sysdig

Java仮想マシンから JMXメトリクスを 統合する

本文の内容は、Integrate JMX Metrics from Java Virtual Machinesのドキュメント(https://docs.sysdig.com/en/integrate-jmx-metrics-from-java-virtual-machines.html)を元に日本語に翻訳・再構成した内容となっております。

Java仮想マシンからJMXメトリクスを統合する

Sysdigエージェントは、JMXプロトコルを使用してJava仮想マシンからデータを取得します。エージェントは、アクティブなJava仮想マシンを自動的に検出し、ヒープメモリやガーベージコレクタのような基本的なJVMメトリクスや、ActiveMQ、Cassandra、Elasticsearch、HBase、Kafka、Tomcat、Zookeeperのようなアプリケーション固有のメトリクスをポーリングするように設定されています。

エージェントは、独自の Java プロセスからカスタム JMX メトリクスを抽出するように簡単に構成することもできます。抽出されたメトリクスは、事前に定義されたアプリケーション・ビューに表示されるか、Metrics > JVMおよびJMXメニューの下に表示されます。

注意事項
JVMとJMXの両方のメトリクスを収集するためには、Sysdigエージェントにjava.managementモジュールをロードする必要があります。

デフォルトのJMXメトリクス設定は、/opt/draios/etc/dragent.default.yamlファイルにあります。既存のエントリをカスタマイズする場合は、そのデフォルトのyamlファイルから完全なアプリケーションのビーンリストをユーザー設定ファイル/opt/draios/etc/dragent.yamlにコピーします。Sysdigエージェントは、両方のファイルの設定をマージします。

注意事項
Javaバージョン7～10は現在Sysdigエージェントでサポートされています。

Java 11-14の場合は、最低エージェントバージョン10.1.0を実行している必要があり、JMX Remoteオプションを使用してアプリを実行する必要があります。

以下は、Sparkアプリケーション用にカスタマイズされたエントリの dragent.yaml ファイルがどのようになっているかを示しています。

```
customerid: 07c948-your-key-here-006f3b
tags: local:nyc,service:db3
jmx:
  per_process_beans:
    spark:
```

```
    pattern: "spark"
    beans:
      - query: "metrics:name=Spark shell.BlockManager.disk.diskSpaceUsed_MB"
        attributes:
          - name: VALUE
            alias: spark.metric
```

application/beanの先頭に jmx: と per_process_beans: セクションヘッダを含めます。コンテナエージェントの設定ファイルにパラメータを追加する方法の詳細については、「エージェント設定ファイルの理解」を参照してください。

Bean設定

基本的なJVMメトリクスは、default_beans: セクション内で事前に定義されています。このセクションは、エージェントのデフォルト設定ファイルで定義されており、メモリやガベージコレクタの使用量など、すべてのJavaプロセスに対してポーリングされるビーンズや属性が含まれています。

```
jmx:
  default_beans:
    - query: "java.lang:type=Memory"
      attributes:
        - HeapMemoryUsage
        - NonHeapMemoryUsage
    - query: "java.lang:type=GarbageCollector,*"
      attributes:
        - name: "CollectionCount"
          type: "counter"
        - name: "CollectionTime"
          type: "counter"
```

各アプリケーションに固有のメトリクスは、アプリケーションにちなんだ名前のセクションで指定されています。例えば、これがTomcatのセクションです。

```
per_process_beans:
    tomcat:
      pattern: "catalina"
      beans:
        - query: "Catalina:type=Cache,*"
```

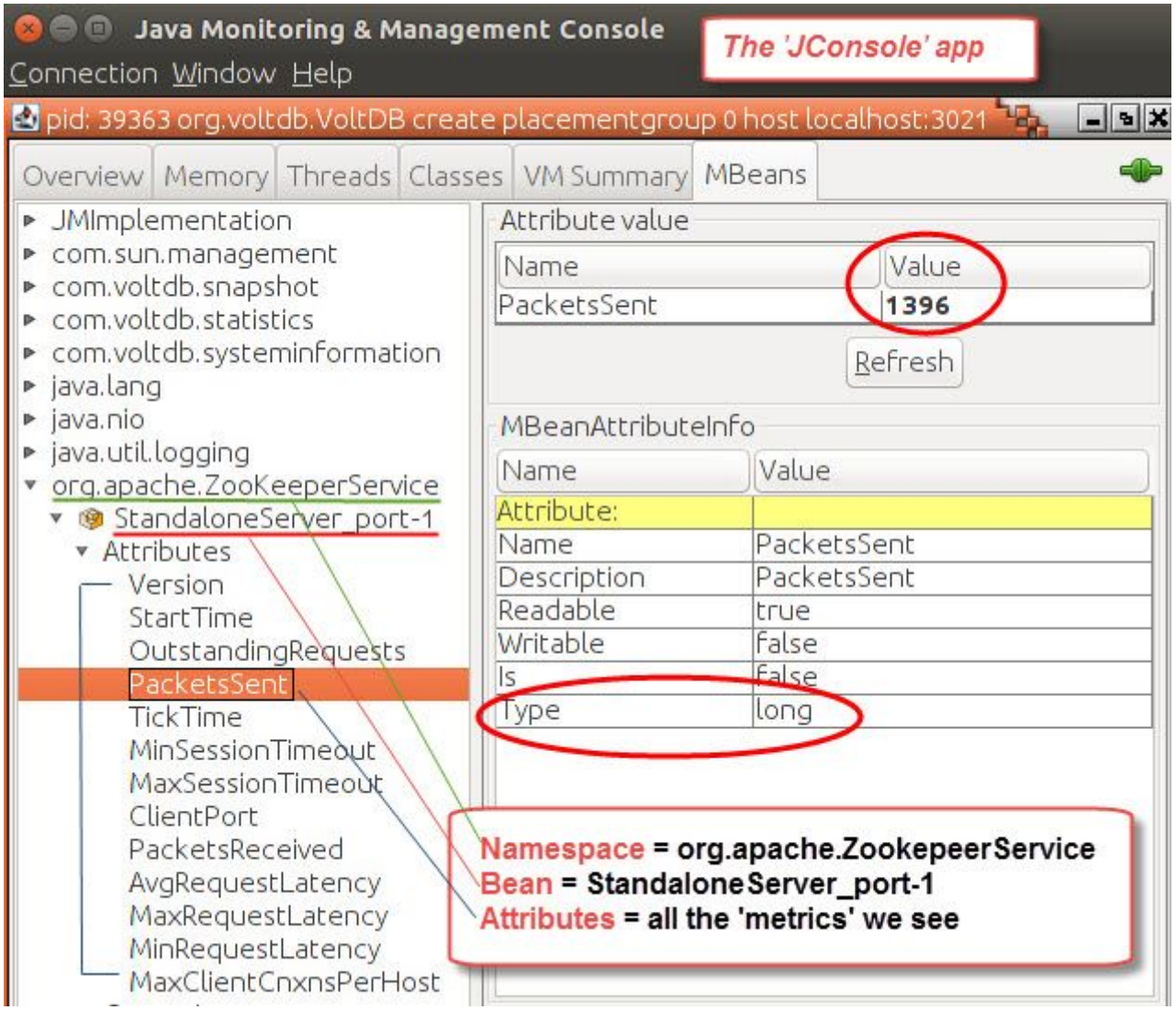
```
attributes:
  - accessCount
  - cacheSize
  - hitsCount
  - . . .
```

キー名（この場合は tomcat）は、Sysdig Monitor のユーザーインターフェイスでは、java だけではなく、プロセス名として表示されます。pattern: パラメータは、java プロセス名と引数をこの JMX メトリクスのセットと一致させるために使用される文字列を指定します。プロセス・メイン・クラスのフル・ネームに与えられたテキストが含まれている場合、プロセスはタグ付けされ、セクションで指定されたメトリクスがフェッチされます。

注意事項
クラス名は、プロセス引数リストと照合されます。コマンド・ラインでクラス名を公開しないカスタムの方法で JMX メトリクスを実装する場合は、Java 呼び出しコマンド・ラインに都合よくマッチするパターンを見つける必要があります。

beans: セクションには、JMX パターンに基づいて、問い合わせ対象のビーンのリストが含まれています。JMXパターンは、Oracleのドキュメントで詳細に説明されていますが、実際には、クエリ行の形式は非常に単純です。java.lang:type=Memoryのように、ビーンの完全な名前を指定することもできますし、以下のようにワイルドカード*を使用して、1つの行で複数のビーンを取得することもできます。

アプリケーションがエクスポートするすべてのビーンのリストを取得するには、JVisualVM、Jmxterm、JConsole、または他の類似のツールを使用することができます。以下は、ネームスペース、bean、属性（メトリクス）情報をどこで見つけるかを示すJConsoleのスクリーンショットです（JConsoleはJava開発キットをインストールしたときに利用できます）。



各クエリに対して、取得したい属性を指定することができます。以下のJMX属性をサポートしています（これらの属性については、すべての副属性が取得されます）。

- 数値属性
- [コンポジットデータサポート]

属性には、絶対値やレートを指定することができます。値の場合は、送信前に秒単位のレートを計算する必要があります。この場合、type: counter を指定することができます。デフォルトは rate で、省略することができます。

制限値

ホストごとにポーリングされる JMX メトリクスの総数は 500 に制限されています。プロセスごとに照会されるビーンの最大数は300に制限されています。より多くのメトリクスが必要な場合は、ユースケースを添えて営業担当者にお問い合わせください。

エージェント 0.46 以前のバージョンでは、プロセスごとに 100 ビーンが制限されていました。

エイリアス

JMXビーンと属性は非常に長い名前を持つことができます。インターフェースの乱雑さを避けるために、エイリアスのサポートを追加しました。例えば、以下のようになります。

```
    cassandra:
        pattern: "cassandra"
        beans:
          - query: "org.apache.cassandra.db:type=StorageProxy
            attributes:
              - name: RecentWriteLatencyMicros
                alias: cassandra.write.latency
              - name: RecentReadLatencyMicros
                alias: cassandra.read.latency
```

このようにして、エイリアスはraw bean名の代わりにSysdig Monitorで使用されます。エイリアスは、bean名からデータを取得して動的に使用することもできます - パターンビーンクエリを使用する場合に便利です。例えば、以下のようになります。

```
          - query: "java.lang:type=GarbageCollector,*"
            attributes:
              - name: CollectionCount
                type: counter
                alias: jvm.gc.NAME.count
              - name: CollectionTime
                type: counter
                alias: jvm.gc.NAME.time
```

このクエリは複数のbean(すべてのガベージコレクタ)にマッチし、メトリクス名はガベージコレクタの名前を反映します。例：jvm.gc.ConcurrentMarkSweep.count . 一般的な構文は次のとおりです。

{<bean_property_key>}} です。すべてのビーンズのプロパティを取得するには、JVisualVMやJmxtermのようなJMXエクスプローラを使用することができます。

トラブルシューティング：Java (JMX) メトリクスが表示されないのはなぜですか？

Sysdigエージェントは通常、ホスト上で実行されているJavaプロセスを自動検出し、それらをポーリングするためにJMX拡張機能を有効にします。

JMXリモート

Java アプリケーションがエージェントによって自動的に検出されない場合は、アプリケーションのコマンドラインに以下のパラメータを追加してみてください。

```
-Dcom.sun.management.jmxremote
```

詳細については、JMXテクノロジーを使用した監視に関するオラクルのWebページを参照してください。

Javaバージョン

留意事項

Javaバージョン7～10は現在Sysdigエージェントでサポートされています。

Java 11-14の場合は、最低エージェントバージョン10.1.0を実行している必要があり、JMX Remoteオプションを使用してアプリを実行する必要があります。

JavaベースのアプリケーションとJMX認証

Javaベースのアプリケーション(Cassandra、Elasticsearch、Kafka、Tomcat、Zookeeperなど)では、Sysdigエージェントがメトリクス(bean)をポーリングするためにJava実行環境(JRE)をインストールする必要があります。

警告

SysdigエージェントはJMX認証をサポートしていません。

Docker-コンテナベースのSysdigエージェントがインストールされている場合、JREはエージェントバイナリと一緒にインストールされ、それ以上の依存関係は存在しません。しかし、サービスベースのエージェント(非コンテナ)をインストールしていて、JVM/JMXメトリクスのレポートが表示されない場合、ホストにJREがインストールされていないか、期待した場所にインストールされていない可能性があります: usr/bin/java

SysdigエージェントがJREを見つけることができるかどうかを確認するには、Service dragent restartでエージェントを再起動し、エージェントの/opt/draios/logs/draios.logファイルで、エージェント起動時に記録された2つのJava検出ログとロケーションログのエントリを確認してください。

Javaが見つからない場合や見つからない場合の例

```
2017-09-08 23:19:27.944, Information, java detected: false
2017-09-08 23:19:27.944, Information, java_binary:
```

Javaが検出された場合の例。

```
2017-09-08 23:19:27.944, Information, java detected: true
2017-09-08 23:19:27.944, Information, java_binary: /usr/bin/java
```

Javaがインストールされていない場合、解決策はJavaランタイム環境をインストールすることです。ホストにJavaがインストールされているが、期待される場所(/usr/bin/java)にない場合は、/usr/bin/javaから実際のバイナリへのシンボリックリンクをインストールするか、Sysdigエージェントの設定ファイルにjava_home:変数を設定してください。/opt/draios/etc/dragent.yaml

```
java_home: /usr/my_java_location/
```

JMXポーリングを無効にする

必要ない場合や、JMXメトリクス報告を無効にしたい場合は、エージェントのユーザー設定ファイル/opt/draios/etc/dragent.yamlに以下の2行を追加します。

```
jmx:
  enabled: false
```

ファイルを編集した後、 dragent restart サービスを使ってネイティブの Linux エージェントを再起動するか、コンテナ化されたエージェントを再起動して変更を有効にしてください。

コンテナ化されたエージェントを使用している場合は、dragent.yamlファイルを編集する代わりに、エージェント起動時のdocker runコマンドに以下のパラメータを追加することができます。

```
-e ADDITIONAL_CONF="jmx:\n  enabled: false\n"
```