Inside Track
Executive Brief

FREEFORM DYNAMICS

Persistent Container Storage: The Business Case

Considerations for organizations deploying containers in production

in association with

The container revolution depends on storage

Containers are the rising stars of IT, and usage is growing rapidly. By packaging application software with just enough "system" to enable them to run, and with standard interfaces to the outside world, containers avoid both the bloat and unneeded duplication of virtual machines and the operating system dependencies of application streaming. The result is a compact, packaged workload that can run anywhere that supports that container infrastructure, regardless of its physical or virtual location or the underlying architecture.

Containers can address many of today's deployment problems

But containers can have dependencies of their own. In particular, applications more sophisticated than the likes of a webserver or Java app typically require stateful containers. These "remember" their data between runs, and must catch up with changes in the wider data environment whenever they restart. That means they require persistent storage – storage that can support the container as it stops and starts, and 'follow' it if it moves between platforms.

In this paper, we take a dive into the world of persistent storage for containers. How do the needs of stateful containers differ from those of traditional systems, what are the key enterprise features required of persistent storage, and how can we specify, implement and connect the two for IT and business advantage?

Why containers?

Right from the very start, container technology was designed to allow you to move software between different environments. More importantly, the container architecture enables any software running in it to work on any platform which supports that same container solution, from bare metal systems to virtual machines, and to operate in almost any public or private cloud.

That's because at its core, a container contains everything needed to allow the software to run, all packaged with transportability and automation in mind. But unlike virtual machines, containers hold only what is required to run the software, nothing else. This makes containers very resource-efficient, especially in terms of the CPU, memory and storage required to run effectively.

This means software development can be far faster and more flexible than is possible with traditional architecture-specific approaches, especially when containers are used in sync with techniques such as agile development, DevOps, and continuous delivery and testing. There's also a good fit between containers and microservices – these are a new implementation of an old idea, where you assemble applications from smaller elements. Used effectively, containers can make your infrastructure more agile and

responsive and with the constant pressure to keep software development lifecycle costs under control, stateful containers can help address such matters while reducing the time to market.

The question then is, given that container systems have been available for at least five years, why have they not already taken over the production universe?

What's holding back container adoption?

Containers are already making their mark, but as discussed to date they have mainly been used for hosting stateless applications. Stateless has many definitions, but for the purposes of this paper we mean applications whereby all parameters are passed into the container service. Its output depends only on the input, and it makes no direct attempt to store information or create any history itself. If you restart the container, the service has exactly the same function and information set each time.

By contrast, because a stateful application stores information about what has changed during the use of the service, it must write information as it is executed. Similarly, whenever it restarts it must catch up with any changes that have taken place in the data environment within which it operates. In many ways, stateful containers are no different than, say, a database running in a VM and writing to a VMDK, but containers were originally built without the storage layer. What is changing is who is provisioning and administering the storage.

Stateful containers require access to persistent storage

Each of these modes of operation has different requirements, with stateful systems making more demands than stateless systems. But the use of both systems has been limited until recently by several shared inhibitors.

An obvious one is simply familiarity and maturity. Any new technology also has to prove itself to the specialists before the mainstream begins to see it as an option for business-critical systems. Containers have followed a typical adoption curve, with organizations first experimenting with test systems, then rolling the technology out in carefully selected solutions where the benefits of containerization were most obvious.

History shows that as familiarity with containers grows, so too will their adoption.

Another inhibitor has been the availability of tools to monitor and actively manage container usage, especially at scale. But the surrounding ecosystem is maturing rapidly, and such tools are now reaching a state where they are able to support deployments.

Operationalizing containers: the role of storage

A key concern for any organization planning the broad use of container solutions is storage. On the one hand, containers are designed with flexibility and change in mind, so they need powerful, enterprise-grade storage. On the other hand, the need to support rapid business change makes it more challenging to use traditional enterprise storage platforms in this role.

This is even more true when looking at the type of business services likely to be supported by stateful containers. Already these include running database platforms such as PostgreSQL, MariaDB, or MySQL database where used as the back end to web applications and operating them next to the Web Apps they support. Other use cases could include running a NoSQL database such as MongoDB in a container, or supporting a message bus like RabbitMQ connecting applications together with guaranteed message delivery.

For container usage to grow, the supporting storage must be able to support critical business services

Such systems should of course be highly available and secure, as we will discuss, but they also need to function as part of the broader storage environment. The alternative is that they end up in a silo, something which would negatively affect the deployment flexibility that we need for container support.

Storage characteristics

More specifically, let's look at the key capabilities a storage platform needs in order to be able to move stateful container systems rapidly, perhaps automatically, while maintaining data access and security.

Platform independence

First and foremost, the storage platform must be able to provide access to data wherever that data is held and however the container moves. This means the container storage platform must be able to run anywhere and support a wide range of storage devices and services – cloud resources, virtual machines, bare metal, etc. – wherever they be located – on site, off site, hosted or in the public cloud. A stateful containers platform should also be able to work using so called commodity storage hardware to drive down the total infrastructure cost as well as supporting the adoption of hybrid cloud.

Container awareness

Automation is an absolute requirement for containers, so any supporting storage platform must also interoperate with the automation tools that exist in the ecosystem.

That means you must be able to administer it via APIs, with no need for bespoke management tools or scripts.

Dynamism

Given the nature of containers, all storage and security characteristics must dynamically move with their containers as the workloads they support are relocated. The same can be said for updates to the containers – security and storage characteristics must be maintained by policy during updates.

Enterprise-readiness

By their very nature, most stateful container applications will not tolerate session failure. The supporting storage must therefore provide high availability and be resilient to failure. It should also support other capabilities that are often associated with enterprise storage systems, such as data protection services, and data replication without service interruption.

Effective and efficient

Storage has become less expensive per GB over the years, but enterprises have storage at scale, so storage efficiency remains a necessity. Any storage platform used for stateful containers must therefore ensure that the storage underlying the platform is used effectively, regardless of what and where it is. Thus, mainstream storage services such as pooling and data reduction should be available.

Storage supporting stateful containers must be efficient, secure and deliver consistent QoS

Quality of service

Any storage platform designed to support stateful containers at scale should ensure that the quality of service delivered to the containers is consistent. Equally important, the quality of service delivered by the container storage platform must be performant and not require vast resources to ensure it is maintained at the desired level.

Manageability and ecosystem integration

Containers supporting both stateful and stateless applications will rarely exist in isolation. It is therefore imperative that the storage platform on which they operate be able to integrate with the expanding container ecosystem and with tools and technologies such as Kubernetes, Docker, Rancher, and OpenShift. They should also integrate with existing management solutions covering the broader storage and IT infrastructures.

Consumption flexibility

As has been mentioned, a primary driver for the adoption of containers is the ability to deploy applications flexibly, wherever the organization requires, whenever it requires. This demands that the charging model for the supporting storage platform be equally

flexible. It is thus essential that licensing should span a range of models, embracing capacity and device/node models, in perpetual and consumption-based approaches.

Vendor support and services

Open source communities such as the Cloud Native Foundation are extremely important, but most enterprises looking for a storage platform to support mainstream container deployment will still require vendor support and associated services.

Security

Any storage platform designed for use with containers running stateful applications must have all the security features that are demanded by mainstream enterprise compliance requirements. The platform itself must be robust in security terms, and should support or offer directly security services such as encryption etc. without adding undue complexity.

In conclusion – and looking forward

It will be a lot harder to deploy stateful applications in containers ('stateful containers') at scale without appropriately designed supporting infrastructure, and in particular persistent storage. Only by designing for persistence can you minimize management complexity and risk.

Storage is extremely important for stateful applications that are deployed via containers

When you look at the requirement to enable software development to deliver rapid time to market, a major driver for container usage is obvious. Add in the business benefit of employing on-site and off-site solutions effectively and dynamically in hybrid cloud models, and the need for secure storage flexibility to support container usage is clear. Thus, when the use of containers is planned, storage should be considered as part of the platform deployment, especially when supporting stateful applications.

Container usage is already growing and is being deployed to support an expanding range of business services. It is time to plan for the storage you will need for a containerized future. Looking for solutions that can provide the container storage characteristics discussed in this paper would be a good starting point.

About Freeform Dynamics

Freeform Dynamics is an IT industry analyst firm. Through our research and insights, we help busy IT and business professionals get up to speed on the latest technology developments, and make better-informed investment decisions.

For more information, and access to our library of free research, please visit www.freeformdynamics.com.

About StorageOS

StorageOS is a cloud native, persistent storage solution for containers that makes it easy to build stateful containerized apps. It supports production workloads for transactional databases, messaging systems and other business-critical data stores. StorageOS enhances your applications adding: high availability, rapid recovery, data encryption and the lowest possible latency. Benefit from container storage, as agile as your application, with automated policy management on par with traditional enterprise storage solutions.

Built to run with any stateful application, on any infrastructure with any orchestrator and as a container, StorageOS easily integrates with your favorite platforms – Kubernetes, OpenShift or Docker. Learn more at www.storageos.com.