WildFly 10 / WildFly Swarm

talks4nerds 03/2016

Harald Pehl
http://hpehl.info - @haraldpehl

About Me

Senior Software Engineer - Red Hat

WildFly Team - Management API / Management Console

http://hpehl.info - @haraldpehl

WildFly

WildFly

Previously called JBoss AS 7

Upstream for JBoss EAP

Fast, lightweight, manageable

Developer friendly

Open Source

Supports Java EE specs

Highlights

Java 8 / Java EE7 support

Ports–

Enhanced Management Console

Undertow (JS) - HTTP/2

Role Based Access Control

Offline CLI Support

Highlights

Updated APIs & Subsystems

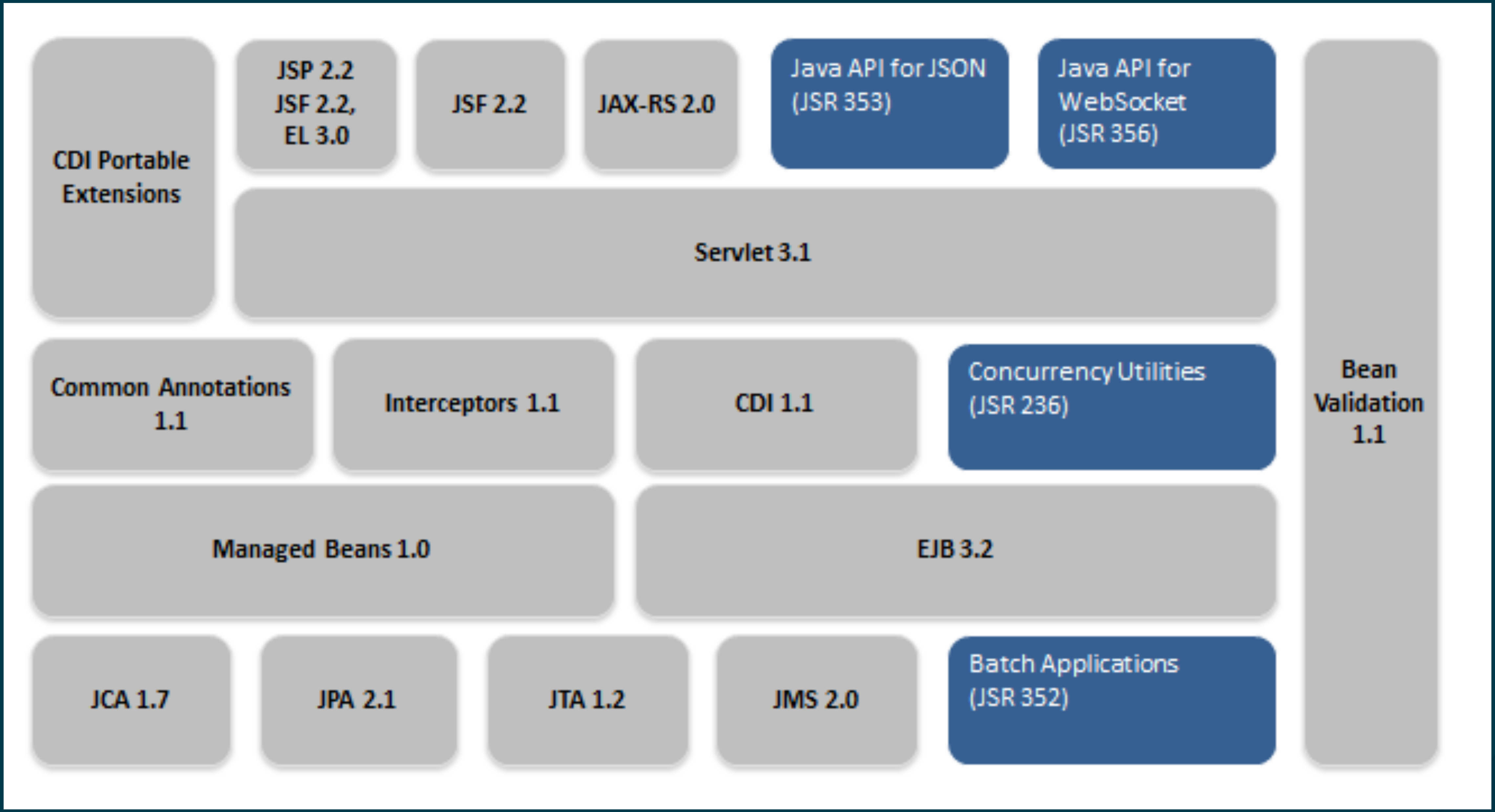
SLSB and MDB Automatic Pool Sizing

Patching

Server Suspend Mode & Graceful Shutdown

HA Singleton Deployments

Java EE7


CDI Portable Extensions
JSP 2.2 JSF 2.2, EL 3.0
JSF 2.2
JAX-RS 2.0
Java API for JSON (JSR 353)
Java API for WebSocket (JSR 356)
Servlet 3.1
Common Annotations 1.1
Interceptors 1.1
CDI 1.1
Concurrency Utilities (JSR 236)
Bean Validation 1.1
Managed Beans 1.0
EJB 3.2
JCA 1.7
JPA 2.1
JTA 1.2
JMS 2.0
Batch Applications (JSR 352)

Undertow

Flexible and high-performance

Blocking / non-blocking based in NIO

Composition / handler based architecture

Lightweight & fully embeddable

Servlet 3.1 & HTTP upgrade

Non-Blocking Handler

```
public class HelloWorldServer {

  public static void main(final String[] args) {
    Undertow server = Undertow.builder()
        .addHttpListener(8080, “localhost")
        .setHandler(new io.undertow.server.HttpHandler() {
            @Override
            public void handleRequest(final HttpServerExchange ex)
            throws Exception {
              ex.getResponseHeaders().put(Headers.CONTENT_TYPE, “text/plain");
              ex.getResponseSender().send("Hello World”);
            }
        }).build();
    server.start();
  }
}
```

Undertow JS

JavaScript on the server side

Based on Nashorn

Supports hot deployment

Injection support

Undertow JS sample

```
$undertow
    .alias("db", "jndi:java:jboss/datasources/ExampleDS")
    .onGet("/customers/{id}",['db', function ($exchange, db) {
        var customer = db.selectOne("select * from customer where id=?”,
            $exchange.param(‘id’));
        if(customer != null) {
            $exchange.send(JSON.stringify(customer));
        } else {
            $exchange.status(404);
        }
    }])
    .onGet("/customers",['db', function ($exchange, db) {
        var customers = db.select("select * from customer");
        $exchange.send(JSON.stringify(customers));
    }]);
```

Port Reduction

Uses HTTP Upgrade

Number of ports in default installation is ***two***

- 8080 for application
- 9990 for management

Only overhead is initial HTTP Upgrade

RBAC

Pre-defined administrative and privileged **roles**

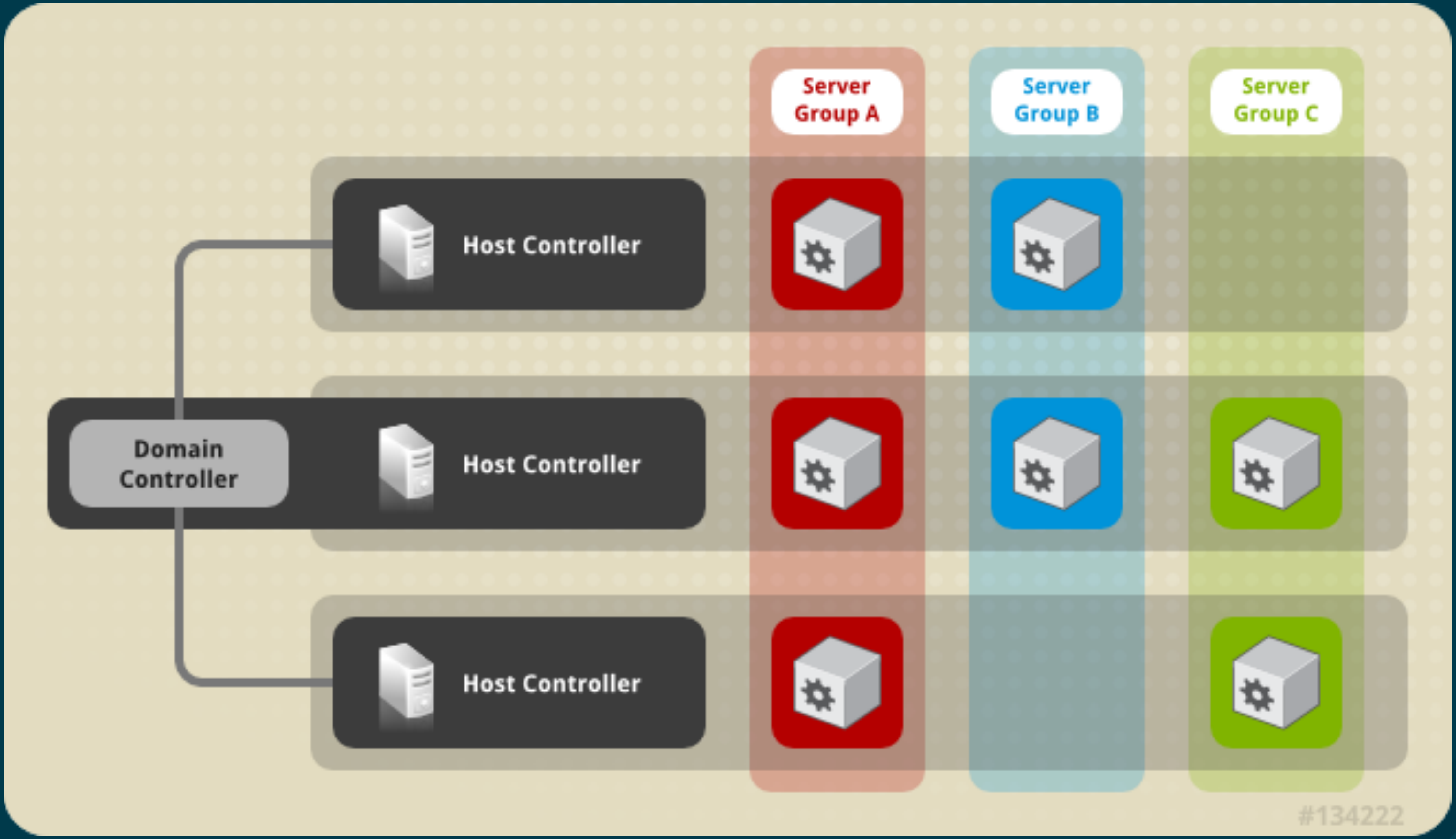
- Monitor, Operator, Maintainer, Deployer, Administrator, Auditor, SuperUser
- Plus scoped roles

Roles are a set of **permissions**

Permissions specify which management **actions** are allowed on resources

Users and **groups** are defined in roles

Standalone / Domain Mode



Command Line Interface

Tab completion

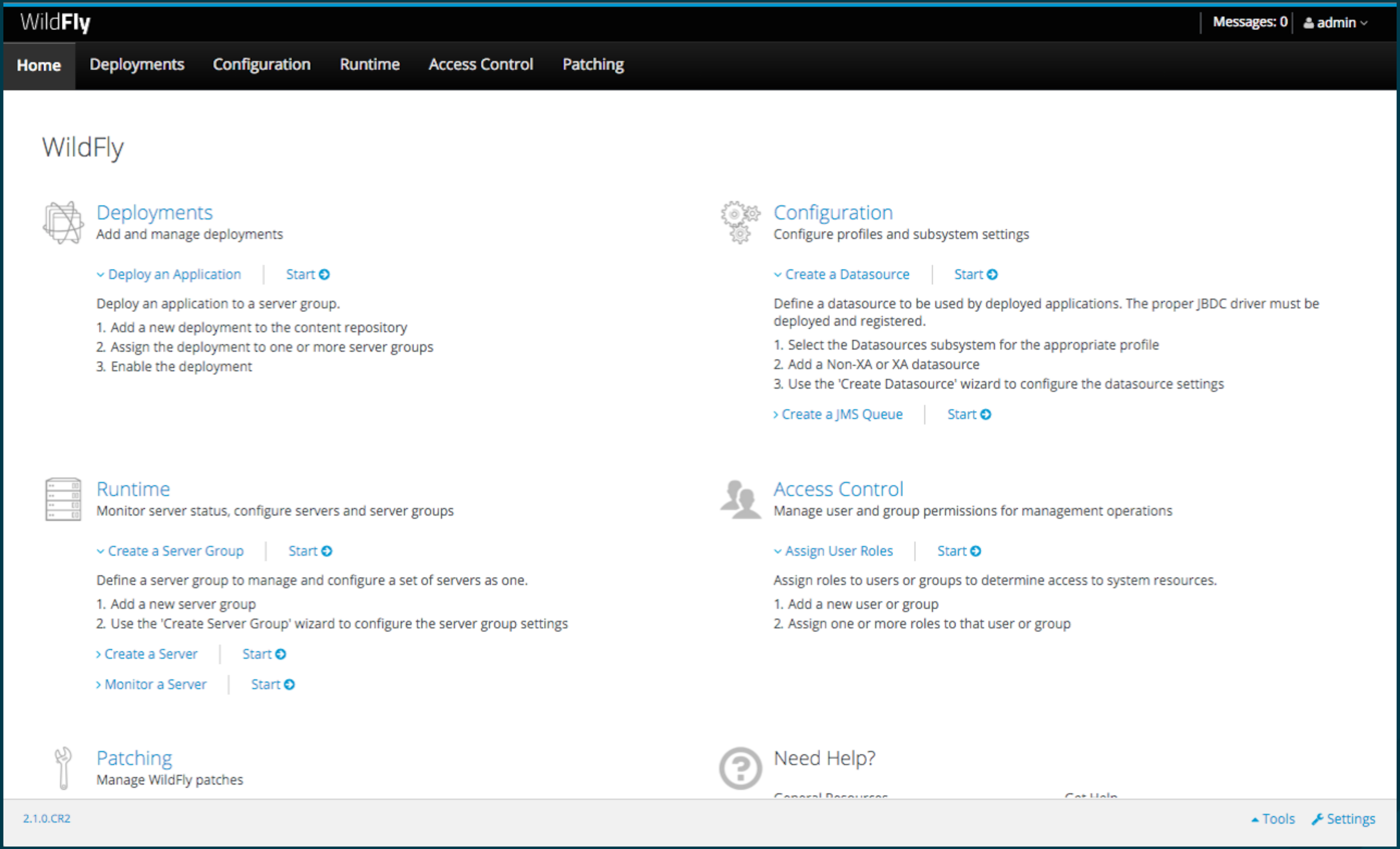
Scripts

High-level commands

Persistent changes

Offline CLI Support

Management Console



WildFly

Messages: 0 admin

Home Deployments Configuration Runtime Access Control Patching

WildFly

Deployments
Add and manage deployments

Deploy an Application | Start

Deploy an application to a server group.

1. Add a new deployment to the content repository
2. Assign the deployment to one or more server groups
3. Enable the deployment

Configuration
Configure profiles and subsystem settings

Create a Datasource | Start

Define a datasource to be used by deployed applications. The proper JBDC driver must be deployed and registered.

1. Select the Datasources subsystem for the appropriate profile
2. Add a Non-XA or XA datasource
3. Use the 'Create Datasource' wizard to configure the datasource settings

Create a JMS Queue | Start

Runtime
Monitor server status, configure servers and server groups

Create a Server Group | Start

Define a server group to manage and configure a set of servers as one.

1. Add a new server group
2. Use the 'Create Server Group' wizard to configure the server group settings

Create a Server | Start

Monitor a Server | Start

Access Control
Manage user and group permissions for management operations

Assign User Roles | Start

Assign roles to users or groups to determine access to system resources.

1. Add a new user or group
2. Assign one or more roles to that user or group

Patching
Manage WildFly patches

Need Help?

2.1.0.CR2

Tools Settings

WildFly

RED HAT® JBOSS®
ENTERPRISE
APPLICATION
PLATFORM

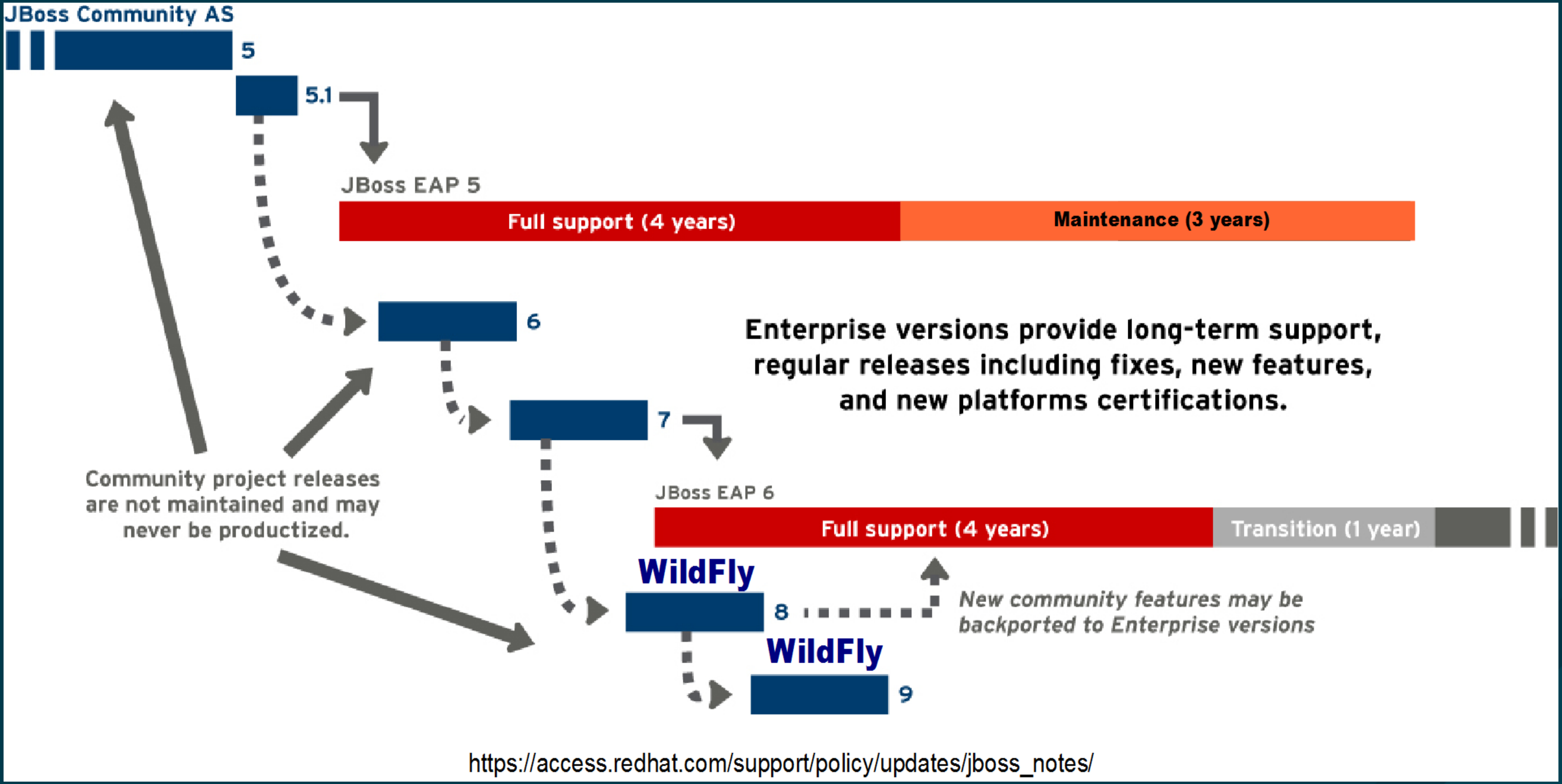
Project ↔ EE Spec

JBoss AS 2	→	J2EE 1.2
JBoss AS 3	→	J2EE 1.3
JBoss AS 4	→	J2EE 1.4
JBoss AS 5	→	Java EE 5
JBoss AS6, AS7	→	Java EE 6
WildFly 8, 9, 10	→	Java EE 7

Project ↔ EE Spec ↔ Product

JBoss AS 2	→	J2EE 1.2		
JBoss AS 3	→	J2EE 1.3		
JBoss AS 4	→	J2EE 1.4	→	EAP 4
JBoss AS 5	→	Java EE 5	→	EAP 5
JBoss AS6, AS7	→	Java EE 6	→	EAP 6
WildFly 8, 9, 10	→	Java EE 7	→	EAP 7

Interaction



Enterprise versions provide long-term support, regular releases including fixes, new features, and new platforms certifications.

https://access.redhat.com/support/policy/updates/jboss_notes/

WildFly SWARM

WildFly Swarm

OSS project sponsored by Red Hat

Sidekick of WildFly Application Server

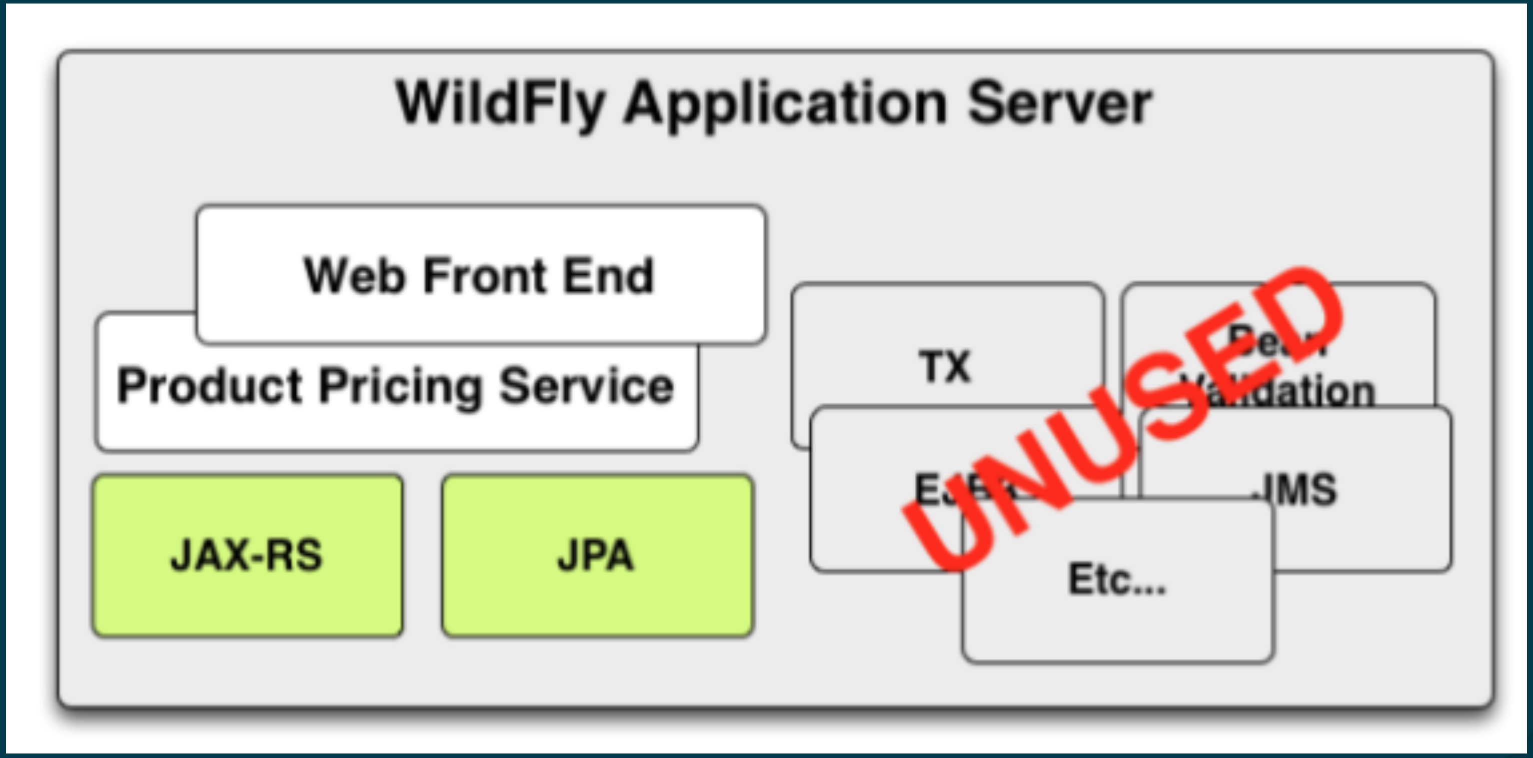
Part of a bigger system of interrelated projects under the JBoss / Red Hat umbrella

Just Enough App Server

Use the APIs you want

Include the capabilities you need

Wrap it up for deployment



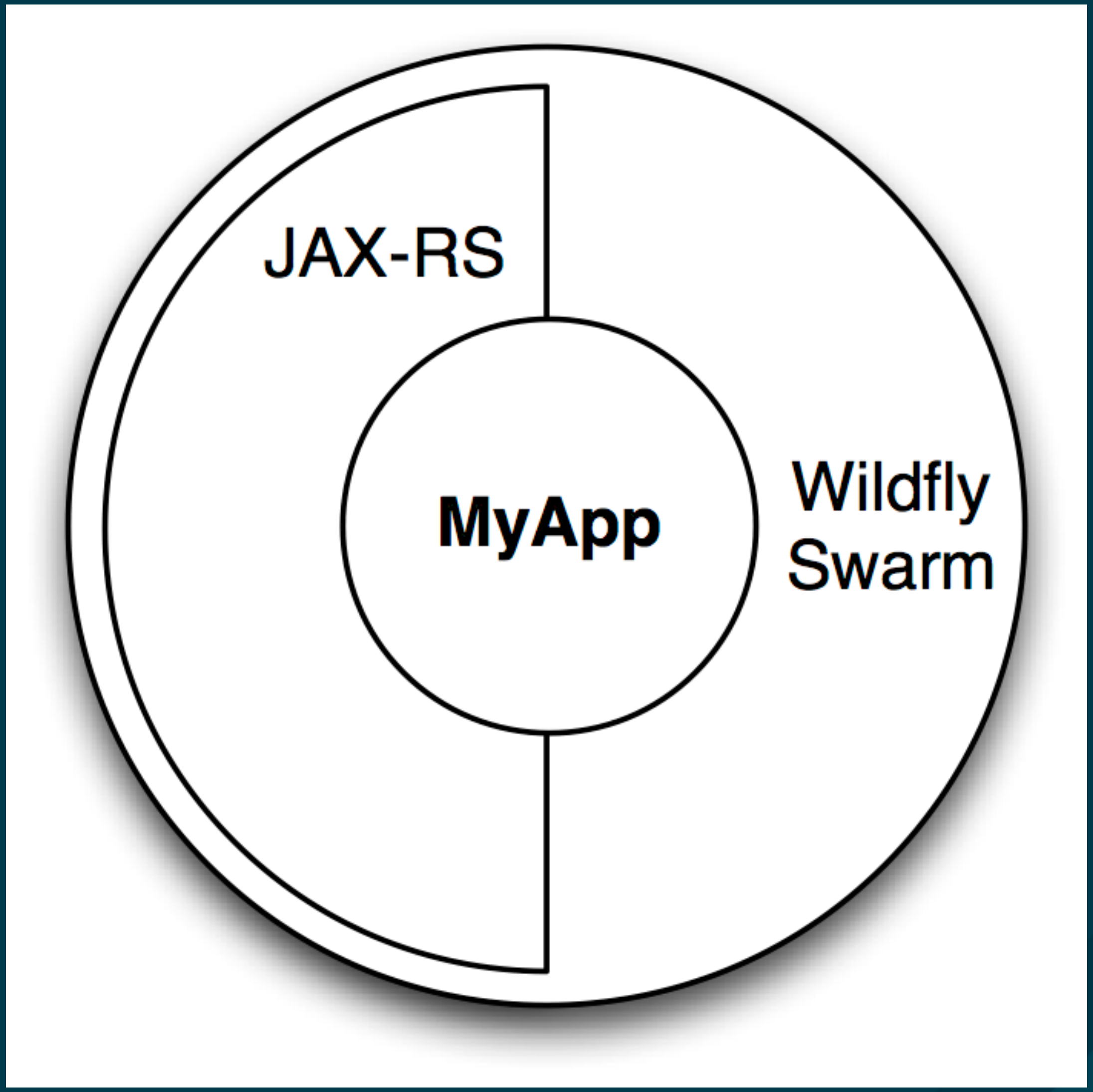
Uberjar

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



Fractions

Well-defined collection of 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)

Integrate additional system capabilities (Topology)

Provide deployments (ribbon-webapp, jolokia)

Alter deployments (keycloak)

Example Fractions

Datasources	Key cloak (SSO)	Undertow (HTTP / Web)
EJB	Messaging	Clustering
JAX-RS	JPA	Infinispan
Transactions	CDI	Management

Adding Fractions

```
<dependency>
    <groupId>org.wildfly.swarm</groupId>
    <artifactId>jaxrs-weld</artifactId>
    <version>${version.wildfly-swarm}</version>
</dependency>

<dependency>
    <groupId>org.wildfly.swarm</groupId>
    <artifactId>jaxrs-jaxb</artifactId>
    <version>${version.wildfly-swarm}</version>
</dependency>
```

Adding the WildFly Swarm Plugin

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

Building a WildFly Swarm App

```
$ mvn package
$ ls -l target/javaee7-simple-sample-swarm.jar
```

Running a WildFly Swarm App

```
$ java -jar target/javaee7-simple-sample-swarm.jar
$ mvn wildly-swarm:run
```

Going beyond simple
(and Java EE)

Custom Configuration

```
public class Main {

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

        container.fraction(new DatasourcesFraction()
                .jdbcDriver("h2", (d) -> {
                    d.driverClassName("org.h2.Driver");
                    d.xaDatasourceClass("org.h2.jdbcx.JdbcDataSource");
                    d.driverModuleName("com.h2database.h2");
                })
                .dataSource("MyDS", (ds) -> {
                    ds.driverName("h2");
                    ds.connectionUrl("jdbc:h2:mem:test;...");
                    ds.userName("sa");
                    ds.password("sa");
                })
        );

       container.start();
    }
}
```

alternatively use standalone.xml

Advertising Services

```
public class Main {

    public static void main(String[] args) throws Exception {

        Container container = new Container();

        JAXRSArchive deployment = ShrinkWrap.create(JAXRSArchive.class);
        deployment.addPackage(Main.class.getPackage());
        deployment.as(Topology.class).advertise("events");

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

supports different service registries

Load Balancing & Circuit Breaker

```
@ResourceGroup( name="time" )
public interface TimeService {

    TimeService INSTANCE = Ribbon.from(TimeService.class);

    @TemplateName("currentTime")
    @Http(method = Http.HttpMethod.GET, uri = "/" )
    @Hystrix(fallbackHandler = TimeFallbackHandler.class)
    RibbonRequest<ByteBuf> currentTime();
}
```

Integration of Ribbon with Topology , supports Hystrix

Securing Access to Services

```
public class Main {

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

        JAXRSArchive deployment = ShrinkWrap.create(JAXRSArchive.class);
        deployment.addPackage(Main.class.getPackage());
        deployment.as(Secured.class)
                .protect("/items")
                .withMethod("GET")
                .withRole("*");

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

provided by Keycloak: OpenID, SAML, Social Login, OAuth, LDAP, Active Directory

Publish Service Interface Description

```
@Path("/time")
@Api(value = "/time", description = "Get the time", tags = "time")
@Produces(MediaType.APPLICATION_JSON)
public class TimeResource {

    @GET
    @Path("/now")
    @ApiOperation(value = "Get the current time",
            notes = "Returns the time as a string",
            response = String.class
    )
    @Produces(MediaType.APPLICATION_JSON)
    public String get() {
        return String.format("{\"value\" : \"The time is %s\"}", new DateTime());
    }
}
```

provided by Swagger

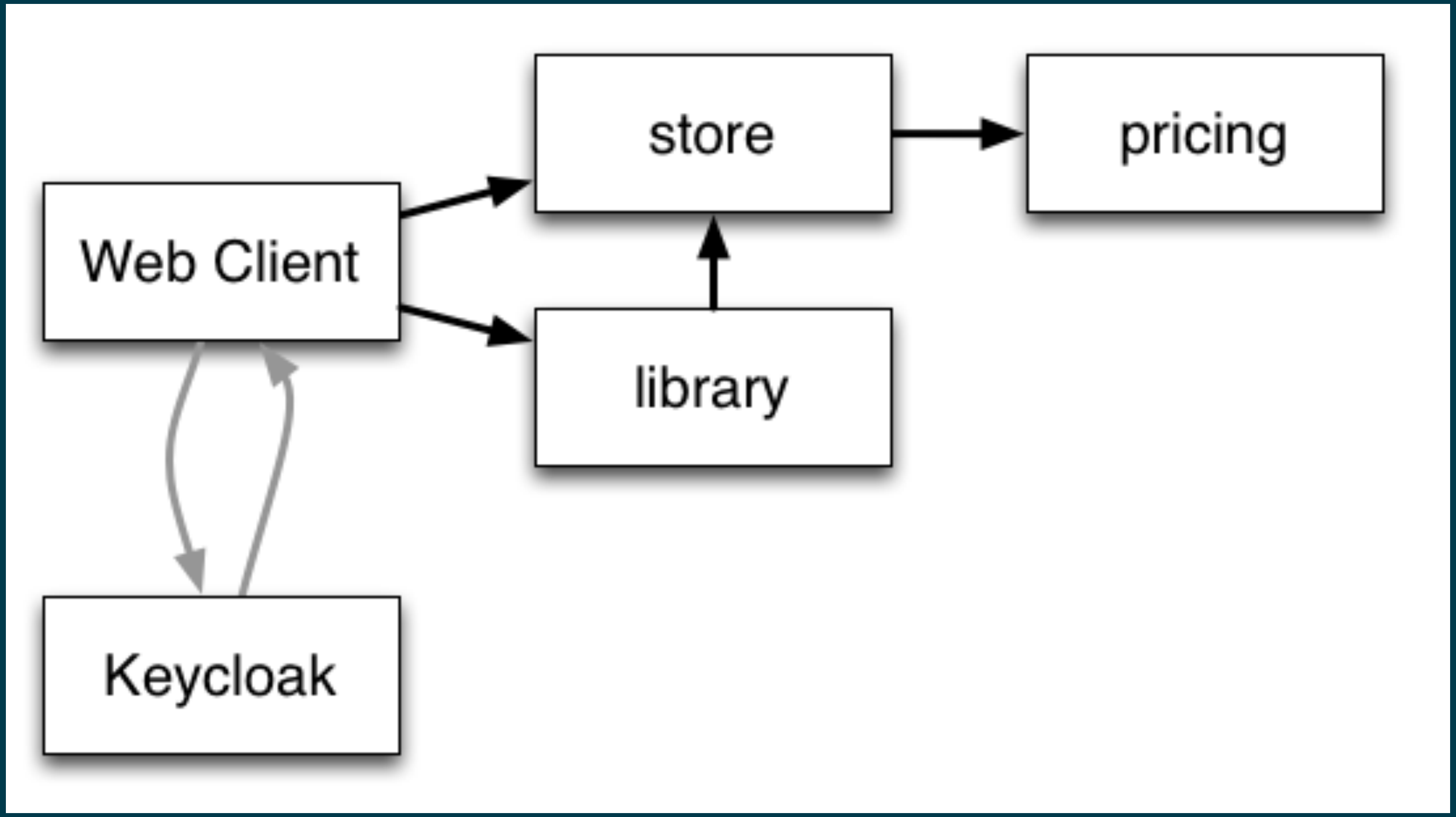
curl http://localhost:8080/swagger.json

```
{
  "swagger": "2.0",
  "info": {},
  "basePath": "/",
  "tags": [
    {
      "name": "time"
    }
  ],
  "paths": {
    "/time/now": {
      "get": {
        "tags": [
          "time"
        ],
        "summary": "Get the current time",
        "description": "Returns the time as a string",
        "operationId": "get",
        "produces": [
          "application/json"
        ],
        "parameters": [],
        "responses": {
          "200": {
            "description": "successful operation",
            "schema": {
              "type": "string"
            }
          }
        }
      }
    }
  }
}
```

Booker Demo

Booker! is an electronic bookstore that demonstrates how many WildFly Swarm-based microservices can play together.

https://github.com/wildfly-swarm/booker



The Road Ahead

API Gateway - Integration with APIMan

Integration with Kubernetes / OpenShift v3 (Service Discovery)

Environment Abstractions (Local, CI, Cloud)

Tooling (Forge, Eclipse, IntelliJ)

Spring Support

Resources

Project Home	http://wildfly.org	http://wildfly-swarm.io
Source Code	https://github.com/wildfly	https://github.com/wildfly-swarm
Twitter	@WildFlyAS	@wildflyswarm
Chat	https://www.hipchat.com/gW90m6pls	#wildfly-swarm
Issue Tracker	http://jira.jboss.org/browse/WFLY	https://issues.jboss.org/projects/SWARM