HOME

NEWS

SME SUPPLIERS CRY FOUL OVER CAPITA'S GOVERNMENT DEAL

DIGITAL CHEQUES MAY BE FATAL BLOW TO BANK BRANCHES

SERVICE PROVIDER PARTNERSHIPS KEY TO FOOD FIRM'S IT

EDITOR'S COMMENT

OPINION

BUYER'S GUIDE TO CLOUD-OPTIMISED IT

MICROSERVICES: HOW TO DEPLOY THEM

DOWNTIME

# How to deploy microservices to derive maximum benefit

*Microservices are offering businesses a more efficient way of implementing systems and services. Janakiram MSV reports on how they can be best utilised*

DENPHUMI/ISTOCK/THINKSTOCK

In March 2014, Martin Fowler and James Lewis from ThoughtWorks published an article on microservices. Since then, the concept has gained prominence among web-scale startups and enterprises.

A microservice architecture promotes developing and deploying applications composed of independent, autonomous, modular, self-contained units. This is fundamentally different from the way traditional, monolithic applications are designed, developed, deployed and managed.

Distributed computing has been constantly evolving in the past two decades. During the mid-1990s, the industry evaluated component technology based on Corba, DCOM and J2EE. A component was regarded as a re-usable unit of code with immutable interfaces that could be shared among disparate applications.

The component architecture represented a shift away from how applications were previously developed using dynamic-link libraries, among others.

However, the communication protocol used by each component technology was proprietary – Remote Method Invocation (RMI) for Java, Internet InterORB Protocol (IIOB) for Corba and Remote Procedure Call (RPC) for DCOM. This made interoperability and integration of applications built on different platforms using different languages complex.

Using the cloud to enable microservices

Discover the role of smart endpoints in microservices

With the acceptance of Extensible Markup Language (XML) and Hypertext Transfer Protocol (HTTP) as standard protocols for cross-platform communication, service-oriented architecture (SOA) attempted to define a set of standards for interoperability.

Suppliers such as IBM, Tibco, Microsoft and Oracle started to ship enterprise application integration products based on SOA principles.

While these gained traction among the enterprises, young Web 2.0 companies started to adopt representational state transfer (Rest) as their preferred protocol for distributed computing. With JavaScript gaining ground, JavaScript Object Notation (JSON) and Rest quickly became the de facto standards for the web.

MICROSERVICES ARE FINE-GRAINED UNITS OF EXECUTION, DESIGNED TO DO ONE THING VERY WELL

## Key attributes of microservices

Microservices are fine-grained units of execution. They are designed to do one thing very well. Each microservice has exactly one well-known entry point. While this may sound like an attribute of a component, the difference is in the way they are packaged.

Microservices are not just code modules or libraries – they contain everything, including the operating system, platform, framework, runtime and dependencies, packaged as one unit of execution. Each microservice is an independent, autonomous process with no dependency on other microservices. It doesn't even know or acknowledge the existence of other microservices.

Microservices communicate with each other through language and platform-agnostic application programming interfaces (APIs). These APIs are typically exposed as Rest endpoints or can be invoked via lightweight messaging protocols such as RabbitMQ. They are loosely coupled with each other, avoiding synchronous and blocking calls whenever possible.

## Factors that influence and accelerate the move to microservices

Contemporary applications rely on continuous integration and continuous deployment pipelines for rapid iteration. To take advantage of this phenomenon, the application is split to form smaller, independent units based on the functionality.

Each unit is assigned to a team that owns the unit and is responsible for improving it. By adopting microservices, teams can rapidly ship newer versions of microservices without disrupting the other parts of the application.

WAVEBREAKMEDIALTD/THINKSTOCK

HOME

NEWS

SME SUPPLIERS CRY FOUL OVER CAPITA'S GOVERNMENT DEAL

DIGITAL CHEQUES MAY BE FATAL BLOW TO BANK BRANCHES

SERVICE PROVIDER PARTNERSHIPS KEY TO FOOD FIRM'S IT

EDITOR'S COMMENT

OPINION

BUYER'S GUIDE TO CLOUD-OPTIMISED IT

MICROSERVICES: HOW TO DEPLOY THEM

DOWNTIME

The evolution of the internet of things and machine-to-machine communication demands new ways of structuring the application modules. Each module should be responsible for one task participating in the larger workflow.

Container technologies such as Docker, Rocket and LXD offer portability of code across multiple environments. Developers are able to move code written on their development machines seamlessly across virtual machines, private cloud and public cloud. Each running container provides everything from an operating system to the code responsible for executing a task.

Infrastructure as code is a powerful concept, enabling developers to programmatically deal with underlying infrastructure. They will be able to dynamically provision, configure and orchestrate a few hundred virtual servers. This capability, when combined with containers, offers powerful tools such as Kubernetes to dynamically deploy clusters that run microservices.

Developers are choosing best-of-breed languages, frameworks and tools to write parts of applications. One large application might be composed of microservices written in Node.js, Ruby on Rails, Python, R and Java. Each microservice is written in a language that is best suited for the task.

WITH MICROSERVICES, DEVELOPERS AND OPERATORS CAN DEVELOP AND DEPLOY SELF-HEALING APPLICATIONS

This is also the case with the persistence layer. Web-scale applications are increasingly relying on object storage, semi-structured storage, structured storage and in-memory cache for persistence. Microservices make it easy to adopt a polyglot strategy for code and databases.

### Benefits of microservices

With microservices, developers and operators can develop and deploy self-healing applications. Since each microservice is autonomous and independent, it is easy to monitor and replace a faulty service without affecting any other. Unlike monolithic applications, microservice-based applications can be selectively scaled out.

Instead of launching multiple instances of the application server, it is possible to scale-out a specific microservice on demand. When the load shifts to other parts of the application, an earlier microservice will be scaled-in while scaling-out a different service. This delivers better value from the underlying infrastructure, as the need to provision new virtual machines shifts to provisioning new microservice instances on existing virtual machines.

Developers and administrators will be able to opt for best-of-breed technologies that work best with a specific microservice. They will be able to mix and match a variety of operating systems, languages, frameworks, runtimes, databases and monitoring tools.

Finally, by moving to microservices, organisations can invest in re-usable building blocks that are composable.

› Reasons you need microservices architecture
› How microservices bring agility to SOA
› Microservices is more than a buzzword

Each microservice acts like a Lego block that can be plugged into an application stack. By investing in a set of core microservices, organisations can assemble them to build applications catering to a variety of use cases.

Docker is the easiest way to get started with microservices. The tools and ecosystem around Docker make it a compelling platform for both web-scale startups and enterprises.

Enterprises can sign up for hosted container services such as Google Container Engine or Amazon EC2 Container Service to get a feel of deploying and managing containerised applications. Based on that learning, enterprises can consider deploying container infrastructure on-premise. ■