

Master's thesis presentation

Domain-Oriented Microservices Gateway Monitoring Using Big Data Techniques

Cristian M. Abrante Dorta

24th, February 2022

Aalto University & Unity Technologies

Table of contents

1. Introduction
2. Problem definition
3. Technical solution
4. Evaluation
5. Conclusion and future work

Introduction

Microservices architecture

This architecture pattern has the following characteristics (Lewis et al. [3]):

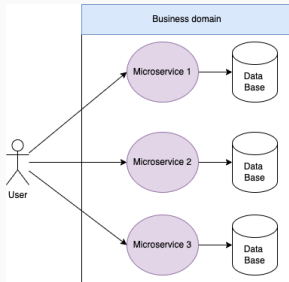
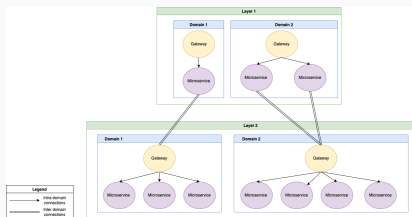


Figure 1: Diagram representing microservices architecture

- **Modularity of services**
- **Organized around business capabilities**
- **They are product not projects**
- **Decentralized data governance**

Domain-Oriented Microservices Architecture

Introduced by Gluck A. [2], it is composed by the following elements:



- Domain
- Layers
- Gateway
- Extension architecture

Figure 2: Diagram representing DOMA

This architecture make sense in **big organizations** with hundreds of interconnected microservices.

Company background

This project it is done in the context of the Acquire Unit of **UnityAds** product.

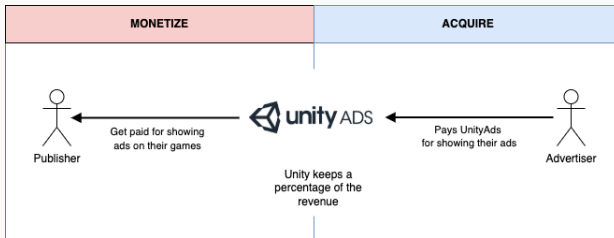
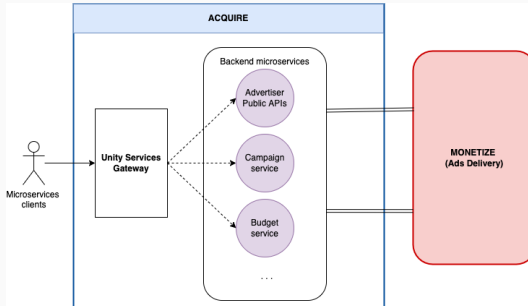


Figure 3: Diagram representing the UnityAds advertisement network

Acquire Unit architecture

The Acquire Unit is a **DOMA** from the technical point of view.



One of the main elements of DOMA is the gateway, which serves as the entry point for the domains. The gateway has to fulfil these high-level responsibilities (M. Thangavelu et al. [1]):

- Auditing pipeline
- Identity
- Rate limiting
- Documentation
- Response field trimming
- Datacenter affinity
- Short-term user bans

On the company context, the gateway is the **Unity Services Gateway**.

Problem definition

Problem definition

The gateway present some drawbacks on the visibility of the errors emitted by those high level responsibilities:

Advertiser public APIs Team

They should be aware of the errors that the advertisers are producing, and that are handled on the gateway side, e.g. rate limiting.

Problem definition

The gateway present some drawbacks on the visibility of the errors emitted by those high level responsibilities:

Advertiser public APIs Team

They should be aware of the errors that the advertisers are producing, and that are handled on the gateway side, e.g. rate limiting.

Product managers and client partners

They are responsible for the communication with the client so they should be aware on when the requests passing through the gateway are failing.

Problem definition

The gateway present some drawbacks on the visibility of the errors emitted by those high level responsibilities:

Advertiser public APIs Team

They should be aware of the errors that the advertisers are producing, and that are handled on the gateway side, e.g. rate limiting.

Product managers and client partners

They are responsible for the communication with the client so they should be aware on when the requests passing through the gateway are failing.

Security team

If any user is doing malicious usage of the APIs the security team has to be aware of it.

This lead to the three research questions addressed on this thesis:

Q1 *What is the best way to represent the possible high-level errors that a DOMA gateway can produce?*

This lead to the three research questions addressed on this thesis:

- Q1** *What is the best way to represent the possible high-level errors that a DOMA gateway can produce?*
- Q2** *What is the optimal technology for storing the errors that a DOMA gateway can produce?*

This lead to the three research questions addressed on this thesis:

- Q1** *What is the best way to represent the possible high-level errors that a DOMA gateway can produce?*
- Q2** *What is the optimal technology for storing the errors that a DOMA gateway can produce?*
- Q3** *How to correctly visualize and communicate with the stakeholders the errors a DOMA gateway can produce?*

Technical solution

The way for doing the configuration is to compose a series of filters:

- **Auditing filter**
- **Identity filter**
- **Rate limit filter**
- **Documentation filter**
- **Response fields trimming**

Gateway configuration file

```
1 - path: path/to/my/resource
2   envs: ['prd', 'stg', ... ]
3   methods: ['get', 'post', 'put' ... ]
4   endpointFilters:
5     ...
6   transport: 'http'
```

Figure 4: General structure

```
1 endpointFilters:
2   - authentication: service-account | OAuth
3   - authorization:
4     create: permission-identifier
5     read: permission-identifier
6     update: permission-identifier
7     delete: permission-identifier
```

Figure 5: Identity filter

```
1 endpointFilters:
2   - httpMethods: [get, put, patch, delete] // http
3     methods to limit
4     - perSecond: {prd: 6, stg: 100, local: 10000}
5     - perThirtyMinutes: 4000
```

Figure 6: Rate-limit filter

Error events format

This format will define the errors that the gateway will emit in relation with its responsibilities.

Common event fields
<code>source</code>
<code>requestId</code>
<code>path</code>
<code>organizationId</code>
<code>httpMethod</code>
<code>apiVersion</code>
<code>apiType</code>
<code>apiNamespace</code>
<code>timestamp</code>

Error events format

This format will define the errors that the gateway will emit in relation with its responsibilities.

Common event fields
source
requestId
path
organizationId
httpMethod
apiVersion
apiType
apiNamespace
timestamp

Rate limit specific field
rateLimitReason
quota
group

Error events format

This format will define the errors that the gateway will emit in relation with its responsibilities.

Common event fields
source
requestId
path
organizationId
httpMethod
apiVersion
apiType
apiNamespace
timestamp

Rate limit specific field
rateLimitReason
quota
group

Other possible responsibility specific fields ...

Data storage selection

The data storage selection has to be done according to **three criteria**:



	C1 - Price	C2 - Data Access	C3 - Integration
BigQuery	\$8.4/month	Easy	High
Amplitude	Free	Medium	Medium
PostgreSQL	\$109.9/month	Medium	High

Table 1: Data storage technology

Given those three criteria **BigQuery** is the selected technology.

BigQuery setup

For doing the setup a new table is created using **Terraform** configuration, the result is the following on the UI:

SCHEMA					DETAILS	PREVIEW
Table schema						
 Filter Enter property name or value						
Field name	Type	Mode	Policy Tags		Description	
source	STRING	REQUIRED				
type	STRING	REQUIRED				
requestId	STRING	REQUIRED				
path	STRING	REQUIRED				
httpMethod	STRING	REQUIRED				
apiVersion	STRING	NULLABLE				
apiType	STRING	REQUIRED				
apiNamespace	STRING	REQUIRED				
timestamp	TIMESTAMP	REQUIRED				
rateLimitReason	STRING	NULLABLE				
quota	INTEGER	NULLABLE				
group	STRING	NULLABLE				
resourceId	STRING	NULLABLE				
organizationId	INTEGER	NULLABLE				
url	STRING	REQUIRED				
customPath	STRING	REQUIRED				

Events gathering

The events gathering has to be placed on the events handlers of the **Unity Services Gateway**.

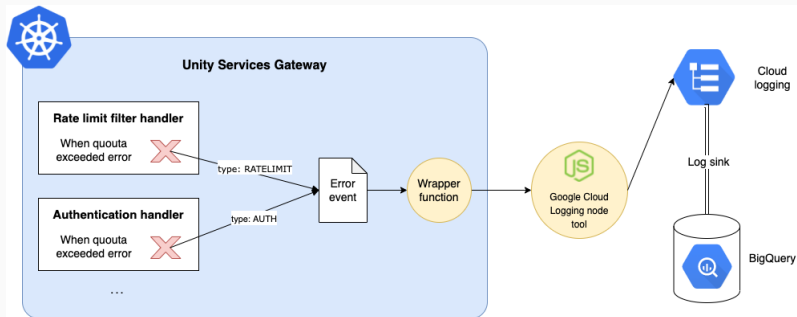


Figure 7: Diagram showing the events workflow

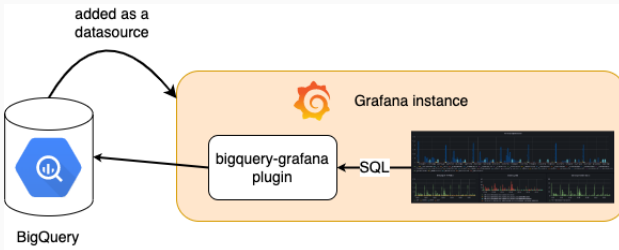
Log Sinks

Log sink allows the **persistence** on BigQuery of the logs emitted on the gateway side. It is done by applying a Terraform configuration.

```
1 resource "google_project_sink" "log_sink" {
2   name                = var.sink_name
3   destination         = "bigquery.googleapis.com/..."
4   filter               = var.filter
5   unique_writer_identity = true
6
7   bigquery_options {
8     use_partitioned_tables = true
9   }
10 }
11
```

Error visualization dashboard

The dashboard constructed using Grafana uses the `bigquery-grafana-plugin` to access the BigQuery datasource.



DEMO

Evaluation

Evaluation of the event gathering

For the evaluation of the event gathering **Jest** is used with mock servers and mock data.

```
Apple | ~/dev-environment/unity-services-gateway | master !1 | npmR test  
  
> unity-services-gateway@0.0.1 test  
> jest --testRegex='metrics\\.test\\.js$'  
  
PASS src/filters/endpoint/metrics.test.js  
  metrics  
    ✓ should call logger one time (4 ms)  
    ✓ should call logger with correct data (2 ms)
```

E2E evaluation

The method for the end-to-end evaluation consisted on doing **6.000** fake API calls to the services that are dependant on the gateway, on the staging environment.

	production	staging	local
Rate Limit	30	1000	3000

Table 2: Rate limit configuration

Expected result (on staging)

5.000 rows should be stored on BigQuery, as the rate limit should be triggered for each call after the first 1.000.

E2E evaluation

After that execution the query to the data source return the following value:

```
1 SELECT count(*) FROM `unity-ads-gke-stg.gateway_events.metrics`  
2 WHERE DATE(timestamp)  
3 BETWEEN "2021-01-01" AND "2022-02-24"
```

Figure 8: SQL query executed

Row	f0_
1	5000

Figure 9: Number of rows retrieved

Row	source	type	requestId	path
1	RATELIMITER	QUOTA_EXCEEDED	4389c1ac-0357-4f16-8765-425261b00035	/advertise/v1/organizations/organizationId/apps
2	RATELIMITER	QUOTA_EXCEEDED	4389c1ac-0357-4f16-8765-425261b00049	/advertise/v1/organizations/organizationId/apps
3	RATELIMITER	QUOTA_EXCEEDED	4389c1ac-0357-4f16-8765-425261b00090	/advertise/v1/organizations/organizationId/apps
4	RATELIMITER	QUOTA_EXCEEDED	4389c1ac-0357-4f16-8765-425261b00265	/advertise/v1/organizations/organizationId/apps
5	RATELIMITER	QUOTA_EXCEEDED	4389c1ac-0357-4f16-8765-425261b00280	/advertise/v1/organizations/organizationId/apps
6	RATELIMITER	QUOTA_EXCEEDED	4389c1ac-0357-4f16-8765-425261b00328	/advertise/v1/organizations/organizationId/apps
7	RATELIMITER	QUOTA_EXCEEDED	4389c1ac-0357-4f16-8765-425261b00342	/advertise/v1/organizations/organizationId/apps
8	RATELIMITER	QUOTA_EXCEEDED	4389c1ac-0357-4f16-8765-425261b00375	/advertise/v1/organizations/organizationId/apps
9	RATELIMITER	QUOTA_EXCEEDED	4389c1ac-0357-4f16-8765-425261b00395	/advertise/v1/organizations/organizationId/apps
10	RATELIMITER	QUOTA_EXCEEDED	4389c1ac-0357-4f16-8765-425261b00445	/advertise/v1/organizations/organizationId/apps

Figure 10: Subset of rows retrieved

Framework comparison

The solution is compared with other open source gateway According to three criteria

- C1 High-level responsibilities matching to individual routes
- C2 Error format definition
- C3 Data storage selection
- C4 Visualization of high-level errors

Framework comparison

The solution is compared with other open source gateway According to three criteria

C1 High-level responsibilities matching to individual routes

C2 Error format definition

C3 Data storage selection

C4 Visualization of high-level errors

	C1	C2	C3	C4
Kong	Yes - with UI	No	Yes	No
Zuul	Yes - with code	Yes	No	No
Proposed solution	Yes - with file	Yes	Yes - BigQuery	Yes

Table 3: Framework comparison table

Conclusion and future work

Conclusions

The solution has been satisfactory for all the stakeholders and has been used for identifying some problems with their integration.

The solution has been satisfactory for all the stakeholders and has been used for identifying some problems with their integration.

Depending on the size of the organization the possibilities for implementing this solution for gateway monitoring.

The solution has been satisfactory for all the stakeholders and has been used for identifying some problems with their integration.

Depending on the size of the organization the possibilities for implementing this solution for gateway monitoring.

- **Startups:** on those organizations there is usually no need to have a DOMA, so the monitoring is not needed.

The solution has been satisfactory for all the stakeholders and has been used for identifying some problems with their integration.

Depending on the size of the organization the possibilities for implementing this solution for gateway monitoring.

- **Startups:** on those organizations there is usually no need to have a DOMA, so the monitoring is not needed.
- **Mid-size organizations:** On those, the gateway monitoring might be needed, but maybe with simpler options like Amplitude.

The solution has been satisfactory for all the stakeholders and has been used for identifying some problems with their integration.

Depending on the size of the organization the possibilities for implementing this solution for gateway monitoring.

- **Startups:** on those organizations there is usually no need to have a DOMA, so the monitoring is not needed.
- **Mid-size organizations:** On those, the gateway monitoring might be needed, but maybe with simpler options like Amplitude.
- **Multinational companies:** On those this type of solutions might be suitable due to the high number of teams and microservices.

There are three improvement ideas that can be applied to this project:

- Extend error object to support more types of errors.

There are three improvement ideas that can be applied to this project:

- Extend error object to support more types of errors.
- Create a composable gateway logic.

There are three improvement ideas that can be applied to this project:

- Extend error object to support more types of errors.
- Create a composable gateway logic.
- Create a plugin that will pack the solution for being used in one of the standard gateways.



M. T. et al.

The Architecture of Uber's API gateway.

<https://eng.uber.com/architecture-api-gateway/>.

Accessed: 2022-01-01.



A. Gluck.

Introducing Domain-Oriented Microservice Architecture.

<https://eng.uber.com/microservice-architecture/>.

Accessed: 2021-12-31.



M. F. J. Lewis.

Microservices a definition of this new architectural term.

<https://martinfowler.com/articles/microservices.html>.

Accessed: 2021-12-31.

Thank you very much for your attention!
Any questions?