Weight Watcher2 Demo

Stateless CEP Decision Server Use Case
For OpenShift Origin Vagrant VM
and JBoss Drools 6.3.0.FINAL
as tested on OS/X El Capitan

Stefano Picozzi
Sydney, Australia

spicozzi@emergitect.com

http://blog.emergile.com

Table of Contents

1 Introduction

1.1 Overview

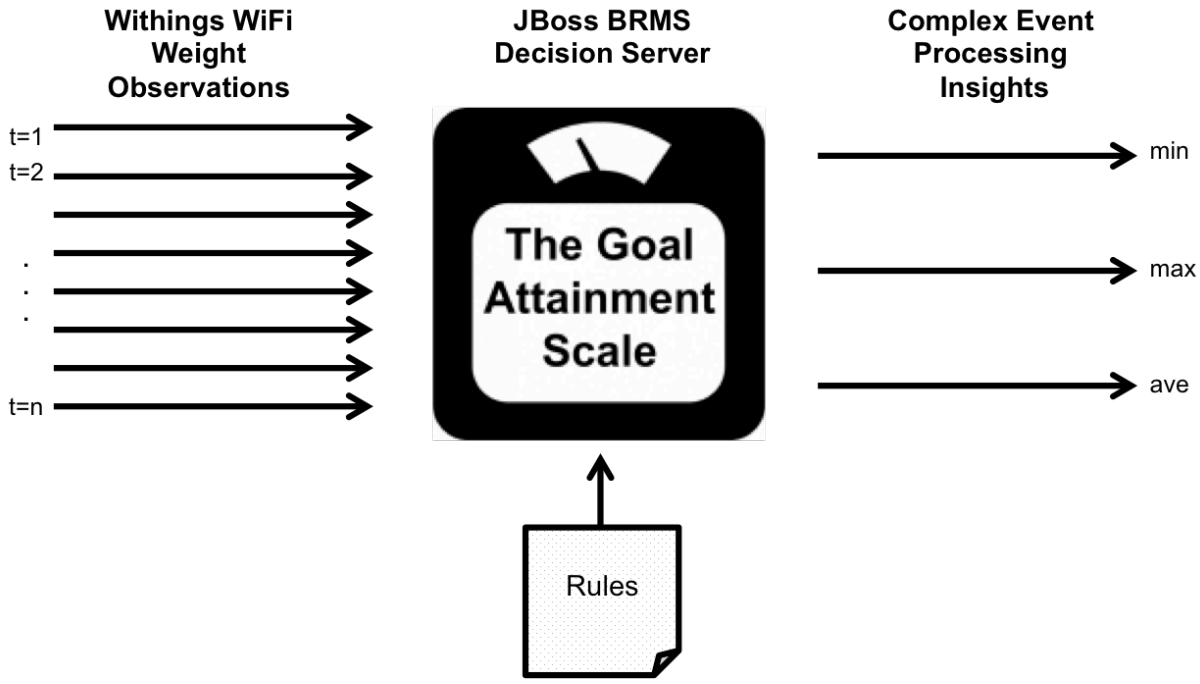
Interested in a demo that gets you started with OpenShift and showcases the Drools (6.3.0.FINAL) Decision Server? Then look here. The application is a stateless Decision Server with complex event processing (CEP) support based on a pseudo clock.

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 pHp, cURL, 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/

Note that by porting to the Drools 6.3.0.FINAL release, this demonstration application is ready for OpenShift Enterprise V3.1 and xPaaS support such as with JBoss BRMS 6.2. Docker image based deployments are supported in both OpenShift V3 mode and basic Docker.



1.2 Learning Goals

We are going to showcase some use cases using the all-in-one Vagrant virtual machine (VM) for OpenShift Origin located at http://www.openshift.org/vm/ . If you have access to a working OpenShift installation you can omit any setup instructions specific to Vagrant and proceed directly to application setup and use cases.

Note that the OpenShift PaaS provides first class native support for Docker image based container specification and leverages Kubernetes for container (pod) life-cycle management. The learning outcomes of this Demonstration include:

- Install your own OpenShift 3 Vagrant VM
- Install a local DNS service using dnsmasq
- Configure OpenShift 3 to use the local DNS service
- Learn how to pull down Docker images and build new OpenShift applications
- Access the OpenShift locally using URL routes such as *.cloudapps.example.com
- Familiarise yourself with the OpenShift 3 Developer console
- Familiarise yourself with the Drools Work console
- Test out various Drools KIE server REST APIs using clients such as pHp, RStudio, cURL and SoapUI
- Learn how to change rules using the Workbench and oc rsync the changes to the KIE Server
- A little taste of xPaaS!

2 Setup OpenShift 3

Create a working directory for your demonstration such as ~/Vagrant/OpenShift origin and clone down the main repository for the demo:

https://github.com/StefanoPicozzi/weightwatcher2

2.1 For OpenShift Origin Vagrant VM

Follow these instructions that follow to download, install and prepare the your host with dnsmasq and the guest Vagrant VM. The steps are documented assuming OS/X (El Capitan) and can be summarised as per below.

1. Install the all-in-one VM and make a few changes to overcome a Vagrant ssh timeout issue
2. Setup a dnsmasq service on your host machine
3. Configure your OpenShift Origin VM to reference your dnsmasq service

The setup instructions are currently predominantly manual and OS/X centric - but can be automated and generalised with tools such as Ansible and simplified with the incorporation of a separate VM to host the DNS service. That's on the to-do-list for the Community.

2.2 For Existing OpenShift System

Proceed directly to section 3.2. Ignore references to Vagrant ssh and users and replace with credentials that represents how you access OpenShift as a Developer and as the root user for system administration purposes.

2.3 OpenShift Origin Vagrant Install

```
$ cd ~/Vagrant/OpenShiftOrigin

# Visit http://www.openshift.org/vm/ and download all bits to .
$ vagrant box list
$ vagrant box remove openshift3
$ vagrant box add --force --name openshift3 openshift-bootstrap*.box
# Follow the website instructions to install the oc client tools
$ oc version

$ vi Vagrantfile
config.vm.network "private_network", ip: "192.168.33.10"
vb.memory = "4096"
#vb.name="openshift3"

$ vagrant up
$ vagrant ssh
$ su -
Password: vagrant

$ wget -no-check-certificate
https://raw.github.com/mitchellh/vagrant/master/keys/vagrant.pub -O
/home/vagrant/.ssh/authorized_keys
$ chmod 0600 /home/vagrant/.ssh/authorized_keys

$ vagrant halt

$ vi Vagrantfile
config.ssh.username='vagrant'
config.ssh.password='vagrant'
config.ssh.insert_key=false

# Verify successful boot sequence a few times
$ vagrant up
$ vagrant halt
```

2.4 Mac OS/X Host Configuration for dnsmasq

```
# From Mac OS/X desktop

# Find your <IP> address to use with dns.example.com
$ ifconfig | grep 192.

# Edit /etc/hosts file
$ vi /etc/host
192.168.33.10  master.example.com openshift
192.168.0.14   dns.example.com dns
127.0.0.1 localhost localhost.localdomain localhost4
localhost4.localdomain4
::1  localhost localhost.localdomain localhost6 localhost6.localdomain6

$ sudo cp /etc/resolv.conf /etc/resolv.conf.upstream
$ cat /etc/resolv.conf.upstream

$ brew install dnsmasq

$ sudo cp /usr/local/opt/dnsmasq/dnsmasq.conf.example
/usr/local/etc/dnsmasq.conf

$ sudo vi /usr/local/etc/dnsmasq.conf
strict-order
domain-needed
local=/example.com/
bind-dynamic
address=/.cloudapps.example.com/192.168.33.10
log-queries
resolv-file=/etc/resolv.conf.upstream
dhcp-lease-max=1000

$ sudo launchctl unload
/Library/LaunchDaemons/homebrew.mxcl.dnsmasq.plist
$ sudo launchctl load /Library/LaunchDaemons/homebrew.mxcl.dnsmasq.plist
```

```
$ sudo launchctl stop homebrew.mxcl.dnsmasq
$ sudo launchctl start homebrew.mxcl.dnsmasq

$ tail -f /var/log/system.log

# Flush cache using
$ sudo killall -HUP mDNSResponder
```

```
# Desktop Menu > System Preference > Network
# Select your network connection
> Advanced …
> DNS

DNS Servers:
# Add 192.168.33.10 to top of list

Search Domains:
# Add cloudapps.example.com to top of list
# Add example.com to top of list

# Check ANSWER SECTION: for 192.168.33.10
$ dig testdrive2.cloudapps.example.com
```

2.5 OpenShift Origin Vagrant Final Preparation Steps

```
# Find your <IP> address
$ ifconfig | grep 192.

# cd ~/Vagrant/OpenShiftOrigin
$ vagrant up

$ vagrant ssh
Password: vagrant
$ su -
Password: vagrant

# Edit /etc/resolv.conf and prevent NetworkManager overwrite
# The first nameserver points to your Host <IP> address
$ chattr -i /etc/resolv.conf
$ vi /etc/resolv.conf
search cloudapps.example.com
nameserver 192.168.0.14
nameserver 10.0.2.3
$ chattr +i /etc/resolv.conf

# Edit /etc/hosts file
$ vi /etc/host
192.168.33.10  master.example.com openshift
192.168.0.14   dns.example.com dns
127.0.0.1 localhost localhost.localdomain localhost4
localhost4.localdomain4
::1  localhost localhost.localdomain localhost6 localhost6.localdomain6

# Verify ANSWER SECTION: reports 192.168.33.10
$ dig testdrive2.cloudapps.example.com
```

3 Setup Demonstration Applications

Once you have completed these prerequisite OpenShift Origin guest VM and host dnsmasq steps you can move onto setting up the demonstration OpenShift application and use cases. We are going to create three OpenShift (Docker-image-based) applications for the weightwatcher Decision Server, the companion website and another to host RStudio Server. The setup steps are summarised as follows:

1. Verify your environment. Do this every time before you try the demo.
2. Make some configuration changes (as root) to your OpenShift Origin system.
3. Create the applications as a Developer user (as admin)

3.1 Environment Checklist

```
# Always verify your environment before demonstrating use cases
# Especially if you have changed networks since last test


$ cat /etc/resolv.conf
$ cat /etc/resolv.conf.upstream

# Check your Host <IP> matches entry in /etc/resolv.conf and /etc/hosts
$ ifconfig | grep 192.

# If your IP has changed then on guest you will need to change the
# /etc/hosts and etc/resolv.conf files to reflect this new Hosy IP.


# And on the Host system you will need to go to System Preferences
# Network tab.  Note the note the dhcp supplied DNS settings.
# Change the resolv settings in Network GUI to your IP and then change
# /etc/resolv.conf.upstream with the saved DNS settings.


# Check ANSWER SECTION: for 192.168.33.10
$ dig testdrive2.cloudapps.example.com


# Check Internet addresses are reachable
$ ping www.google.com


# If changes needed edit files as per Appendix and restart dnsmasq
$ sudo launchctl stop homebrew.mxcl.dnsmasq
$ sudo launchctl start homebrew.mxcl.dnsmasq

$ tail -f /var/log/system.log
```

3.2 Demo Setup as OpenShift root User

```
$ cd ~/Vagrant/OpenShiftOrigin

$ vagrant status
$ vagrant up

$ vagrant ssh
Password: vagrant
$ su –
Password: vagrant

# Check OpenShift security context constraints is RunAsAny
$ oc edit scc restricted
runAsUser
type: RunAsAny

# Pull down the images
$ docker pull spicozzi/weightwatcher2
$ docker pull spicozzi/testdrive2
$ docker pull spicozzi/rstudio2
$ docker pull spicozzi/workbench2

$ docker images
$ oc get images

# Clone done the weightwatcher2 distribution
$ cd ~
$ git clone https://github.com/StefanoPicozzi/weightwatcher2
```

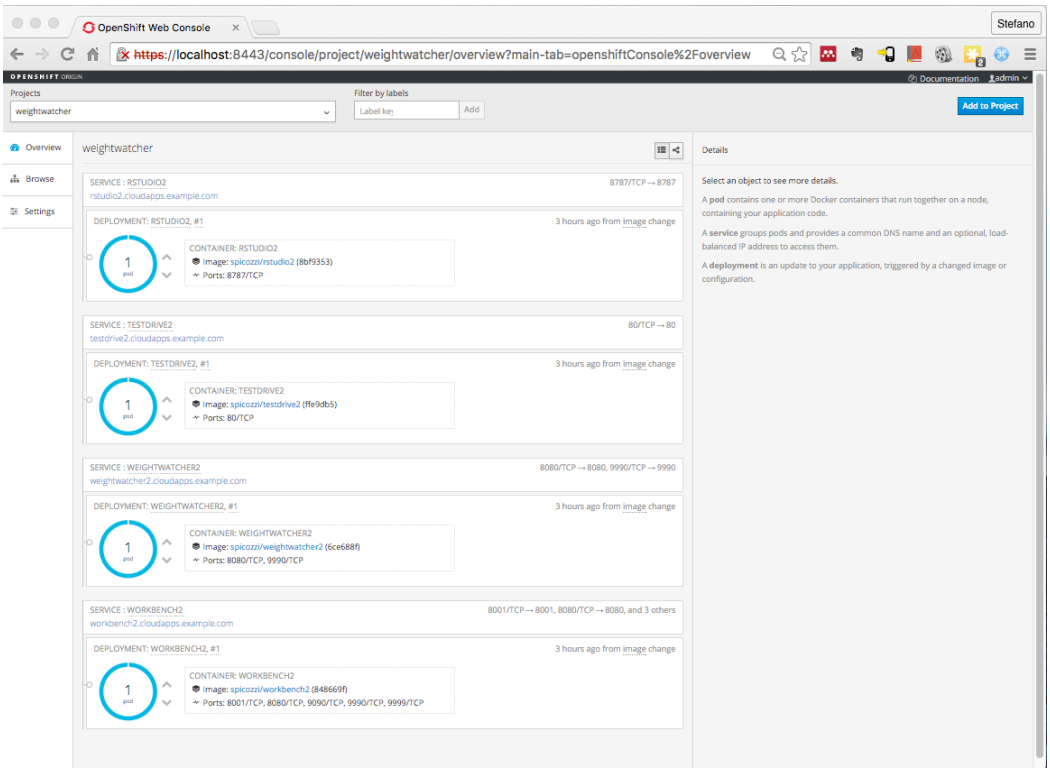
3.3 Demo Setup as OpenShift Origin (admin) Developer User

```
$ cd ~/Vagrant/OpenShiftOrigin
$ git clone https://github.com/StefanoPicozzi/weightwatcher2

$ vagrant status
$ vagrant up

$ oc login
Username: admin
Password: password

$ oc delete project weightwatcher
$ oc new-project weightwatcher
$ oc project weightwatcher

# Create the OpenShift applications
$ oc new-app spicozzi/weightwatcher2
$ oc new-app spicozzi/testdrive2
$ oc new-app spicozzi/rstudio2
$ oc create –f weightwatcher2/workbench2.yaml

$ oc expose service weightwatcher2 --name=weightwatcher2-route --
hostname=weightwatcher2.cloudapps.example.com
$ oc expose service rstudio2 --name=rstudio2-route --
hostname=rstudio2.cloudapps.example.com
$ oc expose service testdrive2 --name=testdrive2-route --
hostname=testdrive2.cloudapps.example.com
$ oc expose service workbench2 --name=workbench2-route --
hostname=workbench2.cloudapps.example.com

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

# Now point your Browser to https://master.example.com:8443/console
```

4 Standard Use Cases

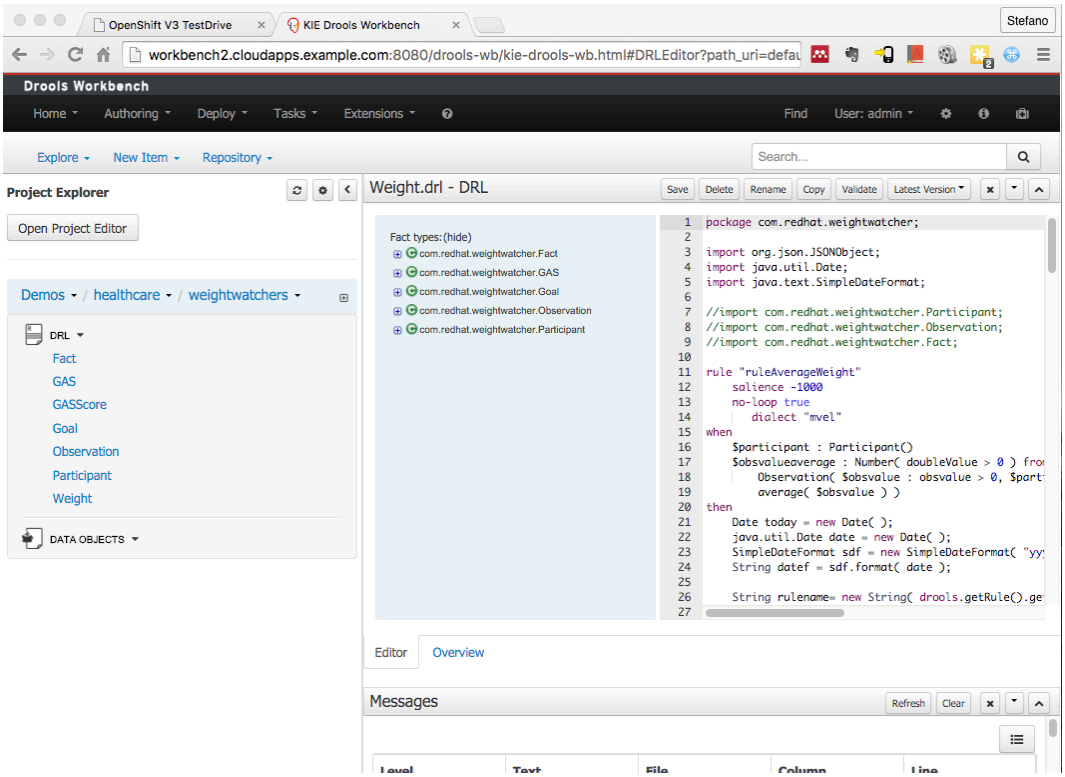
4.1 OpenShift Developer Console Tour

Open your browser at the address of the OpenShift Developer application – https://master.example.com:8443. Tour around the application and try out some of the features such as inspecting the log files of a running pod and logging into a container.



4.2 Drools Workbench Tour

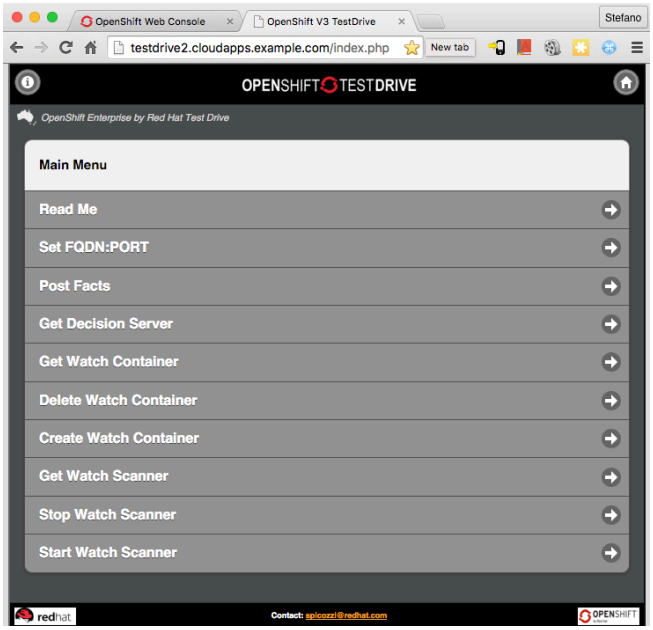
Open your browser at the address of the Drools Workbench – workbench2.cloudapps.example.com/drools-wb - and login as admin/admin. Tour around the application and try out some of the features such as inspecting the supplied project, DRL files and Data Models. You may use this later if you wish to change some of the rules and apply the changes to the Decision Server (aka KIE Server).

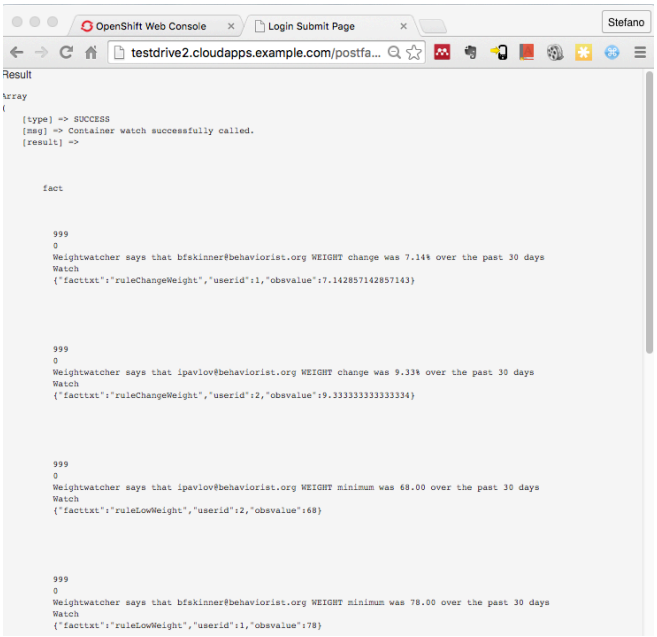


4.3 Companion Website Test Case

Open your browser at the address of the PHP companion application – testdrive2.cloudapps.example.com. This showcases some of the many KIE Server REST APIs.

Set the FQDN:PORT to point to your Decision Server instance which defaults to weightwatcher2.cloudapps.example.com and click Submit to save. Now try the Post Facts menu option to verify that it is accepting requests correctly. Output should look similar to the 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.





4.4 Simple cURL Test Case

These samples assumes you have cloned down https://github.com/StefanoPicozzi/weightwatcher2 . Some simple cURL scripts have also been supplied to test the health of your configuration are located under the /tools/cURL. These are similar to the test cases available at the companion website and assume weightwatcher2.cloudapps.example.com as the KIE Server endpoint. Edit each script to change the http://FQDN:PORT if necessary:

```
$ 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 Drools CEP notifications
```

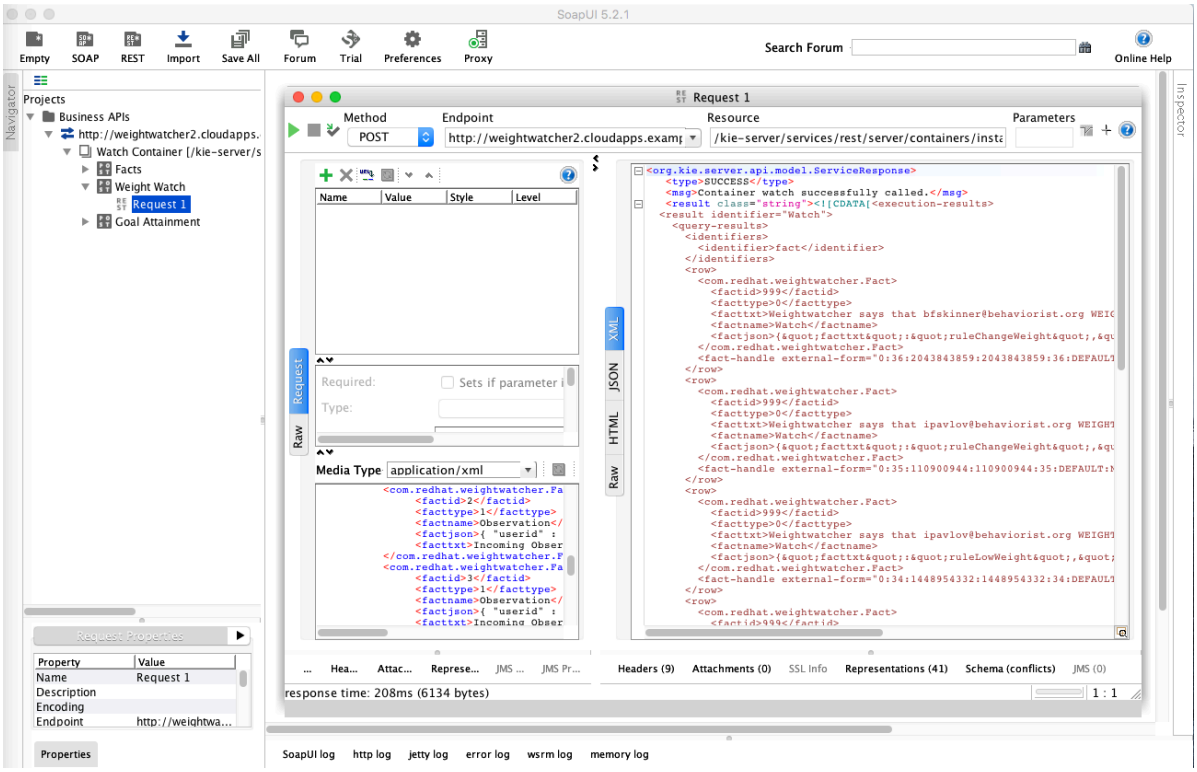
4.5 SoapUI Samples

These samples assumes you have SoapUI installed on your workstation and that you have cloned down https://github.com/StefanoPicozzi/weightwatcher2 . Launch SoapUI and then import the project with a name that includes the label "Business APs" as located at weightwatcher2/tools/SoapUI. 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.

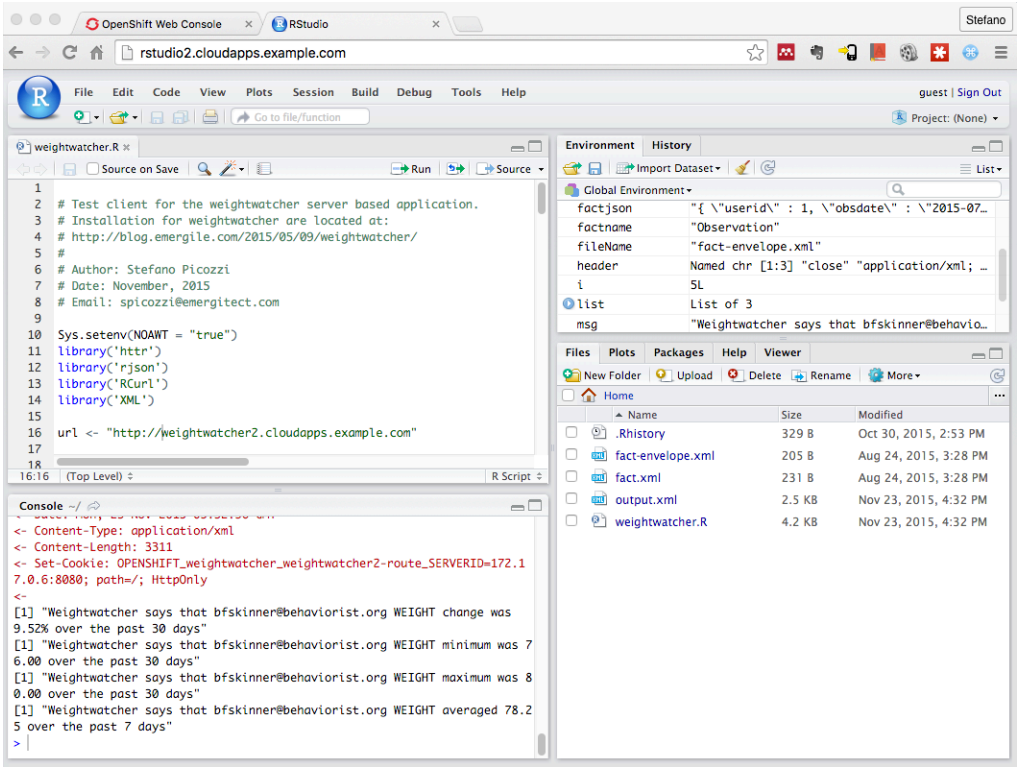
The "Weight Watch" sample shows an invocation in which a set of facts containing weight measurements is sent to the Decision Server. CEP rules are then applied to derive insights as per the response payload. The request consists of facts representing Participant, Goal and Observation data records. The 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.



4.6 R using RStudio

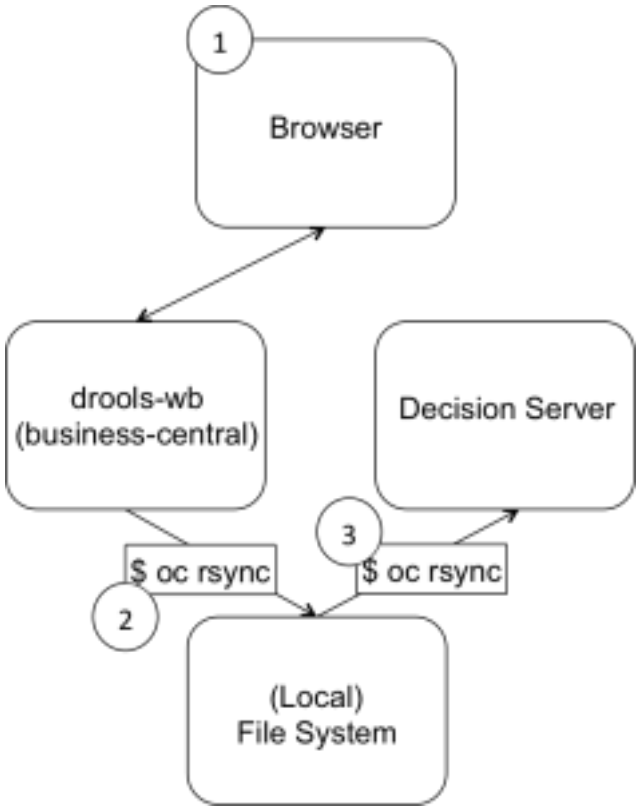
The demonstration shows R script using RStudio Server as a client interacting with the KIE Server. Launch the RStudio pod – rstudio2.cloudapps.example.com. Login as guest/guest. Edit the weightwatcher.R script URL end point to reflect your environment if necessary, then source to run. Inspect the R script for hints of techniques on how to prepare, send and receive payloads between R and the KIE Server.



5 Advanced Use Cases

5.1 Rule Changes using oc rsync

A more advanced rule change use case is also documented later that makes use of the OpenShift (oc) rsync feature. Basic schematic as follows. Refer also http://blog.emergile.com/2015/11/16/drools-rules-rsynchronicity/



5.1.1 Start the KIE Server Scanner

```
# This is necessary so that the KIE Server will pick up rule changes

# Launch a Browser and point it to testdrive2.cloudapps.example.com
Click the Start Watch Scanner menu option
Click the Get Watch Scanner menu option to verify
```

5.1.2 Rule Changes using Workbench

```
# Launch a Browser to workbench2.cloudapps.example.com/drools-wb
# Login as admin/admin

# Use the workbench GUI to make some changes to a DRL file
# Click build & deploy to save the changes to the rules .jar file

$ cd ~/Vagrant/OpenShift
# Now oc rync the repository directory to your local file system

# First find the NAME of the pod
$ oc get pods | grep workbench2

# Now pull down the rules repo
$ oc rsync <NAME>:/opt/jboss/.m2/repository .
```

5.1.3 Apply Rule Changes to the KIE Server

```
# Now rynch the local repository directory up to the weightwatcher2 pod

$ cd ~/Vagrant/OpenShift
# First find the NAME of the pod
$ oc get pods | grep weightwatcher2

# Now push up the rules repo
$ oc rsync repository <NAME>:/opt/jboss/.m2

# Check that the scanner has picked up the changed rules file
$ oc logs –f <NAME>

# Launch a Browser and point it to testdrive2.cloudapps.example.com
Click the Post Facts menu option to verify changes applied
$ oc logs –f <NAME>
```

6 Extras

6.1 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 directly 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 case included
$ docker pull spicozzi/rstudio2

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

6.2 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

https://github.com/StefanoPicozzi/workbench2
```

6.3 Launch Container Instances suing boot2docker

You can run launch these images directly from Docker as follows.

```
# Launch 4 terminal windows and run the container instances in each

# 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

# Terminal 4 (optional)
$ docker run -it -p 8081:8080 spicozzi/workbench2
```

6.4 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

# Disconnected access to load Docker images
$ docker save -o workbench2.tar spicozzi/workbench2
$ docker load –i workbench2.tar

# 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 rmi -f $(docker images | grep "<none>" | awk "{print $3}")
```

6.5 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>

# Inspect the openshift configuration
$ locate master-config

# Stop/start openshift
$ systemctl stop openshift
$ systemctl start openshift
$ systemctl status openshift
```

6.6 Changing Rules boot2docker example

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 directly from Docker as tested using boot2docker on OS/X. The basic steps are as follows:

1. (Re)launch the JBoss Drools Workbench container (workbench2) with the Maven repository mounted as an external volume (workbench2/m2)
2. (Re)launch the Decision Server container (weightwatcher2) with the Maven repository mounted as an external volume (weightwatcher2/m2)
3. (Re)launch the companion website PHP application (testdrive2)
4. From testdrive2 Browser, click Start Scanner to start the Decision Server scanner
5. From workbench2 Workbench Browser, make some changes to your rules, save them and then build-and-deploy the artefact. Login to Workbench using IP_ADDRESS:8018/drools-wb and credentials admin/admin
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 Browser, click Post Facts and verify that the changes have been applied
9. From workbench2 container window, check the log for evidence of a change to the rule jar file
10. From weightwatcher2 container window, check the log for evidence of a scanner event to for a new rule jar file

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 Drools Workbench. Steps 6 and 7 can be automated for more real-life scenarios using your, e.g. favourite CI/CD tooling.