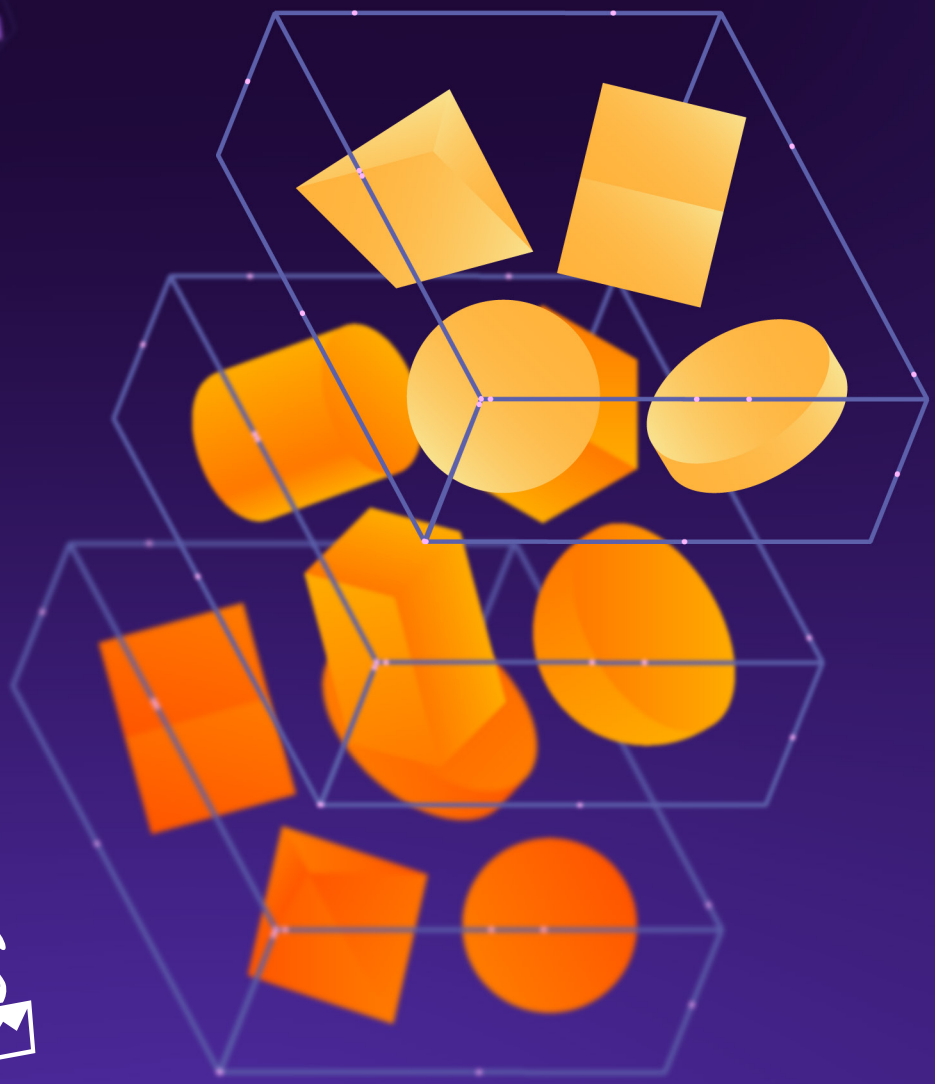
How MoneySupermarket drives cloud cost efficiency with performance monitoring

THOMAS DITTMER

Head Architect, TravelSupermarket
MoneySupermarket Group



DATADOG

Travel Supermarket

Moneysupermarket Group

About the Author

THOMAS DITTMER

Thomas Dittmer is Head Architect for TravelSupermarket at MoneySupermarket Group, a large UK-based price comparison business and a member of the FTSE 250 Index. He is responsible for driving best practices internally and continuously improving the efficiency of TravelSupermarket's product engineering organisation. Thomas has nearly 15 years of experience in software development, agile delivery, and leading technical teams.

Table of Contents

03 **Introduction**

04 **Rightsize the production estate**

05 Resize resources

05 Switch off unnecessary resources and move to more efficient architectures

06 Pre-purchase capacity

07 **Eliminate extraneous testing environments**

07 Testing outside of production is a false safety net

07 Multiple environments are a burden on the team

08 **Monitor in production**

08 From fewer, larger deploys to many smaller deploys

08 Drive towards faster incident resolution

09 Favour forward fixes over rollbacks

09 Monitor partners and third parties

10 **Accompanying team structure, cultural, and process changes**

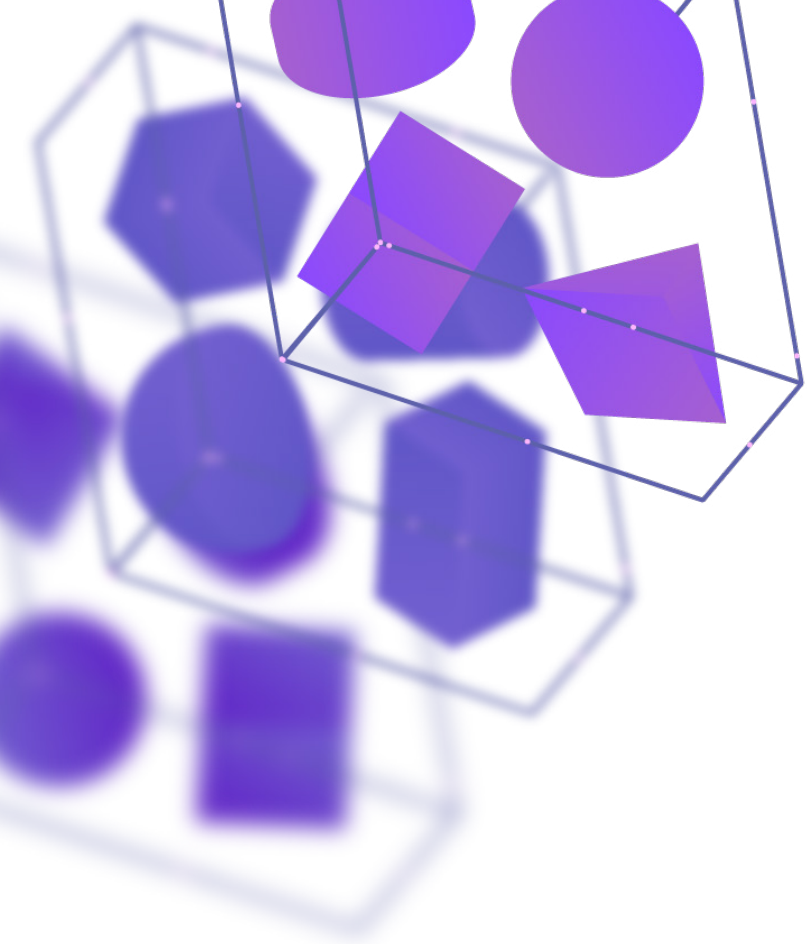
10 From a technology to a product orientation

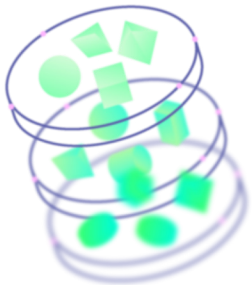
11 End-to-end responsibility for services

11 From specialists to generalists

12 **Conclusion**

13 **Acknowledgements**





Introduction

Cloud infrastructure sprawl sneaks up on organisations over time, through a series of individual decisions that in aggregate become inefficient.

So when we were presented with an opportunity to review some of our operational costs, cloud infrastructure was very much the place to start. MoneySupermarket Group PLC is a strategic UK client for AWS, operating with a material footprint of IaaS instances. As a result, optimising the use of these resources in both development and production is a key consideration in our cost management.

We knew we could be more efficient and that the growth in costs was driven by our tooling, processes, organisational structure, and mindset. But first, we had to ask for more money; as the saying goes, "You have to spend money to make money." But sometimes you also have to spend money now in order to save money later.

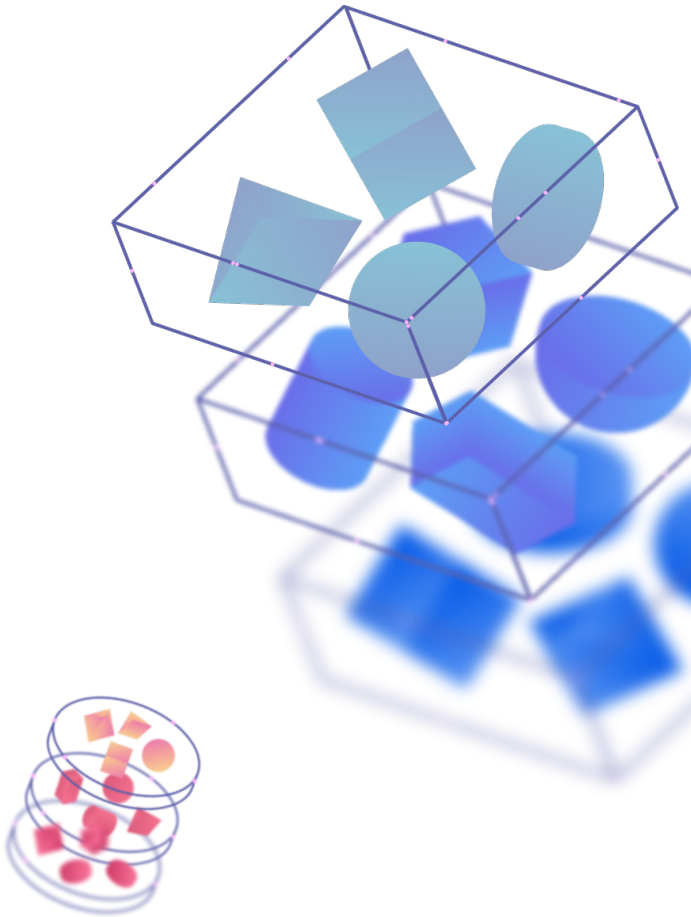
Therefore, 18 months ago, we invested deeply in full-stack performance monitoring capabilities with the goal of improving the efficiency of these IaaS costs. The sales pitch to the team was that any money saved would more than pay for the monitoring solution, and it would also provide wider benefits to the team.

At this point, we had some existing monitoring capabilities, but these legacy tools were fragmented and not very robust. In addition, production monitoring wasn't a significant part of our process; instead, we relied on extensive pre-production testing.

The introduction of more powerful monitoring and an emphasis on monitoring in production enabled us to fundamentally change our structure and delivery model. Among the business benefits we've realised are:

1. **A reduction of cloud infrastructure costs by 50%**
2. **Several percentage points of increased revenue due to lower latencies—a material advantage when it comes to a high-traffic, online metasearch business**
3. **Decreased squad cycle time, fewer incidents, and faster fixes**

In the sections below, I explain how our investment in performance monitoring in production enabled our organisation to transform our processes, organisational structure, and mindsets, which in turn significantly enhanced our cost efficiency in the cloud.

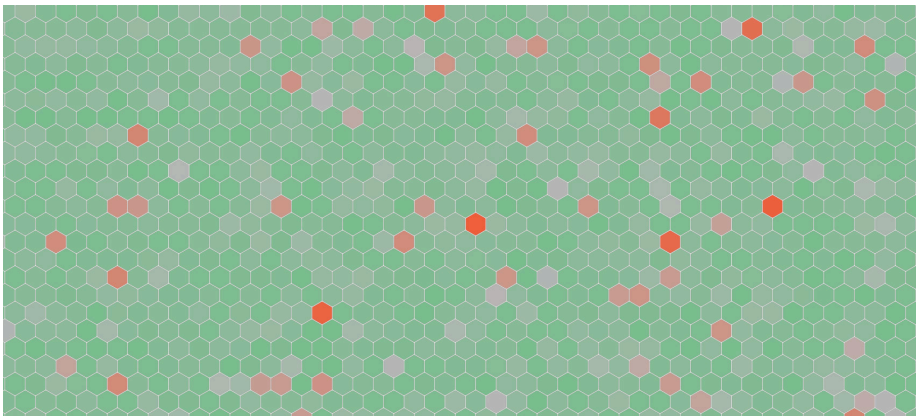


Rightsize the production estate

We started with the low-hanging fruit: understanding our usage and needs, and paying only for what we needed. There are three main tactics in this bucket: resizing resources, shifting to more efficient architectures, and pre-purchasing capacity.

Resize resources

On average, our CPUs were operating at just one or two percent utilisation. We discovered that we often overallocated resources ("overspec-ed" them) by selecting more powerful instances than our workloads required. By resizing instances and consolidating capacity, we were able to realise significant savings.

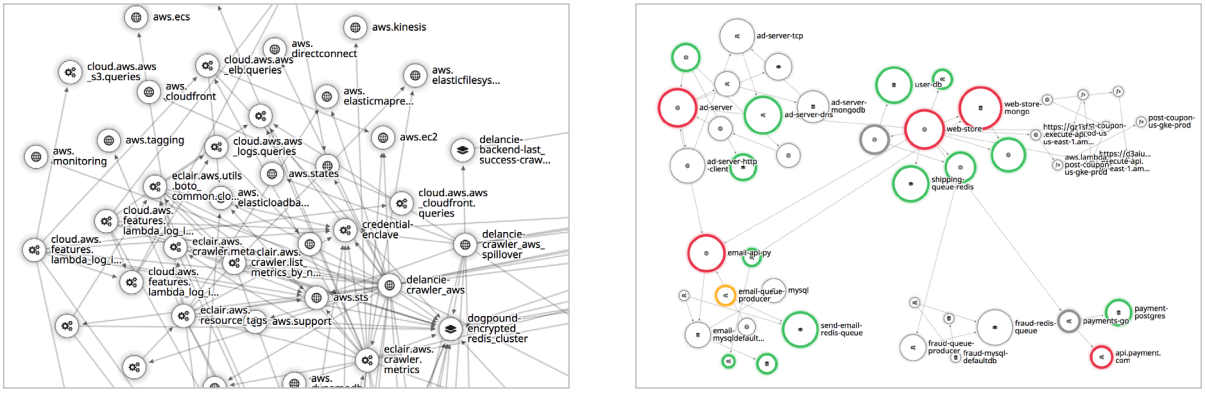


Instances coloured by CPU utilisation

Switch off unnecessary resources and move to more efficient architectures

Static architectural diagrams are almost always a poor representation of how systems are actually operating. Robust monitoring gave us a real-time architectural map, and helped us detect unnecessary tools, services, and even inefficient architectural patterns.

For example, we hosted a database solely for the purpose of reading individual records. To save money, we switched to Amazon S3, a solution that was far cheaper than running our own, in-house database clusters—and one that outsources resilience and operational requirements to AWS.

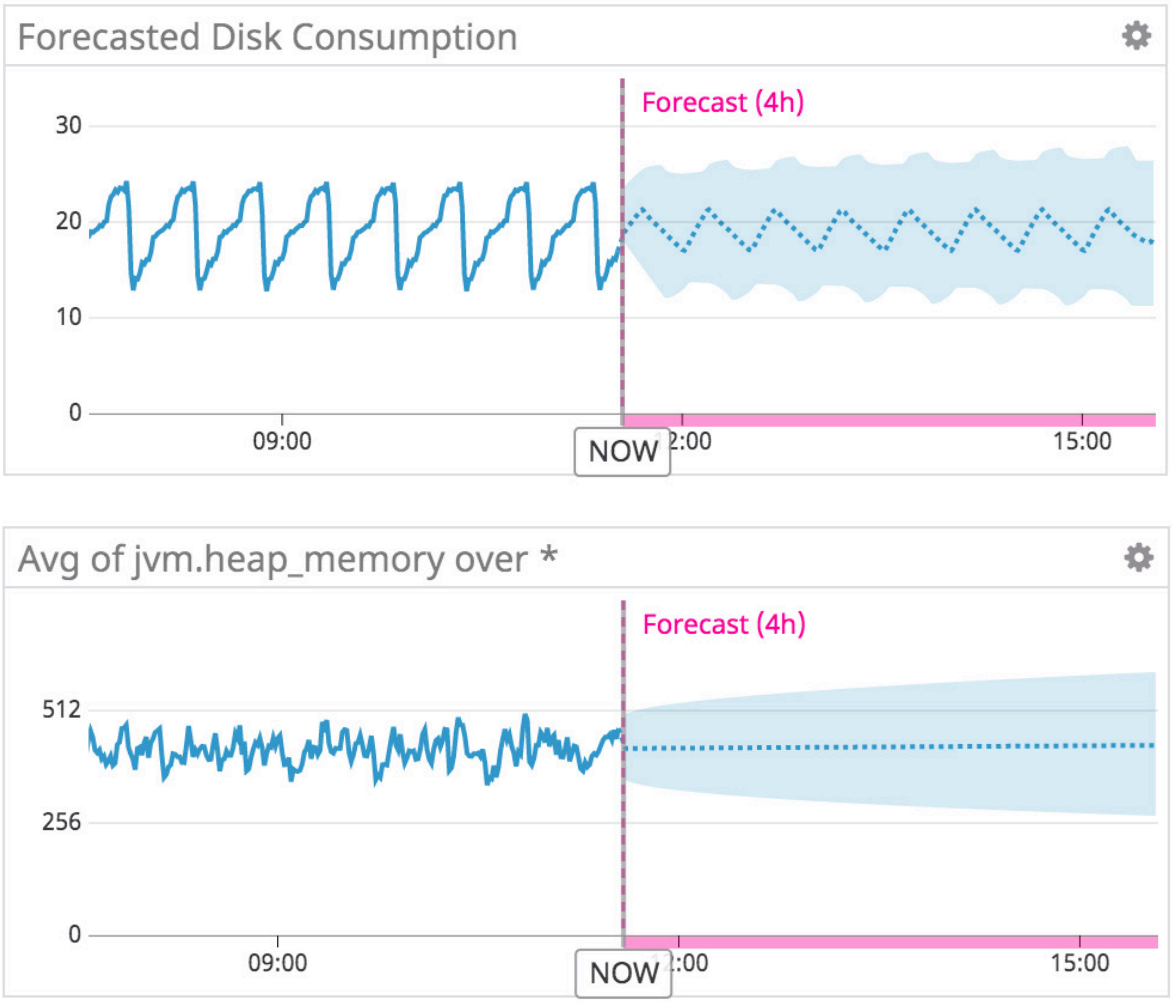


Before and after: Service Maps showing a complex, inefficient, and expensive architecture and a simpler, efficient, and cheaper architecture

Now, we're very cautious about introducing new tools, which could generate both direct costs and additional administrative overhead. Monitoring helps us rationalise our tools and services—and keep things in check.

Pre-purchase capacity

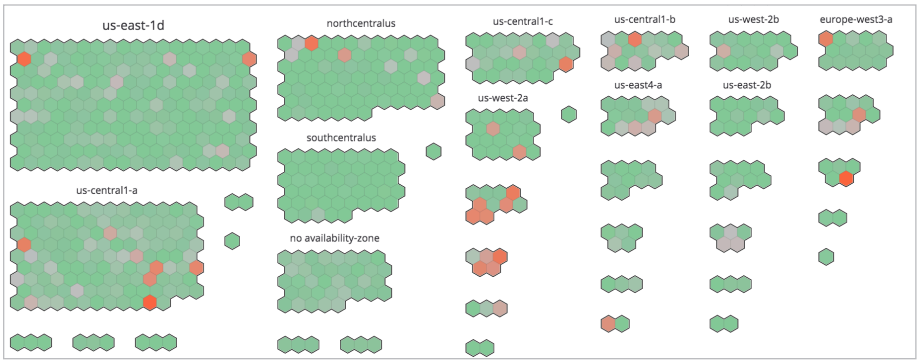
Robust monitoring helps us understand our current needs and predict our future needs. We can see exactly how much each instance is being utilised, as well as all instances in aggregate. It's extremely valuable to be able to see machine learning-based forecasts of expected demand based on our historical data. All this allows us to be smarter commercially, such as reserving instances upfront for large discounts.



Forecasting algorithms help us with capacity planning

Eliminate extraneous testing environments

We also reduced the number of development and testing environments. We realised we had lots of test environments and weren't really getting any value out of moving new code through different environments. In addition, operating so many different environments taught us two important lessons: (1) testing outside of production is a false safety net; (2) multiple testing environments create administrative and cost burdens.



Too many testing environments generate cost, inefficiency, and false confidence

Testing outside of production is a false safety net

We realised these artificial testing environments weren't giving us a good understanding of how code would perform in production. For one, we couldn't actually replicate our live traffic. Regression testing and other tests weren't giving us confidence and were a false safety net. Plus, having so many environments wasn't cost effective. As a result, we transitioned from pre-production testing to monitoring and alerting in production. Being able to alert proactively and watch anomalies gives us better confidence than running regression tests and allows us to streamline our testing to the higher-risk areas.

Multiple environments are a burden on the team

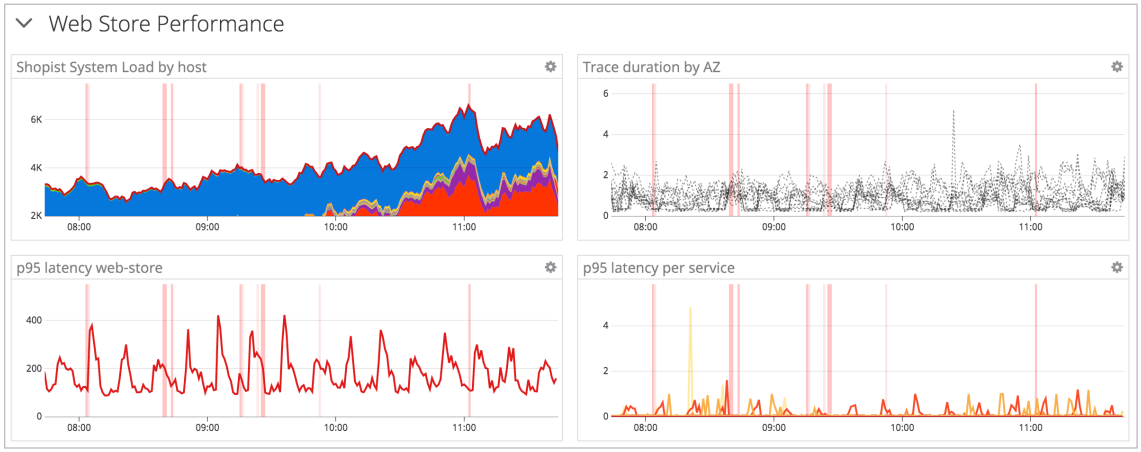
Our developers were frustrated with the complexities of deploying to multiple environments, especially when they still required a decent amount of manual configuration. Each environment is never technically the same—and it's hard to keep environments from drifting. The more environments you have, the more complex and costly things get. It's not just infrastructure costs: teams had to deploy to one environment, then the next one, then to production. This adds up to significant process cost and extra delivery cycle time. Removing extra steps, while keeping the same level of quality assurance, helped improve the efficiency of teams and reduce cost. Today, we just have one test environment and a production environment.

Monitor in production

Eliminating testing environments and some associated testing tools reduced cost, but the gap had to be filled. Our expanded monitoring capabilities enabled us to fill that gap. But we also needed to make changes to our delivery process.

From fewer, larger deploys to many smaller deploys

We focused on reducing the batch size of our deliverables. By making the smallest possible changes, we were able to reduce our cycle time, and easily isolate any possible issues that may arise. If something goes wrong with a small change, we can either roll it back quickly or make a forward fix. In addition, robust instrumentation also helps us learn about our architecture and identify ways we can safely deconstruct monolithic applications and move to simpler microservices patterns with fewer dependencies (and thus, fewer things that can go wrong).

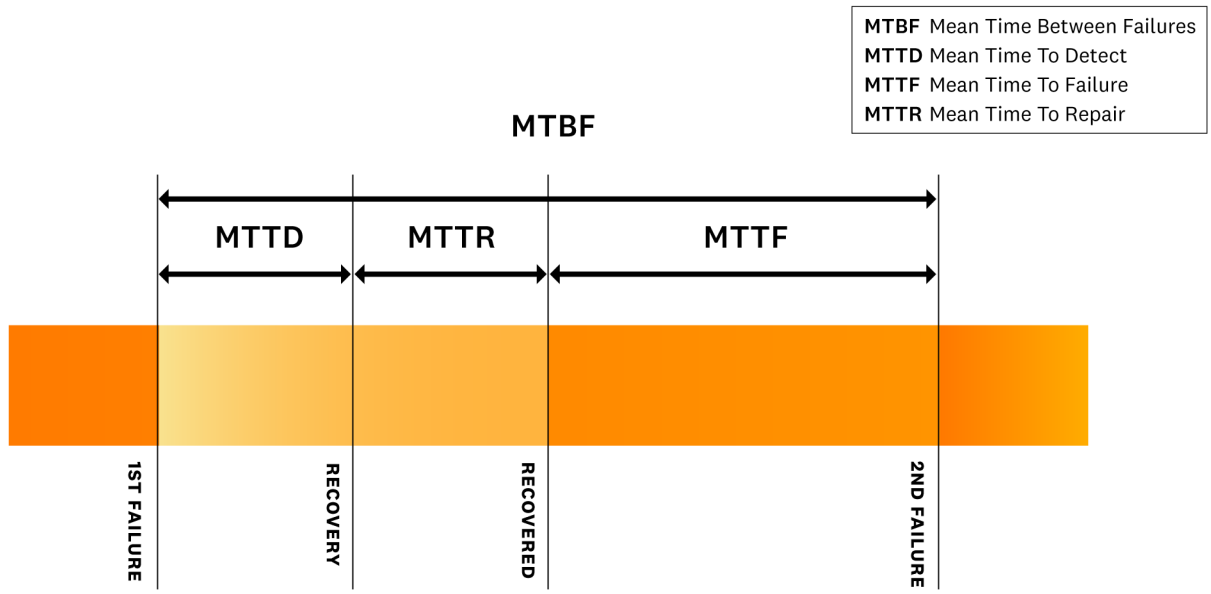


We can immediately understand the impact of each small deployment indicated by a pink bar

Drive towards faster incident resolution

While smaller deploys eliminate complexity, we also needed to empower teams to quickly resolve problems. In the past, our fragmented monitoring tools didn't make it easy, especially for teams under pressure, to fix problems in the middle of the night or on weekends. Investing in a highly integrated monitoring solution accelerated incident resolution time (and also made incidents less stressful for teams).

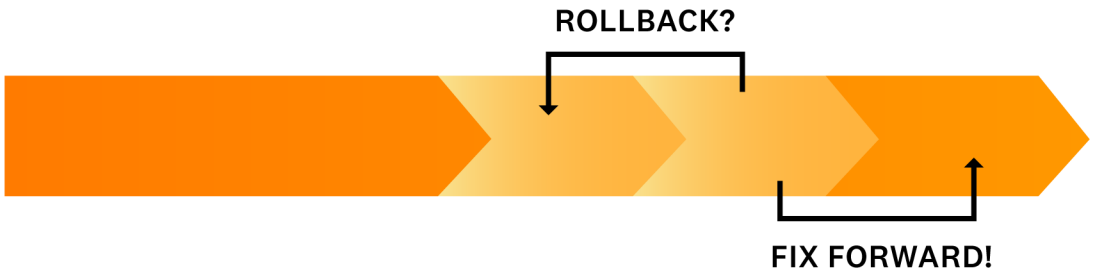
While we don't yet rigorously calculate MTTD (Mean Time To Detection) and MTTR (Mean Time To Repair), we do track our ticket count and the number of active P1, P2, and P3 tickets (these categories designate the priority level of each incident). The number of tickets resolved outside our desired resolution window (for example, within 60 days for P3 incidents) has decreased.



Favour forward fixes over rollbacks

More recently, we've adopted a bias for forward fixes rather than rollbacks. Forward fixes enable us to maintain the momentum of our delivery cycle.

Still, forward fixes are only possible if you're confident in your ability to quickly find and remediate the issue at hand. Robust monitoring helps us do precisely that: because we can seamlessly correlate logs, events, infrastructure metrics, and other monitoring data, we can also quickly identify and resolve the root cause of a problem.

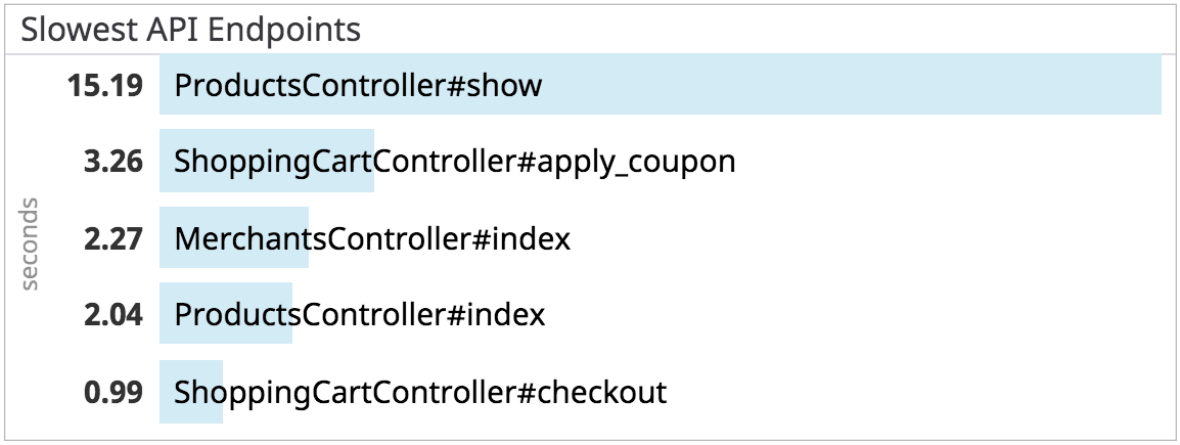


Monitor partners and third parties

TravelSupermarket's core product is search and comparison of holiday/vacation packages. For each search on our website, we call results from our partners. Depending on traffic and the time of day, some partners don't respond very quickly, and for users, their wait time will be affected by the response time of our slowest partner. Clearly, partner response times matter a lot to our business.

With our new monitoring capabilities, we started graphing all of our partners' response times, which gave us a good sense of our partner landscape. This data has enabled us to do three

impactful things: (1) reduce overall wait times because we can automatically stop waiting for partners who are down; (2) alert partners when they're down; and (3) work proactively with partners to help them reduce their response times and meet traffic demands. This work has generated a few additional percentage points of revenue—a very meaningful impact in a business like ours, as users quickly navigate away when partners are slow or stalled.



Slow or underperforming partners can be a drag on the business

Accompanying team structure, cultural, and process changes

In this section, I discuss some changes to our team, culture, and processes that might not seem directly related to cloud cost efficiency. But as I mentioned earlier, small individual choices over time—influenced by culture, processes, and mindsets—lead to cloud sprawl. The changes I describe below are absolutely essential to cost efficiency because they enable everything I've described so far.

In the past, we've had a fairly centralised team with a rigid division of labour focused on technology silos, such as frontend, middleware, and backend. Today, we're organised into squads aligned to key product areas, which primarily consist of technology generalists, rather than specialists. Production performance monitoring played an integral role in enabling this shift.

From a technology to a product orientation

Squads really own their product areas. They have a sense of purpose and mission, and can be very passionate about the customer journey. For example, we've got a squad focused on traffic, dealing with concerns such as getting users to the site, and bringing customers into the sales funnel: they're passionate about page speed and UX. Nobody on the team will pick work up unless they understand the business value and how the work will help us meet our overall business objectives.

End-to-end responsibility for services

For a product orientation to work, squads must have end-to-end responsibility for their services. They don't just write code and forget about it. They must set up their own monitoring and alerts, troubleshoot issues, maintain, and enhance their services. Robust, easy-to-use monitoring helps us enable these autonomous teams. Once a release is approved, practically everybody on the squad can deploy, and with monitoring, they can immediately know that they haven't broken anything.

From specialists to generalists

An end-to-end product orientation also means that our organisations aren't siloed, and that our teams mainly comprise generalists rather than specialists. Two elements are critical to enabling generalist squads: simplicity and robust monitoring. A simple stack means the squad doesn't get bogged down in technical minutia. And, if you don't know a stack very well, monitoring helps you understand it and deploy changes with confidence (and without breaking anything).

Two elements are critical to enabling generalist squads: simplicity and robust monitoring.

We've still got specialists in certain technology areas, but everybody's a bit of a generalist now. We've lost a few people because of these shifts; however, the people who remain like the challenge, the variety, and the customer-centric orientation of the work.

"I-SHAPED"
Expert at one thing

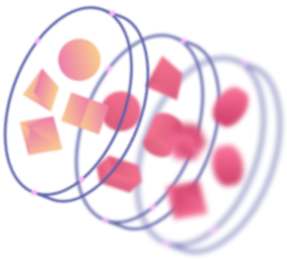
GENERALIST
Capable in a lot of things but not expert in any

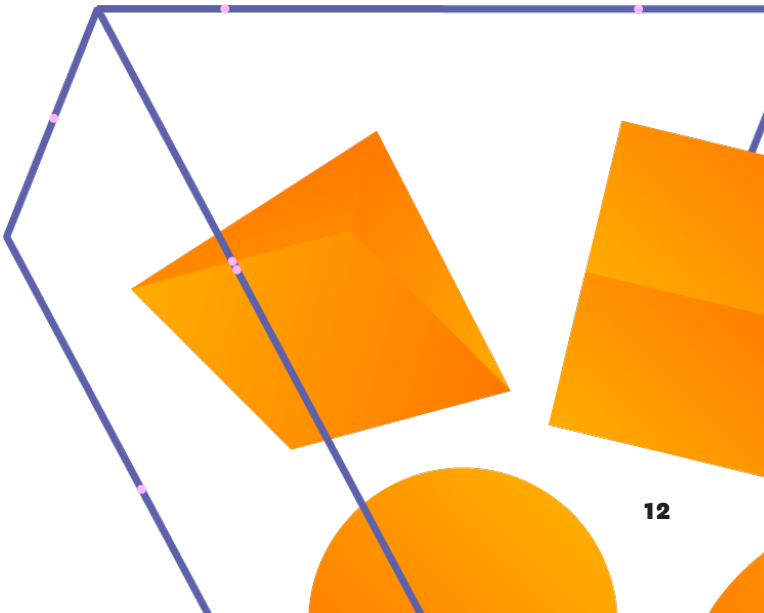
"T-SHAPED"
Capable in a lot of things and expert in one of them

Our team has gone from "I-shaped" to "T-shaped"

Conclusion

Robust cloud performance monitoring is a big factor that has enabled us to adjust our teams, processes, and technology to be more simple, lean, and efficient. Our evolved delivery model has generated some significant wins for the business, including faster delivery, a 50% reduction in cloud infrastructure costs, and several percentage points of increased revenue for our business unit. There's plenty more work to do, but these initial wins indicate that our investment in monitoring was well worth it.





ACKNOWLEDGEMENTS

I would like to thank TravelSupermarket's Product Engineering team for its hard work, and our Technology Director, Catherine Trotter, for sponsoring our efforts and giving us the space to evolve our solution.