Drools 6.3.0.FINAL
Weight Watcher2 Demo

Stateless CEP Decision Server

showcase for Docker and OpenShift 3

Stefano Picozzi
Sydney, Australia

spicozzi@emergitect.com

http://blog.emergile.com

Table of Contents

1 Introduction

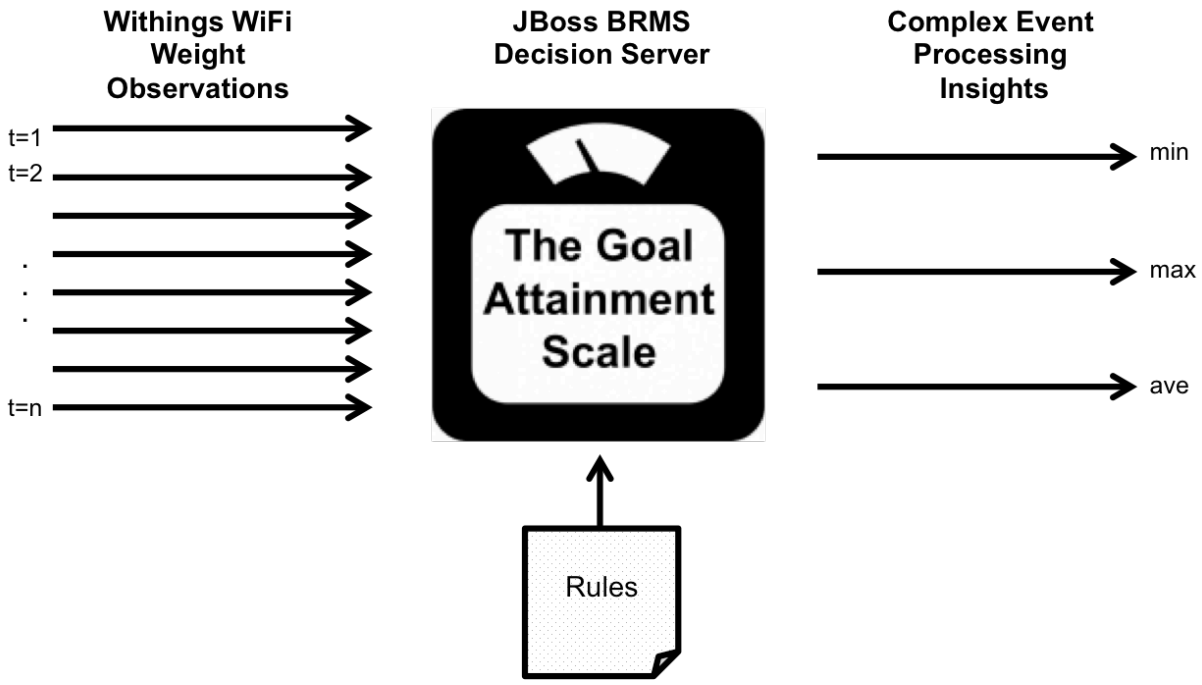
Interested in a demo that showcases the Drools (6.3.0.FINAL) Decision Server? Then look here. The application is a stateless Decision Server with complex event processing (CEP) support based on a pseudo clock.

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 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 the upcoming OpenShift Enterprise V3.1 and JBoss BRMS 6.2 releases. 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 case included
$ 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

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

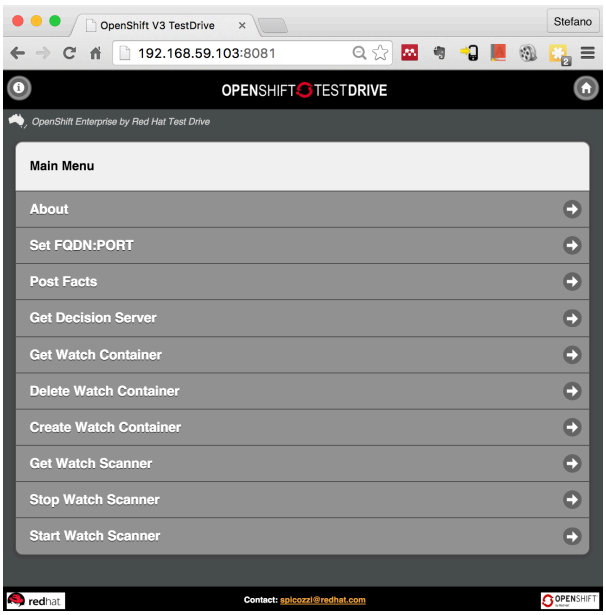
3 Running the Basic Demos

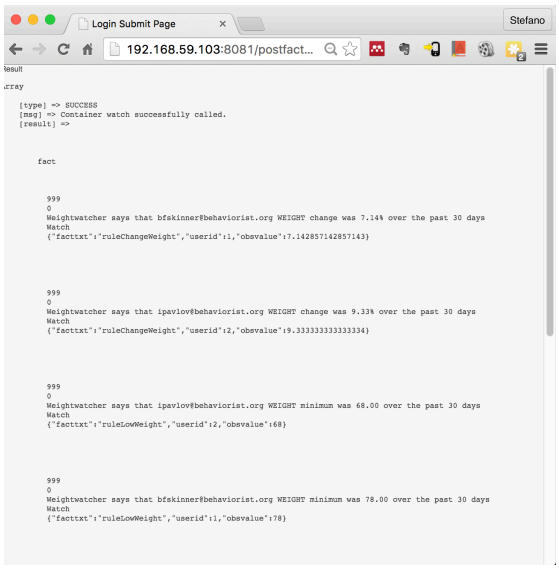
3.1 Launch Container Instances

```
# Launch 3 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
```

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
```

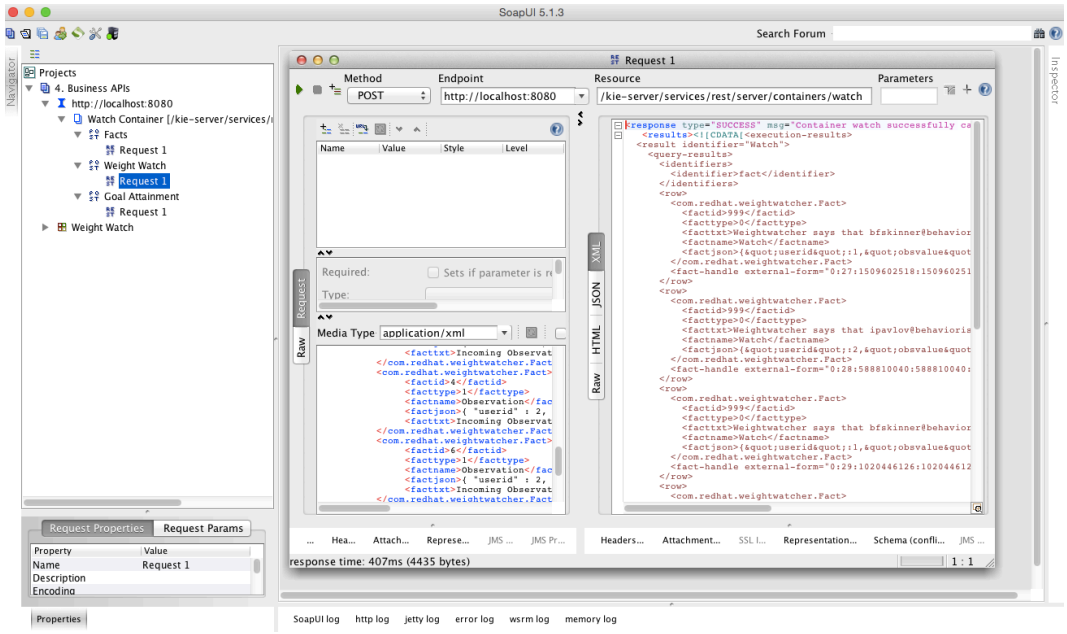
3.4 SoapUI Samples

This example assumes you have SoapUI installed on your workstation. Launch 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.

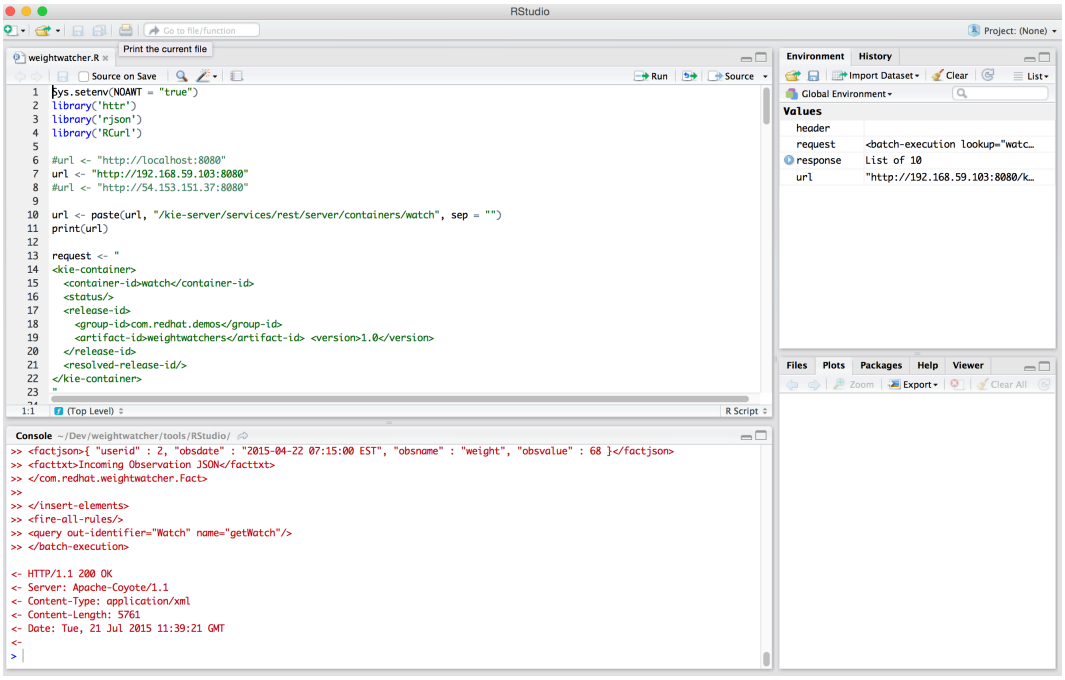


3.5 R using RStudio

An R script based sample showing how to interact with the Weight Watcher Services is located under the weightwatcher2/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, edit the url and the source weightwatcher.R to run
```

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. (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. More detailed documentation will follow later.

5 OpenShift Origin using Vagrant All-in-one VM

OpenShift can load and run an arbitrary Docker image. We are going to shocasw some use cases using the all-in-one Vagrant virtual machine (VM) for OpenShift Origin located at http://www.openshift.org/vm/ .

IMPORTANT. Follow the instructions to download, install and prepare the VM and your Host as detailed in the Appendix below before proceeding. The documentation assumes Mac OS/X but should be transferrable to other Linux environments. Once you have completed these prerequisite steps you can move onto the Demo use cases as described in this section.

We are going to create three OpenShift applications for the weightwatcher Decision Server, the companion website and another to host RStudio Server. A more advanced rule change use case is also documented that makes use of the oc rsync feature.

Deploy. Enjoy!

5.1 Demo Setup as Vagrant root User

```
$ cd ~/Vagrant/OpenShiftOrigin
$ vagrant status
$ vagrant up

$ vagrant ssh
$ su –
Password: vagrant

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

$ docker images
$ oc get images

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

5.2 Demo Setup as OpenShift Origin admin User

```
$ cd ~/Vagrant/OpenShiftOrigin
$ 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 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 describe pod weightwatcher2
$ oc describe pod rstudio2
$ oc describe pod testdrive2

# Now point your Browser to https://master.example.com:8443/console
# Explore and click away!
```

5.3 Rule Changes using oc rsync

5.3.1 Rule Changes using local Drools Workbench

```
# To make rule changes we need to launch an instance of drools_wb

# From a terminal window using eg boot2docker
$ docker pull spicozzi/workbench2

# Note the <IP> return from this command
$ boot2docker ip

$ cd ~
$ git clone https://github.com/StefanoPicozzi/workbench2
$ cd workbench2
$ ./docker-run-with-m2-attached.sh

# Now launch a Browser and point it to <IP>:8180/drools_wb
# Login as admin/admin

# Use the drools_wb to make some changes to a DRL file
```

5.3.2 oc rsynch Rule Changes as OpenShift Origin admin User

```
$ cd ~/workbench2/m2
$ oc rsync repository <PID>:/opt/jboss/.m2

# Check that the scanner has picked up the change
$ oc logs –f <PID>
```

5.3.3 Verify Rule Change as Vagrant root User

```
# Return to the terminal window where you are the Vagrant root user
$ cd ~/weightwatcher2/tools/cURL

$ ./post-facts.sh
# Now check outputs and log files as necessary to verify change applied
```

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>

# Inspect the openshift configuration
$ locate master-config

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

A.3 OpenShift Origin Vagrant with dnsmasq for OS/X

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 –name openshift3 openshift-bootstrap*.box

$ vi Vagrantfile
config.vm.network "private_network", ip: "192.168.33.10"
#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
```

Mac OS/X Host Configuration

```
# From Mac OS/X desktop

# 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
```

OpenShift Origin Vagrant Final Preparation Steps

```
# cd ~/Vagrant/OpenShiftOrigin
$ vagrant up

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

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

# 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
```