

The World Blew Up But We're All Okay

Managing a massive-scale incident at Datadog

Laurent Bernaille
Laura de Vesine



(Laura)

About Datadog

Data Processed

Tens of Trillions of events per day
Millions of hosts reporting
Billions of containers reporting
Over 600 integrations

Infrastructure

10 000s hosts in our infra
Dozens of k8s clusters
Multi-cloud

Company Size

Almost 5 000 employees
Over 25 000 customers
Rapid growth



you probably already know what Datadog does (we provide monitoring as a service in the cloud). For some extra context, here's some things about our scale – tl;dr, it's pretty big! We have pretty substantial scale in data processing, the size of our infrastructure, and the size of the company.

Scale Photo by [Piret Ilver](#) on [Unsplash](#)

March 8, 2023

Starting on March 8, 2023, at 06:03 UTC, we experienced an outage that affected the US1, EU1, US3, and US5 Datadog regions across all services.

When the incident started, users could not access the platform or various Datadog services via the browser or APIs and monitors were unavailable and not alerting. Data ingestion for various services was also impacted...

We...declared our first major service operational by 16:44 UTC... [and] declared all services operational in all regions on March 9, 2023, 08:58 UTC. The incident was fully resolved on March 10, 2023, at 06:25 UTC once we had backfilled historical data.

So up on the screen is an excerpt from our official blog post about this outage – you can find this on the Datadog engineering blog (<https://www.datadoghq.com/blog/2023-03-08-multiregion-infrastructure-connectivity-issue/>). This was a bad one. We were essentially globally down for 10, still had major services down 24 hours later, and essentially *two full days* before we declared impact fully over. Laurent and I are going to walk you through exactly what broke, the bones of how we started recovery, and what the scale and realtime organization of our response looked like.

What happened?




Laurent: @1:00-8:00

The story begins in 2020

<https://github.com/systemd/systemd/pull/17477>

network: do not serialize/deserialize routing policy rules #17477

 Merged yuwata merged 5 commits into `systemd:master` from `yuwata:network-drop-serialization-routing-policy-rule` 

 Conversation 15

 Commits 5

 Checks 0

 Files changed 8



yuwata commented on Oct 28, 2020

Member · ·

No description provided.



The story begins in 2020

<https://github.com/systemd/systemd/pull/17477>

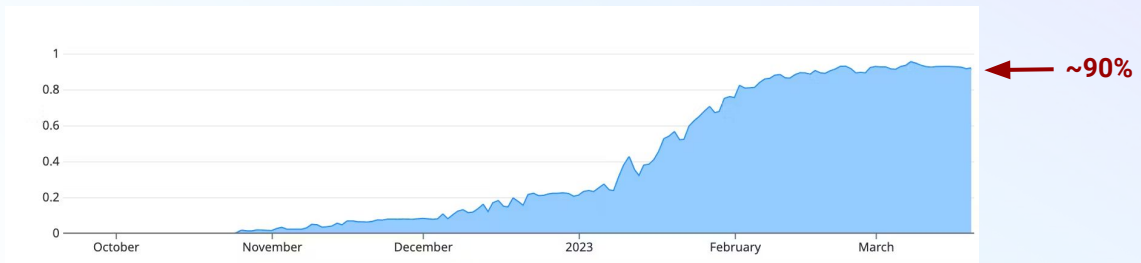
The motivation of this PR is to drop foreign routing policy rules except created by kernel on startup. `networkd` already drop foreign addresses, routes, and nexthops. The serialization/deserialization of rules does not work on startup, and work only limited cases on restart of `networkd`. More specifically, the rules created by other tools like `ip rule` are not removed. In other words, the serialization/deserialization logic only handle rules previously created by `networkd`.

systemd v248: **on networkd restart, drop routing policy rules**
systemd v249: make this behavior configurable

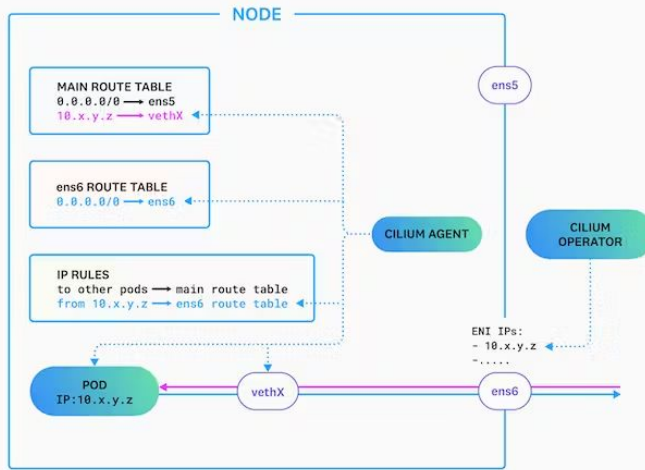
Backported to v247/v248

Systemd at Datadog

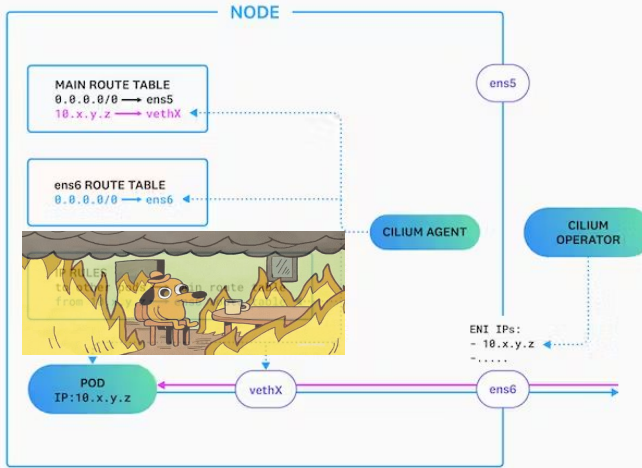
- We run Ubuntu LTS
 - Ubuntu 20.04: systemd v245
 - Ubuntu 22.04: systemd v249
 - We started to rollout Ubuntu 22.04 in November 2022



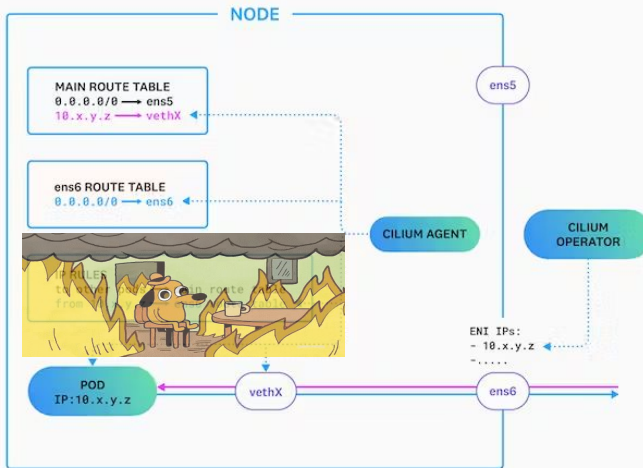
Kubernetes networking at Datadog



On systemd-networkd restart



On systemd-networkd restart



- Pods lose network connectivity
- Nodes lose network connectivity

Why did we discover this on March 8th?

- We never restart systemd / networkd

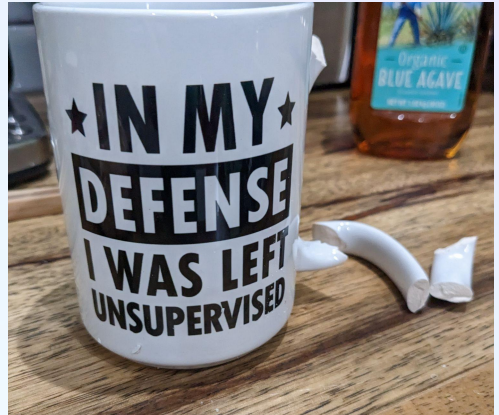
Why did we discover this on March 8th?

- We never restart systemd / networkd
- March 7, 2023, patch for CVE in systemd

Why did we discover this on March 8th?

- We never restart systemd / networkd
- March 7, 2023, patch for CVE in systemd (CVE unrelated to incident)
- Ubuntu defaults for unattended (automated) upgrades

```
[Timer]
OnCalendar=*-*-* 6:00
RandomizedDelaySec=60m
Persistent=true
```

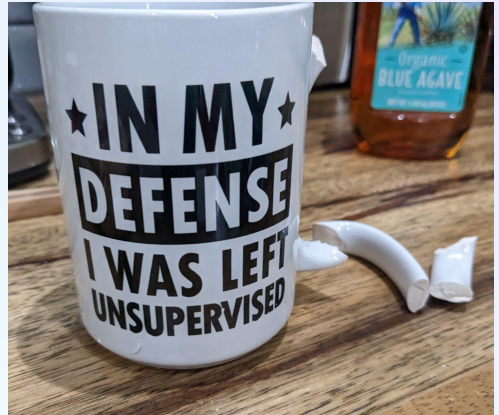


Why didn't we see it before March 2023?

- We never restart systemd / networkd
- March 7, 2023, patch for CVE in systemd
- Ubuntu defaults for unattended (automated) upgrades

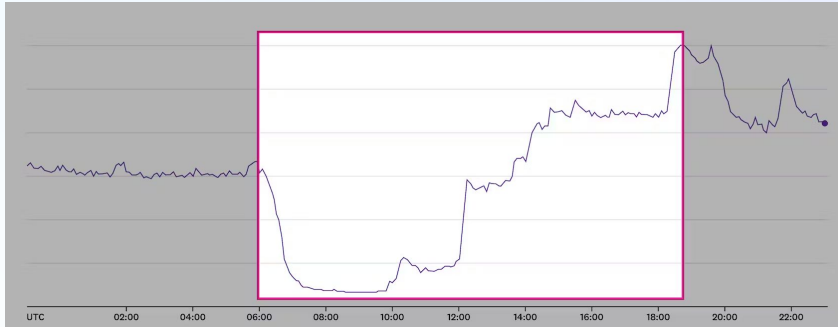
```
[Timer]
OnCalendar=*-*-* 6:00
RandomizedDelaySec=60m
Persistent=true
```

- Patch applied **everywhere** 6:00-7:00 UTC



Consequences for GCP / Azure

- Pods lost network connectivity
- Nodes lost network connectivity



Sum of packets sent by instances in EU1 on March 8

Consequences for GCP / Azure

- Pods lost network connectivity
- Nodes lost network connectivity
- Nodes remain running

Consequences for GCP / Azure

- Pods lost network connectivity
- Nodes lost network connectivity
- Nodes remain running
- **Restarting nodes fixes the issue**

Consequences for AWS

- Pods lost network connectivity
- Nodes lost network connectivity

Consequences for AWS

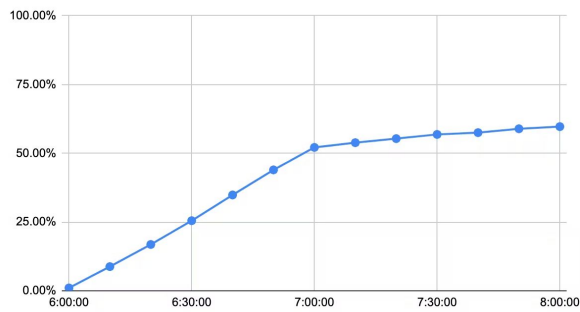
- Pods lost network connectivity
- Nodes lost network connectivity
- Nodes fail instance health-checks

Consequences for AWS

- Pods lost network connectivity
- Nodes lost network connectivity
- Nodes fail instance health-checks
- **AWS autoscaling group controller replaces the instance**

Consequences for AWS

- Pods lost network connectivity
- Nodes lost network connectivity
- Nodes fail instance health-checks
- **AWS autoscaling group controller replaces the instance**



Proportion of instances running at 6:00 **terminated** between 6:00 and 8:00 (cumulated)

Consequences for AWS

- Pods lost network connectivity
- Nodes lost network connectivity
- Nodes fail instance health-checks
- AWS autoscaling group controller replaces the instance
- **Great for stateless services (web) which recover in a few minutes**

Consequences for AWS

- Pods lost network connectivity
- Nodes lost network connectivity
- Nodes fail instance health-checks
- AWS autoscaling group controller replaces the instance
- Great for stateless services (web) which recover in a few minutes
- **But we use local SSD almost everywhere**
 - Kafka
 - Zookeeper
 - FoundationDB
 - Cassandra

Consequences for AWS

- Pods lost network connectivity
- Nodes lost network connectivity
- Nodes fail instance health-checks
- AWS autoscaling group controller replaces the instance
- Great for stateless services (web) which recover in a few minutes
- **But we use local SSD almost everywhere**
 - Kafka
 - Zookeeper
 - FoundationDB
 - Cassandra
- **We lost data / quorum in 100s of these systems**

Apt-ocalypse summary

- March 8th 6am: Patch for systemd CVE
- Networkd restarts and flush ip rules required for Kubernetes networking
- Pods lose connectivity
- **60% of Nodes get disconnected from the network**
 - GCP/Azure: nodes continue to run
 - AWS: **nodes are terminated**, severely impacting systems with local disks
- Initial (very wrong) assessment: GCP/Azure are more broken

Everybody Panic!



Laura: 8:00-13:45

01:22 [REDACTED] This might be a SEV1

01:23 silverrose hi folks

we're fully down it sounds like for alerting?
(core IC here; catching up)

I don't see a zoom, do we have one up?

01:24 [REDACTED] I'm getting a ton of pages from clobox and bubs

Yes

Let's open zoom

01:24 silverrose ack

this is a sev 1

01:24 Zoom APP

Call ▾

Zoom meeting started by laura.devesine

[REDACTED]

Meeting ID: [REDACTED]

493 people joined

Meeting passcode: Wi91UEpPTTI5MHdBallsZVBaU2JpQT09

01:24 [REDACTED] We are probably blind

01:25 silverrose who needs to be here to help with troubleshooting?

I'm putting up a status page

01:25 Datadog APP

@silverrose changed Severity to SEV-1

01:25 Datadog APP set the channel topic: SEV-1 ■ Active — Alerting is down in all DCs

Incident Commander: [REDACTED]

So, it's about 1:20am. Like any perfectly normal SRE (look, I'm a night owl) I'd just gone to bed and turned out the light... and the pager goes off. Pretty annoying, these things usually take a couple of hours but I can cancel my first couple of meetings in the morning and sleep in. I log on to my work machine, pull up the incident slack channel, and see... <click>

What you're looking at there are graphs showing the error rate from our alerting system going to 100% in multiple regions, and a responder suggesting that this is a SEV1. The same responder also noted that the datadog app itself was only intermittently responsive. That seems... bad. I do a very quick initial triage: we're not evaluating alerts for any customers, globally. The way that our alerting system works, that's generally a sign that intake or processing of metrics are failing in some way. Right now, top priorities are to get a status page up for customers and get more responders in the room <click>

My first actions are to state the impact as I understand it, escalate to SEV1, get a video call opened for coordination, start working on a status page, and get started assembling our response team. That's enough going on that I also paged my secondary to help take over some of that communication and coordination.

Fortunately, we planned ahead

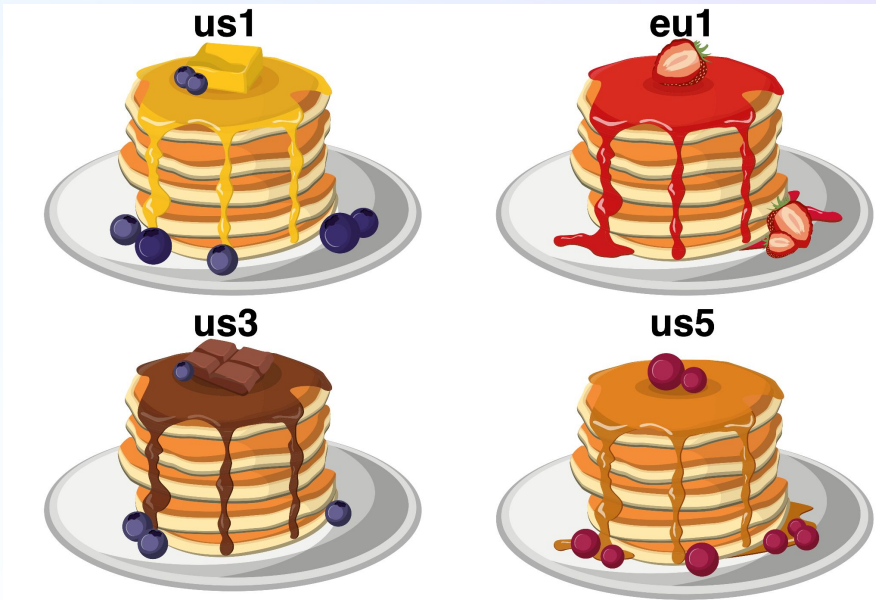
1. You build it, you run it
2. Out-of-band monitoring
3. Datadog incident app
4. High severity response rotation
5. On-call engineering execs



Taking a step back, I want to talk through how I got pulled into this incident and some behind-the-scenes things that allowed us to get a decisive incident response spun up very quickly – you don’t invent incident response on the fly. There’s a number of important moving parts in our incident response planning that are visible in these first few minutes. [First](#), every DD engineering team operates and is oncall for their own services (you build it, you run it). Our SRE org is responsible for making sure they have the tools they need to succeed in doing that, but the actual ops and oncall are done by engineers. [While](#) we mostly monitor Datadog with Datadog, we run some basic “liveness” monitors on completely separated infrastructure with absolutely no shared fate with our own services – and those are what alerted our very first responder to the incident. [That responder](#) had training and practice in opening incidents and knew they were going to need help, so they opened a high-severity incident. Opening an incident in the app automatically opens a dedicated slack channel for coordinating response. [Fourth](#), high-severity incidents automatically page a major incident response rotation – a rotation of senior engineers across the company whose job is to take over command of high-severity incidents. [And finally](#), we also have automatic paging of engineering execs, who can support especially for customer communication in our highest severity incidents – bumping to a SEV1 triggered that page automatically as well. All of this together meant that we had a senior engineer ready and able to coordinate response, an engineering exec to help prioritize and communicate to customers, a customer-facing status page up, and a structure in place to start building out the rest of our response, within about 10 minutes of the incident being opened.

[Planning Image by GraphicaStock](#) on Freepik
Piggybank Image by [Alexa](#) from [Pixabay](#)
Umbrella Image by [Anrita](#) from [Pixabay](#)

This can't happen



But like... now what? Here's an important thing to understand about Datadog: all of our regions are **completely** separated. They don't have a shared control plane, or shared networking, and we don't do global rollouts. Which means it should be impossible for any one thing to take us down globally simultaneously... but here we are. Complicating things, we mostly monitor DD with DD, so our visibility into what was actually going on was *incredibly* limited.

Pancake Image by [pch.vector](https://www.freepik.com/free-vector/pch-vector) on Freepik

Who are you going to call?

- Shared configuration?
- Network?
- Kubernetes?



We fell back on brainstorming and extrapolating from the limited signals we had:

- We have a shared configuration system that is built on discrete infrastructure, with separate configuration for each region, but... maybe it behaved globally somehow? The last team to make a change to that configuration was our web ui team; let's page them in
- When "everything is down", it's usually the network. So we paged the network team
- Attempts to SSH into k8s nodes to see some telemetry there aren't always working – let's page compute, who run our k8s infra

This incident was so extreme. Normally, when you have an incident, there's at least *some* idea what's gone wrong – you can at least see which services are returning errors or crashing! Here, we didn't even have a working model of "what's broken" for a full hour of our incident response. It actually took us until around 5 hours into the incident response to identify the trigger as the unattended upgrade mechanism Laurent walked you through. Some of that was because we were prioritizing recovery, not investigation, but some of it was also because all of our own telemetry was so badly impacted and the actual mechanism was so difficult to pinpoint. Alright, so we're about an hour in, various people have joined the response, and we now understand that *something* happened that broke our k8s nodes. It took us another 45 minutes or so to validate that whatever it was had stopped happening... and now we had to fix it. At scale.

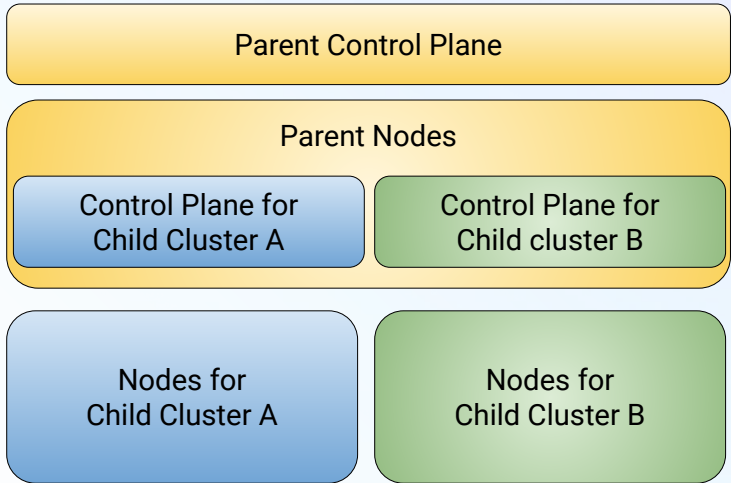
[Question mark icons created by Freepik](#) - Flaticon
[Ghost image](#) by Vecteezy

Recovering the platform: EU1 (GCP)

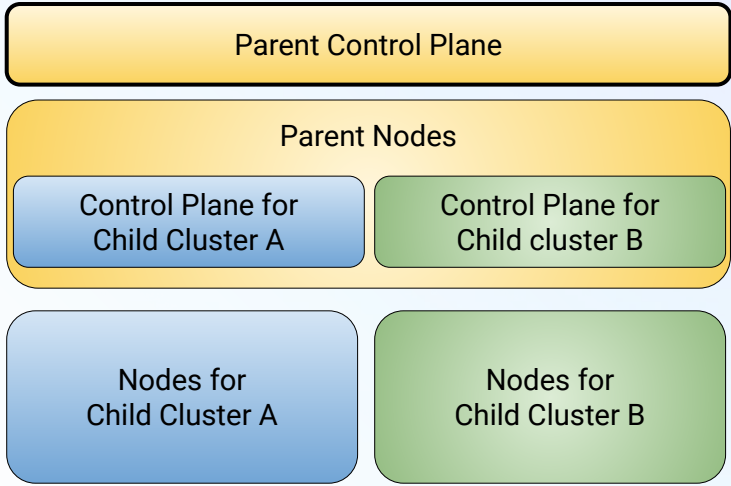


Laurent: @13:45-18:15

Different flavors of Kubernetes Parent / Child clusters



Recovery sequence

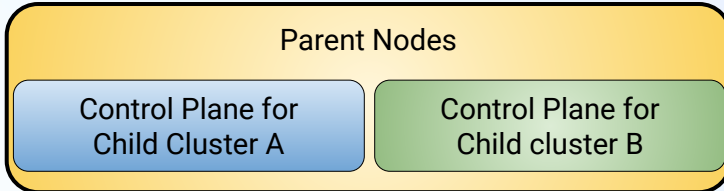


≤ 1

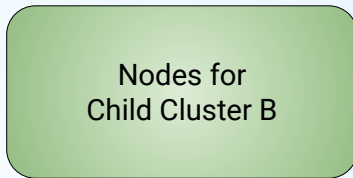
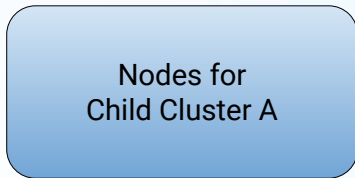
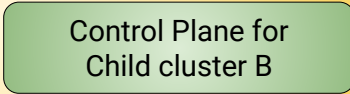
Recovery sequence



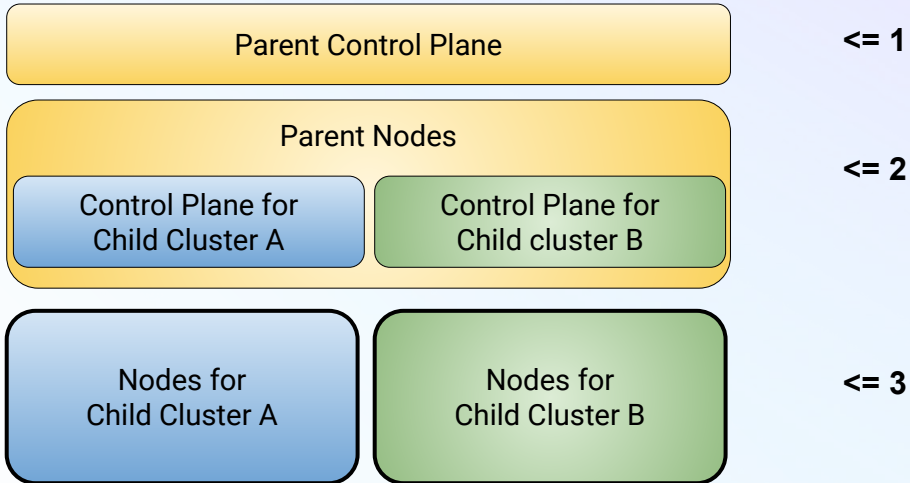
≤ 1



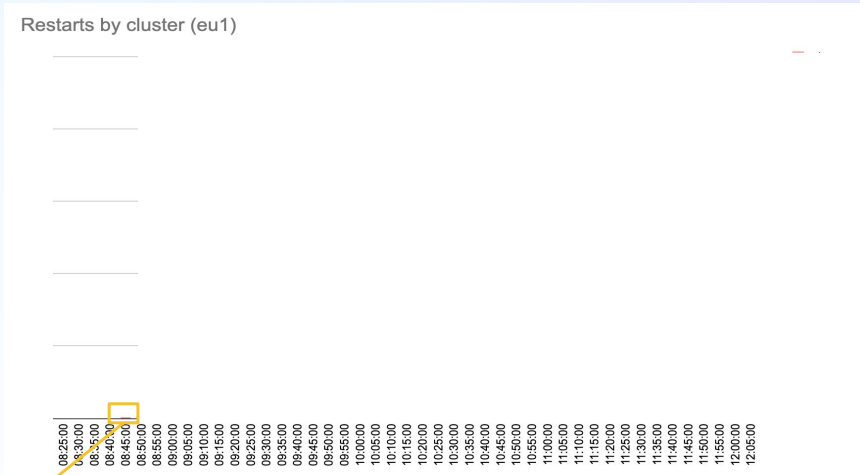
≤ 2



Recovery sequence



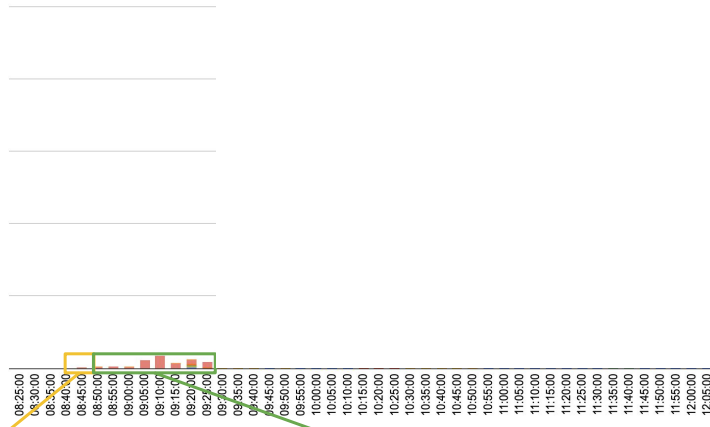
Recovery sequence



Parent control plane nodes

Recovery sequence

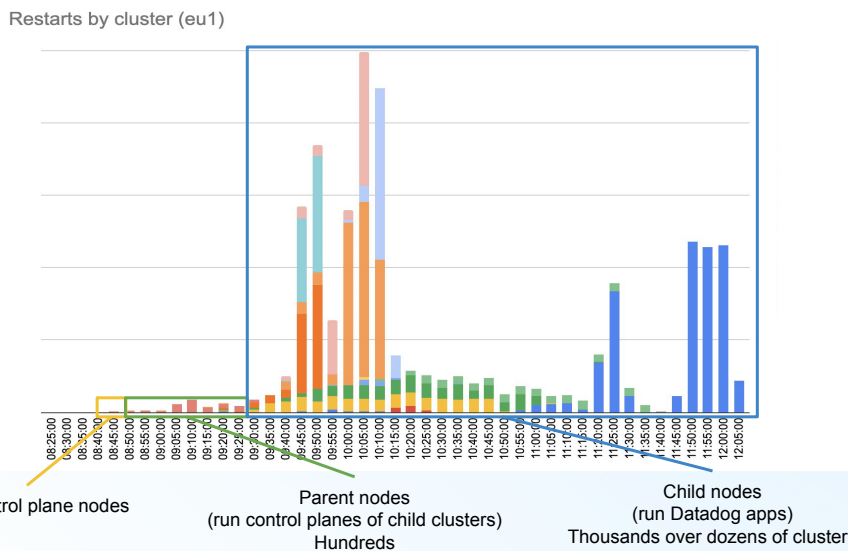
Restarts by cluster (eu1)



Parent control plane nodes

Parent nodes
(run control planes of child clusters)
Hundreds

Recovery sequence



Parent control plane nodes

Parent nodes
(run control planes of child clusters)
Hundreds

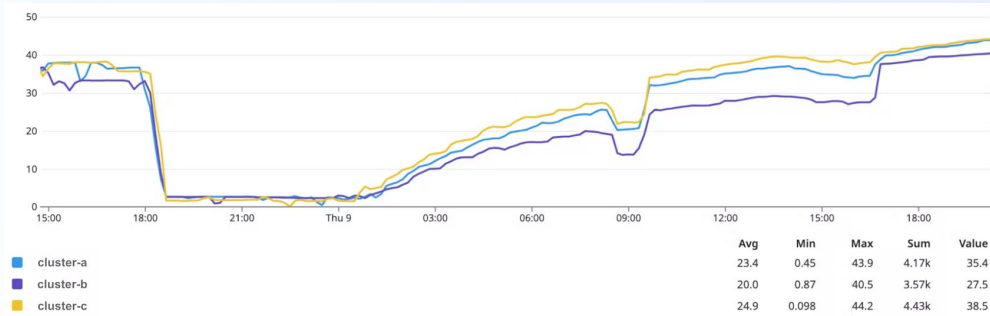
Child nodes
(run Datadog apps)
Thousands over dozens of clusters

Getting capacity to process backlog: limits

- Maximum number of instances in a peering group

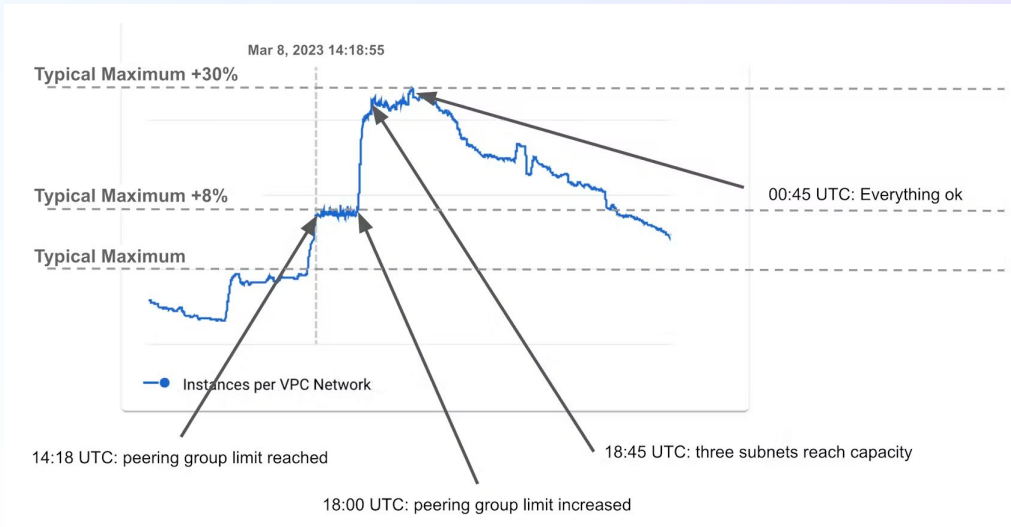
Getting capacity to process backlog: limits

- Maximum number of instances in a peering group
- Subnets for 3 clusters processing logs and traces are full



Proportion of IP available for the 3 clusters

EU1: Timeline of Compute capacity recovery

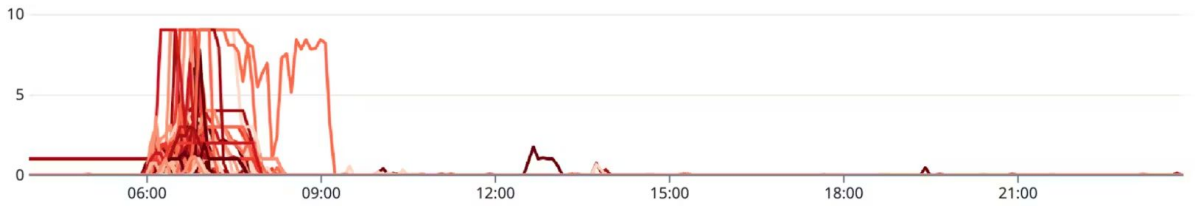


Recovering the platform: US1 (AWS)



Laurent: @18:15-26:45

Recovering control planes

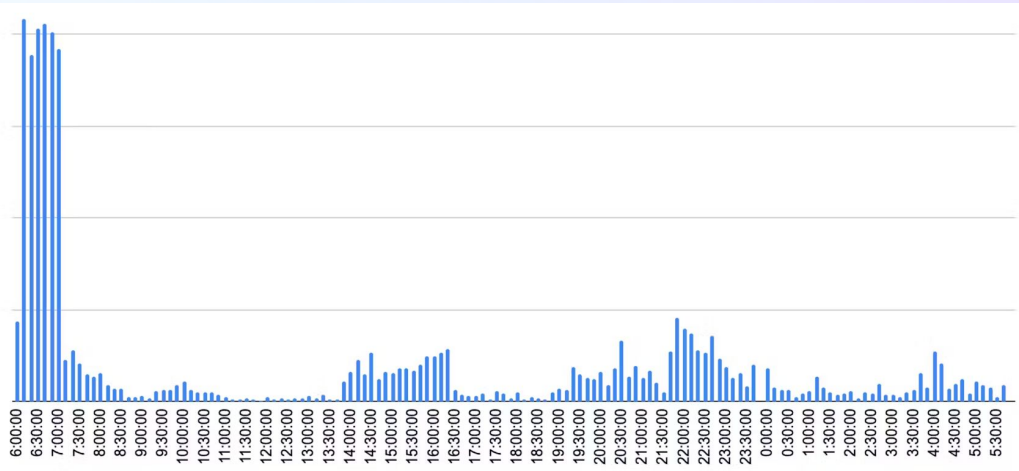


kube_namespace in unavailable		↓ Avg	Min	Max	Sum	Value
■	cluster-a	0.90	0	9.00	215.1	0
■	cluster-b	0.54	0	9.00	128.7	0
■	cluster-c	0.45	0	9.00	106.5	0
■	cluster-d	0.28	0	9.00	67.4	0

Number of API servers not available per cluster in US1

Stateless + etcd uses remote disks => recovered by themselves
It still took time => We were hitting AWS APIs hard and many calls were failing

Many new nodes



Number of instances created in US1 via RunInstances API calls

Steps to create new nodes

NotStarted

Request for new node made, not created

Steps to create new nodes

NotStarted

Request for new node made, not created

Unregistered

Node has not registered with the cluster
Node needs certificate from Vault

Steps to create new nodes

NotStarted

Request for new node made, not created

Unregistered

Node has not registered with the cluster
Node needs certificate from Vault

Unready

Node networking is not Ready (usually)

Steps to create new nodes

NotStarted

Request for new node made, not created

Unregistered

Node has not registered with the cluster
Node needs certificate from Vault

Unready

Node networking is not Ready (usually)

Ready

Pods can be scheduled on the node

Steps highly dependent on Cloud provider APIs

NotStarted

Request for new node made, not created

RunInstances

Unregistered

Node has not registered with the cluster
Node needs certificate from Vault

DescribeInstances

Unready

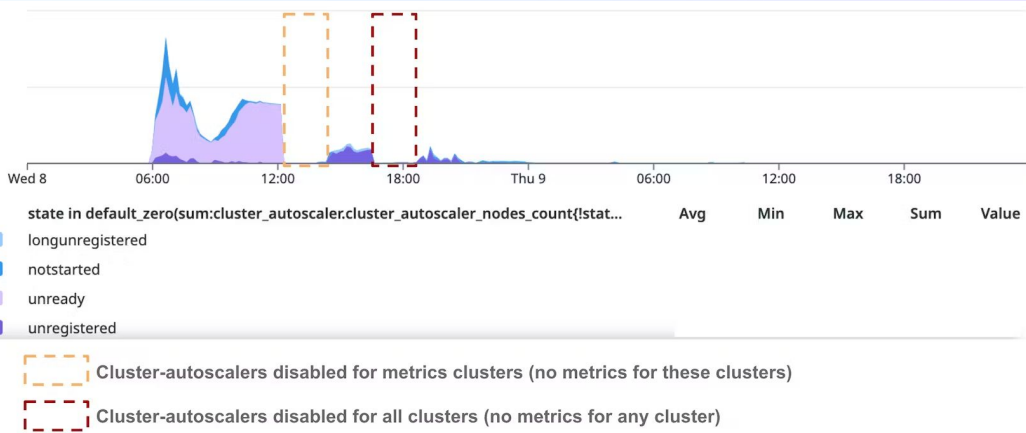
Node networking is not Ready (usually)

CreateNetworkInterfaces

Ready

Pods can be scheduled on the node

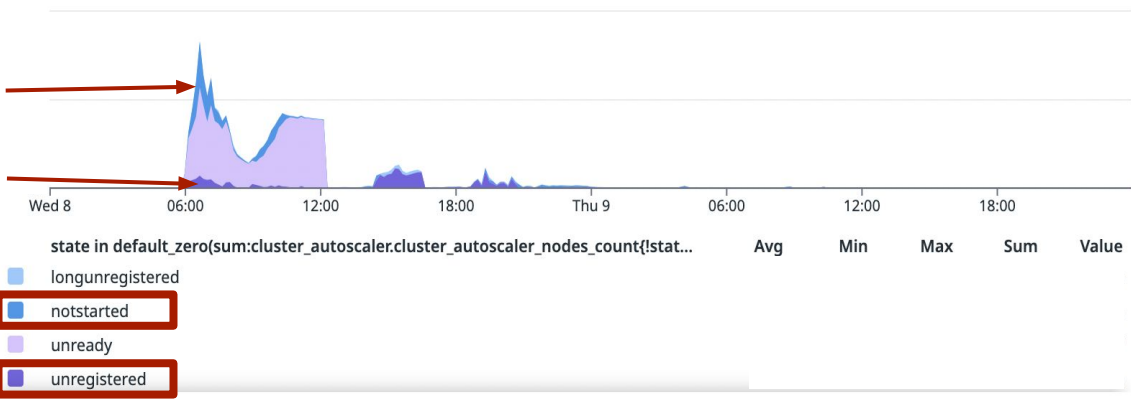
Restoring compute capacity



Instances not in "Ready" State

6am-8:30am: NotStarted/Unregistered

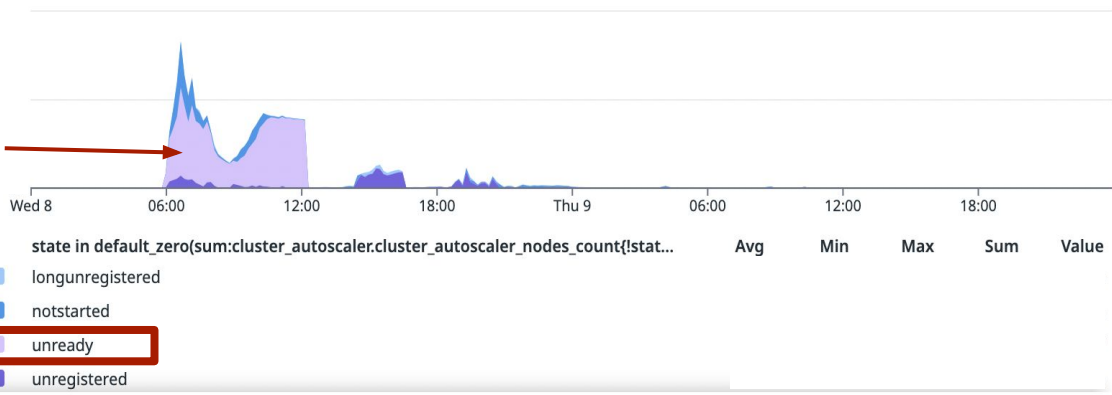
Nodes not in Ready state in us1



6-8:30: control planes unstable
NotStarted: RunInstances API
Unregistered: DescribeInstances API

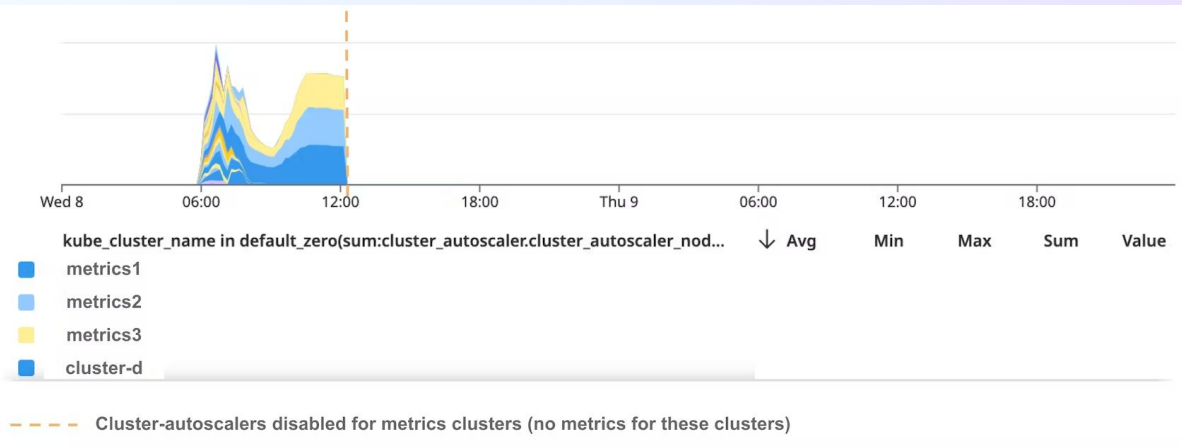
6am-3pm: Unready

Nodes not in Ready state in us1



Focus on unready

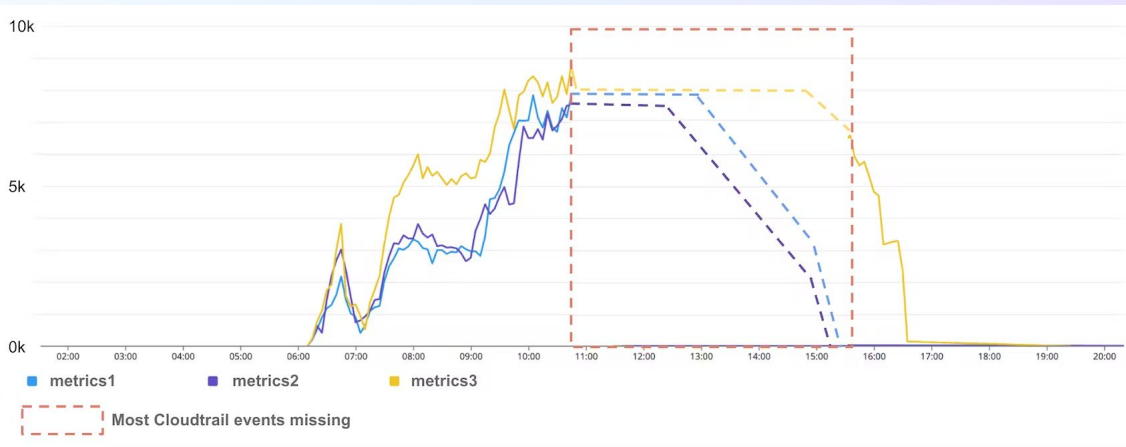
6am-3pm: Unready



Unready nodes

- Only three clusters
- The biggest ones
- They can't get additional interfaces which are required for pods

6am-3pm: Unready



Number of CreateNetworkInterface API calls being rate limited per minute

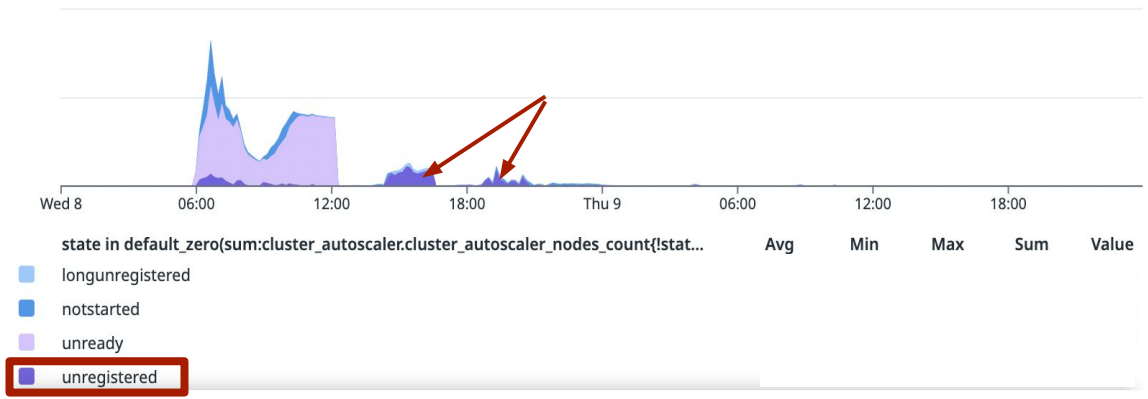
99+% of calls to CreateNetworkInterfaces were rate limited

Decrease pressure by removing Unready nodes / increasing limits (AWS)

Ultimately : found a bug in Cilium-operator: aggressive retries and no exponential backoffs

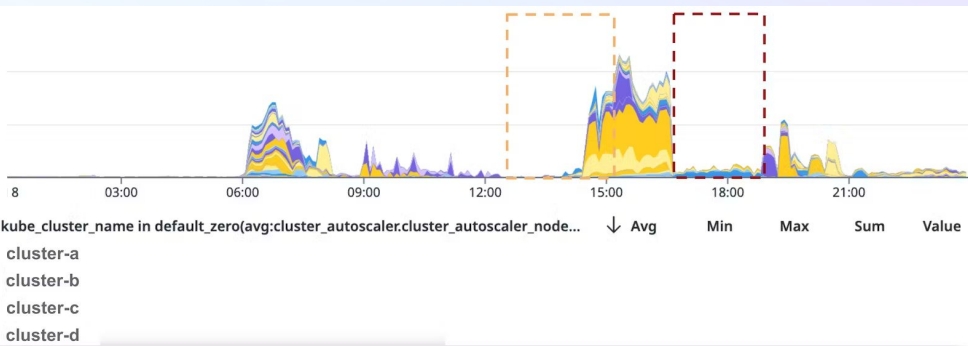
2pm-7pm: Unregistered

Nodes not in Ready state in us1



Once things stabilized => scale up to process backlog (start @2pm)
Let's focus in on unregistered nodes

2pm-7pm: Unregistered

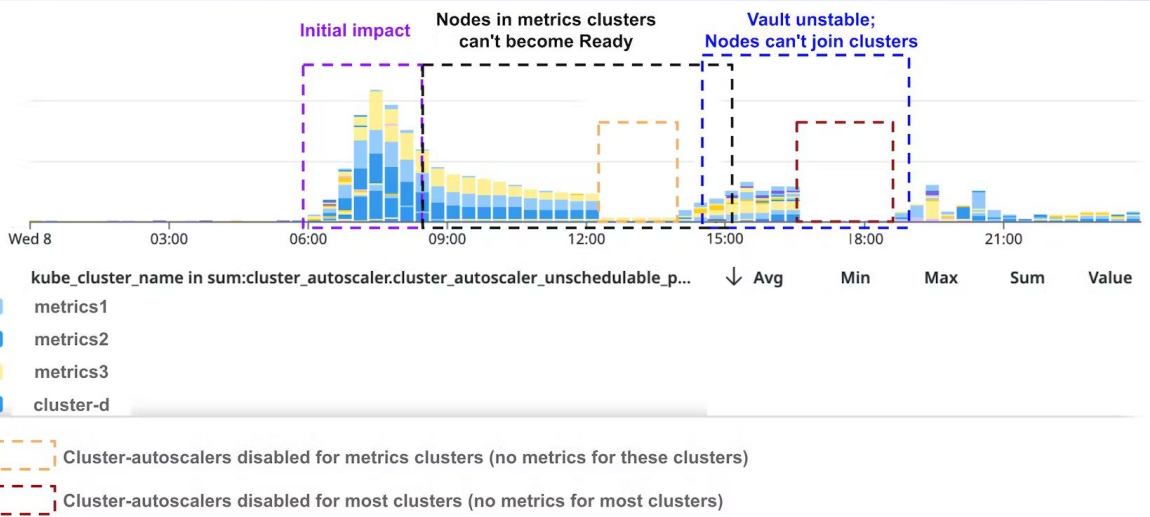


- Cluster-autoscalers disabled for metrics clusters (no metrics for these clusters)
- Cluster-autoscalers disabled for most clusters (no metrics for most clusters)

Unregistered nodes

14:30 => increase in node creation increase rate limits, vault gets slow and unstable
Recovery: decrease pressure on vault by rate limiting incoming cert requests
Once vault was stable again slowly increase rate limits

US1 Recovery timeline looking at pending pods



Number of pending pods by cluster

3 main periods

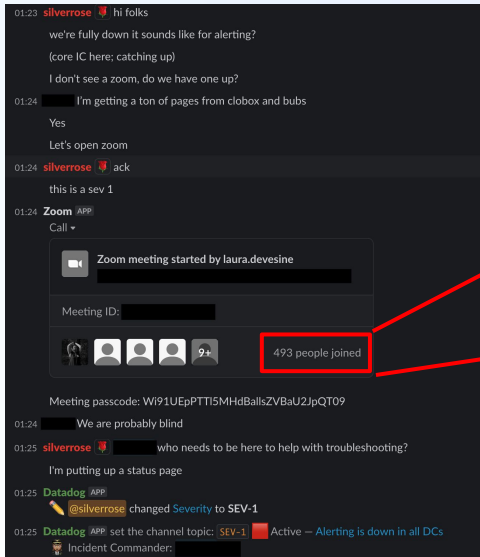
- Initial issue
- Metrics clusters can't get nodes because they can't get network interfaces
- Vault is unstable due to scale up to process backlog

Humans Press Buttons



Laura: @26:45-38:00

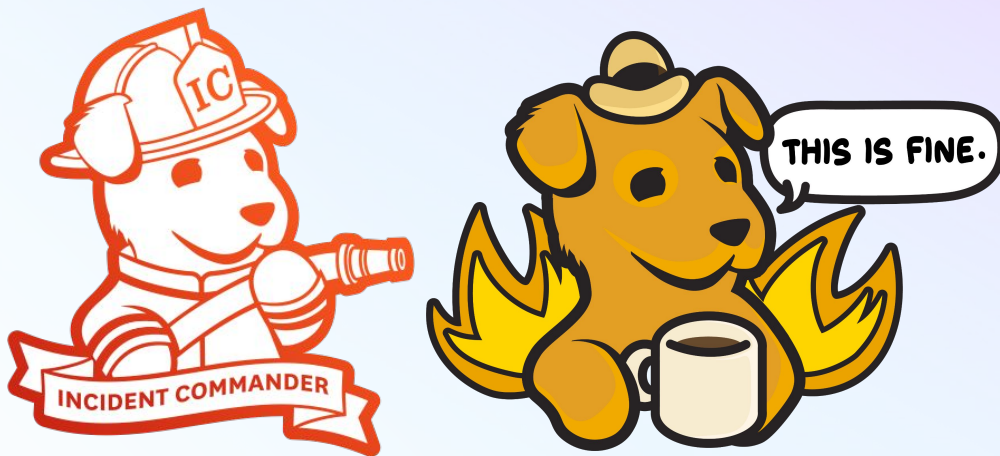
Teamwork makes the dream work



493 people joined

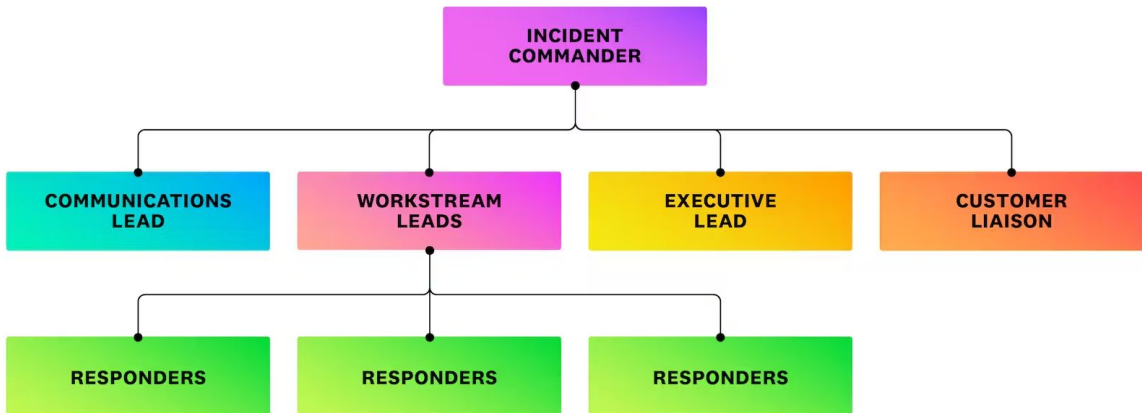
So now that you have a sense of how we put our infrastructure back together – which obviously was only the first step in getting our services back up and running – I want to draw your attention to something you might or might not have spotted previously when I was talking about our initial response. Yep, that’s nearly 500 people joining our (first) incident response call (which lasted ~14 hours). This isn’t the only call that was open for the incident (there’s an unfortunate feature of zoom where one might accidentally end the call when handing off…), and certainly not every responder joined the call. But it’s safe to say, conservatively, that our response to this incident included **at least** 500 people, the vast majority of them engineers trying to fix parts of Datadog. How do you successfully manage 500+ responders to an incident?

Response is both trained and cultural



When we have any large incident, Datadog engineers, who remember are responsible for operating their own systems, will join proactively. They'll do this either because they're also getting alerts for their systems or, in the case of something like this incident, because they happen to be online (either because their work/life balance is leaky, or because it's during working hours) and they see in slack that there's a very large incident, and simultaneously that nothing in the Datadog app seems to be working. Culturally, our engineers feel *responsible to our customers* in this situation, so they'll join the incident response to try and help out. We've also trained every DD engineer on how to integrate into an ongoing incident response and, when it's large, to spin up a sub-team for their own system, with a little sub-tree of responsibility

Who Loves Org Charts?



The principle here is pretty simple – each team (or major sub-problem) for the response gets a workstream, each workstream has a lead, and those workstream leads function as localized incident commanders who will keep up with the larger response (as its overall priorities shift) and kick problems back up to the overall incident commander. Workstreams here are both a human designation (“this group of people is working on this task”) and a feature of the DD incident app, which opens a slack channel for each workstream.

That scale though...

```
▼ 19524
# 19254-customer-co...
# incident-19253
# incident-19254
# incident-19254-aaa
# incident-19254-aaa-...
# incident-19254-aler...
# incident-19254-apm
# incident-19254-ap...
# incident-19254-atla...
# incident-19254-ciapp
# incident-19254-clo...
# incident-19254-clo...
# incident-19254-co...
# incident-19254-com...
# incident-19254-coo...
# incident-19254-dat...
# incident-19254-dbm

# incident-19254-dep...
# incident-19254-dis...
# incident-19254-dx
# incident-19254-ela...
# incident-19254-err...
# incident-19254-eve...
# incident-19254-evp
# incident-19254-evp...
# incident-19254-fdb
# incident-19254-flink
# incident-19254-fro...
# incident-19254-gse
# incident-19254-hist...
# incident-19254-imp...
# incident-19254-inc...
# incident-19254-infr...
# incident-19254-k9
# incident-19254-kafka

# incident-19254-live...
# incident-19254-logs
# incident-19254-log...
# incident-19254-me...
# incident-19254-me...
# incident-19254-me...
# incident-19254-mo...
# incident-19254-net...
# incident-19254-net...
# incident-19254-no...
# incident-19254-npm
# incident-19254-op
# incident-19254-pla...
# incident-19254-pm-...
# incident-19254-pos...
# incident-19254-pro...
# incident-19254-pro...
# incident-19254-red...

# incident-19254-regi...
# incident-19254-rev...
# incident-19254-rum
# incident-19254-rum-k
# incident-19254-sec...
# incident-19254-ser...
# incident-19254-slos
# incident-19254-sup...
# incident-19254-sync
# incident-19254-syn...
# incident-19254-trus...
# incident-19254-us5...
# incident-19254-xpq
# incident-19254-zoo...
# incident-19255
# incident-19259
# incident-19263
# incident-19263-coord
```

Which means that we can also see, roughly, how many streams of work we had involved for those >500 responders by looking at how many slack channels were opened for this incident. It's a lot (I'll save you counting, there's more than 70 channels listed there). So, those are some things about organizing a response that went really well – we had the training in place, a skeleton for our response structure that was well-understood and people were able to integrate into successfully, and smart engineers who were ready and willing to work on the problem. Now that I've told you it was all sunshine and roses... let me talk about a few things that we had to iterate on live.

(1) As responders join...

- Maximum of 100 editors
 - Make sure the link you share is for “view” mode
- Discoverability?
 - Pin in Slack, also share directly
 - As workstreams spin up, context is shared there
- Audience
 - Separate documents for support, non-eng



So, you have a rapidly-evolving state of “what we know about what’s happened, what theories we’re looking into right now, what we need new responders to do/not do”. You want to be able to communicate that state to people and keep it easily up to date. So, I don’t know about you, <click> but reach for a Google doc. Turns out there’s some challenges here. Firstly, <click> Google docs has a limit that a [maximum of 100 browser tabs](#) can have the same document open for editing at once. <click> we can adjust to solve this by making sure the “main” link we share with responders is for the document in “view” mode (which we worked out after a bit). <click> our second big challenge with a google doc was that it’s just surprisingly hard to make sure that new responders see and open the link for the doc. <click> We pinned the document in the incident slack channel and the incident app, but we also found we just needed to link it directly in the slack channel regularly as people joined so they could get up to speed without reading the whole channel. <click> as our response got larger and folks had team-specific workstreams to join, we found that mostly what happened is people were brought up to speed by their workstream lead. <click> another interesting challenge with a “status” google doc is that folks who aren’t engineers but have some form of customer-facing role *very much* want to know what’s going on with this huge thing. They have a tendency to open up that status document and read (and sometimes try to share) it, missing the context that this a lot of this is ongoing investigation. <click> the solution to that is to make sure we spin up separate document explainers for customer-facing roles with “things we’re sure enough to share, that don’t get into internal details that won’t make sense”, and one for non-engineers like our reception staff.

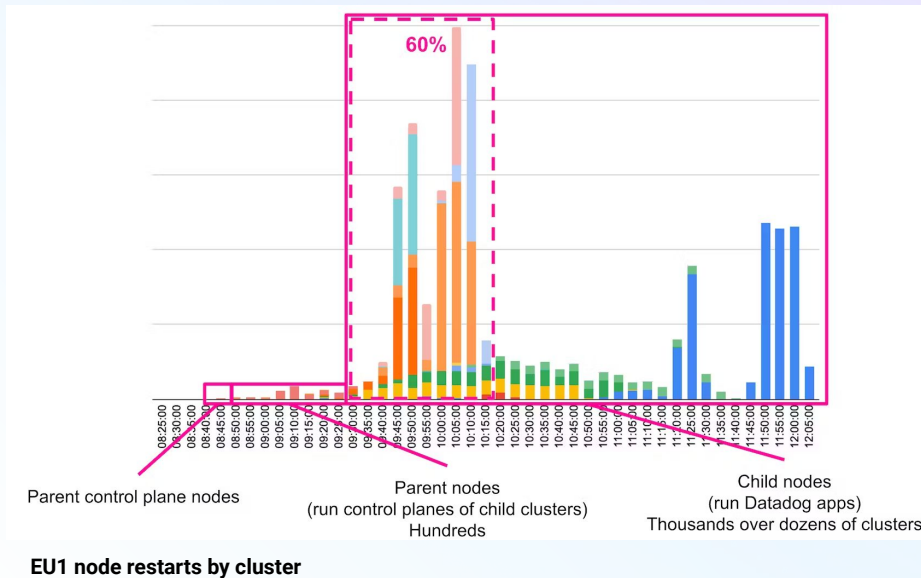
(2) As Slack channels proliferate...

- IC cannot possibly read every channel live
 - Workstream leads surface issues
 - Incident app updates
- Getting everyone's attention
 - Slack group for workstream leads
 - Some responders who just... post in a lot of channels



I need to emphasize again that this incident (eventually) had more than 70 workstreams, each with its own Slack channel. <click> it's more or less impossible for an incident commander (or anyone else) to keep up with all of those channels "live". Here, our solutions were somewhat built in by the DD incident app: <click> each workstream had a designated lead, and those leads understood their role of coordination because it's part of our incident training (and regular practice). <click> in addition, we could ask for updates from workstream leads in via the incident app, which then puts all of those updates in essentially a single place, to keep track of the high level state of things. <click> the other thing that having a response spread across many slack channels makes difficult is anything that needs to halt responders and get everyone's attention. Not every responder joins (or is paying attention to) the "main" incident channel, which is *by design*. Mostly, we want responders able to focus on restoring their own services, not feeling responsible for keeping track of the global state of response, and having their own dedicated slack channels for that is a great solution. But when we need to get everyone's attention... how do we do that quickly? <click> in this case, we didn't have tooling ready to go that helped, so we made a slack group for workstream leads and kept it up to date as responders rotated, so that we could send major pings to that group. <click> we did also wind up with some people as part of our response (coordinating with our communications lead) whose main role really did boil down to "ping individuals in various channels as needed". Sometimes, it's appropriate to just throw a warm body at a problem

(3) Human factors at scale...



Coordination is challenging, and it depends in part on the personalities of the people involved. Coming back to this chart of node recovery we saw before... it turns out that those blue clusters are the ones where we run the services for our metrics processing pipeline. Clearly they should have been very high priority to recover, but we didn't get to them until basically last. What happened? Two things: first, the teams for metrics services are much more concentrated in the US, so they weren't online as incident response ramped up, and weren't as present and ready to flag that their clusters were having issues and needed attention. And second, the norms of "how loudly to advocate for your needs" are different between the teams running in some of those other clusters and our metrics teams – the responders who *were* online from our metrics teams just weren't very loud about raising their issue, and they got missed in the chaos of such a large response until later.

Another interesting feature of this graph that you might notice is that some clusters were restarted a *lot* faster than others (each color in the graph is a cluster). How quickly the restarts happened for a different cluster mostly came down to the personality of the team member restarting the cluster – some folks are very slow and cautious, and some are comfortable saying "let's restart everything fast!"

Ad Hoc Response

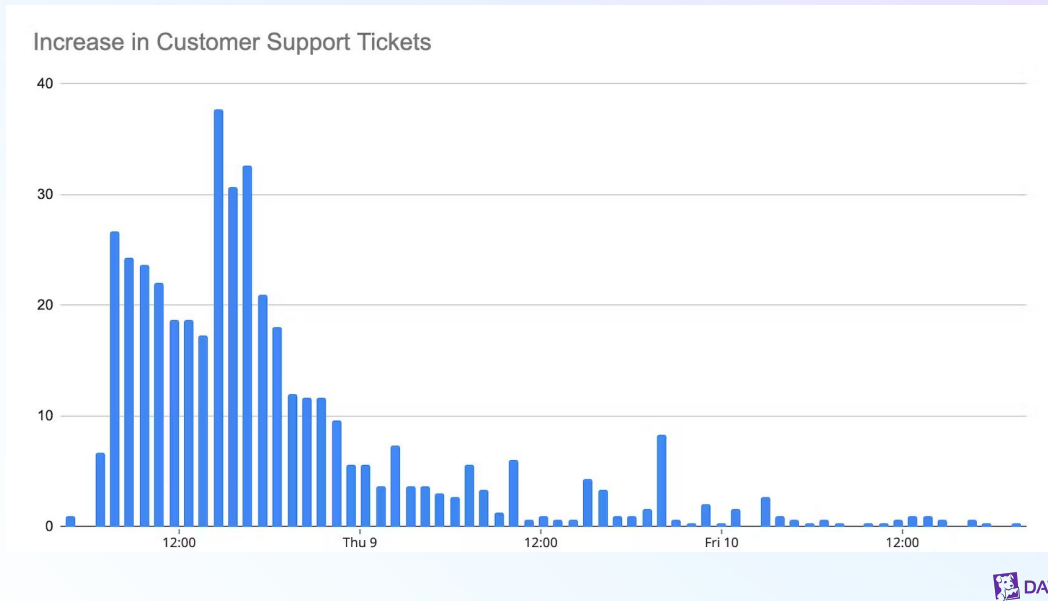
- Problem solving at all levels
 - Coordinating ~= scripting
- Using our major response rotation
 - One overall IC
 - One communications lead
 - One “roving troubleshooter”



One thing I want to highlight is that a lot of this response coordination was at least a little bit ad hoc. We built it live, as we needed it. I think that's normal and reasonable, because an incident like this one is very hard to predict and basically impossible to practice with any fidelity. That means your responders, and especially your group leading the response, need to be confident to build any organizational tools they need live and in the moment and make them work, the same way a product engineer/responder might need to write an ad hoc script during an unanticipated outage. The skills to develop in your organization are not “specific process for each thing”, but rather “confidence and authority to identify problems and fix them live, when we have to”. A second organizational piece we built as the incident happened was some extra uses of our major incident rotation. The last time DD had an incident anything like this large, the company was much smaller, and the major incident rotation was fewer than 10 people. Over the last few years, both in support of better work-life balance and as part of planning for major events and continuity, we've expanded that rotation to about 30 people. That meant we had enough people we could use them as a semi-dedicated “expert response coordination” team. We found that basically three roles were really useful to us: a “traditional” IC as I discussed earlier, a communications lead to continue to make sure engineers (and the IC) understood the state of the world, and someone I'll call a “roving troubleshooter”. This person (with help/direction from the IC) basically looked for workstreams that were blocking recovery, or seemed to be struggling, and stepped in to get engineers back on track. Often that was by functionally taking over as workstream lead for a little while, but the overall role was “whatever we need to unblock this group”.

[Image by dooder](#) on Freepik

Shout out to support



I want to take a moment to recognize a group of responders who I think don't get enough love from SRE: our support team. One thing you may or may not know about DD is that when you send a message to support, an actual human being on our support team reads that message and does their best to get you an accurate answer. That's in addition to customers who have some form of premium support, with a dedicated point of contact. Obviously, as a customer, this outage was a big deal to you, so a huge volume of customers reached out, and our support org did an incredible job of responding with as much information as we had throughout. The chart here is normalized so that our regular rate of support tickets is =1; you can see we saw a 25-35x increase in customers reaching out as part of this incident.

What did execs do?

- Look calm
- Compose customer messaging
- Take customer calls
- Approve expenses
- Remind people to rest



Johan Andersen, Datadog VP of Infra and SRE

During this outage, we had around 10 engineering execs involved (not generally all at once). It can be a little hard to visualize what they were up to if it **wasn't** trying to make decisions about response (which they generally weren't doing), so I want to give a little color to that. Firstly, they projected calm. A lot of engineers were obviously very stressed and worried – being able to look at your leadership and see that they're not panicking can make an enormous difference in those engineers' ability to respond well. Our execs also composed and approved messaging to customers, making sure that we were sharing details that were both accurate and made sense for what customers needed. For larger customers, they frequently got on calls both to give that message personally and to make sure customers could see that yes, we were absolutely responding to this incident as quickly as we could. They also made extremely clear to engineers that folks needing to do things like expense cabs, or dinner, or anything related to the fact that they were working these sudden emergency hours, could do so. And finally they helped remind individual responders to rest and hand off.

Image: Johan Andersen, Datadog VP

Speaking of handoffs

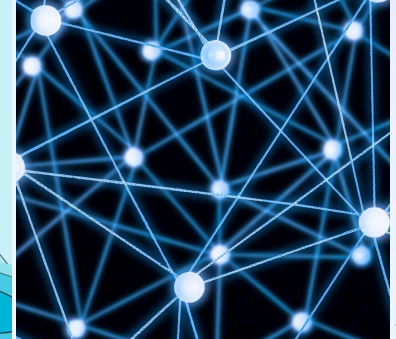
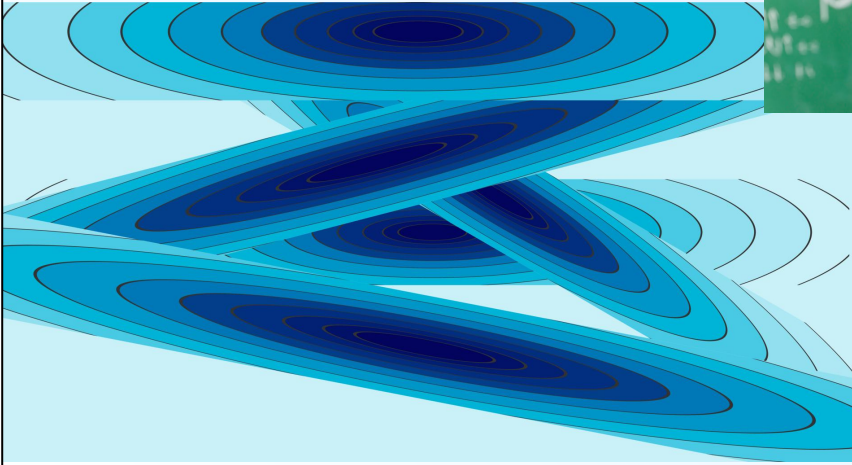
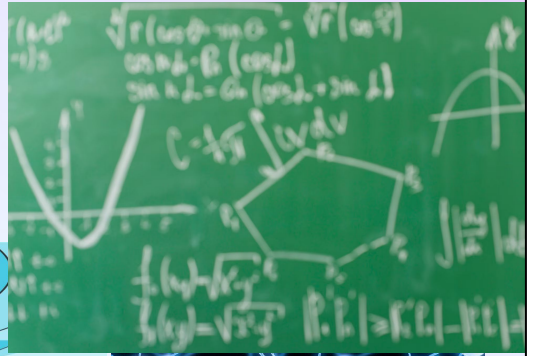


We don't have the tooling to track this so I can't guarantee it, but I did some asking various folks in the company, and I wasn't able to find any individual responder who was actively involved in the incident for more than 8 hours at a time. Obviously some of our ability to do that is based on having a very deep "bench" of engineers able to work on their systems, and some was because execs (and ICs) were actively encouraging people to go home, get rest, etc. I also want to call out the value of leading by example. Remember I said that the original page for this incident came in about 1:20am for me. By about 5:30, we understood the basic cause of the outage, had a start on how to recover, and had identified that the different regions on different cloud providers were in very different recovery states. We decided that the right path forward at that point was to split into two responses for our largest two regions and recover them separately. And I thought about coordinating splitting out that response, getting everyone lined up, and directing priorities and my honest reaction was to very nearly burst into tears. Remember, I haven't slept at all, and I've been coordinating the response for a global outage through this frankly awful uncertainty for the last 4 hours – I'm overwhelmed, and I can tell if the response gets any more complex than it already is, I'm going to start dropping important balls. Y'all, there is no need to be a hero. In fact, if you take that moment of overwhelm to hear yourself and realize you need to hand off... you can sell it as awesome leading by example to a big room of people later ;)

Image by [Freepik](#)

But that's just the first layer!

Can you turn it off and back on again?



And that's the first ~12 hours of our 48 hour outage. Once our infrastructure was back up and running, we obviously still had to recover all of the services that run on top of that infrastructure stack... and as you are probably unsurprised to hear, that involved discovering some surprising interdependencies and systems that didn't turn off or on very gracefully. Teams had to recover from backups, fast forward through backlogs (and then scale up to process those backlogs), find solutions for problems with data completeness, and do it all with still extremely limited or no visibility into the state of their products. As with most complex systems... it's very hard to turn off and back on again!

[Complex Vectors by Vecteezy](#)

Chalkboard Image by [Freepik](#)

Network [Image by rawpixel.com](#) on Freepik

Let's never do this again (aka Lessons Learned)

- Common infrastructure is inherently global, even when it's not
- Our coordination processes work, though there's always room for improvement
 - Making them work well is built from a long-term process of trust, blamelessness, and practice
 - Ad hoc process building is an important skill
- Scale is always a challenge
- Unchecked, systems grow surprising dependencies that are often discovered when they break

Many thanks to Partners



Thank you!

Blog posts: <https://www.datadoghq.com/blog/engineering/>

We're hiring! <https://www.datadoghq.com/careers/>

silverrose@datadoghq.com
laurent@datadoghq.com

@38:00