14 Supervision JMX

Aperçu

La supervision JMX peut être utilisée pour surveiller les compteurs JMX d'une application Java.

La supervision JMX a un support natif sous Zabbix sous la forme d'un démon Zabbix appelé “Zabbix Java gateway”, introduit depuis Zabbix 2.0.

Pour récupérer la valeur d'un compteur JMX particulier sur un hôte, le serveur Zabbix interroge la **passerelle Java** Zabbix, qui à son tour utilise l'API de gestion JMX pour interroger l'application à distance.

Pour plus de détails et des informations sur l'installation, voir la section de la passerelle Java Zabbix.

La communication entre la passerelle Java et l'application JMX surveillée ne doit pas passée par un pare-feu.

Activation de la supervision JMX à distance pour l'application Java

Une application Java ne nécessite pas l’installation de logiciel supplémentaire, mais elle doit être démarrée avec les options de ligne de commande spécifiées ci-dessous pour prendre en charge la surveillance JMX distante.

Au strict minimum, si vous souhaitez simplement commencer en surveillant une application Java simple sur un hôte local sans sécurité, lancez-le avec les options suivantes :

```
java \
-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.port=12345 \
-Dcom.sun.management.jmxremote.authenticate=false \
-Dcom.sun.management.jmxremote.ssl=false \
-jar /usr/share/doc/openjdk-6-jre-headless/demo/jfc/Notepad/Notepad.jar
```

Cela fait en sorte que Java écoute les connexions JMX entrantes sur le port 12345, à partir de l'hôte local uniquement, et lui dit de ne pas exiger d'authentification ou de SSL.

Si vous souhaitez autoriser les connexions sur une autre interface, définissez le paramètre -Djava.rmi.server.hostname sur l'adresse IP de cette interface.

Si vous souhaitez être plus strict sur la sécurité, il existe de nombreuses autres options Java disponibles. L'exemple suivant démarre l'application avec un ensemble d'options plus polyvalent et l'ouvre sur un réseau plus large, pas seulement sur l'hôte local.

```
java \
-Djava.rmi.server.hostname=192.168.3.14 \
-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.port=12345 \
-Dcom.sun.management.jmxremote.authenticate=true \
-Dcom.sun.management.jmxremote.password.file=/etc/java-6-
```

```
openjdk/management/jmxremote.password \
-Dcom.sun.management.jmxremote.access.file=/etc/java-6-
openjdk/management/jmxremote.access \
-Dcom.sun.management.jmxremote.ssl=true \
-Djavax.net.ssl.keyStore=$YOUR_KEY_STORE \
-Djavax.net.ssl.keyStorePassword=$YOUR_KEY_STORE_PASSWORD \
-Djavax.net.ssl.trustStore=$YOUR_TRUST_STORE \
-Djavax.net.ssl.trustStorePassword=$YOUR_TRUST_STORE_PASSWORD \
-Dcom.sun.management.jmxremote.ssl.need.client.auth=true \
-jar /usr/share/doc/openjdk-6-jre-headless/demo/jfc/Notepad/Notepad.jar
```

La plupart (si ce n'est la totalité) de ces paramètres peuvent être spécifiés dans /etc/java-6-openjdk/management/management.properties (ou à l'endroit où ce fichier se trouve sur votre système).

Notez que si vous souhaitez utiliser SSL, vous devez modifier le script startup.sh en ajoutant des options `-Djavax.net.ssl.*` à la passerelle Java, afin qu'elle sache où trouver les magasins de clés et de confiance.

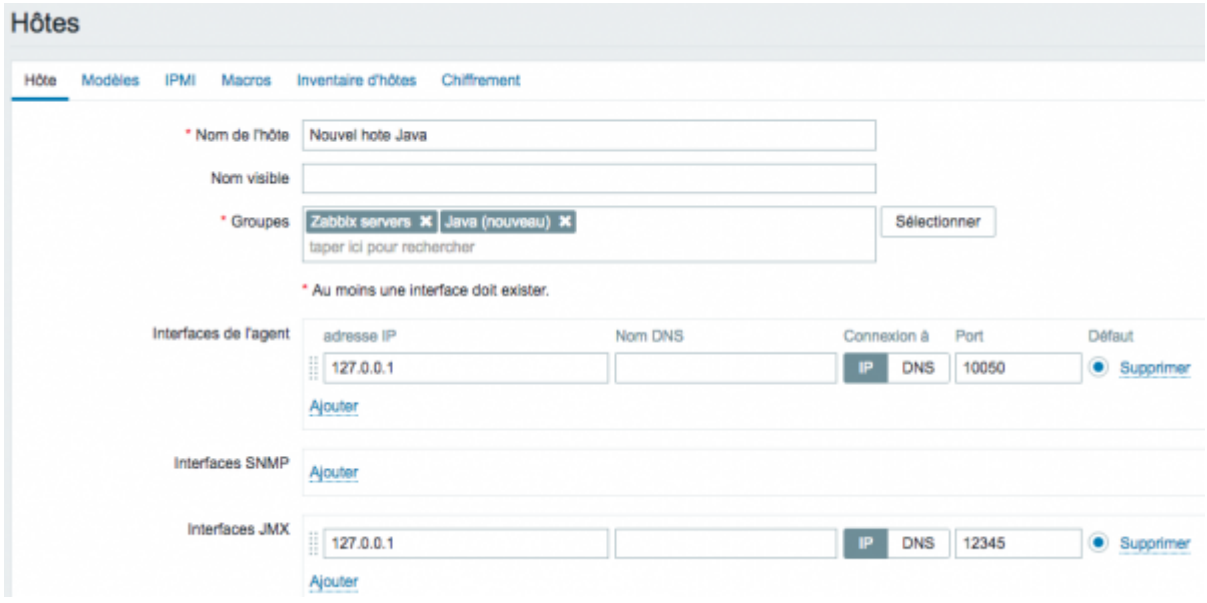
Voir Surveillance et gestion à l'aide de JMX pour une description détaillée.

Configuration des interfaces et des éléments JMX dans l'interface Zabbix

Avec la passerelle Java en cours d'exécution, le serveur sachant où trouver la passerelle et une application Java démarrée avec la prise en charge de la surveillance JMX à distance, il est temps de configurer les interfaces et les éléments dans l'interface web de Zabbix.

Configuration de l'interface JMX

Commencez par créer une interface de type JMX sur l'hôte :

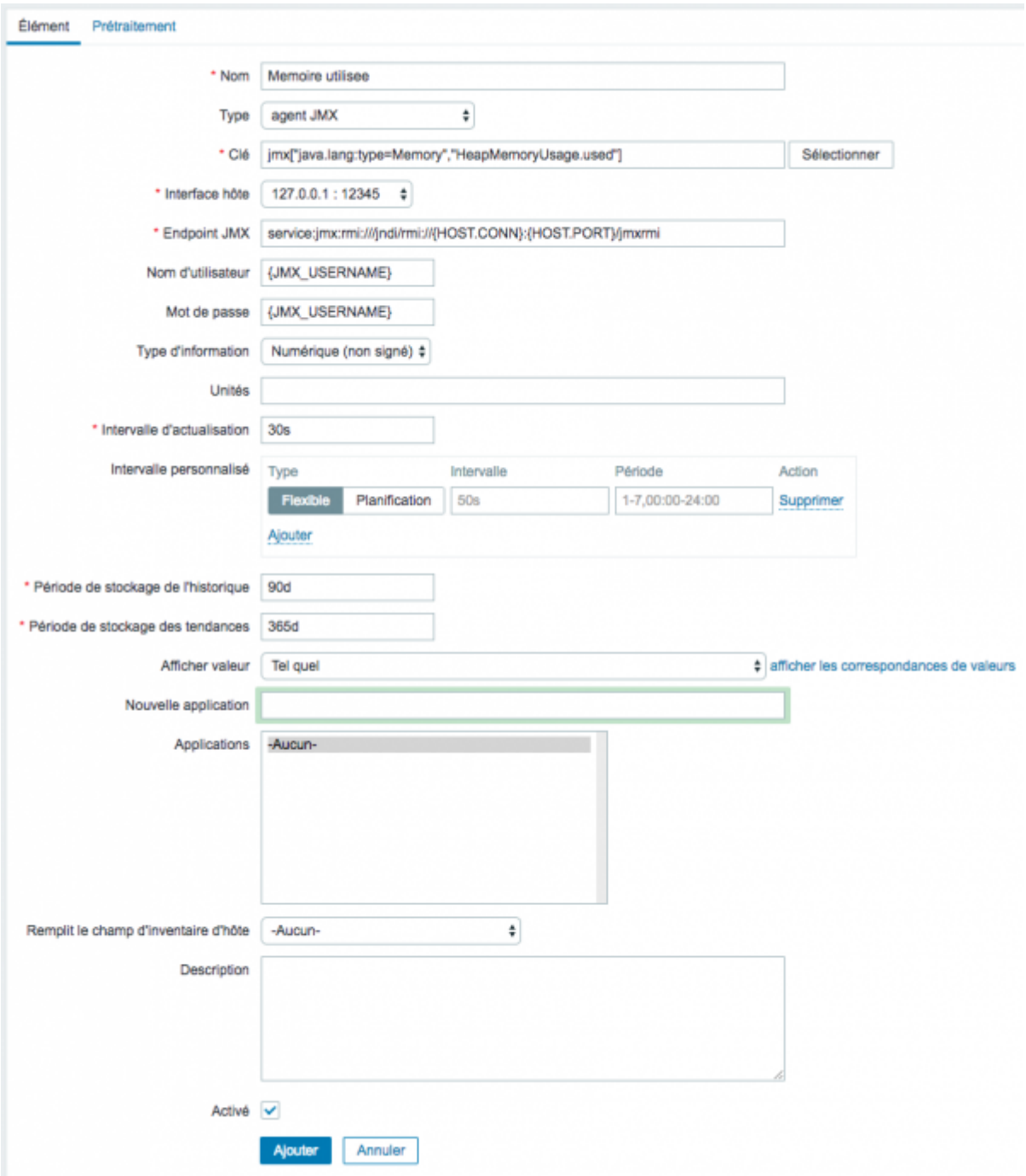


Tous les champs de saisie obligatoires sont marqués d'un astérisque rouge.

Ajout d'un élément d'agent JMX

Pour chaque compteur JMX, vous souhaitez ajouter un élément d'agent JMX à cette interface.

La clé dans la capture d'écran ci-dessous indique `jmx["java.lang:type=Memory","HeapMemoryUsage.used"]`.



Tous les champs de saisie obligatoires sont marqués d'un astérisque rouge.

Les champs nécessitant des informations spécifiques pour les éléments JMX sont :

Type	Définissez **agent JMX** ici.

Clé	La clé de l'élément `jmx[]` contient 2 paramètres : **object name** - le nom de l'objet d'un MBean ; **attribute name** - le nom de l'attribut MBean avec des noms de champs de données composites facultatifs séparés par des points. Voir ci-dessous pour plus de détails sur les clés d'éléments JMX. Depuis Zabbix 3.4, vous pouvez découvrir les attributs MBeans et MBean à l'aide de l'élément de découverte de bas niveau `jmx.discovery[]`.
JMX endpoint	Vous pouvez spécifier un point de terminaison JMX personnalisé. Assurez-vous que les paramètres de connexion du noeud final JMX correspondent à l'interface JMX. Cela peut être réalisé en utilisant des macros {HOST.*} Comme dans le point de terminaison JMX par défaut. Ce champ est supporté depuis 3.4.0. Les macros {HOST.*} et les macros utilisateur sont supportées.
Nom d'utilisateur	Indiquez le nom d'utilisateur si vous avez configuré l'authentification sur votre application Java. Les macros utilisateur sont supportées.
Mot de passe	Indiquez le mot de passe si vous avez configuré l'authentification sur votre application Java. Les macros utilisateur sont supportées.

Si vous souhaitez surveiller un compteur booléen qui est “vrai” ou “faux”, spécifiez le type d'information “Numérique (non signé)” et sélectionnez l'étape de prétraitement “Booléen vers décimal” dans l'onglet Prétraitement. Le serveur stockera les valeurs booléennes comme 1 ou 0, respectivement.

Clés d’élément JMX plus en détails

Attributs simples

Un nom d'objet MBean n'est rien d'autre qu'une chaîne que vous définissez dans votre application Java. En revanche, un nom d'attribut peut être plus complexe. Dans le cas où un attribut renvoie un type de données primitif (un entier, une chaîne, etc.), il n'y a rien à craindre, la clé ressemblera à ceci :

```
jmx[com.example:Type=Hello,weight]
```

Dans cet exemple, le nom d'objet est “com.example:Type=Hello”, le nom d'attribut est “weight” et probablement le type de valeur retourné devrait être “Numérique (flottant)”.

Attributs renvoyant des données composites

Cela devient plus compliqué lorsque votre attribut renvoie des données composites. Par exemple : votre nom d'attribut est “apple” et il renvoie un hachage représentant ses paramètres, comme “poids”, “couleur”, etc. Votre clé peut ressembler à ceci :

```
jmx[com.example:Type=Hello,apple.weight]
```

C'est ainsi qu'un nom d'attribut et une clé de hachage sont séparés, en utilisant le symbole point. De même, si un attribut renvoie des données composites imbriquées, les parties sont séparées par un point :

```
jmx[com.example:Type=Hello,fruits.apple.weight]
```

Problème avec les points

Jusqu'ici tout va bien. Mais que faire si un nom d'attribut ou une clé de hachage contient un point ? Voici un exemple :

```
jmx[com.example:Type=Hello,all.fruits.apple.weight]
```

C'est un problème. Comment dire à Zabbix que le nom de l'attribut est “all.fruits”, et pas seulement “all” ? Comment distinguer un point faisant partie du nom, du point qui sépare un nom d'attribut et des clés de hachage ?

Avant la version **2.0.4**, la passerelle Java de Zabbix était incapable de gérer de telles situations et les utilisateurs restaient avec des éléments NON SUPPORTÉ. Depuis 2.0.4 cela est possible, tout ce que vous devez faire est d'échapper les points qui font partie du nom avec un backslash :

```
jmx[com.example:Type=Hello,all\.fruits.apple.weight]
```

De la même façon, si votre clé de hachage contient un point, vous l'échapperez :

```
jmx[com.example:Type=Hello,all\.fruits.apple.total\.weight]
```

Autres problèmes

Un backslash dans un nom d'attribut doit être échappé :

```
jmx[com.example:type=Hello,c:\\documents]
```

Pour gérer d'autres caractères spéciaux dans la clé d'élément JMX, consultez la section sur le format des clés d'élément.

C'est en fait tout ce qu'il y a à faire. Bonne supervision JMX !

Types de données non-primitif

Depuis Zabbix 4.0.0, il est possible de travailler avec des MBeans personnalisables retournant des types de données non-primitif, qui remplace la méthode **toString()**.

Exemple de points de terminaison JMX personnalisés avec JBoss EAP 6.4

Les points de terminaison JMX personnalisés permettent de travailler avec différents protocoles de transport autres que le RMI par défaut.

Pour illustrer cette possibilité, essayons de configurer la supervision de JBoss EAP 6.4 comme exemple. Tout d'abord, faisons quelques hypothèses :

- Vous avez déjà installé la passerelle Java Zabbix. Sinon, vous pouvez le faire conformément à la documentation.
- Le serveur Zabbix et la passerelle Java sont installés avec le préfixe /usr/local/
- JBoss est déjà installé dans /opt/jboss-eap-6.4/ et fonctionne en mode autonome
- Nous supposerons que tous ces composants fonctionnent sur le même hôte
- Le pare-feu et SELinux sont désactivés (ou configurés en conséquence)

Faisons quelques réglages simples dans zabbix_server.conf :

```
JavaGateway=127.0.0.1
StartJavaPollers=5
```

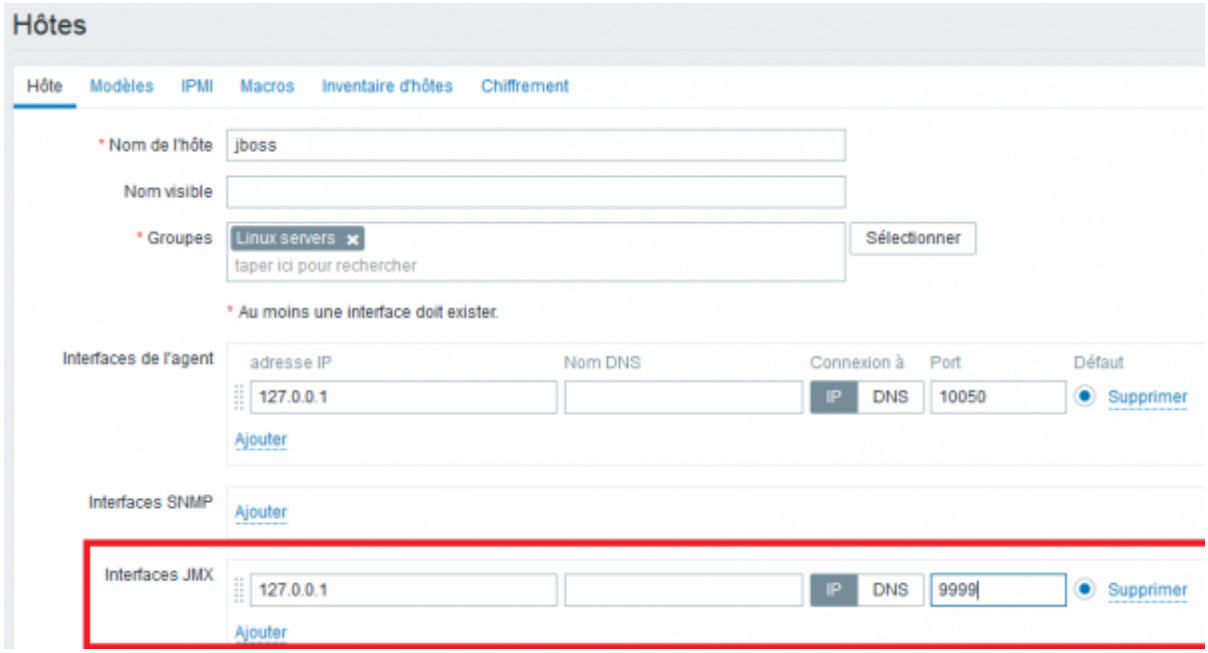
Et dans le fichier de configuration zabbix_java/settings.sh (ou zabbix_java_gateway.conf) :

```
START_POLLERS=5
```

Vérifiez que JBoss écoute sur son port par défaut :

```
$ netstat -natp | grep 9999
tcp        0      0 127.0.0.1:9999          0.0.0.0:*               LISTEN
10148/java
```

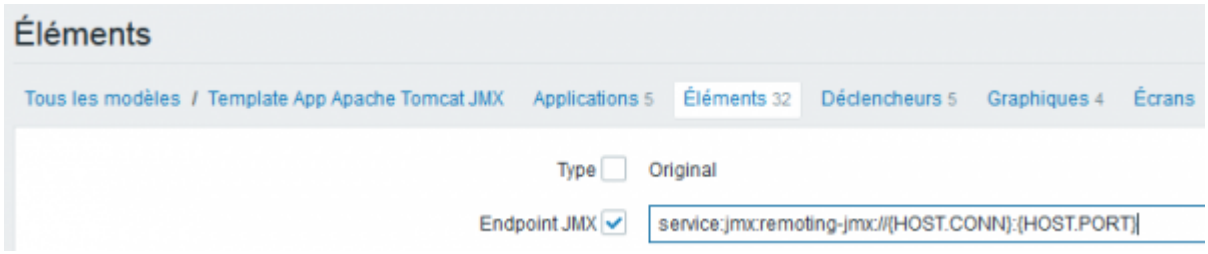
Maintenant, créons un hôte avec l'interface JMX 127.0.0.1:9999 dans Zabbix.



Comme nous savons que cette version de JBoss utilise le protocole JBoss Remoting au lieu de RMI, nous pouvons mettre à jour en masse le paramètre JMX endpoint dans notre modèle JMX en

conséquence :

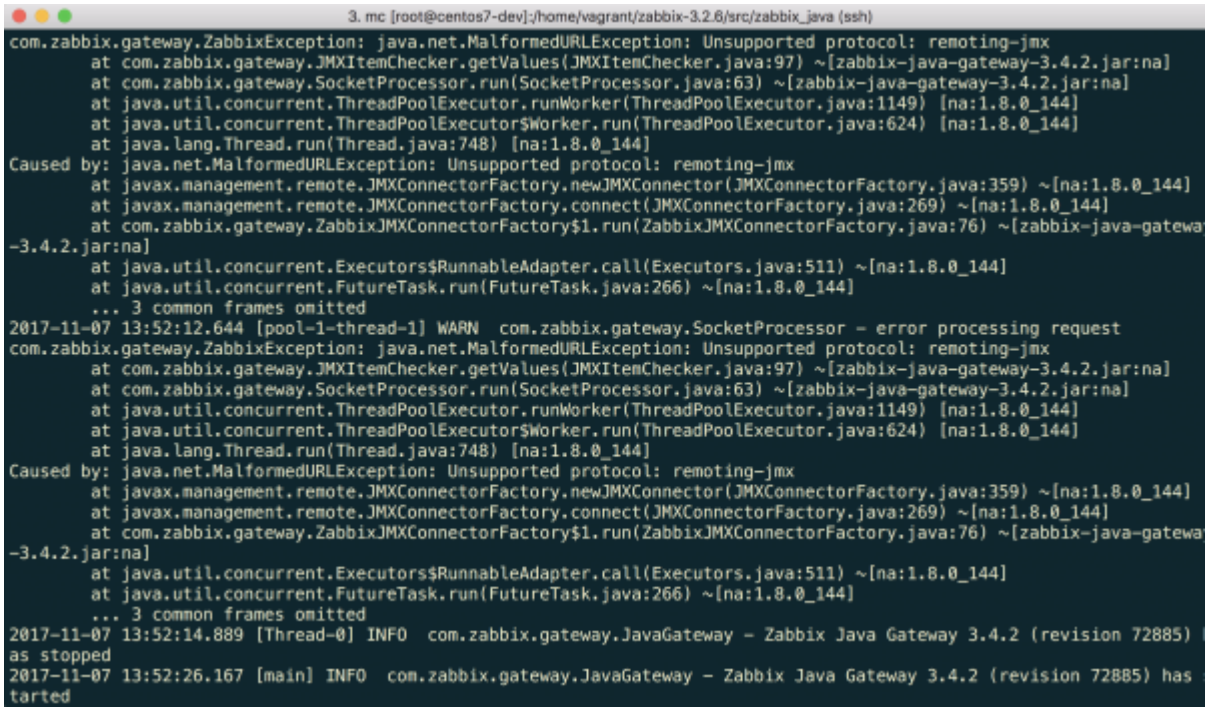
```
service:jmx:remoting-jmx://{HOST.CONN}:{HOST.PORT}
```



Mettons à jour le cache de configuration :

```
$ /usr/local/sbin/zabbix_server -R config_cache_reload
```

Notez que vous pouvez rencontrer une erreur la première fois.



“Unsupported protocol: remoting-jmx” signifie que la passerelle Java ne sait pas fonctionner avec ce protocole. Cela peut être corrigé en créant un fichier ~/needed_modules.txt avec le contenu suivant :

```
jboss-as-remoting
jboss-logging
jboss-logmanager
jboss-marshalling
jboss-remoting
jboss-sasl
jcl-over-slf4j
jul-to-slf4j-stub
log4j-jboss-logmanager
remoting-jmx
slf4j-api
xnio-api
```

```
xnio-nio</pre>
```

et ensuite exécutez la commande :

```
$ for i in $(cat ~/needed_modules.txt); do find /opt/jboss-eap-6.4 -iname
${i}*.jar -exec cp {} /usr/local/sbin/zabbix_java/lib/ \; ; done
```

Ainsi, la passerelle Java disposera de tous les modules nécessaires pour travailler avec JMX-Remoting. Il reste à redémarrer la passerelle Java, attendez un peu et si vous avez tout fait correctement, vous pouvez voir que les données de supervision JMX commencent à arriver dans Zabbix :