jakob-dittrich-dyn Add alerting setup documentation for product limits dashboard

5fb172e · 2 weeks ago

Name	Name	Last commit date
..		
README.md	Add alerting setup docum...	2 weeks ago

main limit-dashboard / limits-alerting / Top

Product Limits Alerting Setup with Davis Anomaly Detection

Overview

This guide provides a focused walkthrough on setting up product limits alerting in Dynatrace using Davis Anomaly Detection. By following the steps outlined here, you can configure alerts to notify you when product usage exceeds defined thresholds, ensuring proactive monitoring and management of your environment.

Step 1: Prepare and Ingest Data

Before setting up alerting, ensure that product usage data is ingested into Grail as business events. This is typically achieved using a workflow that collects product limits and sends them to Grail for analysis. The workflow should generate BizEvents on a regular basis, such as every 15 minutes, to ensure consistent data availability. Refer to the workflow setup instructions in the GitHub readme for detailed guidance on configuring this ingestion process.

Once ingested, these business events will serve as the foundation for creating metrics and configuring anomaly detection.

Step 2: Generate Metrics from Business Events

After data ingestion, create a metric from the business events using the `makeTimeseries` command in a Dynatrace Query Language (DQL) query. This metric will represent the percentage of product limit usage and will be used for anomaly detection.

Example Query:

```
fetch bizevents
| filter event.type == "dynatrace.limits"
| filter isNotNull(limit)
| fields timestamp, start, end, schemaId, current, limit, level =
(current * 1.0/limit) * 100
| sort isNotNull(current) desc, level desc
| filter current != "null"
| makeTimeseries max(level),
    by:{schemaId},
    interval: 1m
```

While this is a common example, you can customize the query based on your specific use case, such as focusing on particular fields or adjusting the aggregation interval.

Step 3: Configure Davis Anomaly Detection

With the metric in place, if you haven't already installed the **Davis Anomaly Detection** app within Dynatrace Apps, make sure to do so. Then set it up to monitor product usage and trigger alerts when thresholds are exceeded. Follow these steps:

1. Open Davis Anomaly Detection

- Navigate to **Davis Anomaly Detection** in the Dynatrace menu.
- Click **Create New Anomaly Detector** > **Advanced**.

2. Set Scope

- Define the scope by entering a DQL query that specifies the metric to monitor. Use the query from Step 2 or a variation tailored to your needs.

3. Define Alert Conditions

- **Analyzers**: Choose an appropriate detection method, such as **Static threshold anomaly detection**.
    - ***Static Thresholds***: Trigger alerts when a metric exceeds a fixed upper or lower limit.
    - ***Auto-Adaptive Thresholds***: Dynamically adjust thresholds based on recent trends in the

      data.
    - ***Seasonal Baselines***: Use historical patterns to detect deviations from expected behavior
- **Threshold**: Set a threshold value (e.g., `75`) to trigger an alert when product usage exceeds this percentage.
- **Alert Condition**: Specify whether the alert should trigger when the metric is above or below the threshold.
- **Alert on Missing Data**: Decide whether to enable this option. Typically, it is disabled to avoid false notifications when data points are missing.
- **Violating Samples**: Define the number of samples that must breach the threshold to trigger an alert. For immediate alerts, set this to `1`.
- **Sliding Window**: Define the evaluation period in minutes, with each sample representing 1 minute (e.g., `30 -> 30-minute` time window). If the BizEvents are generated by the workflow on a regular basis (e.g., every 15 minutes), setting a sliding window of `30` ensures the metric provides at least one data point during the evaluation period.
- **Dealerting Samples**: Set the number of samples required to resolve the alert when the metric returns to normal levels. A good practice is to set the **Dealerting Samples** to the same value as the **Sliding Window**, ensuring the notification automatically closes as soon as all data points in the sliding window fall below the threshold.

4. Add Details

Anomaly Detector Name and Description

- The name and description of the anomaly detector are primarily for organizational purposes. These help you and your team avoid confusion when managing multiple detectors.

- Name: Choose a clear and concise name, such as "Product Limit Usage Monitor", to easily identify the detector's purpose.
- Description: Add a brief explanation of what the detector does. For example: "This detector monitors product usage to identify when predefined limits are exceeded, enabling proactive resource management."

Event Title and Description

- The event title and description are critical because they are stored in the database and used in downstream processes, such as alerting and reporting.
  - **Event Title**: This should be a meaningful and descriptive label for the event. For example: "product_limit_alert" This title will be used to identify the event in the database and workflows. You will work with this title later to filter, analyze, or manage events.
  - **Event Description**: The description provides additional context about the event and can be used directly in alert notifications. For example:

    ```
    "An anomaly was detected on {dims:schemaId}. Within the sliding window, {violating_samples} violation samples were detected that were {alert_condition} the threshold of {threshold}."
    ```

    Including variables like {dims:schemaId}, {violating_samples}, and {threshold} makes the description dynamic and actionable.

Event Properties

- **Storage and Accessibility**:
  Event properties, including defined variables, are stored in the database and can be accessed later for deeper analysis, troubleshooting, or to enhance alerting workflows.

  In addition to predefined properties, you can define and add your own custom variables to include more detailed information in the event. For example:

  - `event.schemaId` : Add custom variables like `"event.schemaId" – "{dims:schemaId}"` to provide specific metadata or context for the event.
- **Use `CUSTOM_INFO` for `event.type`** :
  By default, the `event.type` is set to `CUSTOM_ALERT` , which is used for events that require triggering alerts and potentially creating problems in the Problems app. However, if the event is purely informational and does not need to trigger an alert, set the `event.type` to `CUSTOM_INFO` .

Using `CUSTOM_INFO` ensures that the event is categorized as informational and does not create unnecessary problem notifications, keeping your monitoring environment clean and focused on critical issues.

Step 4: Set Up Notifications

To ensure timely responses to alerts, configure a workflow to notify your team via email, Slack, or other communication channels.

1. Create a Workflow

- Navigate to **Workflows** in Dynatrace.
- Click **Create Workflow** to start a new workflow configuration.

2. Add a Trigger

- Select **Davis Event Trigger** as the workflow trigger.
- Configure the following fields:
  - **Event State**: Use the default value `active`.
  - **Davis Event Name**: Enter the event name defined in the anomaly detector (e.g., `product_limit_alert`).
  - **Affected Entities**: Include all entities by default.

3. Add a Notification Task

- Set Up a Notification Task:

  Add a task to notify your team via the desired communication channel, such as email, Slack, or Microsoft Teams. This ensures that the right people are promptly informed about anomalies and can take action.

- Use Variables from the Davis Event:
  - Leverage variables from the Davis event to make your notifications dynamic and informative. For example:
    - Event Description: Provides detailed context about the anomaly. Variable: {{ event()["event.description"] }}
    - Event Name: Identifies the source or type of the alert. Variable: {{ event()["event.name"] }}

These variables allow you to include real-time, event-specific information in your notification messages, making them more actionable and relevant.

Additional Configuration Details

Straightforward Setup:

- Configuring notification tasks is simple and intuitive. For example:
  - Slack Integration: To notify your team via Slack, you'll need to set up a bot with a name and token. Once the bot is configured, you can customize the notification by choosing:
    - The channel where the message will be sent.
    - The message content, which can include variables like event descriptions and names.
    - Optional reactions to the message, such as emojis, to further customize how the notification appears.