Kong THE CLOUD CONNECTIVITY COMPANY

Kong Gateway Operations for Kubernetes

Kong Vitals

Course Agenda

1. Kong Gateway Installation on K8s
2. Intro to Kong Ingress Controller
3. Securing Kong Gateway
4. Securing Services on Kong
5. OIDC Plugin
6. **Kong Vitals**
7. Advanced Plugins Review
8. Troubleshooting
9. Test your Knowledge

Learning Objectives

1. What is Kong Vitals
2. Understand the difference between health and traffic metrics
3. Pulling metrics using Vitals API
4. Using InfluxDB or Prometheus with Vitals

Reset Environment

Before we start this lesson, be sure to reset your environment:

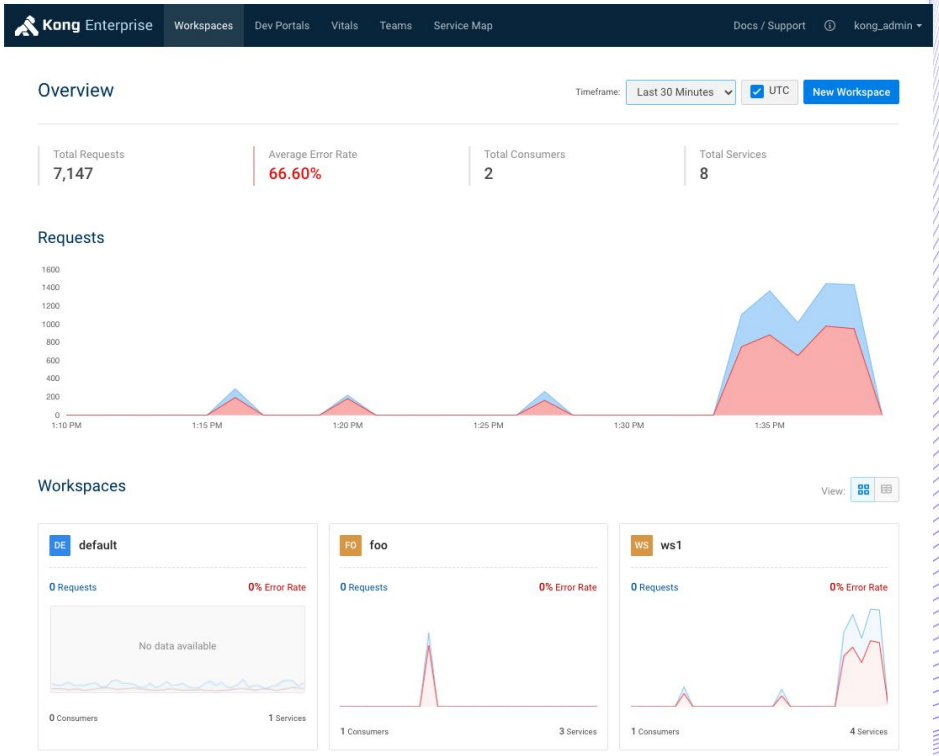
```
$ cd /home/labuser
$ git clone https://github.com/Kong/edu-kgac-202.git
$ source ~/edu-kgac-202/base/reset-lab.sh
```

Monitoring with Vitals

What is Kong Vitals?

Kong Vitals allows you to

- Monitor the health and performance of Kong Gateway
- Visualize the API transactions traversing Kong
- Access key statistics
- Monitor vital signs, and
- Pinpoint anomalies in real time
- View workspaces dashboards, charts, status codes in Kong Manager
- Access Vitals data via endpoints over Admin API



Accessing Vitals

Kong Vitals can be accessed via Kong Manager or Admin API.

Use Kong Manager to see visualizations of Vitals data, including the Workspaces Overview Dashboard, Workspace Charts, Vitals tab, and Status Codes, and to generate CSV Reports.

Any authorized user can view Vitals for the entire Kong cluster or for a particular Workspace

- If a user does not have access to a Workspace, its Vitals are hidden from view
- If a user does not have access to Vitals data, charts will not display

Use Kong Admin API to access Vitals data via endpoints allowing integration with other monitoring systems.

A sample Vitals API spec can be seen here https://git.io/JEAQK - these endpoints may be secured via RBAC.

Vitals Metrics

Vitals metrics fall into two categories:

- Health Metrics for monitoring the health & performance of a Kong cluster
- Traffic Metrics for insight into which of your services are being used, who is using them, and how they are responding

All metrics are collected at 1-second intervals and aggregated into 1-minute intervals

- 1-second intervals are retained for one hour
- 1-minute intervals are retained for 25 hours

If longer retention times are needed, the Vitals API can be used to pull metrics out of Kong and into a data retention tool, such as InfluxDB Time Series Database or Prometheus.

Health Metrics

Health metrics are tracked **per cluster** as a whole, and **per node**.

A node is a running process with an identifier unique to it, and not to the host on which the process runs, so restarting Kong results in a new node, hence new node ID.

Health metrics include

- Proxy Latency: min, max, and average time values in ms, that Kong proxy spends processing API proxy requests.
- Upstream Latency: min, max, and average time elapsed, in ms, between Kong sending requests upstream and Kong receiving the first bytes of responses from upstream.
- Datastore Cache Hit/Miss: count of requests to Kong's L1(worker)/L2(node-wide) node-level datastore cache. If neither cache contains the necessary information, Kong requests it from the datastore.
- Datastore Cache Hit Ratio: ratio of datastore cache hits to the total count of datastore cache requests.

* The Vitals API may return `null` for Latency metrics when no API requests were proxied during the timeframe. These are not graphed in Kong Manager and appear as gaps in Vitals charts.

Traffic Metrics

Traffic metrics include

- ➔ Request Counts
  - **Total Requests** - count of all API proxy requests received, including those rejected due to rate-limiting etc.
  - **Requests Per Consumer** - count of all API proxy requests received from each specific consumer
- ➔ Status Codes
  - **Total Status Code Classes** - count of all status codes grouped by status code class
  - **Total Status Codes per Service** - total count of each specific status code for a given service
  - **Total Status Codes per Route** - total count of each specific status code for a given route
  - **Total Status Codes per Consumer** - total count of each specific status code for a given consumer
  - **Total Status Codes per Consumer Per Route** - total count of each specific status code for a given consumer and route

Vitals API

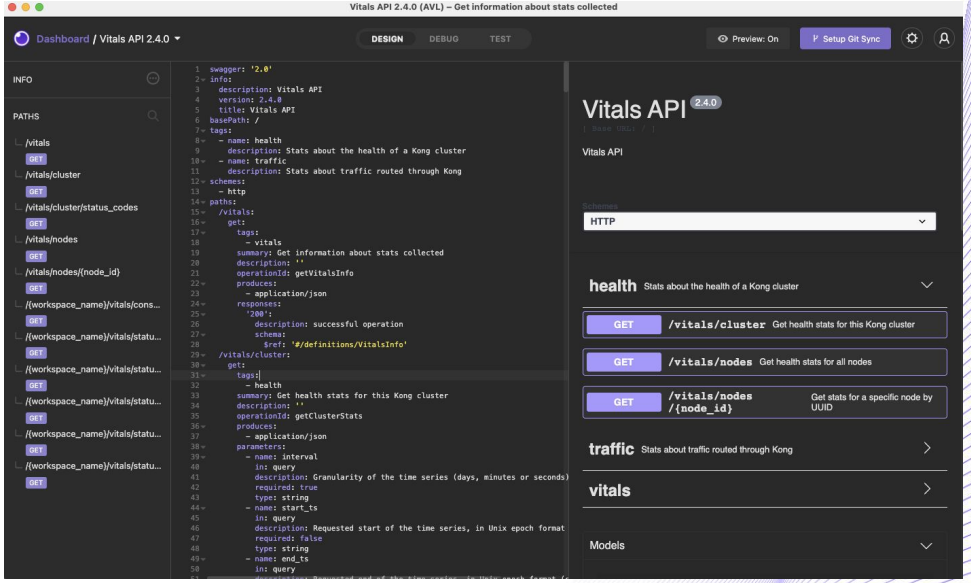
The Vitals API is documented at https://docs.konghq.com/gateway/2.8.x/vitals/vitalsSpec.yaml

Endpoints are accessible at a global level, e.g.

- `/vitals/cluster/status_codes`
- `/vitals/nodes`
- `/vitals/nodes/{node_id}`

Or at a workspace level, e.g.

- `/{workspace_name}/vitals/consumers/{consumer_id}/cluster`
- `/{workspace_name}/vitals/status_code_classes`



Enabling/Disabling Kong Vitals

Kong Gateway ships with Vitals enabled by default. To enable or disable Vitals:

- Modify the configuration file for Kong:
  - vitals = on *# vitals is enabled*
  - vitals = off *# vitals is disabled*

  Or

- Modify environment variables:
  - `$ export KONG_VITALS=on`
  - `$ export KONG_VITALS=off`

You will have to restart Kong for changed to become effective. We are using the default setting in this lesson, so Kong Vitals are enabled.

Task: Configure Service/Route/Consumer/Plugins

Let’s configure a Configure Service/Route/Consumer/Plugins which will use to demonstrate information provided by Kong Vitals when traffic is introduced.

```
$ cd /home/labuser/edu-kgac-202/exercises/vitals
$ kubectl apply -f ./httpbin-vitals.yaml
namespace/httpbin-demo created
kongplugin.configuration.konghq.com/httpbin-auth created
kongplugin.configuration.konghq.com/httpbin-rate-limiting created
service/httpbin-service created
ingress.networking.k8s.io/httpbin-ingress created
secret/jane-apikey created
kongconsumer.configuration.konghq.com/jane created
```

Task: Let's Create Some Traffic

Copy and paste the command, and leave it running while you perform the following steps. This will proxy a request every second for 64 seconds - you'll quickly see the rate limit being exceeded.

```
$ (for ((i=0;i<64;i++))
     do
       sleep .5
       http -h GET $KONG_PROXY_URI/httpbin?apikey=JanePassword
     done)
```

```
HTTP/1.1 429 Too Many Requests
...
RateLimit-Limit: 16
{
  "message":"API rate limit exceeded"
}
```

We will monitor these requests via the vitals' Admin API endpoints.

Task: Get Metrics from Vitals API

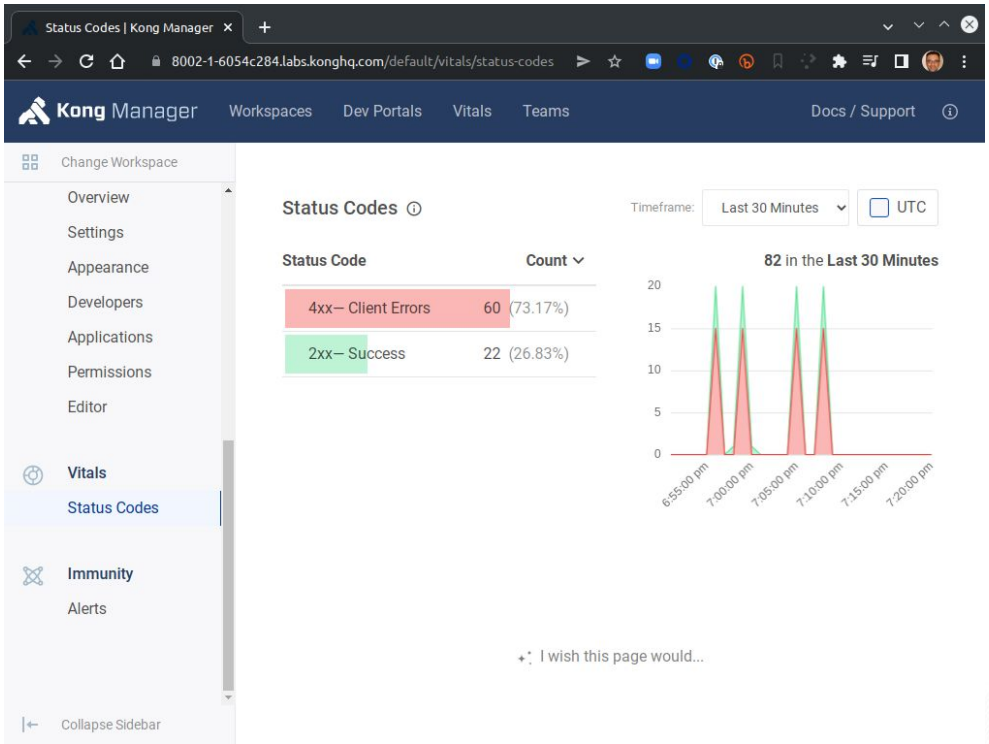
Let's get the Status Codes being generated via the Vitals Admin API

```
$ http kongcluster:30001/default/vitals/status_code_classes?interval=minutes \
    | jq .stats.cluster
```

```
{
  "1658507760": {
    "2xx": 1
  },
  "1658507820": {
    "2xx": 16,
    "4xx": 20
  },
  "1658507880": {
    "2xx": 16,
    "4xx": 10
  }
}
```

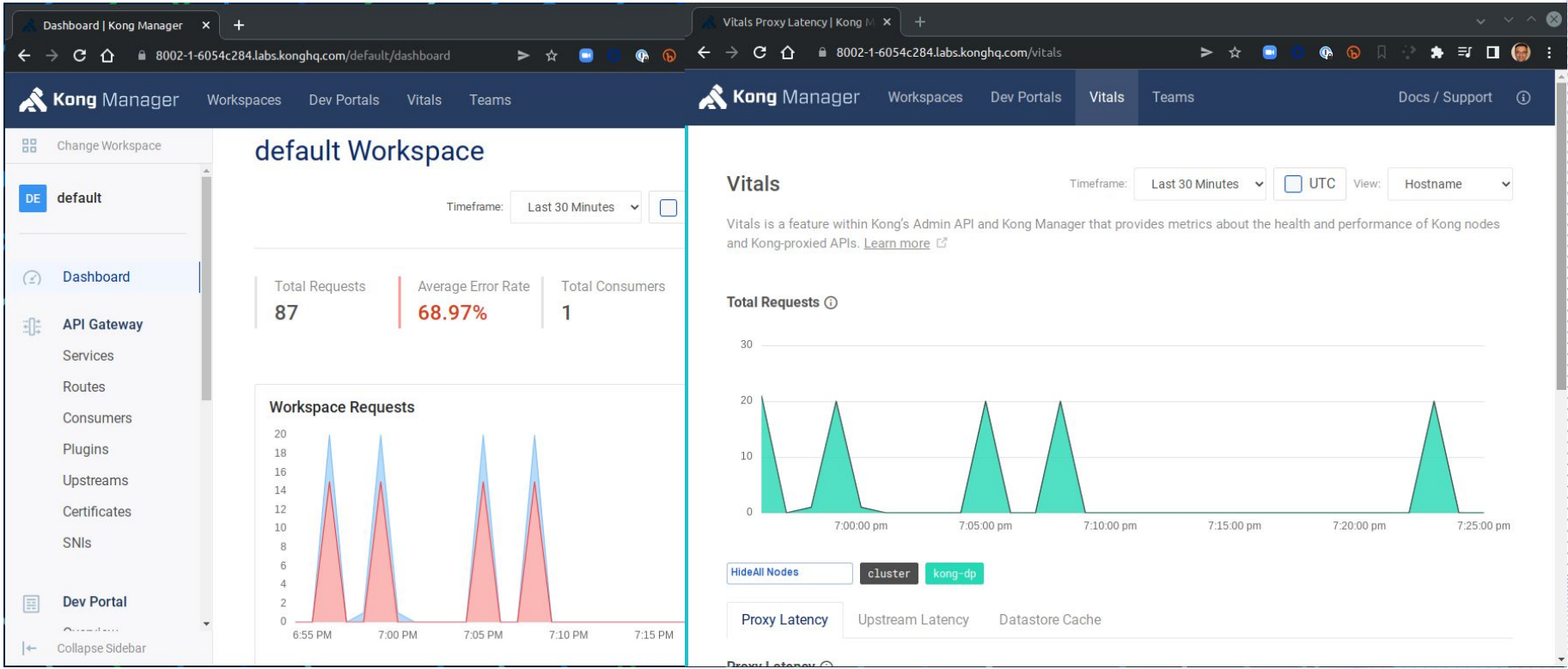
Task: Get Metrics from Kong Manager

Let's get the Status Codes being generated via the Kong Manager by selecting Default Workspace > Vitals > Status Codes:



Task: Get Metrics from Kong Manager

You can also see Vitals under Workspace overview page and under Vitals from main menu:



Exporting Metrics Data

Kong can be configured to store Vitals data in a time series database, such as InfluxDB or Prometheus, whereby it can be viewed in Grafana. Using such an external database for Vitals improves the performance of the Kong cluster, as it reduces the write load on the database backing the Kong cluster, which is quite important in very-high traffic Kong clusters.

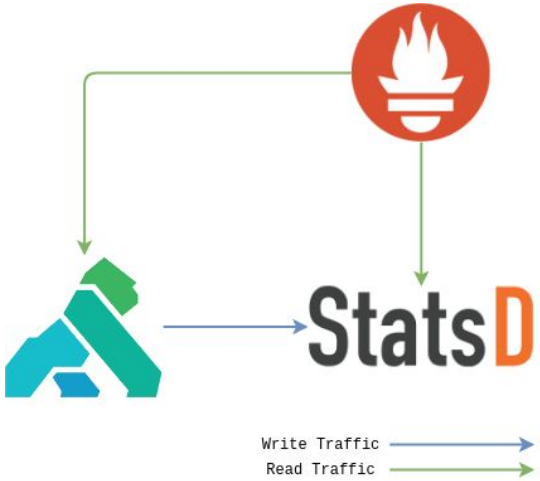
Using InfluxDB as datastore for Vitals gives you access to Vitals Reports in Kong Manager:

- Generate a report across all Workspaces or a single Workspace.
- Filter report parameters, such as object, metric, metric type, and values.
- Select the time period or interval for the report, such as minutes, hours, or days.
- View a line chart that automatically generates depending on the selected criteria.
- Export your report to a CSV file.

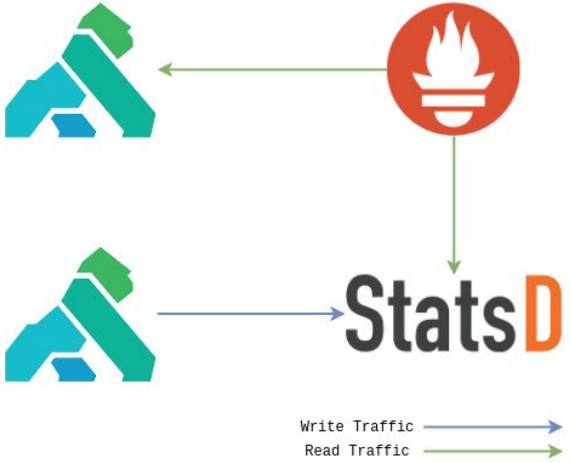
Important: The Vitals Reports feature is not compatible with a Postgres or Cassandra database. If using one of these databases, the Reports button will not display on the Vitals view.

Kong Vitals with Prometheus

Kong Vitals integrates with Prometheus via an intermediary data exporter, the Prometheus StatsD exporter. This allows Kong to efficiently ship Vitals metrics to StatsD process, where data points can be aggregated and offered to Prometheus, without affecting the performance of Kong. In this design, Kong writes Vitals metrics to the StatsD exporter as StatsD metrics, and Prometheus scrapes this exporter. Kong then queries Prometheus to retrieve and display Vitals data via the API and Kong Manager. A simple workflow looks like this:



In case of hybrid deployments, DPs write to StatsD and Prometheus reads from CP:



Task: Inspect Kong Vitals Configuration

We have a Prometheus deployment active on $PROMETHEUS_URL. We also have a StatsD deployment active with ports 9125/9102 for read/write traffic. Take a look at configurations for Kong Vitals and Prometheus/StatsD to see the setup:

```
$ cat ~/edu-kgac-202/base/cp-values.yaml | grep vitals | sort | uniq
```

```
vitals: on
vitals_statsd_address: statsd-prometheus-statsd-exporter.monitoring.svc.cluster.local:9125
vitals_strategy: prometheus
vitals_tsdb_address: prometheus-kube-prometheus-prometheus.monitoring.svc.cluster.local:9090
```

As you can see, Vitals strategy is set to prometheus, and addressed for writing to statsd and reading from prometheus are configured.

Task: Inspect Prometheus/StatsD Helm Values

```
$ cat ~/edu-kgac-202/exercises/monitoring/prometheus-values.yaml
```

```
  dashboards:
    kong:
      kong-official:
        gnetId: 7424
        revision: 7
        datasource: Prometheus
      kong-vitals:
        gnetId: 11870
        revision: 5
        datasource: Prometheus
```

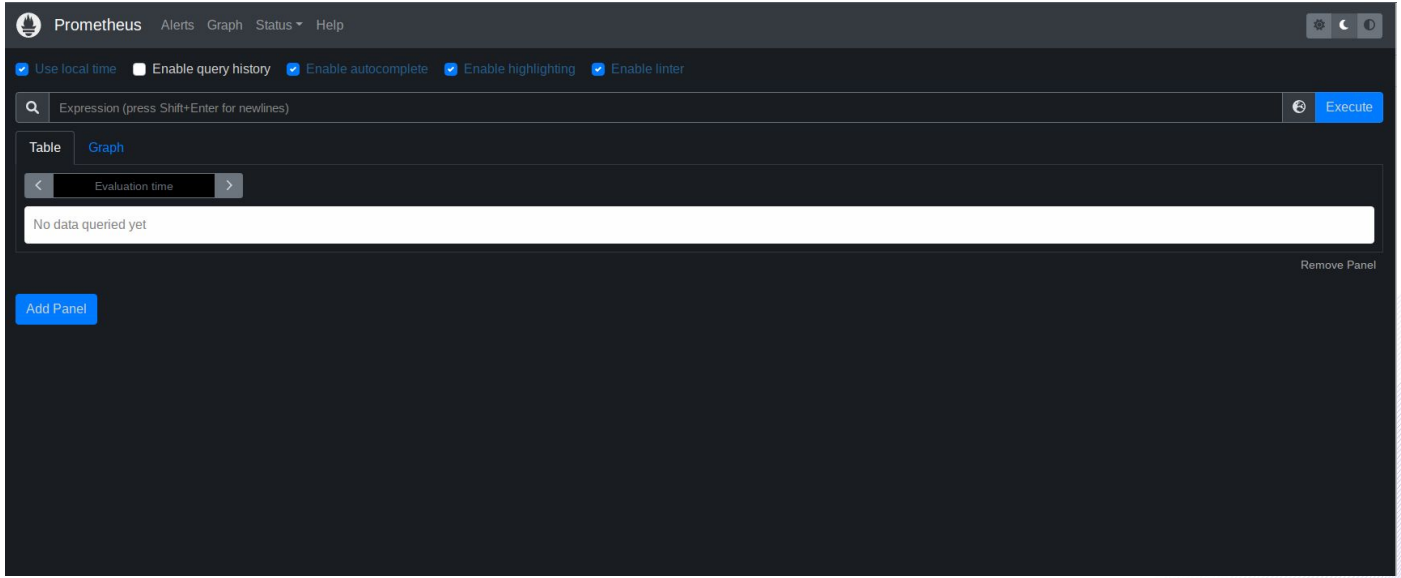
```
$ cat ~/edu-kgac-202/exercises/monitoring/statsd-values.yaml
```

Task: Get to GUI for Prometheus

Get the URI for accessing Prometheus from the environment variable and open it in a browser:

```
$ echo $PROMETHEUS_URL
```

```
https://30006-1-dfe23186.labs.konghq.com
```



Task: Let's Create Some Traffic

Copy and paste the command, and leave it running while you perform the following steps. This will proxy a request every second for 256 seconds - you'll quickly see the rate limit being exceeded.

```
$ (for ((i=0;i<256;i++))
     do
       sleep 1
       http -h GET $KONG_PROXY_URI/httpbin?apikey=JanePassword
     done)
```

```
HTTP/1.1 429 Too Many Requests
...
RateLimit-Limit: 16
{
  "message":"API rate limit exceeded"
}
```

We will monitor these requests via the Prometheus and Grafana.

Task: Get Metrics from Prometheus

1. Go to the browser tab showing the GUI for Prometheus
2. Query a data item such as kong_latency_proxy_request_sum
3. Visualize it over a period of time such as 15 minutes as a graph

Task: Get Metrics from Grafana

1. Switch to the browser tab for Grafana ($GRAFANA_URL) - Login with **admin** and **prom-operator**
2. Change the time range on the Kong Vitals Dashboard to Last 15 minutes
3. Observe Available graphs
4. Select Request By Status > View
5. Observe the graphs for response HTTP status codes

Here you will see a cyclic variation for status codes in the graph, which is correlated with the steps like graph observed in Prometheus in the previous slide.

That is due to Rate Limiting plugin blocking the request once the limit has reached and letting it through once the limit window is reset.

Summary

In this lesson we

- Learnt what Kong Vitals is and how it is used
- Learnt the difference between health and traffic metrics
- Pulled metrics using Vitals API
- Looked at using Prometheus/Grafana with Vitals

Questions?

What's next?

In the next section we will look a little closer at some advanced plugins.

Thank You