The Case for Containers: Why Docker is Coming on Strong

You can't go too far without seeing the word Docker among technology circles these days. This product/concept/company is getting some significant airtime over the course of the last few months, and there is no sign of that slowing down.

So, what exactly is the big deal around Docker? It's all about the containers. Well, mostly about the containers. Let's have a look at what the container concept is all about and how Docker comes into play among it.

The Container Concept

It's been said written in many forms, but for the TL;DR version, here is my quick summary:

> Containers are encapsulated application environments that contain the necessary libraries to drive applications, with a common, open interface to run on top of a container engine such as Docker Engine. The container is lightweight as a result, and runs in isolation to provide the best flexibility and portability. The engine can run on multiple platforms, including bare-metal, which allows the containers to be deployed into on-premises and public clouds using the same application builds.

Based on this quick description, we can easily see why it's attractive. We have to be very clear though in what problem is being solved by containers. This allows us to attach the solution to the problem. Technology for technology's sake is not the right reason to deploy something.

The Problem

Application development and deployment creates a number of challenges. Included among them are:

- Application dependencies (libraries, hooks into the OS etc.)
- Consistency (many apps = many potential configurations)
- Programmability (repeatable, scriptable, measurable)
- Policy (business rules and other sometimes intangible dependencies)

These problems have been handled in a number of ways up to now. Some of them are clear constraints, but others are actually a collection of constraints which could include a lot of business processes.

Now that we've seen some of the problems, let's see what containers can do to move towards some solutions.

What Containers Solve

Based on the requirements above, we can now look at how containers are able to solve the problems, or at least get us closer to solving them. Remember that the real driver to anything we build a technology for is to answer an existing problem that exists to reduce cost, or increase the value in some way for the business.

Application Dependencies

Remember how Java was all about “write once, run anywhere”? While it’s conceptually true, there are lots of issues around library and dependency versions when running Java. The same holds true for Ruby, Python, and many more languages. By moving the libraries for the language runtimes into the container, you isolate it from other application environments which reduces version issues and collisions with fighting for runtime access and contention.

Consistency

You’ve been there before. It’s the moment that you deploy an application to a production server with a set of instructions handed to you by someone, and half way through you hit a snag of some kind.

Maybe the server is different, or maybe the configuration of the application or folder structure is not the same as the development environment. Sometimes it may have even been developed for another OS!

When adding the container as a consistent, predictable wrapper for the application environment, we ensure that all of the underlying requirements are already met. The dependencies are completely encapsulated inside the container.

Programmability

I’ve done my fair share of application deployments using typed out instructions, or worse, just verbal ones which invariably didn’t end up being the actual steps needed to complete the deployment.

Making the application deployment programmable with a common configuration process is as close to a guarantee as you can ever get that the application will look identical in the development environment, as it will in QA, and ultimately in production where it matters the most.

Programmable configuration and deployment allows you to use your configuration management tool of choice to deploy your apps, such as Puppet, Chef, SCORCH, SCCM, vCAC and many more.

Policy

After years in financial services environments, and dozens of IT audits, I can assure you that integrating policy management into your application deployment strategy is of absolute importance.

Although containers don’t wholly cover policy as a part of what they do, they will provide assurance that the deployment from one environment to another is more audit friendly.

We will leave policy for now, but in a later blog post I’m going to go much further into how policy management and containers come together.

What Containers Won’t Fix

I like Docker, and I like containers as a deployment model, but before you run out and tell the CIO that you’re changing to a “containers first” model, let’s understand some of the adoption challenges you may face.

Here are a few points to think about:

- New tools don’t fix old applications

- Cultural acceptance inside your organization
- Performance
- Full manageability

Have you ever heard the phrase “money won’t fix money problems”. The message in that phrase is that by simply throwing resources at the situation won’t fix the core issue that caused the problem in the first place.

Adopting a new development and deployment model is not just a flip of s switch. There are many factors, most importantly ones wrapped around people and processes, that can impede changing to a container deployment model.

The cultural changes we talk about are similar to those for enabling a DevOps culture at your organization. There are significant changes to the process and people management that will lead toward better application development and deployment practices. This also holds true for containers to a degree.

Note that you don’t have to be a DevOps shop to use containers, nor will using containers ensure that DevOps methodologies are in place. It can be a great pairing, but the culture shift has to be there to embrace containers as a platform

Performance and Manageability

I’ve separated these two out from the other two constraints, and paired them together for a reason. Performance management and overall management are particular challenges with containers as they are today.

Containerizing your applications is only dealing with the deployment and packaging. There are no baked in tools with container technologies that ensure performance or provide a robust management tool to understand your application environment.

Docker, for example, is meant to help the container. It lacks a monitoring tool that natively ensures that you are getting optimal performance for your application. It is designed so that you can easily deploy more of your application containers, provided you’ve designed your application to be able to horizontally scale.

On management, there is also a lack of s robust management tool that can provide an overall view of the performance and topology of your Docker environment.

New tools are coming up that are attempting to take on these tasks, but there is still much more development in these areas to be done. It will come in time, especially as enterprise customers make the foray into using Docker as a part of their IT practice.

Study Up, it’s Coming

The important thing about Docker and container concepts is that they are inevitably going to play a role, perhaps a significant one, in organizations.

I recommend that you take some time to do some reading, and I’ll be preparing some Docker 101 goodness to help you along the journey.

Time to start to containerize!