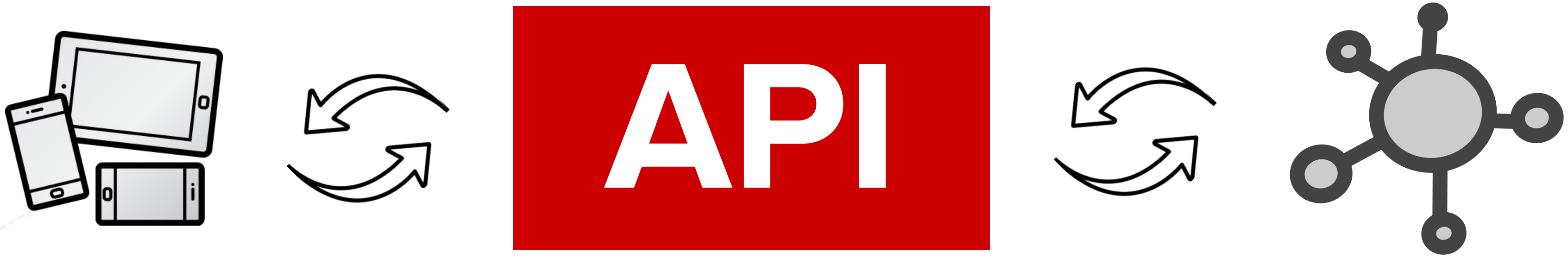
RED HAT DEVELOPERS

The Microservices Journey: Part I

- Skinny on Fat, Thin, Hollow, and Uber -

Daniel Oh
Specialist Solution Architect
Agile & DevOps CoP Manager
@danieloh30

THE NEW DIGITAL ARCHITECTURE



event-driven

asynchronous

antifragile

reactive

serverless

polyglot

scalable

velocity

agile

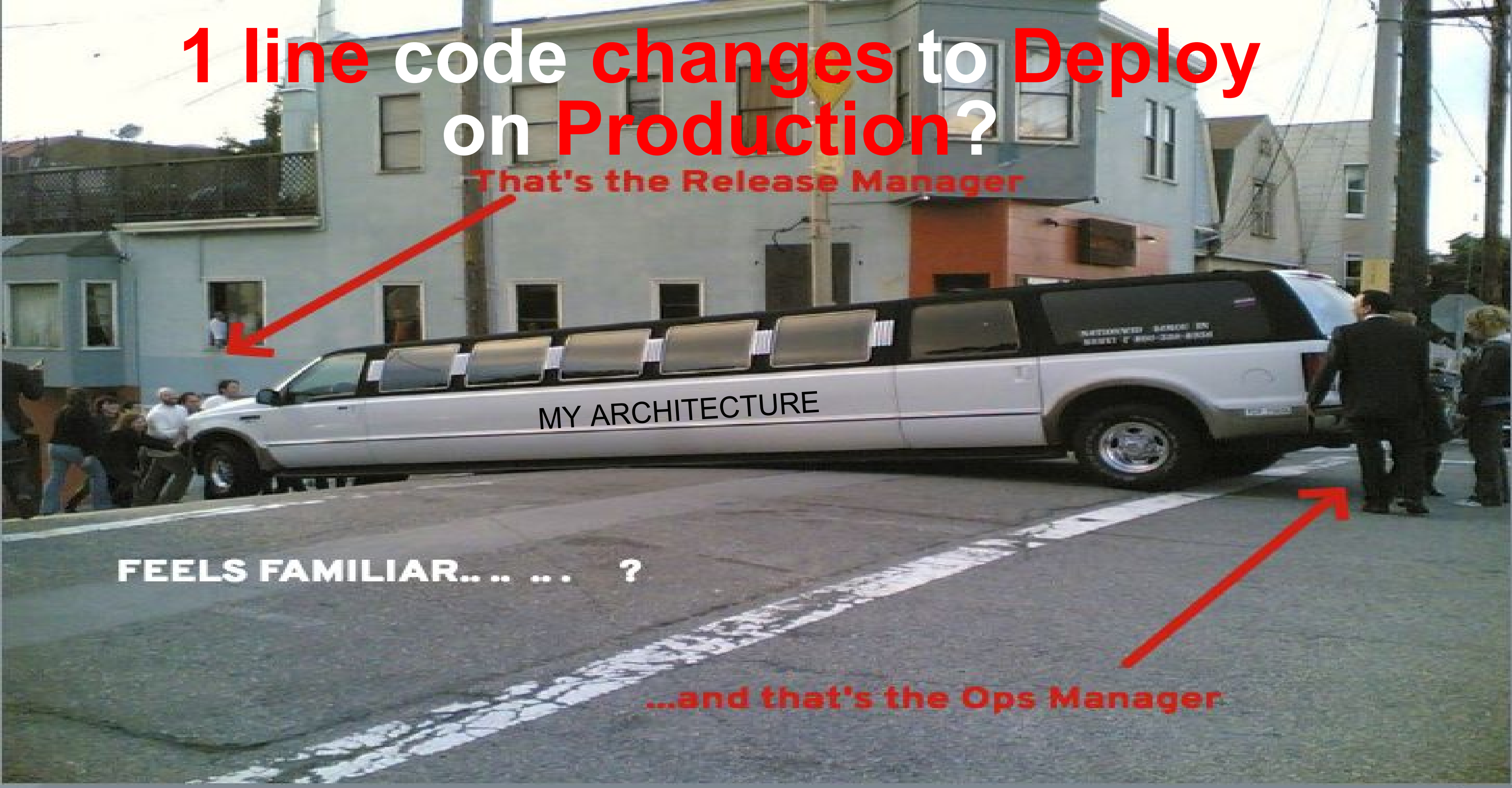
micro-services

Let’s talk about your Reality on Enterprise

Pet vs Cattle

Snowflake vs Phoenix

1 line code changes to Deploy on Production?


That's the Release Manager
MY ARCHITECTURE
FEELS FAMILIAR.. .. ... ?
...and that's the Ops Manager

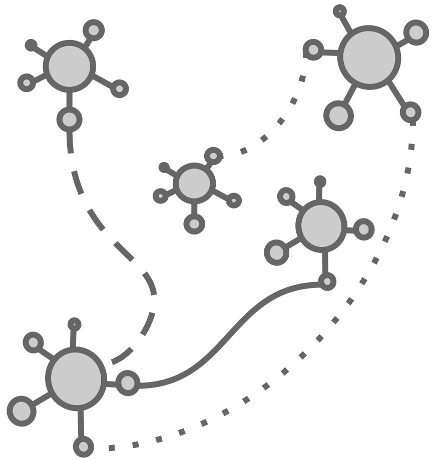
What’s Solution?

Trends : Microservices

“... is an approach to developing a single application as a suite of small services, each running in its **own process** and **communicating with lightweight** mechanisms, often an HTTP resource API. These services are built around business capabilities and **independently deployable** by **fully automated** deployment machinery. There is a bare minimum of **centralized management** of these services, which may be written in **different programming languages** and use different data storage technologies.”

- *Martin Fowler*

  http://martinfowler.com/articles/microservices.html



Why are organization adopting microservices?

Faster deployments

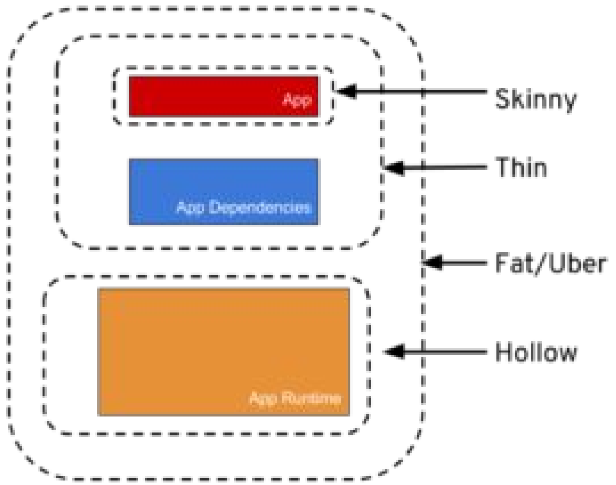
Less complex code

Quicker development

Easier to scale

How to deploy faster?

- Skinny
- Thin
- Hollow
- Fat/Uber



Fat/Uber JAR

- Maven and Spring Boot popularized approach to packaging
- Standard Java Runtime environment
- The amount of extra runtime stuff with framework and runtime features

Thin WAR

- For over a decade with Java EE developers
- Java EE web application with only web content, business logic, and 3rd-party dependencies
- Not anything provided by the Java EE runtime, hence it’s “thin”
- Can’t run “on its own”
- Must be deployed to a Java EE app server or Servlet container

Thin JAR

- Same as a Thin WAR, except using the JAR packaging format
- Typically used by specialized applications/plugin architectures
- The .kjar format from Drools

Skinny WAR

- Thinner than a Thin WAR
- Not include any of the 3rd-party libraries
- ONLY contains the (byte) code
- **Pros on layered container images for DevOps sanity**

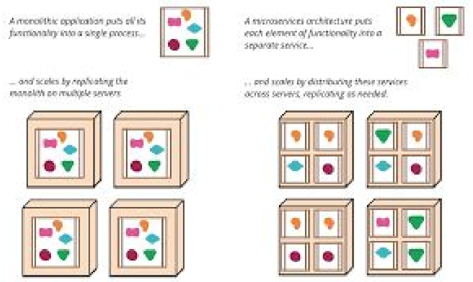
Skinny JAR

- Same as a Skinny WAR, except using JAR packaging and frameworks such as WildFly Swarm
- CI/CD sanity (and your AWS bill) – just ask Hubspot
- Take a Thin WAR and remove all the 3rd-party dependencies
- The smallest atomic unit of app
- Must be deployed to a runtime, the other needed bits(e.g. Hollow JAR)

Hollow JAR

- Java application runtime like “just enough” app server
- Not contain any applications itself
- WildFly Swarm allow you to customize how much is “just enough”
- Eclipse MicroProfile(e.g. Paraya Micro, TomEE) provide pre-built distributions of popular combinations of runtime components

Why bother?


A monolithic application puts all its functionality into a single process...
... and scales by replicating the monolith on multiple servers
A microservices architecture puts each element of functionality into a separate service...
... and scales by distributing these services across servers, replicating as needed.

- Deploying to Dev, Test, and Prod lots of times a day **even** [**2 billion times a week**]
- Minimizing the app sizing for overall devops efficiency, operational sanity
- Don't have to minimize the lines of code
- Must reduce the number of times your app, dependencies across network, disk, etc.
- Finally, breaking your app into different packaged parts with properly separated, treated even versioned like **Container Image Layers**

Just tell me which one to use!

DEMO

- WildFly Swarm -

WildFly Swarm

"… is a mechanism for packaging Java applications that contain **just enough** functionality to run the app. It has an abstraction called a **Fraction**, each of which embodies some functionality that apps need. You can select which Fractions you need, and **package only those fractions** along with your app to produce a **minimized and specialized runnable image** for your app. WildFly Swarm has the ability to create many of the above types of packaged apps."

Grab the code

```
$ git clone
https://github.com/jamesfalkner/wfswarm-packaging-demo
```

Fat/Uber JARs

```
$ cd fat-thin; mvn clean package
$ du -hs target/*.jar
45M target/weight-1.0-swarm.jar

$ curl http://localhost:8080/api/hello
```

Thin WARs

```
$ du -hs target/*.war
512K target/weight-1.0.war
```

Skinny WARs

```
% unzip -l target/*.war
Archive: target/weight-1.0.war
Length Date Time Name
-------- ---- ---- ----
0 08-26-17 01:14 META-INF/
132 08-26-17 01:14 META-INF/MANIFEST.MF
0 08-26-17 01:14 WEB-INF/
0 08-26-17 01:14 WEB-INF/classes/
0 08-26-17 01:14 WEB-INF/classes/com/
0 08-26-17 01:14 WEB-INF/classes/com/test/
0 08-26-17 01:14 WEB-INF/classes/com/test/rest/
0 08-26-17 01:14 WEB-INF/lib/
746 08-26-17 01:14 WEB-INF/classes/com/test/rest/HelloEndpoint.class
402 08-26-17 01:14 WEB-INF/classes/com/test/rest/RestApplication.class
634048 08-26-17 01:14 WEB-INF/lib/joda-time-2.9.9.jar   <------ This one
0 08-26-17 01:14 META-INF/maven/
0 08-26-17 01:14 META-INF/maven/com.test/
0 08-26-17 01:14 META-INF/maven/com.test/weight/
2829 08-26-17 01:14 META-INF/maven/com.test/weight/pom.xml
97 08-26-17 01:14 META-INF/maven/com.test/weight/pom.properties
-------- -------
638381 16 files
```

Skinny WAR: Removing direct dependencies

```
$ cd joda-fraction; mvn clean package install

[INFO] --- maven-install-plugin:2.4:install (default-install) @ joda ---
[INFO] Installing /Users/daniel/ws/fat/joda-fraction/target/joda-1.2.jar to
/Users/daniel/.m2/repository/com/jhf/joda/1.2/joda-1.2.jar
[INFO] Installing /Users/daniel/ws/fat/joda-fraction/pom.xml to
/Users/daniel/.m2/repository/com/jhf/joda/1.2/joda-1.2.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

Skinny WAR: Removing direct dependencies

```
$ cd ../skinny; mvn clean package
$ ls -l target/*.war
-rw-r--r-- 1 daniel daniel  2240 Aug 3 01:45 target/weight-1.0.war

$ unzip -l target/*.war
Archive: target/weight-1.0.war
Length Date Time Name
-------- ---- ---- ----
99 08-26-17 01:45 META-INF/MANIFEST.MF
0 08-26-17 01:45 META-INF/
0 08-26-17 01:45 WEB-INF/
0 08-26-17 01:45 WEB-INF/classes/
0 08-26-17 01:45 WEB-INF/classes/com/
0 08-26-17 01:45 WEB-INF/classes/com/test/
0 08-26-17 01:45 WEB-INF/classes/com/test/rest/
0 08-26-17 01:45 WEB-INF/lib/
402 08-26-17 01:45 WEB-INF/classes/com/test/rest/RestApplication.class
746 08-26-17 01:45 WEB-INF/classes/com/test/rest/HelloEndpoint.class
-------- -------
1247 10 files
```

Hollow JARs

```
$ mvn clean package -Dswarm.hollow=true
$ du -hs target/*.jar target/*.war
44M target/weight-1.0-hollow-swarm.jar
4.0K target/weight-1.0.war ⇒ Block size 4K

#Run Skinny app on Hollow server
$ java -jar target/weight-1.0-hollow-swarm.jar
target/weight-1.0.war

$ curl http://localhost:8080/api/hello
hello, the date is 2017-08-26
```

What about Spring Boot?

```
$ cd spring-boot-fat; mvn clean package
$ du -hs target/*.jar
14M target/greeting-spring-boot.jar
```

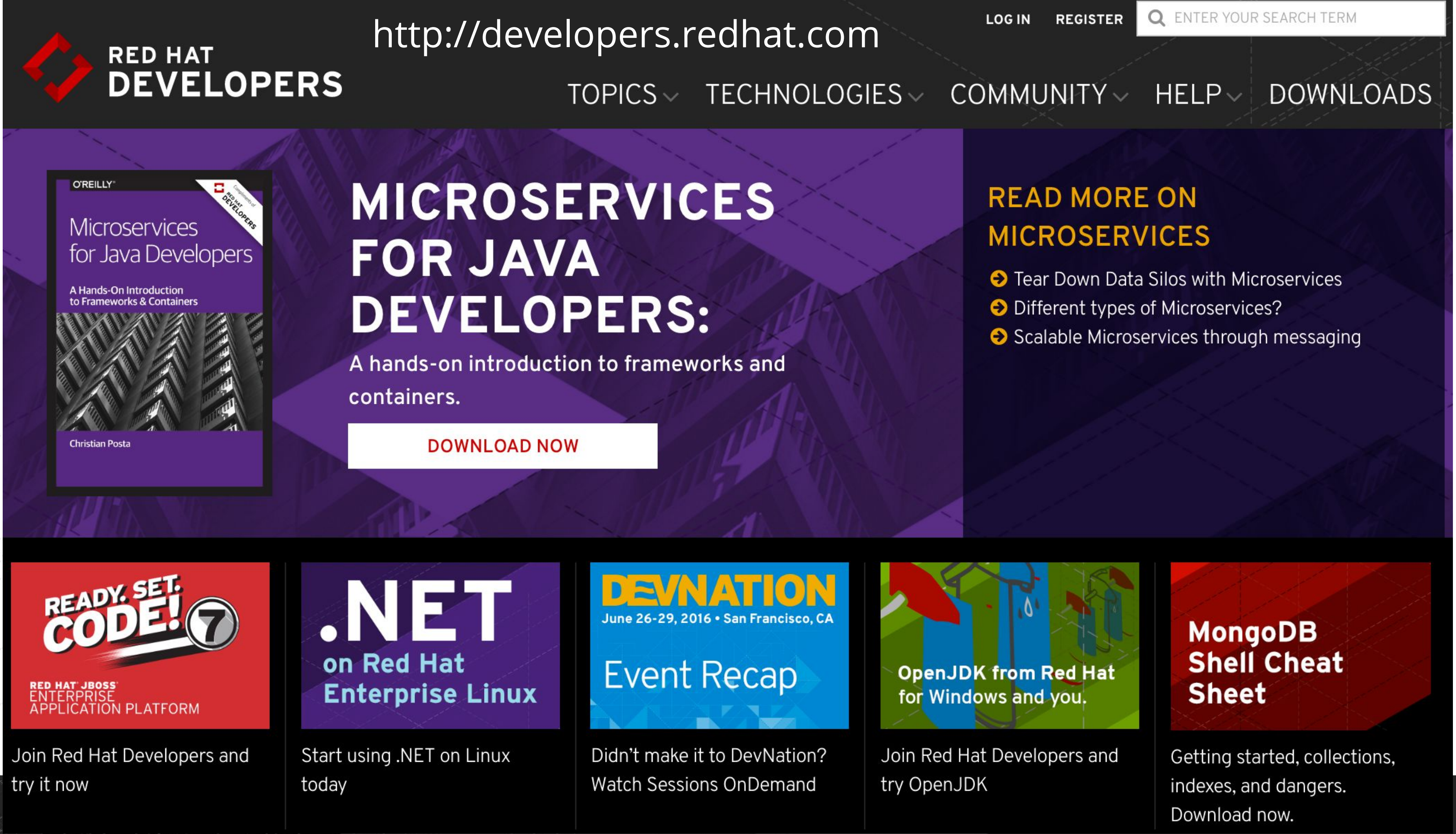
What if Spring code changes to WildFly Swarm's hollow JAR / skinny WAR duo?

https://en.wikipedia.org/wiki/Fourth_wall

SUMMARY

- Shrank app from 45M ➔ 512K ➔ 2243 bytes via WildFly Swarm
- Separated app from runtime dependencies,
- And put them in separate Linux container image layers
- Make your CI/CD pipelines faster
- Make your developers faster at edit, build, and test
- Provide assurance that you are testing with the same bits that will land in production.

http://developers.redhat.com



RED HAT DEVELOPERS
LOG IN
REGISTER
ENTER YOUR SEARCH TERM
TOPICS
TECHNOLOGIES
COMMUNITY
HELP
DOWNLOADS
O'REILLY
Microservices for Java Developers
A Hands-On Introduction to Frameworks & Containers
Christian Posta
DOWNLOAD NOW
READY. SET. CODE! 7
RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM
.NET on Red Hat Enterprise Linux
DEVNATION
June 26-29, 2016 • San Francisco, CA
Event Recap
OpenJDK from Red Hat for Windows and you.
MongoDB Shell Cheat Sheet

MICROSERVICES FOR JAVA DEVELOPERS:

A hands-on introduction to frameworks and containers.

READ MORE ON MICROSERVICES

- Tear Down Data Silos with Microservices
- Different types of Microservices?
- Scalable Microservices through messaging

Join Red Hat Developers and try it now

Start using .NET on Linux today

Didn't make it to DevNation? Watch Sessions OnDemand

Join Red Hat Developers and try OpenJDK

Getting started, collections, indexes, and dangers. Download now.

@danieloh30