Management and Monitoring

Zbyněk Roubalík

Senior Quality Engineer, JBoss by Red Hat

Advanced Java EE Lab @ MUNI

May 10 2018

Agenda

- Monitoring
  - JDK tools, System tools, WildFly specifics
- WildFly history and overview
- WildFly 10
  - Architecture, Domain Model,RBAC
- WildFly Swarm
- WildFly 10 Management
  - CLI / Scripting + Java API + HTTP API
  - WebUI
- RHQ, Hawkular
- Openshift

Monitoring – motivation

You are using WildFly 10, so bright future lies ahead ...

Really?

We will learn how to do some basic investigation and JVM monitoring.

JDK tools - JAR level investigation

- List files in given jar archive
  - jar
  - unzip
- Disassemble the class file
  - javap

JDK tools – process

- List of JVMs
  - jps -l [-m -v]
  - JDK specific

JDK tools – memory

- Memory map
  - jmap
    - Show heap, create heap dump
- Analyze heap dump
  - jhat
    - Parses a java heap dump, launches a webserver to browse the dump

JDK tools – stack trace and JVM stats

- Java stack traces of threads
  - jstack
    - stack traces of Java threads for a given Java process, core or remote server
    - for investigating thread locking issues
- JVM statistics monitoring
  - jstat

JDK tools – GUI

jconsole

- Heap and Non-Heap memory usage, CPU usage, VM summary
- Number of threads and classes, stack trace for each thread
- MBeans details

VisualVM (jvisualvm before)

- Nicer look & feel, based on NetBeans platform
- Heap and PermGen memory usage, CPU usage, VM summary
- Number of threads and classes, details for each thread, not stack trace
- Lightweight CPU and memory profiling + sampling

System information

- OS version
- Memory usage
- Disk space
- Processes
- Network – traffic and ports

WildFly specifics

JDR - JBoss Diagnostic Reporter

- $WF_HOME/bin/jdr.sh [.bat]
- JBoss specific tool for diagnostic
- add at least one user into ManagementRealm using bin/add-user.sh

jconsole

- $WF_HOME/bin/jconsole.sh [.bat]
- Jconsole with added WildFly management extension (JBoss Remoting + JSR 160)

Advanced tools

- your IDE debugger
- your IDE profiler
- JProfiler - http://www.ej-technologies.com/products/jprofiler/overview.html
- Java Decompiler - http://java.decompiler.free.fr/
- TDA - Thread Dump Analyzer - http://java.net/projects/tda/
- MAT - Memory Analyzer - http://www.eclipse.org/mat/

- Wireshark - http://www.wireshark.org/

WildFly history and overview

- Named JBoss AS before
- Why was AS7 rewritten from scratch?
  - Legacy subsystems
  - Boot time
  - Memory footprint
  - Bad modularity
  - Administration options
  - Not “good enough”

WildFly history and overview

- Wildfly 8
  - Builds on top of JBoss AS7
  - Small and even #@*%ing faster
  - No legacy stuff
  - Better manageability
  - Multi-node management
  - Simplified configuration
  - Modular

WildFly history and overview

- Wildfly 9
  - HTTP/2 Support
  - Front End Load Balancer Support
  - Graceful Shutdown
  - WildFly Swarm
- Wildfly 10
  - Java 8+
  - ActiveMQ Artemis
  - JavaScript Support with Hot Reloading

WildFly 10 Architecture

- **core**
- **extensions** to the core
- **clients** for management interface
  - CLI and web based management console

Core

- jboss-modules
  - is the first thing started
  - modular and concurrent classloading
  - O(1) dependencies resolution
  - Module sees only its imports
- jboss-msc: modular service container
  - Everything is (interface based) service
  - Services are deployed on demand and in parallel
- Extensible management layer
  - Mediate access to service container
  - Provides configuration model for the AS

Domain vs. standalone

Standalone

- Traditional JBoss single JVM server
- Managed individually: 1 configuration file
- No lifecycle management, just shutdown
- Development and embedded solutions

Domain

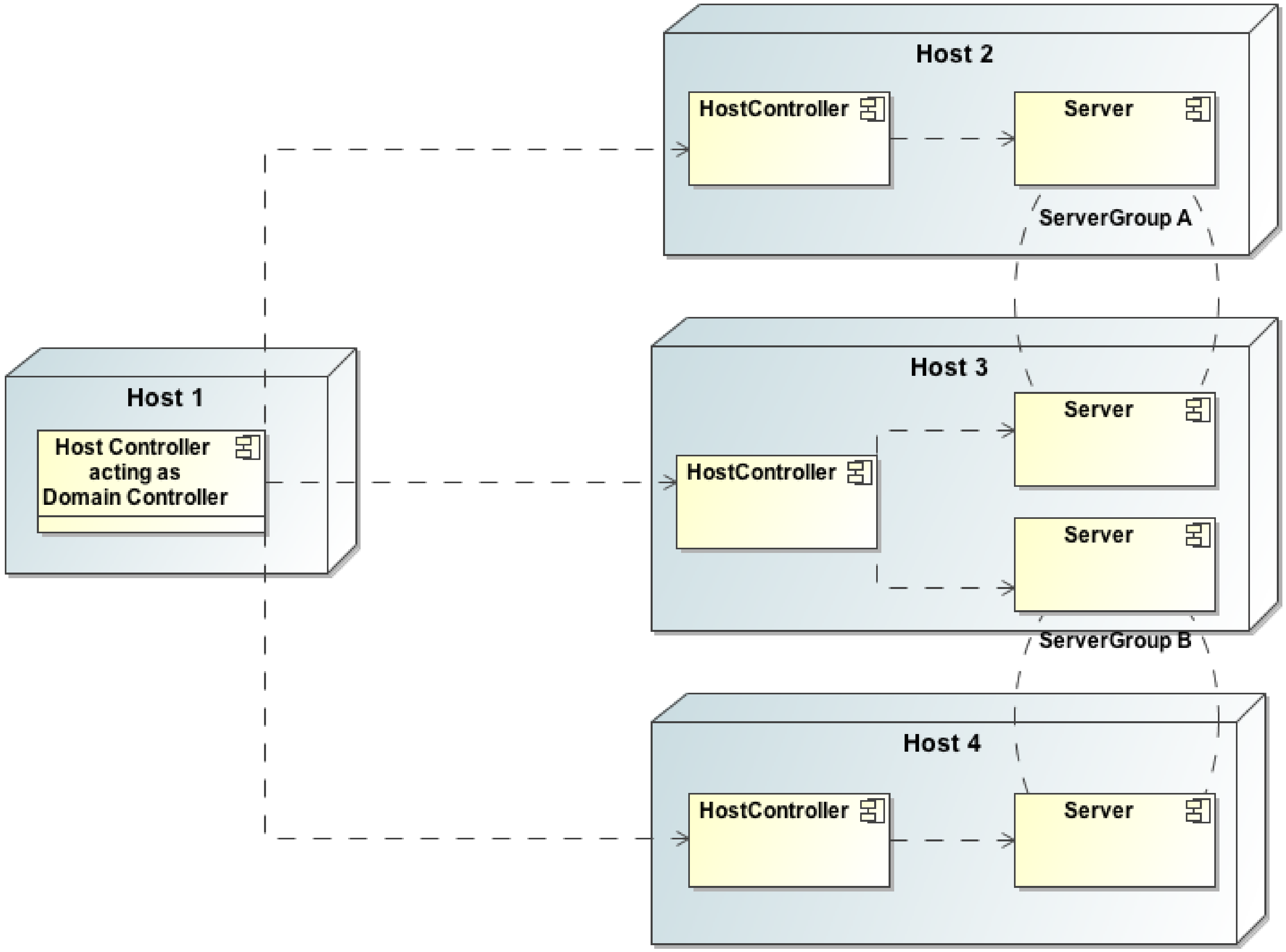
- Multi-JVM, multi-server model
- Lifecycle managed by Process Controller (PC)
- Management coordinated by Domain Controller (DC)
- Multiple server instances per host managed by Host Controller (HC)
- HC on master node is DC

The only difference between domain and standalone is in how severs are managed, not in the capabilities

Domain model: key goals

- manage multiple servers via a single control point
  - configure a cluster, start/stop nodes in a cluster, deploy an application to all nodes in the domain,...
- end user configuration centralized in a few files
- schema files for all configurations
- everything in the configuration is exposed via management API

Domain model



Domain model - terms

- **server** - one AS instance
- **server group** - set of server instances that will be managed and configured as one
- **cluster** - server group with group communication services configured
- **module** - classloading space, grouping of classes in some jar(s)s
- **subsystem** - block of configuration, has its own namespace, basically some grouping of services
- **profile** - set of subsystems

Role Based Access Control (RBAC)

- Different users have different sets of permissions to read and update parts of the management tree
- Replaces the simple permission scheme used in JBoss AS 7, when authenticated user have all permissions

- **Role** - named set of permissions (read, modify management resource
- Mapping users and groups to roles
- https://docs.jboss.org/author/display/WFLY10/RBAC

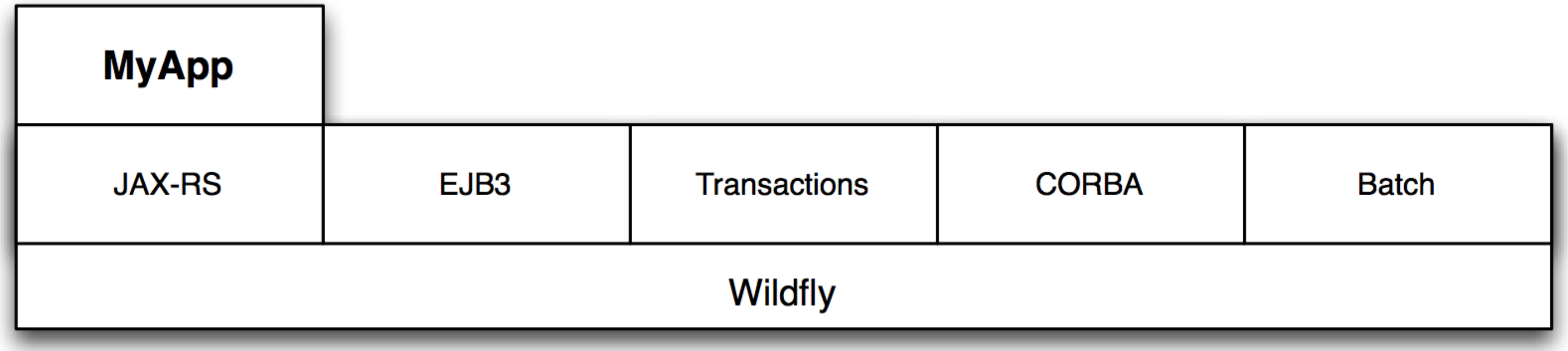
RBAC roles

- Not given permissions for "security sensitive" items:
  - **Monitor** – read only
  - **Operator** – Monitor + modify runtime state
  - **Maintainer** – Operator + modify persistent config.
  - **Deployer** – Operator + modify "application resources"

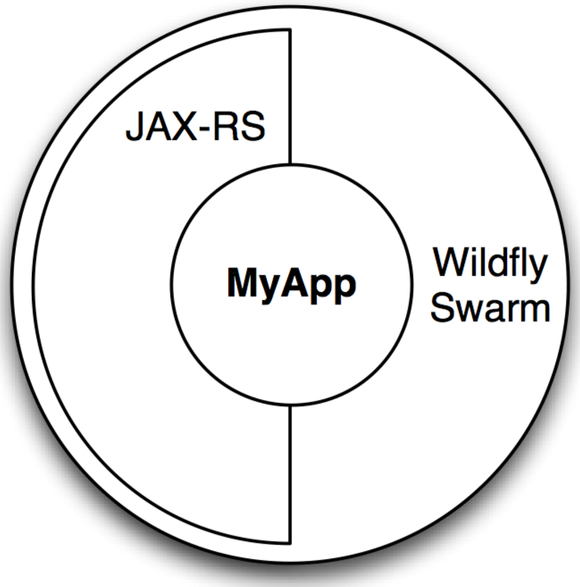
- Given permissions for "security sensitive" items:
  - **SuperUser** – all permissions (== JBoss AS 7 admin)
  - **Administrator** – all permissions except cannot read or write resources related to the administrative audit logging system
  - **Auditor** – can read anything. Can only modify the resources related to the administrative audit logging system.

WildFly Swarm

- Monolithic App server
  - Traditional model – more functionality than needed



- WildFly Swarm Uberjar
  - Just enough app server
  - Smaller usage of resources
  - Microservices



WildFly Swarm

- Fraction
  - well-defined collection of capabilities to add, (in most cases maps directly to WF subsystem)
- Uberjar
  - A self-contained, executable Java archive
- Requires JDK 8+
- Maven
- http://wildfly-swarm.io

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

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

```
java -jar MyApp-swarm.jar
```

Management

- The problem: management model too large and complex
- The requirements for the API:
  - Simple, powerful, stable
  - As few compile time and runtime dependencies as possible
  - Backward compatibility

- WF uses de-typed management API and a small library:

  jboss-dmr.jar

DMR – dynamic model representation

- https://github.com/jbossas/jboss-dmr
- https://docs.jboss.org/author/display/WFLY10/Detyped+management+and+the+jboss-dmr+library
- All management operations operate with/on DMR
- Compatibility is stressed
- Convertible from/to JSON

- Wildfly Model Reference Documentation:
  - https://wildscribe.github.io/index.html

Java API

- Native management interface uses an open protocol based on the JBoss Remoting library
- The management protocol is an open protocol, so a completely custom client could be developed without using prepared libraries (e.g. using Python or some other language)

- Maven artifact org.wildfly.core:wildfly-controller-client

- https://docs.jboss.org/author/display/WFLY10/The+native+management+API

Java API

```
ModelControllerClient client = ModelControllerClient.Factory.
        create(InetAddress.getByName("localhost"), 9999);

ModelNode op = new ModelNode();
op.get("operation").set("read-resource");
op.get("recursive").set(true);
op.get("include-runtime").set(true);
op.get("recursive-depth").set(10);

ModelNode returnVal = client.execute(op);
System.out.println(returnVal.get("result").toString());
client.close();
```

HTTP API

- http://localhost:9990/management
- Sometimes called REST API
- HTTP request in JSON like format
- The default operation is read-resource
- add user into ManagementRealm using bin/add-user.sh

- https://docs.jboss.org/author/display/WFLY10/The+HTTP+management+API
- https://community.jboss.org/wiki/HTTPJSON-likeAPI

CLI

- Command line management tool for the WF server
- Command bin/jboss-cli.sh or bin/jboss-cli.bat
- Interactive mode
- Non-interactive mode
- Batch mode
- GUI mode
- Operations based on model

CLI

```
$ ./bin/jboss-cli.sh --connect controller=IP_ADDRESS
[standalone@IP_ADDRESS:9999 /] /system-property=foo:add(value=bar)
[standalone@IP_ADDRESS:9999 /] /system-property=foo:read-resource
{
    "outcome" => "success",
    "result" => {"value" => "bar"}
}
[standalone@IP_ADDRESS:9999 /] /system-property=foo:remove
{"outcome" => "success"}
```

```
[domain@IP_ADDRESS:9999 /] /system-property=foo:add(value=bar)
[domain@IP_ADDRESS:9999 /] /system-property=foo:read-resource
[domain@IP_ADDRESS:9999 /] /system-property=foo:remove
```

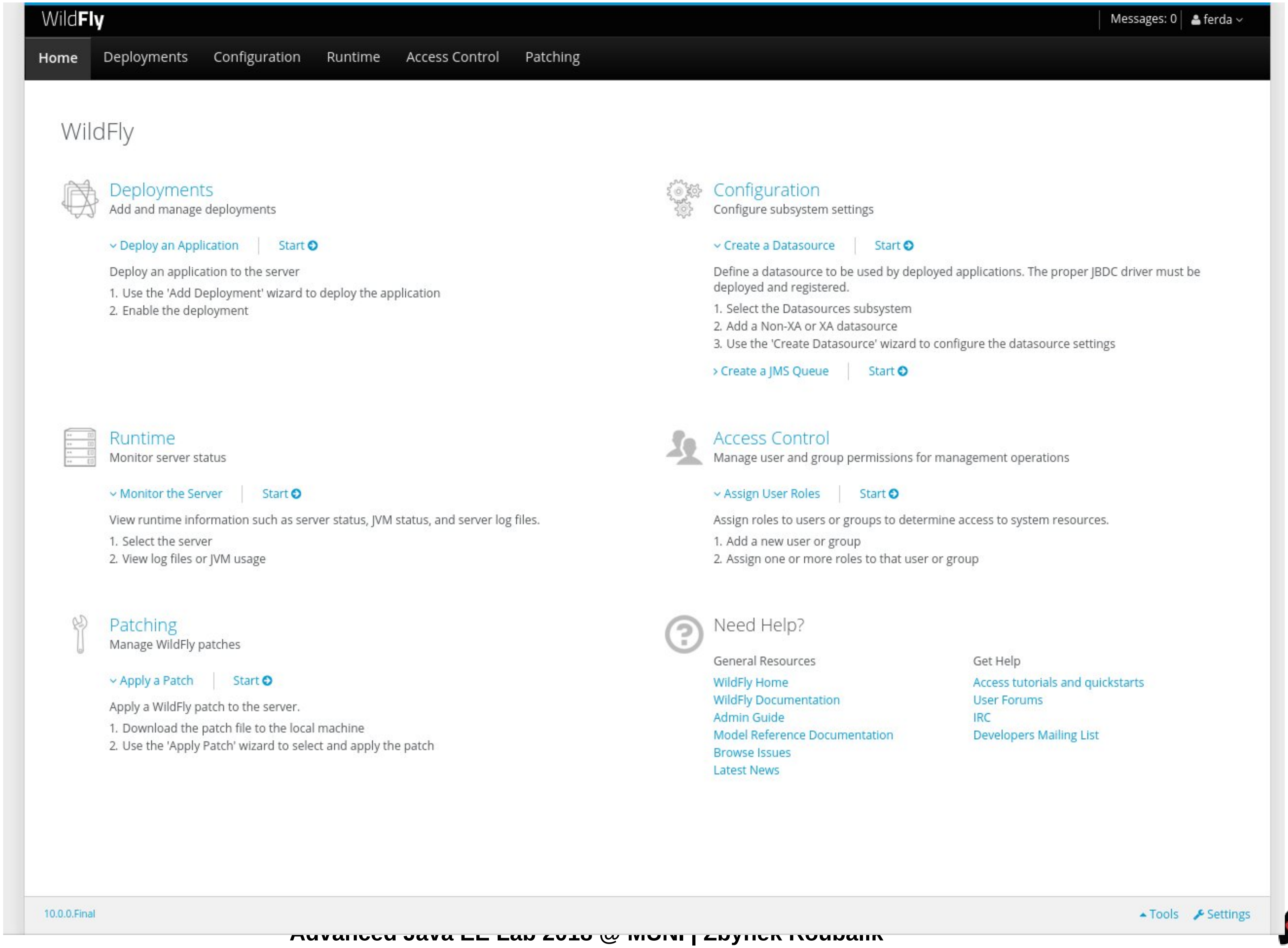
```
[domain@IP_ADDRESS:9999 /] /host=master/system-property=foo:add(value=bar)
[domain@IP_ADDRESS:9999 /] /host=master/system-property=foo:read-resource
[domain@IP_ADDRESS:9999 /] /host=master/system-property=foo:remove
```

```
[domain@IP_ADDRESS:9999 /] /host=master/server-config=server-one/system-property=foo:add(value=bar)
[domain@IP_ADDRESS:9999 /] /host=master/server-config=server-one/system-property=foo:read-resource
[domain@IP_ADDRESS:9999 /] /host=master/server-config=server-one/system-property=foo:remove
```

CLI

- https://community.jboss.org/wiki/CommandLineInterface
- https://community.jboss.org/wiki/GenericTypeCLICommands
- https://community.jboss.org/wiki/CLICompoundValueFormat
- https://community.jboss.org/wiki/CLINon-interactiveMode
- https://community.jboss.org/wiki/CLIBatchMode
- https://docs.jboss.org/author/display/WFLY10/CLI+Recipes

Web console



WildFly

Deployments
Add and manage deployments

Deploy an Application | Start

Deploy an application to the server

1. Use the 'Add Deployment' wizard to deploy the application
2. Enable the deployment

Configuration
Configure 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
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

Monitor the Server | Start

View runtime information such as server status, JVM status, and server log files.

1. Select the server
2. View log files or JVM usage

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

Apply a Patch | Start

Apply a WildFly patch to the server.

1. Download the patch file to the local machine
2. Use the 'Apply Patch' wizard to select and apply the patch

Need Help?

General Resources
- WildFly Home
- WildFly Documentation
- Admin Guide
- Model Reference Documentation
- Browse Issues
- Latest News

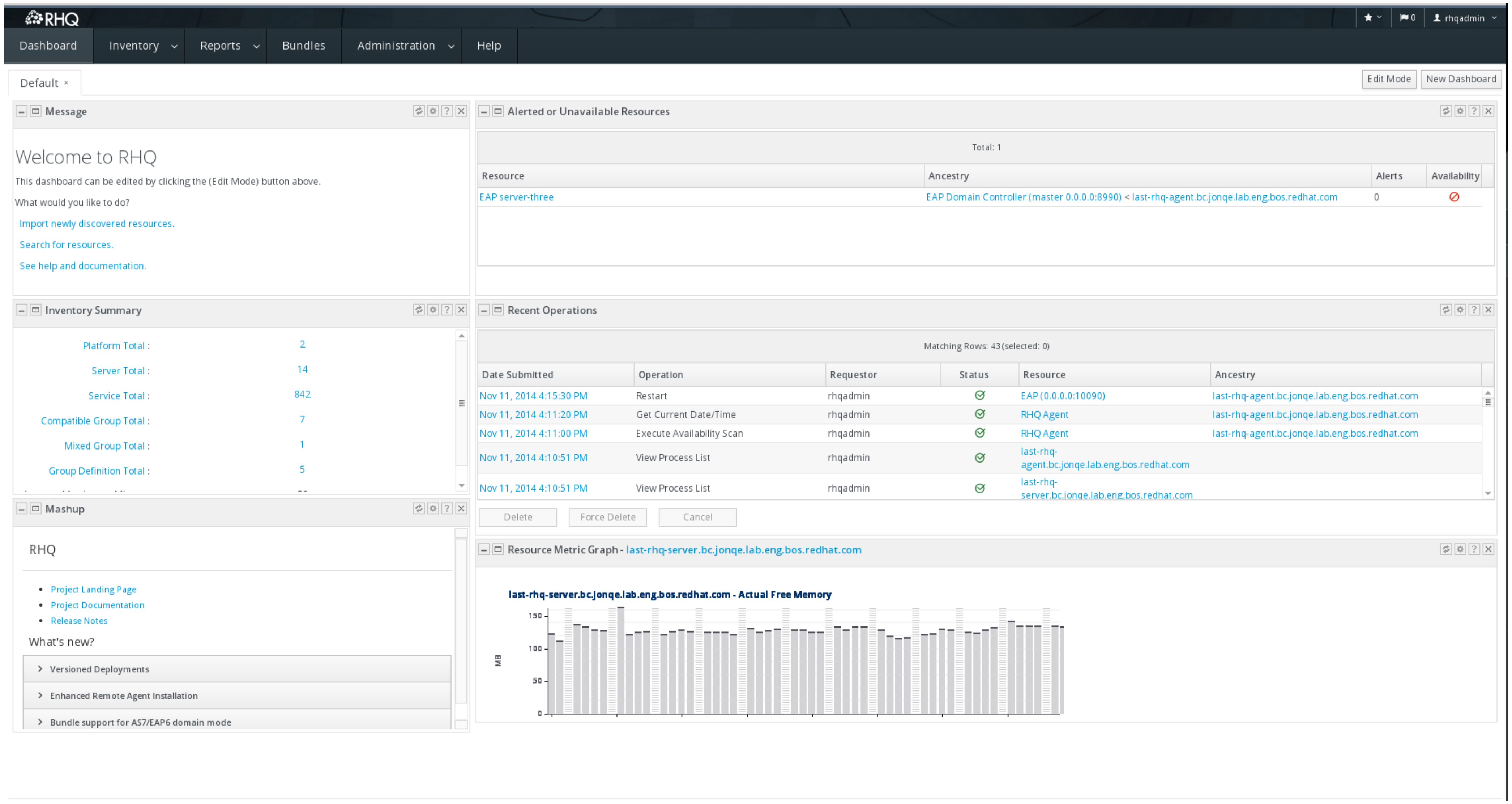
Get Help
- Access tutorials and quickstarts
- User Forums
- IRC
- Developers Mailing List

RHQ

- RHQ is an enterprise management solution for JBoss middleware projects, Tomcat, Apache Web Server, etc.
- Server-side and agent-side (extendable via plugins)
- Features
  - Inventory – tracking resources (autodiscovery)
  - Configuration – audited, rollback
  - Monitoring – collection of statistics
  - Alerts – to provide notifications of user defined conditions
  - Operations – ability to execute actions against managed resources in the inventory

- https://rhq-project.github.io/rhq/

RHQ



Message

Welcome to RHQ

This dashboard can be edited by clicking the (Edit Mode) button above.

What would you like to do?

Import newly discovered resources.

Search for resources.

See help and documentation.

Alerted or Unavailable Resources

Total: 1

Resource	Ancestry	Alerts	Availability
EAP server-three	EAP Domain Controller (master 0.0.0.0:8990) < last-rhq-agent.bc.jonqe.lab.eng.bos.redhat.com	0	⊘

Inventory Summary

Platform Total :	2
Server Total :	14
Service Total :	842
Compatible Group Total :	7
Mixed Group Total :	1
Group Definition Total :	5

Recent Operations

Matching Rows: 43 (selected: 0)

Date Submitted	Operation	Requestor	Status	Resource	Ancestry
Nov 11, 2014 4:15:30 PM	Restart	rhqadmin	✓	EAP (0.0.0.0:10090)	last-rhq-agent.bc.jonqe.lab.eng.bos.redhat.com
Nov 11, 2014 4:11:20 PM	Get Current Date/Time	rhqadmin	✓	RHQ Agent	last-rhq-agent.bc.jonqe.lab.eng.bos.redhat.com
Nov 11, 2014 4:11:00 PM	Execute Availability Scan	rhqadmin	✓	RHQ Agent	last-rhq-agent.bc.jonqe.lab.eng.bos.redhat.com
Nov 11, 2014 4:10:51 PM	View Process List	rhqadmin	✓	last-rhq-agent.bc.jonqe.lab.eng.bos.redhat.com	
Nov 11, 2014 4:10:51 PM	View Process List	rhqadmin	✓	last-rhq-server.bc.jonqe.lab.eng.bos.redhat.com	

Delete | Force Delete | Cancel

Mashup

RHQ

- Project Landing Page
- Project Documentation
- Release Notes

What's new?

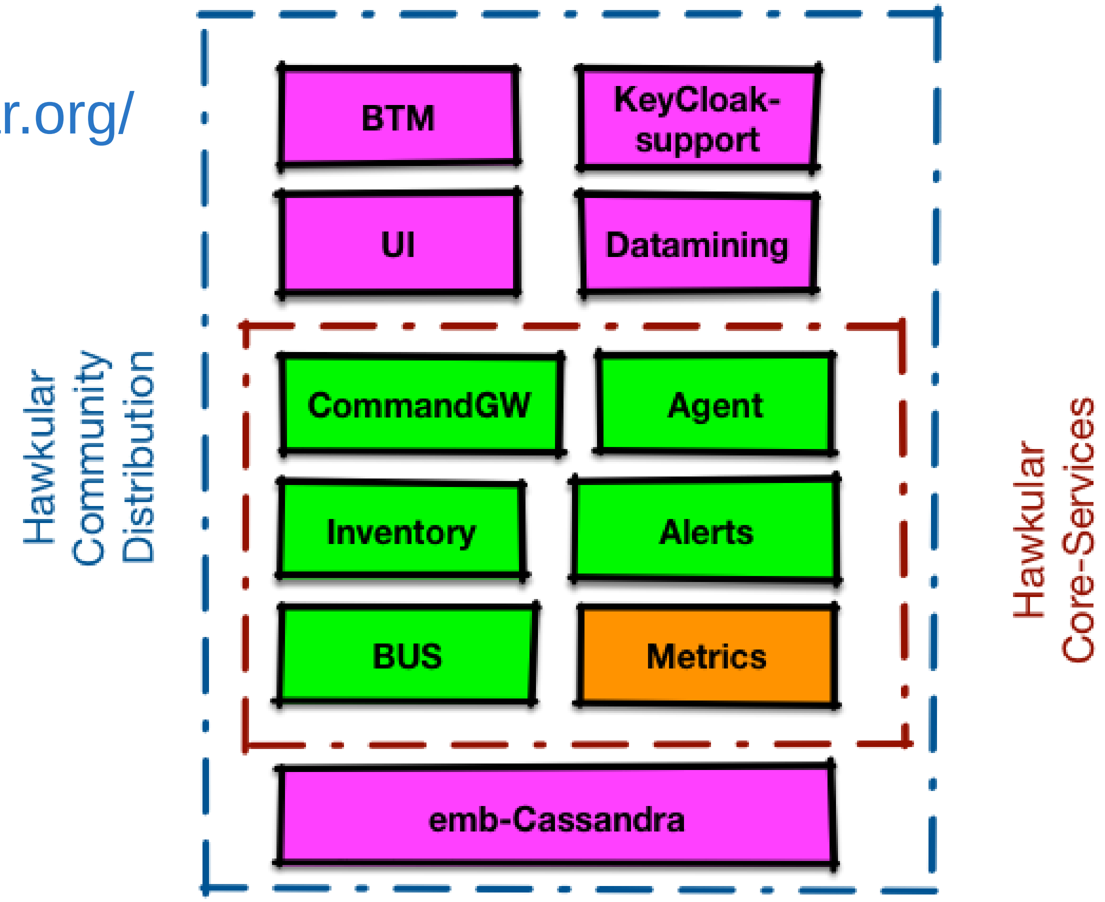
- Versioned Deployments
- Enhanced Remote Agent Installation
- Bundle support for AS7/EAP6 domain mode

Resource Metric Graph - last-rhq-server.bc.jonqe.lab.eng.bos.redhat.com

last-rhq-server.bc.jonqe.lab.eng.bos.redhat.com - Actual Free Memory
MB
150
100
50
0

Hawkular

- Successor of RHQ
  - Set of independent services sharing information over a communication bus
- http://www.hawkular.org/



Openshift

- Cloud vs standard deployment model
  - PaaS
  - Cattle vs pets
  - DevOps
- Container
  - Docker
- Kubernetes
- UI Demo
- https://www.openshift.com/

Thank you for your attention.

Questions?