tier 2 CONSULTING

Red Hat JBoss Middleware.
It's all we do – let's do it together

OpenShift Container Platform – Q & A

OPENSHIFT

With the recent release of Red Hat's OpenShift Container Platform v3, Tier 2 puts the important questions (like "what is it and what does it do?") to Red Hat Senior Solutions Architect, Ian Lawson:

What on earth is a 'Container'?

A Container is a slice of the Operating System. Put simply it is a secure, self-contained and sandboxed part of the Operating System that can run independently of, but is controlled by, the underlying Operating System.

So it is Virtualisation?

No. Virtualisation requires a process intensive layer that converts all the actions of the virtualised Operating System into the underlying Operation System – Containers are slices of the Operating System and can therefore run at a faster rate than Virtualisation.

Why should I be interested?

Container technology allows you to run many varied Operating Systems and Application stacks much more efficiently on a single platform. In addition the Container itself, which is made up of a set of layered file systems, is transportable between any hosts that can support Containers, making it very easy to move, replicate and control.

That doesn't sound very safe – is it secure?

The underlying Container technology, called 'Docker', was designed and implemented in a security-agnostic way, meaning that it is up to the Operating System hosting the Containers to implement security. So no, if you use the lowest form of technology, 'Docker', it is not safe.

So how do I use these Containers safely?

Through the use of two additional technologies on top of the Containerisation technology. Google have produced a technology called 'Kubernetes' (which, like Docker, Red Hat contributes massively to) that adds a control layer on top of Containerisation, providing much better aggregation and control. This suffers from some of the same security issues as Docker. On top of this technology, Red Hat has produced a system called 'OpenShift', which is currently the only Enterprise strength secure Container orchestration system available.

Why is OpenShift secure when the underlying technologies are not?

OpenShift itself uses the Containerisation technology to host the components needed for OpenShift. It does this by treating all Containers running within itself as one of two types – privileged and non-privileged. All OpenShift core components (i.e. the bits that are OpenShift itself) run as privileged, whereas all the user-created and hosted Containers are forced to run as non-privileged. This mechanism means that OpenShift controls the execution of potentially insecure Containers in such a way as to disallow them the access they need to be dangerous.

This sounds good. What's the catch?

There is no catch. OpenShift implements a RBAC (role-based access control) approach which gives the administrators total control over what user can do what action. In addition, OpenShift applies the OpenSCAP standard to the containers, which is a compliance mechanism for ensuring that Linux systems remain secure in the way they were created. These approaches mean that the Containers hosted within OpenShift, and OpenShift itself, are highly secure and locked down, which addresses an inherent insecurity of the underlying Container technologies.

redhat

PREMIER
BUSINESS PARTNER
Solution Provider
Middleware Solutions

tier 2 CONSULTING

Red Hat JBoss Middleware.
It's all we do – let's do it together

Tier 2 Consulting Ltd
Business and Technology Centre,
Bessemer Drive, Stevenage,
SG1 2DX

Tel:
+44 (0) 1438 310124

Email:
info@tier2consulting.com
support@tier2consulting.com

So that's security covered, what can I do with Containers?

There's not a lot you can't do with Containers. Put simply, they are slices of the Operating System, so you can host applications, services, web-based functionality, basically anything you can do with a standard Linux Operating System. Typical use-cases for Containers include standing up multiple, highly-available Webservers, Enterprise Service Buses, web-application end-points and the like.

So, I can take my existing applications and put them into Containers?

Absolutely. In addition, because of the functionality provided by the orchestration components within OpenShift, you can massively simplify the applications as well. You no longer have to work out the topology of your applications within the applications themselves – using Containers allows you to abstract away the data-sources, the highly-available failover operation, the disaster recovery options, the networking. In fact, using OpenShift gives you a number of features that would be very difficult to implement as standard, such as auto-scaling, where an Application can be made to replicate or shrink depending on the metrics of the usage of the Container (e.g. CPU usage, network load and disk usage). All of these features are applied to the Application as part of the deployment mechanisms, meaning that there is a distinct divide between the Application functionality and the hosting functionality. Put simply, you no longer need to change the code of your application to change the nonfunctional behaviour. This greatly simplifies the code you need to produce for the Application, removing a lot of the 'boilerplate' baggage that used to make Applications so hard to maintain.

I've heard about the concept of 'Microservices'. Do these Containers have anything to do with that?

Microservices are a different way of implementing systems, with functionality broken down into many small, manageable pieces, and Applications becoming a set of Microservice calls instead of the old-fashioned vertical, huge, hard-to-maintain Applications. OpenShift was specifically designed to make the hosting of Microservices extremely easy, through the use of specific Application stacks offered natively within the platform such as the Apache Camel based 'Fuse' implementation. The nature of Containers, especially via OpenShift, around the multiple replication, fire-and-forget, dependency-injection model (which, put simply, means each of the Containers is a sausage-machine) means that the creation of a framework for Microservices is extremely simple.

My present system for development involves multiple, separate environments for the sake of development, testing and production.

If I use OpenShift does this mean all of my environments will be in the same place, as this sounds dangerous?

Absolutely not. OpenShift is a fully distributed system and uses an approach called 'labelling' to allow Containers to be specifically targeted to execution environments. Also the basic currency of the system, which is Containers and Deployment Configuration, means you can easily transition 'Applications' from one system to another. So there's no mandate from the OpenShift side to force you to either host all of your systems in the same place, or to distribute. It is entirely up to you.

For example, you could have a singular OpenShift with multiple sandboxed environments (i.e. devops, testing, QA, pre-production, production) and strictly control the deployment of Applications using labelling, or you could have physically separate and secure OpenShift instances with a manual stage-gate for transferring Applications from development into testing, and then testing into production.

The choice is entirely yours, OpenShift is flexible enough to cater for any topology of usage.