TechTarget | Search Microservices

E-guide

PRO+ Content

2017 Guide to Microservices & Containers

Strategies, tools, and what's behind the accelerated adoption rates

In this e-guide

In this e-guide:

Microservices deploy faster in containers than on VMs. Containers can offer wonderful benefits for microservices, but that doesn't mean they should always be used in conjunction.

Ahead, expert Tom Nolle explains the pros and cons of combining microservices and containers in 2017.

To finish, independent analyst Chris Tozzi explains why and how developers can achieve persistent storage in containerized environments.

In this e-guide

Microservices and containers present a new deployment model in 2017

Tom Nolle, President, CIMI Corporation

Applications in the cloud break all the traditional rules; they can move around failures, scale when workloads change, and their components can be mixed and matched to speed development and improve deployment efficiency. None of these benefits are automatic, though. We learned in 2016 that the same old application architectures running in the cloud end up running the same old way. To maximize cloud benefits, you need to change application models and optimize the cloud to suit your changes. Microservices combined with container deployment promise just that for 2017.

An alternative to SOA

For almost two decades now, it's been standard development practice to divide applications into components. SOA started that trend by using remote procedure call technology to network-distribute application pieces while retaining control over component access. Many believe that SOA has become bloated, though, and while SOA applications can be mixed and matched in the cloud, they are often difficult to move and restart under failure conditions, and usually don't scale well with workload changes. The

In this e-guide

limitations are particularly frustrating given that web applications have nearly all the properties that the cloud needs for growth.

Microservices seem to offer a solution to the problem. A microservice looks much like a web API in that it can be accessed with simple HTTP or JSON interfaces. Best practices for microservice design discourage writing stateful logic that can't be moved or scaled because some data is stored in the application between uses. From the beginning, interest in microservices has exploded and user trials followed. However, not all of those trials were successful.

The problem with microservices

One problem with microservices in traditional cloud deployments is the latency associated with accessing them. Every microservice is an inquiry-response combination, and if the microservice is accessed frequently in the course of doing work, the delays that accumulate can seriously impact user response time and productivity. This is an even greater problem if the microservices are brokered through an API management tool since the tool introduces an extra hop between the microservice user and the microservice.

Another problem that can hurt microservice adoption is resource waste. Microservices are typically small, far smaller than traditional application components. When deployed on virtual machines (VMs), the operating system and middleware needed to run those microservices can make up over 90% of the machine image, and even then the machine images

In this e-guide

themselves are often much smaller than usual. Most companies size their VMs to support typical machine images, and microservices use up a large amount of those resources.

Combining microservices and containers

The union of microservices and containers can break down nearly all the barriers to optimized cloud use. Microservices embody the scalability, reusability and resiliency features encouraged by the cloud, and containers can solve most of the resource efficiency problems. The value of deploying microservices in the cloud was recognized in 2016, but the value of joining microservices and containers is going to become clear for the first time in 2017.

Containers differ from VMs in that the applications running in containers share an operating system and much of the middleware. Eliminating the duplication of these big software elements allows many more microservices to run on a single server: five to even 10 times as many in routine deployments. Some users report they can run 30 times as many microservices with containers versus their VMs.

Microservices also deploy faster in containers than on VMs. That can be significantly useful during horizontal scaling of services with load or when a microservice must be redeployed due to a network or server failure. In fact, they can even be deployed on demand without generating an unacceptable performance impact.

In this e-guide

The power of the swarm

Efficiency in resources and deployment isn't the only benefit containers bring to microservices, and probably not the ones that will drive their adoption in 2017. Container-clouds are built from individual hosts, combined into clusters and networked on a larger scale. The automatic behavior of Docker, for example, is to place cooperative software elements like microservices on the same host as the components that use them and then on clusters of hosts called swarms, which are made up of coordinated host-centric deployment platforms. Docker and other container tools tend to do what microservices need -- colocate them with the rest of the applications by default. This kind of optimization can be done in VMs, but they require more explicit management through policy when hosting locations are selected.

The ability of containers to naturally group application components in defined clusters whose network latency can be controlled easily means that users are encouraged to think about the hosting policies for microservices. It also encourages them to define the scope over which such services can deploy, scale and redeploy. Since API managers are also container-hosted elements, they can be deployed where they're close to the microservices or to the applications that use them. In either case, this will limit the additional network delay that can accumulate.

In this e-guide

Thinking beyond microservices

The final benefit of combining microservices and containers is the impact on overall cloud-hosting policies, planning and tools. We're certain to see a growing set of tools designed to identify the best way to cluster containers for microservices. These tools won't limit themselves to microservice deployment. Instead, they'll work to optimize component placement in general, ending current practices where application components tend to be thrown at available resources, regardless of whether the location of those resources raises failure risks or introduces unacceptable levels of delay.

Microservices and containers are both relatively new concepts, which means they can develop symbiotically and the needs of one can optimize the behavior of the other. This beneficial relationship will advance both microservice and container usage significantly, in 2017 and beyond.

Next article

In this e-guide

Deciding when it's time to combine containers and microservices

Tom Nolle, President, CIMI Corporation

Containers are probably the fastest-growing and most broadly exciting development in virtualization and cloud computing. Microservices are a close second, so anything that could combine these two developments is something to look at closely.

However, cloud planners and application developers should frame the benefits of containers and microservices independently, and then in combination. They should pay particular attention to the kind of dynamism their applications will develop and also make plans for service federation.

It's complicated

The relationship between containers and microservices is complicated because it's not fixed. Containers are not necessary for microservices deployment, nor are microservices needed to justify container use. Containers are an operations tool that can improve deployment time and efficiency, but there are other comparable tools, such as virtual machines (VMs). Microservices are a way of enhancing application agility and code reuse, but there are other approaches here as well. That's why it's

In this e-guide

critical to look at all your software goals when assessing the benefits containers will bring to your microservices environment.

Let's start with the most compelling point: operational technology. Most IT professionals know containers are a form of virtualization that avoids duplicating the operating system across machine images by providing a common platform that is shared amongst all components. The most common container tool is Docker, and it's already well-known as an easy way to deploy virtual and cloud components. Containers currently lag VMs as the dominant deployment technology in virtualized data centers and private clouds, but they're gaining and will likely overtake VMs quickly.

On the development side, microservices are an evolution of the longstanding trend toward application componentization and loose binding of components. Small and generally useful software functions are published as network-connected services for reuse across a range of applications. This can also reduce software image size by letting every application use one copy of a general function or feature, and it can enhance development by creating a new software model that approaches one of module assembly or no-code. Microservices are gaining ground on other models, including SOA, for new applications.

For users who have not considered virtualization or private cloud adoption because they fear the complexity of hypervisors and VMs, containers offer a wonderfully lightweight way of hosting microservices. You can deploy them quickly, replace them quickly, and move them and even replicate them if you have a performance problem or a server failure. There's no question

In this e-guide

microservices in containers are a better approach than hosting them traditionally.

So, container or VM?

If containers are a logical alternative to traditional hosting of microservices, then the only question on container adoption is whether the VM alternative is even better. When a real decision around combining containers and microservices is to be made, look at how containers and VMs compare in their support for microservices.

There's no question containers are efficient microservices hosts. A microservices model involving a large number of separately hosted components could waste a lot of memory if each microservice were to be hosted in a VM with an independent OS. The multiplicity of OSes would also make maintaining version control in machine images more burdensome and put more pressure on operations.

The primary factor to consider when deciding if microservices can be effectively hosted in containers is the extent to which your company demands strict isolation of key applications for security or compliance reasons. Because they share an OS, containers are not as robust as VMs in sustaining independence among tenant application components. Microservices that are shared between applications with isolation requirements and those without them offer a similar possible point of compliance risk.

In this e-guide

Where no isolation of applications is needed, both containers and microservices can be freely deployed, and using both will improve development and operations efficiency. But when application isolation is needed, both container technology and microservices will have to be deployed with compliance guidance. This should start with a network connectivity plan that separates applications that demand special security from the others and places them onto protected subnetworks.

This is easily accomplished with enterprise software-defined networking technology, and even traditional IP networks can be partitioned this way. Look for support that specifically provides connectivity and access control to the subnets that host applications with security and compliance requirements. Also, be sure there's no risk of data or access leakage if you share microservices between secure and traditional applications.

Suppose you successfully address the compliance risks of containers and microservices. Can containers then help with microservices deployment? The answer is yes in nearly all cases, and the combination is more valuable with every microservices deployment. The biggest benefit will come from the memory savings generated by eliminating the OS from each image, but you may also find server usage efficiency is much higher with containers in microservices applications.

Good microservices practices dictate that you don't want to make a microservice out of a function that is constantly invoked. Network-coupling these functions will generate propagation delay that will eventually affect runtimes and quality of experience. This means most microservices will be lightweight, minimal users of resources. You'll limit the number you can host

In this e-guide

per server and increase operations costs if you allocate each microservice to a VM.

The operational downside of containerized microservices could be DevOps. DevOps tools typically support VMs better than containers, and container systems like Docker have traditionally provided limited internal support for complicated deployment orchestration. The more complex your microservice deployment is, the more important it will be to ensure you have proper DevOps tools available. So, before you commit to containers, look carefully at your DevOps options.

Some Docker users have found great value in turning to DevOps based on the OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) specification. Unlike most DevOps tools that have their roots in traditional data center deployment, TOSCA is a cloud architecture, and blueprints are available for most container deployments.

In summary

Microservices should be easy, and containers can make them so. But container use in microservices deployment can exacerbate security risks across application boundaries that could violate corporate security and compliance guidelines. Take care with containers and microservices in order to keep everyone happy.

Next article

In this e-guide

In this e-guide

Persistent storage for containers: What developers need to know

Chris Tozzi, Analyst, Fixate IO

Containers are a key ingredient for building an Agile, DevOps-oriented infrastructure, but reliable and scalable storage is just as important. To date, persistent storage for containers has been a challenge. The result is that persistent storage limitations have constrained the adoption of containers in the enterprise.

New storage solutions that adapt distributed storage systems for containerized infrastructure are helping to solve that challenge. And at the same time, older container storage tools like Docker Data Volumes can still be leveraged as an effective persistent storage solution, if used the right way.

Wondering how to identify and deploy the best persistent storage solution for your containerized environment? This article provides an overview of the current state of container storage, a description of the solutions available and an explanation of the advantages and limitations of each one.

The Persistent Storage Conundrum

Delivering persistent storage for containers is difficult because, by design, most containers themselves are both ephemeral and isolated. "Ephemeral,"

In this e-guide

in this case, means the apps within the containers will spin up and shut down at unpredictable rates, rather than remaining constantly on. The isolation of containers makes it more difficult to transfer data between different containerized apps or between the host system and a container.

These characteristics are what help make containerized infrastructure Agile and modular. They also present challenges for persistent data storage, since it's hard to store and share data persistently inside environments that are themselves intermittent and isolated.

Container Storage Solutions

Existing solutions for the persistent storage conundrum fall into two main categories: data volumes and cluster file systems.

The first, data volumes, involves using containers themselves to store data. Docker Data Volumes, for example, allow developers to create a special directory inside a Docker container that is dedicated to data storage. Because the volume can be mapped to a directory on the container host's file system, which remains in place after an individual container spins down, the data storage can be persistent. Data volumes can also be shared between containers.

The second major approach is to use a cluster file system for data storage. Under this model, file systems are made available over the network and shared with containers. This is the approach behind CoreOS' new Torus file system, which uses etcd to expose storage to hosts as a network block

In this e-guide

device. It's also the model that Red Hat utilized for OpenShift, its container deployment platform. This model makes use of Gluster, a cluster file system it has used for several years within noncontainerized environments, for persistent container storage.

There are other ways to deliver persistent storage for containers. Apart from the ones outlined above, however, container strategies are often too simplistic to be useful in real-world settings.

Which Persistent Storage Solution is Right for You?

So which of these two options is the best fit for your needs?

The best way to answer that question is to identify the pros and cons of each approach:

Should You Containerize?

Of course, you might also come to the conclusion that the existing storage options for containers won't work for you at all. If that's the case, then you're not yet ready to migrate to containerized infrastructure.

That doesn't mean you'll be forever stuck in the world of legacy virtual servers. Persistent storage technology for containers remains in rapid

In this e-guide

development, and better solutions are on the way. There's nothing wrong with waiting for them to arrive if the existing options don't meet your needs.

Further reading

In this e-guide

About SearchMicroservices

Over 4 million programmers, architects, IT managers, and developers turn to our site for industry news, expert advice and peer-to-peer learning opportunities around managing microservices and service-oriented architecture (SOA), application modernization and digital transformation, Business Process Management (BPM), application integration and APIs, software containers and microservices design and development, DevOps, and more.

Whether you're looking to solve a specific application architecture problem or just trying to stay on top of recent industry developments, our site is your online portal for in-depth and relevant information.

For further reading, visit us at http://SearchMicroservices.com/

Images; Fotalia