Monitoring and Alerting Installation Details

This document will cover the details pertaining to installing Tomcat and Hyperic for use as the Monitoring and Alerting (M&A) back end. It will also cover the configuration needed to use the M&A client in a component such as Test Item Bank. It will not cover what is necessary for running components in a High Availability or other scaled arrangement.

Tomcat

A standard Tomcat installation is needed to run the M&A components. There are some changes that need to be made to the standard installation in order to create a running JMX server. We have created some scripts that we use in our EC2 provisioning process to perform the setup.

tomcat/base-install.sh

The base-install.sh script performs the actual installation of Tomcat using apt-get. It goes on to setup various environment variables including JAVA_HOME and JAVA_OPTS. An important setting is the value of CATALINA_OPTS. This contains the usual heap size and perm gen size settings. It also contains part of the configuration for setting up JMX.

The important properties for the JMX setup are:
com.sun.management.jmxremote.ssl=false
com.sun.management.jmxremote.authenticate=true
com.sun.management.jmxremote.password.file=<filename>
com.sun.management.jmxremote.access.file=<filename>
java.rmi.server.hostname=<FQDN>

At this point in time we have SSL turned off. Turning on SSL will require more setup so that the keystores on all of the machines have all of the correct certificates.

FQDN is the fully qualified domain name of the machine. The script populates this from an environment variable since it knows which machine is being provisioned.

Two extra properties can be set in CATALINA_OPTS: mnaServerName and mnaNodeName. These properties are used by the M&A client to correctly set the source server and node. Currently the script only sets the server name.

Two environment variables called DRC_CONFIG_DIR and SB11_CONFIG_DIR are set in the script. These point to an external directory that is used for external configuration and property files. Since configuration and property files tend to have passwords in them, the directory should be a secure directory with very limited read and write permissions.

The script does echo role and password information into the jmx configuration files.

Some permissions are set on various files and directories involved in the Tomcat installation.

The script also downloads the catalina-jmx-remote.jar from the internet. This jar file is what allows Tomcat to support JSR-160 for exporting visibility to JMX. The server needs this jar in the classpath as well as any clients connecting to the server.

tomcat/tomcat-server.xml
There is a custom Tomcat server.xml file included in the provisioning files. There is a line in the file that creates a Listener called "org.apache.catalina.mbeans.JmxRemoteLifecycleListener" and specifies some ports. All of the servers are setup with the same ports for JMX.

tomcat/add-vhost.sh
This script adds Virtual Host sections to the server.xml file to define separate Tomcat Virtual Hosts for each component deployed to the server. The master provisioning setup file for each Tomcat server will call add-vhost.sh to create the correct entry in server.xml as well as a deployment directory in /opt/tomcat.

tomcat/install.sh
This script ensures that Java is installed on the machine and then calls the base-install.sh script.

sb11-tomcat-1/setup.sh
This is just an example of the highest level script that is run to provision a machine. This script calls install.sh, calls add-vhost.sh, and then calls the Hyperic Agent installation script.

Hyperic

Hyperic consists of two pieces, the server and the agent. The server is deployed to a single location (or more than one if an HA environment is required) and will also install an embedded PostgreSQL database. An agent should be installed on any machine that monitoring should be performed on. The following describes the installation process and any customizations that have been made.

Server

sb11-hyperic/setup.sh
Server installation starts with setup.sh. It ensures that Java is installed and then calls hyperic-server/install.sh.

hyperic-server/install.sh
This is the main installation script for Hyperic Server. It will download the installation package from SourceForge and unzip it to the local filesystem. A custom properties file is copied to the installation directory.

A script called tune-os.sh is then run as the root user. This modifies some system settings that Hyperic needs in order to run.

The installer is then run which installs the Hyperic server software plus the embedded PostgreSQL instance.

A number of permissions settings are made.

The startup/shutdown script hyperic-server-init.sh is copied to the /etc/init.d folder and added to the RC.

After the script sleeps for a bit (it can take a minute or so to fully start the server), the agent is installed on the same machine.

At the end of the script Apache is installed so that we can create a virtual host to access the Hyperic UI from.

hyperic-server/hyperic-install.properties
These are a few customized properties that we want to set during installation of the server. We set some install directories, mail server information, and an admin username and password.

hyperic-server/hyperic-server-init.sh
This is the script copied to /etc/init.d to use to start and stop the Hyperic server.

hyperic-server/apache-default
This file is used by the Apache installation process to define the virtual host that will be used to get to the Hyperic UI.

Agent

hyperic-agent/install.sh
The main agent install script. The first thing it does is download the agent installation package from SourceForge and unzips it. Then a custom agent properties file is copied to the agent directory. Some string replacement is performed on the properties file to inject the correct Hyperic Server name and fully qualified domain name of the machine the agent is running on.

The Tomcat catalina-jmx-remote.jar file is downloaded and put into the classpath of the agent.

Some permissions are fixed and the agent is started.

hyperic-agent/hyperic-agent.properties
This is a custom properties file for running the agent. We use string replacement to put in the

correct name of the server the agent is running on and to put in the domain name of the Hyperic server to connect to. We also setup the correct username and password for connecting to the server and setup different defaults for how often runtime auto-discovery should occur.

hyperic-agent/hyperic-agent-init.sh
This script is copied to /etc/init.d and used to start and stop the agent process.

DNS Entries

In our test environment on EC2 we use Amazon's DNS service to create DNS entries for each of the applications so that it is easy to get to the various service endpoints regardless of how many times machines are provisioned.

Externalized property files

There are a number of externalized property files that are used by the M&A system.

M&A

email.properties
This contains all of the properties necessary to send emails from within M&A. M&A sends out emails for notification of Alerts. The properties should be clear in function.

hyperic-connection.properties
These are various properties needed to connect to Hyperic and for the functionality of the various integrating processes. Here are descriptions of what the properties are for:

hq.hostname, hq.port, hq.useSsl, hq.user, hq.password are all used to create the connection to the Hyperic server.

hq.jmx.url is the JMX URL that should be used to connect to one of the running JMX servers. The URL should use localhost for the servername since this URL is relative to where the agent is running. Since the intent is to run an agent on every node in the system, it will connect to the JMX server running locally.

hq.jmx.username and hq.jmx.password are the credentials to use to connect to the JMX server.

hq.tomcat.child.resource.type is the name of the Tomcat resource in Hyperic. This is used to determine whether or not we have found the correct Tomcat server while enumerating Hyperic resources. It is setup as a property so that it can be changed in the event of server upgrades or changes.

auto.approve.rate, load.metrics.rate, refresh.metrics.rate, and create.alert.rate are all times in milliseconds that refer to how often the periodic Hyperic processes will run.

mongo.properties
MongoDb connection properties.

rest-endpoints.properties
The mna.rest.service.endpoint and mna.jmx.service.endpoint properties in this file are used to define the service URLS that the M&A /rest module will use to talk to different parts of M&A.

TIB
mna_registration.properties
TIB uses these properties to define various things pertaining to registering with M&A using the common M&A client component.

tib.mna.description is a description of the component.

tib.mna.mnaRegistrationUrl, tib.mna.mnaUrl, and tib.mna.mnaDeleteComponentUrl are URL definitions to use various M&A services.

tib.mna.registrationDelayInSeconds is used to delay the registration by that many seconds.

tib.mna.healthMetricIntervalInSeconds is used to define how often this component will create its Availability metric. It is also used to define how often Hyperic will ask for the value of the Availability metric.

tib.mna.availability.metric.email is used to define the email address to send the auto-generated Availability alert to.

mongo.properties
MongoDb connection properties.

sftp.properties
Connection properties for the TIB SFTP site.