crablab Hugh — 3C95 — Oct '17

We're all liberal Guardian readers on here 😁

LOOKE: I liked the Guardian for it's investigative journalism, idk about other people)

2 Replies — 2

Chepuya Kevyn Regular — 1 — 3C95 — Oct '17

I never said they were good articles 😂 . The point given was it was widely reported through the press. Broadsheets? Not so much. But it was reported throughout the press and these two examples and were also not behind a paywall.

2

mark1 Mark Edmonds Crowdfunding Investor — crablab — Oct '17

I used to like them years ago, but now they just whine about everything. I actually blocked their articles on Apple News as they was so irritating.

Chad — Oct '17

Wow

HSBC having a glitch

Now that's something

1

DaveTMG Crowdfunding Investor — crablab — Oct '17

The grauniad, unfortunately, has started posting these 'articles' that are basically shills for a book the author is peddling. Tedious.

oliver Oliver Beattie Monzo — oliver — Oct '17

Hi everyone 👋 I'm Monzo's Head of Engineering, and as I promised on Friday I'd like to share some more information about what happened during this outage. Because the nature of the issue was technical, this post is also quite technical. 🤓

It's important to note that we had two major incidents last week that many of you will have experienced (sorry again.) The first incident lasted most of the week and affected only our prepaid product – ie. Monzo Alpha and Beta cards. The second outage affected both the prepaid product and our new current account for a period of around 1½ hours on Friday afternoon. This post is about the latter.

You can learn more about our overall backend architecture in this blog post (⏱8) I published last year, but it's important to understand the role of a few components in our stack at a high level to understand this issue:

- Kubernetes (⏱8) is a system which deploys and manages all of our infrastructure. Monzo's backend is written as several hundred microservices, packaged into Docker containers. Kubernetes manages these Docker containers and ensures they are running properly across our fleet of AWS nodes.
- etcd (⏱8) is a distributed database used by Kubernetes to store information about which services are deployed, where they are running, and what state they're in. Kubernetes requires a stable connection to etcd in order to work properly, although if etcd does go down all of our services do continue running – they just can't be upgraded, or scaled up or down.
- linkerd (⏱8) is a piece of software that we use to manage the communication between all of the services in our backend. In a system like ours, thousands of network calls are happening every second, and linkerd does the job of routing and load balancing all of these calls. In order to know where to route these calls, it relies on being able to receive updates about where services are located from Kubernetes.

Timeline

- **Two weeks before:** The Platform team makes some changes to our etcd cluster to upgrade it to a new version, and also to increase the size of the cluster. Previously, this cluster consisted of three nodes (one in each of our three zones (⏱8)); we raise this to nine (three in each zone.) Because etcd relies on being able to achieve a quorum (⏱8) to make progress, this means that in this setup we can tolerate the simultaneous loss of an entire zone and a single node in another zone.

  This upgrade went according to plan, and didn't involve any downtime. We're satisfied that this cluster is behaving correctly, but this context is important as it triggers a bug in another system later.
- **One day before:** A team developing a new feature for the current account deploys a new service to production, but notices that it is experiencing issues. As a precautionary measure, they scale the service down such that is has no running replicas, but the Kubernetes service (⏱8) still exists.
- **14:10:** An engineer deploys a change to a service needed to process payments for the current account. Making changes is not unusual and is something our engineers do all the time: to minimise the risk of changes we make them small and frequent, using a repeatable, well-defined process to release them. When this service was deployed however, all requests to it started to fail. **This is when current account customers started experiencing payment failures.** At this time, the prepaid card is not affected as it does not use the broken service.
- **14:12:** The change is rolled back. This is standard practice for deployments that don't go according to plan, and when interfaces are changed they are backwards- and forwards-compatible to ensure that rolling back is a safe operation. However, in this case even after rolling back, the errors persisted and payments remained unavailable.
- **14:16:** We declare an outage internally. Members of the team start to convene to establish the impact of the problem and start to debug it.
- **14:18:** Engineers identify that linkerd appears to be in an unhealthy state, and attempt to use an internal tool designed to identify individual nodes that are experiencing problems and restart them.

  As described earlier, linkerd is a system which we use to manage communication between our backend services. To know where to send a particular request, it takes a logical name like `service.foo` from the request and turns it into an IP addresport. In this case, linkerd had not received an update from Kubernetes about where on the network the new pods (⏱8) were running. As such, it was trying to route requests to IP addresses that no longer correspond to a running process.
- **14:26:** We believe that the best path forward is to restart all linkerd instances in our backend, of which there are several hundred, under the assumption that they are all experiencing the same issue. In parallel, many engineers are attempting to minimise the customer impact seen making card payments or receiving bank transfers by activating internal processes designed to provide backup when we are experiencing problems. This means that most customers are still able to use their card successfully despite the ongoing instability.
- **14:37:** Replacement linkerd's cannot start because the Kubelet that runs on each of our nodes is failing to retrieve the appropriate configuration from the Kubernetes `apiservers` (⏱8). At this point, we suspect an additional issue with Kubernetes or etcd and restart the three `apiserver` processes. When this is complete, the replacement linkerd instances are able to start successfully.
- **15:13:** All linkerd pods are restarted, but services that process thousands of requests per second are now receiving no traffic. At this point, customers are completely unable to refresh their feed or balance in the Monzo app and our internal COps ("Customer Operations" 💁‍♀️) tools stop working. **The issue has now escalated to a full platform outage, and no services are able to serve requests.** As you can probably imagine, practically all of our automated alerts started triggering. 🚨🔥
- **15:27:** We notice that linkerd is logging `NullPointerException` (⏱8) when it is attempting to parse the service discovery response from the Kubernetes `apiserver`. We discover that this is an incompatibility (⏱8) between the versions of Kubernetes and linkerd that we're running, and specifically is a failure to parse empty services.

  Because we have been testing an updated version of linkerd in our staging environment for several weeks which contains a fix for the incompatibility, engineers from the Platform team begin deploying a new version of linkerd in an attempt to roll forward.
- **15:31:** After inspecting the code change, engineers realise that they can prevent the parsing error by deleting Kubernetes services which contain no endpoints (ie. the service mentioned earlier that was scaled down to 0 replicas as a precautionary measure.) They delete the offending service and linkerd is successfully able to load service discovery information. **At this point, the platform recovers, traffic starts transiting between services normally, and payments start to work again. The incident is over.** 😅

Root cause

At this point, while we'd brought our systems back online, we did not yet understand the root cause of the problem. The network is very dynamic in our backend because of deployment frequency and automated reaction to node and application failure, so being able to trust our deployment and request routing subsystems is extremely important.

We've since found a bug (⏱8) in Kubernetes and the etcd client that can cause requests to timeout after cluster reconfiguration of the kind we performed the week prior. Because of these timeouts, when the service was deployed linkerd failed to receive updates from Kubernetes about where it could be found on the network. While well-intentioned, restarting all of the linkerd instances was an unfortunate and poor decision that worsened the impact of the outage because it exposed a different incompatibility (⏱8) between versions of software we had deployed.

Remarks

A large scale failure in a distributed system can be very difficult to understand, and well-intentioned human action can sometimes compound issues, as happened here. When things like do happen, we want to learn as much as possible from the event to ensure it can't resurface. We've identified several steps we'll take in the short-term:

1. Fix the bug in Kubernetes that can trigger timeouts following a cluster reconfiguration.
2. Roll out a new version of linkerd that fixes the parsing error.
3. Create better health checks, dashboards and alerts for the components impacted to surface clearer signals about what is wrong and prevent human error.
4. Improve our procedures to ensure we communicate outages internally and externally as clearly and quickly as possible.

I want to reassure everyone that we take this incident very seriously; it's among the worst technical incidents that have happened in our history, and our aim is to run a bank that our customers can always depend on. We know we let you down, and we're really sorry for that. I hope that this post-mortem gives some clarity on what happened and what we're doing to make sure it doesn't recur. I'll make sure we post something similar for any other incident of this severity; if I were a customer I'd want to know, and also I personally find this kind of post fascinating as an insight into production systems. Do let me know if you have any questions. 🙏

13 Replies — 113

🔗 Anatomy of a Production Kubernetes Outage - presentation 📹 ⑧
🔗 Community Digest 3/11/17 (⏱)
🔗 RESOLVED: Monzo Services Degraded/Outage - Card Payments may fail (16/01/18) ⑧
🔗 Monzo Stability ⑧

tomer — Oct '17

What an absolutely fascinating read. Thanks for such detailed insight into the inner workings (or not 😂) of Monzo. 👍

10

tomer — oliver — Oct '17

The three services you mention - are these products that you subscribe to or purchase outright? Are they physically on local servers or purely in the cloud? Do they force upgrades on you?

2 Replies

Johnny — Oct '17

Some organisations try to gloss over things that went wrong. This sort of explanation only gives me greater confidence in Monzo.

2 Replies — 17

anon72173902 — Oct '17

I was gonna mention that but I am sure someone is watching 👀

DaveTMG Crowdfunding Investor — tomer — Oct '17

kubernetes is based on the platform that runs google

1

oliver Oliver Beattie Monzo — tomer — Oct '17

All three of these components – like the vast majority of our backend – are free, open-source software. Often these projects have started by or are built upon technology used by large internet companies. Upgrades are totally within our control, but we generally try to run within a few versions of the latest. 🆕

7

anon38274058 — Oct '17

I'm with @Johnny on this one. Thank you very much with the detailed explanation.

I have a question if it may?

Following the impact that this change had on the service, are there any plans to run potential service affecting changes "out of hours"? Not that there is any such thing in the banking world really but between the hours of 10pm - 6am I would imagine the traffic would be lower?

Thanks!

1 Reply — 2

danfury — Oct '17

As someone that works on backend systems using Docker etc, that was a fascinating read! I love how open, detailed and technical it was 👌

5

Avishai Marta Coral Crew — 2 — Oct '17

Thaaank you so much for sharing. I'll gonna read it few times later on, just so I'm sure I understood it all. 😂

1 Reply — 3

crablab Hugh — oliver — Oct '17

This is a superb insight and I really appreciate you writing it 👍

As someone that has used Docker/Kubernetes (on a much smaller level) I can see how when scaled up in can be very difficult to identify exactly where you have an issue - especially when the breaking change doesn't manifest itself immediately.

3