ComputerWeekly.com TechTarget

E-guide

CW+ Content

The role of microservices in agile enterprises

In this e-guide

Over the course of the past decade, enterprises have looked to introduce more flexibility and agility into their software development processes, so they can responsively adapt their products and services in the face of changing consumer demand.

This has seen many organisations take steps to reinvent their monolithic application stacks by embracing the concepts of cloud-native computing, containers, agile software development, DevOps, and microservices.

Where microservices is concerned, this involves creating applications that are composed of multiple, independently-deployable services that communicate with each other, and that allow for the application as a whole to be updated in a piecemeal and iterative way.

It is an approach to software development that brings with it a lot of advantages, such as making it easier for organisations to scale up or down their applications in response to changing demand, while also reducing the risk of downtime caused by total app failures.

In this e-guide, we take a closer look at the latest thinking about microservices, from a pros and cons perspective, while also shining a

light on the other aspects of agile computing that enterprises need to know about as their digital transformations gather steam.

Microservices: What enterprises need to know

Daniel Robinson,

Microservices is an approach to software development that has seen a rising tide of interest over the last decade or so, going hand-in-hand with other trends such as cloud-native, agile development and, most notably, the use of containers as a vehicle for deploying software components.

Adoption of microservices has been increasing over the past several years. A survey carried out by O'Reilly in 2020 of over 1,500 organisations found that only about a quarter were not using microservices at all. Of the 75% that were, only about 10% had been using them for more than five years, which means the majority have taken the plunge with microservices during the past few years.

Microservices is not a specific technology, but instead is a style of software architecture and an approach to designing applications and services. Instead of creating an application as a single monolithic entity, the microservices approach is to break down the required functionality into a collection of smaller, independently deployable services that communicate with each other.

This approach has several advantages, one of which is easier scalability, as the individual components can be scaled independently of each other. Only the parts of the application that might be experiencing high demand, for example, need to be scaled, typically by starting up new instances of that component to balance the workload, rather than scaling the entire application.

Microservices also lends itself to an agile development process, because the smaller size of the component parts makes continuous improvement easier and allows for faster updates to the code. It also makes it possible for different programming languages to be used for different components, as some languages may be better suited to certain kinds of task. Because the component parts are small, it makes it easier for developers to understand the code, which can have a knock-on effect on reliability.

Another advantage is the potential to reduce downtime through better fault isolation. If a single microservice instance fails, it is less likely to bring down the entire application or service as a result.

Potential disadvantages

While there are advantages to a microservices approach, there are also some downsides that organisations need to consider. For example, although the development of each microservice component might be simpler, the application or service in its entirety might become more complex to manage.

This is especially so with a deployment of any scale, which might involve dozens or hundreds of individual instances of different microservice components. The complexity comes not just from managing the communication between all these separate instances, but monitoring each of them to ensure they are operating within expected parameters, and not consuming more resources than they would normally require, which may indicate there is a problem.

Testing and debugging may also become more of a challenge with microservices, simply because tracing the source of a bug can be more difficult among a web of microservice instances, each with its own set of logs. Testing the entire application can also be tricky, because each microservice can only be tested properly once all the services it is dependent upon have also been tested.

In particular, monitoring is more important in a microservices deployment, but the distributed nature of the components makes it more complex than traditional applications, which are largely self-contained. The monitoring system has to operate at the level of each individual microservice instance, while at the same time keeping an eye on the web of dependencies between the different components.

The way that microservices operate also has implications for the organisation's infrastructure. Supporting automatic scaling to meet increased demand implies that resources for new microservice instances must be capable of being provisioned by application programming interface (API) calls, which implies a certain level of cloud-like, software-defined infrastructure.

Data can be another thorny issue when building an application or service based on a microservices architecture, or to be more precise, where to store data. This is because each microservice instance is likely to have its own data store, but some applications may call for the ability to access a shared repository. Different services will also have different data storage requirements, with some in the industry saying that a NoSQL database makes the most sense, while others advocate sticking to SQL, if that is what the organisation has already deployed.

There are other differing opinions on this issue, with some experts advising that a single database (but not perhaps a single schema) shared by multiple services is the best approach, because, for one thing, it allows organisations to reuse the procedures they have in place for database backup and restore. Others advise against this, because it creates a potential single point of failure that goes against the microservices ethos.

Plan carefully

What this all means is that the microservices architecture may not suit every organisation, nor every type of application. However, the reasons behind its growing adoption are that microservices make it easier to implement a more agile approach to the deployment of services, which many organisations are now seeking.

“Organisations going down the microservices route tend to be more cutting-edge than the rest,” says independent analyst Clive Longbottom. “As such, they will also tend to be more open to thinking of what a move to a new architectural topology needs.

Historically, the majority of changes have been evolutionary: successful microservices architectures are revolutionary, requiring a complete rethink of what is being done."

In other words, microservices might be more suitable to a "green field" deployment that is being built from scratch, rather than organisations trying to refactor or update an existing application.

As already noted, Docker-style software containers are a technology that has become associated in the minds of many with microservices, although they are just one way of implementing a distributed deployment such as microservices. Other ways might include lightweight virtual machines, or even deploying microservice instances as non-virtualised code running in a server environment, just like everyday applications. Serverless computing functions would be another way of implementing microservices.

Containers are perhaps better suited than virtual machines, because they are less resource-heavy, and it is much quicker to spawn a new container instance than spin up a new virtual machine. Containers are also now a relatively mature technology, with a broad ecosystem of tools to support orchestration (such as Kubernetes), communications (such as Istio) and monitoring.
Interestingly, the O'Reilly survey found that a higher-than-average proportion of respondents who reported success with microservices chose to instantiate them using containers, while a higher proportion of respondents who had described their microservices efforts as unsuccessful had not used containers.

This might suggest that containers are a less risky option when implementing microservices, but again it is more a matter of choosing the right technology for the organisation's specific application and requirements.

"If we just look at a microservice, it is just a functional stub," says Longbottom. "The container should provide the environment the microservice needs, with orchestration and so on managing the provisioning, patching, updating and movement of the microservices as required across the broader platforms."

In other words, building microservices involves a different kind of complexity from traditional, more monolithic application styles. For this reason, it may be regarded as a technology better suited for new-build modern or cloud-native applications, or for organisations overhauling their IT as part of a digital transformation process.

How containerisation helps VW develop car software

Cliff Saran, Managing Editor

A modern car is a mobile server room, comprising many computers that manage and monitor critical parameters to improve safety, fuel efficiency and reduce component failures. There are also computers and embedded controllers for climate control and mechanics such as motorised folding mirrors and the entertainment system.

Software developed for these computers not only needs to be tested individually, but integration testing is also needed to make sure that any changes or modifications do not negatively affect other systems running in the car. It is a complex software environment that needs to encompass road tests and digital twins, too.

In 2018, global management consulting firm McKinsey noted that the number of lines of software contained in modern cars had grown from about 10 million in 2010 by a factor of 15 to roughly 150 million lines in 2016.

"Snowballing complexity is causing significant software-related quality issues, as evidenced by millions of recent vehicle recalls," warn the authors of McKinsey's *Rethinking car software and electronics architecture* article.

Given that the trend to push more software in automotives is unlikely to recede, Volkswagen (VW) has worked with Red Hat consultants to build its future software integration platform. This platform, based on Red Hat's OpenShift container orchestration system, has been built to enable VW to develop and test new software for electronic control units (ECUs) that provide the car with its processing brain.

"Containers may take an important role in automotive software development," says Marcus Greul, chief product owner of Cariad SE, an automotive technology company which is part of the Volkswagen Group.

"In our case, we use containers for building up scalable test environments in a mixed (virtual and real components) infrastructure. Maybe in the future, we will get the chance to use these concepts in a vehicle. The chance for using containers for safety-relevant functions is rather small."

The electric development department, part of VW Group's passenger cars research and development (R&D) department, tests car components such as electric mirrors and ECUs. A vehicle can have as many as 60 ECUs, each of which needs to go through extensive testing.

"Snowballing complexity is causing significant software-related quality issues, as evidenced by millions of recent vehicle recalls"

McKinsey

The ECU is effectively a computer that runs applications. During the setup of a test bench, ECUs must be integrated with model and simulation components. Each time one is updated or added, all related tests must be repeated and integration becomes more complex.

Delays due to a lack of on-demand provisioning for integration test environments was another factor driving the need for VW to update its software testing.

Describing the company's ambition for the new platform, Greul says: "We want to completely standardise and automate the release cycle of software components into our vehicles – including development, testing and deployment – by creating a shared environment for using both virtual and physical components."

By using Open Container Initiative models to follow best practices for container formats and runtimes, combined with the standardised infrastructure provided by Red Hat OpenShift, Greul says it is now possible to dynamically link container models to the ECUs, which cuts test bench delivery time from days to hours.

The company runs digital simulations of ECUs in containers using Red Hat OpenShift, an enterprise Kubernetes container platform. “There are different kinds of ‘virtual ECUs’ available. You can run the whole software stack on an emulated ECU, you can run single software components or modules on a sufficient runtime environment or just a model that behaves like the ECU,” adds Greul.

The company plans to connect the results of its integration tests with the results it gets from road tests. “For getting best performance and speed during test cycles, we are testing both in parallel,” he adds.

He says Volkswagen is about to use digital twins running on OpenShift on each of its vehicle platforms, each model and each equipment line, which are currently not managed by OpenShift.

Discussing the lifetime ownership of cars, and how automotive manufacturers such as Volkswagen will need to continue providing support for owners of their vehicles for many years in the second-hand market, Greul says: “We are strictly heading for over-the-air updates, thus testing gets more relevant for providing quick updates for customers during vehicle lifetimes.”

In this e-guide

Volkswagen uses Dell servers for hosting its on-premise OpenShift container platform. Greul says the servers must be certified both to run OpenShift and provide the hardware specification in terms of processor cores, memory, storage and GPU to support the workloads Volkswagen needs to run.

To support specialist hardware such as vehicle ECUs on an OpenShift cluster, Greul says the hardware needs to be described as a Kubernetes resource, also known as a custom resource definition (CRD).

The new IT architecture includes several other Red Hat technologies. Red Hat Quay is used as a private container registry for OpenShift, which stores, builds and deploys container images. Red Hat Runtimes provide the tools the company needs to develop and maintain cloud-native applications. Messaging is managed by Red Hat AMQ and Red Hat Virtualisation provides a software-defined platform to run virtualised workloads on Red Hat Enterprise Linux.

Preparing for enterprise-class containerisation

Adrian Bridgwater,

Despite the option to move essentially ephemeral computing resources and data between public, private and hybrid clouds, there is still an all-encompassing push to deploy unmodified monolithic applications in virtual machines (VMs) running on public cloud infrastructure.

However, it is more efficient to break down an application into functional blocks, each of which runs in its own container. The Computer Weekly Developer's Network (CWDN) asked industry experts about the modern trends, dynamics and challenges facing organisations as they migrate to the micro-engineering software world of containerisation.

Unlike VMs, containers share the underlying operating system (OS) and kernel, which means a single OS environment can support multiple containers. Put simply, containers can be seen as virtualisation at the process (or application) level, rather than at the OS level.

Those essential computing resources include core processing power, memory, data storage and input/output (I/O) provisioning, plus all the modern incremental "new age" functions and services, such as big data analytics engine calls, artificial intelligence (AI) brainpower and various forms of automation.

Although the move to containers provides more modular composability, the trade-off is a more complex interconnected set of computing resources that need to be managed, maintained and orchestrated. Despite the popularisation of Kubernetes and the entire ecosystem of so-called "observability" technologies, knowing the health, function and wider state of every deployed container concurrently is not always straightforward.

Migrating to containers

“The question I am often asked is how best to migrate applications from a VM environment to containers,” says Lei Zhang, tech lead and engineering manager of Alibaba’s cloud-native application management system, Alibaba Cloud Intelligence. “Every customer is trying to build a Kubernetes environment, and the ways to do it can seem complex. However, there is a range of methods, tools and best practice available for them to use.”

Zhang recommends that the first thing organisations looking to containerise their VM stack should do is create a clear migration plan. This involves breaking the migration into steps, beginning with the most stable applications, for example their website, and leaving the more complex applications until the container stack is more mature.

According to Lewis Marshall, technology evangelist at Appvia, the mitigation of risk alone is a huge benefit that seemingly makes the decision to containerise legacy systems easier to make. “Using inherently immutable containers with your legacy systems is an opportunity to remove the bad habits, processes and operational practices that exist with systems that have to be upgraded in place, and are therefore non-immutable,” he says.

Three quick wins of rehosting

By breaking the containerisation process into manageable pieces sorted out by complexity, you can begin to prioritise quick wins and create a longer-term strategy, says Jiani Zhang, president of the alliance and industrial business unit at Persistent Systems. Here are three steps she suggests IT decision-makers should consider when looking at containerisation:

- **Rehosting:** Look to apply the simplest containerisation technique possible to get quick wins early. Rehosting, otherwise known as the lift-and-shift method, is the easiest way to containerise your legacy application and move it to the cloud. Rehosting can dramatically increase return on investment in a short time. Not all applications can be rehosted, but the earlier you start, the longer you can enjoy the benefits while you spend time on the more difficult tasks.
- **Refactoring:** Refactoring is certainly more time-consuming than rehosting, but by isolating individual pieces of legacy applications into containerised microservices, you can get the benefits of moving the most important aspects of the application without having to refactor the entire codebase. From a time and effort standpoint, it often makes sense to only move the most important components, rather than the entire application. One practical example of this is by refactoring a legacy application's storage mechanism, such as the logs or user files. This will allow you to run the application in the container without losing any data, but also without moving everything into the container.
- **Rebuild:** Sometimes you have to cut your losses and rebuild an application that has passed its shelf life. Although this is time-consuming, these are often the

most expensive and least productive applications running on your system, and the work can pay off in the long run.

In Marshall's experience, containers have the capacity to increase security while decreasing operating and maintenance costs. For instance, some legacy systems have a lot of manual operational activities, which makes any sort of update incredibly labour-intensive and fraught with risk.
Marshall recommends that IT administrators try to ensure that the cost of operating legacy systems trends downwards, towards zero. "If your system is a cost sink while adding limited business value, then updating or upgrading it should become your priority," he says. "If your system is dependent on a few individuals who regularly put in lots of overtime to 'keep the lights on', that should be a huge red flag.

"It is also worth remembering that as a system ages, it generally becomes more expensive to maintain and the security and instability risks rise."

Challenges of containerisation

The immutable nature of container-based services, which can be deleted and redeployed when a new update is available, highlights the flexibility and scale they present. But, as Bola Rotibi, research director CCS Insight, pointed out in a recent Computer Weekly article, while containers may come and go, there will be critical data that must remain accessible and with relevant controls applied.

She says: "For the growing number of developers embracing the container model, physical computer storage facilities can no longer be someone else's concern. Developers will need to become involved in provisioning storage assets with containers. Being adept with modern data storage as well as the physical storage layer is vital to data-driven organisations."

Douglas Fallstrom, vice-president of product and operations at Hammerspace, says applications need to be aware of the infrastructure and where data is located. This, he warns, adds to the overall complexity of containerisation and contributes to the need to reconfigure applications if something changes. Also, the idea of data storage is not strictly compatible with the philosophy of cloud-native workloads.

"Just as compute has gone serverless to simplify orchestration, we need data to go storageless so that applications can access their data without knowing anything about the infrastructure running underneath," he says.

"When we talk about storageless data, what we are really saying is that data management should be self-served from any site or any cloud and let automation optimise the serving and protection of data without putting a call into IT."

From a data management perspective, databases are generally not built to run in a cloud-native architecture. According to Jim Walker, vice-president of product marketing at Cockroach Labs, management of a legacy database on modern infrastructure such as Kubernetes is very difficult. He says many organisations choose to run their databases alongside the scale-out environment provided by Kubernetes.

"This often creates a bottleneck, or worse, a single point of failure for the application," he adds. "Running a NoSQL database on Kubernetes is better aligned, but you will still experience transactional consistency issues."

Without addressing this issue with the database, Walker believes that software developers building cloud-native applications only get a fraction of the value offered by containers and orchestration. "We've seen great momentum in Kubernetes adoption, but it was originally designed for stateless workloads," he says. "Adoption has been held back as a result. The real push to adoption will occur as we build out data-intensive workloads to Kubernetes."

Management considerations

Beyond the challenges of taking a cloud-native approach to legacy IT modernisation, containers also offer IT departments a way to rethink their software development pipeline. More and more companies are adopting containers, as well as Kubernetes, to manage their implementations, says Sergey Pronin, product owner at open source database company Percona.

"Containers work well in the software development pipeline and make delivery easier," he says. "After a while, containerised applications move into production, Kubernetes takes care of the management side and everyone is happy."

Thanks to Kubernetes, applications can be programmatically scale up and down to handle peaks in usage by dynamically handling processor, memory, network and storage requirements, he adds.

However, while the software engineering teams have done their bit by setting up auto-scalers in Kubernetes to make applications more available and resilient, Pronin warns that IT departments may find their cloud bills starting to snowball.

For example, an AWS Elastic Block Storage user will pay for 10TB of provisioned EBS volumes even if only 1TB is really used. This can lead to sky-high cloud costs. “Each container will have its starting resource requirements reserved, so overestimating how much you are likely to need can add a substantial amount to your bill over time,” says Pronin.

As IT departments migrate more workloads into containers and put them into production, they will eventually need to manage multiple clusters of containers. This makes it important for IT departments to track container usage and spend levels in order to get a better picture of where the money is going.

Getting more CW+ exclusive content

As a CW+ member, you have access to TechTarget’s entire portfolio of 140+ 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.