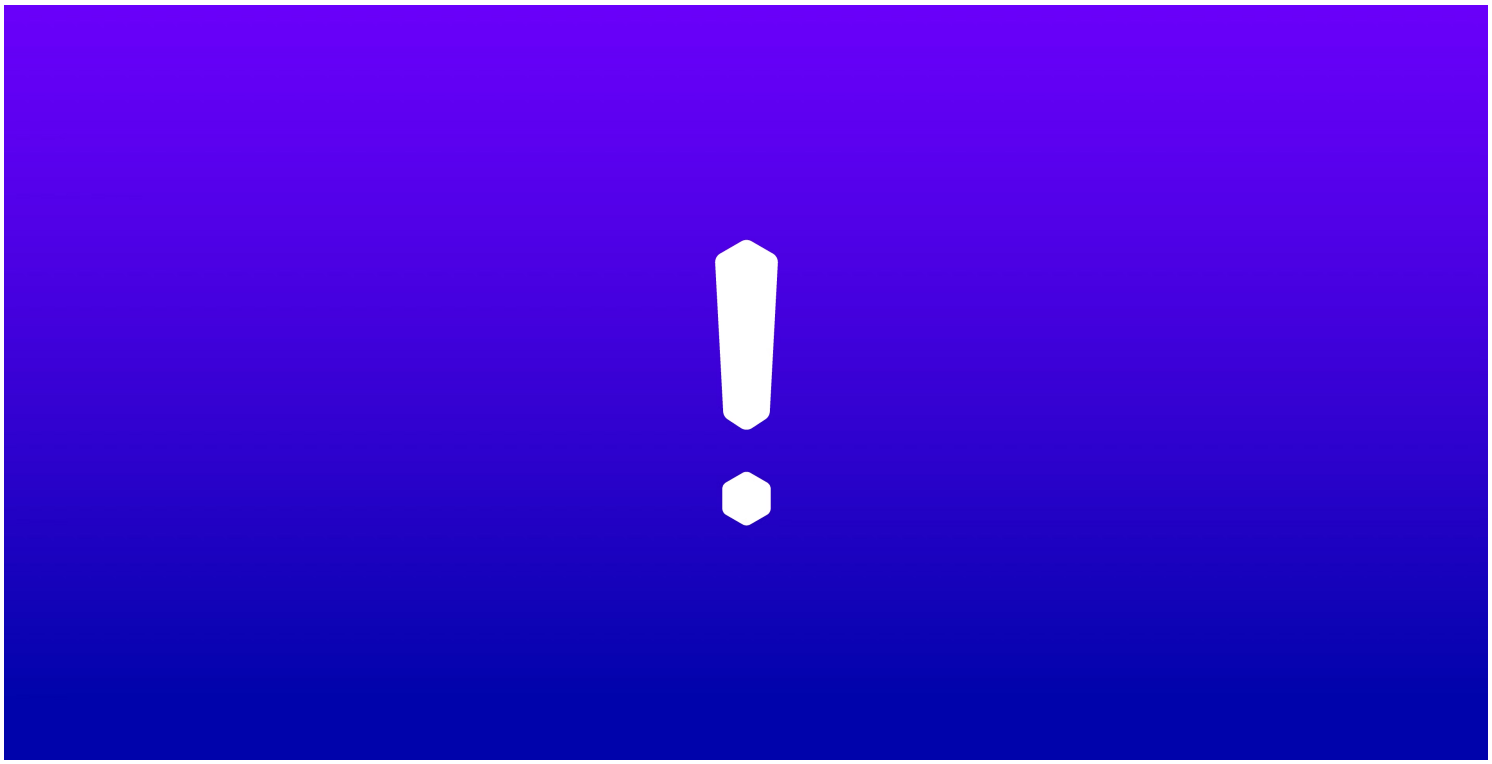
2023-03-08 Incident: Infrastructure connectivity issue affecting multiple region

Alexis Lê-Quôc

Published: May 16, 2023



company news

Impact, chronology, and response

Incident and impact

Starting on March 8, 2023, at 06:03 UTC, we experienced an outage that affected the US1, EU1, US3, US4, 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 at the beginning of the outage.

Chronology

We began our investigation immediately, leading to the first status page update indicating an issue at 06:31 UTC, and the first update indicating potential impact to data ingestion and monitors at 07:23 UTC.

We started to see initial signs of recovery at 09:13 UTC (web access restored) and declared our first major service operational by 16:44 UTC. We continued our recovery efforts, 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.

Response

We were first alerted to the issue by our internal monitoring, three minutes after the trigger of the first faulty upgrade (March 8, 2023, at 06:00 UTC). We declared an internal high-severity incident 18 minutes into the investigation, and we made it a public incident shortly thereafter, at 06:31 UTC.

Our incident response team had an emergency operation center of 10 senior engineering leaders, about 70 local incident commanders and a pool of 450 to 750 incident responders active throughout the incident, which required four shifts to bring the incident to full resolution.

Our communications lead orchestrated about 200 status updates across four regions, and involved hundreds of support engineers in constant communication with our customers around the clock.

Mitigation and remediation proceeded as follows:

1. **We focused our initial efforts on restoring normal ingestion and processing for real-time telemetry data** so that our customers could use the platform, even with limited access to historical data.
2. **Once real-time data was usable, we switched our recovery efforts to historical data** that had been ingested but left unprocessed at the beginning of the outage.
3. Simultaneously, we started providing frequent updates on the progress on our status page to keep customers apprised of the situation.

Root cause

We have identified (and since disabled) the initial trigger for the outage: **on March 8, 2023, at 06:00 UTC, a** security update to systemd **was automatically applied to a number of VMs, which caused a latent adverse interaction in the network stack (on Ubuntu 22.04 via systemd v249) to manifest upon systemd-networkd restarting.** Namely, `systemd-networkd` forcibly deleted the routes managed by the Container Network Interface (CNI) plugin (Cilium) we use for communication between containers. This caused the affected nodes to go offline.

The aggravating factor is that neither a fresh node nor a rebooted node exhibit this behavior because during a normal boot sequence `systemd-networkd` always starts before routing rules are installed by the CNI plugin. In other words, outside of the `systemd-networkd` update scenario on a running node, no obvious test reproduces the exact sequence.

What immediate effect did it have? This affected tens of thousands of nodes in our fleet between 06:00 and 07:00; by 06:31 UTC, enough of our fleet was offline that the outage was visible to our customers.

Why did the update automatically apply in five distinct regions that span dozens of availability zones and run on three different cloud providers? Whenever we push new code, security patches or configuration changes to our platform, we do so region-by-region, cluster-by-cluster, node-by-node. As a matter of process, we don't apply changes in place; rather, we replace nodes and containers in a blue/green fashion. Changes that are unsuccessful trigger an automatic rollback. We have confidence in this change management tooling because it is exercised at a large scale, dozens of times a day.

However, the base OS image we use to run Kubernetes had a legacy security update channel enabled, which caused the update to apply automatically. By design we use fairly minimal base OS images so these updates are infrequent and until that day, weren't ever disruptive.

To compound the issue, the time at which the automatic update happens is set by default in the OS to a window between 06:00 and 07:00 UTC. Thus it affected multiple regions at the exact same time, even though the regions have no direct network connection or coupling between them. This made the scale of the impact markedly larger, and confounded the initial responders who, like the rest of us, were unaware of this very indirect coupling of behavior between regions.

Will it happen again? No. We have disabled this legacy security update channel across all affected regions so it will not happen again. We have also changed the configuration of `systemd-networkd` so that it leaves the routing table unchanged upon restart. Finally, we have audited our infrastructure for potential similar legacy update channels.

Disabling this legacy security update channel has no adverse impact on our security posture because it is redundant with the way we apply security updates today, which are handled via the

aforementioned change management tooling.

Recovery

The high-level steps of the recovery had to be sequenced because of the magnitude of the compute capacity drop:

1. First, for each region, **recover enough compute capacity** so that it could scale back to what it needed to process incoming data at the normal rate. In fact, working with each cloud provider, we scaled the compute capacity to a higher level than normal in order to accommodate the processing of real-time and historical data at the same time.
2. Once compute capacity was ready, **recover each service** in parallel, such that all services could be restored as soon as possible.

Compute capacity

The effects of losing the network stack were felt differently across cloud providers and that complicated our recovery efforts.

Some cloud providers' auto-scaling logic correctly identify the affected node as unhealthy but do not initiate the immediate retirement of the node. This meant that simply rebooting the affected instance was the correct recovery step and was much faster for stateful workloads that still had their data intact and led to an overall faster recovery.

Other cloud providers' auto-scaling logic immediately replaces the unhealthy node with a fresh one. Data on the unhealthy node is lost and needs to be recovered, which adds to the complexity of the task at hand. At the scale of tens of thousands of nodes being replaced at once, it also created a thundering herd that tested the cloud provider's regional rate limits in various ways, none of which was obvious ex ante.

Once we had raw compute capacity at our disposal, we had to pay attention to the order of recovery: each region's internal control plane, whose role is to keep all Kubernetes clusters healthy, had to be recovered before the hundreds of Kubernetes clusters that power the platform could be repaired. When the regional control plane was fully restored, the various compute teams could fan out and make sure that each cluster was healthy enough to let the services running on it be repaired by other teams.

Service recovery

Each of our services had different recovery steps but they all share enough similarities to be presented here. In all cases, our number one priority was to restore the processing of live data.

Most of our services use one system to intake and serve live data (meaning the last 15 minutes or the last 24 hours, depending on the product) and a separate system to serve historical data. All systems were affected to a different degree but that separation allowed us to bring our products to a functioning state earlier in the recovery process.

Another common theme that all teams encountered during the service recovery was the fact that distributed, share-nothing data stores handle massive failures much better than most distributed data stores that require a quorum-based control plane. For example a fleet of independent data nodes with static shard assignment degrades roughly linearly as the number of nodes drops. The same fleet of data nodes, bound together by a quorum will operate without degradation so long as the quorum is met and refuse to operate once it's not. Of course the quorum-based fleet is a lot easier to manage in the day-to-day, so going one way or another is not an obvious decision, but this outage highlights the need for us to re-examine past choices.

Generally speaking, our primary telemetry datastores (e.g., those that store point values and raw log data) are statically sharded, while various metadata (e.g., those that index and resolve tags) is stored in systems that require a quorum to operate. Thus, much of the recovery focused on restoring these metadata stores to the point that they would accept new writes for live data, while a smaller team was able to concurrently repair the independent static shards that indexed and stored live data.

Lessons learned

The following is not the sum total of everything we learned during the incident but rather important themes that pertain to the whole platform.

This was our first global incident. We built Datadog regions with the explicit design goal of being autonomous and resilient to local failure. This is why they are built across multiple zones on completely independent cloud providers without any direct connections to one another. This is also why we have no global, shared control plane. And for good measure, each service has, in each region, failovers, active-active setups across availability zones, and extra capacity to deal with spikes during recovery. Still, we failed to imagine how they could remain indirectly related, and that miss will inform how we continue to strengthen our foundational infrastructure.

We heard time and time again that there is a clear hierarchy among the data we process on our customers' behalf. **Most important, usable live data and alerts are much more valuable than access to historical data.** And even among all the live data, data that is actively monitored or visible on dashboards is more valuable than the rest of live data. We will take this clear hierarchy into account in how we handle processing and access in degraded mode.

We run chaos tests (and are continuously subjected to a certain degree of chaos from our cloud providers) but we fell short in considering the scale of degradation we need to consider in our tests. **Going forward we will use the level of infrastructure disruption we saw to prove that the platform can operate in degraded conditions without being completely down.** Coupled with the previous point, this may take the form of having only urgent data accessible and processed in degraded mode.

Even with separate systems that serve live data when historical data is not accessible (e.g., APM's Live Search and Log Management's Live Tail), **we did not provide our customers with clear guidance on how they could access our systems** as soon as they were available, even while the broader incident was still being remediated. Going forward, we will prioritize sharing guidance with our customers about which services and data are available and how to access them.

Our communication during the outage improved in precision and depth as time passed but **we realize the inadequacy of the status page to communicate nuanced product status.** We will devise a way to communicate in real-time a detailed status for each product, even when we run the platform in degraded mode.

In closing

This outage has been a humbling experience for everyone involved at Datadog. We all understand how important uninterrupted service is to our customers and we are committing to verifiably improve the resilience of our services based on all that we learned during and after this outage.

FIND OUT HOW

FREE TRIAL

Download mobile app

Download on the App Store

GET IT ON Google Play

RESOURCES

Pricing
Documentation
Support
Learning Center
Certification
Open Source

Resources
Webinars
Security
Privacy Center
Knowledge Center
Learning Resources

PRODUCT

Features
Infrastructure Monitoring
Container Monitoring
NPM
NDM
Serverless
Cloud Cost Management
Cloudcraft
Log Management
Sensitive Data Scanner
APM
Error Tracking
Continuous Profiler
Data Streams Monitoring

Browser Real User Monitoring
Mobile Real User Monitoring
Synthetic Monitoring
Continuous Testing
Session Replay
Application Vulnerability Management
Application Security Management
Cloud Security Management
Cloud SIEM
Bits AI
Workflow Automation
CoScreen
Dashboards
Watchdog

Database Monitoring
CI Visibility
Service Catalog
Dynamic Instrumentation
Universal Service Monitoring

Alerts
Incident Management
Integrations
API

ABOUT

Contact Us
Partners
Press
Leadership
Careers

Legal
Investor Relations
Analyst Reports
ESG Report
Vendor Help

BLOG

The Monitor
Engineering

Pup Culture
Security Labs

 Terms | Privacy | Cookies

日本語