RED HAT SUMMIT

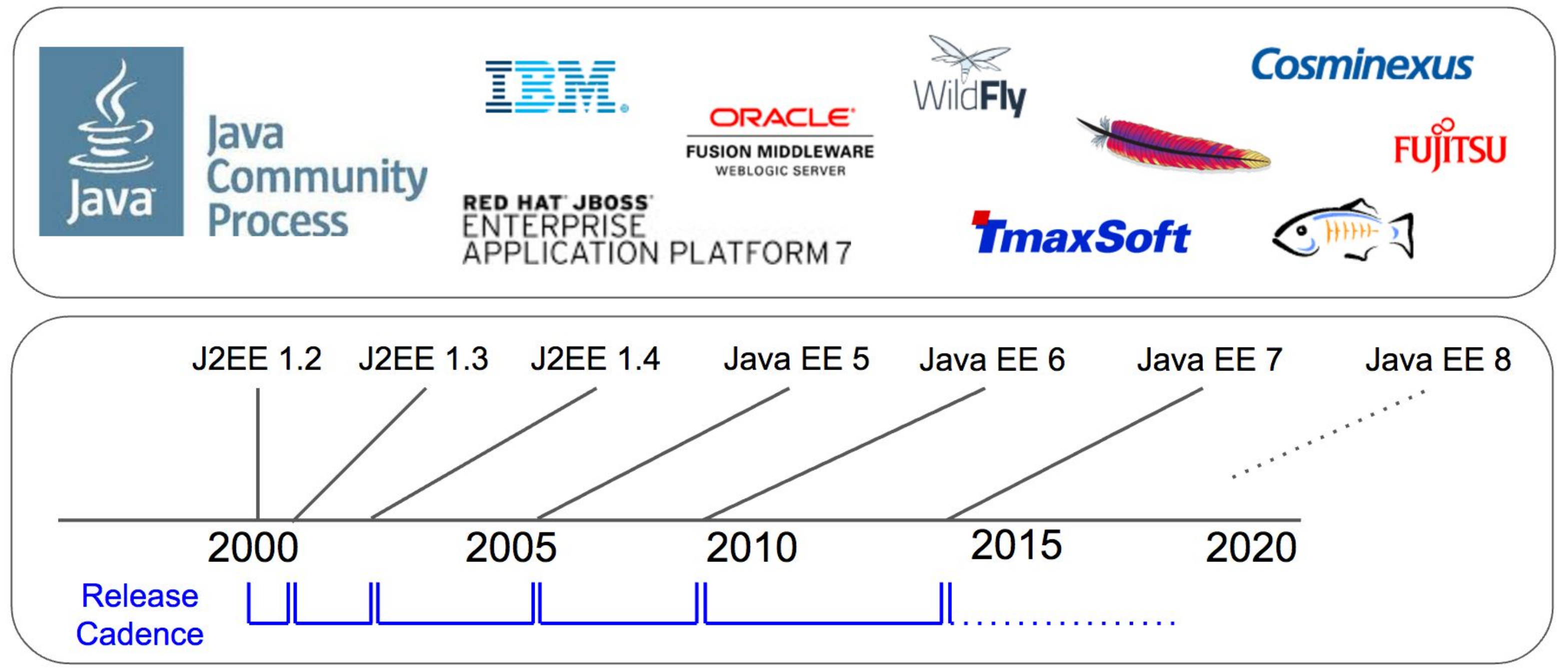
Developing Microservices with WildFly Swarm and MicroProfile

Ken Finnigan
Principal Software Engineer

John Clingan
Senior Principal Product Manager

Eclipse MicroProfile

Enterprise Java Standards History



Eclipse MicroProfile Background

- Many innovative “microservices” efforts in existing Java EE projects
        - WildFly Swarm
        - WebSphere Liberty
        - Payara
        - TomEE
    - Projects already leveraging both Java EE and non-Java EE technologies
    - Creating new features/capabilities to address microservices architectures

- Wanted to avoid splitting into separate communities

- *So we are collaborate in one community!*

An Eclipse Foundation Project

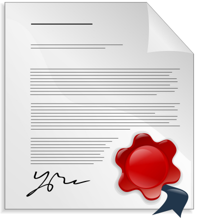
- Meritocracy; vendor neutrality
- MicroProfile leadership can change over time
- Legal and technical infrastructure
- Trademark Ownership
- Accepts Apache License

The Path to Microservices with MicroProfile

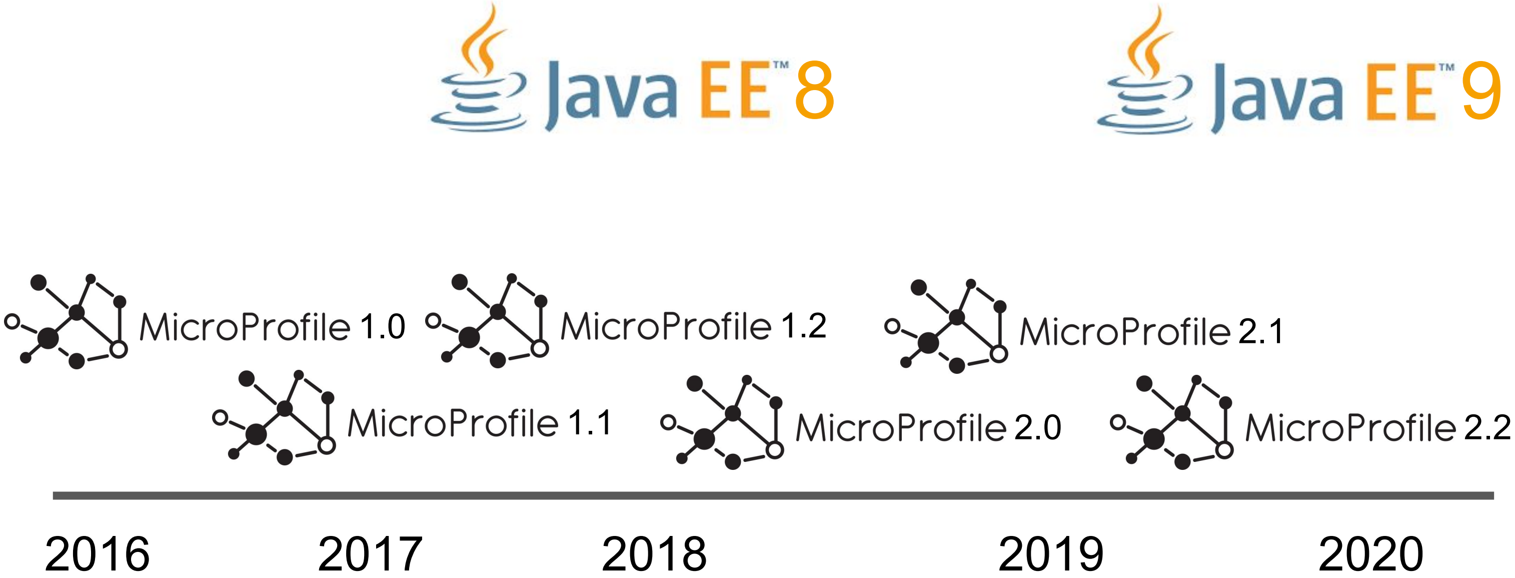
1. Leverage relevant Java EE Technologies

2. Organic Innovation

3. Collaborate in Open Source



Quickly Put Features in Developers Hands



Java EE 8

Java EE 9

MicroProfile 1.0

MicroProfile 1.1

MicroProfile 1.2

MicroProfile 2.0

MicroProfile 2.1

MicroProfile 2.2

2016

2017

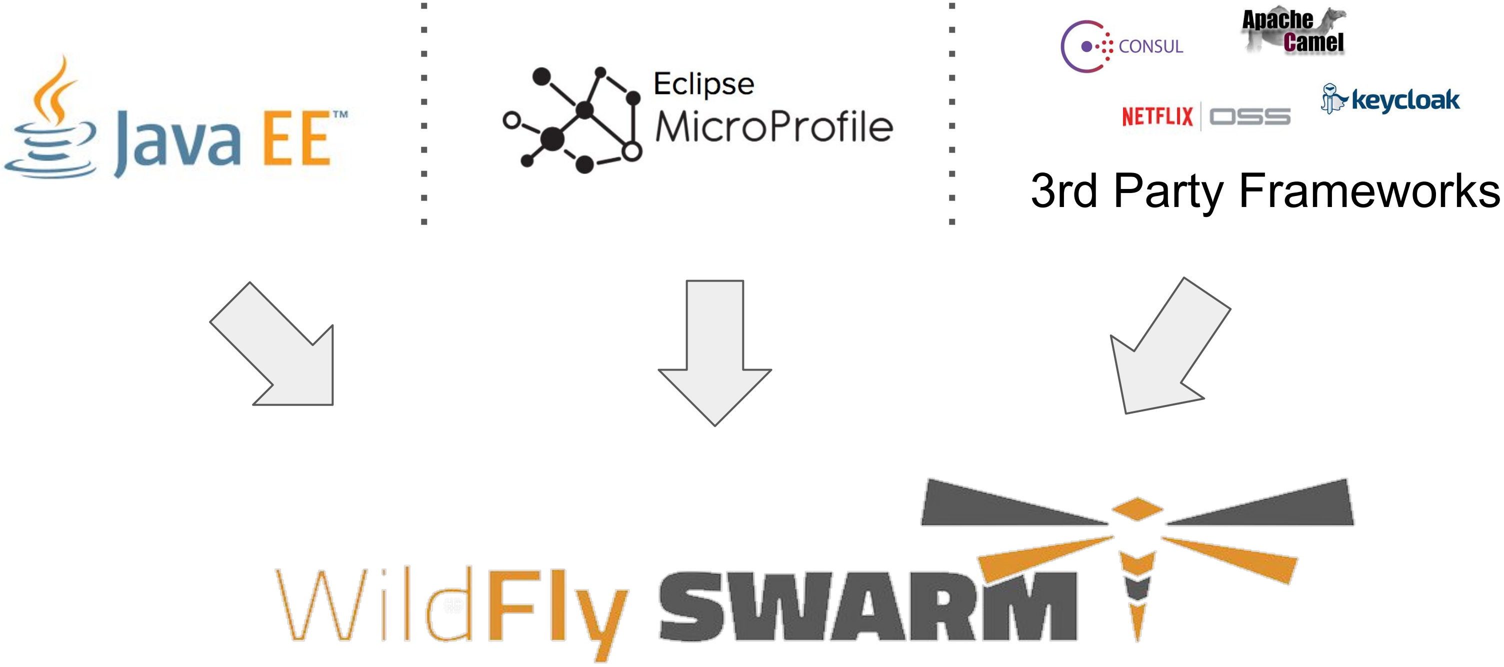
2018

2019

2020

* 2-4 releases per year

Practical Usage of MicroProfile - Example



3rd Party Frameworks

Join Us Later Today

Optimizing Java EE for a microservices architecture

4:30 PM - 5:15 PM

Room 102B

WildFly Swarm

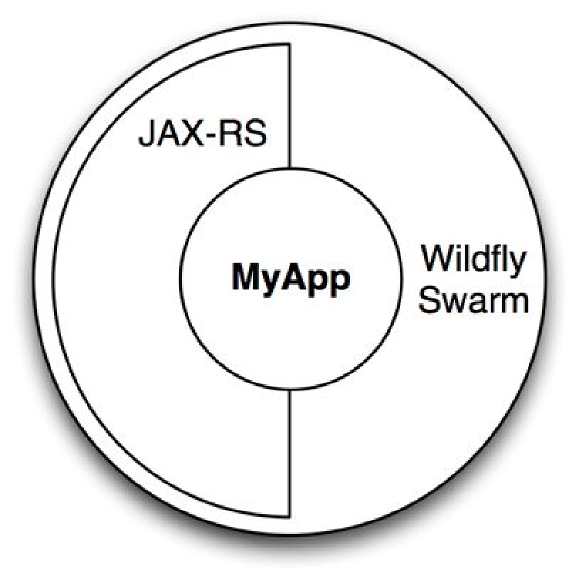
What is it?

- Begins with WildFly / Java EE
- Extends Java EE with cloud native capabilities
- Package deployment
    - Uber Jar - including deployment
    - Hollow Uber Jar - without deployment

WildFly SWARM

Just enough App Server (JeAS)

- Bundles your deployment (JAR/WAR)
- Fractions define available functionality for deployment
- Internal Maven repository with dependencies
- Bootstrap Code



Fractions...

- Enable WildFly subsystems (e.g. Infinispan)
- Integrate additional frameworks/services (e.g. Topology)
- Provide deployments (e.g. Swagger, Jolokia)
- Add API dependencies (e.g. JAX-RS)
- Alter deployments (e.g. Keycloak)

Fractions...

- Expressed as Maven (GAV) coordinates:
    - org.wildfly.swarm:<fraction>:<version>, e.g
    - org.wildfly.swarm:undertow:2017.5.0
- 184 fractions currently available
    - 158 stable
    - 26 experimental
- About 80% wrap WildFly related components (Java EE, WF Camel, internal WF)
- More in the pipeline

Enabling WildFly Swarm

```
<plugin>
  <groupId>org.wildfly.swarm</groupId>
  <artifactId>wildfly-swarm-plugin</artifactId>
  <version>${version.wildfly.swarm}</version>
  <executions>
    <execution>
      <id>package</id>
      <goals>
        <goal>package</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

AUTO DETECTION DEMO

Enabling specific WildFly Swarm Fractions

```
<dependency>
  <groupId>org.wildfly.swarm</groupId>
  <artifactId>bom</artifactId>
  <version>${version.wildfly.swarm}</version>
  <type>pom</type>
  <scope>import</import>
</dependency>
```

Enabling specific WildFly Swarm Fractions

```
<dependency>
  <groupId>org.wildfly.swarm</groupId>
  <artifactId>jaxrs</artifactId>
</dependency>
```

Build / Run

Build

```
mvn package
```

Run

```
mvn wildly-swarm:run
```

OR

```
java -jar <my-app>-swarm.jar
```

OR

```
IDE > Run … org.wildfly.swarm.Swarm()
```

Flexible Configuration

- Maven Plugin

```
… <swarm.http.port>8081</swarm.http.port> …
```

- Java properties

```
java -Dswarm.http.port=8081 myapp.jar
```

- JBoss EAP configuration

```
java myapp.jar -c standalone.xml
```

- Environment-specific configuration

```
java myapp.jar -s project-production.yml
```

MICROPROFILE DEMO

Proposed Eclipse MicroProfile Features

- WildFly Swarm status:
  - **Configuration** - Implementation being worked on at present
  - **Health Check** - Align with `monitor` fraction when released
  - **Security** - May just require validation with `keycloak` fraction

Health Checks

- `monitor` fraction was basis for MicroProfile Health Check specification
- Ability to add `/health` endpoints to microservice
- Integration with OpenShift Readiness/Liveness Checks
  - via fabric8-maven-plugin

Health Checks

```
<dependency>
  <groupId>org.wildfly.swarm</groupId>
  <artifactId>monitor</artifactId>
</dependency>
```

```
<enricher>
  <includes>
    <include>wildfly-swarm-health-check</include>
  </includes>
</enricher>
```

HEALTHCHECK DEMO

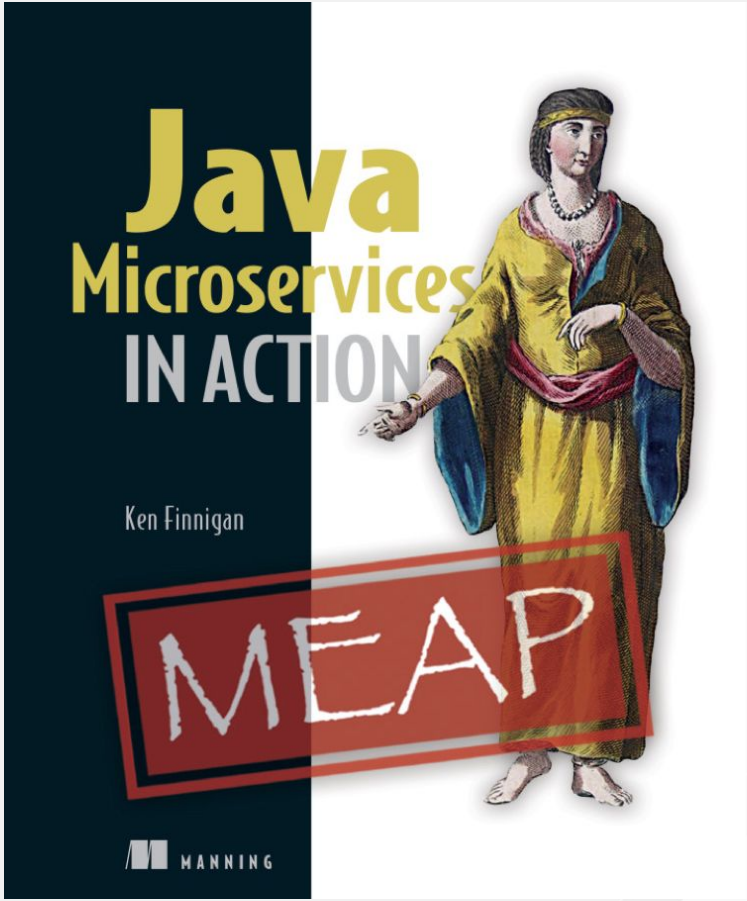
WildFly Swarm Community

- IRC - #wildfly-swarm on Freenode
- Google Group - https://groups.google.com/forum/#!forum/wildfly-swarm
- JIRA - https://issues.jboss.org/browse/SWARM
- Twitter - @wildflyswarm

Java Microservices Book

- Recently released into MEAP
- Uses WildFly Swarm
- 39% discount on all Manning books with code: **ctwrhsummit17**

https://www.manning.com/books/java-microservices-in-action



QUESTIONS ?

Demo Code

Available from:

https://github.com/kenfinnigan/wfswarm-rhsummit2017

RED HAT SUMMIT

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos

RED HAT
SUMMIT

LEARN. NETWORK.
EXPERIENCE
OPEN SOURCE.