Red Hat JBoss BRMS Weight Watcher2 Demo

Use case: stateless CEP decision service

Stefano Picozzi
Sydney, Australia

spicozzi@emergitect.com

http://blog.emergile.com

Table of Contents

1 Introduction

Interested in a demo that showcases the Drools (6.3.0.FINAL) Decision Server? Then look here. The application is a stateless Decision Server with complex event processing (CEP) support based on a pseudo clock.

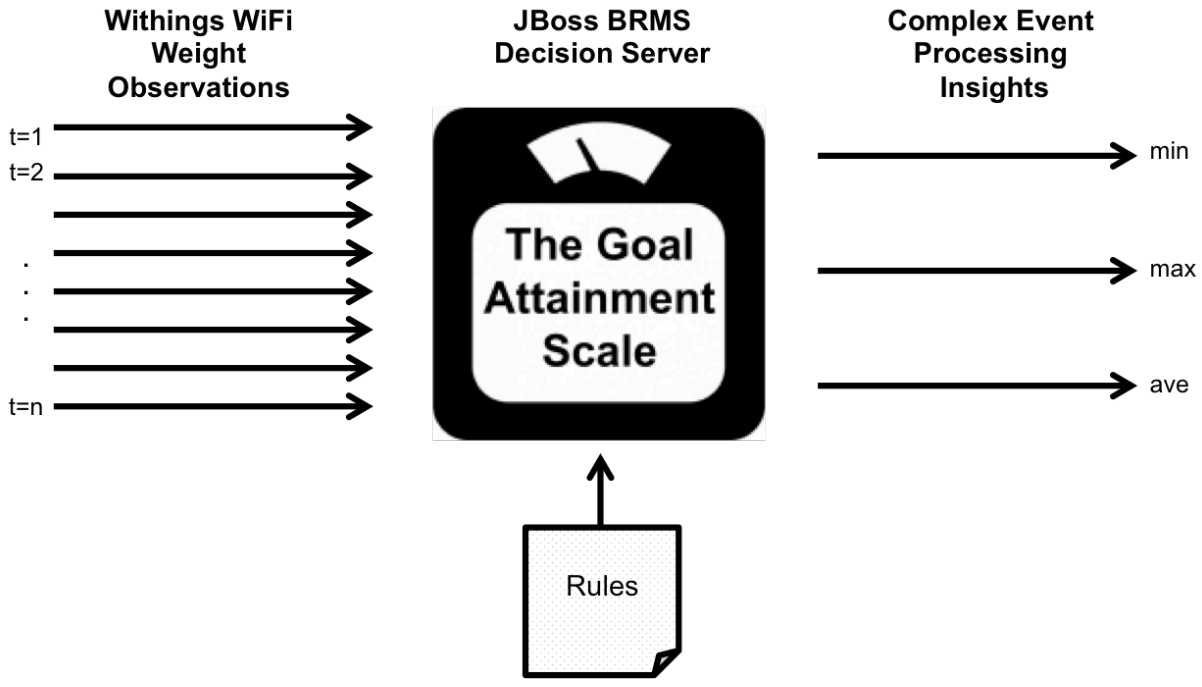
Note that by porting to the Drools 6.3.0.FINAL release, this demonstration application is ready for the upcoming OpenShift Enterprise V3.1 and JBoss BRMS 6.2 releases. This matters because OpenShift Enterprise V3.1 plans to introduce support for the JBoss BRMS 6.2 Decision Server. JBoss BRMS 6.2 will be based on a derivative of the Drools 6.3.0 code base.

An example use case demonstrated includes a (REST) client sending a time series of *facts* in the form of weight observations to the Decision Server. The Decision Server then reasons over the inputs to derive CEP insights such as average weight, least weight and weight change of a rolling time window. These insights are returned to the calling client as *facts*.

This is a facts-in-facts-out (FIFO) pattern using a standardized fact interface representation. This technique makes it easier for a simple thin client application such as SoapUI or RStudio to send request/response payloads to the Decision Server without knowledge of the underlying rules data model.

http://blog.emergile.com/2014/12/08/really-simple-rules-service/

Docker image based deployments are supported in both standalone and OpenShift V3 mode.



2 Setup

2.1 Image Download

The runtime artefacts for this demonstration application have all been packaged up according to the Docker container specification. The various Docker images can be downloaded as follows:

```
# The Decision Server as a Docker image
$ docker pull spicozzi/weightwatcher2

# A companion PHP website with test cases
$ docker pull spicozzi/testdrive2

# An optional RStudio Server image with R test cases
$ docker pull spicozzi/rstudio2

# An optional JBoss Drools workbench image used for rule changing use case
$ docker pull spicozzi/workbench2
```

2.2 Source Download

Repositories containing source code, content and support files related to the three images listed above can be inspected as per the GitHub links below.

```
https://github.com/StefanoPicozzi/weightwatcher2

https://github.com/StefanoPicozzi/testdrive2

https://github.com/StefanoPicozzi/rstudio2
```

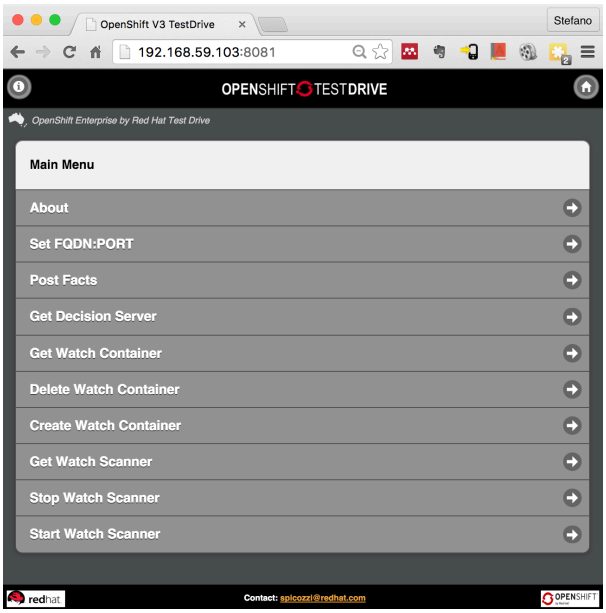
3 Running the Basic Demos

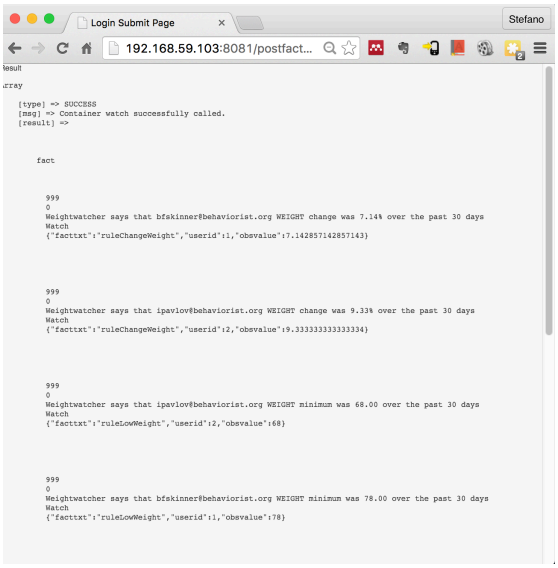
3.1 Launch Container Instances

```
# Launch 3 terminal windows and then run the container instances

# Find the IP_ADDRESS used by your instances, e.g.
$ boot2docker ip

# Terminal 1
$ docker run -it -p 81:80 spicozzi/testdrive2

# Terminal 2
$ docker run -it -p 8080:8080 spicozzi/weightwatcher2

# Terminal 3 (optional)
$ docker run -it -p 8087:8087 spicozzi/rstudio2
```

3.2 Companion Website Test Case

Open your browser at the IP_ADDRESS and port (8081) of the PHP companion application. Set the FQDN:PORT to point to your Decision Server instance. Then try the Post Facts menu option to verify that it is accepting requests correctly. Output should look similar to below in the side-by-side screen shots. The other choices refer to other optional Decision Server API features and test cases that are documented separately.





3.3 Simple cURL Test Case

Some simple cURL scripts have also been supplied to test the health of your configuration are located under the /tools/cURL subdirectory of the weightwatcher2 GitHub repository. These are similar to the test cases available at the companion website. Edit each script to change the http:// FQDN end point to reflect your deployment as follows:

```
$ cd <path-to-git-download>
$ cd tools/cURL

# Edit the post-facts.sh script and change the FQDN as necessary
$ ./post-facts.sh
# Check for 200 http response code
# Response payload should show list of JBoss BRMS CEP notifications
```

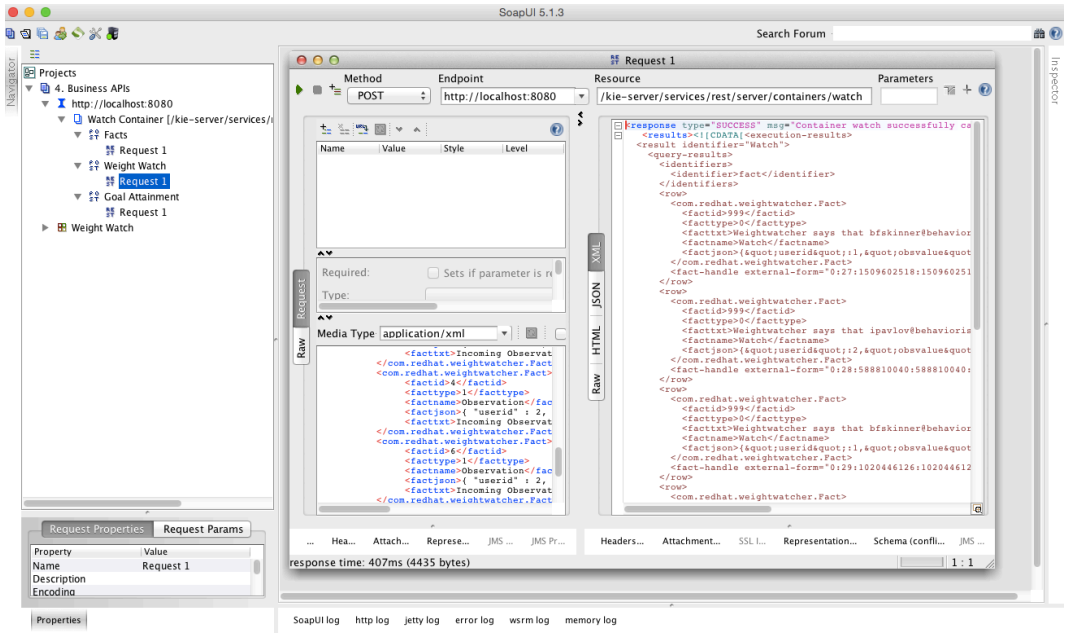
3.4 SoapUI Samples

This example assumes you have SoapUI installed on your workstation. Launch the SoapUI tools and then import the project with a name that includes the label "Business APs". The 3 supplied resources and REST POST requests are samples representing the following.

"Facts" shows a simple request in which a request payload of facts are *inserted* into the Decision Server knowledge and then a *query* is issued to verify this action has been successful.

The "Weight Watch" sample shows an invocation in which a set of facts containing weight measurements is sent to the Decision Server. CEP rules are then applied to derive insights as per the response payload. The request consists of facts representing Participant, Goal and Observation data records. The Participant records capture details of the user, Goal captures the Participant's target weight objectives and Observation records a time series of weight measurements. The response payload then returns a set of facts reporting minimum, maximum and weight change statistics over a sliding time window.

The "Goal Attainment" sample demonstrates a use case in which the Participant has elected to enter into a period of intermittent fasting, known as the Fast Diet http://thefastdiet.co.uk/. The GAS fact represents the Participant's number of fasting day goals over the week, described in ranges of worst through to best outcomes, refer http://en.wikipedia.org/wiki/Goal_Attainment_Scaling for details on the method. The Observation records then report back actual days of fasting in the previous weeks. The Decision Server then responds back with performance against goals. The GAS fact table is a candidate for remodelling using, e.g. a Guided Decision Tables.

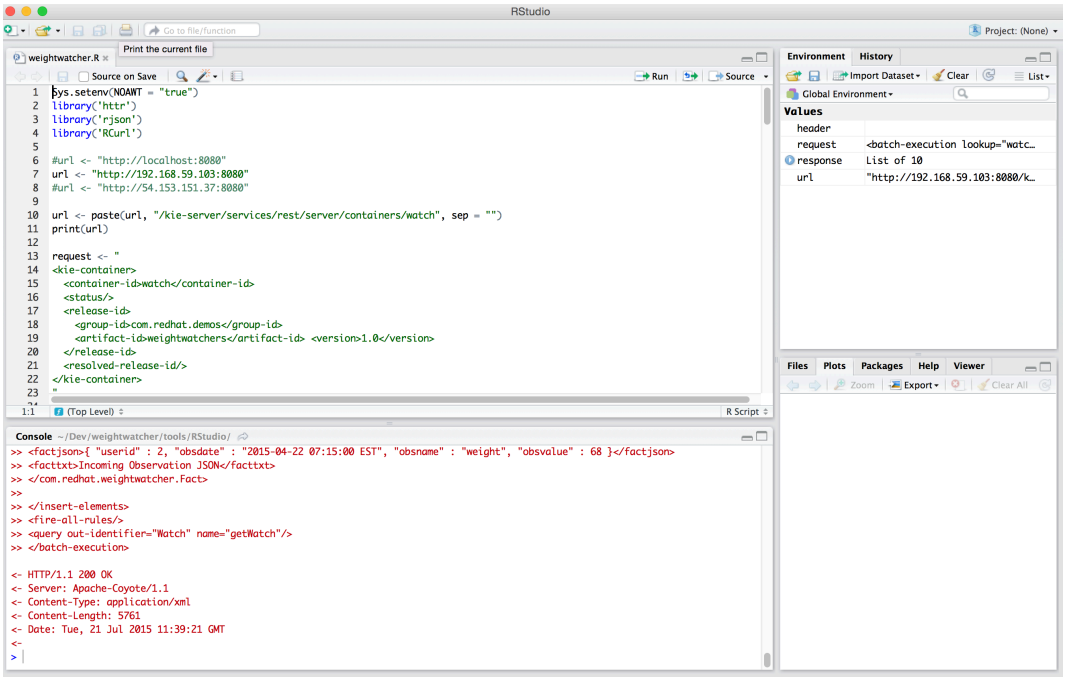


3.5 R using RStudio

An R script based sample showing how to interact with the Weight Watcher Services is located under the /tools/RStudio directory. You can reproduce this test case either using a local instance of RStudio on your workstation or by (preferred) using the prebuilt RStudio Server container image. Note that you may need to install a few missing packages expected of the supplied .R script if using a local RStudio installation. RStudio Server instructions are as follows:

```
# Find the IP_ADDRESS for your docker container instances
$ boot2docker ip

# Point your favourite browser at the RStudio Server cointainer
$ firefox http://IP_ADDRESS:8787

# Login as guest/guest
# Find and source weightwatcher.R
```

Edit the weightwatcher.R script url end point to reflect your environment, then source to run.



4 Running the Changing Rules Demo

This demonstration shows a use case in which rule are changed using the JBoss Business Central studio and then those changes reflected in the Decision Server. The basic steps are as follows:

1. Relaunch the JBoss Business Central container (workbench2) with the Maven repository mounted as an external volume (workbench2/m2)
2. Relaunch the Decision Server container (weightwatcher2) with the Maven repository mounted as an external volume (weightwatcher2/m2)
3. Launch the companion website PHP application (testdrive2)
4. From testdrive2, start the Decision Server scanner
5. From workbench2, make some changes to your rules, save them and then rebuild the deployment artefact
6. From the workbench/m2 file system, copy all the contents in com/redhat/demos/weightwatchers/1.0 to your clipboard
7. From the weightwatcher2/m2 file system, paste the clipboard contents to com/redhat/demos/weightwatchers/1.0
8. From testdrive2, post facts and verify that the changes have been applied

For steps 1 and 2, check the sample Docker launch scripts supplied at the GitHub repository for examples on how to approach the volume attachment requirement. These instructions assume familiarity with authoring rules using Business Central. More detailed documentation will follow later.

5 OpenShift V3.1 Beta

OpenShift V3 can load and run an arbitrary Docker image. The following instructions assume you have a working OpenShift V3 installation. We are going to create three OpenShift applications for the weightwatcher Decision Server, the companion website and another to host RStudio Server. Trying out the various basic test cases is just as documented previously for the basic Docker deployment scenario and so are not repeated here. The more advanced rule change use case will be documented later. Deploy. Enjoy!

5.1 OpenShift Environment Setup

```
# Pull down the image
$ su – root

# Set appropriate OpenShift security context constraints
$ oc edit scc restricted
> change runAsUser.Type to RunAsAny

# Create project that ensure pods land in certain nodes
$ oadm new-project weightwatcher --node-selector='region=primary'
$ oadm policy add-role-to-user admin <DEVUSER> -n weightwatcher
$ oadm new-project weightwatcher --display-name="Weight Watcher" --
admin=marina --node-selector='region=primary'

# Pull down my docker images on each user workload node
# for each user workload node <NODE>
$ ssh root@<NODE>
$ docker pull spicozzi/weightwatcher2
$ docker pull spicozzi/rstudio2
$ docker pull spicozzi/testdrive2
```

5.2 Create OpenShift Applications

```
# For instructions below replace <PARAMETER> to reflect your environment


# Login as an OpenShift V3.0 user <DEVUSER>
$ su - <DEVUSER>


# Authenticate to the OpenShift master at <MASTER-FQDN>:<PORT>
$ oc login -u <DEVUSER> --insecure-skip-tls-verify --
server=https://<MASTER-FQDN>:<PORT>


# Switch to the weightwatcher project
$ oc get projects
$ oc project weightwatcher


# Create an OpenShift application
$ oc new-app spicozzi/weightwatcher2
$ oc new-app spicozzi/testdrive2
$ oc new-app spicozzi/rstudio2


$ oc expose service weightwatcher2 --name=weightwatcher2-route --
hostname=weightwatcher2.<CLOUDAPPS-FQDN>
$ oc expose service rstudio2 --name=rstudio2-route --
hostname=rstudio2.<CLOUDAPPS-FQDN>
$ oc expose service testdrive2 –name2=testdrive2-route --
hostname=testdrive2.<CLOUDAPPS-FQDN>


$ oc describe pod weightwatcher2
$ oc describe pod rstudio2
$ oc describe pod testdrive2
```

A Appendix

A.1 Useful boot2docker Docker Commands

```
# Find <IP> of boot2docker virtual machine
$ boot2docker ip

# If you encounter strange problems while pull/push of images
$ boot2docker stop
$ boot2docker start

# List of docker images
$ docker images

# List of running docker containers showing <CONTAINER_ID>
$ docker ps –l

$ docker attach <CONTAINER_ID>

# Kill a running docker container with <CONTAINER_ID>
$ docker rm –f <CONTAINER_ID>

# Pull down a docker image
$ docker pull spicozzi/nginx

# Commit changes in a running Container as an image
# From another terminal get the <CONTAINER_ID>
$ docker ps -l
$ docker commit <CONTAINER_ID> spicozzi/nginx

# Assume you have a running container named weightwatcher1
$ docker logs -f weightwatcher1
$ docker rm -f weightwatcher1

# Remove all running containers
$ docker rm -f $(docker ps -aq)

# Remove all untagged images
$ docker rm -f $(docker images | grep "^<none>" | awk "{print $3}")
```

A.2 Useful OpenShift Commands

```
# To scale up replicas
$ oc scale dc weightwatcher2 --replicas=2

# Tail the log file for the created pod
$ oc get pods
$ oc logs -f <PODNAME>

# If you make any errors just delete the <PROJECT> and repeat
$ oc delete project <PROJECT>
```