

jconsole monitoring

install openjdk-7-jdk or openjdk-6-jdk in server side of jconsole not in pentaho server

the jconsole can find in “/usr/lib/jvm/java-7-openjdk-amd64/bin”

sudo jconsole > shows a graphical window for monitoring

edit */opt/pentaho/biserver-ce/start-pentaho.sh*

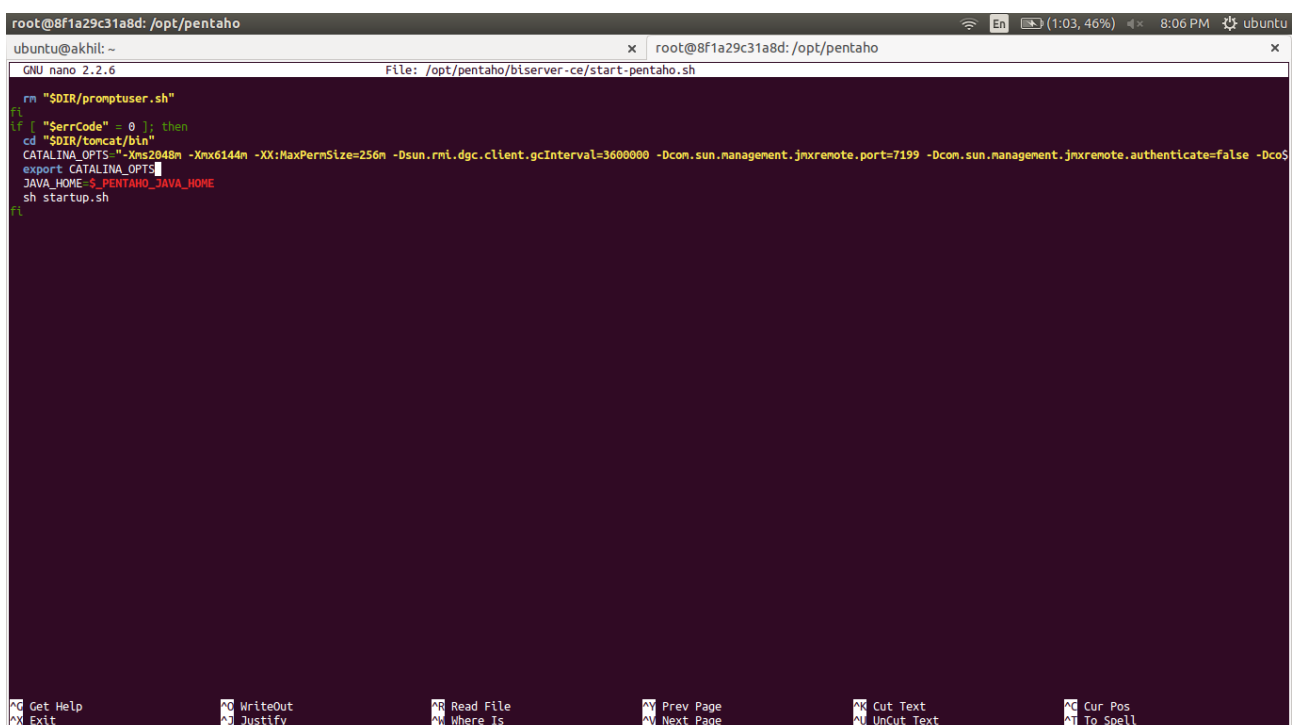
JMX Configuration

In order to access Mondrian metrics from a JMX client, you'll need to first configure the JVM process running biserver. For example, to set up unsecured remote JMX access on port 7199, add the following JVM parameters to the CATALINA_OPTS variable in start-pentaho.sh:

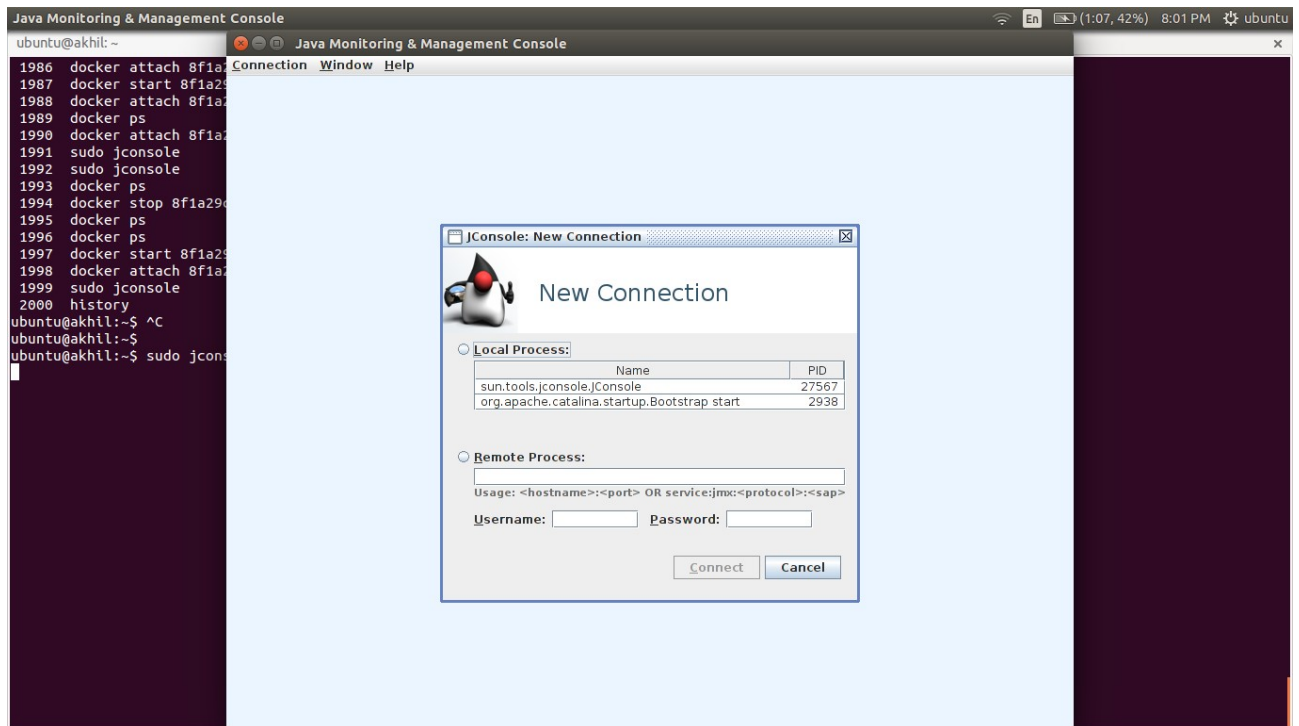
- Dcom.sun.management.jmxremote.port=7199
- Dcom.sun.management.jmxremote.authenticate=false
- Dcom.sun.management.jmxremote.ssl=false

Link:

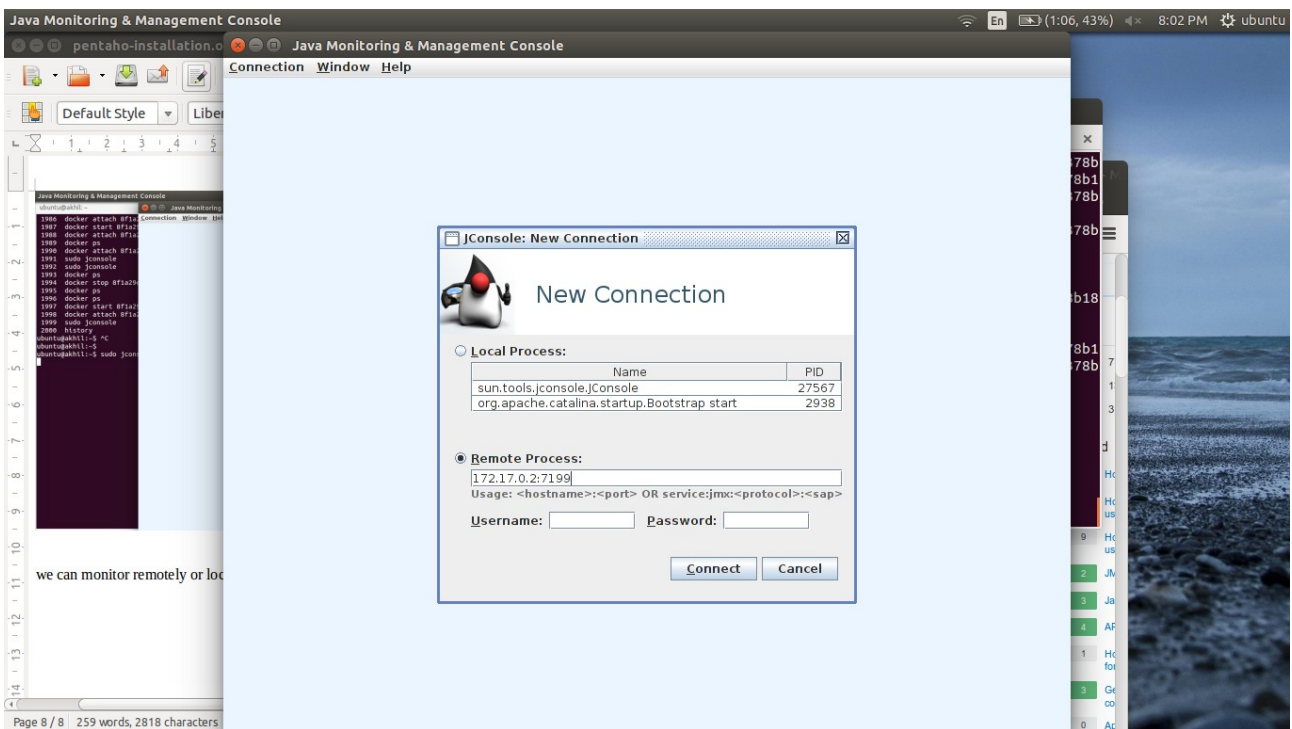
[http://wiki.pentaho.com/display/analysis/Monitoring+Mondrian+System+Metrics+with+Java+Management+Extensions+\(JMX\)](http://wiki.pentaho.com/display/analysis/Monitoring+Mondrian+System+Metrics+with+Java+Management+Extensions+(JMX))



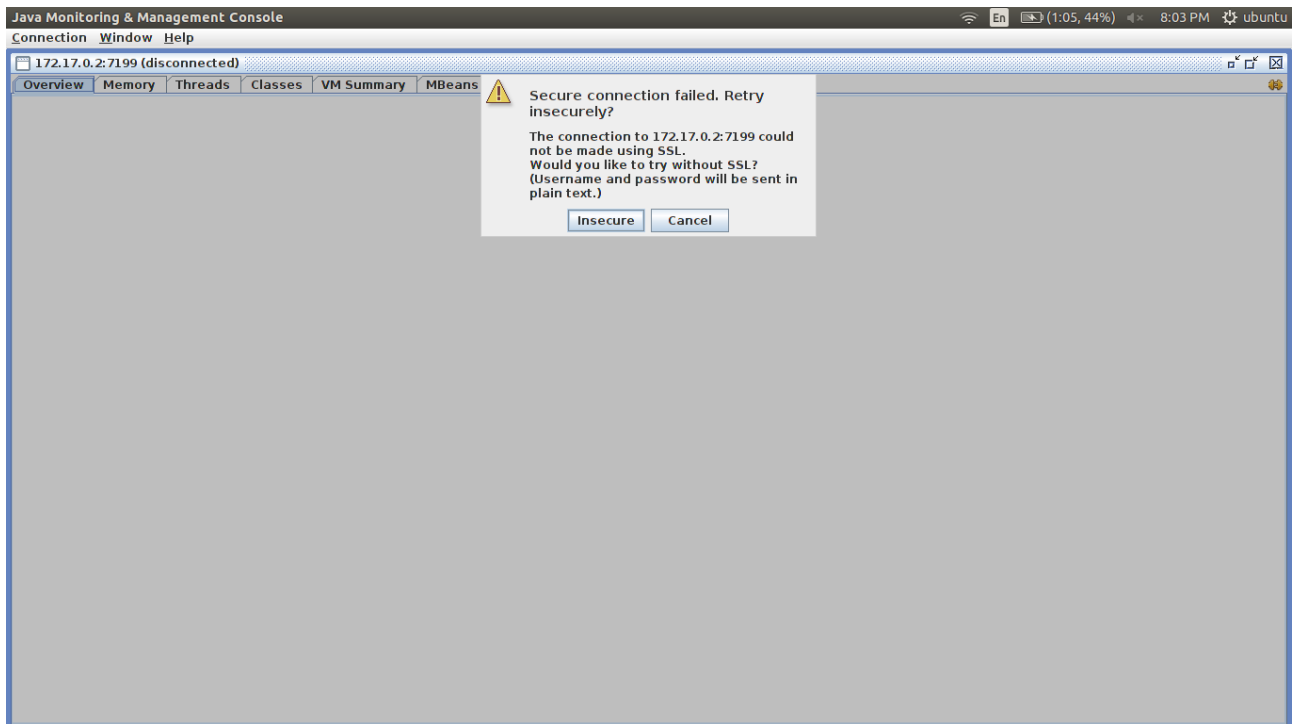
```
root@8f1a29c31a8d: /opt/pentaho
ubuntu@akhil: ~
GNU nano 2.2.6 File: /opt/pentaho/biserver-ce/start-pentaho.sh
rm "$DIR/promptuser.sh"
fi
if [ "$SerrCode" = 0 ]; then
cd "$DIR/toncat/bin"
CATALINA_OPTS="-Xms2048m -Xmx6144m -XX:MaxPermSize=256m -Dsun.rmi.dgc.client.gcInterval=3600000 -Dcom.sun.management.jmxremote.port=7199 -Dcom.sun.management.jmxremote.authenticate=false -DcoS
export CATALINA_OPTS
JAVA_HOME="$PENTAH0_JAVA_HOME"
sh startup.sh
fi
PC Get Help WriteOut Read File Prev Page Cut Text Cur Pos
XX Exit Justify XX Where Is XX Next Page UnCut Text To Spell
```



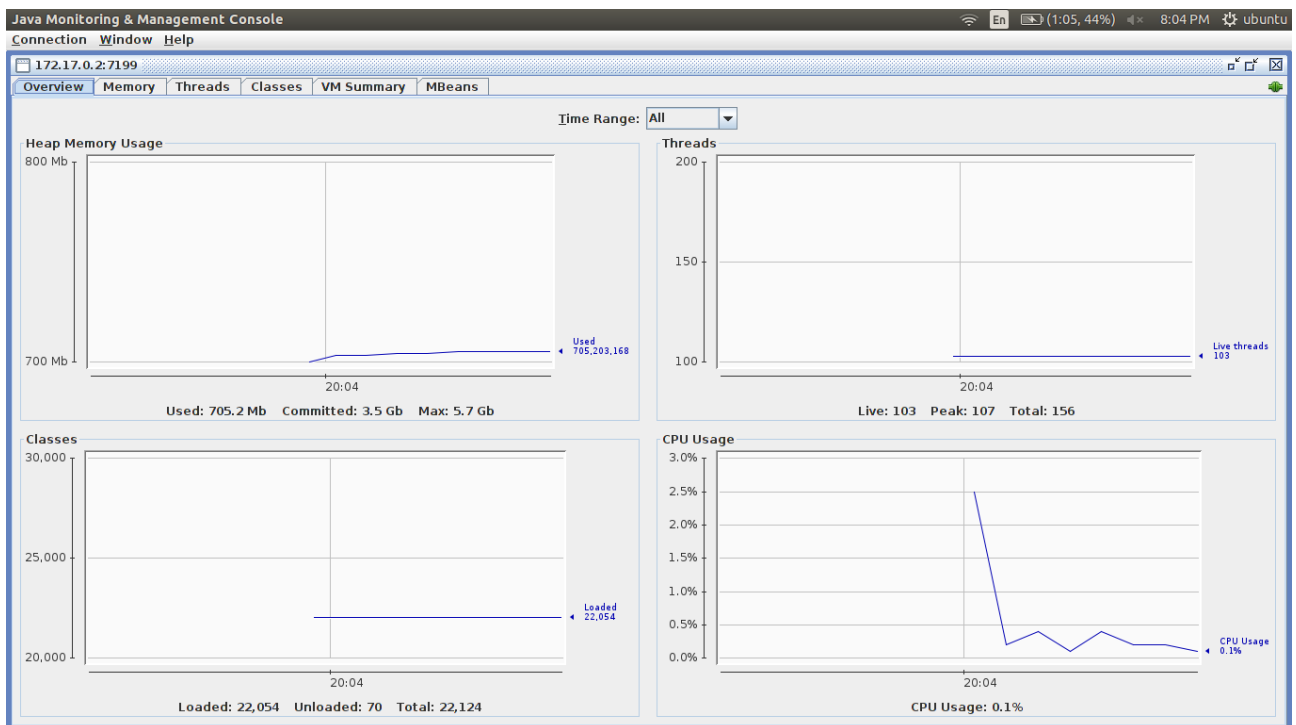
we can monitor remotely or locally



here i use remote because our pentaho running remotely 172.17.0.2 and i provide a port 7199 for jconsole accessing.



Click on insecure if configuration in pentaho is insecure



jconsole manual testing:
download a jar file and run it with

```
java -Dcom.sun.management.jmxremote \  
-Dcom.sun.management.jmxremote.port=9010 \  
-Dcom.sun.management.jmxremote.local.only=false \  
-Dcom.sun.management.jmxremote.authenticate=false \  
-Dcom.sun.management.jmxremote.ssl=false \  
-Djava.rmi.server.hostname=192.168.1.234 \  
-jar Notepad.jar
```

here i use **Notepad.jar** for testing,

so in jconsole provide 192.168.1.234:9010 so it will connected.

Links::>>>>

<http://stackoverflow.com/questions/856881/how-to-activate-jmx-on-my-jvm-for-access-with-jconsole>

<http://stackoverflow.com/questions/2591516/why-has-it-failed-to-load-main-class-manifest-attribute-from-a-jar-file>

<http://www.oracle.com/technetwork/articles/java/jconsole-1564139.html>

http://www.wellho.net/mouth/2081_Connecting-jconsole-remotely-the-principles.html

CATALINA_OPTS parameters

-Xmx for maximum heap size, and

-Xms for initial heap size

Increase maximum perm size for web base applications to 4x the default amount
-XX:MaxPermSize

eg: -XX:PermSize=128m
-XX:MaxPermSize=512m

6 Common Errors in Setting Java Heap Size

Two JVM options are often used to tune JVM heap size: `-Xmx` for maximum heap size, and `-Xms` for initial heap size. Here are some common mistakes I have seen when using them:

- Missing `m`, `M`, `g` or `G` at the end (they are case insensitive). For example,

```
java -Xmx128 BigApp
java.lang.OutOfMemoryError: Java heap space
```

The correct command should be: `java -Xmx128m BigApp`. To be precise, `-Xmx128` is a valid setting for very small apps, like `HelloWorld`. But in real life, I guess you really mean `-Xmx128m`

- Extra space in JVM options, or incorrectly use `=`. For example,

```
java -Xmx 128m BigApp
Invalid maximum heap size: -Xmx
Could not create the Java virtual machine.
java -Xmx=512m HelloWorld
Invalid maximum heap size: -Xmx=512m
Could not create the Java virtual machine.
```

The correct command should be `java -Xmx128m BigApp`, with no whitespace nor `=`. `-X` options are different than `-Dkey=value` system properties, where `=` is used.

- Only setting `-Xms` JVM option and its value is greater than the default maximum heap size, which is `64m`. The default minimum heap size seems to be `0`. For example,

```
java -Xms128m BigApp
Error occurred during initialization of VM
Incompatible initial and maximum heap sizes specified
```

The correct command should be `java -Xms128m -Xmx128m BigApp`. It's a good idea to set the minimum and maximum heap size to the same value. In any case, don't let the minimum heap size exceed the maximum heap size.

- Heap size is larger than your computer's physical memory. For example,

```
java -Xmx2g BigApp
Error occurred during initialization of VM
Could not reserve enough space for object heap
Could not create the Java virtual machine.
```

The fix is to make it lower than the physical memory: `java -Xmx1g BigApp`

- Incorrectly use `mb` as the unit, where `m` or `M` should be used instead.

```
java -Xms256mb -Xmx256mb BigApp
Invalid initial heap size: -Xms256mb
Could not create the Java virtual machine.
```

- The heap size is larger than JVM thinks you would ever need. For example,

```
java -Xmx256g BigApp
```

```
Invalid maximum heap size: -Xmx256g
The specified size exceeds the maximum representable size.
Could not create the Java virtual machine.
```

The fix is to lower it to a reasonable value: `java -Xmx256m BigApp`

- The value is not expressed in whole number. For example,

```
java -Xmx0.9g BigApp
Invalid maximum heap size: -Xmx0.9g
Could not create the Java virtual machine.
```

The correct command should be `java -Xmx928m BigApp`

links : >>>

<http://javahowto.blogspot.in/2006/06/6-common-errors-in-setting-java-heap.html>

<https://gist.github.com/terrancesnyder/986029>

<https://www.experts-exchange.com/questions/23159662/Difference-between-CATALINA-OPTS-and-JAVA-OPTS-in-TOMCAT.html>