InfoWorld

November 18, 2015

GET TECHNOLOGY RIGHT®

INSIDER

Review: OpenShift 3 rocks Docker containers

Red Hat's robust, easy-to-use, and highly scalable PaaS now builds on Docker, with some temporary limitations

LAST YEAR OPENSHIFT 2 WAS MY FAVORITE open source PaaS. I said at the time, "OpenShift is outstandingly easy to use, manage, and install, and it presents little learning curve for developers familiar with Git and administrators familiar with Puppet. Enabling automatic horizontal scaling is as simple as checking a box in the application configuration. Automatic gear idling is enabled by default and allows for very high application density. And updating an application is as simple as doing a git push. For both developers and operators, OpenShift fulfills the promise of PaaS."

Since then, OpenShift has undergone a complete rewrite to use Docker containers, instead of "cartridges" and "gears," to deploy applications. In theory, that should make the OpenShift PaaS even simpler to use and give it an even larger pool of languages and applications, thanks to the containers already on Docker Hub. But how does it work in practice? Let's find out.

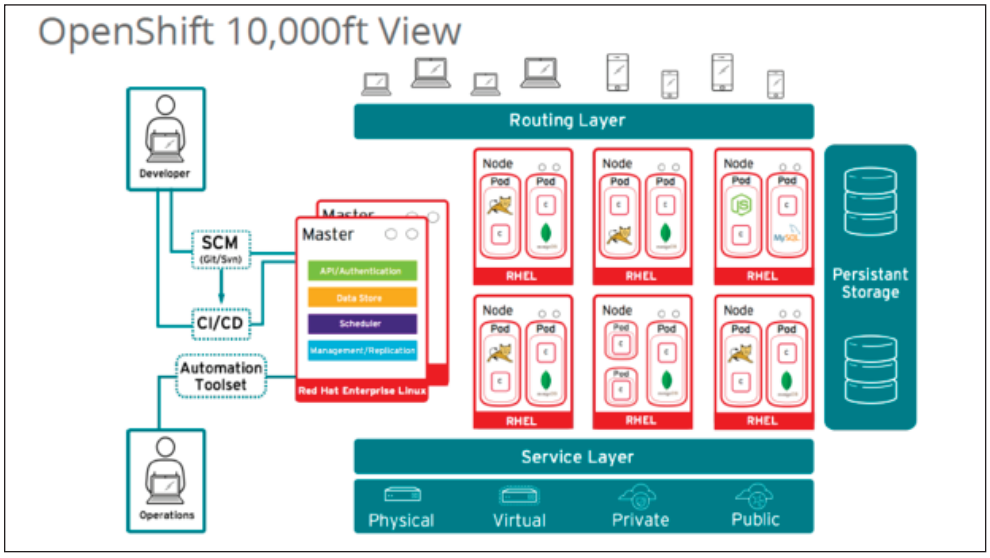

Figure 1: Diagram of the OpenShift 3 system.

OpenShift 3 architecture

As you can see in Figure 1, OpenShift 3 is still built on Red Hat Enterprise Linux. Beyond that, however, almost everything has changed. The running unit is no longer a "gear" — it's a "pod," and the pod consists of one or more Docker containers that run together on a "node." Language environments are no longer "cartridges"; they are Docker images, which are combined with code using a source-to-image tool. Nodes are instances of RHEL where apps will run.

Scheduling, management, and replication are now part of the master, which implements Kubernetes on top of RHEL. A service layer talks to underlying physical or virtual hardware or to public or private clouds. A routing layer connects the apps to the Internet so that they can be used by clients running on computers or mobile devices.

A developer can do self-service provisioning on OpenShift 3 using the oc command-line interface or the Web console. A few functions are currently implemented only in the CLI, but will be added to the Web console in OpenShift 3.1.

OpenShift SKUs and installation

OpenShift is really four different but related products: OpenShift Origin, OpenShift Online, OpenShift Dedicated, and OpenShift Enterprise.

OpenShift Origin 3 is open source and flexible: You can run it as a container by itself, as a cluster using Ansible, or in the public cloud using Amazon Web Services or Google Cloud Engine. This is the upstream code and clearly demonstrates the change from OpenShift 2. There's an OpenShift Origin 3 Vagrant VirtualBox VM you can install in a few minutes.

Online is hosted on the public cloud and used to be periodically cut from the Origin code. Online still uses gears, however, and that's an obsolete term in OpenShift 3. That's because Online is still running OpenShift 2.

OpenShift Dedicated offers a dedicated, customized, and managed application platform in the public cloud, hosted on Ama-

Copyright © 2015 by InfoWorld Media Group, Inc., a subsidiary of IDG Communications Inc. Posted from InfoWorld, 501 Second Street, San Francisco, CA 94107. #C45080 Managed by The YGS Group, 800.290.5460. For more information visit www.theYGSgroup.com.

zon Web Services in any availability zone. It's powered by OpenShift 3 Enterprise and administered by Red Hat.

OpenShift Enterprise is the most stable and typically the biggest deal to install. You need to create a master, the infrastructure (the registry, router, and storage), and at least two nodes, and that involves starting with RHEL and adding Docker, Kubernetes, and finally OpenShift. Ideally each master and node will have at least 8GB of RAM for production use. Enterprise is the version that competes with Pivotal CF and Apprenda, and it's the version I reviewed hands-on, using a "roadshow environment" set up for me by Red Hat.

You can install and run OpenShift Enterprise on a laptop, and that's what Erik Jacobs, the principal technical marketing manager for OpenShift Enterprise, did when he first demo'd the product for me. Jacobs calls it "a three-VM" configuration, which uses one VM as master and infrastructure host (registry, router) and two nodes to host apps.

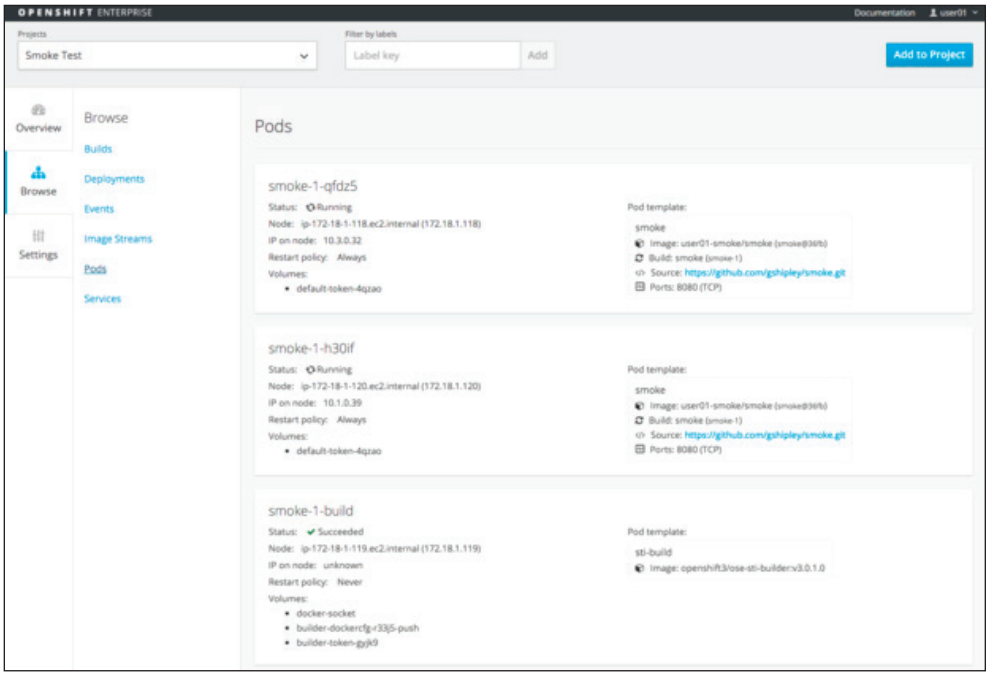

Figure 3: The Pods view of the OpenShift 3 Smoke Test app from the Web console. Two of the pods contain the running app; the third shows the successful build.

Developing with OpenShift 3

To understand how OpenShift Enterprise currently works for development, I eschewed my usual random exploration of the PaaS and instead went through Red Hat's nine labs. Lab 1 is basically to install and introduce oc, the CLI tool. Note that the OpenShift Enterprise version of oc is different from the OpenShift Origin version of oc in relatively minor ways, although they are enough to keep a few of the lab steps from working as described.

Lab 2 has you running a Smoke Test project, learning a little about using oc, and learning a little about using the Web console, as shown in Figure 2.

On the right-hand side of Figure 2, you may be able to read the helpful definitions provided in the Details pane of the Project Overview screen. These are the following:

- A pod contains one or more Docker containers that run together on a node, containing your application code.
- A service groups pods and provides a common DNS name and an optional, load-balanced IP address to access them.
- A deployment is an update to your application, triggered by a changed image or configuration.

On the left-hand side of Figure 2, you can see two pods running at the bottom and the service URL at the top. The Smoke Test is basically a "hello, world" Web app. The entire source code is the following:

```
<?php
echo "Welcome to the OpenShift 3
  Roadshow Smoke Test Application";
```

Figure 3 shows the two running Smoke Test pods and a completed Build pod.

Lab 3 leads you through getting an image from Docker Hub, the Kubernetes Guestbook application, and deploying the image onto OpenShift Enterprise to create a running pod and service. Along the way you'll encounter this message:

> *Note: It is important to understand that, for security reasons, OpenShift 3 does not allow the deployment of Docker images that run as root by default. If you want or need to allow OpenShift users to deploy Docker images that do expect to run as root (or any specific user), a small configuration change is needed.*

In other words, you can't really run any random Docker image you want on OpenShift under its normal security settings. I'm told that a future version of OpenShift Enterprise, possibly 3.1.1, will have a way to automatically provide a sandbox for Docker images that want to run as a specific user: the running image will think it's root within its sandbox, but it won't

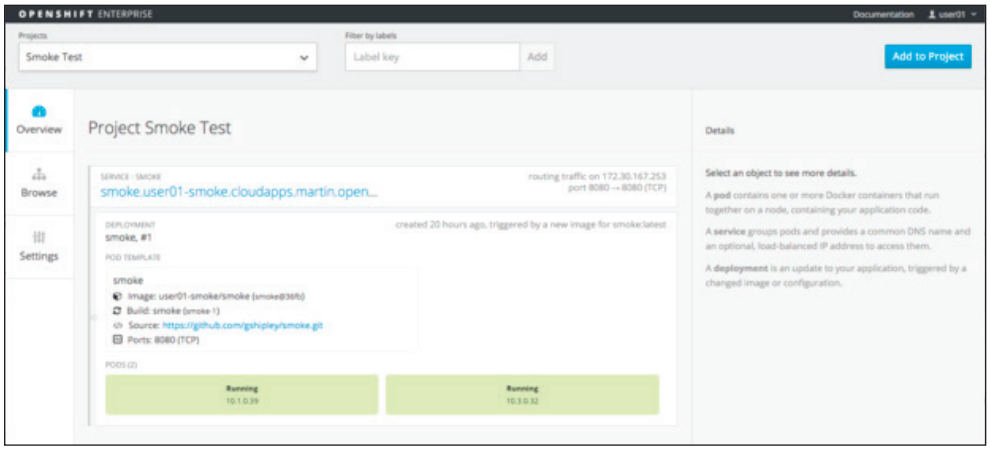

Figure 2: Overview of the OpenShift Smoke Test app from the Web console. See the details on the right that explain pods, services, and deployments.

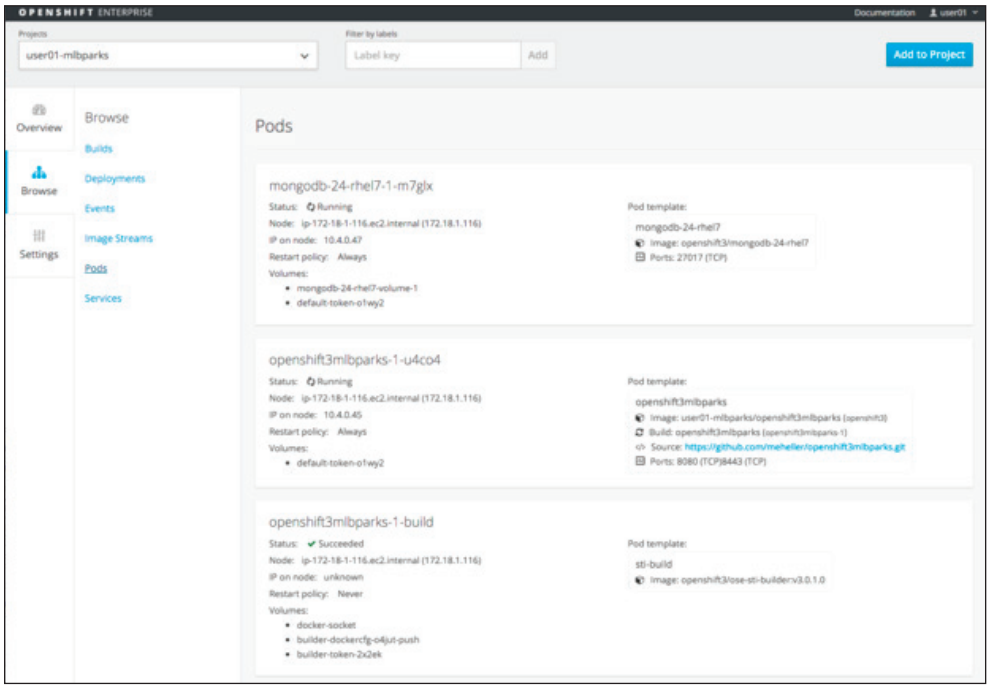

Figure 4: The pods for the OpenShift 3 mlbparks app, which maps the major league ballparks. The first pod is the running MongoDB database, the second is the running Java app, and the third is the completed build.

be root for the outer OpenShift Enterprise environment.

Lab 4 teaches you to create routes for your services, so they can talk to the outside world. Lab 5 teaches you to scale your application with a command line such as:

```
$ oc scale –replicas=3 rc guestbook-1
```

This sets the desired number of replica pods for the guestbook-1 controller to three in its replication controller (rc). The rc then looks at the actual number of pods in the service, sees there is only one, and creates two more. If you kill one pod, if it dies, or if it stops responding and needs to be killed, the replication controller will create a new one almost immediately.

Lab 6 introduces you to s2i (Source-to-Image), a "magic" service (and free open source project) that "produces ready-to-run images by injecting source code into a Docker image and assembling a new Docker image which incorporates the builder image and built source." In this lab you fork a JBoss app repository on GitHub and use your forked repo with s2i and the JBoss EAP image to create, build, and run an app:

```
$ oc new-app jboss-eap6-
  openshift~https://github.
  com/<YOURUSER>/
  openshift3mlbparks.git
```

Once it builds and runs, you create a route by exposing the service:

```
$ oc expose service
  openshift3mlbparks
```

That service displays the map of North America, but without any ballparks.

In Lab 7 you add a MongoDB pod to the service, along with some environment variables providing the database credentials. Then you set the same environment variables into the deployment controller (dc); OpenShift immediately detects the changed environment, bumps the version of the dc, and redeploys it, this time with the app connected to the database so that the ballparks display. Figure 4 shows the pods for the new deployment.

In Lab 8 you set up a Web hook in your GitHub repo that will notify your OpenShift Enterprise deployment controller of any code pushes in the repo. Then you modify the source code for the Web page, check it in, and push the check-in to GitHub. OpenShift sees the hook notification, pulls the changes, rebuilds the app, and on a successful build increments the version of the dc and redeploys it. Figure 5 shows the openshift3mlbparks app after a couple of check-ins.

Lab 9 teaches you how to use application templates, which speed deployments. It doesn't teach you how to create templates, but that can be found in the developer's guide.

OpenShift Enterprise 2 had a few capabilities that are currently missing from OpenShift Enterprise 3. An important one is gear idling: OpenShift Enterprise 3 currently has

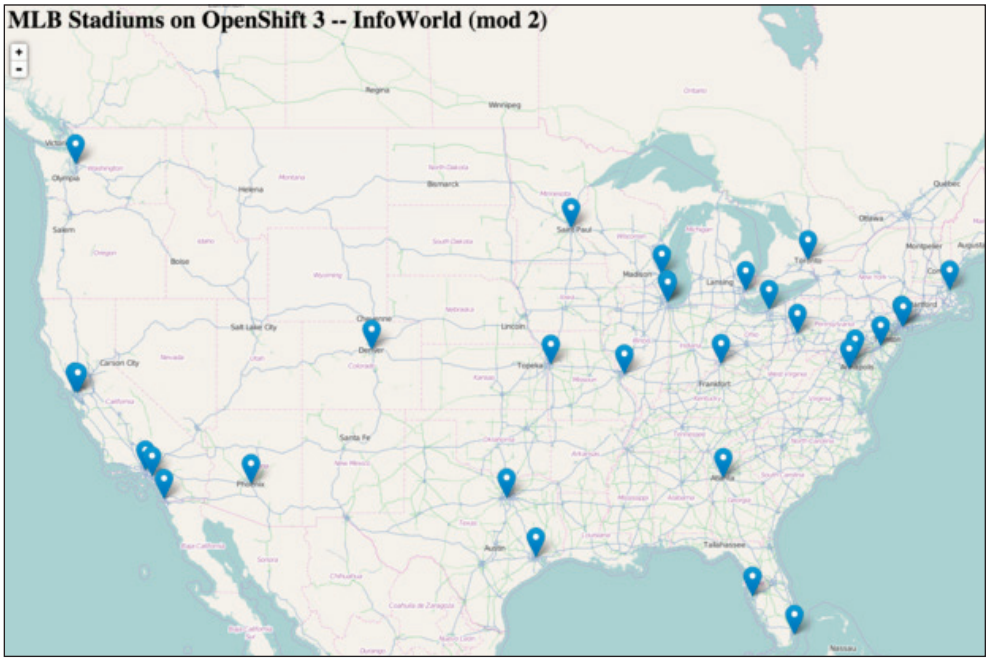

Figure 5: The MLB ballparks app running on OpenShift. The app is implemented in Java with a MongoDB database. The "InfoWorld" and "mod 2" parts of the title came from changes I checked into the source repository, which triggered a Web hook that caused OpenShift to rebuild and redeploy the app.

no way to "park" pods that aren't getting any traffic. That is supposed to be coming in OpenShift Enterprise 3.2. Similarly, OpenShift Enterprise 3 cannot automatically scale pods when load goes up or down; I'm sure that will be in a future version.

On the other hand, OpenShift Enterprise 3 can support blue/green deployments. Rolling back from a bad "green" deployment is easy:

```
$ oc edit route/blue
```

The switch to Docker containers hasn't opened up as many images for use in OpenShift Enterprise 3 as I would have expected, due to security issues. Once a future version of OpenShift Enterprise implements a sandbox for images that want to run as root, that should be fixed.

On the whole, I'm impressed with OpenShift 3. I can still say with a straight face, "For both developers and operators, OpenShift fulfills the promise of PaaS."

— Martin Heller — Contributing Editor

Martin Heller is a contributing editor and reviewer for InfoWorld. Formerly a Web and Windows programming consultant, he developed databases, software, and websites from his office in Andover, MA, from 1986 to 2010. More recently, he has served as VP of technology and education at Alpha Software and chairman and CEO at Tubifi.

OpenShift 3.0 / Red Hat

AT A GLANCE

OpenShift Enterprise 3 has been completely rewritten to use Docker containers. While a few desirable features have been deferred to future versions, the PaaS is robust, easy to use, and highly scalable.

OpenShift Origin: Free; OpenShift Enterprise pricing starts at $4,000 per year depending on configuration (core/VM or socket pair). OpenShift Online is still running version 2; it is not covered in this review. OpenShift Dedicated is open for Early Access, running version 3 in all AWS availability zones.

PROS

- Wide assortment of containers, languages, Web frameworks, databases, and application stacks available and supported
- Easy and fast self-service deployment
- Git integration at the source code level
- Runs on any hardware, cloud, or VM that supports Red Hat Enterprise Linux
- Can run any Linux Docker container that meets security requirements

CONS

- Many common Docker containers do not meet security requirements
- Does not currently support the equivalent of "gear idling"
- Does not currently support Windows Docker containers
- Pod scaling is currently only done manually from the CLI

InfoWorld Scorecard

	Ease of use	Management	Breadth of Support	Documentation	Instruction and setup	Value	Overall Score
	20%	20%	20%	15%	15%	10%	
OpenShift 3.0	9	8	9	8	9	9	8.7 ★★★★½