RED HAT CONSULTATION

Decision Manager Engagement report

PREPARED FOR: AIA

Table of Contents

Preface

Confidentiality, Copyright and Disclaimer

This is a confidential document between Red Hat, Inc. and AIA ("Client").

This document is not a quote and does not include any binding commitments by Red Hat.

About This Document

This is a confidential document between Red Hat, Inc. and **AIA** detailing a Red Hat Decision Manager Architecture review engagement. It is intended for technical staff responsible for the following functions:

- Decision Manager installation and configuration
- Execution Server installation and configuration
- Deployment flow

Software

The latest version of Decision Manager (DM) and Enterprise Application Platform (EAP) is being installed on both SIT, UAT and PROD environments.

Component	Version
EAP	7.2
Decision Manager	7.3

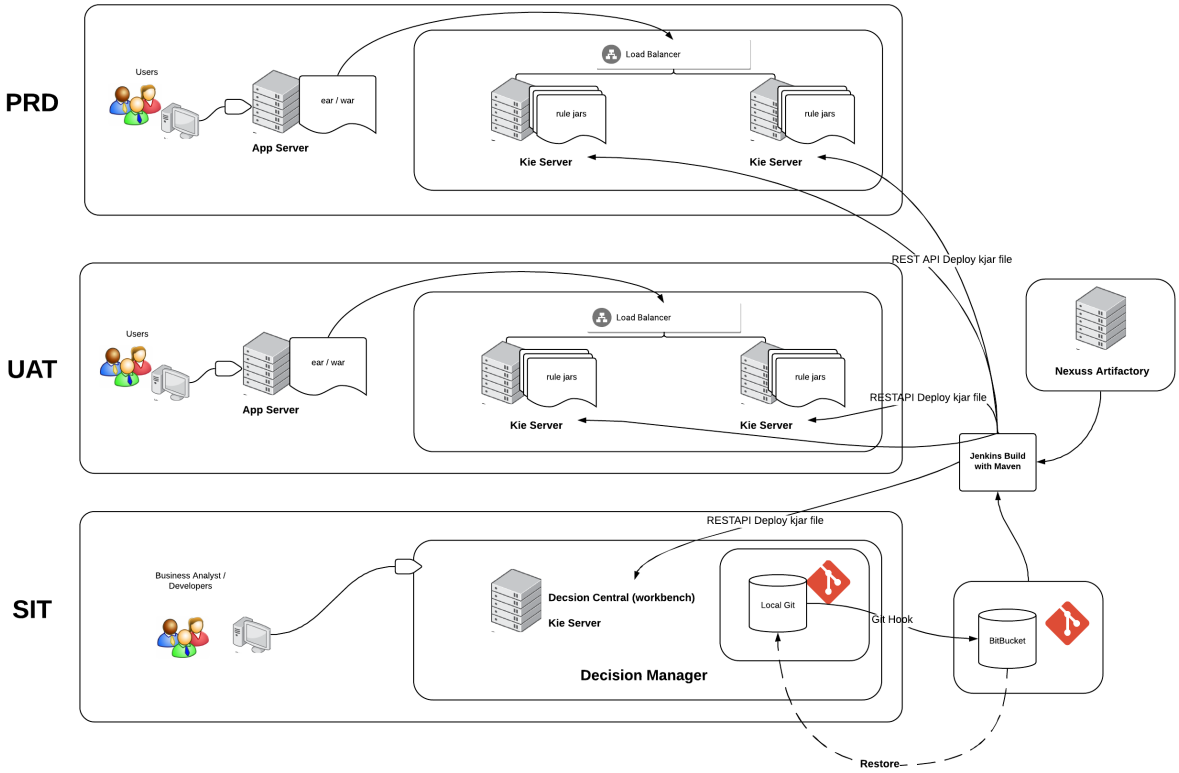
SUGGEST UPDATE THE OS TO RHEL7 AS DM7 ONLY HAVE TESTED ON RHEL7

It is found that the OS running the Decision Manage is SUSE Linux. This version of OS is not tested and verify by Red Hat. Suggest to upgrade to Red Hat Linux 7.6, which is latest version of tested with Decision Manager 7.

Ref: https://access.redhat.com/articles/3354301

Proposed Architecture and Operation Flow

Architecture



Short Notes for the architecture:

1. The Decision Manager and Execution Server is being installed on SIT
2. The SIT local git configure git-hook to auto sync source change to external git (Git Stash)
3. Developers (Administrators) will login to the Decision Central through 38080 and 38443 port on SIT
4. The Execution Servers on UAT and PROD enable ports 38080 and 38443. Applications will access through load balancer with 443 port and redirect to 38443 port of the Execution Servers.

Tune the GC algorithm used in DM

It is found that the deployment is using default GC algorithm that comes with JBoss EAP installation, which could be good in general but not fine tuned. Although tuning GC is a long way to go, it is suggest to use Red Hat JVM config online application as a starting point. Also it is recommended to use latest G1 algorithm to enjoy benefits to brings to us.

JVM config url: https://access.redhat.com/labs/jvmconfig/

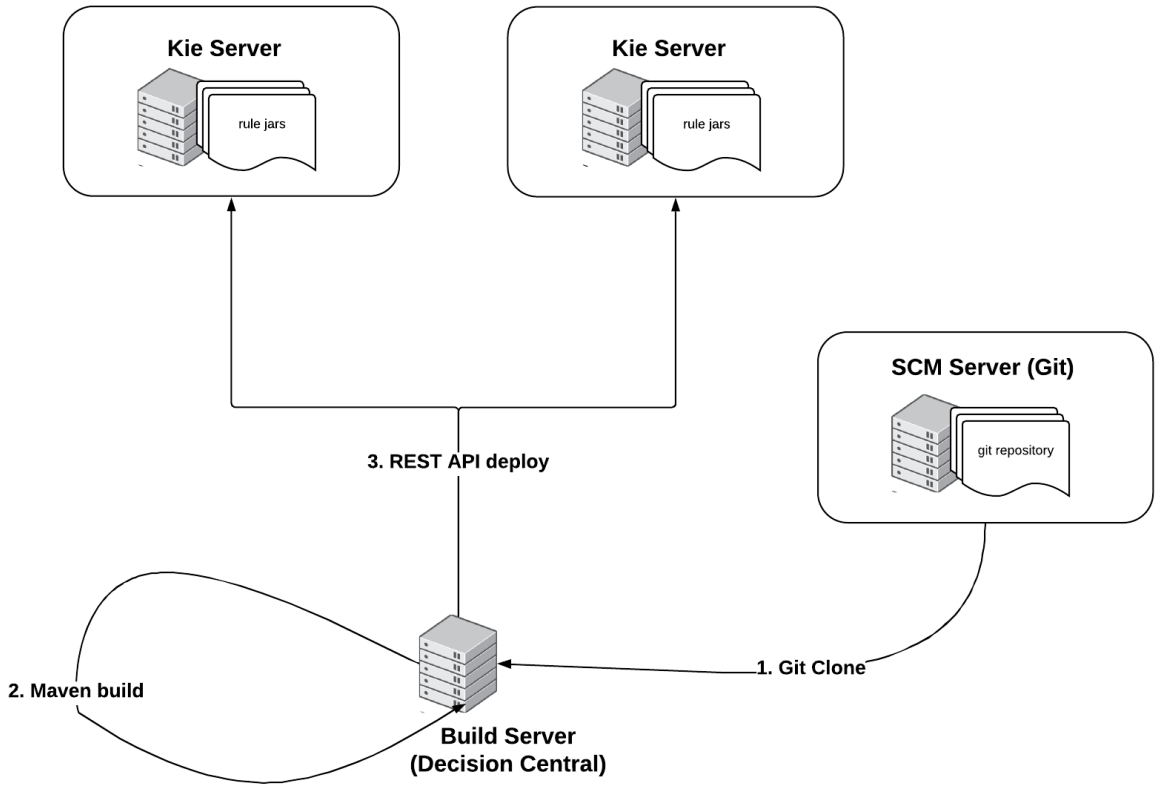
JVM Option	Description
-server	The Server VM has been specially tuned to maximize peak operating speed. It is intended for executing long-running server applications, which need the fastest possible operating speed more than a fast start-up time or smaller runtime memory footprint. Since JBoss is generally a long running process, using -server is recommended
-XX:+DoEscapeAnalysis	Escape analysis is a technique by which the Java Hotspot Server Compiler can analyze the scope of a new object's uses and decide whether to allocate it on the Java heap. Escape analysis is supported and enabled by default in Java SE 6u23 and later. Seehttp://docs.oracle.com/javase/7/docs/technotes/guides/vm/performance-enhancements-7.htmlfor more information.
-XX:+UseG1GC	The G1 collector is a server-style garbage collector, targeted for multi-processor machines with large memories. It meets garbage collection (GC) pause time goals with high probability, while achieving high throughput. Whole-heap operations, such as global marking, are performed concurrently with the application threads. This prevents interruptions proportional to heap or live-data size. See Garbage-First Collector for more info.
-XX:+ExplicitGCInvokesConcurrent	Ensures explicit GC calls from System.gc() will be done concurrently.
-XX:MaxGCPauseMillis=500	Sets a target for the maximum GC pause time. This is a soft goal, and the JVM will make its best effort to achieve it.

JVM Option	Description
-XX:+UseCompressedOops	Enables the use of compressed pointers (object references represented as 32 bit offsets instead of 64-bit pointers) for optimized 64-bit performance with Java heap sizes less than 32gb. Compressed oops is supported and enabled by default in Java SE 6u23 and later. In Java SE 7, use of compressed oops is the default for 64-bit JVM processes when -Xmx isn't specified and for values of -Xmx less than 32 gigabytes. For JDK 6 before the 6u23 release, use the -XX:+UseCompressedOops flag with the java command to enable the feature.
-XX:+UseCompressedClassPointers	Enable UseCompressedClassPointers. UseCompressedClassPointers uses a 32-bit offset to represent the class pointer in a 64-bit process as does UseCompressedOops for Java object references.
-XX:CompressedClassSpaceSize=1024M	If UseCompressedOops is turned on and UseCompressedClassesPointers is used, then two logically different areas of native memory are used for class metadata. UseCompressedClassPointers uses a 32-bit offset to represent the class pointer in a 64-bit process as does UseCompressedOops for Java object references. A region is allocated for these compressed class pointers (the 32-bit offsets). The size of the region can be set with CompressedClassSpaceSize. The space for the compressed class pointers is reserved as space allocated by mmap at initialization and committed as needed.
-XX:MetaspaceSize=1024M	Sets the size of the allocated class metadata space that will trigger a garbage collection the first time it is exceeded. This threshold for a garbage collection is increased or decreased depending on the amount of metadata used. The default size depends on the platform.
-XX:MaxMetaspaceSize=1024M	Sets the maximum amount of native memory that can be allocated for class metadata. By default, the size is not limited. The amount of metadata for an application depends on the application itself, other running applications, and the amount of memory available on the system.

Setup:

1. SIT host will run an instance of JBoss EAP hosting Decision Central, Execution Server
2. UAT hosts will run only Execution Server on the JBoss EAP instance
3. PROD hosts will run only Execution Server on the JBoss EAP instance
4. Load balancer will be placed in between intranet and Execution Servers on both UAT and PROD environments
5. Addition of new execution server will be easy, only need to add new set of host running execute server and config them to the Load balancer

Deployment Flow



Steps:

1. Administrator git clone the source code in build server
2. Administrator build the source Coe into kjar (Current setting is the SIT host) (build.sh)
3. Administrator deploy the build kjar into maven server hosted by the KIE Servers (deploy.sh)

Appendix

build.sh

```
#!/bin/bash
export PATH=$PATH:/usr/local/apache-maven/bin/
#requires xmllint

CURRENTDIR=$(dirname $0)
JARFILE=$(readlink -f $1)
MAVENPATH=$(jar tf $JARFILE  |grep pom.xml |grep maven)
GROUPID=$(unzip -p $JARFILE $MAVENPATH | /data/jbossEAP/au-
tomation/readpom.py | grep groupId |awk -F= '{print $2}')
ARTIFACTID=$(unzip -p $JARFILE $MAVENPATH | /data/jbossEAP/au-
tomation/readpom.py | grep artifactId |awk -F= '{print $2}')
VERSION=$(unzip -p $JARFILE $MAVENPATH | /data/jbossEAP/au-
tomation/readpom.py | grep version |awk -F= '{print $2}')
GROUPPATH=$(echo $GROUPID |tr '.' '/')
TMPSTR=$( date +%s)
CONTAINERNAME="demo"

echo $GROUPID
echo $ARTIFACTID
echo $VERSION
echo $GROUPPATH
cd $CURRENTDIR


echo "putting the binary into decision central"
mvn deploy:deploy-file -DgroupId=$GROUPID -DartifactId=$ARTIFACTID
-Dversion=$VERSION -Dpackaging=jar -Dfile=$JARFILE -Durl=http://
10.145.83.47:8080/decision-central/maven2  -DrepositoryId=dm

echo "checking if there is existing containers"
RESULT=$(curl -s -H 'Content-Type: application/xml' -u
rhdmAdmin:redh@T123  http://10.145.83.47:8180/decision-central/
rest/controller/management/servers/default-kieserver/containers/
$CONTAINERNAME |grep "does not have container with id")

if  [ "x$RESULT" = "x" ]; then
   echo "$CONTAINERNAME already defined, will stop it and remove
it"
   echo "Stopping container $CONTAINERNAME"
   curl -X POST -s -u rhdmAdmin:redh@T123  http://
10.145.83.47:8180/decision-central/rest/controller/management/
servers/default-kieserver/containers/$CONTAINERNAME/status/stopped
```

```
   echo "Removing container $CONTAINERNAME"
   curl -X DELETE -s -u rhdmAdmin:redh@T123  http://
10.145.83.47:8180/decision-central/rest/controller/management/
servers/default-kieserver/containers/$CONTAINERNAME


   sleep 10
   echo "check again, if it didn't removed, will abort"
   RESULT_AGAIN=$(curl -s -H 'Content-Type: application/xml' -u
rhdmAdmin:redh@T123  http://10.145.83.47:8180/decision-central/
rest/controller/management/servers/default-kieserver/containers/
$CONTAINERNAME |grep "does not have container with id")
   if  [ "x$RESULT_AGAIN" = "x" ]; then
      "Failed to remove container, exiting with ERROR"
      exit -1
   fi
fi

echo "Crating new containers"
RET=$(curl -s --write-out "%{http_code}\n" --output /dev/null -H
'Content-Type: application/xml' -XPUT -u rhdmAdmin:redh@T123
http://10.145.83.47:8180/decision-central/rest/controller/manage-
ment/servers/default-kieserver/containers/$CONTAINERNAME -d $(/
data/jbossEAP/automation/createContainer.py $CONTAINERNAME $CON-
TAINERNAME $ARTIFACTID $GROUPID $VERSION))
echo "Create result: $RET"
if [ "$RET" != "201" ]; then
   echo "Failed to create container exiting"
   exit -2
fi

echo "starting containers"
RET=$(curl  -s --write-out "%{http_code}\n" --output /dev/null -H
'Content-Type: application/xml' -X POST -s -u rhdmAdmin:redh@T123
http://10.145.83.47:8180/decision-central/rest/controller/manage-
ment/servers/default-kieserver/containers/$CONTAINERNAME/status/
started)

echo "Start result: $RET"

if [ "$RET" != "200" ]; then
   echo "Failed to start container exiting"
   exit -3
fi

echo "checking container readiness"
CONTAINERLIST=$(curl -s -u rhdmAdmin:redh@T123  http://
10.145.83.47:8180/decision-central/rest/controller/management/
```

```
servers/default-kieserver/ | /data/jbossEAP/automation/readServer-
List.py)

TIMEUSED=0
for CONTAINERINSTANCE in $CONTAINERLIST;
do
   STATUS="STOPPED"
   while [ "x$STATUS" != "xSTARTED" ];
   do
     STATUS=$(curl -s -u kieuser:redh@T123  $CONTAINERINSTANCE/
containers/$CONTAINERNAME |/data/jbossEAP/automation/readContain-
erStatus.py )
     echo "SERVER: $CONTAINERINSTANCE STATUS:  $STATUS"
     sleep 1
     TIMEUSED=$(( TIMEUSED +1 ))
     if [ "$TIMEUSED" -gt 30 ]; then
        echo "TIMEOUT!!!!!"
          exit -7
     fi
   done
done
```

deploy.sh

```
#!/bin/bash
export PATH=$PATH:/opt/apache-maven-3.5.3/bin/
#requires xmllint

CURRENTDIR=$(dirname $0)
JARFILE=$(readlink -f $1)
MAVENPATH=$(jar tf $JARFILE  |grep pom.xml |grep maven)
GROUPID=$(unzip -p $JARFILE $MAVENPATH | /data/jbossEAP/au-
tomation/readpom.py | grep groupId |awk -F= '{print $2}')
ARTIFACTID=$(unzip -p $JARFILE $MAVENPATH | /data/jbossEAP/au-
tomation/readpom.py | grep artifactId |awk -F= '{print $2}')
VERSION=$(unzip -p $JARFILE $MAVENPATH | /data/jbossEAP/au-
tomation/readpom.py | grep version |awk -F= '{print $2}')
GROUPPATH=$(echo $GROUPID |tr '.' '/')
TMPSTR=$( date +%s)
CONTAINERNAME="demo"

echo $GROUPID
echo $ARTIFACTID
echo $VERSION
echo $GROUPPATH
cd $CURRENTDIR


echo "putting the binary into decision central"
mvn deploy:deploy-file -DgroupId=$GROUPID -DartifactId=$ARTIFACTID
-Dversion=$VERSION -Dpackaging=jar -Dfile=$JARFILE -Durl=http://
10.145.83.47:8080/decision-central/maven2  -DrepositoryId=dm

echo "checking if there is existing containers"
RESULT=$(curl -s -H 'Content-Type: application/xml' -u
rhdmAdmin:redh@T123  http://10.145.83.47:8180/decision-central/
rest/controller/management/servers/default-kieserver/containers/
$CONTAINERNAME |grep "does not have container with id")

if  [ "x$RESULT" = "x" ]; then
   echo "$CONTAINERNAME already defined, will stop it and remove
it"
   echo "Stopping container $CONTAINERNAME"
   curl -X POST -s -u rhdmAdmin:redh@T123  http://
10.145.83.47:8180/decision-central/rest/controller/management/
servers/default-kieserver/containers/$CONTAINERNAME/status/stopped
   echo "Removing container $CONTAINERNAME"
   curl -X DELETE -s -u rhdmAdmin:redh@T123  http://
10.145.83.47:8180/decision-central/rest/controller/management/
servers/default-kieserver/containers/$CONTAINERNAME
```

```
   sleep 10
   echo "check again, if it didn't removed, will abort"
   RESULT_AGAIN=$(curl -s -H 'Content-Type: application/xml' -u
rhdmAdmin:redh@T123  http://10.145.83.47:8180/decision-central/
rest/controller/management/servers/default-kieserver/containers/
$CONTAINERNAME |grep "does not have container with id")
   if  [ "x$RESULT_AGAIN" = "x" ]; then
      "Failed to remove container, exiting with ERROR"
      exit -1
   fi
fi

echo "Crating new containers"
RET=$(curl -s --write-out "%{http_code}\n" --output /dev/null -H
'Content-Type: application/xml' -XPUT -u rhdmAdmin:redh@T123
http://10.145.83.47:8180/decision-central/rest/controller/manage-
ment/servers/default-kieserver/containers/$CONTAINERNAME -d $(/
data/jbossEAP/automation/createContainer.py $CONTAINERNAME $CON-
TAINERNAME $ARTIFACTID $GROUPID $VERSION))
echo "Create result: $RET"
if [ "$RET" != "201" ]; then
   echo "Failed to create container exiting"
   exit -2
fi

echo "starting containers"
RET=$(curl  -s --write-out "%{http_code}\n" --output /dev/null -H
'Content-Type: application/xml' -X POST -s -u rhdmAdmin:redh@T123
http://10.145.83.47:8180/decision-central/rest/controller/manage-
ment/servers/default-kieserver/containers/$CONTAINERNAME/status/
started)

echo "Start result: $RET"

if [ "$RET" != "200" ]; then
   echo "Failed to start container exiting"
   exit -3
fi

echo "checking container readiness"
CONTAINERLIST=$(curl -s -u rhdmAdmin:redh@T123  http://
10.145.83.47:8180/decision-central/rest/controller/management/
servers/default-kieserver/ | /data/jbossEAP/automation/readServer-
List.py)

TIMEUSED=0
for CONTAINERINSTANCE in $CONTAINERLIST;
```

```
do
   STATUS="STOPPED"
   while [ "x$STATUS" != "xSTARTED" ];
   do
     STATUS=$(curl -s -u kieuser:redh@T123  $CONTAINERINSTANCE/
containers/$CONTAINERNAME |/data/jbossEAP/automation/readContain-
erStatus.py )
     echo "SERVER: $CONTAINERINSTANCE STATUS:  $STATUS"
     sleep 1
     TIMEUSED=$(( TIMEUSED +1 ))
     if [ "$TIMEUSED" -gt 30 ]; then
        echo "TIMEOUT!!!!!"
          exit -7
     fi
   done
done
```

createContainer.py

```
#!/usr/bin/python
import xml.etree.ElementTree as ET
import sys

CONTAINERID=sys.argv[1]
CONTAINERNAME=sys.argv[2]
ARTIFACTID=sys.argv[3]
GROUPID=sys.argv[4]
VERSION=sys.argv[5]

XMLSTR=""

# first list
root = ET.Element("container-spec-details")
container_id = ET.SubElement(root,"container-id")
container_name = ET.SubElement(root,"container-name")
server_template_key = ET.SubElement(root,"server-template-key")
release_id = ET.SubElement(root, "release-id")
#configs = ET.SubElement(root, "configs")
#status = ET.SubElement(root, "status")

#second level
server_id = ET.SubElement(server_template_key,"server-id")
artifact_id = ET.SubElement(release_id,"artifact-id")
group_id = ET.SubElement(release_id,"group-id")
version = ET.SubElement(release_id,"version")
#config_PROCESS =  ET.SubElement(configs,"entry")
#config_RULE =  ET.SubElement(configs,"entry")


container_id.text=CONTAINERID
container_name.text=CONTAINERNAME
server_id.text="default-kieserver"
artifact_id.text=ARTIFACTID
group_id.text=GROUPID
version.text=VERSION
ET.dump(root)
```

readContainerStatus.py

```
#!/usr/bin/python
import xml.etree.ElementTree as ET
import sys

XMLSTR=""

#tree = ET.parse("test.xml")
for line in sys.stdin:
    XMLSTR += line
root = ET.fromstring(XMLSTR)
container = root.find("kie-container")
print container.get("status")
```

readServerList.py

```
#!/usr/bin/python
import xml.etree.ElementTree as ET
import sys

XMLSTR=""

#tree = ET.parse("test.xml")
for line in sys.stdin:
    XMLSTR += line
tree = ET.fromstring(XMLSTR)
serverinstances = tree.findall("server-instances")
for serverinstance in serverinstances:
    url = serverinstance.find("server-url")
    print url.text
```

readpom.py

```
#!/usr/bin/python
import xml.etree.ElementTree as ET
import sys

XMLSTR=""

#tree = ET.parse("test.xml")
for line in sys.stdin:
    XMLSTR += line
tree = ET.fromstring(XMLSTR)
for child in tree:
   if child.tag.find("groupId") >= 0:
     print "groupId="+child.text
   if child.tag.find("artifactId") >= 0:
     print "artifactId="+child.text
   if child.tag.find("version") >= 0:
     print "version="+child.text
```