Next step app development: Microservices

Your expert guide to building next-gen software architectures

In this e-guide

In this e-guide:

Adopting a microservices approach to application development is increasingly considered an essential part of any bid to modernise the legacy IT setup an organisation relies on.

The process sees organisations move away from developing large, monolithic applications in favour of developing ones that are built using a series of building blocks containing the code and operating environment needed to deliver a specific feature or functionality.

Each of these building blocks independently contribute towards the working of the overall application, allowing developers to update the code contained within each one without necessarily affecting the wider workings of the application itself.

In this e-guide, we take a closer look at what microservices are, the technologies that complement and enhance their use, and get a first-hand insight into the impact their use can have on an organisation's IT strategy and setup.

Caroline Donnelly, datacenter editor

In this e-guide

What are containers and microservices?

Bob Tarzey, guest contributor

Containers encapsulate discrete components of application logic provisioned only with the minimal resources needed to do their job.

Unlike virtual machines (VM), containers have no need for embedded operating systems (OS); calls are made for OS resources via an application programming interface (API).

Containerisation is, in effect, OS-level virtualisation (as opposed to VMs, which run on hypervisors, each with a full embedded OS).

Containers are easily packaged, lightweight and designed to run anywhere. Multiple containers can be deployed in a single VM.

A microservice is an application with a single function, such as routing network traffic, making an online payment or analysing a medical result.

The concept is not new; it has evolved from web services, and stringing microservices together into functional applications is an evolution of the service-oriented architecture (SOA), which was all the rage a few years ago.

In this e-guide

Not the same thing

Containers and microservices are not the same thing. A microservice may run in a container, but it could also run as a fully provisioned VM. A container need not be used for a microservice. However, containers are a good way to develop and deploy microservices, and the tools and platforms for running containers are a good way to manage microservice-based applications. In many cases, the terms can be interchanged.

Containers have been integral to Unix and Linux for years. A recent change has been the ease with which they can be used by all developers, and an entire supporting ecosystem has grown up around them. Containerisation is not something happening on the fringes of IT, it is core to the way many web-scale services operate and is increasingly being adopted by more conservative organisations. The suppliers mentioned in this article cite customers ranging from the NHS to large banks.

There are many suppliers involved, but no one disputes that Docker has led the charge and sits at the heart of the market. Docker says millions of developers and tens of thousands of organisations are now using its technology. However, another statistic indicates the novelty of containerisation for many, with only 40% of Docker's customers running containers in production.

In this e-guide

Docker's dominance does not mean it holds a monopoly; far from it. Across the whole container ecosystem, there is plenty of choice. There are many startups and the great and good of the IT industry are all on board, as a glance at the sponsors of the February 2016 Container World event shows. The top sponsor is IBM, which is one of Docker's three main global go-to market partners, along with Microsoft and Amazon Web Services (AWS).

Open source componentisation

Multiple containers are deployed in clusters and managed using a range of tools. Many of these containers will be pre-built components that can be layered together to build up application images. A prime benefit is that it is easy to "overwrite" an individual container while the application is still running – less scheduled downtime means better business continuity.

This has led to the rise of the DevOps concept, which allows faster deployment of new software capabilities directly into an operating environment.

Much of the core containerisation technology is open source, and suppliers that have previously eschewed it, such as VMware, are being drawn in. At its heart is the Open Container Initiative (OCI) launched in 2015. This operates under the auspices of the Linux Foundation to create open industry standards around container formats and their runtime environment. Docker

has donated its own format and runtime to the OCI to serve as the cornerstone.

Many containerised components are downloadable from open collaboration projects such as GitHub and Docker Hub. As with all open source technologies, the suppliers that operate in the market must earn their money by providing stable versions with associated support services.

The container stack

There are four technology layers that need consideration:

1. Container operating systems

Even though containers do not have an embedded OS, one is still needed. Any standard OS will do, including Linux or Windows. However, the actual OS resources required are usually limited, so the OS can be too.

This has led to the development of specialist container operating systems such as Rancher OS, CoreOS, VMware Photon, Ubuntu Snappy, the Red Hat-backed Project Atomic and Microsoft Nano Server. The benefit here is that the VMs provisioned to run containers are lightweight (some run in about 25MB) and when it comes to security, the attack surface is minimised. Cloud platform providers are embedding their own support.

In this e-guide

2. The container engine

This is where Docker dominates, but there are competitors, such as CoreOS Rocket (Rkt). AWS says Docker is by far the most popular engine with its customers, and therefore the focus of its support plans. Engines come with supporting tools, for example the Docker Toolbox, which simplifies the setup of Docker for developers, and the Docker Trusted Registry for image management. There are also third-party tools, such as Cloud66.

3. Container orchestration

Containers need to be intelligently clustered to form functioning applications, and this requires orchestration. Orchestration is where much of the differentiation lies in the containerisation ecosystem and it is where the competition is hotting up most.

The engines provide basic support for defining simple multi-container applications, for example Docker Compose. However, full orchestration involves scheduling of how and when containers should run, cluster management and the provision of extra resources, often across multiple hosts. Tools include Docker Swarm, the Google-backed Kubernetes and Apache Mesos. You could use general-purpose configuration tools, such as Chef and Puppet (both open source) or commercial offerings such as HashiCorp Atlas or Electric Cloud ElectricFlow. None of these is container-specific, however.

4. Application support services

Many additional tools are emerging to support containerised applications – some examples follow.

There is a danger of containerisation ending up like herding cats, which is a problem for application portability. An organisation may want to move an app from one cloud platform to another. Software suppliers will need to consistently recreate their applications for user deployments. How do you ensure all the dependencies and necessary containers are copied and recreated? Rancher Labs' core product (it also has an OS) enables applications to be built up from containers so that the full operating environment can be recreated, including the containers themselves, load balancers, networking, and so on.

Networking is an issue, especially across platforms. In 2015, Docker released Docker Networking to enable virtual connections between containers. UK-based Weaveworks also focuses on networking with WeaveNet, a micro-software-defined network (SDN). Metaswitch's Project Calico is all about making container networking more secure through the dynamic construction of firewalls, taking policy from orchestrators.

Docker, too, is developing new tools to support the lifecycle of containerised applications. Last year, it acquired a company called Tatum, an on-demand service for building, deploying and managing applications. The Docker Universal Control Plane (UCP) provides similar on-premise capability. Both

products are yet to go into production. Weaveworks also has a tool called WeaveScope for production monitoring of containerised applications.

Containerisation platforms

All the big industry players are joining the containerisation bandwagon, with a range of initiatives under way.

Google is an old hand with containers – it has been developing and deploying billions internally for years. The company has been a major contributor to various container-related open source projects, included the Kubernetes orchestrator, which it donated.

Google has now opened up this expertise to its customers and added the Google Container Engine to the Google Cloud Platform.

Microsoft has added container support with Windows Server Containers enabling the sharing of the OS kernel between a host and the containers it runs. Hyper-V Containers expands on this by running each container in an optimised virtual machine. For the cloud, there is the Azure Container Service (ACS), developed in conjunction with Docker, which can manage clusters of containers with “master machines” for orchestration. ACS also supports other orchestration tools, such as Kubernetes.

AWS customers were quick off the mark to deploy containers on its EC2 platform, so AWS has followed up by providing a cluster management and

scheduling engine for Docker called the EC2 Container Service (ECS). This is supported by the EC2 Container Registry (ECR) to support the storing, management and deployment of container images. ECS is widely available, whereas ECR is currently available only in the eastern US.

VMware has not taken the move to containers lying down. Later this year, it will release vSphere Integrated Containers, using VMware's Photon OS to turn VMs into Docker-like containers based on OCI. This will allow users to take advantage of existing vSphere support tools. In a first for VMware, it has open-sourced both the PhotonOS and the associated Photon Controller.

Other examples include IBM Containers for Bluemix, Rackspace Carina (based on OpenStack Magnum, embedded support for containers and orchestration). Another open source initiative is Deis, a platform-as-a-service (PaaS) based on CoreOS.

Big decisions

For developers, the open source nature of the container marketplace makes it easy to access the technology and supporting tools and crack on with building agile applications through a DevOps-style process.

This offers many benefits to businesses, but they must consider the supporting platforms and technologies that are endorsed to ensure longer-

term stability and support. Making such decisions will not be easy as the containerisation market changes.

Next article

In this e-guide

Divide and conquer in software architecture

Randy Heffner, guest contributor

To innovate in customer engagement and to drive and adapt to digital disruption, enterprises must continuously change at increasing velocity.

Application development and delivery professionals must increase their velocity too, while also maintaining – better yet, improving – the quality and resiliency of what they deliver.

Large web-native and mobile-native players such as Netflix, Amazon, Google, PayPal, Uber, eBay and Yahoo, as well as startups and, increasingly, a broader community of enterprises are changing their software architectures to meet the challenge of continuous business innovation.

There are two major sides to this challenge – how and what. For the how, agile development and continuous delivery improve software delivery processes.

For the what, application programming interfaces (APIs), containers and microservices improve software flexibility and deployment, providing a central foundation for a quiet revolution in software delivery and stability.

In this e-guide

Why microservices are important

Contrast microservices with monolithic applications. The distinction between them is, first of all, one of deployment, but it may also be a design distinction.

For example, a Java-based web application can be written as a collection of well-designed, modular Java classes. However, these classes are not designed for independent deployment, so all of them get packaged into one large file for deployment. Microservices refactor an application into a series of smaller, separately deployable units.

There are four major benefits to microservices. First, when individual parts of a system are separately deployed, they can also be separately scaled. A system might need 20 running instances for one service and only two instances of another. Wal-Mart credits Node.js-based microservices for its ability to handle Black Friday volume spikes when other retailers had issues.

Second, separately deployed services can be implemented using entirely different platforms or design models. One service might need a huge in-memory database and a specialised programming language, while another might have a very small footprint and be written in JavaScript on Node.js. The order history service of Amazon's retail store takes this principle further, using different implementations for recent orders and for older orders.

Third, when services are being run separately, it is more difficult for the whole system to go down at the same time, and a failure in processing one customer's request is less likely to affect other customers' requests. This is particularly true since each service's operation can be optimised for its specific scale, performance, security and transaction management requirements. Netflix, for example, ensures resilience by periodically knocking out its own production services to be sure that the whole keeps running.

And fourth, microservices also offer more options for incremental development and deployment. Agile methods – and particularly continuous delivery – can move more quickly when parts of a system can be built and deployed independently. But even with waterfall development, a bug fix is easier to deploy with microservices. Prior to production deployment, smaller units of development allow for smaller tests, more frequent testing, more options for testing and more frequent feedback.

Components, APIs – or both

Industry discussions on microservices are quite vague concerning what they actually are. Breaking the word into its two parts helps to clarify the debate.

The "services" part of "microservices" has three potential meanings. First, "service" means a component-based, container-based deployable unit. This definition of microservices borrows heavily from early-2000s concepts of

components, including the notion of marketplaces for components. However, it pushes beyond component-based development into individual deployment of each component. Operating system-level container technologies such as Docker are emerging as a primary means to structure an application as a collection of relatively small, separately deployable units. In this context, “service” is used in a general and generic sense of “doing something useful”.

Second, “service” means an API or messaging destination. In this definition of microservices, “service” is used in the specific sense of a network-accessible function, as with APIs, service-oriented

architecture (SOA), application messaging and message-oriented middleware (MOM). Most commonly, the implication is that the microservice would be a JSON/REST-based API, but other messaging styles and payload formats are also valid (AMQP, MQTT, SOAP, XML, Google Protocol Buffers, etc).

Third, “service” means both a component and an API. The first two definitions given here may be used together or separately. For example, microservices providing mobile APIs running on Node.js were behind Wal-Mart’s 2013 “no downtime on Black Friday” success story. The component definition is more important for deployment flexibility. The API definition helps mainly with application layering.

In this e-guide

The “micro” part of the term is harder to clarify, but it does not necessarily mean small. A microservice should do one thing well – from the point of view of the service user – and this may be as small as a type-ahead API or as big as an API to initiate a complete e-commerce order fulfilment process. The real point is how size interacts with a system’s amenability to change. For this, one should begin with the two central principles of good software design: high cohesion and low coupling.

High cohesion is achieved by keeping together in one unit the things that change together. Low coupling is achieved by designing so as to maximise the degree to which unit A’s implementation can change without affecting the design or implementation of other units that refer to unit A.

A unit’s size fits with the microservices concept if it is highly cohesive, with loose coupling to other units. For service users, the order fulfilment API achieves low coupling by insulating them from ongoing changes in order processing. For developers building the internals behind the order fulfilment API, low coupling will make it appropriate to divide the API’s internal implementation into multiple separately deployable microservices.

How to start down the path to microservices

Microservices are emerging as an important part of the solution architecture for rapid and scalable response to business and technology change. But moving to microservices is not simple. It’s a journey.

In this e-guide

The investments in and benefits from microservices can vary widely and, at the high end, can be quite high and strategic. Wal-Mart's backing of the Node.js-based Hapi project represents an investment of more than $2m – and this is just one part of Wal-Mart's microservice investment.

The microservice architectures for the Netflix and Amazon retail services are not merely online extensions, but rather the foundations of both businesses' core business models. If enterprise use of microservices is limited in some way – for example, in a context where the rate of change is relatively low or for a contained mobile app context – an initial foray into microservices may provide limited benefit while allowing the organisation to take on the microservice risks and challenges one by one.

This article is an extract of the Forrester July 2015 report, *Microservices have an important role in the future of solution architecture*, by vice-president and principal analyst Randy Heffner.

Next article

In this e-guide

How to start building enterprise momentum for microservices

Cliff Saran, managing editor

The Cloud Native Computing Foundation (CNCF) is just a year old, but at the start of its second conference in Berlin, it unveiled a number of initiatives that aim to improve support for some of the major container technologies.

Docker's core container runtime, together with Kubernetes runtime and gRPC, a high-performance remote procedure call technology, have been accepted by the Technical Oversight Committee as incubating projects within CNCF.

CNCF's vision is to promote interoperability and portability across different container execution environments. In other words, it should be possible for an application running in a Docker container to talk to a containerised Kubernetes application, and vice versa.

A cloud-native application runs in such containers and can make use of additional code, or microservices, running in different containers.

In this e-guide

While Docker and Kubernetes are well established, many of the technologies presented during the opening keynote are very much unknown, particularly in mainstream enterprises.

What is even more remarkable is that some of the key open source technologies began just a few years ago as tiny operations, and are now supported by leading open source companies with hundreds of code contributors.

One of these is Prometheus. Brian Brazil, founder of Robust Perception, worked on the original project. “It started in 2012 as one company with two people,” he said. “Prometheus now has more than 300 contributors, and 500 companies are using it in production.”

Prometheus is a metrics monitoring system which uses a time series database and has its own query language. A few enterprises are now engineering IT systems to use containers to improve the way software is developed, and Prometheus is among the tools of choice for monitoring such systems.

Justin Dean, senior vice-president for technical operations at Ticketmaster, said: “We use Docker for containers and the Kubernetes ecosystems, plus we make heavy use of Prometheus. People love how much easier it is to handle time series with Prometheus.”

Ticketmaster was one of the major organisations on a CNCF panel discussion to showcase the use of containers in big business. “It is really hard to be nimble and roll out new products and features compared to startups,” he said. “A few years ago, we started to realise we needed to get faster and deliver software faster, otherwise we would start losing to competitors.”

Using a DevOps approach

Industry consensus is to use a DevOps approach to software development, delivery and deployment, giving coding teams responsibility for all aspects of the software they build.

“Anyone who has been through the process understands the work required,” said Dean. “We wanted autonomous software delivery, where every team had everything they needed to deliver their product to market, and be responsible even for profit and loss. We were trying to create mini micro businesses all across the company that could move as fast as the competition instead of large, monolithic teams.”

The challenge in achieving this is both cultural and technical. There were too many tools required for the engineers at Ticketmaster to use. “We quickly got to a situation where we needed to revamp the tech, and ended up in the container space and the Kubernetes ecosystem,” he said.

In this e-guide

Global ticket distribution system Amadeus began its journey in 2014. Eric Mountain, senior expert of distributed systems at the company, said research and development teams also need to buy into the idea that DevOps is the right approach to develop software.

“R&D already has a system that works, so why move? You need an awful lot of communication to convince them it will given them something easier,” he said. “Containers makes that conversation easier.”

A shift in culture

Dean agreed the shift in culture was Ticketmaster’s biggest challenge. “We have 350 products and tonnes of teams with different software stacks, and they deliver software differently. There’s a lot of muscle memory built up. When you force a change, it has to be welcome.”

But, like Mountain at Amadeus, Dean found containers made the approach easier. The container effectively ring fences the code being developed. The coders cannot simply tweak some other piece of code or even alter the hardware, to make their software work. Essentially, the new code being developed plugs into a pipeline.

“When you deliver software in a container through a pipeline, it forces a massive change,” said Dean. “This single-handedly had more impact than anything else and dovetailed into DevOps.”

In this e-guide

The combination of containers, software-based infrastructure and microservices represents a major evolution in the way applications are deployed and managed. It is a fast-paced, changing world, where standards and preferred tools are still emerging. When Amadeus began its journey, Prometheus did not exist, so it needed to engineer its own monitoring, but this was done in a way that allowed it to plug in and replace functionality as the tools ecosystem evolved.

Arguably, for the traditional enterprise, this is perhaps a major roadblock. Unlike old-school enterprise applications that were vertically integrated and monolithic, the new world order is cloud-native, built around loosely coupled containerised applications where developers need to define the bits of code their applications require (known as dependencies).

“There is a pretty big hurdle to get an application into a container,” said Dean. “The barrier to entry is steep and there is a whole new language to learn.”

Next article

In this e-guide

Troubleshooting microservices performance problems

Kurt Marko, guest contributor

Using microservices to build applications can significantly improve developer productivity, project agility and code reuse. However, the resulting architecture is more complex, which makes isolating and debugging performance problems much harder.

As we discussed in an earlier article, analyzing and managing microservice performance is a "two-level problem comprising both granular telemetry of individual microservices and comprehensive, end-to-end, pan-application measurement. Microservice telemetry is used to identify internal bottlenecks, inefficiencies and bugs while application-wide monitoring seeks to see performance from the user's perspective and identify problems within the microservice ecosystem and its connections." In this follow up, we look at specific techniques and tools for troubleshooting microservices performance problems.

Data integration and analysis

The raw material for application performance analysis is data and, particularly when using microservices, the more the better. This means the

overall application and individual microservices should be designed upfront with telemetry in mind and application architects should have a strategy for data generation, collection and analysis with specific requirements each microservice developer is expected to follow. The strategy should incorporate both macroscopic, end-to-end performance measures for the application as a whole and internal telemetry from individual microservices. The latter is what's known as *in-situ monitoring* in which microservice developers add functions to each service that log relevant data and events that can be aggregated and analyzed for performance-robbing behaviors.

A microservices performance and troubleshooting plan should incorporate three broad categories of data:

1. **Metrics** that record the services and functions or operations within microservices that failed or exceeded baseline limits and by what amount.
2. **Logs** that detail the sequence of application activity that can be analyzed after an incident to trace the events leading up to a problem. For example, logs can show the specific microservices active during a problem and the API calls used and parameters passed.
3. **External events** affecting an application preceding or during a problem. Integrating with external systems such as code repositories, continuous integration/continuous deployment software or container orchestrators allows developers and IT to determine exogenous factors that could have caused a problem. For example, a code check-in generating a new version of a particular microservice, one or

more microservices being moved to a different physical machine, resource starvation from other workloads on a microservice cluster, or individual server.

Given the number of microservices a cloud-native application is likely to use and the resulting wide variety of data sources and types, the data is most useful when aggregated into a single repository. A unified data pool increases the effectiveness of log analysis software such as Loggly, Splunk, Sumo Logic or any of the growing ecosystem of services using the open source ELK stack of Elasticsearch (search), Logstash (log management) and Kibana (data visualization). For example, Elasticsearch can find terms that are similar to, but not exactly like, our search terms using the "fuzzy" operator or words or strings within a certain distance of each other (proximity searches). Creative use of powerful search operators can help troubleshooters identify and correlate events that aren't necessarily in perfect temporal order in log files.

Other tools and methods

There are several other techniques and automation tools that can be valuable in isolating and solving microservices performance problems, including:

- **Application performance management (APM) software** designed to analyze the end-to-end performance of distributed systems and

provides response time and throughput metrics from the customer's external perspective. Some APM software can perform automatic microservice topology discovery to provide a graphical view of microservice interactions. Such topology visualization and discovery is very helpful in understanding the flow of activity and events in complex microservice-based applications. Other advanced APM capabilities include distributed transaction tracing with the ability to drill into individual microservices to see internal state and events.

- **Network analysis and management software** to provide packet tracing at multiple levels of the network stack, including network, IP and application protocol. For rudimentary analysis, a command line tool like tcpdump is built into most OSes. However, Wireshark is a much more powerful open source program that provides a GUI and supports customized plug-ins for analyzing new protocols.
- **Synthetic transaction monitoring** is a technique adapted from web transaction monitoring in which developers build test scripts to simulate typical user interactions and stress different parts of the microservice functional chain. Scripts can be regularly scheduled and can trigger alerts when performance baselines are exceeded, detecting problems before end users experience them. Synthetic transactions can trace the flow through a microservice application and show microservice interactions by embedding a correlation tag or ID into each test.
- **Build metric and logging APIs** into each microservice to facilitate debugging by having a consistent interface to every service.
- **Build an incident response checklist** by taking a cue from airline pilots who don't leave important tasks to chance by using ad hoc procedures. Instead, specify the sequence of steps developers and IT

staff should take with each performance exception and alert. A good sample troubleshooting checklist comes from Netflix, with one of the most complex sets of microservice-based applications around.

Troubleshooting microservice-based software requires some new approaches and tools from those used for conventional monolithic applications. However, by aggressively logging, collecting and analyzing performance and event data, and taking a systematic approach to problem solving, adding microservices to your enterprise architecture need not lead to a never-ending maze of troubleshooting confusion.

Next article

In this e-guide

Case study: Trinity Mirror Group taps into cloud, agile and microservices to boost web user experience

Caroline Donnelly, datacentre editor

The rise of search engine optimisation (SEO) and social networking sites have drastically changed the way many internet users access and consume news content, while making the process of engendering brand loyalty that bit harder for online publishing houses.

In response to these pressures, regional and national newspaper publisher Trinity Mirror Group overhauled its technology strategy and the management of its IT delivery teams to ensure the 100 million unique visitors to its news sites each month keep coming back.

For Neil McIntyre, director of engineering for digital at Trinity Mirror Group, and his 70-strong team of developers, product testers, project managers and system administration staff, this has meant adopting agile working methodologies, cloud-based computing services and new user experience monitoring tools.

In this e-guide

"We are responsible for delivering the desktop and mobile websites and the mobile app for our core newspaper business. We ensure that the technology strategy and delivery align with the wider business and product roadmap," he said.

"The goal for us is to be as agile as possible, so we can deliver new code as frequently as we can and measure the effect of that, because we have to make sure everything we do has a positive business and technology impact as we develop our platforms," he continued.

A move to AWS cloud

The preparatory work for this started around 2011- 2012, when Trinity Mirror Group canned a nine-year partnership with its previous hosting provider, and set about moving its sites over to the Amazon Web Services (AWS) cloud. Around the same time, it also embarked on a push that would see it swap its bespoke content management system (CMS) for an off-the-shelf, commercial offering.

The latter move was embarked on as part of a wider redesign of its news sites – which include *The Daily Mirror*, *The Daily Record* and *The Manchester Evening News* – that fill the company's product portfolio, said McIntyre.

In this e-guide

"We host everything in Amazon, and the migration over to them was also part of the move over to the new CMS," he explained.

"We were using a dedicated hosting provider before, which had a single point of failure as it just had one datacentre, and we suffered a few outages that ended up being quite expensive. One of the remits I was given, on the back of this, was to look at our current hosting strategy."

During this process, the company considered going down the private cloud route and sourcing the services of another dedicated hosting provider, but decided to go with AWS instead.

"There was nothing that could compete with AWS from a cost point of view, and their self-service tools for creating instances were way ahead of what other providers in the market were offering," he said.

In light of the organisation's commitment to delivering a positive user experience and its past history of outages, AWS's promises about resiliency and failover datacentres were also a big lure, added McIntyre.

Performance monitoring challenges

As the roll-out of the CMS and its site redesigns progressed through 2012 and 2013, McIntyre said his team started to realise the tools they had in place to monitor the performance of its web properties were not fit for purpose any more.

Particularly when it came to ascertaining, in real-time, what the user experience was like for visitors to the news sites as his team began rolling out website code, he added.

“We realised the tools we had for measuring the performance of page loads and the advertising content in the pages,vfor example, weren’t really sufficient. The tools couldn’t give us an accurate baseline of where we were and how we could improve our page load times going forward,” he said.

For example, if users reported slow page loading times at a specific time of night on a particular mobile device, it would be difficult for the firm to ascertain the cause.

“What we couldn’t say with any degree of certainty was if it was a piece of code we released that was causing a problem for a specific mobile operating system, which is the level of detail we required,” added McIntyre.

This kicked-off another period of product evaluation in 2014, as the company set about finding a suitable tool to fit the bill, in anticipation of another wave of website redesigns.

McIntyre said the priority was to find a customisable technology that would allow it monitor the performance of any part of its front-end operations, including page loading times and the activities of the third-party tool – called Omniture – it uses to measure website traffic.

The organisation had some ambitious page load speed targets to hit in the wake of its redesigns, McIntyre said, and ended up opting for SOASTA's performance analytics tool, mPulse.

"The nice thing about the mPulse product was the depth and the amount of data we are able to obtain from it," said McIntyre.

Having this in place, he explained, paved the way for the organisation to adopt a continuous delivery approach to rolling out new code across its sites.

"We release code around three times a week, and sometimes that isn't frequent enough, but having this product in place allows us to understand what changes we're making and where we need to make improvements," he continued.

Reorganising the team for continuous delivery

In support of its continuous delivery ambitions, McIntyre's team recently underwent a restructure that has seen it adopt a more DevOps-friendly style of working too.

"We've now set up cross-functional, product delivery teams, featuring business analysts, developers and testers that are all working towards achieving specific business goal, as we move to adopt a more services-based model," he said.

In this e-guide

However, there are still some areas still to address before his team is in a position to release code changes throughout the day, but – with mPulse and AWS' technology powering the charge – a lot of the groundwork has been laid.

Next on McIntyre's agenda is working out how to break up its CMS – which is used across 29 of its core websites and is built using a single code base – into a microservices architecture.

This is the process by which a larger web application is built using a compositive of smaller, modular services that are easier to update individually and on an iterative basis, rather than trying to upgrade the entire thing at once.

"We're nowhere near being able to deliver the continuous delivery model, as it does rely on us taking a more microservice-based approach to what we do. One constraint we have with doing that is the CMS, which is shared across many of our products," he said.

"What we are looking to do to get round that is break it up architecturally from a technology perspective into microservices. The more we can do that, the more we can start to release code that doesn't have any dependencies across other product streams.

In this e-guide

"There are other players in the market who have managed this, such as *The Guardian*, Spotify and Netflix. It is a model we are aspiring to, but we have some technology changes to make before we get there."

Next article

In this e-guide

Getting more CW+ exclusive content

As a CW+ member, you have access to TechTarget's entire portfolio of 120+ websites. CW+ access directs you to previously unavailable "platinum members-only resources" that are guaranteed to save you the time and effort of having to track such premium content down on your own, ultimately helping you to solve your toughest IT challenges more effectively – and faster – than ever before.

Take full advantage of your membership by visiting www.computerweekly.com/eproducts

Images; Fotalia