apigee

API Monitoring

Google Cloud

API Monitoring

Monitoring your API's health is key to maintaining a trusted, reliable, and robust API program, and to quickly identifying and resolving issues. When designing your API, consider how to monitor in a lightweight and maintainable fashion.

API monitoring must be contextual to SLAs imposed over every API. Any successful monitoring strategy must begin by understanding SLAs.

An appropriate balance of API traffic generated by monitoring tools and real users generated API traffic should be used. After all, many metrics and associated SLAs may be related to trends and patterns generated by real traffic rather than individual API execution performance.

Where to start?

Ask yourself the following questions when you start to think about API Monitoring:

What are the requirements for monitoring API health?

- Is a simple ping enough?
- Are there certain resources that are critical?
- How deep does the monitoring need to be?

Are there requirements for monitoring the target endpoint health through the API?

- Are you looking to monitor both proxy and target health? Differentiating between proxy health vs target health can be key when diagnosing issues in production.

Which environments are important to have monitoring in place?

- Production is obvious but it could be just as important to monitor alpha, beta and dev integration environments.

Requirements

The main objectives of a monitoring strategy are:

- Defining various request/response patterns that touch as many components as possible to test the health of the overall system.
- Defining an external system that can execute these requests reliably and consistently. Ideally, this system should have the capability to run requests from multiple different data centers around the world.

A general best practice consists of:

- Select various cheap-to-execute requests to monitor real API resources to assess the health of individual proxy components.
- Broad coverage is more valuable than deep coverage. Instead of targeting every proxy API, prioritize having one or more proxies by relevant business functions hitting every backend system consumed by your APIs.
- Minimize the exposure of customer data but hit the real APIs and backend systems.

Common patterns

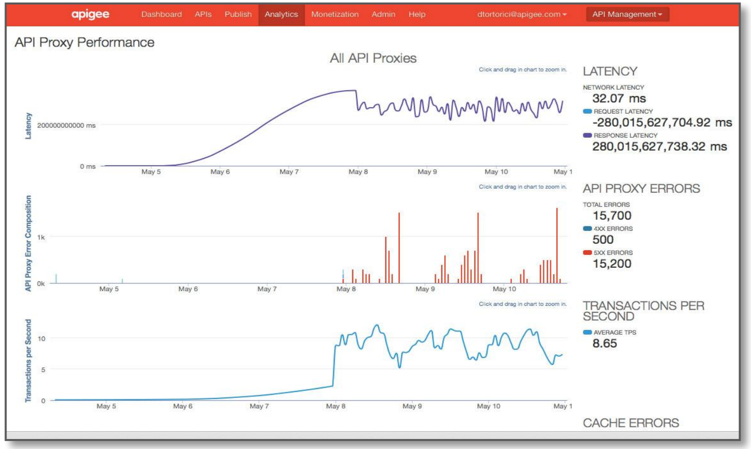
The following are some examples of resources we commonly use to fulfill the above requirements. The patterns described in this article are:

- **Ping sub resource** -- A specialised sub resource exposed by the proxy to test proxy network connectivity and proxy deployment status.
- **Status resource** -- A specialized resource to test proxy-to-target network connectivity and assess target API health.
- **Using real requests** -- Using the existing API resources to check the health of the system.

Analytics

If you are using real requests for monitoring, and if APIs are protected by API keys or OAuth, create a new separate application for monitoring. That way, requests can be identified in analytics.

Regardless of the monitoring approach you take, the requests will still appear in any analytics report so you may want to consider adding something in the requests to be able to easily filter them out of any reporting.



Tools

There are a number of tools out there to help you monitor your API.
The best tool is the tool that you already own.

- StackDriver
    - https://cloud.google.com/stackdriver/
    - https://cloud.google.com/monitoring/alerts/uptime-checks

apigee

Thank You

Google Cloud