Technology Trends

Application Containers
Version 0.1
Date 2019-5-28

Shared Services Canada
Services partagés Canada
Canada

Table of Contents

Business Brief

Application containers are a form of OS-Virtualization in which distributed applications are run on a single virtual machine or physical host rather than one for each application. A single host OS can support multiple containers. Containers can be used on multiple systems including cloud instances and virtual machines, as well as across different OS such as Linux, Microsoft, and MAC OS. Application containers have huge benefits with their ability to be deployed in cloud environments. The main advantage to containers is their architecture. Because applications can now be placed on different virtual and physical machines this offers greater availability of those applications. These machines can be within a cloud or not. They offer high flexibility with regard to workload management and allow the developer to make vault-tolerant systems.

Application containers transition data centres from being machine-oriented to application-oriented. Containers encapsulate the application environment, abstracting details from the machine, operating system, the application developer, and deployment infrastructure. Because well-designed containers and container images are scoped to a single application, managing containers means managing applications rather than machines. This shift with management application programming interfaces (API) from machine-oriented to application-oriented dramatically improves application deployment.

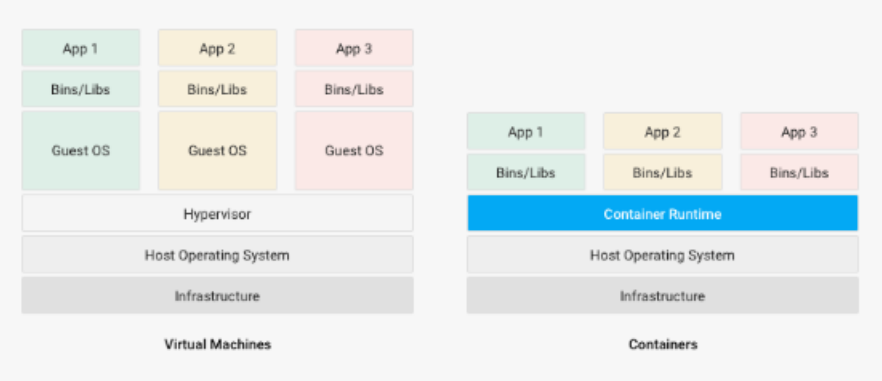

Figure 1: Virtual Machine versus Application Container Architecture

Technical Brief

It is useful to look at the concept of containers as images when trying to understand how containers work. Images can be stacked hierarchically. Applications can be attached to this hierarchy and each node of the tree allows the user to share images among applications. This allows for the configuration of binary states, meaning that you do not have to shift around the entire application's stack in single files. The relationship between images and containers is analogous to that of a class and object. In object

oriented programming an object is an instance of a class. For example, an apple would be an object of class fruit. The class defines the data stored about the object and what functions can be performed on it. With application containers, the image, like a class, provides the blueprint to the container. The container is like an instance or object of this class.

Another advantage of this is when a vulnerability is discovered within a particular node, the hierarchical tree allows the user to verify all other nodes that are impacted by this vulnerability. In order to create images, it is important to look at an example. Docker is a container engine. Docker uses a Docker file to create images. Containers are used to isolate processes within the OS.

Application containers can be thought of as a form of virtualization similar to Virtual Machines (VM). However instead of virtualizing the entire stack, containers virtualize at an operating system level [1]. Because of this they are extremely lightweight, enabling them to use only a fraction of the memory and processing as only a single OS Kernel is needed to host multiple containers.

Application containers facilitate a service oriented architecture (SOA). In an SOA the application is decomposed in into several services. With containers, individual services can be launched as different containers. With clustering this enables the ability to scale up a solution. The proper orchestration tools are required to properly enable scaling. Container orchestration tools manage the lifecycle of containers, a software development team will use them to provision and deploy containers, check redundancy and availability, and to move a container from one host to another.

Monolithic applications can also be deployed as containers. As a monolithic container is essentially a single–tiered application a single instance will be deployed as a single container. This differs from the microservice and SOA approach as a container will represent a single service of that application. With containers, in order to scale up a monolithic application multiple instances of the application and thus several containers need to be deployed. When using Docker, updates to the application can be deployed as docker images which is faster and network efficient. Also since containers can be spun up faster, creating new application instances a quicker process.

Two major vendors of application container technology are Docker and CoreOS. Docker was introduced first, and as a result is used more widely in the industry. Application containers are of most benefit to large companies. It has been four years since Docker's initial foray into the market, and a survey conducted by Datadog in 2016 revealed that adoption of Docker had increased 30%+ over the previous year. What is important to note about this statistic is that these companies are using the software in production rather than development, which suggests that companies are using containers as a long-term approach and not simply on a trial basis.

Docker can be characterized as the container management engine which creates the containers. There are also several management platforms that give users a number of different tools to manage their containers. Three of these tools are OpenShift, Kubernetes, and Rancher.

OpenShift uses Docker and Kubernetes as its underlying engine to manage the containers. The software integrates with Kubernetes to enable the automation of functions such as scaling, health management, and deployment. Kubernetes, which was created by Google, is a well-designed orchestration tool used in the management of containers. It offers the client a high level API that allows the user to logically group containers as well as load balancing and container pools. Rancher is a container management tool built on top of the Kubernetes container engine platform.

Industry Use

Several Large organizations have remarked the benefits of containers within their development lifecycle. ADP is a large vendor of cloud-based human capital management solutions. The company has 630,000 clients with around 35 million users of their systems. They began using application containers to speed up their development lifecycle. As of April 2017, ADP had 469 Docker engines, with 3771 containers running on virtual machines (VM). Paypal is another major user of application containers. The company hosts the largest online payment systems in the world. They have over 210 million users, in 200 international markets, and handle close to $100 billion in transactions per quarter. Paypal used Docker to perform datacenter modernization. In 2016, they launched their first container in QA and production. Since their applications were now decoupled form the operating system they were running the company could now modernize their infrastructure without needed to update the application. With more modern and faster infrastructure they remarked a 10-20% increase in efficiency with no downtime. Paypal now has 700 applications migrated to Docker and has in production 45000 VM hosts with over 150,000 containers running on them.

Canadian Government Use

There is a lack of documented Government of Canada (GC) initiatives and programs promoting the current and future use of application containerization and application container technology. As a GC strategic IT item, application containers is absent from both the GC's Digital Operations Strategic Plan: 2018-2022 and the GC Strategic Plan for Information Management and Information Technology 2017 to 2021. This may be due to the fact that the SSC is currently grappling with consolidation and digital transformations in parallel to the implementation of Cloud Services. The majority of

resources and efforts are occupied with implementation challenges, as well as security concerns related to the protection of the information of Canadians.

However, the inception of containers into the market has shown that large-scale organizations, who are involved in cloud-native application development as well as networking, can benefit greatly from the use of containers.[i] Although the infrastructure applications providing cloud services can be based solely on Virtual Machines (VMs), the maintenance costs associated with running different operating systems on individual VMs outweighs the benefit.[ii] Containers and Containerization is a replacement and/or complimentary architecture for VMs. As the GC moves toward cloud services and development of cloud-native applications, the use of containers and orchestrating them with Kubernetes can become an integral part the GC IT architecture.

Implications for Shared Services Canada (SSC)

Value Proposition

Application containers have a multitude of uses and SSC can benefit from such a flexible tool. Application containers especially shine if they are used for developing and testing new applications, since they can be easily deployed to create a safe testing environment (like a sandbox) for running code. Containers can be easily controlled from a container file, and can be scaled to fit any resource requirements. Since they are so easily customized, application containers are often used within the Services Oriented Architecture (SOA) software development framework to house microservices. An application developed under the SOA framework is usually made up of many small pieces (microservices) that communicate and work together.[iii] Should SSC adopt this or other similar software development models (like Microservices Architecture), application containers will be a necessary development tool.

Containers can facilitate easier and more efficient deployment of applications. There is the potential for increased efficiencies since containers create a smaller footprint from a memory perspective on a physical host. Since multiple containers can be created for a single application (like in SOA), in practical terms this means that version control is simplified. Instead of patching an application's code, a container housing a small portion of the application can be easily pulled out and updated without disrupting the rest of the application. If an orchestration tool is used to manage all of an applications containers, there is less opportunity for disruption since the orchestrator can monitor the "health" of all of its containers.[iv] If the orchestrator determines that a container is malfunctioning, it can shut it down automatically and launch other containers to pick up where it left off.

In addition containers are portable, meaning that they will run whatever program they contain in the exact same way every time it is launched, and wherever it is launched.[v] In other words applications will run consistently in a standardized environment, even if the container is launched from the cloud or from a physical server. This practically eliminates the need to do lengthy custom installations on every single machine in a network, an application can simply be run from a container. Portability will also help with running legacy applications, in the sense that if an older app needs a specific environment for it to run (for example, older versions of Java or Python), that environment can be recreated in a container. However, there are some limitations still present, containers from one platform (like Docker) can't be run on another platform (like Linux Containers) and there are also backwards compatibility issues, containers designed for the newest version of a platform may not work on older versions.[vi]

Challenges

There are four major challenges with the adoption of application containers:

Staff education and training -- The proper utilization of containers requires a whole new skill set. It also requires IT teams to be familiar with the container platform and container orchestration tools they will be using. In addition, IT teams may need to learn new design standards (like SOA) to reap the full benefit of application containers.

Choosing the proper container platform and orchestration program-- This is a crucial decision which will depend on the needs of the department. Depending on the Operating System in which the containers will operate, there will be different tools available. Although the GC is looking to adopt open source tools, there is no shortage of options, each with their own pros and cons.

Re-designing applications – Legacy applications developed under a monolithic model may not always be suitable for containers. Containers may run into trouble when developers attempt to update a single feature of the older application, but this holds true for monolithic applications in general. Monolithic applications can instead be broken into smaller pieces and container-ized, so they can continue to operate on newer systems.

Security -- Security is crucial since the containers can be based on a combination of cloud and local networks. The key factor for security is designing a solution that grants its users freedom and speed but is not vulnerable to data breaches.

Considerations

In general, containers still have their own limitations (just like any technology) and should not be adopted for the sake of innovation. Application containers come with their own unique requirements and challenges, like building skill sets and choosing the right software suites. If SSC were to move in a direction where application containers

are a necessity, then overcoming those challenges and related costs will be a worthwhile investment.

In terms of security, applications running within a container should not be assumed to be secure. Security should be a consideration throughout the entire lifecycle of an application and containers should be no exception. If SSC were to adopt application containers, different security processes would need to be developed to handle the new technology.

In the case that SSC develops applications under the SOA framework or others like it, there will need to be a shift in how these applications are monitored. Rather than monitoring the machines on which they are operating, the containers themselves would need to be monitored for performance. Key Performance Indicators (KPIs) would need to be collected from the orchestration tool that controls an application's containers.

As a tool, application containers present many potential benefits to SSC if they are to be properly managed and integrated into the workplace.

References

1. https://cloud.google.com/containers/
2. https://www.networkcomputing.com/data-centers/5-container-deployment-challenges/1532194322
3. https://rancher.com/
4. https://shadow-soft.com/open-source-container-management-tools/
5. https://www.contino.io/insights/whos-using-docker
6. https://searchitoperations.techtarget.com/definition/application-containerization-app-containerization
7. https://www.anmb.ro/buletinstiintific/buletine/2016_Issue2/FCS/412-414.pdf
8. https://www.gartner.com/smarterwithgartner/6-best-practices-for-creating-a-container-platform-strategy/

[i] CENGN. (2018). CENGN and CloudOps Collaborate to Train Industry on Docker and Kubernetes. [online] Available at: https://www.cengn.ca/docker-kubernetes-training-jan18/ [Accessed 13 Jul. 2018].

[ii] Heron, P. (2018). Experimenting with containerised infrastructure for GOV.UK - Inside GOV.UK. [online] Insidegovuk.blog.gov.uk. Available at: https://insidegovuk.blog.gov.uk/2017/09/15/experimenting-with-containerised-infrastructure-for-gov-uk/ [Accessed 6 Jul. 2018].

[iii] Abeywardhana, Sajith. (February 9th, 2019). *SOA vs Microservices with Docker and Kubernetes-1.* Medium. Retrieved 18-July-2019 https://medium.com/@sajithswa/soa-vs-microservices-with-docker-and-kubernetes-1-291686200a0f

[iv] Wagner, Bill, et al. (June 1st, 2019). *Health Monitoring - .NET Microservices: Architecture for Containerized .NET Applications*. Microsoft Developer Division. Redmon, Washington. Retrieved 19-July-2019. https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/implement-resilient-applications/monitor-app-health

[v] Tozzi, Chris. (2017). *The pros and cos of container platforms for portability*. Search Microservices. Retrieved 22-July-2019 https://searchmicroservices.techtarget.com/tip/The-pros-and-cons-of-container-platforms-for-portability

[vi] Tozzi, Chris. (2017). *The pros and cos of container platforms for portability*. Search Microservices. Retrieved 22-July-2019 https://searchmicroservices.techtarget.com/tip/The-pros-and-cons-of-container-platforms-for-portability