WildFly SWARM

MICROSERVICES AND JAVA EE

redhat.

Lance Ball / @lanceball

WildFly SWARM

JBoss®

by Red Hat

project:odd

MICROSERVICES

A software architecture style in which complex applications are composed of small, independent processes communicating with each other using language-agnostic APIs. These services are small, highly decoupled and focus on doing a small task, facilitating a modular approach to system-building.

Wikipedia - Microservices

BUT WAIT

ISN'T THIS JUST SOA?

DUDE, SOA IS NOT THE PREFERRED NOMENCLATURE
MICROSERVICES, PLEASE
made on imgur

KEY DISTINCTIONS

- Deployment
- Scalability
- Configuration

DEPLOYMENT

Single artifact deployment

Independently / continuously deployable

SCALABILITY

Independently scalable

Small, focused teams

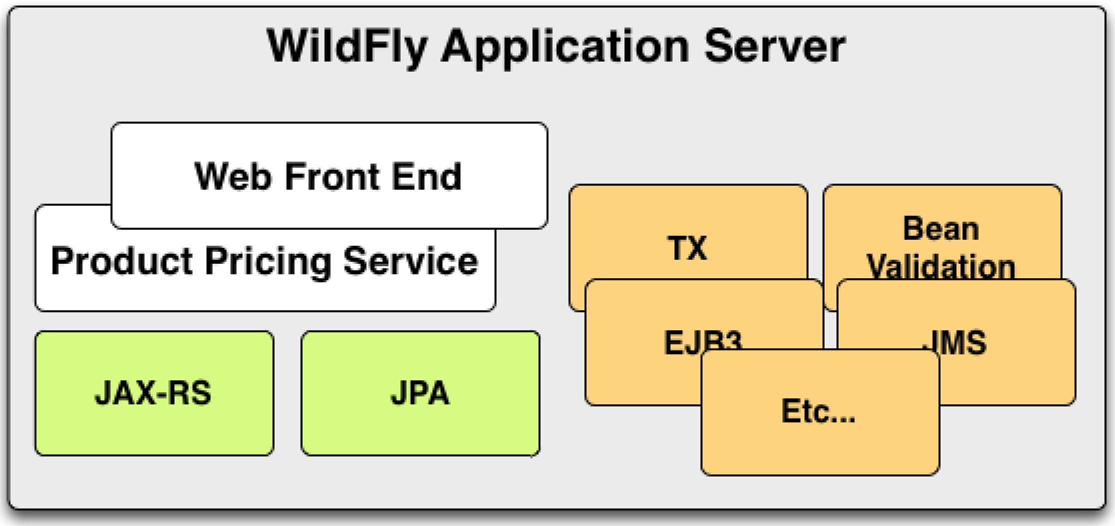
Technology independence

CONFIGURATION

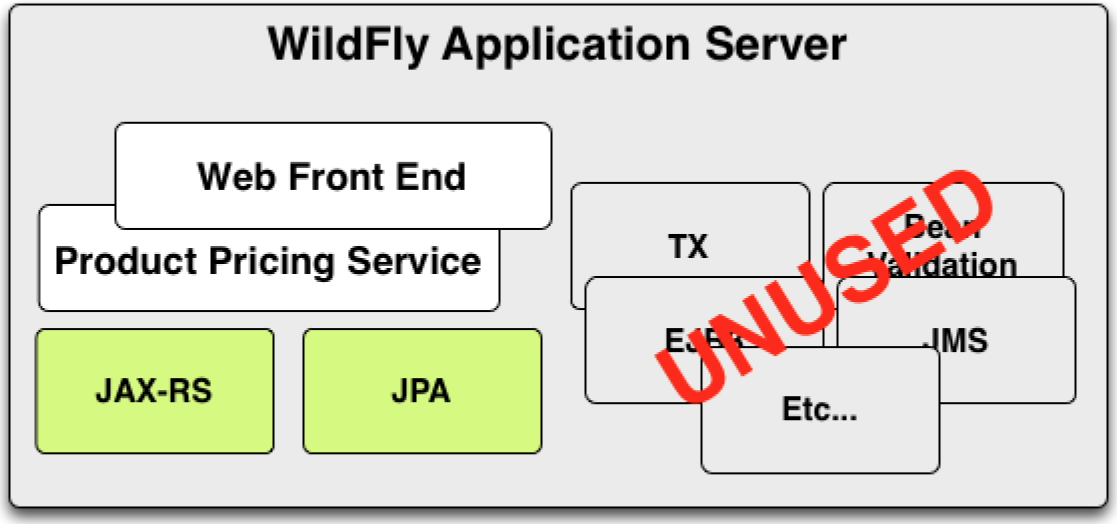
Convention over configuration

JEAS (Just Enough App Server)

TYPICAL WILDFLY DEPLOYMENT



TYPICAL WILDFLY DEPLOYMENT



OK, YOU SOLD ME

BUT… HOW?

TERMINOLOGY

UBERJAR

A single .jar file containing your application, the portions of WildFly required to support it, an internal Maven repository of dependencies, plus a shim to bootstrap it all.

FRACTION

A well-defined collection of application capabilities. May map directly to a WildFly subsystem, or bring in external capabilities such as Netflix Ribbon.

WHAT FRACTIONS CAN DO

Enable WildFly subsystems (JAX-RS, Infinispan)

Provide deployments (ribbon-webapp, jolokia)

Alter deployments (keycloak)

FRACTIONS

WILDFLY SUBSYSTEMS

Datasources	Keycloak	Undertow
EJB	Messaging	Clustering
JAX-RS	JPA	Infinispan
Transactions	CDI	Hawkular

JAX-RS FRACTION

pom.xml

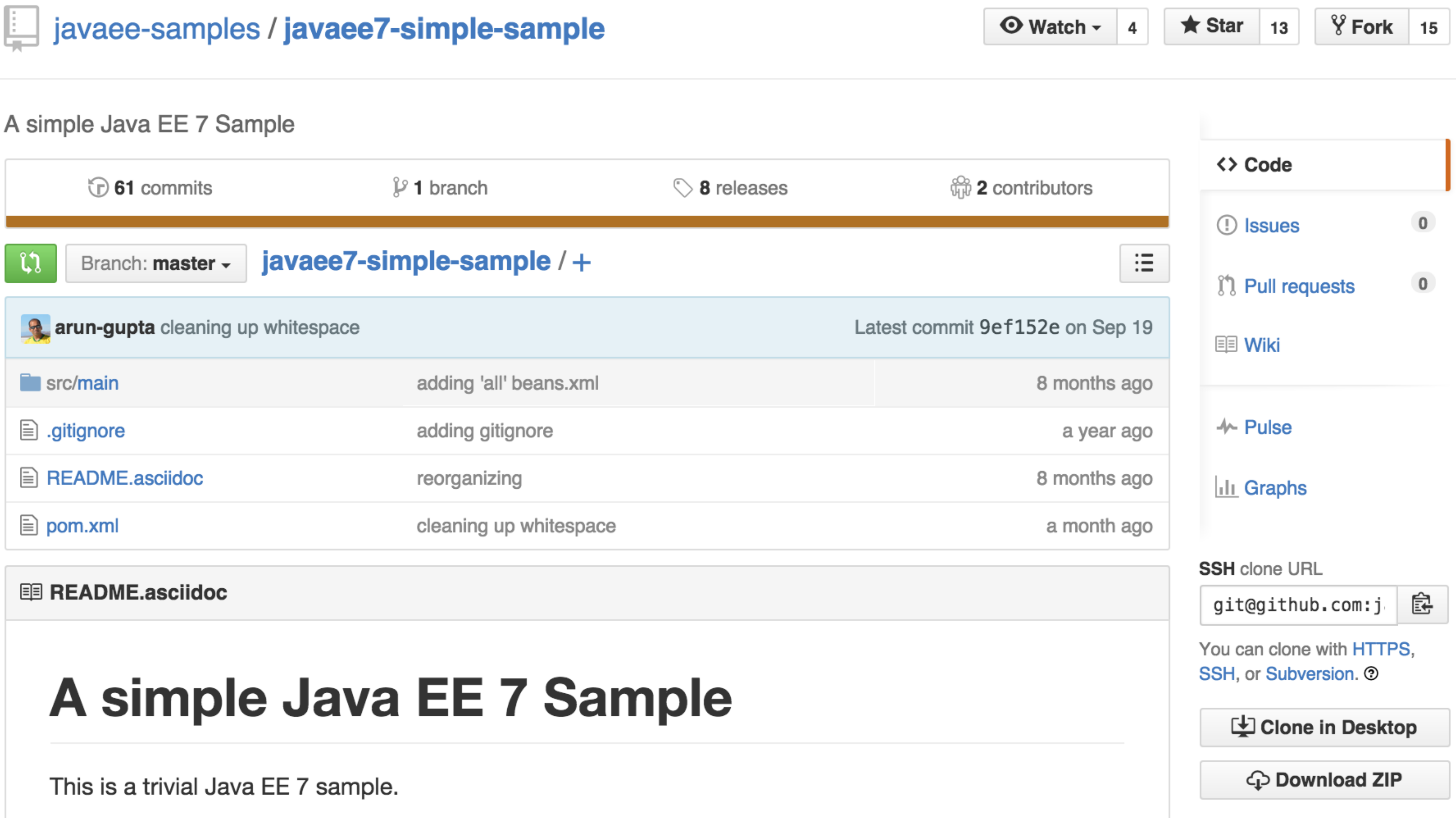
```
<dependency>
  <groupid>org.wildfly.swarm</groupid>
  <artifactid>wildfly-swarm-jaxrs</artifactid>
  <version>${swarm.version}</version>
</dependency>
```

A SWARM APP

pom.xml

```
<plugin>
  <groupid>org.wildfly.swarm</groupid>
  <artifactid>wildfly-swarm-plugin</artifactid>
  <executions>
    <execution>
      <goals>
        <goal>package</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

CONVERT JAVA EE APPLICATION TO USE WILDFLY SWARM



javaee-samples / javaee7-simple-sample

Watch 4 | Star 13 | Fork 15

A simple Java EE 7 Sample

61 commits | 1 branch | 8 releases | 2 contributors

Branch: master | javaee7-simple-sample / +

arun-gupta cleaning up whitespace — Latest commit 9ef152e on Sep 19

src/main	adding 'all' beans.xml	8 months ago
.gitignore	adding gitignore	a year ago
README.asciidoc	reorganizing	8 months ago
pom.xml	cleaning up whitespace	a month ago

README.asciidoc

A simple Java EE 7 Sample

This is a trivial Java EE 7 sample.

pom.xml

```xml
<!-- include the JAX-RS with CDI+JAXB Fractions -->
<dependency>
  <groupid>org.wildfly.swarm</groupid>
  <artifactid>wildfly-swarm-jaxrs-weld</artifactid>
  <version>${version.swarm}</version>
</dependency>

<dependency>
  <groupid>org.wildfly.swarm</groupid>
  <artifactid>wildfly-swarm-jaxrs-jaxb</artifactid>
  <version>${version.swarm}</version>
</dependency>

<!-- Make it a Swarm App -->
<plugin>
  <groupid>org.wildfly.swarm</groupid>
  <artifactid>wildfly-swarm-plugin</artifactid>
  <version>${version.swarm}</version>
  <executions>
    <execution>
      <goals>
        <goal>package</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

THAT'S ALL

BUILDING A SWARM APP

```
$ mvn package
```

PRODUCES

```
target/myApp-swarm.jar
```

RUNNING A SWARM APP

```
$ java -jar myApp-swarm.jar
```

OR

```
$ mvn wildfly-swarm:run
```

FRACTIONS

NOT JUST JAVA EE

KEYCLOAK

WildFly Overlay

SSO, OAuth, OpenID, JWT, SAML, etc.

KEYCLOAK AUTHENTICATION

PricingResource.java

```
@Path("/")
public class PricingResource {

  @GET
  @Path("/book/{id}")
  @Produces("application/json")
  public Integer search(@PathParam("id") String id, @Context Security
    KeycloakPrincipal principal = (KeycloakPrincipal) context.getUser
    if ( principal != null && principal.getKeycloakSecurityContext()
      return 9;
    }
    return 10;
  }
}
```

NETFLIX OSS

Service Discovery

Client Side Load Balancing

NETFLIX OSS

Main.java

```java
public class Main {

  public static void main(String... args) throws Exception {
    // Create a simple shrinkwrapped JAX-RS app
    Container container = new Container();
    JAXRSArchive deployment = ShrinkWrap.create(JAXRSArchive.class);
    deployment.addPackage(Main.class.getPackage());

    // Make it discoverable via Ribbon
    deployment.as(RibbonArchive.class).setApplicationName("pricing");
    deployment.as(Secured.class);

    container.start();
    container.deploy(deployment);
  }
}
```

CONVENTION OVER CONFIGURATION

Reasonable defaults out of the box

Easily customized with a fluent API

CUSTOM CONFIGURATION

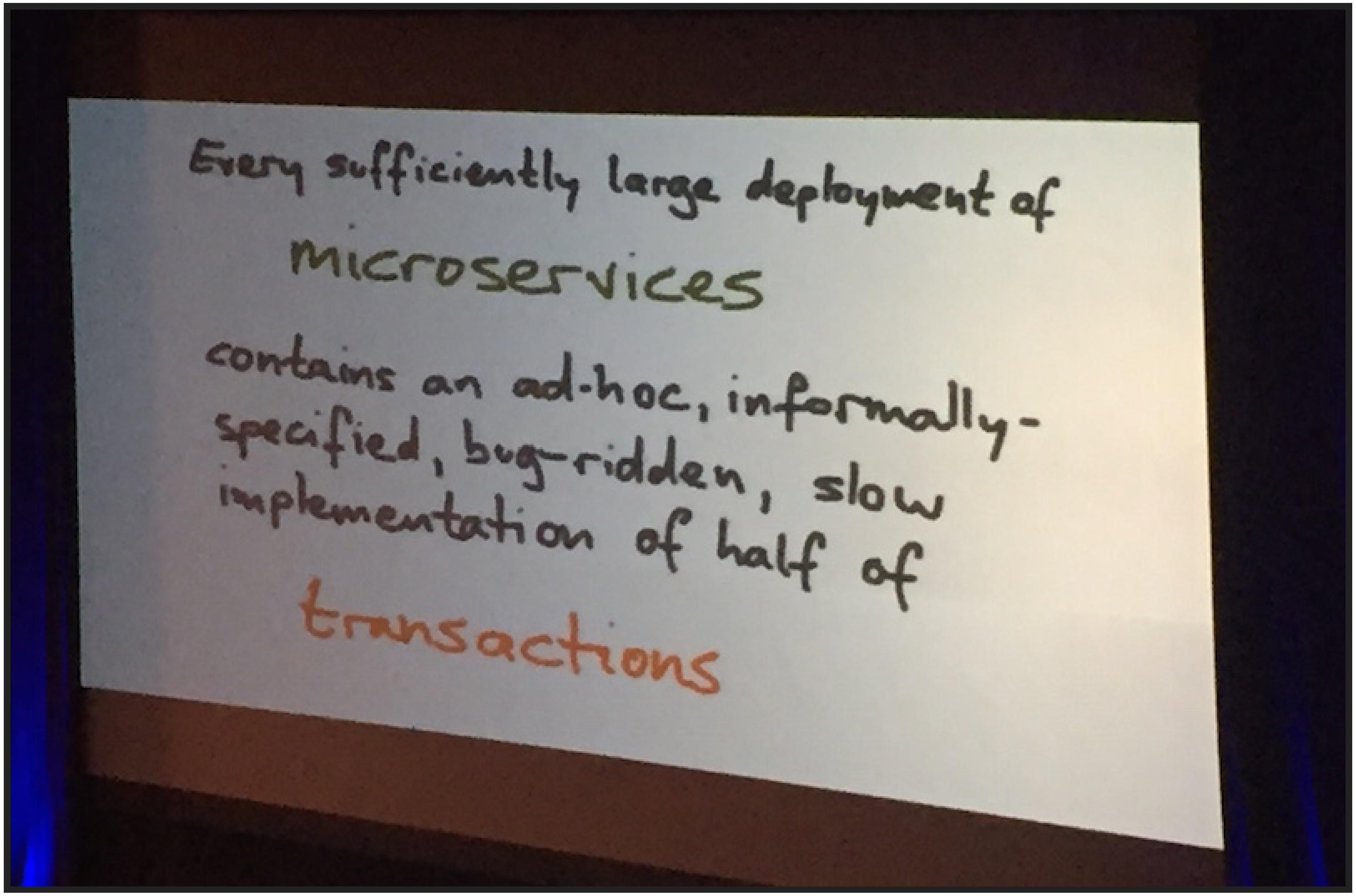
```java
public class Main {

  public static void main(String...args) {

    CacheContainer webCache = new CacheContainer("web")
        .defaultCache("dist")
        .jgroupsTransport(new JGroupsTransport().lockTimeout(60000L))
        .distributedCache("dist", distCache -> distCache
            .mode("ASYNC")
            .l1Lifespan(0L)
            .owners(2)
            .lockingComponent(new LockingComponent().isolation("REPEATABLE_READ")
            .transactionComponent(new TransactionComponent().mode("BATCH"))
            .fileStore(new FileStore()));

    InfinispanFraction fraction = new InfinispanFraction();
    fraction.cacheContainer( webCache );

    Container container = new Container();
    container.fraction( fraction );

    // Start the container
    container.start();
  }
}
```

POTENTIAL HURDLES

Complexity inherent in a distributed system

Potential operational complexity

Tooling



Every sufficiently large deployment of

microservices

contains an ad-hoc, informally-specified, bug-ridden, slow implementation of half of

transactions

TRANSACTIONS

```
public class Main {
  public static void main(String[] args) throws Exception {
    Container container = new Container();

    container.subsystem(new TransactionsFraction(4712, 4713));
    container.start();

    JAXRSArchive deployment = ShrinkWrap.create(JAXRSArchive.class);

    deployment.addResource(MyResource.class);

    container.deploy(deployment);
  }
}
```

TRANSACTIONS

```java
@Path("/")
public class MyResource {

  @Path("begincommit")
  @GET
  @Produces("text/plain")
  public String beginCommit() throws Exception {

    UserTransaction txn = (UserTransaction) new InitialContext()
        .lookup("java:comp/UserTransaction");
    String value = "Transaction ";

    try {
      txn.begin();

      value += "begun ok";

      try {
        txn.commit();

        value += " and committed ok";
      } catch (final Throwable ex) {
        value += " but failed to commit";
    }
  }
}
```

THE FUTURE

Formal release

More WildFly subsystems

Improved integration tests

Community feedback

COMMUNITY

GitHub

https://github.com/wildfly-swarm

Docs

https://wildfly-swarm.gitbooks.io/wildfly-swarm-users-guide/content/

Twitter

@wildflyswarm

Freenode

#wildfly-swarm

THANKS & QUESTIONS

http://lanceball.com/swarm-preso