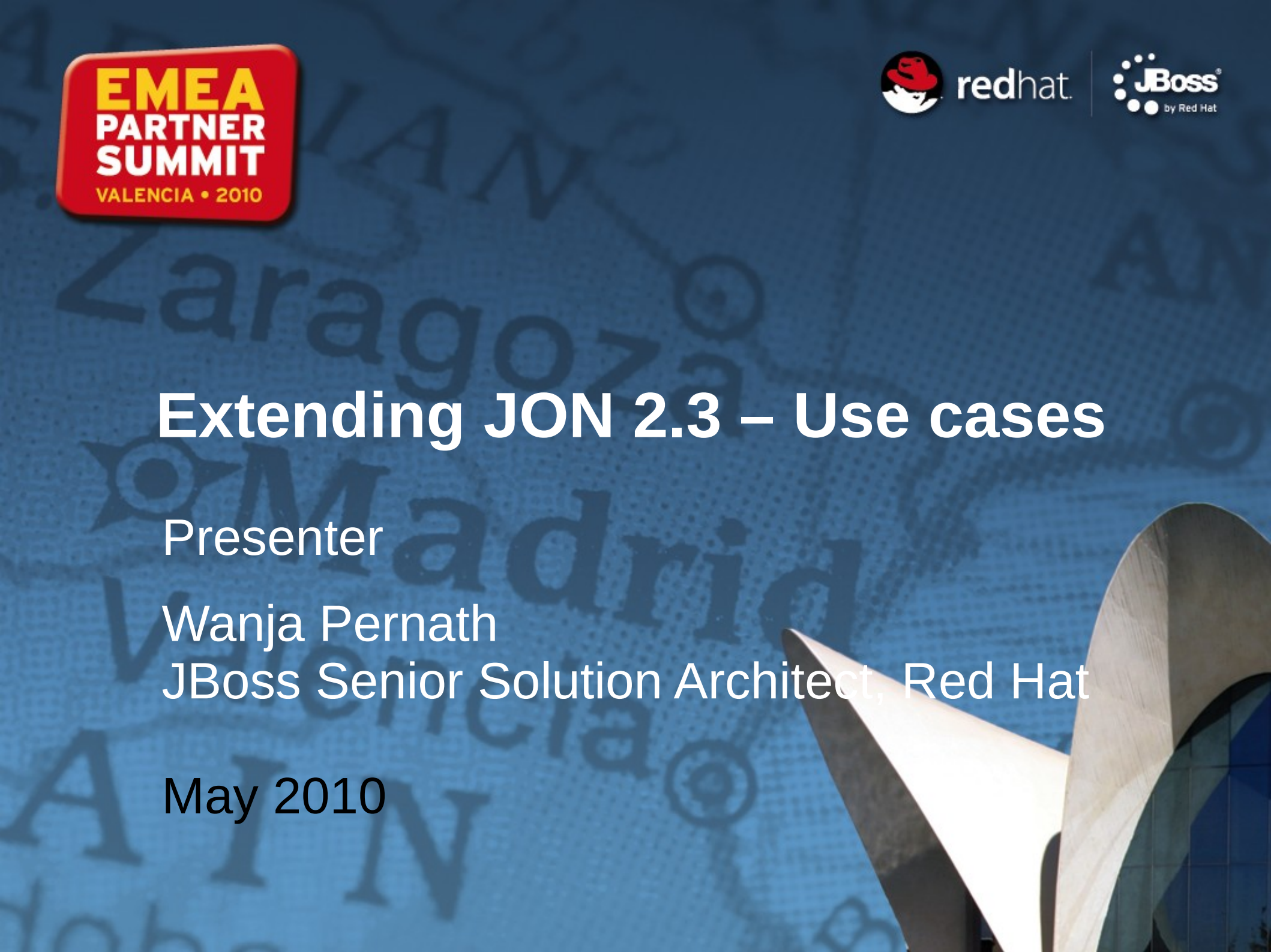


Extending JON 2.3 – Use cases

Presenter

Wanja Pernath
JBoss Senior Solution Architect, Red Hat

May 2010

Agenda

- How does JON help you
- JON in different scenarios
  - New JBoss project
  - Migration project
- JON at different customers
  - A German health card provider
  - Direct bank in Germany
- The solution for the direct bank in Germany in more detail
- The solution for the health card provider in Germany

How does JON help you

- JBoss Operations Network (JON) is a quite flexible monitoring and management solution for JBoss application servers
- However, its real strengths are in its flexible plug in based architecture
  - Each application needs some configuration - why not just putting the configuration part of the application into JON?
  - You need some other special configuration tool for another 3rd party framework? Integrate it into JON
- Use the Command Line API and the Remote API to extend the functionality of JON

JON in different scenarios

- If a customer is migrating from another Application Server vendor
  - Most of the time they are used to some specific tooling
    - Use JON to help them becoming confident with the app server
    - If they are used to a special corporate identity and a special set of use cases, just use the remote API to write a web based management tool for JON
  - If a customer has used the scripting functions, setup a set of scripts via JON CLI
- A customer who just knows jboss.org is often very interested in having a concrete operation model: How can I manage my 100 JBoss instances?
  - --> JON is very flexible

JON for different customers

- A migration project from Weblogic 8.x to JBoss: A direct bank in germany
  - They are using Weblogic since early 2000.
  - All their processes are aligned with the Weblogic products
  - Everything is automated as much as possible
  - Scripts have been created for deployment to ~200 WL instances
  - Solution here was the help them creating a set of scripts for JON (groupdeployment script set)

Deploying a new app to a group

- First we need to get the resource type of the given application type (EAR/WAR)
- We then need to get a package type which belongs to the application type
- Where do we want to deploy to? Specify deployment properties
- And then simply create the package backed resource

```
var appType = ResourceTypeManager.getResourceTypeByNameAndPlugin( appTypeName, "J
if( appType == null ) {
    println("  Could not find application type. Exit.");
    usage();
}
```

```
var realPackageType = ContentManager.findPackageTypes( appTypeName, "JBossAS" );

if( realPackageType == null ) {
    println("  Could not find JON's packageType. Exit.");
    usage();
}
```

```
// create deployConfig
var deployConfig = new Configuration();
deployConfig.put( new PropertySimple("deployDirectory", "deploy"));
deployConfig.put( new PropertySimple("deployZipped", "true"));
deployConfig.put( new PropertySimple("createBackup", "false"));
```

```
ResourceFactoryManager.createPackageBackedResource(
    server.id,
    appType.id,
    packageName,
    null,  // pluginConfiguration
    packageName,
    packageVersion,
    null, // architectureId
    deployConfig,
    fileBytes
);
```


JON for different customers #2

- A Jboss.org to Enterprise migration project: Using JON for their internal datacenter admins
  - They were used to use expensive 3rd party tools to have different monitoring views
    - A 20000 feet bird view for a datacenter admin (i.e. It's running or not)
    - A detailed view for the more experienced admins who need to search for a bug
  - The datacenter admin view must be as simplistic as possible
    - Solution here was to use the Remote API to build their own Seam based web application which is displaying the metrics from JON

The Solution for the Direct Bank in Germany

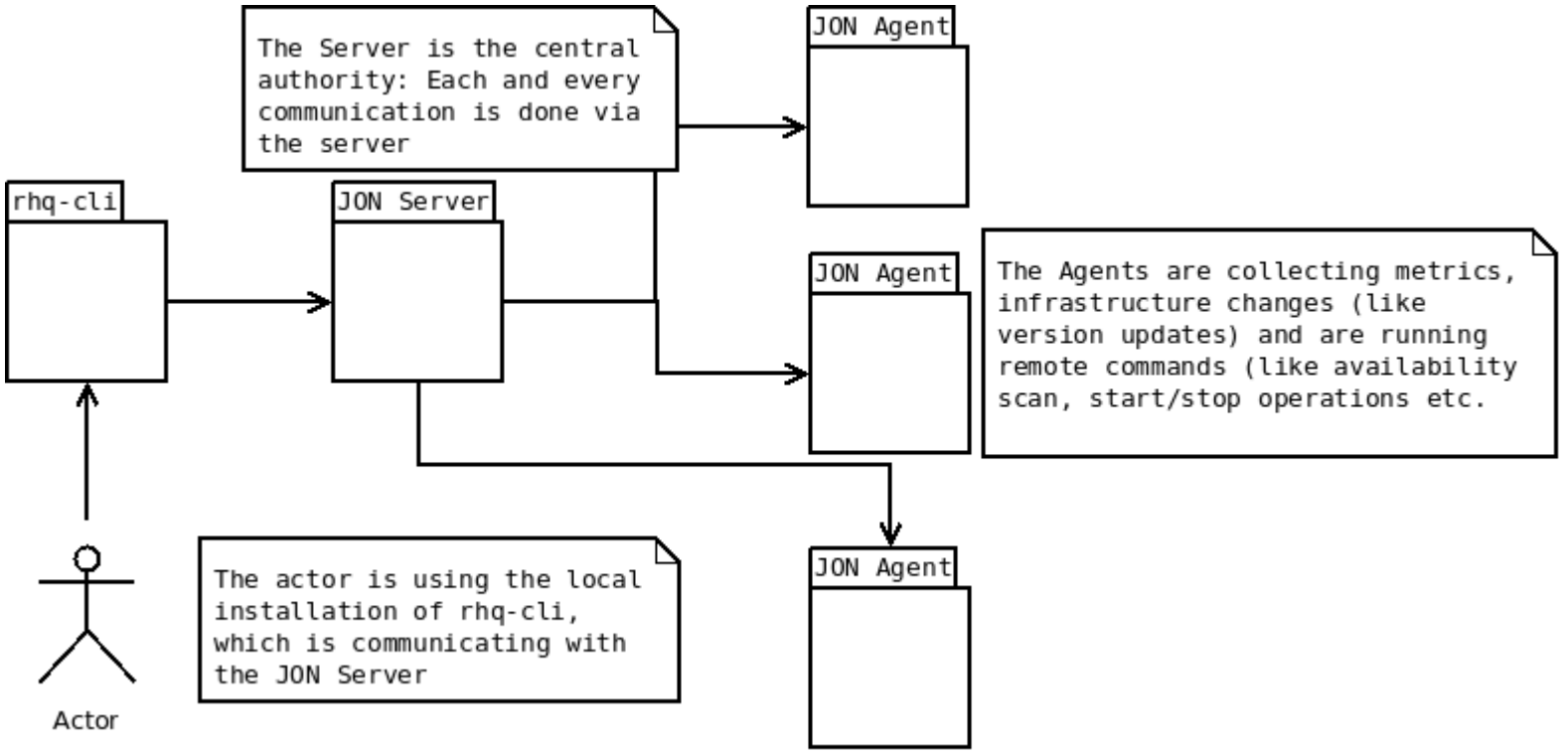
The solution for the direct bank in Germany

- As said, the customer needs to embed the JBoss infrastructure into their existing processes.
  - Deploy an update of the Application to all 50-100 nodes just via CLI
  - Get a list of all instances with IP, host name & version via CLI
  - Get a list of all instances with health status (UP/Down, memory, threads, system load) via CLI
- During the presales phase we said this is possible with JON
  - Now we had to proof it

Using the CLI of JON 2.3 and up

- At the beginning it was quite hard to find documentation of the CLI
  - No docs, just source code
- I was pointed to some test cases the JON dev team uses to ensure a proper working of the CLI
- Thanks to this hint, I was able to start developing a set of scripts
- So I was starting to create a simple deployment script, which looks ugly, but did what I want
- As Red Hat is an open source company, I have decided to write a public WIKI entry for this

CLI Architecture



- The Actor has to install the rhq-cli binary together with a Java 6 JVM
- The Actor can create JavaScript based code snippets for the CLI
- The rhq-cli is executing the script in the context of JON Server
- The JON Agents are communicating with the JON Server bi directionally

Using CLI to do group deployments

- To start with something, we start with command line parsing. As with plain Java, we get a string array called “args” which contains all parameters
- With thanks to Java, we are easily able to parse and to check if the file really exists
- At the end we need to use the API to check if the given resource group really exists. Also, we need to fetch some relationships
- Some more checks and we now have a valid group name with a list of resources we can check for EAP instances

```
if( args.length < 2 ) usage();

var fileName = args[0];
var groupName = args[1];
```

```
// check that the file exists and that we can read it
var file = new java.io.File(fileName);

if( !file.exists() ) {
    println(fileName + " does not exist!");
    usage();
}

if( !file.canRead() ) {
    println(fileName + " can't be read!");
    usage();
}
```

```
// find resource group
var rgc = new ResourceGroupCriteria();
rgc.addFilterName(groupName);
rgc.fetchExplicitResources(true);
var groupList = ResourceGroupManager.findResourceGroupsByCriteria(rgc);
```

```
if( groupList == null || groupList.size() != 1 ) {
    println("Can't find a resource group named " + groupName);
    usage();
}

var group = groupList.get(0);

println("  Found group: " + group.name );
println("  Group ID   : " + group.id );
println("  Description: " + group.description);
```


Redeploying an app to a group

- We now iterate through the list of explicit resources of the group to find all matching server entries. In this case we are looking for “JBossAS Server” resources
- Now iterate over the list of child resources of the JBossAS Server and try to find a child which matches the name of the package to be deployed
- If the child resource exists, we can first retrieve the original content for backup purposes
- Then we can just update the backing content with the file name given at script start

```
for( i in resourcesArray ) {
    var res = resourcesArray[i];
    var resType = res.resourceType.name;
    println("  Found resource " + res.name + " of type " + r

    if( resType != "JBossAS Server") {
        println("    ---> Resource not of required type. Exi
        usage();
    }

    // get server resource to start/stop it and to redeploy
    var server = ProxyFactory.getResource(res.id);
}
```

```
    var children = server.children;
    for( c in children ) {
        var child = children[c];

        if( child.name == packageName ) {
        }
    }
```

```
println("    download old app to /tmp");
child.retrieveBackingContent("/tmp/" + packageName +
```

```
println("    uploading new application code");
child.updateBackingContent(fileName);
```


Using group control

```
[wanja@tolnedra groupcontrol]$ ./groupcontrol.sh
  Usage ./groupcontrol.sh:
  Use this tool to control most group related tasks with a simple script.
  ---------------------------------------------------------------------
    deploy <path-to-app> <groupName> to deploy an app to group
    create <groupName> to create a new group
    delete <groupName> to delete an existing group
    status <groupName> to list all resources in a group and their availability
    start  <groupName> to start all instances in a group
    stop   <groupName> to stop all instances in a group
    add    <eap-name> <groupName> to add an instance to a group
    remove <eap-name> <groupName> to remove an instance from a group
    avail [groupName] Issue the AVAIL command on all agents to send the availability report to JON


  Commands used for whole environment:
    list    Just print all JBossAS Server instances available in repository with version and availability
    list-groups Like list command but list all compatible groups
```

- After the success of the group deployment script, I've decided to do something more
- As you can see here, I've created a simple bash wrapper script which takes the arguments and calls RHQ command line interface with the appropriate script to control JON
- There are scripts for creating / deleting a group, adding / removing resources from / to a group and some other specific features
- The goal for the groupcontrol script set was to have something like an extended “top” for the command line to use. Some admins do like this

Running the scripts / starting all servers of a group

```
[wanja@tolnedra groupcontrol]$ ./groupcontrol.sh start ProductiveCluster
Remote server version is: 1.3.0.GA(5192)
Login successful
About to start all EAP instances of the group with name ProductiveCluster...
  Found resource tolnedra.belgariad JBoss EAP 4.3.0.GA_CP03 node2 (192.168.100.51:1099) of type JBossAS Server and ID 10451
    Starting tolnedra.belgariad JBoss EAP 4.3.0.GA_CP03 node2 (192.168.100.51:1099)....
Invoking operation start
   --> Caught JavaException: java.lang.NullPointerException: null
  Found resource tolnedra.belgariad JBoss EAP 4.3.0.GA_CP03 node1 (192.168.100.50:1099) of type JBossAS Server and ID 10452
    Starting tolnedra.belgariad JBoss EAP 4.3.0.GA_CP03 node1 (192.168.100.50:1099)....
Invoking operation start
   --> Caught JavaException: java.lang.NullPointerException: null
Done!
```

- Before you can start a deployment, you need to make sure to have your servers up and running
- Use groupcontrol start to start all server instances for a given group name
- The script then iterates through the list of explicit resources of the group and issues the start() command for each
- Just ignore the JavaException log entry above. Unfortunately, the operation causes a NPE although the operation finishes successfully

Executing AVAIL command

```
[wanja@tolnedra groupcontrol]$ ./groupcontrol.sh avail
Remote server version is: 1.3.0.GA(5192)
Login successful
Scanning all RHQ Agent instances
Executing on 1 agents
  executing availability scan on agent
    -> wanja.laptop RHQ Agent / 10002
ResourceOperationSchedule: resource=[Resource[id=10002, type=RHQ Ag
-1266246733267], job-group=[rhq-resource-10002], operation-name=[ex

Done!
```

- After starting all instances you either need to wait until the agent of JON has detected that the resources are running, or you simply issue the avail script
- This script is using the OperationManager to schedule the “executeAvailabilityScan” operation now
- It either searches for all agents of a specific group or for all agents in the repository, which is not recommended if you have more than a demo setup

```
62 // for each agent, issue executeAvailabilityScan command
63 if( agents != null && agents.length != 0) {
64   println("Executing on " + agents.length + " agents ");
65
66   for( a in agents ) {
67     var agent = agents[a];
68
69     println("  executing availability scan on agent" );
70     println("    -> " + agent.name + " / " + agent.id);
71     var config = new Configuration();
72     config.put(new PropertySimple("changesOnly", "true") )
73
74     var ros = OperationManager.scheduleResourceOperation(
75       agent.id,
76       "executeAvailabilityScan",
77       0,   // delay
78       1,   // repeatInterval
79       0,   // repeat Count
80       10000000,  // timeOut
81       config,    // config
82       "test from cli" // description
83     );
84
85     println(ros);
86     println("");
87   }
88 }
```


Having something like “top”

```
[wanja@tolnedra groupcontrol]$ ./groupcontrol.sh list
Remote server version is: 1.3.0.GA(5192)
Login successful
Name                                  Version         HostName            UsrLoad Listen Addr          A Last UP-Time      Act.Thr Tot. MB Used MB % Free
-------------------------------------------------------------------------------------------------------------------------------------------------------
RHQ Server,  JBoss AS 4.2.3.GA default   4.2.3.GA        tolnedra.belgariad     40.0%         0.0.0.0:2099 U 02/15/10 15:37:25      65  967.63  217.36  77.54
JBoss EAP 4.3.0.GA_CP03 node2            4.3.0.GA_CP03   tolnedra.belgariad     60.0%  192.168.100.51:1099 U 02/15/10 15:42:29     114  220.56   87.59  60.29
JBoss EAP 4.3.0.GA_CP03 node1            4.3.0.GA_CP03   tolnedra.belgariad     80.0%  192.168.100.50:1099 U 02/15/10 15:42:31     108  230.19  133.64  41.94
```

- After you've started your servers and you've executed the avail command, you should have a look at your environment
- Are they Up or Down? What is the User Load? What about server system load?
- You can use the list command to get a detailed list of your environment
- How do you get those metrics?

Same for groups: list-groups

```
[wanja@tolnedra groupcontrol]$ ./groupcontrol.sh list-groups
Remote server version is: 1.3.0.GA(5192)
Login successful
Name                          Location        Category   ExplRes ImplRes Act.Thr Max MB  Tot.MB  Used MB % Free
------------------------------------------------------------------------------------------------------------------
ProductiveCluster                             compatible       2      96     112    0.00  237.66   88.69   62.68
test-group2                   Datacenter #1   compatible       0       0       0    0.00    0.00    0.00    0.00
```

- Using groupcontrol to get a list of all your groups
- How many resources are in there?
  - Are they explicit (chosen EAP instances)
  - Or implicit (child resources of each EAP instance)
- Other metrics like
  - Active threads
  - Memory consumption

Retrieving Metric data

```
 90 function getLongMetric( resId, metricName ) {
 91   var metricDefId;
 92   if( serverMetrics.get(metricName) == null ) {
 93     var mdc = MeasurementDefinitionCriteria();
 94     mdc.addFilterResourceTypeName("JBossAS Server");
 95     mdc.addFilterDisplayName(metricName);
 96     var mdefs = MeasurementDefinitionManager.
 97               findMeasurementDefinitionsByCriteria(mdc);
 98
 99     if( mdefs == null || mdefs.size() == 0 ) {
100       println("  --> could not find metric " + metricName )
101       return 0;
102     }
103     else {
104       var mdef = mdefs.get(0);
105       metricDefId = mdef.id;
106       serverMetrics.put(metricName, metricDefId);
107     }
108   }
109   else {
110     metricDefId = serverMetrics.get(metricName);
111   }
112
113   if( metricDefId != null ) {
114     var metrics = MeasurementDataManager.findLiveData(
115       resId, [metricDefId]
116     );
117     if( metrics == null || metrics.size() == 0 ) {
118       return 0;
119     }
120     else {
121       var metric = metrics.toArray()[0];
122       if( metric != null && metric.value != null ) {
123         return metric.value.longValue();
124       }
125       else {
126         return 0;
127       }
128     }
129   }
130 }
```

- The heart of the list and list-groups command is the getLongMetric function
- This one takes the resourceId and the name of the metric and returns the live value of the metric
- You first need to get the measurement definition of the given metric
- With the MeasurementDefinition you can use the MeasurementDefinitionManager to actually retrieve the value
- Note, if you use the method findLiveData() the JON server is actually asking the resource for a current value.
- That means the resource must be up and running
- There are other methods to retrieve historical data of a specific resource which can be used to get the latest possible value

Getting a quick status summary

```
[wanja@tolnedra groupcontrol]$ ./groupcontrol.sh status ProductiveCl
Remote server version is: 1.3.0.GA(5192)
Login successful
  tolnedra.belgariad JBoss EAP 4.3.0.GA_CP03 node2 (192.168.100.51:1
    - Availability: UP
    - Started     : 15 Feb 2010 14:42:29 GMT
    - JVM Version : 1.6.0_17

  tolnedra.belgariad JBoss EAP 4.3.0.GA_CP03 node1 (192.168.100.50:1
    - Availability: UP
    - Started     : 15 Feb 2010 14:42:31 GMT
    - JVM Version : 1.6.0_17

Done!
```

- Sometimes you just want to have a quick status summary of a specific group
- The script behind it again just iterates over all resources of a given group
- It then tries to get some extended information regarding JVM and availability

```
41  for( i in resourcesArray ) {
42    var res = resourcesArray[i];
43    var resType = res.resourceType.name;
44
45    if( resType != "JBossAS Server") {
46      println("    ---> Resource not of required typ
47      usage();
48    }
49
50    // get server resource to start/stop it and to
51    // redeploy application
52    var server = ProxyFactory.getResource(res.id);
53    var avail  = AvailabilityManager.
54      getCurrentAvailabilityForResource(server.id);
55
56    // get the jvm
57    var rc = new ResourceCriteria();
58    rc.addFilterResourceTypeName("JBoss AS JVM");
59    rc.addFilterParentResourceId(server.id);
60    var jvm = ResourceManager.
61      findResourcesByCriteria(rc).get(0);
62
63
64    println("  " + server.name );
65    println("    - Availability: " + avail.availabil
66    println("    - Started     : " + avail.startTime
67    println("    - JVM Version : " + jvm.version );
```


Conclusion of using the CLI

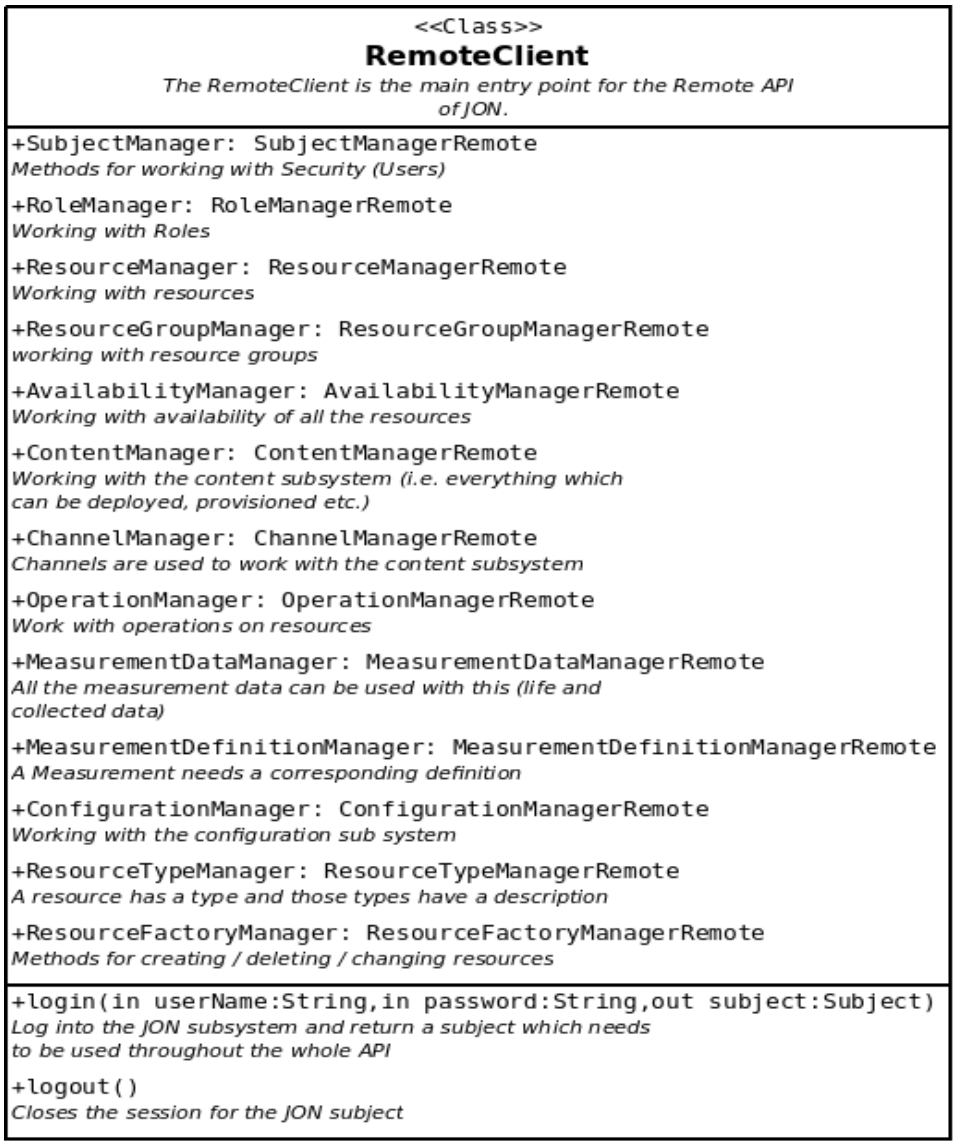
- It was hard at the beginning
- But after a while you got confident with the API
- And you can do great things
- The direct bank prospect was happy to see that it really works and that JON is so flexible
- The prospect is now a customer and migrates ~200 CPUs from Weblogic to JBossEAP

The Solution for the Health Card provider in Germany

The solution for the Health Card Provider

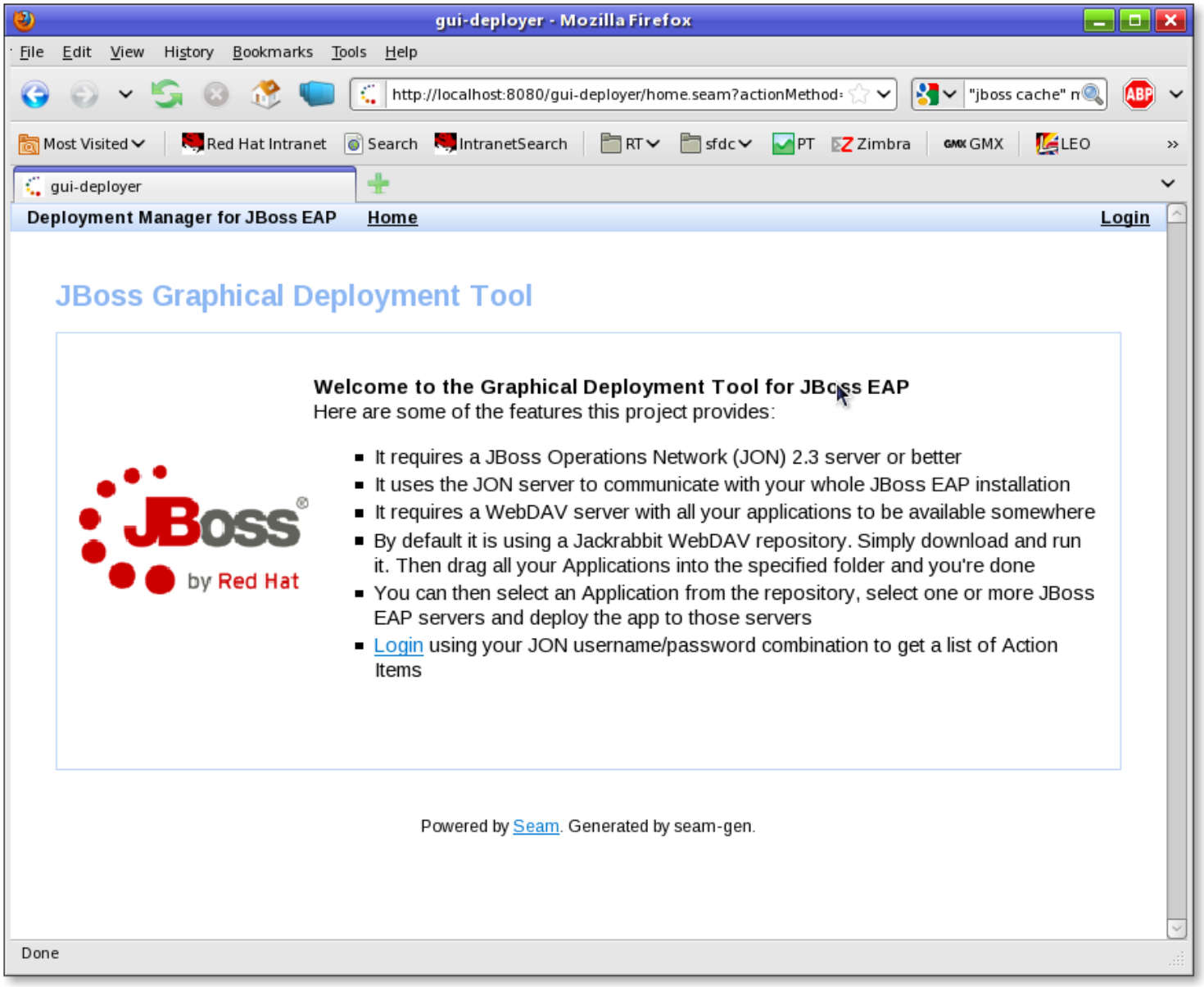
- Embed some functionality of JON into a custom GUI
- The solution has to use the JON security (LDAP)
- At least have a simple 20.000 feet bird view of application server availability
- And the possibility to deploy the application to a selectable set of application servers
  - The source for applications has to be some WebDAV storage

The Remote API



- The RemoteClient is the central point of all communication with the JON API.
- The easiest part is, call the constructor with hostName, portNumber and evtl. Change the transport
- Then do a login() call with userName and password
- Use the API via the various managers
- Do a logout() if you don't need the API any longer

The GUI deployer



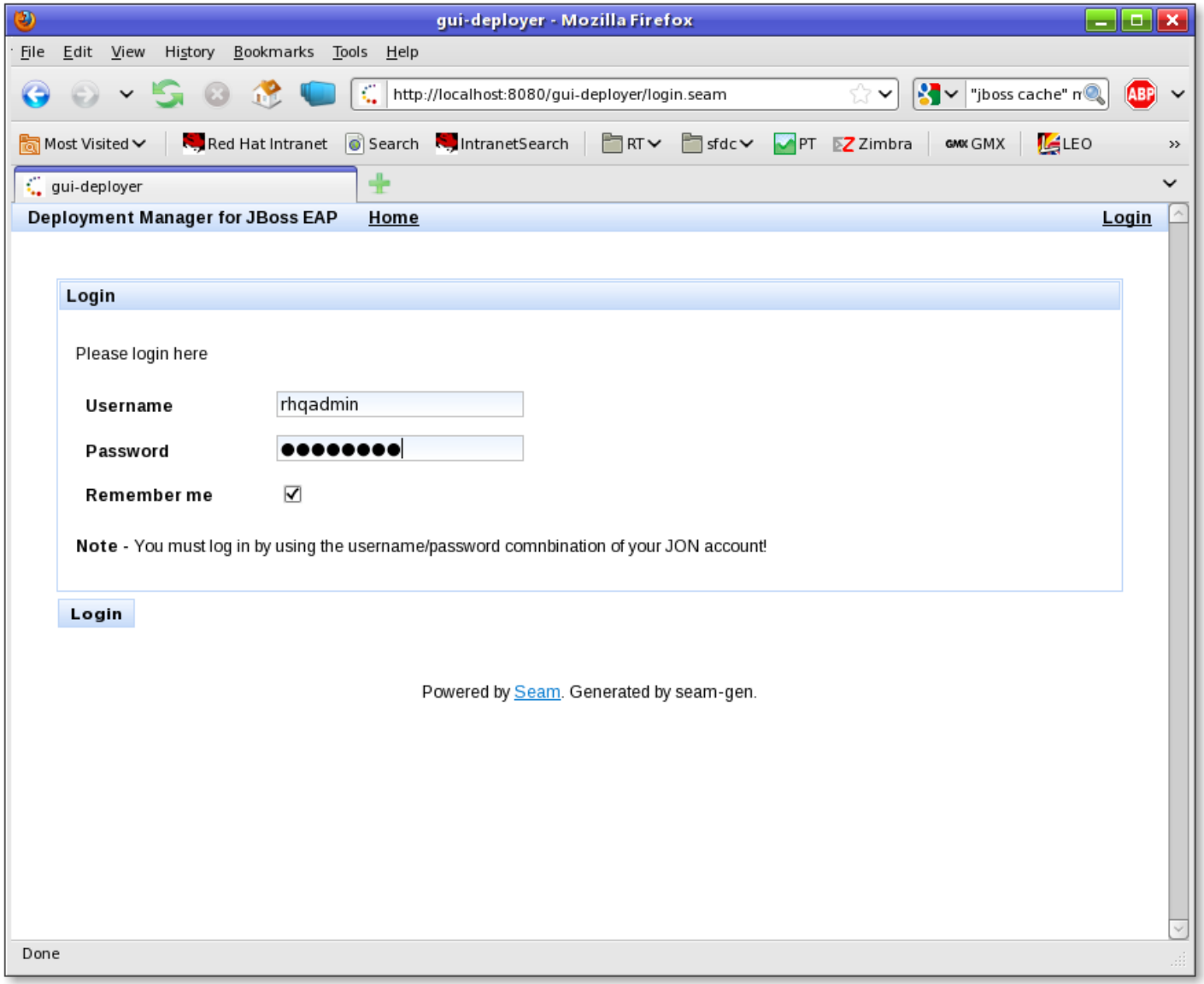
- The graphical GUI deployer is a simple Seam application
- It uses the JON Remote API
- It uses the JON security module
- It is just a simple WAR file to be deployed somewhere in your network
- This prototype uses Jackrabbit as a WebDAV server
- It uses webdavclient4j and apache commons vfs to access the WebDAV server

Deployer: How it is done

- Simple Seam application
  - Based on Seam 2.0 (FP 01 for EAP 4.3)
- Initial work was done by seam-gen
  - No connection to a database
- Base component for communication with JON is called RHQCommunication and is a facade over RHQ class RemoteClient

```
public RHQCommunication(String userName, String password, String hostName, int
    super();
    this.userName = userName;
    this.password = password;
    this.hostName = hostName;
    this.port = port;

    client = new RemoteClient(hostName, port);
}
```


Logging into JON



- Here we have an Authenticator which connects to the JON Remote API
- Within the Seam Authenticator, we use the RHQ RemoteClient class to get access to the JON subsystem

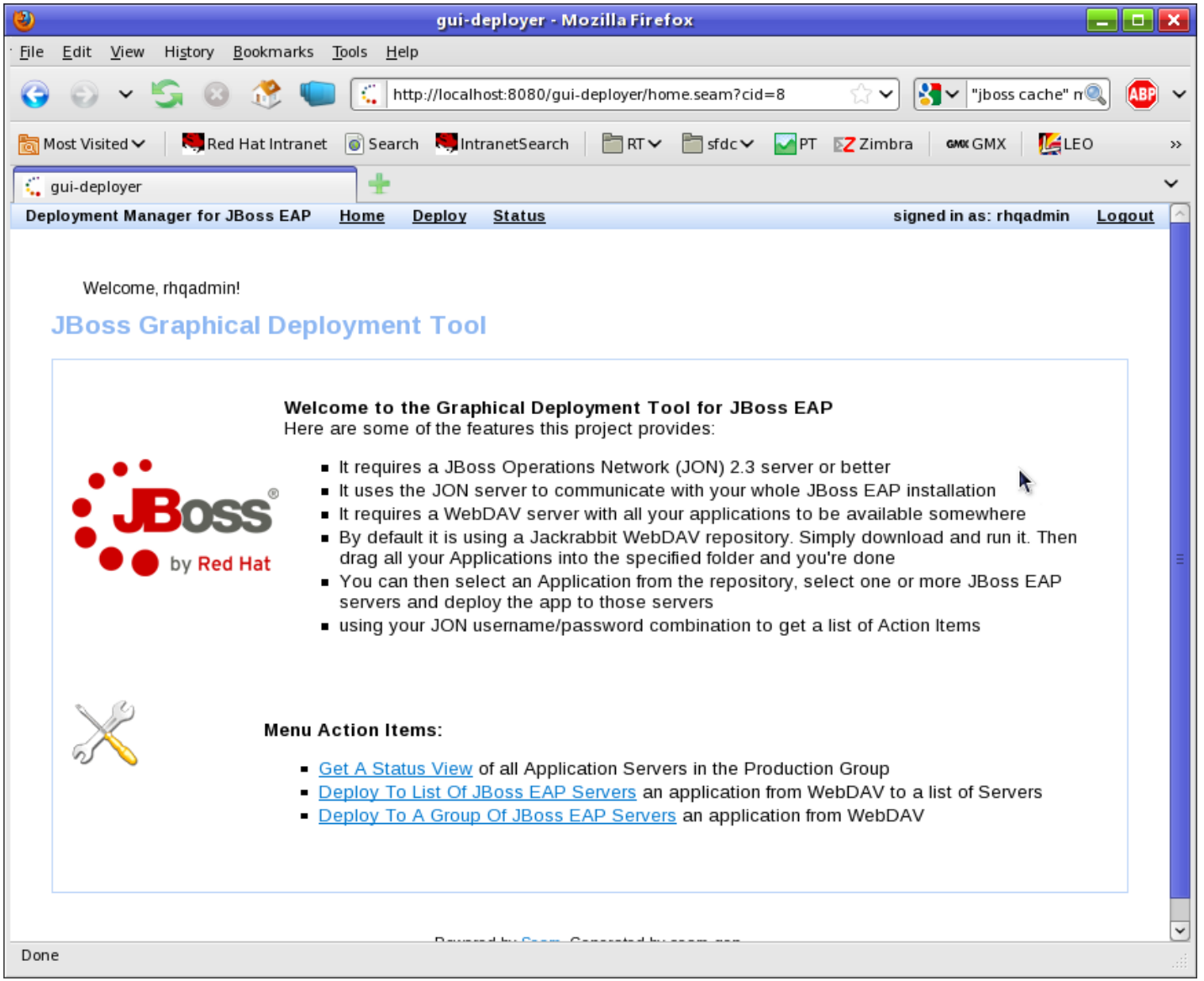
Login: How it is done

```
@Out("rhq")
RHQCommunication rhq;

public boolean authenticate()
{
    log.info("authenticating #0", identity.getUsername());
    //write your authentication logic here,
    //return true if the authentication was
    //successful, false otherwise
    identity.addRole("admin");

    rhq = new RHQCommunication(
            identity.getUsername(),
            identity.getPassword(),
            "localhost", // ok, needs to be configurable
            7080 // also...
    );
    try {
        rhq.login();
        return true;
    }
    catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        return false;
    }
```

```
public void login() throws Exception {
    // Log into JON
    subject = client.login(userName, password);

    // get remote managers for communication with JON
    subjectManager = client.getSubjectManagerRemote();
    resourceGroupManager = client.getResourceGroupManagerRemote();
    roleManager = client.getRoleManagerRemote();
    resourceManager = client.getResourceManagerRemote();
    availabilityManager = client.getAvailabilityManagerRemote();
    channelManager = client.getChannelManagerRemote();
    contentManager = client.getContentManagerRemote();
    resourceTypeManager = client.getResourceTypeManagerRemote();
    operationManager = client.getOperationManagerRemote();
    measurementDataManager = client.getMeasurementDataManagerRemote();
    measurementDefinitionManager = client.getMeasurementDefinitionManager
    resourceFactoryManager = client.getResourceFactoryManagerRemote();
    configurationManager = client.getConfigurationManagerRemote();

    if( resourceManager == null ) {
        System.out.println("resource manager is null");
    }
}
```

- Once the user clicks on Login, the Authenticator component of Seam is doing the delegation work to JON
- In RHQCommunication, we just retrieve all the managers which we need to communicate with JON after calling login()

The main page after log in

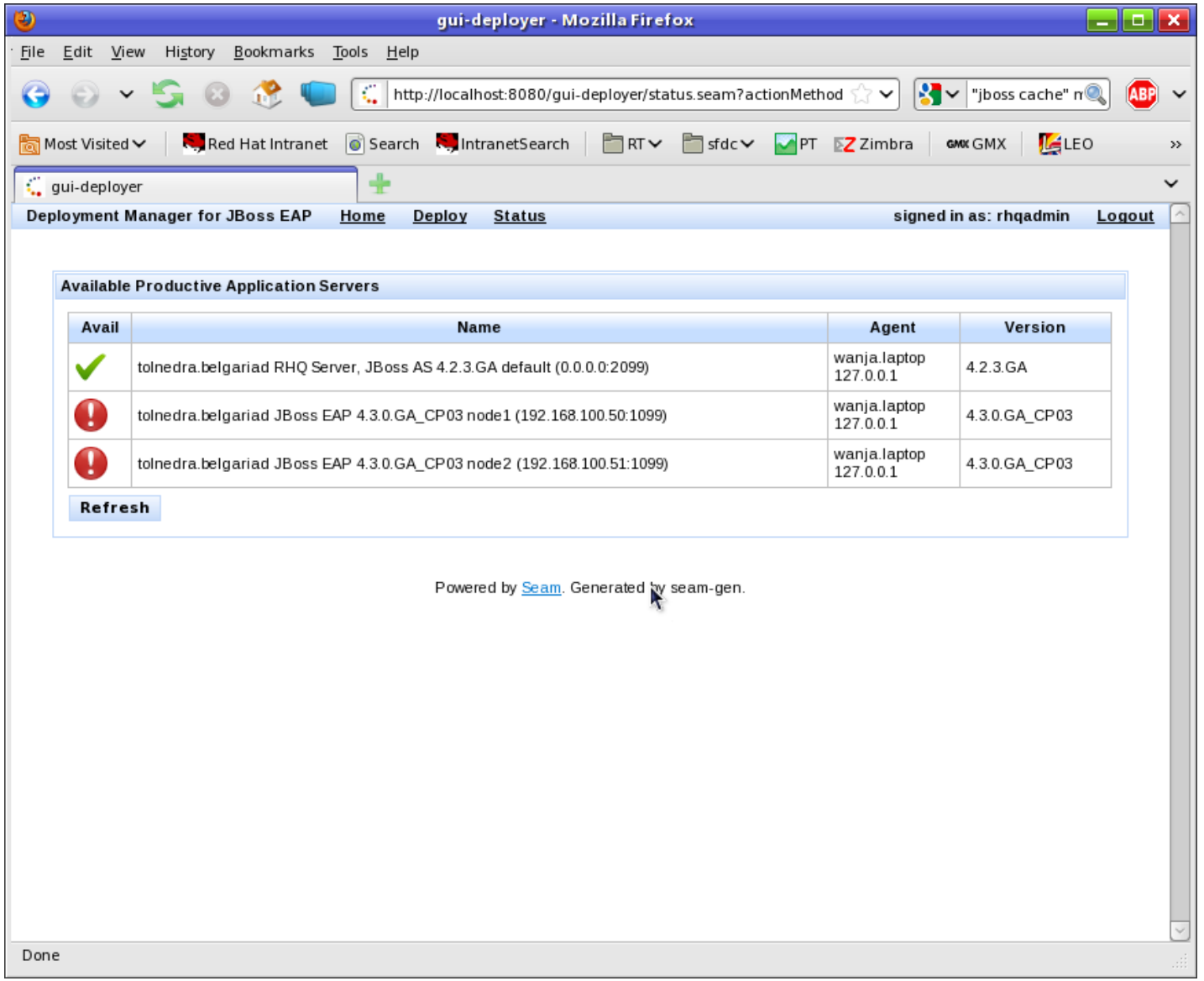


- After logging into the tool, you get some action items
- Get A Status View: This is a simple availability view
- Deploy To EAP: Select an application from a WebDAV repository and deploy it to a set of EAP instances
- Group Deployment: Select an application from a WebDAV repository and deploy it to a group

Reading Resources from JON

```
public List<ServerInstance> retrieveResources() {

    if( this.result != null ) {
        return this.result;
    }
    else {
        ResourceManagerRemote rm = rhq.getResourceManager();
        AvailabilityManagerRemote am = rhq.getAvailabilityManager();

        ResourceCriteria rc = new ResourceCriteria();
        rc.fetchAgent(true);
        rc.fetchParentResource(true);
        rc.fetchChildResources(true);
        rc.addFilterResourceTypeName("JBossAS Server");
        rc.addSortAgentName(PageOrdering.ASC);
        rc.addSortVersion(PageOrdering.ASC);
        rc.addSortName(PageOrdering.ASC);
        PageList<Resource> resources = rm.findResourcesByCriteria(rhq.getSubject(), rc);

        if( resources != null && !resources.isEmpty() ) {

            this.result = new ArrayList<ServerInstance>();
            for( Resource r : resources) {
                ServerInstance si = new ServerInstance();
                Availability av = am.getCurrentAvailabilityForResource(rhq.getSubject(), r.getId());
                si.setAgent(r.getAgent());
                si.setHost(r.getParentResource());
                si.setServer(r);
                si.setCurrentAvailability(av);
                result.add(si);
            }
        }
    }
    return result;
}
```


The Status View

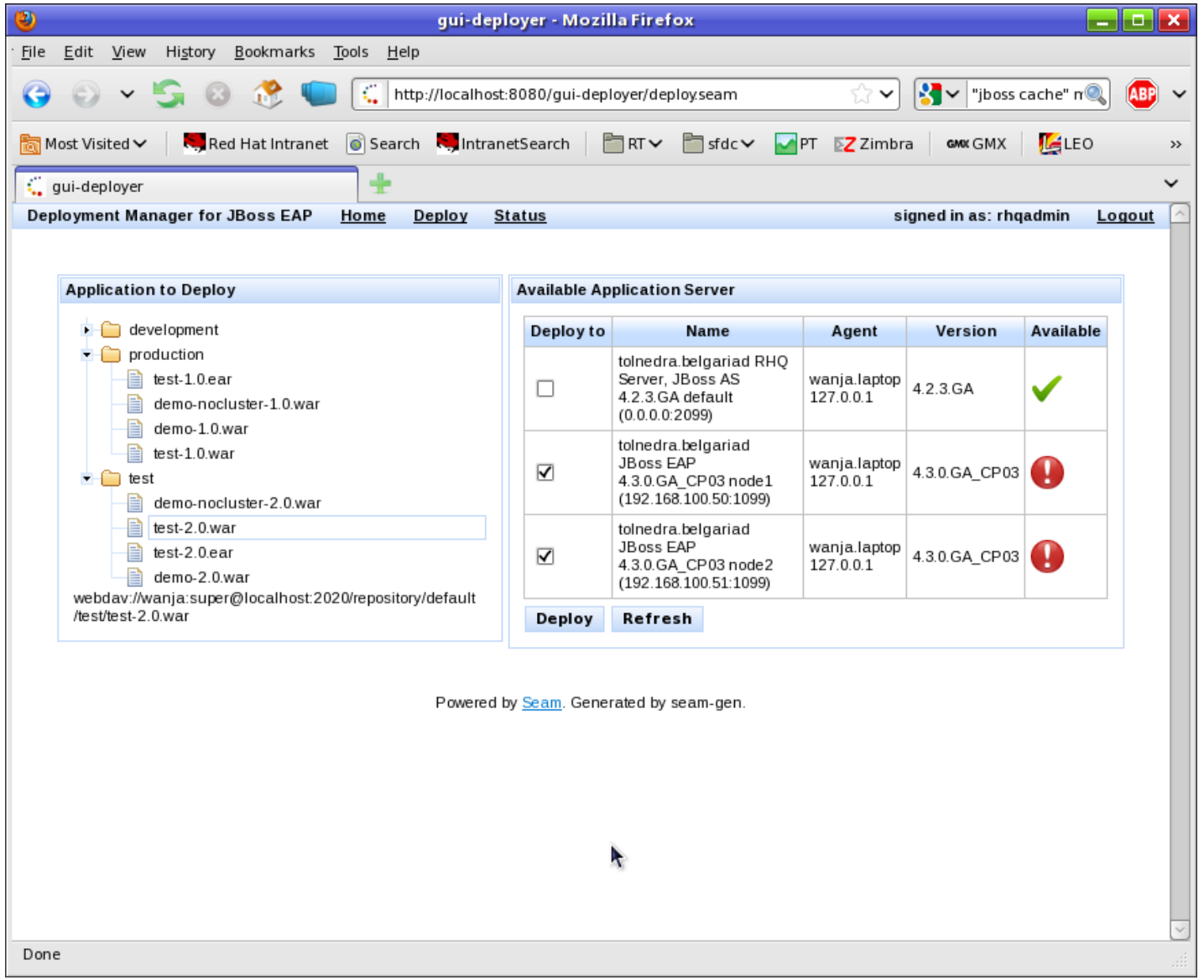


- This is a simplified view of the deploy page.
- It contains an availability marker (green / red)
- The name of the resources
- The name of the Agent
- And the version of the EAP instance
- This view is restricted to users with a datacenter role

Status View: How it is done

```
<h:form>
    <rich:panel>
        <f:facet name="header">Available Productive Application Servers</f:facet>

        <rich:dataTable value="#{resourceReader.retrieveResources()}" var="res" >
            <rich:column>
                <f:facet name="header">Avail</f:facet>
                <h:graphicImage value="/img/availability_green_24.png"
                    rendered="#{res.currentAvailability.availabilityType.name.equals('UP')}" />
                <h:graphicImage value="/img/availability_red_24.png"
                    rendered="#{! res.currentAvailability.availabilityType.name.equals('UP')}" />
            </rich:column>
            <rich:column>
                <f:facet name="header">Name</f:facet>
                #{res.server.name}
            </rich:column>

            <rich:column>
                <f:facet name="header">Agent</f:facet>
                #{res.agent.name}<br />
                #{res.agent.address}
            </rich:column>

            <rich:column>
                <f:facet name="header">Version</f:facet>
                #{res.server.version}
            </rich:column>

        </rich:dataTable>


        <s:button value="Refresh" action="#{resourceReader.refresh()}" />
    </rich:panel>

</h:form>
```


The Deploy View



- Within the deploy view, you can select one application from the tree on the left side. This tree is build from a WebDAV repository
- The right side contains a list of all existing EAP instances in the JON repository
- This page is restricted for specific users
- By using the JON security, you can use the fine grained right management of it. This tool is automatically using it

Deployment: How it works / Updating content

```
/**
 * Updates the backing content of given resource
 * @param pp package parser information regarding version, name etc. of file
 * @param c child resource (WAR or EAR)
 * @param pt package tpye instance
 * @param fileBytes the binary file
 */
private void updateContent(PackageParser pp, Resource c, byte[] fileBytes) {
    log.info("Found package " + c.getName() + ". Updating package content!");
    InstalledPackage ip = rhq.getContentManager().getBackingPackageForResource(rhq.getSubject(),c.getId());
    String oldVersion = ip.getPackageVersion().getVersion();
    String newVersion = pp.getVersion();

    if( oldVersion != null && !oldVersion.isEmpty() ) {
        if( oldVersion.equalsIgnoreCase(pp.getVersion())) {
            log.info("Package " + c.getName() + " of Version " + oldVersion + " already exists. Calculating new Version.");

            String[] parts = oldVersion.split("[^a-zA-Z0-9]");
            String lastPart = parts[parts.length-1];

            try {
                int lastNumber = Integer.parseInt(lastPart);
                newVersion = oldVersion.substring(0, oldVersion.length() - lastPart.length()) + (lastNumber + 1);
            }
            catch (NumberFormatException nfe) {
                newVersion = oldVersion + ".1";
            }
        }
    }

    log.info("Creating package: " + ip.getPackageVersion().getGeneralPackage().getName() + "-" + newVersion);
    // create new package version
    PackageVersion pv = rhq.getContentManager().createPackageVersion(
            rhq.getSubject(),
            ip.getPackageVersion().getGeneralPackage().getName(),
            ip.getPackageVersion().getGeneralPackage().getPackageType().getId(),
            newVersion,
            ip.getPackageVersion().getArchitecture().getId(),
            fileBytes
    );

    log.info("Deploying package...");
    rhq.getContentManager().deployPackages(rhq.getSubject(), new int[] {c.getId()}, new int[] {pv.getId()});
}
```


Deployment: How it works / Creating new content

```
/**
 * deploy a new package of given type to resource res
 * @param pp package parser to get base name, version etc.
 * @param packageName name of the package to deploy
 * @param fileBytes the binary file bytes
 * @param fileType file type (war|ear)
 * @param res parent resource to deploy to
 */
private void deployNewPackage(PackageParser pp, byte[] fileBytes, String fileType, Resource res) {
    String packageName = pp.getPackageName();
    log.info("Package with name " + pp.getPackageName() + " does not exist on " + res.getName());
    ResourceType rt = rhq.getResourceTypeManager().getResourceTypeByNameAndPlugin(rhq.getSubject(), fileType, "JBossAS");
    Configuration deployConfig = new Configuration();
    deployConfig.put( new PropertySimple("deployDirectory", "deploy"));
    deployConfig.put( new PropertySimple("deployZipped", "true"));
    deployConfig.put( new PropertySimple("createBackup", "false"));

    // create & deploy new package
    rhq.getResourceFactoryManager().createPackageBackedResource(
            rhq.getSubject(),
            res.getId(), // parent resource
            rt.getId(), // resourceType
            null,
            null,
            packageName,
            pp.getVersion(),
            null,
            deployConfig, // deployconfig
            fileBytes
    );
}
```


Conclusion of using the RemoteAPI

- After playing with the CLI, it was not that hard to start with using the Remote API
  - Identical concepts
  - CLI is actually using the Remote API
- Using the Remote API helps you to enhance JON in a special way: Putting customized usability on top of JON
  - Simplifying deployment
  - Simplifying use of current / historical metrics
  - Customized reports

Conclusion of JON customization

- It is hard to start with customization
  - Source code is your documentation
  - JON is a powerful and flexible monitoring and management tool. --> The concepts must also be flexible
  - Query API is powerful, but not easy to understand
    - It helps if you know O/R Mappers and the concept of relationships
- Customization of JON is a task for a developer. A non developer (i.e. a typical administrator) is most likely not able to really customize it
- If you just use JON as is, you are loosing too much of the functionality of it.

References

- **Source code of RHQ test cases for CLI execution:**
  http://svn.rhq-project.org/repos/rhq/tags/RHQ_1_3_0_GA/modules/enterprise/remoting/scripts/src/test/script/org/rhq/enterprise/remoting/
- **JON 2.3 CLI Scripted Group Deployments Wiki entry**
  http://community.jboss.org/wiki/JON23ScriptedGroupDeploymentsUsingTheCLIAPI
- **JON 2.3 CLI GroupControl script set Wiki entry**
  http://community.jboss.org/wiki/JON23ControlGroupedServersViaTheCLI


Questions?
www.redhat.com