Data Analytics Project bei Datev

## Kurze Projektbeschreibung

Die Datev steht vor der Herausforderung, dass viele verschiedenen Metriken über verschiedene Wege in den Appdynamics Plattform gelangen. Der Controller speichert diese Metriken in seiner Datenbank und macht Sie über den Metrik Browser, die UI und per API verfügbar. In einigen Anwendungsfällen – wie bei Applikationsübergreifenden Dashboards – sind diese Daten leider nicht über eine dynamische Query abzufragen.

Um diese Daten dynamischer abzurufen, sollen bestimmte Metriken aus dem Controller innerhalb des Event-Service verfügbar gemacht werden, damit die Datev diese dann über Appdynamics Dynamic Query Language abfragen kann.

Diese Synchronisation soll über eine Python Applikation abgebildet werden, die die Metriken per REST-API zieht und über den Analytics Event Service API zum Event Service zurück überträgt.

## Implementierungsherausforderungen

Um die gewählten Metriken per ADQL abfragen zu können müssen diese innerhalb des Event-Service verfügbar sein. Um das zu erreichen müssen die Daten vom Controller abgefragt werden und dann zurück zum Event-Service übertragen werden.

Da jedoch beständig neue Daten entstehen, bestehen die folgenden Herausforderungen

1. Das Abrufen der Daten dauert länger als eine Minute – und somit können die bereits neu entstanden Daten nicht on Time abgerufen werden (Alternativen, wie jede Minute einen neuen Thread zu spawnen für die Abrufe, kann unter Umständen dazu führen, dass sich immer mehr Threads aufbauen und die Anwendung langfristig überlastet)
2. Der Event-Service steht nicht zur Verfügung und die Daten können nicht übermittelt werden
3. Es sind zu viele Daten zu prozessieren, so dass nicht schnell genug abgearbeitet wird (ähnlich wie (1))

Es existieren verschiedene Ansätze, um den genannten Problemen zu begegnen. Der zuerst genannten Punkt ist der kritischste. Es ist zu erwarten, dass bei sequenziellem Abrufen aller Einzel-Metriken die Gesamtlaufzeit allein auf Grund des HTTP Overheads deutlich länger als eine Minute dauern könnte. Um eine solche Verzögerung entgegen zu treten, bestehen die folgenden Lösungsansätze:

1. Abrufen von Zeitfenstern von länger als einer Minute (Beispiel: Alle 5 Minuten, für 5 Minuten Daten ziehen)
2. Abrufen jeder einzelnen Metrik in einem eigenen Thread
3. Abrufen von mehreren Metriken innerhalb eines eigenen Threads

Punkt (1) kann nicht in Betracht gezogen werden, da in diesem Fall keine aktuellen Daten im Event-Service vorliegen und somit auch nicht mehr „on Time“ benachrichtigt werden kann.

Beim zweiten (2) Ansatz besteht die Problematik, dass zu viele Threads erzeugt werden und somit zu viel Overhead durch den Thread entsteht. Das Ziehen einer einzelnen Metrik und anschließende übertragen zum Event-Service passiert deutlich schneller. Der Nachteil von (2) kann bei der Implementierung (3) umgangen werden.

Hier können verschiedenen Strategien angewendet werden, wie

1. Ein Thread pro Applikation
2. Ein Thread pro N Metriken
3. Ein Thread pro Metrik-Typ (Wie Calls per Minute)

Je nach gewählter Strategie können ebenso mehrere Metriken mit einem einzelnen Request abgeholt werden – mit der Hilfe von Wildcards.

## Umsetzung über die Analytics Event Service API

Um die Daten von den jeweiligen lokalen Skripten zur Appdynamics Plattform zu übermitteln, verwenden wir die Analytics Event Service API. Diese API wird in der Dokumentation umfassend beschrieben: <https://docs.appdynamics.com/display/PRO45/Analytics+Events+API>

Das Skript kann sich mit Hilfe eines API Keys der im Headerfeld **X-Events-API-Key** bezeichnet ist gegenüber dem Eventservice autorisieren. Das anlegen eines solchen Schlüssels ist hier beschrieben: <https://docs.appdynamics.com/display/PRO45/Managing+API+Keys>

Wie bereits im vorherigen Abschnitt erläutert, können wir unsere eigene Datenstruktur im Event-Service hinterlegen, dass bedeutet: Wir können die Anzahl der Felder definieren und welches Feld welchen Datentypen hat. Dabei werden die folgenden Datentypen unterstützt: String, Integer, Float, Boolean, Date (ISO 8601 format oder UNIX epoch date format).

Auf Basis der bisher existierenden Daten empfiehlt sich die folgende Struktur

1. Hostname (String)
2. Environment (String) – PROD/UAT/INT
3. DataCollectionDate (Date) – Wann wurde der Wert ermittelt
4. Metrictyp (String) – Infrastructur/Application/Log
5. Metricpath (String) – sollte evtl. Weitgehend aufgeteilt werden, damit man besser aggregieren kann
6. Metricvalue (noch nicht klar)

Ein wesentlicher Punkt dieser Struktur ist das 5 Feld – der Metricpath. Denn zum einfachen aggregieren, empfiehlt es sich hier keine Strukturen der Art „..|..|..“ zuzulassen, da dieser schwerer in den Abfragen adressiert werden können.

## Technische Implementierungsdetails

Um die Synchronisierung der APIs jede Minute durch zu führen, wird in Python das Paket **sched** verwendet.Dieses ist so konfiguriert, dass jede Minute die Methode **process\_all\_metrics** aufgerufen wird.

Falls dieser Aufruf länger als 1 Minute dauert, dann wird der zweite Aufruf direkt dann gestartet, wenn der erste fertig ist.