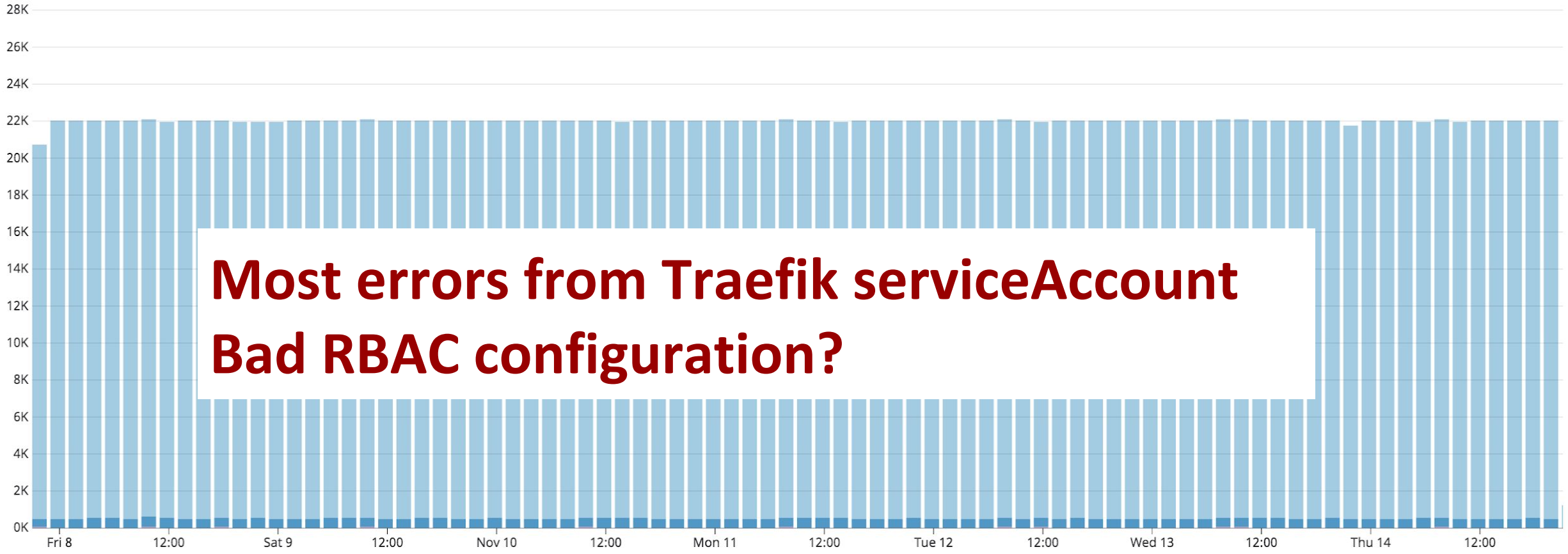
4xx by status code



# Analyzing 403s



403s by user



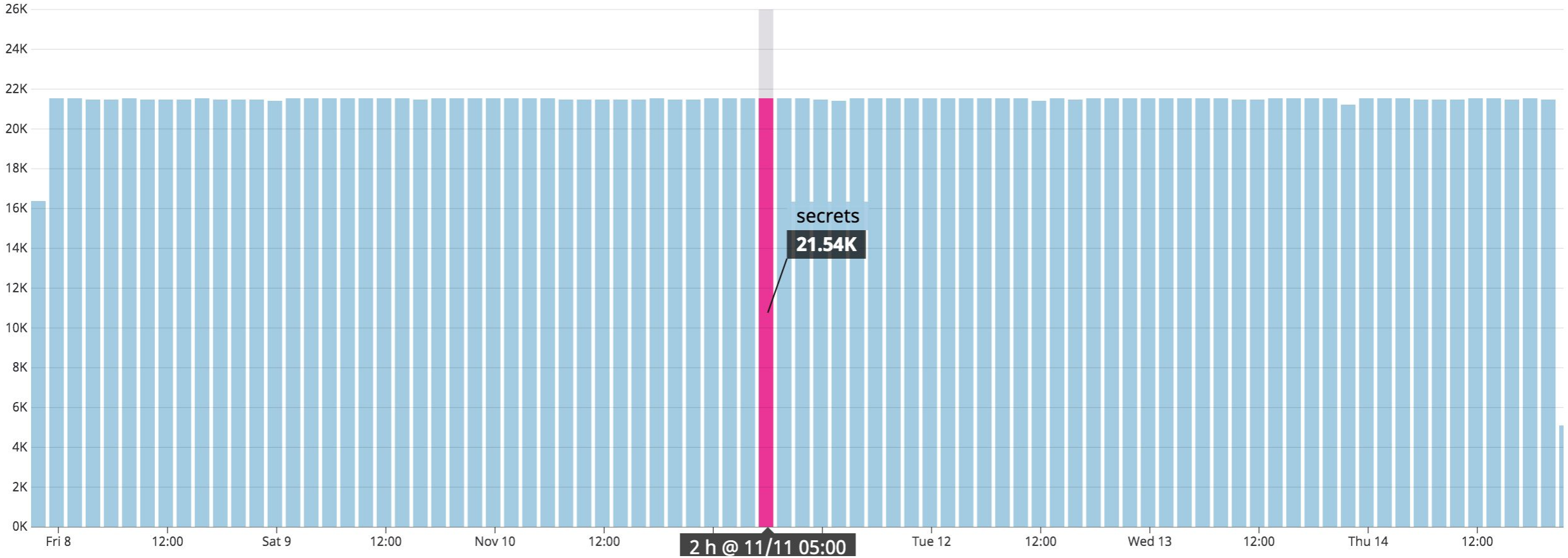

**Most errors from Traefik serviceAccount**

**Bad RBAC configuration?**

# Analyzing 403s for this user



403s for Traefik serviceAccount by resource

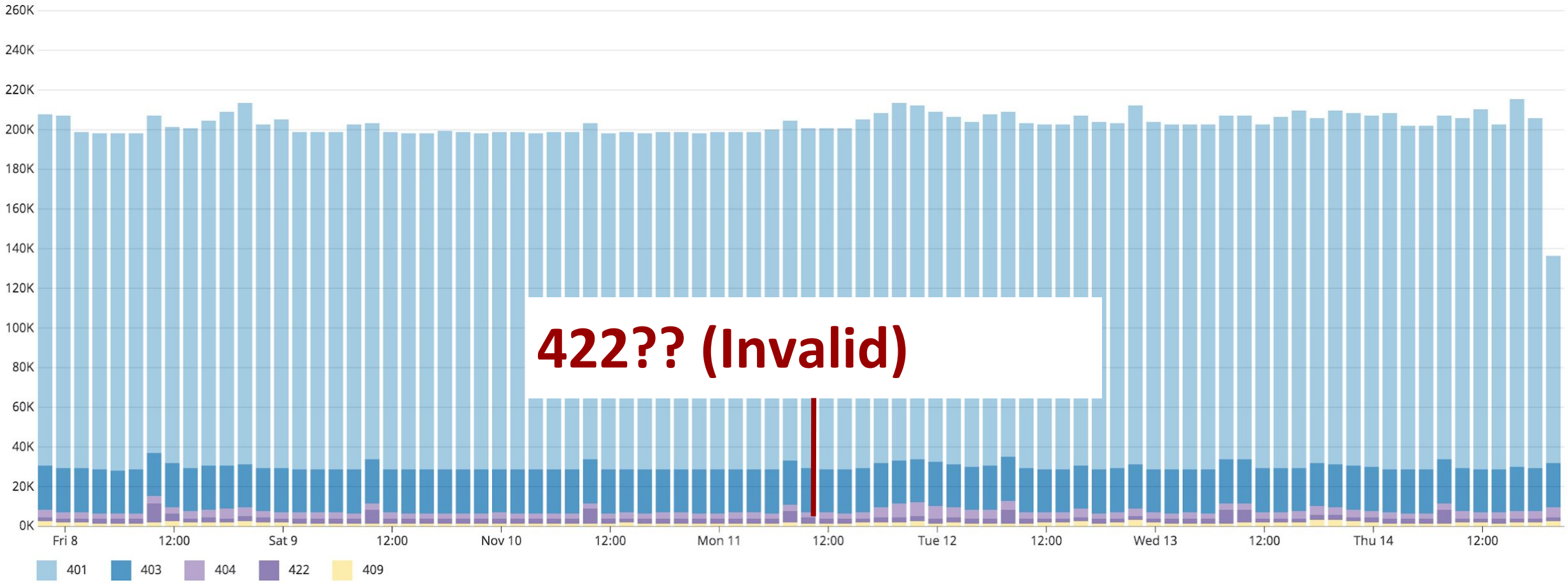



We use Traefik without Kubernetes secrets but it still tries to list them

# 4xx only



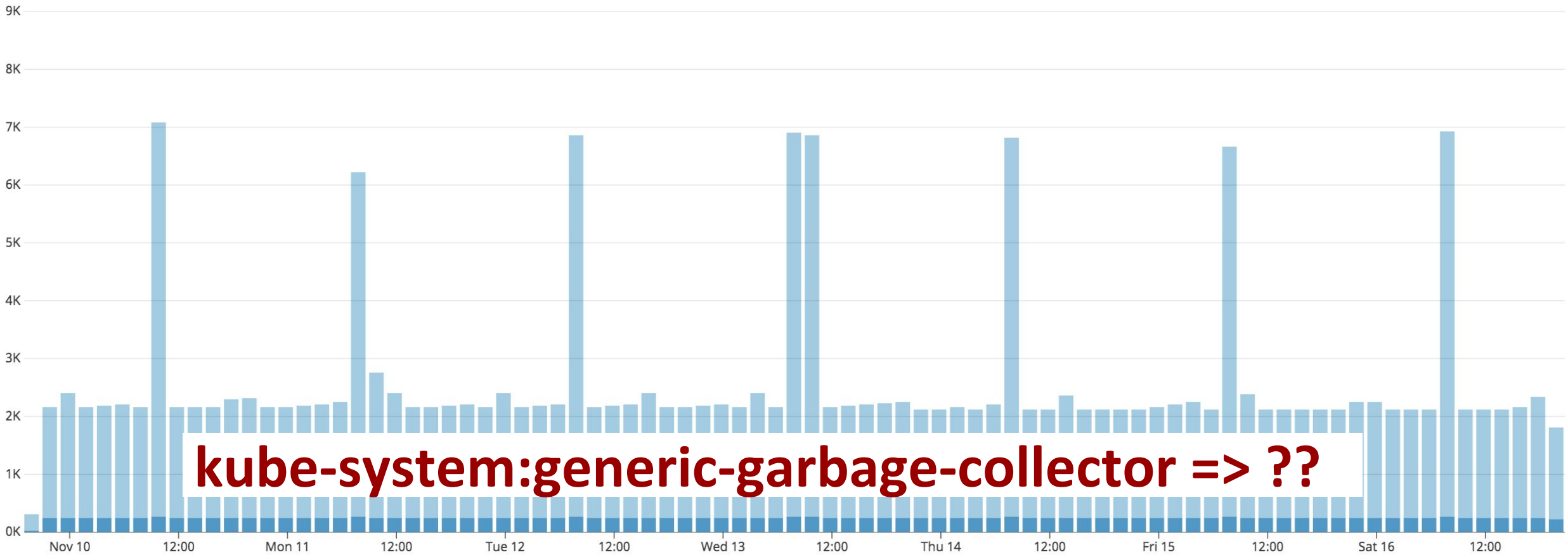
4xx by status code



# What about 422?



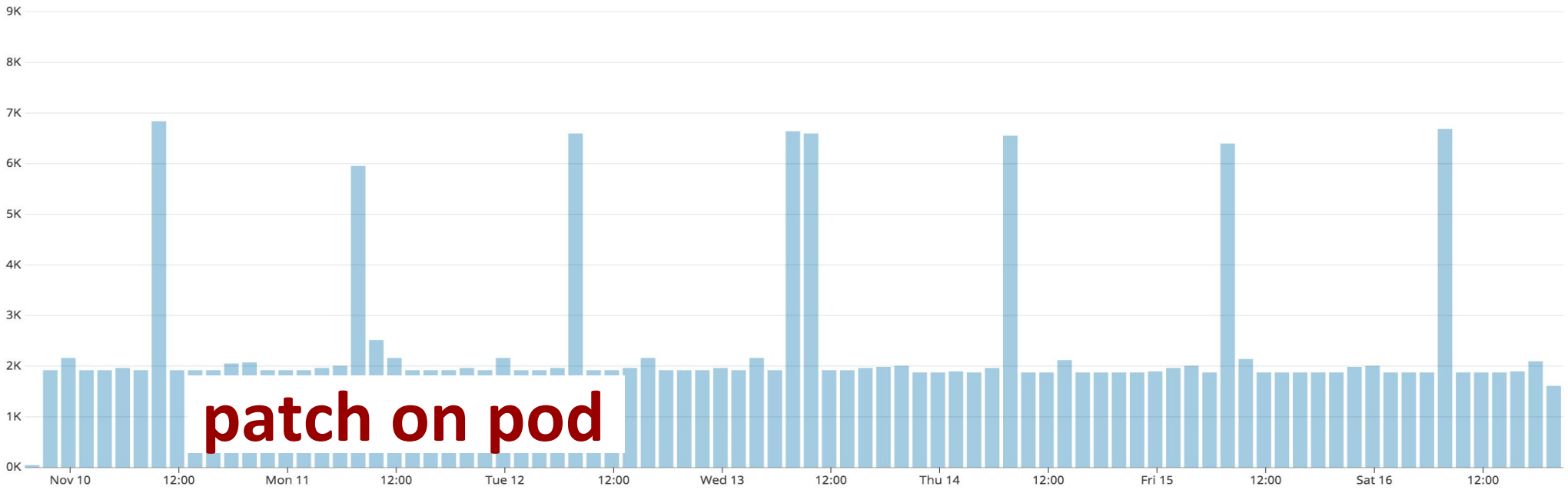
422 by user



# What is failing?



generic-garbage-collector by verb / resource





```
requestObject {
  metadata {
    ownerReferences [
      {"uid":"b430d461-b876-11e9-82b5-42010a50e236","$patch":"delete"}
    ]
```

# What is happening?



- Pods are in “Evicted” status and must be kept
- Controlling RS has been deleted and pods should be orphaned
- Garbage collector fails to orphan them (remove ownerRef)
- ReplicaSet has been Terminating for 2 months...

- Root cause: mutating webhook
  - We modify the pod spec at creation
  - Mutating webhook is registered on pods for CREATE/UPDATE
  - We modify immutable fields
  - Garbage collector patch triggers this...

# Takeaways



- Looking at calls triggering HTTP errors help find issues
  - Misconfigured RBAC (403)
  - Applications doing calls they shouldn’t (403)
  - Expired certificates (401)
  - Other weird things

# Conclusion

# Conclusion



- Audit logs can be incredibly valuable
  - Low-level understanding of Kubernetes
  - Detection of misconfigurations
  - Troubleshooting of issues
  - Identify performance issues

- Taking advantage of them require some effort
  - Policies are not easy to get right
  - They are verbose and require a tool to analyze them

# We're hiring!

**Visit our Kubecon booth**
**or** https://www.datadoghq.com/careers/

**Or contact us directly:**


**@lbernail** laurent@datadoghq.com
**@roboll_** roboll@datadoghq.com


KubeCon | CloudNativeCon
North America 2019

# Thank you