Installation of Pentaho components and setting up connections

Pentaho can download from http://sourceforge.net/projects/pentaho/files/Business%20Intelligence%20Server/6.1/biserver-ce-6.1.0.1-196.zip/download.

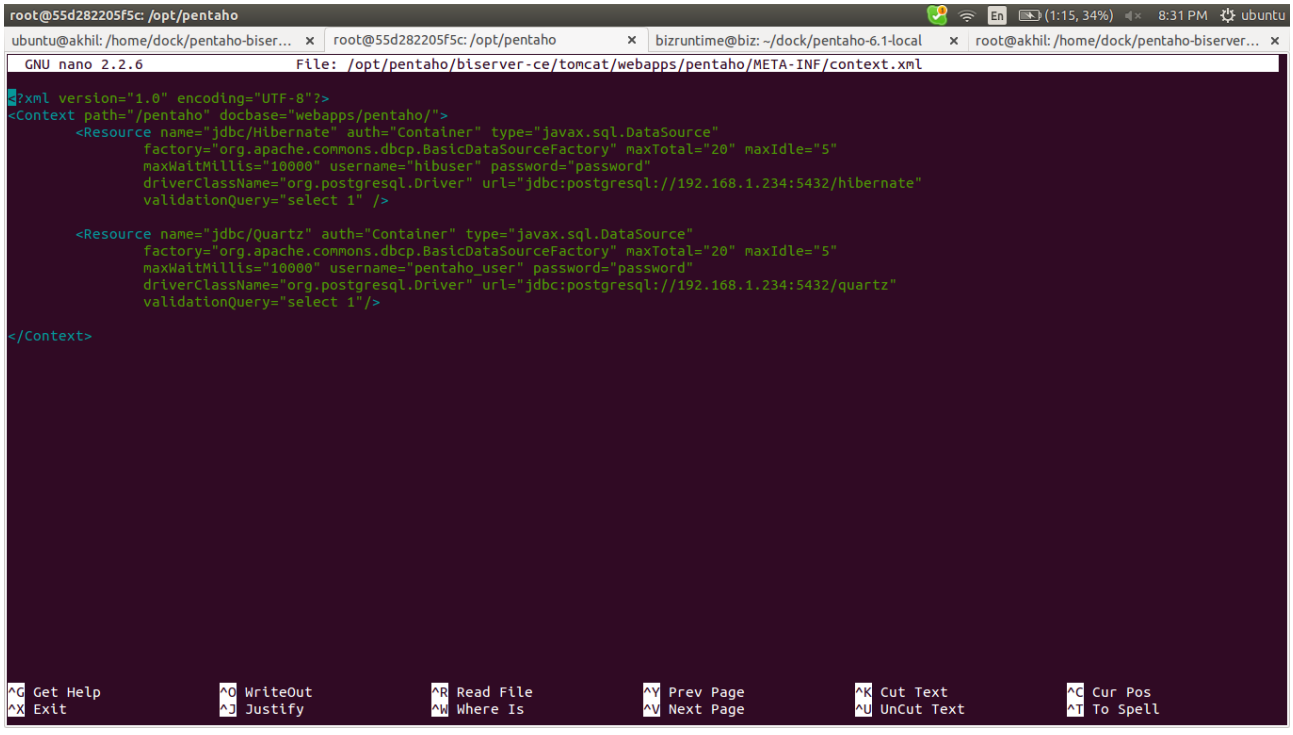
1) installation steps are in dockerfile

a) for local database server
links : git clone -b master **https://github.com/akhilrajmailbox/pentaho-6.1-local.git**

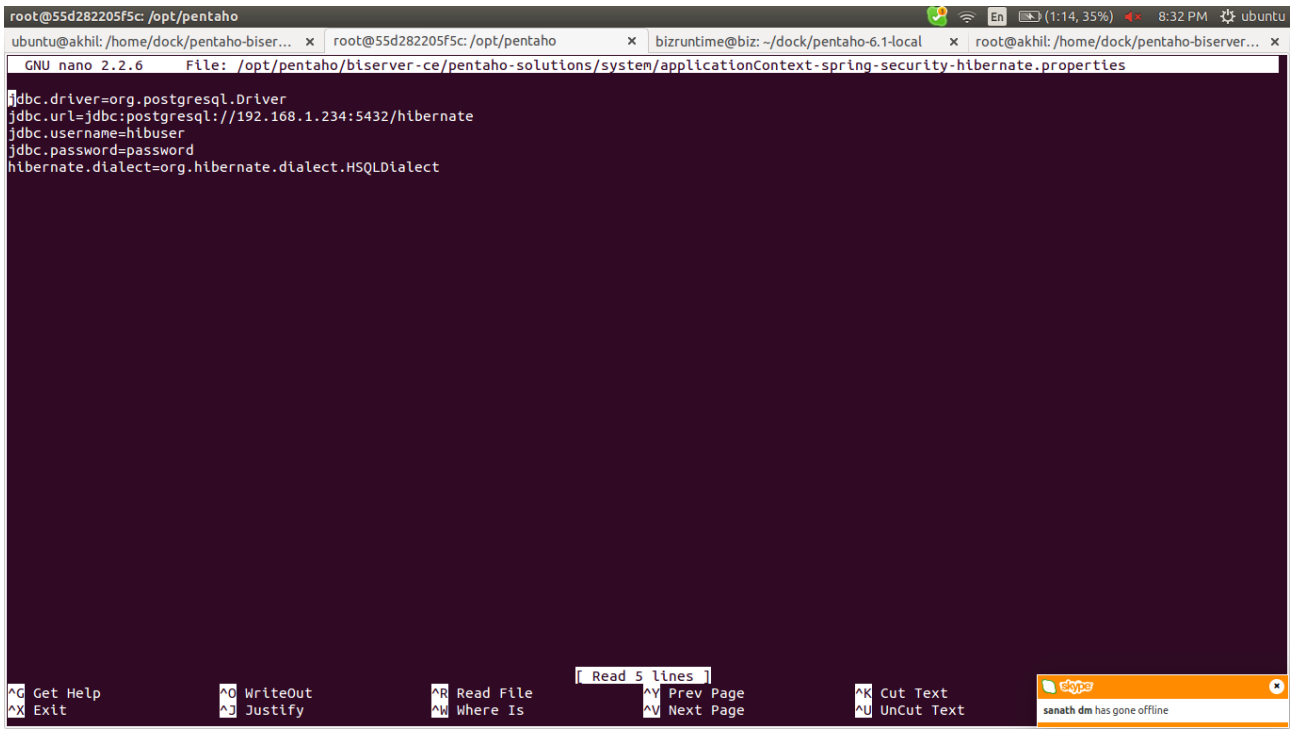
b) for remote database server
links : git clone -b master-remote-db **https://github.com/akhilrajmailbox/pentaho-6.1-local.git**

configuration files which i changed for remote database (192.168.1.234) are given below
for make local database “192.168.1.234” changed to “localhost”

2) ***/opt/pentaho/biserver-ce/tomcat/webapps/pentaho/META-INF/context.xml***

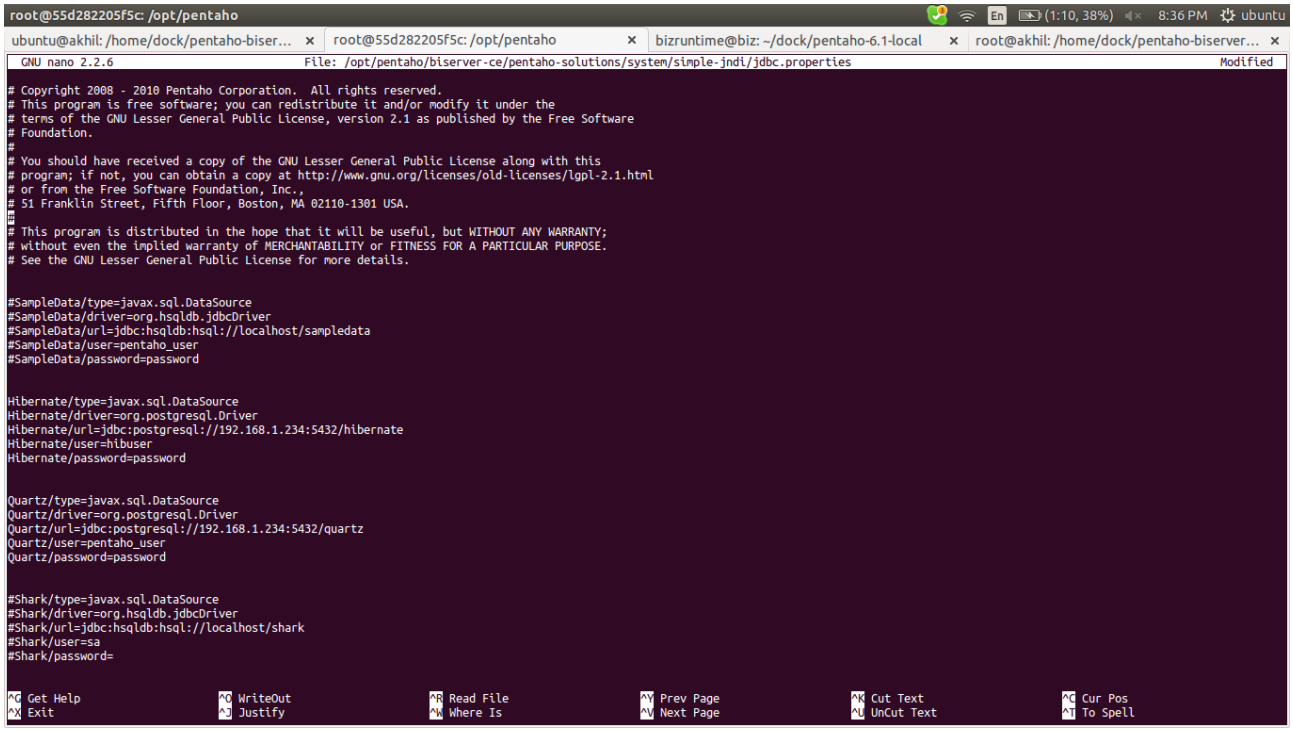


3) ***/opt/pentaho/biserver-ce/pentaho-solutions/system/applicationContext-spring-security-hibernate.properties***

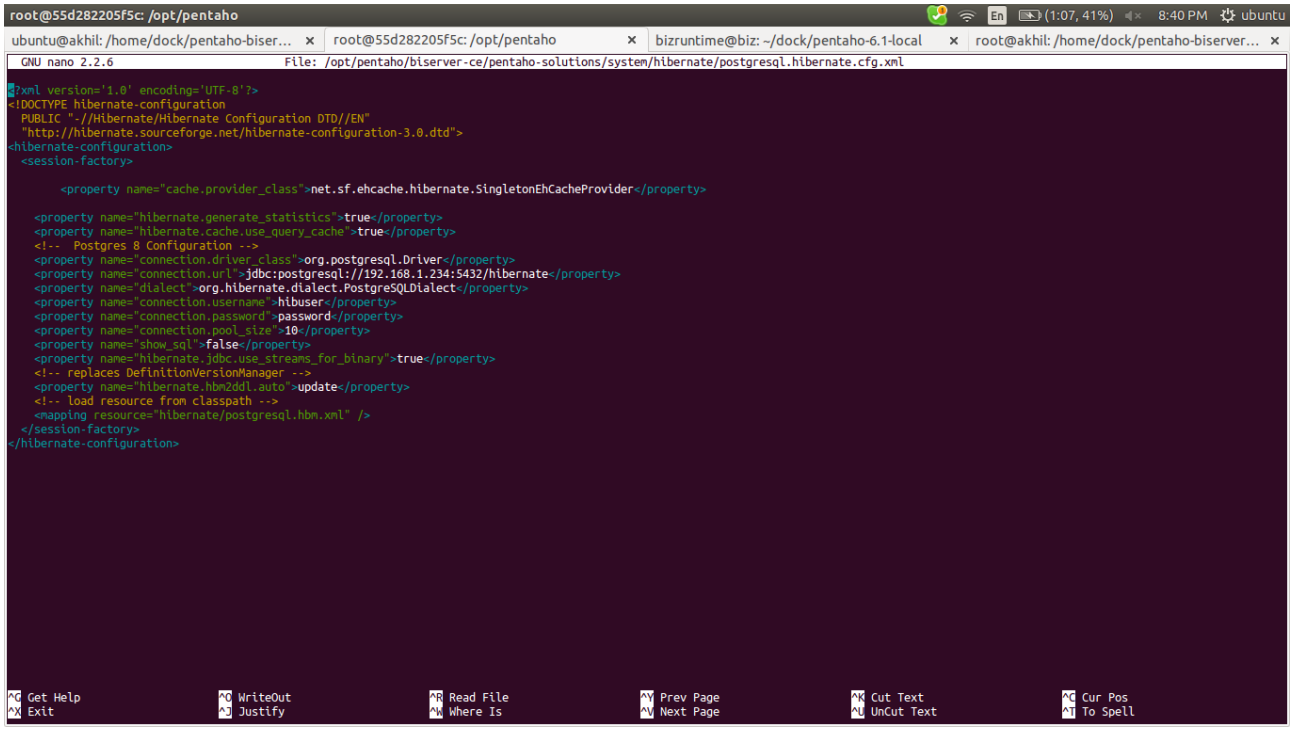


4) */opt/pentaho/biserver-ce/pentaho-solutions/system/hibernate/hibernate-settings.xml*

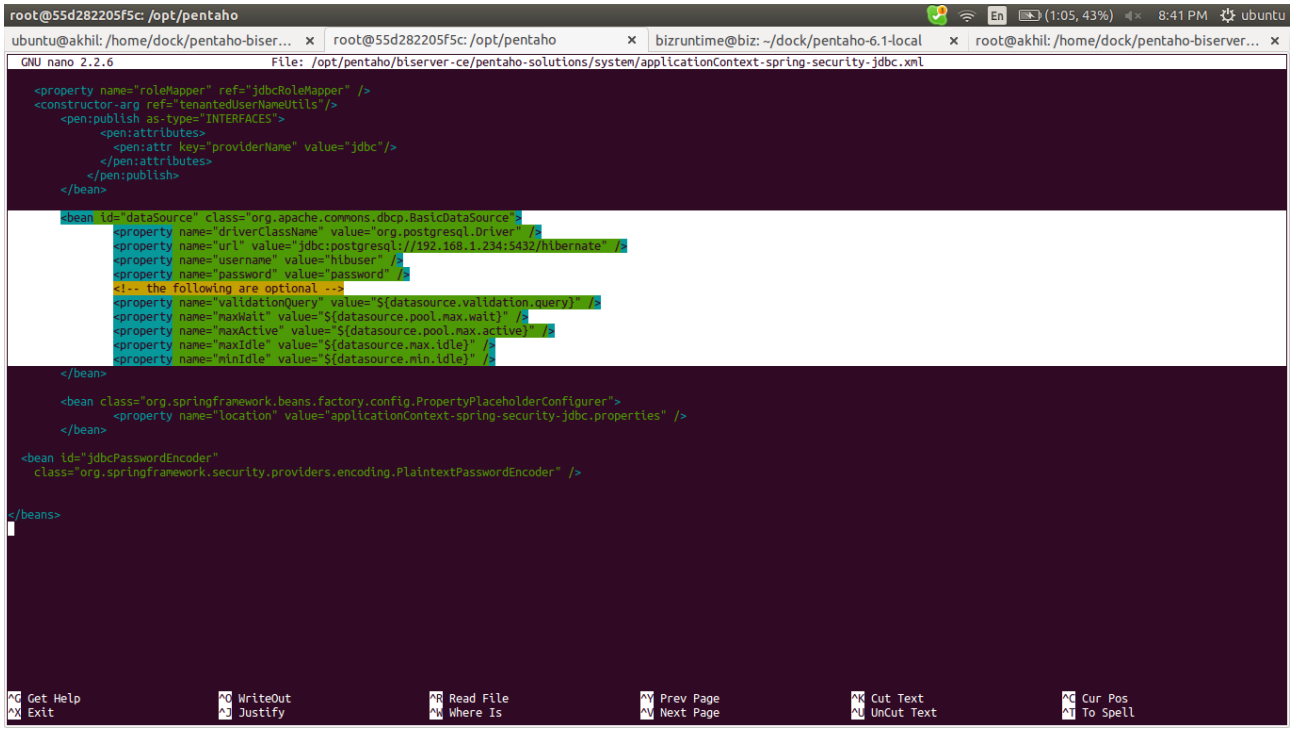
5) */opt/pentaho/biserver-ce/pentaho-solutions/system/simple-jndi/jdbc.properties*



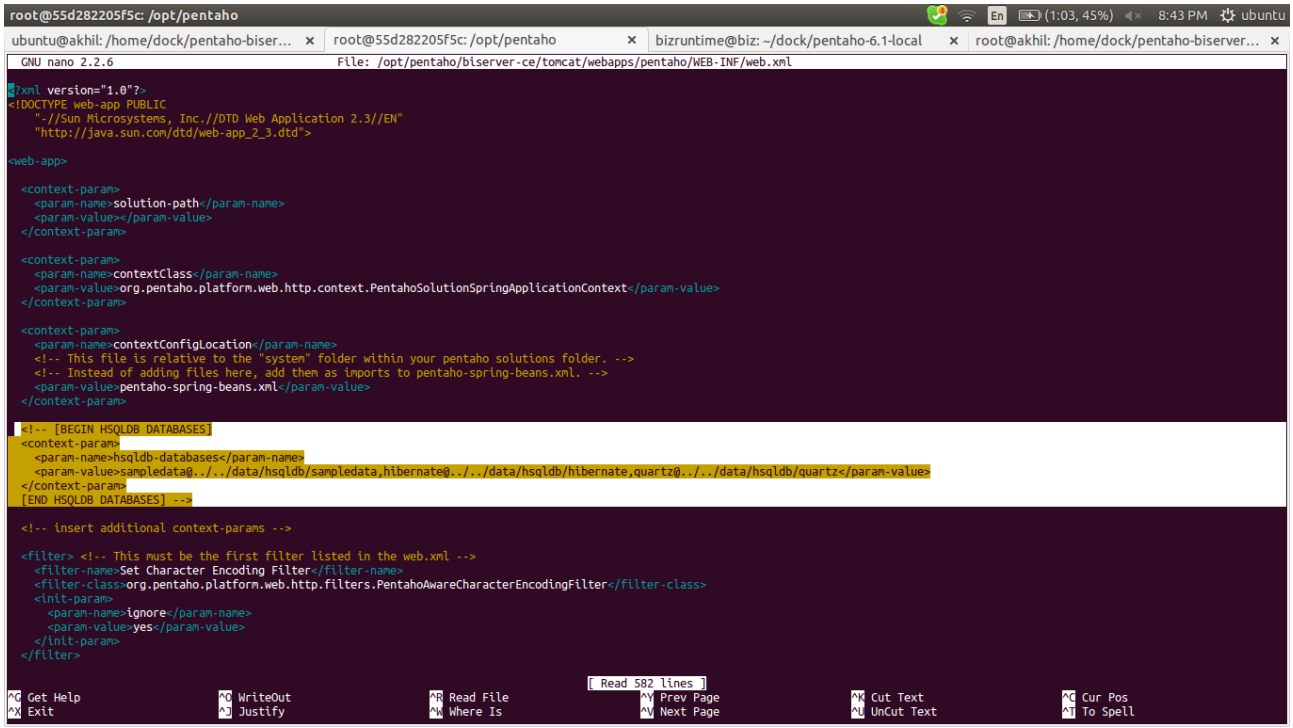
6) */opt/pentaho/biserver-ce/pentaho-solutions/system/hibernate/postgresql.hibernate.cfg.xml*



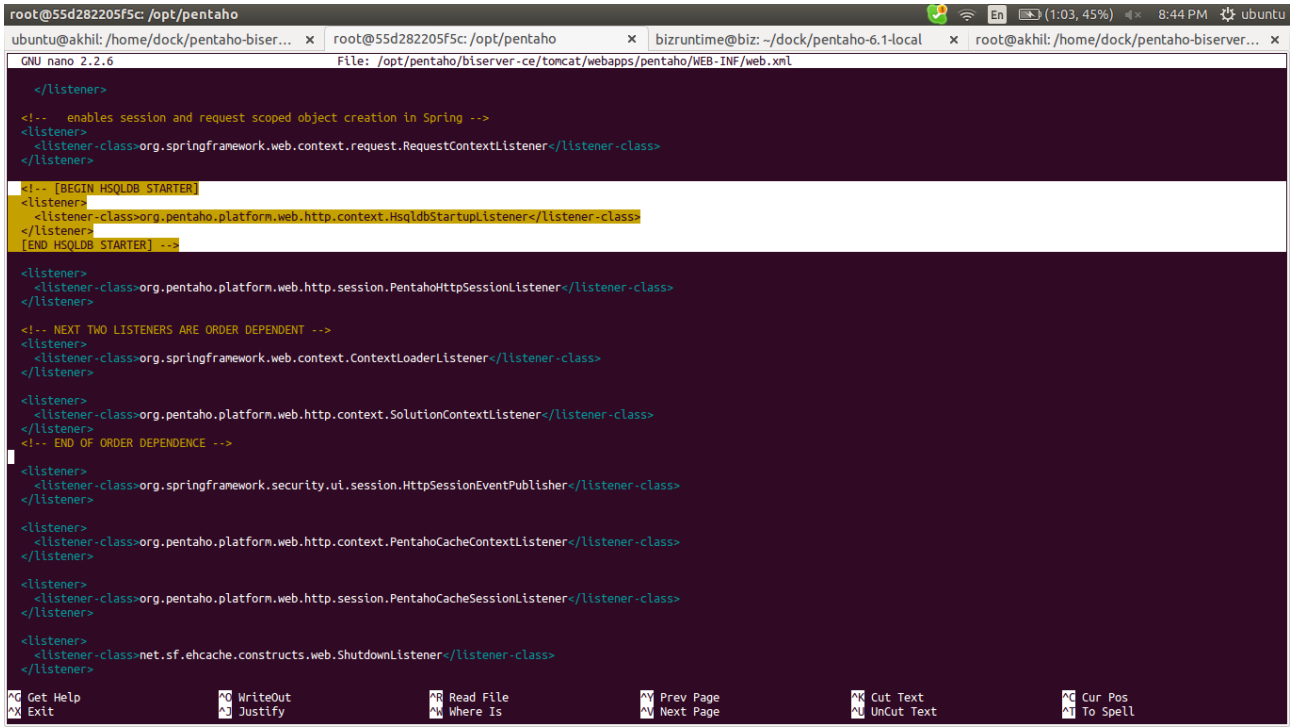
7) */opt/pentaho/biserver-ce/pentaho-solutions/system/applicationContext-spring-security-jdbc.xml*



8) Edit the file */opt/pentaho/biserver-ce/tomcat/webapps/pentaho/WEB-INF/web.xml* to prevent HSQLDB to start



change
<!-- [BEGIN HSQLDB DATABASES] --> to <!-- [BEGIN HSQLDB DATABASES]
and
<!-- [END HSQLDB DATABASES] --> to [END HSQLDB DATABASES] -->



change

<!-- [BEGIN HSQLDB STARTER] --> to <!-- [BEGIN HSQLDB STARTER]
and
<!-- [END HSQLDB STARTER] --> to [END HSQLDB STARTER] -->

9) Make the .sh files executable

```
sudo chmod +x /opt/pentaho/biserver-ce/*.sh
```

Make the .sh files executable

```
/opt/pentaho/biserver-ce/start-pentaho.sh
```

Go to

http://ipaddress-pentaho-server:8080

10) refer links for auto start up

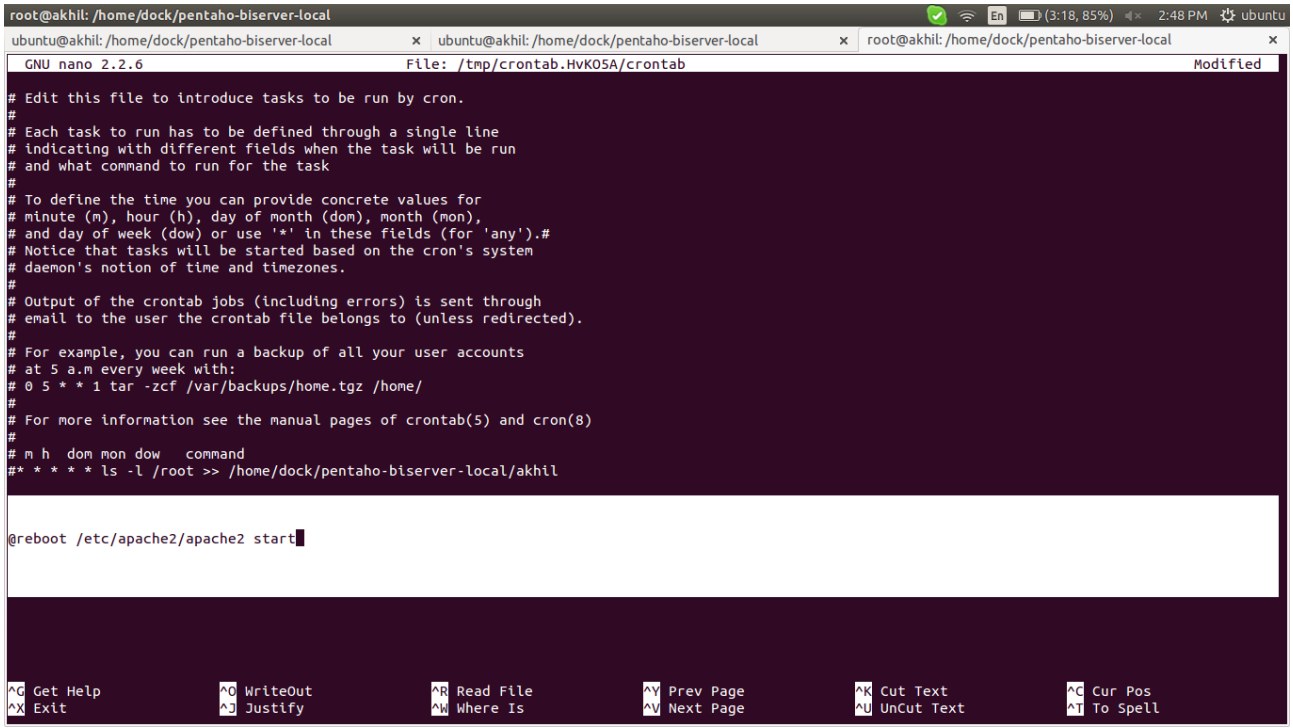
a) cron job

Use a crontab option to make your script run after reboot,
you can do it by adding @reboot code in cron
@reboot /opt/pentaho/biserver-ce/start-pentaho.sh

http://askubuntu.com/questions/290099/how-to-run-a-script-during-boot-as-root

http://www.cyberciti.biz/faq/howto-linux-unix-start-restart-cron/

eg: apache start up script



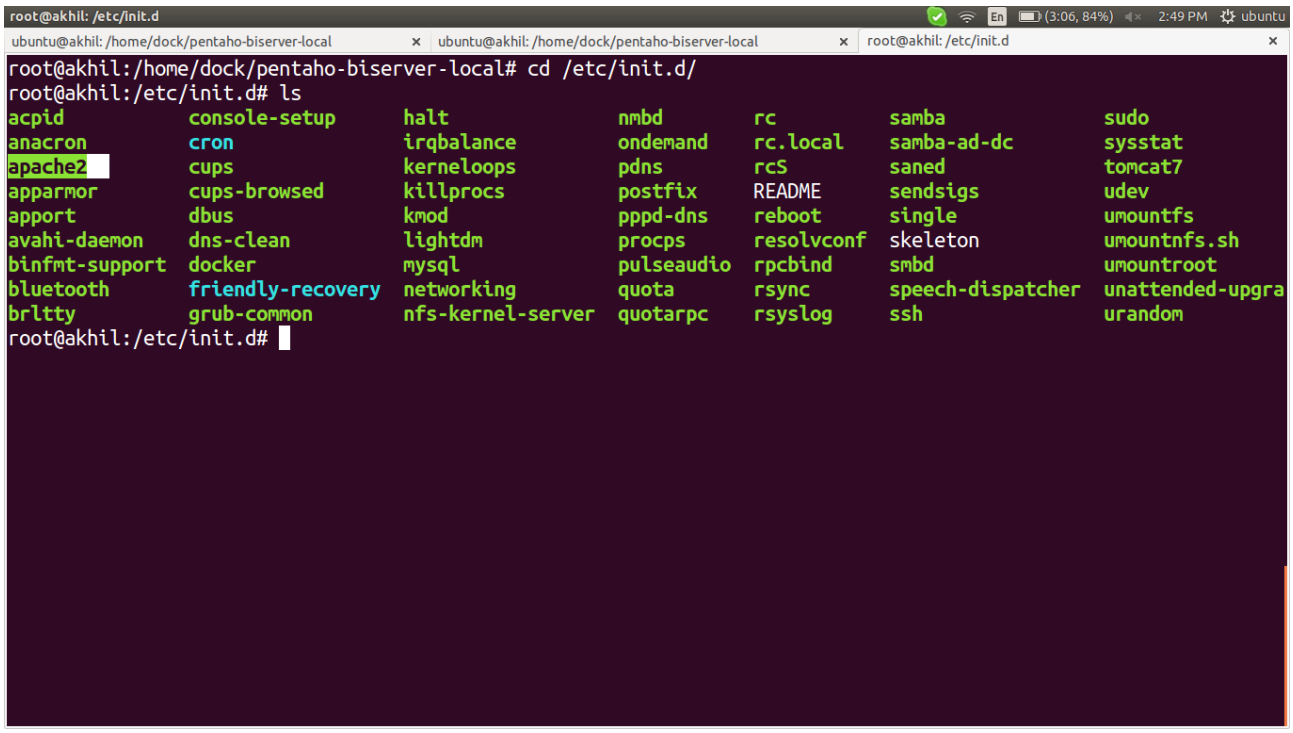
here /etc/apache2/apache2 is the path to that script

http://askubuntu.com/questions/814/how-to-run-scripts-on-start-up

b) put service in /etc/init.d

https://www.digitalocean.com/community/tutorials/how-to-configure-a-linux-service-to-start-automatically-after-a-crash-or-reboot-part-1-practical-examples

https://embraceubuntu.com/2005/09/07/adding-a-startup-script-to-be-run-at-bootup/



copy that script to /etc/init.d/

eg : cp -r /etc/apache2/apache2 /etc/init.d/

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

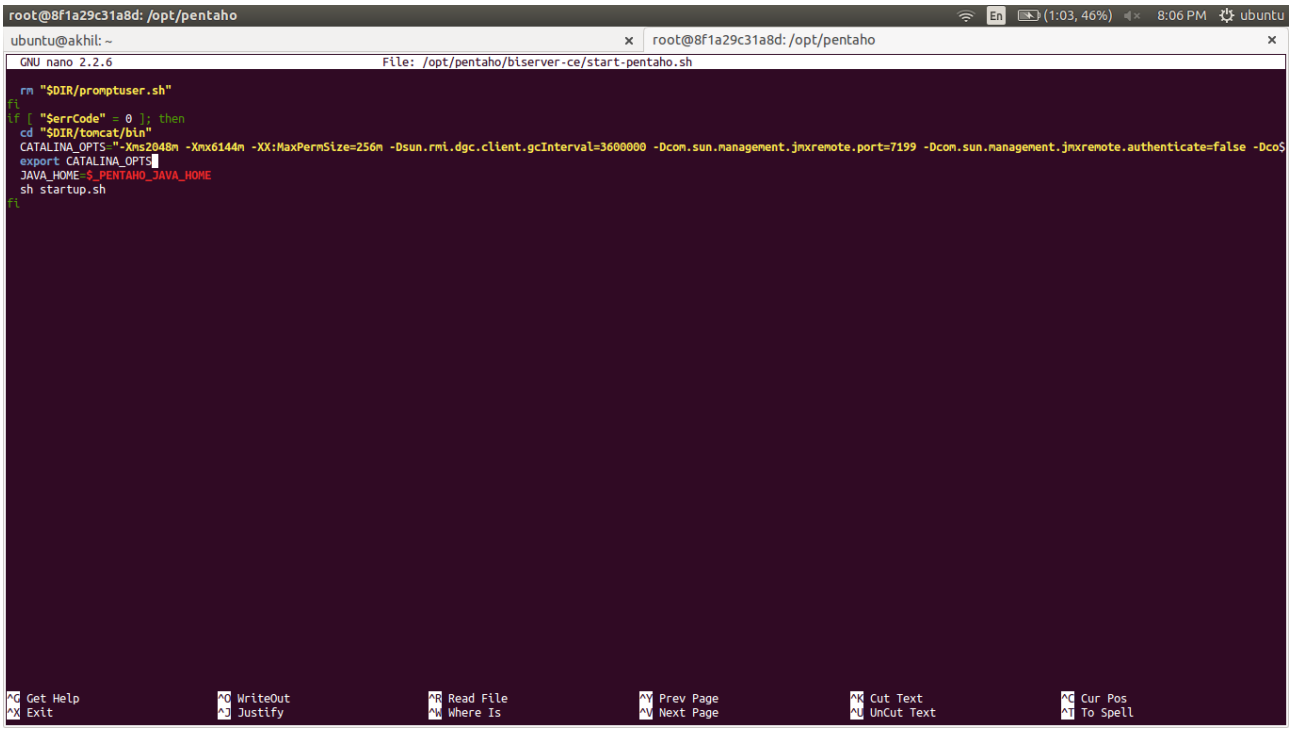
edit */opt/pentaho/biserver-ce/start-pentaho.sh*

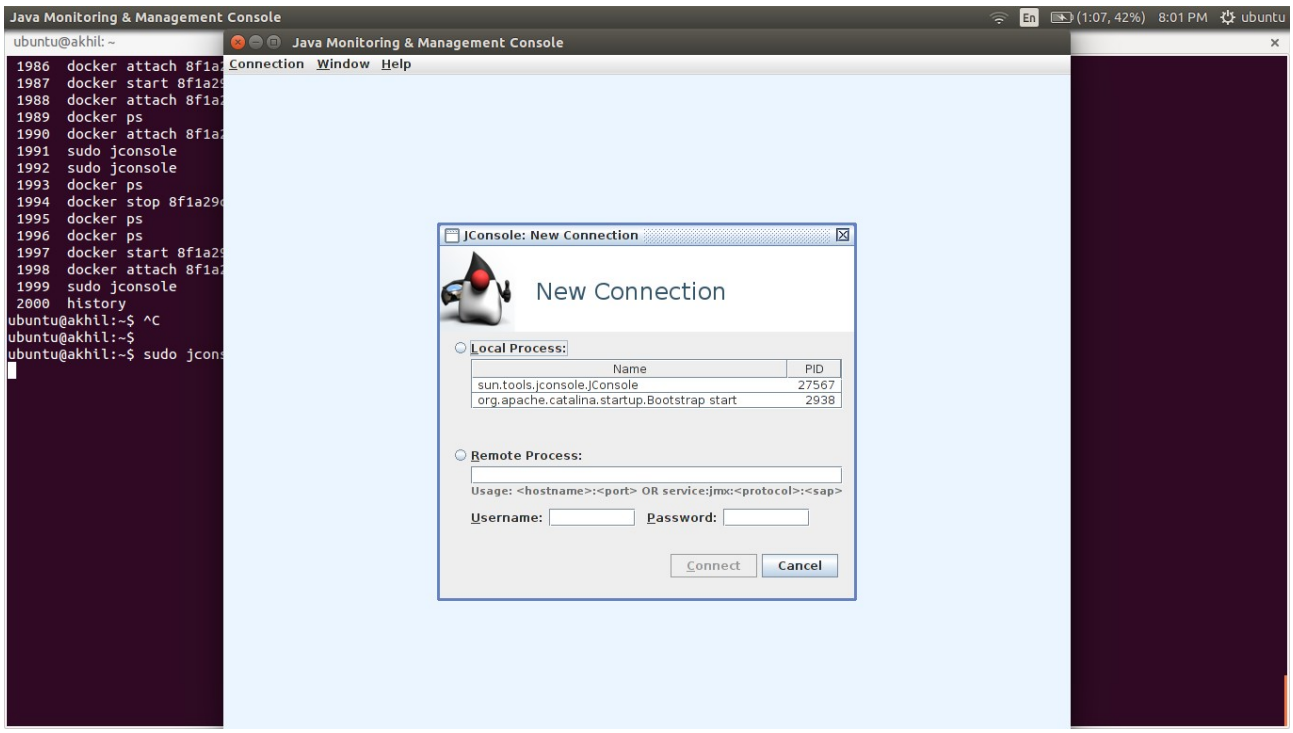
JMX Configuration
In order to access Mondrian metrics from a JMX client, you'll need to first configure the JVM process running biserver. For example, to set up unsecured remote JMX access on port 7199, add the following JVM parameters to the CATALINA_OPTS variable in start-pentaho.sh:

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

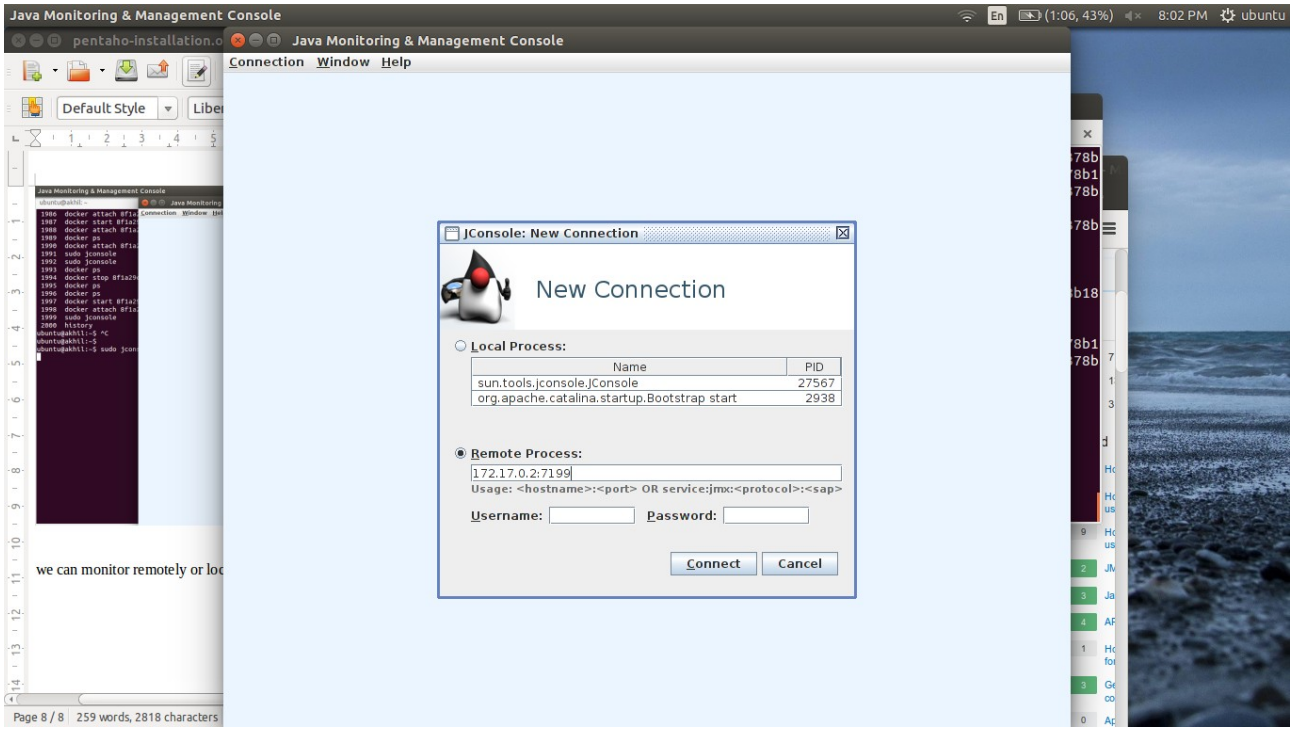
Link:

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

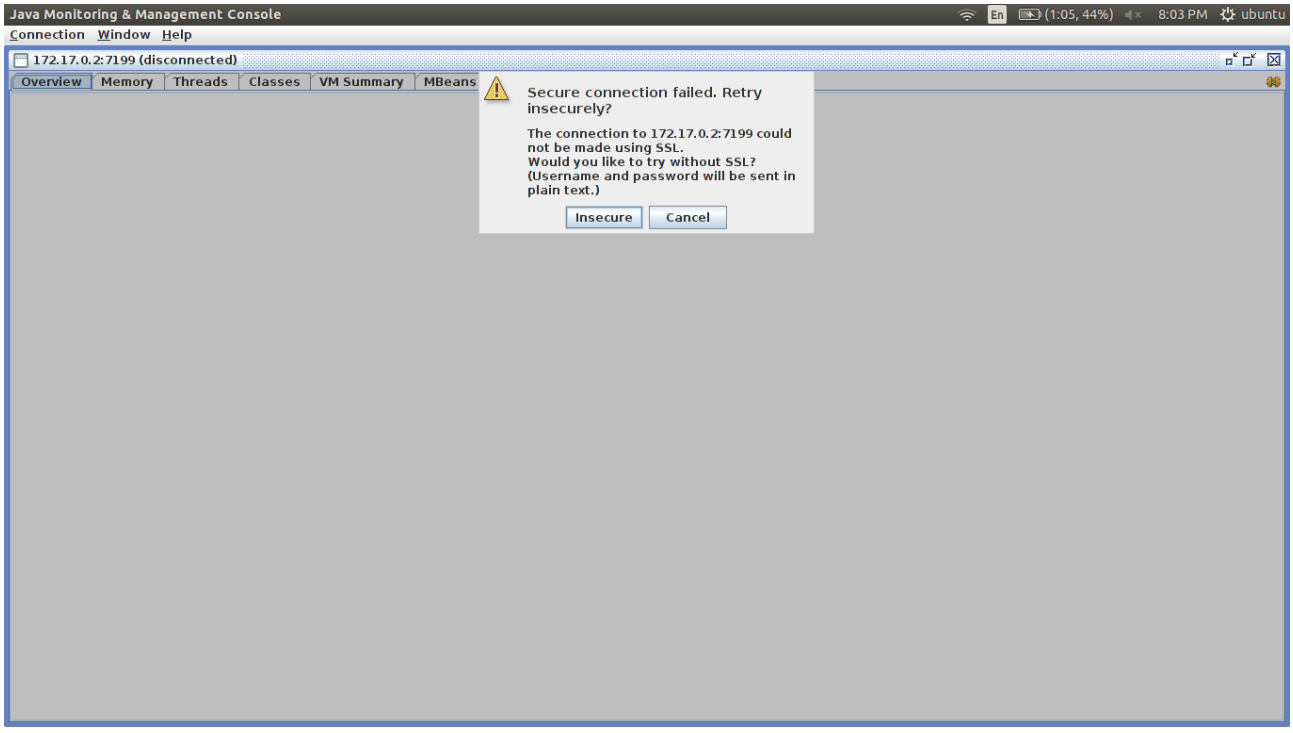




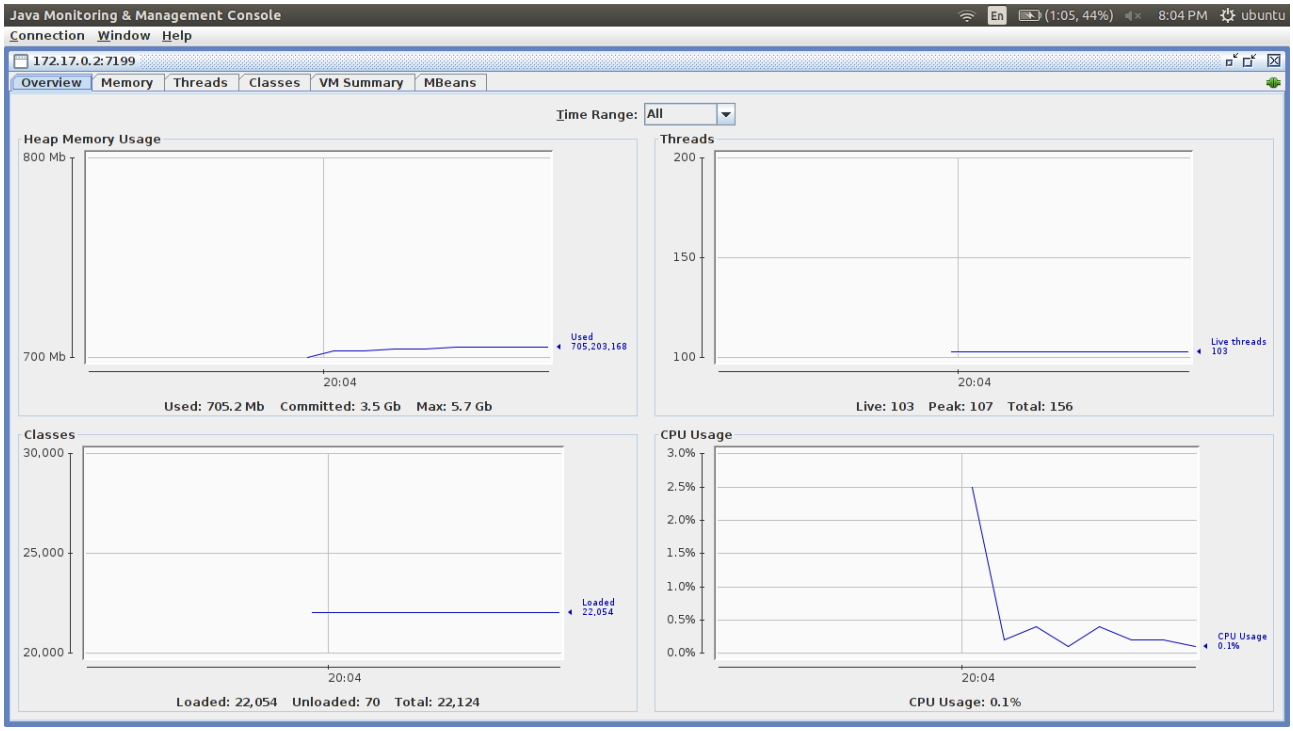
we can monitor remotely or locally



here i use remote because our pentaho running remotly 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