The Virtualization Practice

White Paper:

Managing Applications in Docker Containers

Bernd Harzog

Analyst – Virtualization and Cloud Performance Management

October 2014

Abstract

Docker has captured the attention of the development and deployment community by delivering a container that abstracts an application and the libraries that the application depends upon from the underlying Linux operating system.

This provides several advantages to teams building, testing, and then managing the deployment of Linux based applications in production. These include being able to keep a consistent set of application code and supporting libraries through the entire development lifecycle, being able to develop test and deploy on different distributions of Linux, being able to easily incrementally update the contents of a Docker container, being able to easily migrate containers from one execution environment (a private cloud) to another (perhaps a public cloud), and being able to run multiple Docker containers (and the application within them) on one instance of Linux, with each container isolated from the others. However, with all innovations come new management challenges. This paper discusses the benefits of Docker, these new challenges and how they are best addressed.

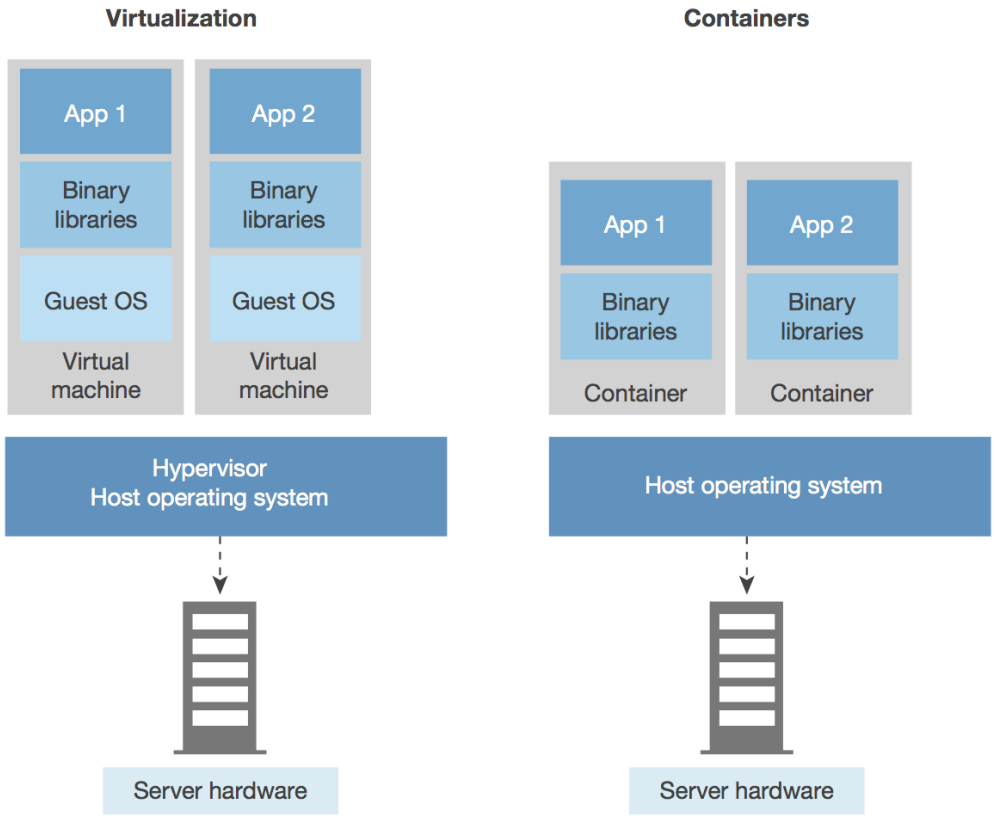
I. A Brief History of Virtualization

VMware pioneered data center virtualization with its vSphere product, which is a layer of software in between the hardware and the guest operating systems and which results in operating systems and their entire stacks running in virtual machines. Note that in a Type 1 Hypervisor system, each application has its own unique stack and instance of an entire operating system. VMware vSphere is widely deployed in enterprise data centers worldwide and is the de-facto standard for on-premise enterprise data center management.

Containerization, i.e. Docker, allows there to be just one host operating system, and provides a layer of software at the top of the operating system that isolates multiple applications and their required supporting stacks of software from each other, and from the operating system.

The architectural differences between a traditional Hypervisor and a Container are shown in the image below.

Figure 1 Unlike Virtual Machines, Containers Do Not Include A Full Guest Operating System



119001 Source: Forrester Research, Inc. Unauthorized reproduction or distribution prohibited.

II. Docker and its Advantages

According to the Docker website, “Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications. Consisting of Docker Engine, a portable, lightweight runtime and packaging tool, and Docker Hub, a cloud service for sharing applications and automating workflows, Docker enables apps to be quickly assembled from components and eliminates the friction between development, QA, and production environments.”

Ultimately, Docker is an open source project that leverages Linux Container technology to enable rapid application deployment, simpler testing, maintenance, and troubleshooting while improving security.

Docker works with and consists of the following components:

- The Docker Container – This is a sandbox for an application or a component of an application. Each container is based upon an image that holds the necessary configuration data for that container.
- The Docker Image – This is a static snapshot of the configuration of the container. It is a read-only layer that is never modified. The image is updated by writing a new image on top of the old one – while the old one is preserved. This allows easy roll-back to previous configurations.
- Base Image – This is an image that has no parent and that just contains the runtime environment for applications, components or services that are in other images.
- The Docker Hub – Registry of Docker images.
- The Dockerfile – a configuration build file with instructions for Docker image. This is how build procedures are automated, shared, and reused.

Advantages of Docker

Ultimately, Docker makes it easy to manage the deployment of complex distributed applications. Docker provides the following advantages for both developers and systems administrators:

- The process of bringing code into production is accelerated and streamlined. Since the containers include the run time components of the operating system needed in order for the application to function, those components remain consistent through the entire development, testing, pilot and production release process.
- Since the libraries and run times that the application needs are included in the Docker container, containers are portable across multiple Linux distributions. This allows enterprises to run fully supported enterprise versions of Linux in production without having to necessarily buy support for Linux instances used in development and test.
- Version control and change control. Since Docker containers are fully versioned, it is easy to roll back to a previous version of problems arise with a new release into production.
- Reuse of containers. Since containers can be nested, standards can be created for certain components and then reused across many applications. This leads to greater consistency across the environment and fewer problems.
- The Docker Hub provides a centralized and shared repositories of Docker containers and images. Docker Hub makes it easy to use a remote repository to share your container with others.
- Docker images are small and have minimal overhead which helps rapid delivery and reduces the time to deploy updates into production
- Application maintenance is simplified as Docker virtually eliminates problems with application dependencies.

III. Virtualization and Containers

While at first glance Docker and VMware seem similar and therefore perhaps competitive there are important details to the similarities and the differences that matter.

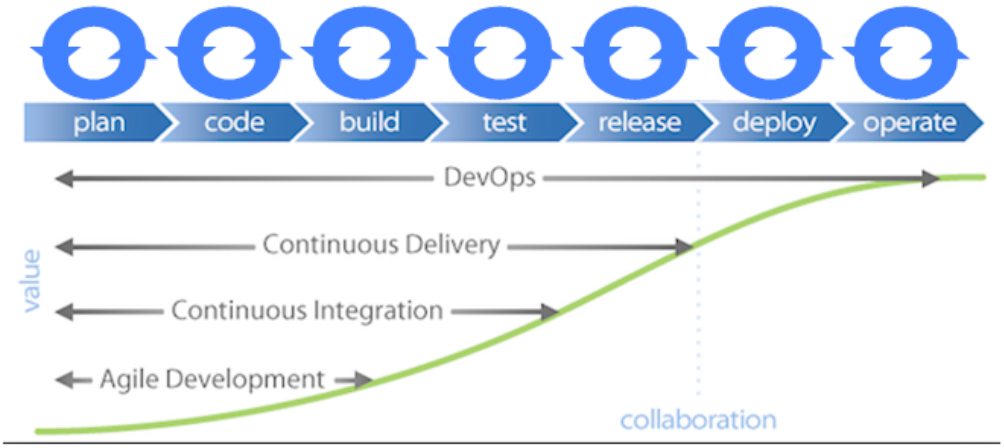
Docker and VMware

At the highest level, both Docker and VMware offer the same benefit – the ability to run multiple applications on the same physical server without the risk of conflicts in versions of software between the various applications and their required supporting software. VMware accomplishes this by putting the application, its required libraries, and the underlying operating system into a virtual machine. Docker accomplishes this by putting the application and its required libraries into a container that shares a base kernel and allows running multiple containers on one instance of Linux (instead of running multiple complete Linux instances). Docker has a significant advantage in that it requires only one instance of the operating system and can then share that operating system across multiple application instances. The proponents of Docker point out that with a Type 1 Hypervisor there are two redundant layers of software managing CPU, memory, network and disk resources – one in the hypervisor, and one in the guest operating system. With Docker this redundancy is eliminated, which the advocates of Docker claim will lead to increased efficiencies and greater densities.

The prospect of Docker emerging as a competitive alternative to VMware vSphere has lead VMware to embark on an effort to demonstrate that the most efficient and effective way to run Docker containers is inside of VMware vSphere virtual machines. This is documented in "VMware and Docker – Better Together" a blog from the CTO Office of VMware. Google has delivered Live Migration for the containers hosted in its Google Cloud Platform. It is rumored that Amazon is working on Live Migration as well. ClusterHQ has launched Flocker, which is an open source volume and container manager for Docker, which is headed in the direction of enabling Live Migration of Docker containers and their storage.

IV. Agile Development and DevOps

Enterprises now face unprecedented pressure from the business to respond with agility, address issues immediately, and to run complex heterogeneous applications in highly distributed and dynamic environments. The Agile Development and DevOps processes for rapidly developing new releases and supporting rapidly changing applications in production are shown in the diagram below.



The important point is that there is rapid iteration in each step of the process. Docker makes it easy to promote a new build across each stage from dev, to qa, to production.

Docker enhances Agile Development and DevOps by integrating with the key parts of the tool chain that is associated with the delivery of code into production. Docker is transforming the industry in a move away from shipping applications and managing dependencies with configuration management tools, but rather shipping entire applications with containers. Using Docker means having less reliance on traditional configuration management tools like Chef and Puppet.

Developers can now build, test and deliver apps as docker containers which that means that ops teams know even less than they did before about the apps they support, increasing the need for APM tools to find performance problems. The reality is that Ops will now rely less on tools like Chef, Puppet and will focus mostly on deployment tasks with docker containers as the deployment unit and the cloud as the deployment target.

V. Managing Docker in Production

The beautiful thing about Docker is that since all of the supporting libraries that an application needs to run are included in the Docker container, that single container can now run on a developers laptop, in a test lab, on-premise in a data center, or in a cloud. It is for this reason that Docker is getting support from a broad range of vendors including Google, Amazon, Red Hat, Microsoft, and VMware.

Docker and the DevOps Process

Prior to Docker, there was an unclear line between the “Dev” part of DevOps and the “Ops” part of DevOps in many organizations. The point of confusion (and in many cases contention) was not over who owned the code (Dev), but who owned the libraries and the configuration that the code depends upon.

Docker gives organizations an opportunity to clearly establish where that line is. Docker containers can be linked together or even nested within each other. This means that your team could decide to just keep all of the libraries that the code depends upon in the same container and have the development team be responsible for the whole thing. Whatever is decided on the front of who supports the libraries it is clear that Docker isolates the code into a container that is now owned by and supported by the development team. This means that Ops is now going to have less visibility into what is happening with the code, and Dev is going to have less visibility into what is happening with the environment that supports the code. The benefit of shipping application via containers creates a need for APM tools that can trace transactions across complex and distributed environments.

If organizations choose to make Developers solely responsible for the containers that contain the application code, and Agile Development is practiced against that code then another problem will arise. Since Ops is not going to have visibility into the code container and Dev is going to rapidly changing the code in that container, visibility into how the code is performing, how it is interacting with its supporting libraries and how the application stack is interacting with the supporting operating system and infrastructure is essential. Developers now own the entire stack at least from the base image upwards and ops are focused on deploying micro-services with Docker containers and have very little or no visibility into how the services are constructed or how they actually work, hence the increased need for APM.

The Docker Hub

Docker has released the Docker Hub, which is a centralized repository of official and quality images. This has the potential to greatly accelerate the rate that code can get released into production as all of the components of the application stack except the custom code itself can now be downloaded and assembled into a working application system.

However while this is a great innovation for developers, it creates problems for Operations as now Ops is going to have to support stacks downloaded and assembled from a third party website into which Ops has less visibility than was the case before. So this is another case where a layer of abstraction (the Docker container) brings with it tremendous productivity benefits.

Docker and Diverse Deployment Environments

One of the benefits of Docker is all of the libraries that an application needs can be isolated into the same container or a set of linked and nested containers with the container that holds the

code. This allows that container or set of containers to be deployed anywhere there is a valid and supported distribution of Linux (and soon Windows) running. This includes internal data center environments like Linux on bare metal, Linux on Type 1 hypervisors like vSphere and KVM, Linux in hybrid clouds, and Linux in public clouds like Amazon, Google, and Microsoft.

Kubernetes

Kubernetes is an open source Docker container orchestration tool developed by Google. It manages workloads within a cluster, automatically scheduling nodes based on demand while ensuring state is valid and meets user requirements. Whereas Docker is focused on the management of individual containers, Kubernetes groups and logically associates containers for management of entire service-oriented - or microservices - architected application systems.

Mesos

Designed to run on every machine in a data center or cloud, Apache Mesos provides APIs to manage CPU, memory, storage, and other computer resources (virtual or physical), allowing distributed systems to be built and run in a fault-tolerant and elastic way. Applications, such as those running in containers and orchestrated by Kubernetes, can then utilize these APIs to schedule and allocate the various computer resources needed. Kubernetes on Mesos is a significant enabler for enterprise adoption of Docker and containers.

Docker and Micro-Services

For quite some time, especially for net new applications deployed in the cloud, there has been a trend to dis-aggregate “applications” into “micro-services”. This is basically the idea of service oriented architecture taken to its logical conclusion where every instance of every service runs in its own instance in the cloud. However, this has come at a price as before Docker a separate cloud instance needed to be provisioned for each instance of each service. With Docker instances of services can be isolated into containers. Organizations can then decide whether to run one or many containers on each instance of the operating system. Shipping many micro-services means managing many application stacks and which makes packaging them as containers a much easier way to manage the deployment of many micro-services to many environments. When there are many micro-services relying on each other to provide application functionality, it can be difficult to pinpoint the root cause of problems. The latest generation of APM tools like AppDynamics make it easy to get complete visibility by providing distributed transaction tracing that gives you a complete view across many micro services.

Containerized micro-services are going to take what has happened in the world of highly distributed service oriented applications to a whole new level as what is now one application running on one JVM will get broken up into many containerized micro-services. The ability to trace the flow of work, time the flow of work and find bottlenecks and errors across this now fine grained and highly distributed environment will be critical to the ability to manage containerized applications in production.

Since these containerized applications will be both highly distributed and rapidly changing, this tracing will also have to be implemented in a “zero-configuration” manner. Therefore, the ability of AppDynamics to automatically discover transactions and business services and to constantly and automatically rediscover transactions and business services as applications change becomes a critical requirement for managing these new applications in production.

VI. Summary and Conclusions

Application run times (Java, .NET, PHP, Python, Ruby, etc,), compute virtualization through Type 1 hypervisors, network virtualization, storage virtualization and now containers like Docker all serve to abstract an application from its supporting hardware resources. Once applications are abstracted from their underlying execution environment in this manner, it is essential to directly measure the performance and operation of the application code in production in order to ensure that the application system is delivering a high level of service to its end users and business constituents.

Docker also creates a series of new problems that must be addressed by an APM solution. Many organizations will have some learning curve with Docker/containers and not everyone will be willing or able to partition their apps correctly: once again, there will be a need for smart, application performance monitoring tools to help make sense of what's actually running where.

Isolating the code in a container is great for the developers of the code, but it creates problems for operations as that code is frequently change, broken up into many micro-services, and distributed across diverse execution environments.

APM solutions like AppDynamics that are specifically designed to operate in diverse, dynamic, highly distributed, and complex environments are, therefore, essential to the successful operation of these applications in production.

VII. About The Virtualization Practice

The Virtualization Practice provides analysis, commentary and resources on current Virtualization and Cloud Computing news, events, and community. We break virtualization into topics, and feature a world class expert in that topic as the analyst for the topic. Topic Analysts are responsible for writing original, objective, analytical posts in their area of expertise, for writing and maintaining a white paper on their market and the vendors that serve that market, and for assisting sponsoring vendors with marketing activities.

Bernd Harzog is an Analyst for Cloud and Virtualization Performance Management. Bernd is also the CEO of APM Experts, a consulting and analysis firm focusing upon this market, vendor strategies in this market and customer use cases in this market. Bernd was formerly a Gartner Group Research Director focusing upon the Windows Server operating system, CEO of RTO Software, VP of Products of Netuitive and has been involved in vendor and IT strategy sinc