How Linux Containers Change The Development and Delivery of Applications

1. SUMMARY

In a bid to create applications reliably, a separation has evolved in most organisations between "Development", which builds applications, and "Operations", which commissions and runs those applications.

There are good reasons for this separation of concerns, but it is increasingly hard to maintain in the face of the challenges confronting business.

Organisations now need to respond rapidly to demands and quickly deliver services that are flexible. This is difficult to do when they have a rigid structure to produce and deliver applications.

Development is moving towards new models of "agile" development, but a more radical movement has emerged which believes the rigid separation of development and operations itself is holding back the production of new services and making companies less agile.

"DevOps" has been proposed as a new way of working based on collaboration and communications between the different parts of the organisation.

But DevOps is just a theory, and it will not develop into something more tangible unless it is adopted by an organisation which is clear about the reasons why development and operation were separated in the first place - and which is ready to adopt technologies which enable them to be reunited safely.

This paper argues that DevOps can be implemented productively if the two disciplines adopt a "cloud" model for production and consumption of services. It also argues that Linux containers are a very powerful and convenient technology change to enable this model as long as they are used with appropriate automation and management technologies.

2. HOW DID WE GET HERE?

On the face of it, there are very good reasons for the separation of IT concerns. Developing code is a very different job from maintaining and supporting that code. Cars are built in a factory; they are maintained and supported by a garage. Why should code be different?

Ultimately, users need different things from developers and operations departments. Business users' time to market and flexibility in the business are key drivers. Developers should understand their needs and be good at designing and creating the systems that meet them. Operations people should be able to maintain the status quo and keep things running reliably and securely.

Development needs to make new functionality quickly, while Operations has the goal of long term stability, control and governance. That separation is important - and it is more important the larger your organisation is. A company developing and delivering a myriad of applications really needs to have a reliable structure to support and manage them.

Developers are often incentivised on how quickly they can deliver code, while Operations is usually incentivised on the length of time between failures. Taken to their (il) logical extreme this would lead to a Development department delivering code that is not ready, and an Operations department with a vested interest in never allowing any new code to be implemented!

In the DevOps model, the Operations side needs to support rapid delivery of multiple versions of an application. At the same time, testing and a concern for reliability must be built in throughout the Development process.

Ultimately though, the goals of the two different departments are not worlds apart, but the teams do approach them from a different perspective.

A good cloud infrastructure supports both roles when it comes to achieving their goals in relation to the likes of stability, security, reliable infrastructure, fast provisioning and efficient staging.

3. WHY DEVOPS NOW?

If Development and Operations have been separated for years, why is this suddenly a problem?

Firstly, the problem is not new. Large and complex IT projects tend to be handled with pure classical project management methodologies. Therefore those projects tend to be less flexible in adapting to changing requirements, responsible roles gain feedback very late and, at times, projects can overrun in remarkable ways as far as time and budget plans are concerned.

More rapid and more incremental development avoids this problem. It is a good idea. But it is more than that. It is essential. Every organisation faces competition from other bodies that are applying more agile techniques. There is a change in the speed of business. Cloud services have emerged which allow applications to scale up quickly, and new start-ups are delivering services fast.

While large systems used to be rolled out just a few times per year, the expectation now is that new features can be brought to market in mere days or weeks at the most.

4. DEVOPS - ENABLED BY THE CLOUD

At this point, anyone outside of IT will ask an obvious question. We can see WHY faster development is needed. We now have a rough idea WHAT DevOps is: it is the closer alignment of the interests of Development and Operations.

But the question is - HOW does this occur?

Is DevOps just another management strategy? Is it yet another business process cult? Are we expected to believe that we can change our organisations by having a different set of inter-departmental meetings?

No. DevOps does require a different approach to managing the development process, and a change to the way the business does IT, but it is far more than a management buzzword.

DevOps is enabled by solid cloud infrastructure technology. In fact it is fundamentally underpinned by those very technologies threatening the status quo.

On one level, we can see DevOps as the implementation of a cloud model within the IT department. Cloud technologies include automation and virtualisation that allow any function

to be delivered as a highly standardised, massively scalable service in the cloud. Storage as a service and infrastructure as a service are just two examples.

Development produces applications, but it needs platforms on which to develop and deliver those applications. If an organisation adopts cloud internally, then Operations can be the cloud service that provides and supports the platforms where Development does its work.

Because cloud services are standard, and can be replicated and scaled, Development is now working on a platform which is functionally identical to the eventual delivery platform. Fast incremental delivery of applications is now possible.

A cloud service model also sets out a clear relationship between Development (the consumer) and Operations (the provider).

It provides an umbrella framework for the Dev and Ops teams, with defined roles and responsibilities. Coupled with a view on common business goals, this leads to better communication and collaboration as the needs and incentives becomes aligned to higher business level goals.

If Operations is automated where possible, it reduces the need for specialised staff, and allows Development to take a self-service approach.

5. PROVIDING THIS CLOUD THROUGH CONTAINERS

When we talk about a cloud model, we can be a bit more specific. Most cloud services have been based around "virtualisation", where one physical server supports multiple "virtual machines". Software called a hypervisor creates these virtual machines, so any program appears to have a whole server to itself.

When you rent a server on Amazon or Google Compute Engine (GCE), this is what you get.

But there is a more efficient way to do this. Running an application does not need a whole copy of a virtual server. That is why the idea of "containers" has emerged.

Containers look and feel like a server to the application running inside them, but are actually slimmed down with a much smaller overhead than a full virtual server. They provide CPU processing power, memory, network access and disk storage.

Containers are exactly the kind of service that Development wants from Operations. They provide space for development work, can provide the required libraries and components

for applications, and an application can be easily moved into a full operational environment supporting the same container technology.

Containers mean the same environment can be used for development and delivery - exactly what DevOps demands. They are standardised blocks in the development lifecycle: the developer takes the container, and once the application is working, an application can be moved through the other phases of its life cycle.

Containers can even be tailored to particular jobs, such as Java, Java EE, Ruby, etc.

6. WHAT ELSE DO WE NEED?

Containers have taken off rapidly within the open source community and are fast becoming the de facto standard for the vast majority of cloud services.

Today, technology in Linux enables the allocation of resources such as CPU and Memory that can be isolated and distributed between containers. The underlying pool of resources can come from physical or virtual machines. They are a core part of Linux, but they need standardisation, management and automation. They also require careful security considerations to be implemented for multi-tenant environments.

For Operations to provide an efficient cloud-like service, the handling of containers must be automated as far as possible, so the containers can be provided on a self-service basis and be managed independently for the creation and delivery of applications.

That requires a standard format for packaging and distribution of applications with dependencies. A clear leader today in providing such a format is the Docker project.

Docker uses an image layering concept to allow Operations to create and manage containers, and allows Development to put content - the application - inside them.

Docker provides a packaging format that is fast becoming popular and containers created in Docker format provide a standard, so they can easily be implemented across an organisation or even between organisations working on joint projects.

But there comes a point where these projects become too complex for the basic tools within Docker.

In a large environment, organisations will find they need a sophisticated platform-as-a-service environment, where their containers can be supported across the full application life cycle - development, testing, quality assurance, production, etc. They also need an environment which can connect and co-ordinate applications running in multiple containers.
OpenShift, available for Red Hat Linux, is just such a service. It maintains services underlying the application and scales them as necessary.

Essentially, OpenShift is a management structure ideally suited for containers. It was utilizing container technology right from beginning and is now being enhanced to allow containers in Docker format to be built and deployed smoothly on the platform.

7. CONCLUSION

DevOps is a great ideal. It may even be a matter of corporate life and death. More collaboration is essential between Operations and Development if organisations are to respond, quickly and at scale, to the current business environment.

But any DevOps adoption will fail unless it has three things:

1. An understanding of the necessary skill separation that led to the creation of Development and Operations functions

2. An appropriate technology to provide the needed collaboration while preserving the skill separation

3. An enabling culture and enabling technology as a foundation

Linux containers, Docker and OpenShift combine to provide an environment where the necessary skills can be deployed in the right way.

Containers enable DevOps while preserving the fundamental - and necessary - concerns of Development and Operations.