Developing applications on OpenShift as regular developers

DevConf CZ 2019
C228
Friday, January 25 • 3:00pm - 4:50pm
Who do you see on stage?
What’s all this **OpenShift** and **Kubernetes** thing?
Kubernetes is the new kernel. We can refer to it as a “cluster kernel” versus the typical operating system kernel. This means a lot of great things for users trying to deploy applications. It also leads to a lot of the same challenges we have already faced with operating system kernels. One of which being privilege isolation. In Kubernetes, we refer to this as multi-tenancy, or the dream of being able to isolate tenants of a cluster.


Have you ever wondered why you are deploying your multi-platform applications using containers? Is it just a matter of “following the hype”? In this article, I’m going to ask some provocative questions to make my case for Why Kubernetes is the new application server.

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.
OpenShift extends Kubernetes to make developer lives easier
The Kubernetes platform for developers
HOW OPENSSHIFT ENABLES DEVELOPER PRODUCTIVITY

- **Self-service Infrastructure**
- **Consistent environments**
- **Automated build & deploy**
- **CI/CD pipelines**
- **Configuration management**
- **App logs & metrics**

**CODE**
- **SPRING & JAVA EE**
- **LANGUAGES**
- **LINUX**

**REVIEW**
- **MICROSERVICES**
- **DATABASES**

**DEPLOY**
- **FUNCTIONS**
- **APPLICATION SERVICES**
- **WINDOWS*”**

**MONITOR**
What’s all this Developer Experience thing?
Developer Experience (DX) is the equivalent to User Experience (UX) when the user of the software or system is a developer.
“It just needs to work! Not look good!”
User Experience (UX) vs Developer Experience (DX)
Developers want to have choice

- Choice of architectures
- Choice of programming languages
- Choice of databases
- Choice of application services
- Choice of development tools
- Choice of build and deploy workflows

They don’t want to have to worry about the infrastructure.
Developers want tools to be productive

- Every function they need at hand
- Stable
- Easy to use
- Performant
- No disruptive
- Intuitive

They don’t want to change the way they work.
Users of your technology are happier, promote it more, and stay longer when the product has good DX.
“If you are using a product that combines enterprise-grade functionality with an unusable experience, then your life will be filled with frustration and pain”
Developers Are People Too!
How can we improve?
VMs? Containers?
All I want to do is program!
Jeesh.
Focus on what’s important!!!
Focus on code
INTRODUCING: OpenShift-Do (aka. odo)

A developer friendly command line tool
odo is a CLI tool that provides developers with:

- **Simple** language to understand
- **Fast** and automated source code deployments
- **In-context** work
- **Easy** iterative development cycle
Who odo is for?

- Developers not familiar with OpenShift/Kubernetes
- Developers that don’t want to deal with yaml/json
- It does NOT replace oc or kubectl, though minimizes the need to use them
Where?

https://github.com/redhat-developer/odo
Designing a CLI is easy. Effective CLI design is difficult.
Contributions are welcome.
Interactive Learning Scenarios provide you with a pre-configured OpenShift instance, accessible from your browser without any downloads or configuration.
odo basics?
In **odo**, applications are the basics

```
$ odo application create odo-demo
```
In **odo**, components are the centerpiece

```bash
$ odo create wildfly
```

- From local (Iterative development)
- From binary (Iterative development)
- From git (cloud based development)
In **odo**, everything is in context

```bash
$ odo storage create
  --path=/opt/my-app/data
  --size=1Gi
```
In **odo**, everything is in context

```
$ odo catalog list
$ odo service create database
```
In **odo**, everything is in context

```
$ odo create httpd frontend
```
In odo, everything is in context

$ odo link backend
In odo, everything is in context

```bash
$ odo url create
```
Iterative development

- **odo push**
  - Push sources that will be built in the container
  - Push binary directly to the container

- **odo watch**
  - Watches for local changes (on sources or binary) and pushes automatically