KubeCon | CloudNativeCon

North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

20221011-2466036-7993067

Preventing Controller Sprawl From Taking Down Your Cluster

When a Scalable Pattern Stops Being Scalable

Madhu C.S., Robinhood Markets

Agenda

- Quick overview of control loops and associated Kubernetes technologies
- Two case studies of when we either
  - Went too far
  - Or when we hit the limits of this pattern
- Best practices for working with this pattern

Kubernetes philosophy

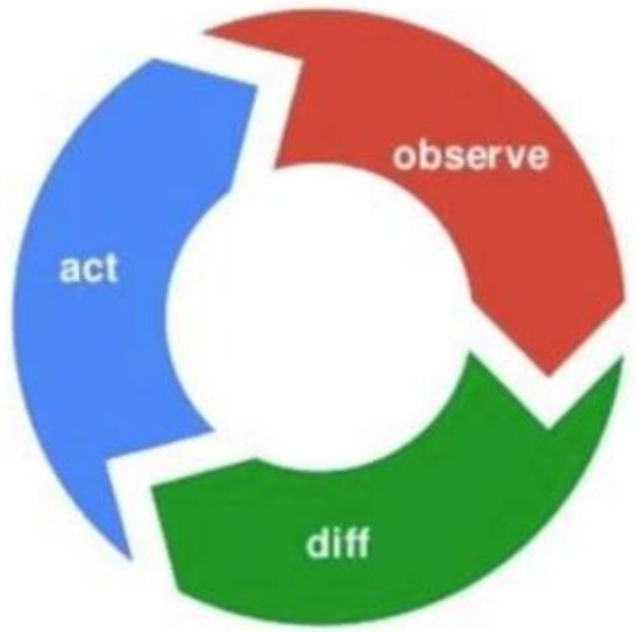
Forever reconciliation loop:

- Observe
- Diff
- Reconcile

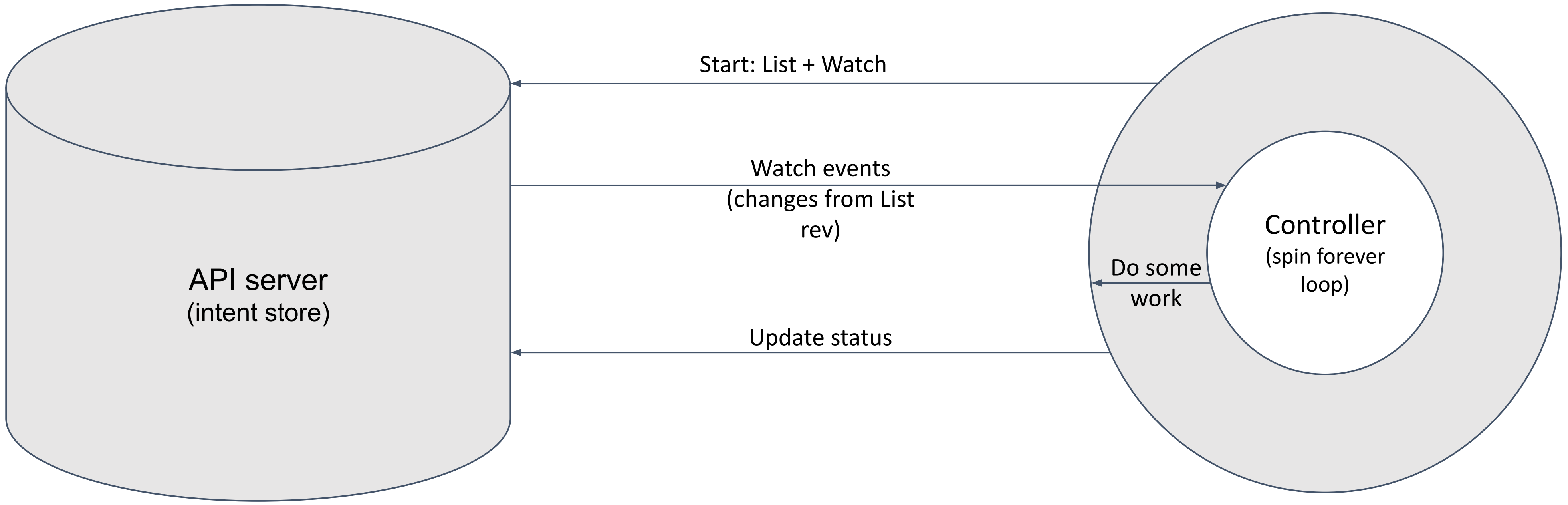
https://kubernetes.io/docs/concepts/architecture/controller/

Example:

- kubelet watches the API server to see if it should run new pods.
- If any pods get assigned to the node and those pod aren't running:
    - kubelet uses the container runtime on the host to start those pods



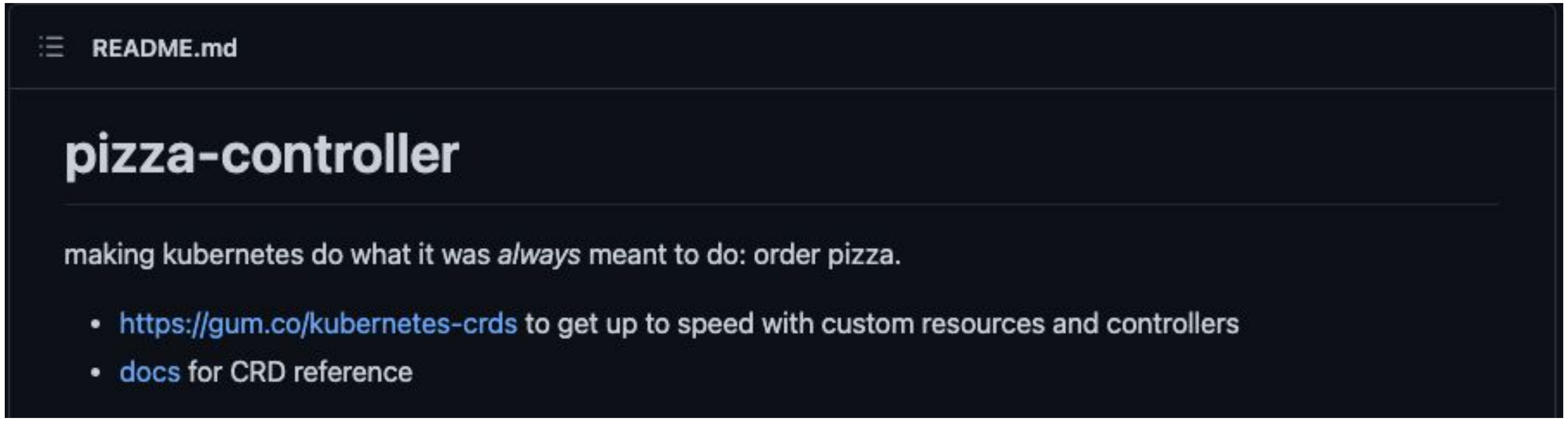
Kubernetes controllers


Start: List + Watch
Watch events (changes from List rev)
Update status
API server (intent store)
Do some work
Controller (spin forever loop)

➔ Is it expensive to replicate resources across the entire cluster?
- Well, yes and no.
- On the one hand, you are duplicating data
- O(N) =~ 100s, 1k or 10k+

Example: Pizza controller

Hilarious example: https://github.com/cirocosta/pizza-controller

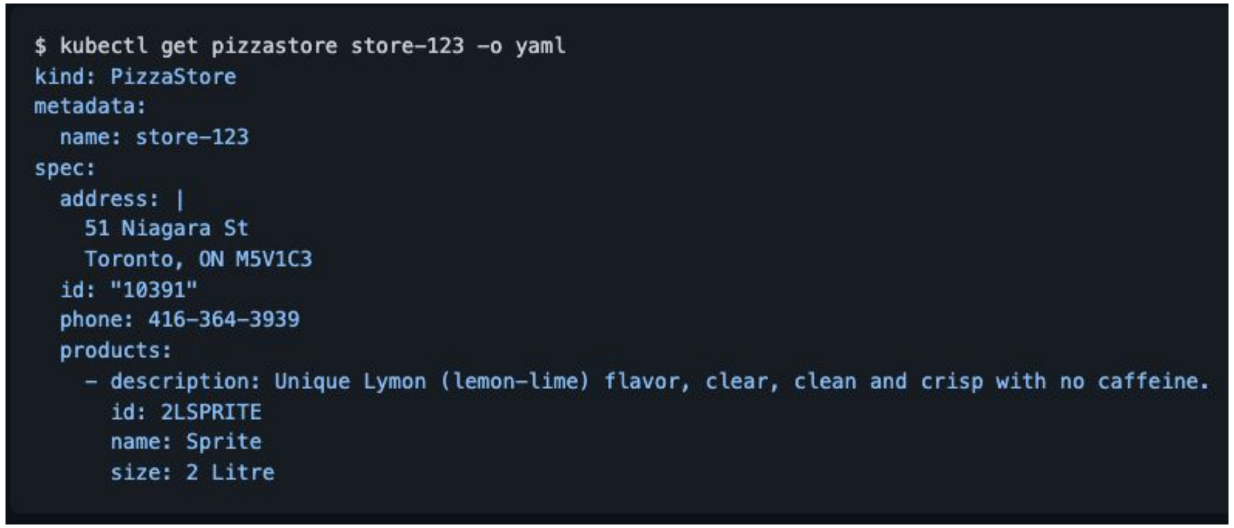


README.md

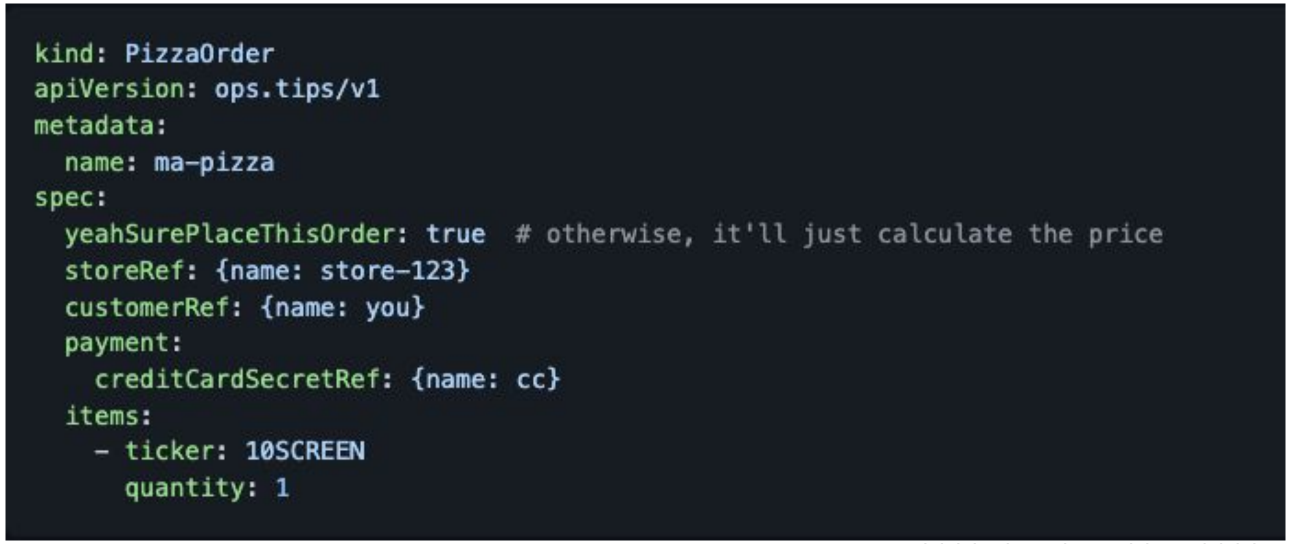
pizza-controller

making kubernetes do what it was *always* meant to do: order pizza.

- https://gum.co/kubernetes-crds to get up to speed with custom resources and controllers
- docs for CRD reference



```
$ kubectl get pizzastore store-123 -o yaml
kind: PizzaStore
metadata:
  name: store-123
spec:
  address: |
    51 Niagara St
    Toronto, ON M5V1C3
  id: "10391"
  phone: 416-364-3939
  products:
    - description: Unique Lymon (lemon-lime) flavor, clear, clean and crisp with no caffeine.
      id: 2LSPRITE
      name: Sprite
      size: 2 Litre
```



```
kind: PizzaOrder
apiVersion: ops.tips/v1
metadata:
  name: ma-pizza
spec:
  yeahSurePlaceThisOrder: true  # otherwise, it'll just calculate the price
  storeRef: {name: store-123}
  customerRef: {name: you}
  payment:
    creditCardSecretRef: {name: cc}
  items:
    - ticker: 10SCREEN
      quantity: 1
```

20221011-2466036-7993067

Case Study: # 1

Dueling wikipedia edits

And the poor people who subscribe to the RSS feed

How did it all start?

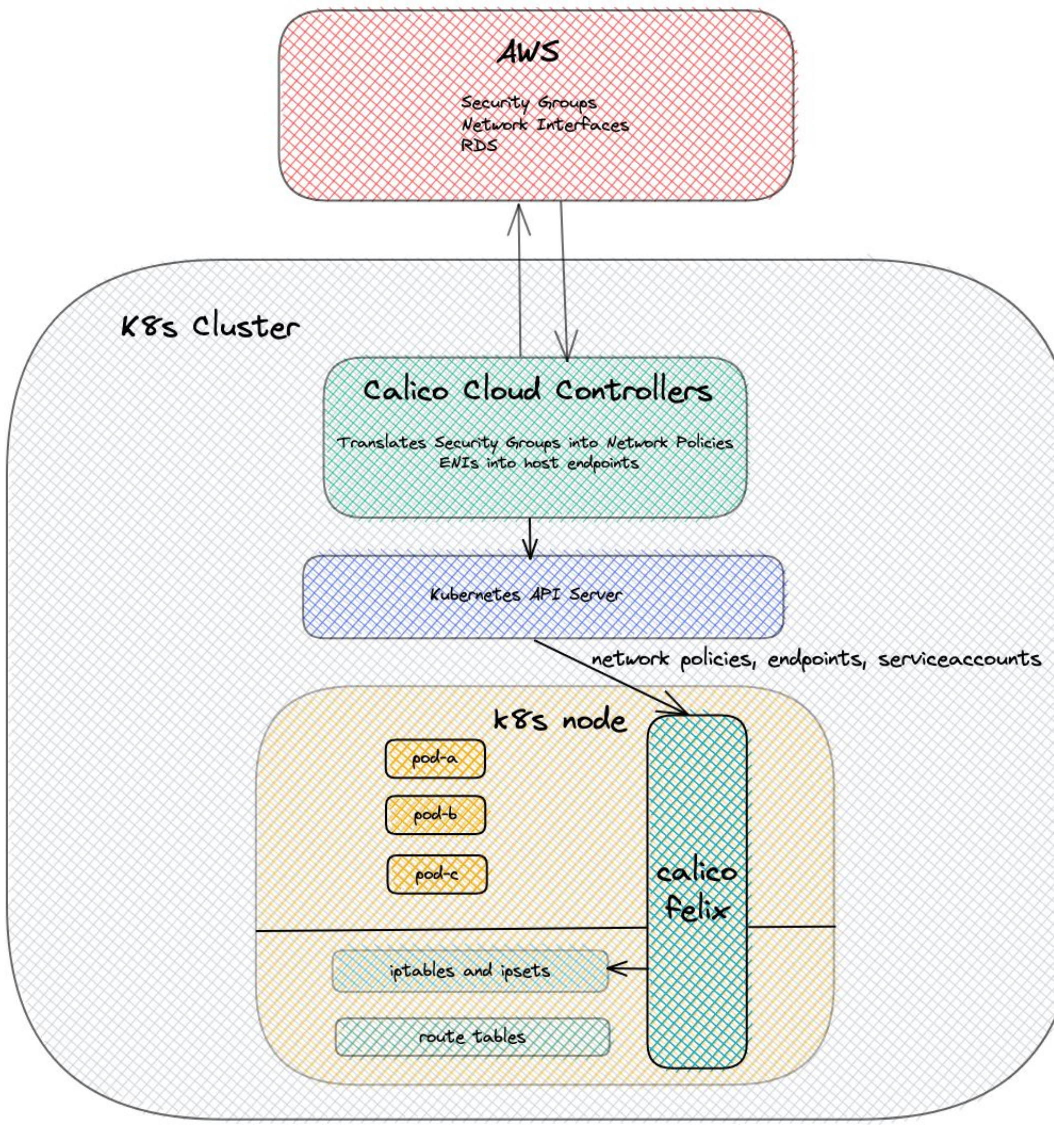
- Got some alerts (and teams complaining): **CronJobs could no longer resolve DNS**

- Anyone who has worked in infrastructure before recognizes this sort of page
    - What is it about CronJob workloads that make DNS resolution fail?
    - Well, it makes no sense

It’s the network, stupid

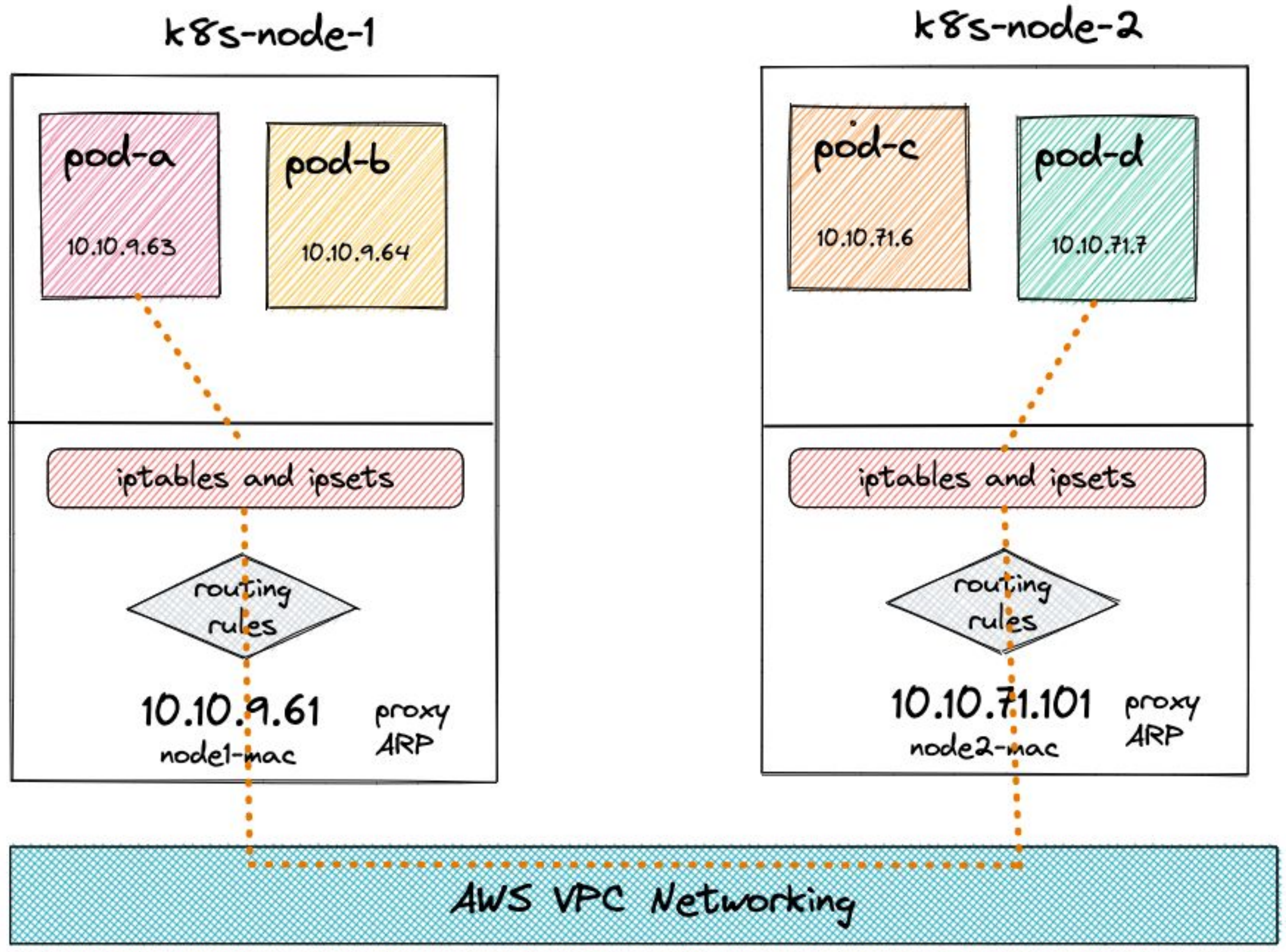
- Quickly discovered why it was CronJobs
- Well it was any new workload.
- Any new pod would have a total network blackout for a few minutes after starting up.
- Blackout time longer than the readiness probe for most pods, so they would die
- Running pods continued to work - the data plane was healthy, so we didn’t have any customer impact.

Calico for L4 segmentation

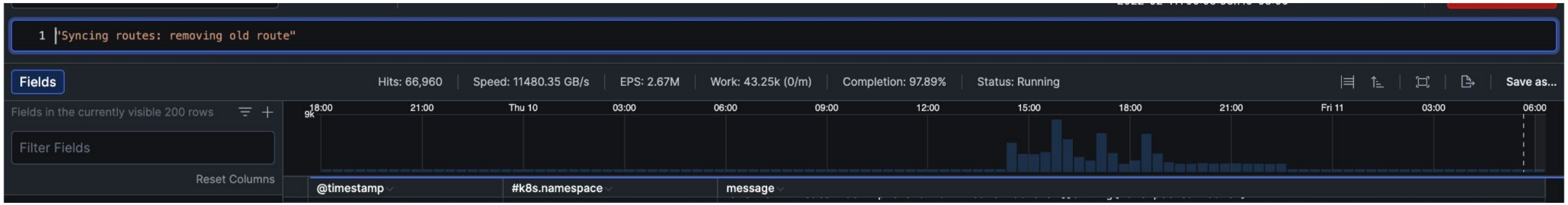


- Network Segmentation solution built for Robinhood Kubernetes
- Built on top of open source Calico
- Supports security group integrations
- Federated network policies across Kubernetes clusters

How does it all work?



Routing problem?

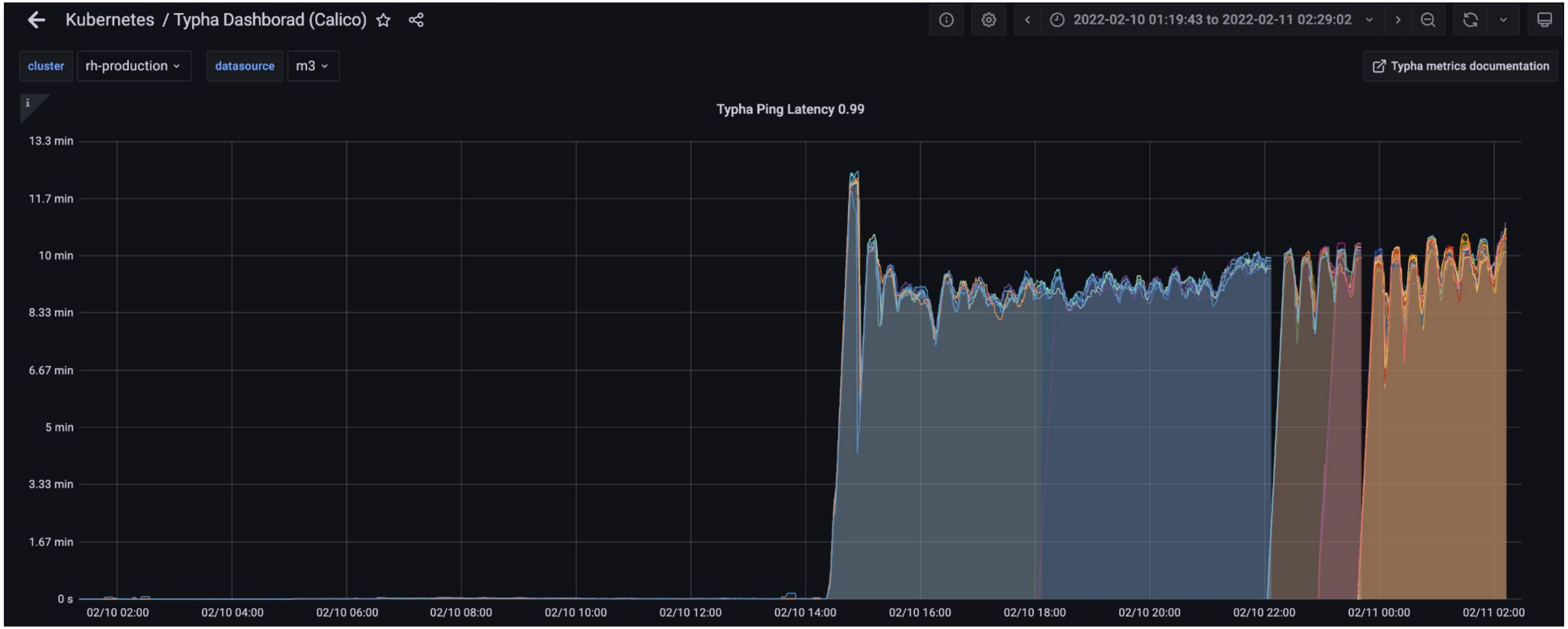


```
"Syncing routes: removing old route"
```

Fields | Hits: 66,960 | Speed: 11480.35 GB/s | EPS: 2.67M | Work: 43.25k (0/m) | Completion: 97.89% | Status: Running | Save as...

- Trigger still unclear though!
- We put some mitigations in place to ensure we could continue serving our customers
- And went on a root causing expedition!

High ping latencies

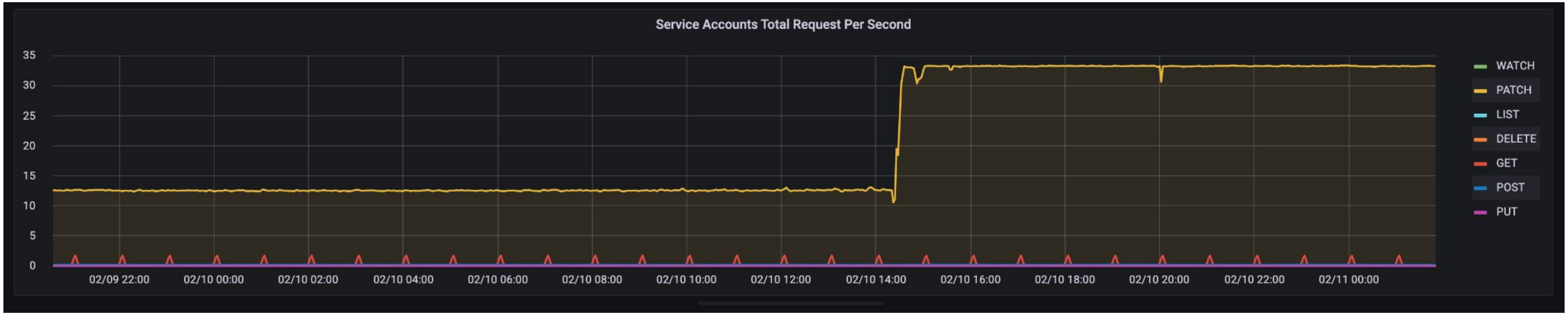


- Typha (Calico's cache + connection pooler) was dropping connections
- Typha metrics showed high ping latencies

Routing problem?

- Everything pointing to Calico not being able to keep up with something
- Not sure what
- We handled much higher load during previous black swan events
- Not enough pod churn or security group changes

A-HA moment!



- Sharp increase in patches(updates) to Service Accounts in K8s API
- Timeline correlates to when the incident started
- Seen in all prod clusters
- Potential faulty change introduced to our in-house service development framework, called Archetype

Battle of the controllers

- Each backend service that uses Archetype can have multiple components
    - Eg. different server instances, worker daemons, etc.

- A controller change in Archetype to add annotations to ServiceAccounts that belong to a component
    - Annotation contained a computed signature of the owning component

- But at the time, more than one component could share a ServiceAccount

- The controller ended up “fighting” over the signature - generating a large number of ServiceAccount update operations non-stop.

Why did this affect Calico?

- Kubernetes and Calico support ServiceAccount-based NetworkPolicies

- Calico watches for mutations to ServiceAccounts

- Happy state: Number of mutations once in a while shouldn't be overwhelming

- In our unhappy state:
    - Calico's in-memory worker queue (controller) got so backed up processing these updates
    - Starved the important updates - new Pod creation
    - Not knowing about the new pods, Calico **removed the routes** it wouldn't recognize!

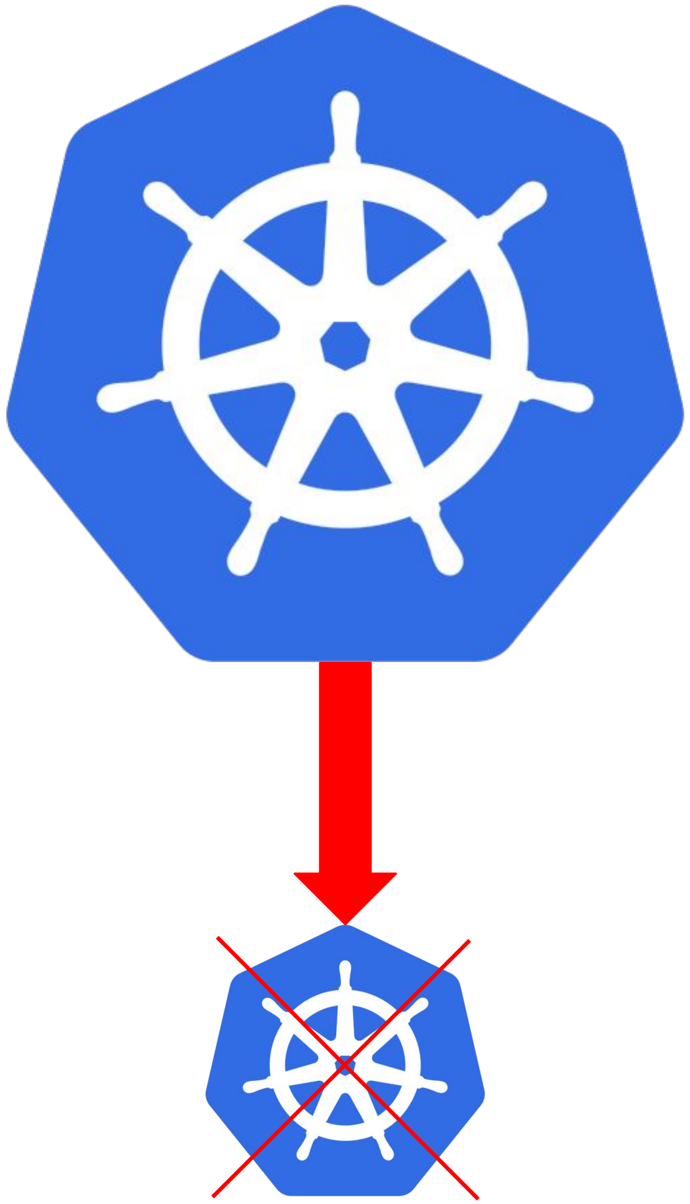
Case study: #2

Caches all the way down

Kubernetes API server keeps unaliving

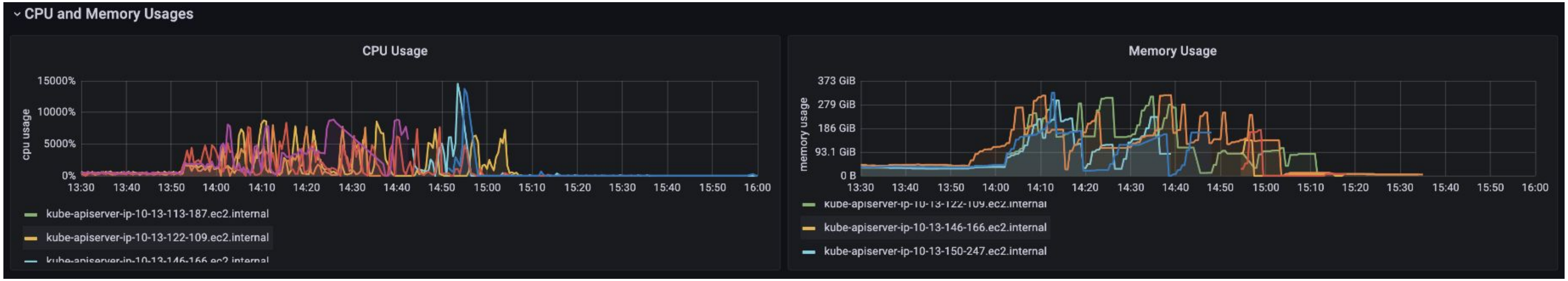
- We did a big scale down of our services
- And Kubernetes API servers started dying

  die → restart → stay healthy for a bit → die
- We were losing the control planes!
- Nearly impossible to do anything
  - How do you get information about nodes or pods?
- Fortunately, no customer impact
  - Kubernetes nodes are designed to keep running workloads in the event of a control plane outage



Off the charts

- API server resource usage was off the charts



Theories and temporary mitigation

- Theories
  - Inefficient audit logs webhook sink with no circuit-breaking

```
Error in audit plugin 'buffered' affecting 1 audit events: audit buffer queue blocked
```

  - Excessive ConfigMap/Secret mounting
- Embiggened the API servers → made the problem seem less severe

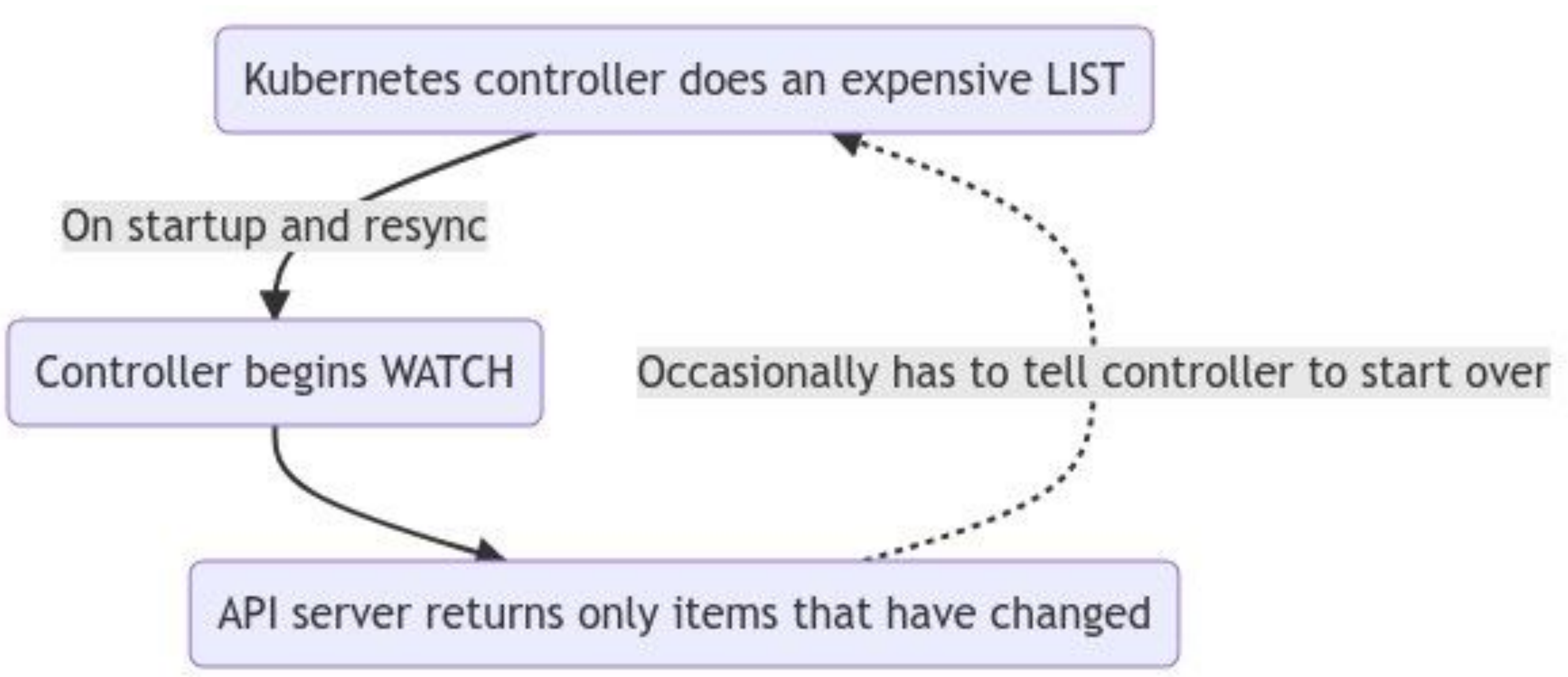
Do you even HTTP 410

- Load tests or deploying a service with large number of pods led to a lot of 410s
- This is well documented!

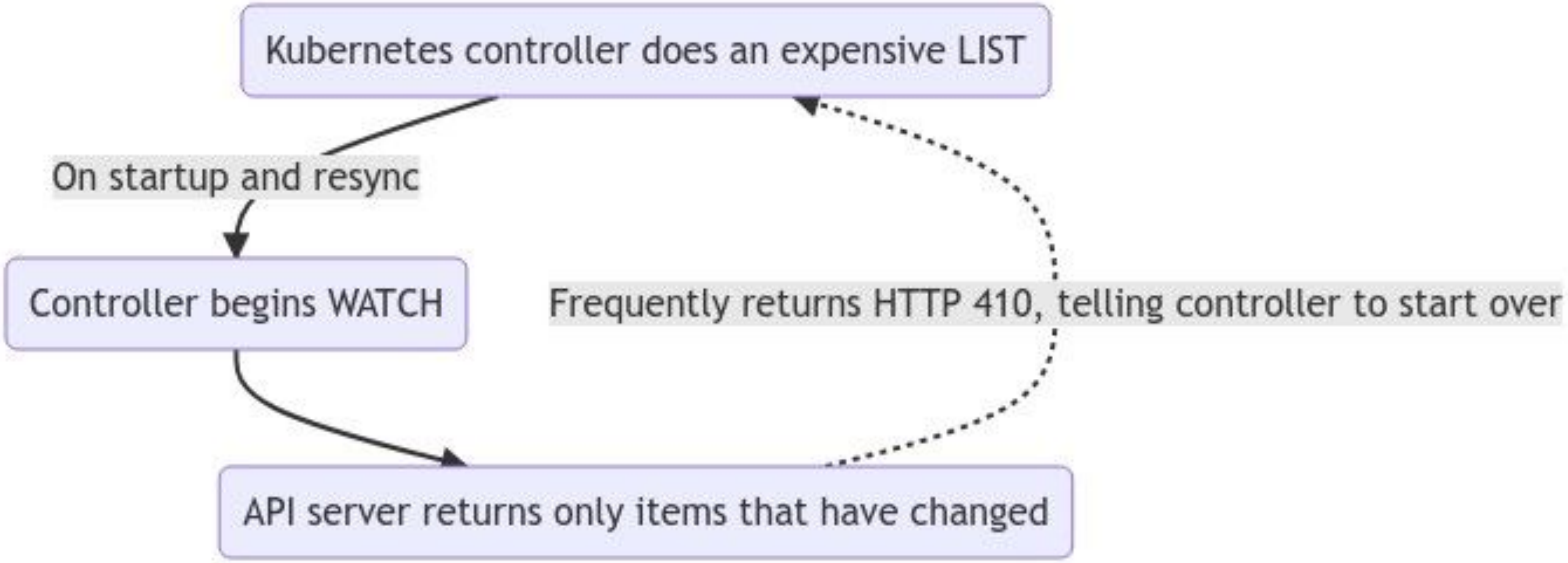
```
When the requested watch operations fail because the historical version of
that resource is not available, clients must handle the case by recognizing
the status code 410 Gone, clearing their local cache, performing a new get
or list operation, and starting the watch from the resourceVersion that was
returned.
```

https://kubernetes.io/docs/reference/using-api/api-concepts/#efficient-detection-of-changes

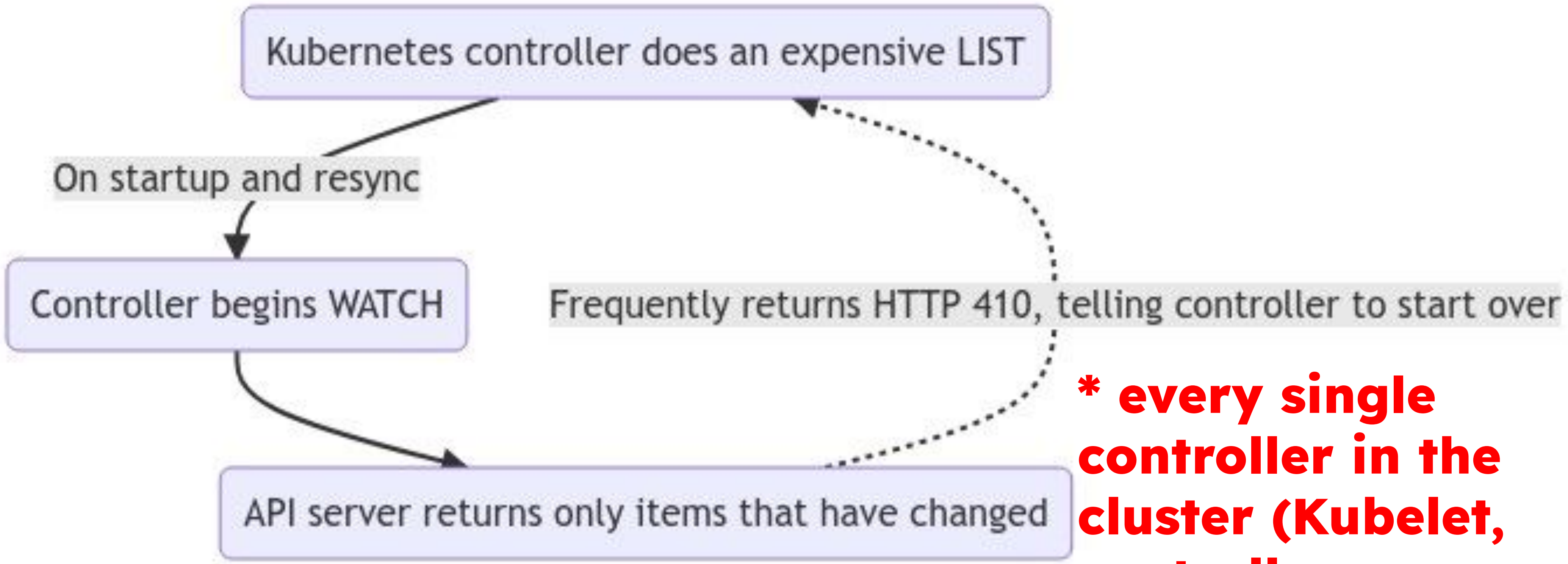
Do you even HTTP 410


Kubernetes controller does an expensive LIST
On startup and resync
Controller begins WATCH
Occasionally has to tell controller to start over
API server returns only items that have changed

Do you even HTTP 410


Kubernetes controller does an expensive LIST
On startup and resync
Controller begins WATCH
API server returns only items that have changed
Frequently returns HTTP 410, telling controller to start over

Do you even HTTP 410


Kubernetes controller does an expensive LIST
On startup and resync
Controller begins WATCH
Frequently returns HTTP 410, telling controller to start over
API server returns only items that have changed

*** every single controller in the cluster (Kubelet, controller manager, custom controllers)**

Downward spiral

Load testing or re-deploying large services

- Created 1000s of pods in a matter of a minute
- Watch cache was having a high miss rate
- Every client would do another list
  - Yes even kubelets … 1000s of them
- Further overloading the API servers
- Sending them into a downward spiral

Takeaway: Observability

Observability is the key to operational success

Takeaway: Observability - Metrics

KubeCon | CloudNativeCon North America 2022

- Kubernetes components export tons of metrics, scrape them all!
- Build meaningful dashboards out of them
    - Eg. Typha and API server dashboards helped us
- Structure the dashboards to make what matters the most easily accessible
- Have the ability to construct ad-hoc graphs, it matters

Takeaway: Observability - Logs

- Logs from components such as controller-manager, kubelet, etc. have been extremely useful
- API server logs are generally noisy and haven’t been useful in practice
- Having a good log querying and filtering UI comes in handy

Takeaway: Observability - Audit logs

- Fills the gaps left by API server logs
- Doesn't just help with security, but is also a powerful debugging tool
- Tells you who made the calls, how frequently and at what times
  - This was how we found out it was relists that were causing API servers to crash during the watch cache incident
- Have the ability to query and filter audit logs as well

Takeaway: Change visibility

Have visibility into changes that are pushed to cluster components, including extensions

Takeaway: Change visibility

- Have a simple system/tool to show the changes made to clusters
  - Example: Notifications of the summary of changes rolled out to a Slack channel dedicated for such notifications.

- Have the ability to construct the timeline of changes

- Kubernetes' extensibility model enables anybody to build extensions
  - Work with all your partner teams building such extensions
  - Ensure they are integrated into your change tracking system as well

Takeaway: Read the code!

Understand your components and how they work. Don’t be afraid to read code. In fact: you must

Takeaways: Read the code!

- Kubernetes is a large code base and has a large surface area
- Not everybody will become an expert in everything
- Build expertise for each areas within the team
- Experts for a given area should be familiar with the code of the components under their purview
- Experts should also be very comfortable with the debugging tools relevant to their area
  - `tcpdump`, `ip`, `dig`, etc. for networking
  - `pprof`, `flamegraphs`, etc. for general Kubernetes components

Please scan the QR Code above to leave feedback on this session