Showoff's Mulesoft API Related Best Practices

Contributors

Sonali Shah Integration Architect

Showoff

Introduction

Sonali Shah Integration Architect

This paper helps to identify the basics of the API and its benefits as well as Showoff's journey with Mulesoft and the best practices which Showoff thinks organisations can use to successfully solve the problem of integrations.

About the Author

Sonali has more than 6 years in the IT and Integration field, with various roles in Software Engineering and Mulesoft Development. This includes 4 years of Mule design & development experience in both Mule 3 and Mule 4. She achieved her Masters in Data Analytics from Dublin Business School in 2019 and has since gone on to complete the Mulesoft Certified Developer Level 1 and MuleSoft Certified Platform Architect - Level 1.
As a Mulesoft Developer at Showoff, Sonali works with various stakeholders and solution consultants to create state-of-the-art solutions in the areas of API-Led, micro-services , Governance and Security, the Anypoint Platform, to solve real-world situations and more complex projects. She aims to be a technology architect and give significant contribution to the Mulesoft community.

Contents

API and it's Benefits

Sonali Shah Integration Architect

An API is an Application Programming Interface. It allows developers and clients to interact with functionality that has been designed to fulfil the requirements of our customers' products. API provides an abstract way of interchange between applications. Application development becomes easy as the complex integration logic is supported by the API thus helping the web and mobile application developers to concentrate on the user experience without having to think about the functionality complexities. Showoff follows standard Restful API standards and hence always developed modern, reusable API's which reduced the development cost. Using APIs also removes the overhead of having to deploy the web and mobile applications as main functionality details are in the API, hence acting as a single point of failure.

Showoff and its journey with Mulesoft API

Sonali Shah Integration Architect

Mulesoft was acquired by Salesforce in 2019 and since Showoff have been Salesforce Partners since 2017, we became aware of Mulesoft and it's strategies to solve complex integrations problems. Using Mulesoft, Showoff felt that it could help its customers solve their integration problems of interactions with the legacy systems and started diving deep into Mulesoft's API approaches and its strategies. Mulesoft considers API as a basic building block of the application i.e it considers that each API is a small functionality.

Showoffs thinking about API being the backbone of the application synchronizes with Mulesoft's API led connectivity approach. Having the functionality development done in the API not only reduces the development overhead from the App developers but also makes the deployment lifecycle easier. With the growing integration demand to migrate the integration solutions and solve the complex integration problem, clients are looking for options which help them get the requirements fulfilled in a shorter span which seems impossible using the traditional IT approach. Using Mulesoft's idea of creating modern,microservice APIs can make the development and deliveries easier and quicker.

The diagram below represents the demands on IT versus its time to deliver the demands.

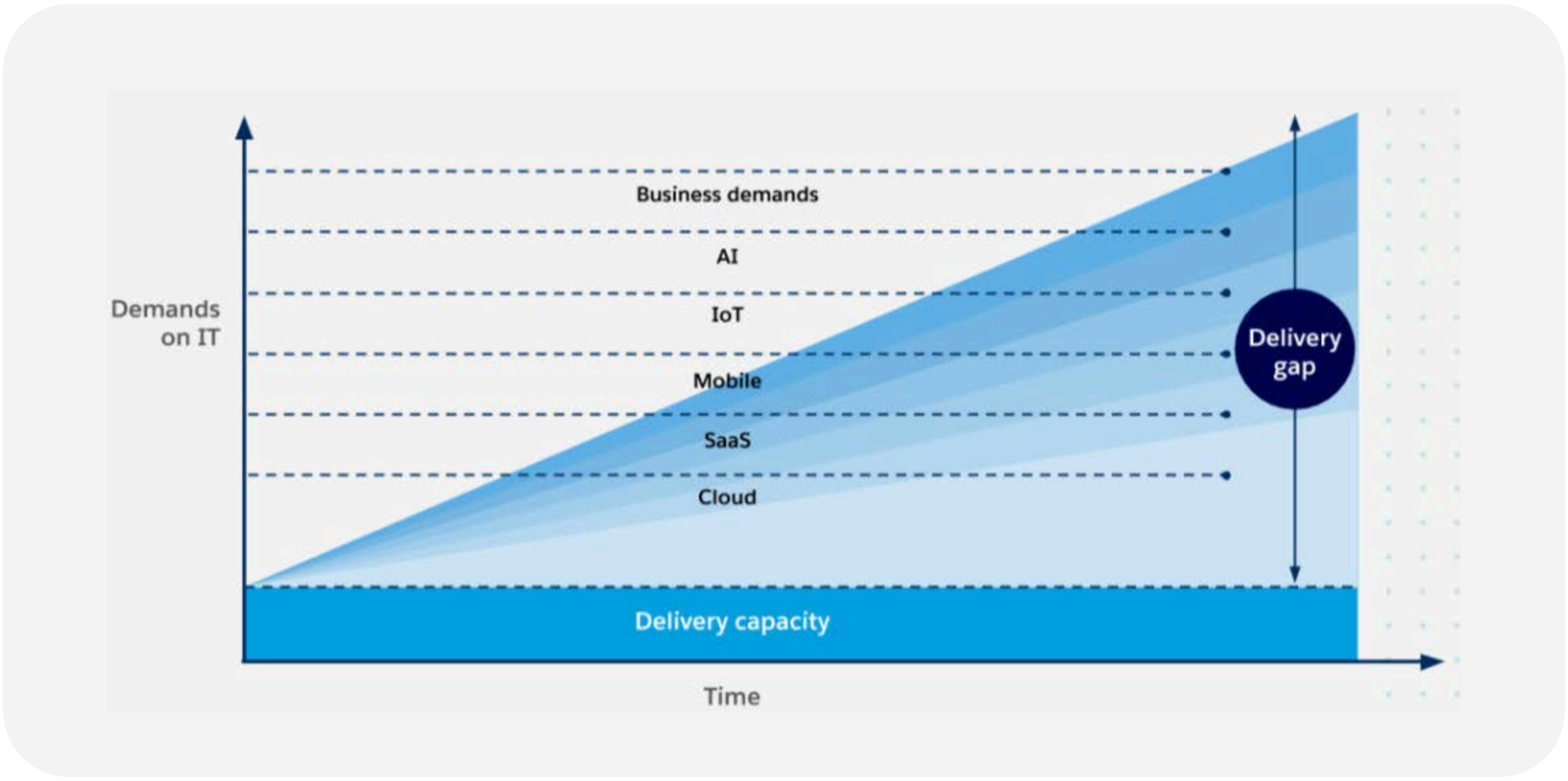
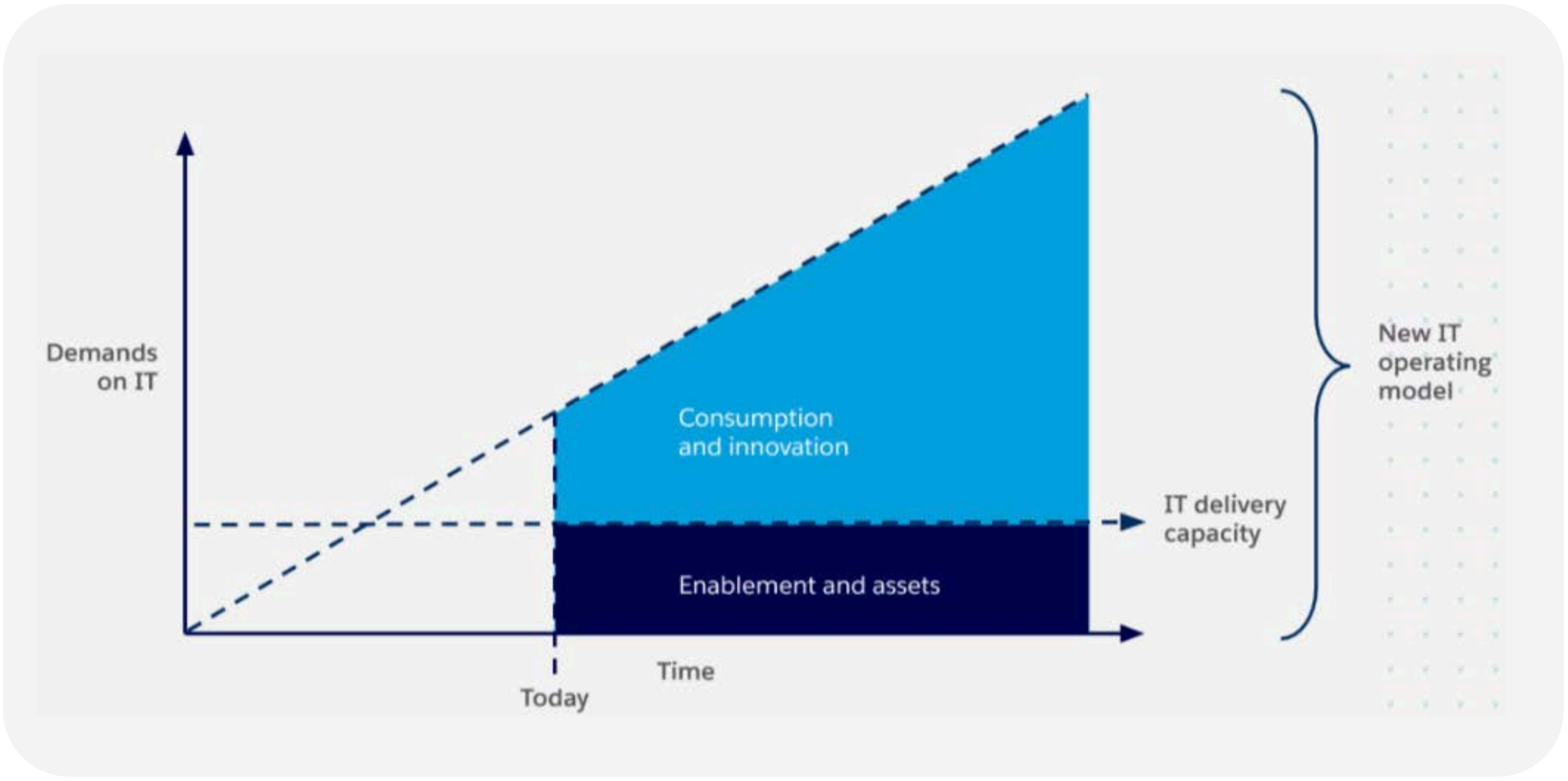
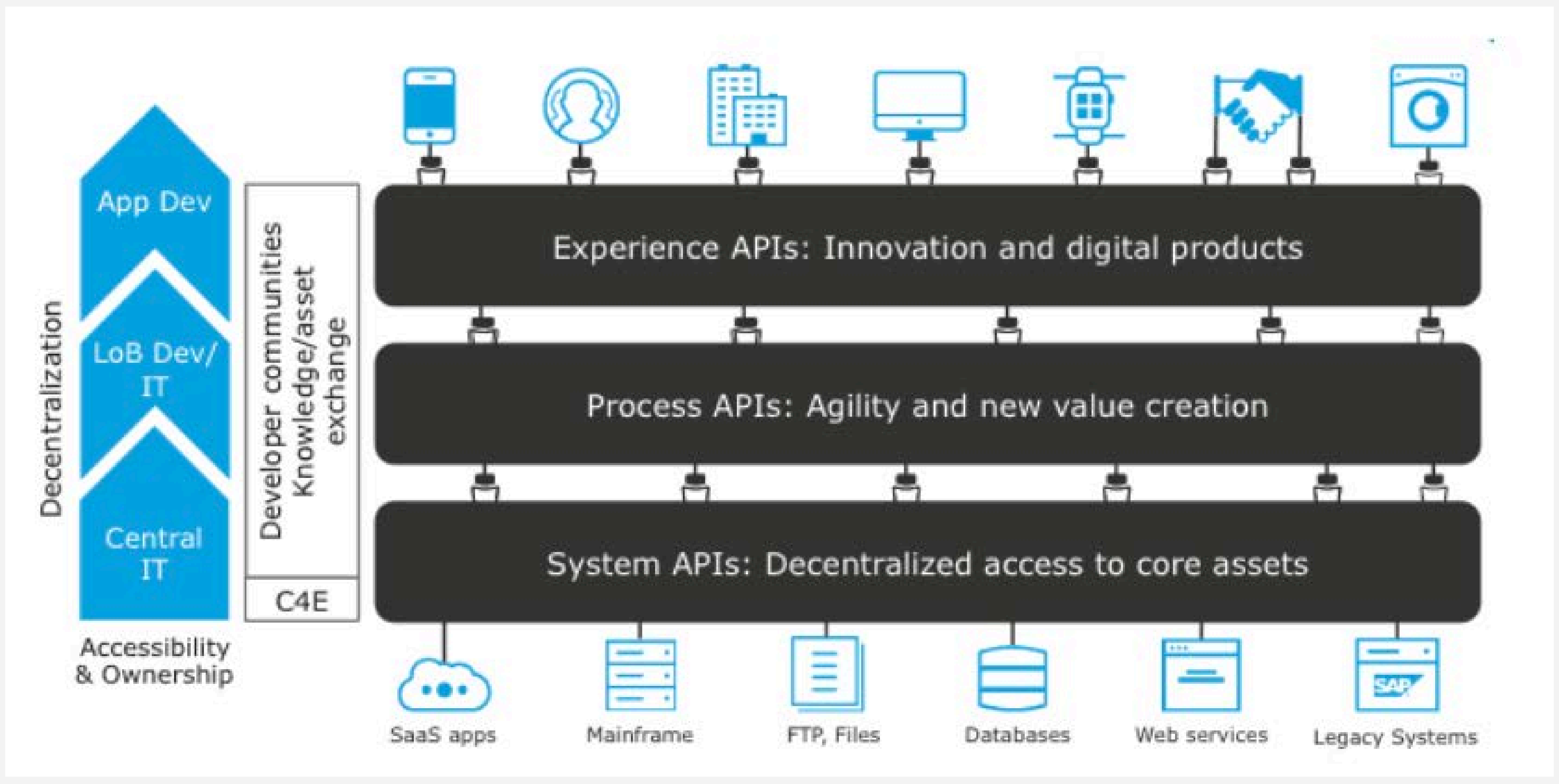


Illustration of the widening IT delivery gap caused by various forces. - **Mulesoft APAAppNets Student Manual**

Mulesoft's API led connectivity approach tries to solve this problem. API Led Connectivity is a way to connect the data to the applications using the reusable , purposeful API's. In this approach the API's are categorised into 3 types based on these building blocks which are: Interface, Orchestration and Connectivity.
Interface: Presentation of the data in a secured form. The API's which serve this purpose are called Experience API's.
Orchestration: Applying logic to data for transformation and enrichment.The API's which serve this purpose are called Process API's.
Connectivity: Accessing the source data through external systems. The API's which serve this purpose are called System API's.



How MuleSoft's proposal for an IT operating model that distinguishes between enablement and asset production on the one hand, and consumption of those assets and innovation on the other hand, allows the increasing demands on IT to be met at constant IT delivery capacity. - **Mulesoft APAAppNets Student Manual**


Decentralization
App Dev
LoB Dev/ IT
Central IT
Accessibility & Ownership
Developer communities Knowledge/asset exchange
C4E
Experience APIs: Innovation and digital products
Process APIs: Agility and new value creation
System APIs: Decentralized access to core assets
SaaS apps
Mainframe
FTP, Files
Databases
Web services
Legacy Systems
SAP

It is important to explain each API in detail.

System APIs - These are the API's which usually access the core systems and provide a layer of abstraction between the user and the core system. Once built these API's can be reused again and again and hence the first set of API's which are always created first are System API's. It is considered that the Central IT team creates these APIs and they are discovered and reused as assets in the entire organization.In terms of volatility these API's are considered as least volatile API's.

Process API's - These API's interact and shape the data in the format which is not specific to the source and target. They not only try to convert the data into the unified format but also perform complex business logic making them the most complex API's in the API led approach. The LOB Dev/IT team created these API's as they are usually use case specific. They are more volatile to changes compared to the System API's.It is not that they are only connected to the System and Experience API's meaning Process API can be connected to other process API whenever there is a need.

Experience API - These API's are designed by keeping the specific user in mind. That is there will be a specific Experience API for the web application and mobile application each. This is because the requirements of web and mobile applications are different and experience API's follow the API first design approach. These are the most volatile API's and act as an entry point for data from different systems. These API's are supposed to be the most robust API's because they are exposed externally. These API's are developed by the App developers.

Some of the benefits of API led approach are as follows:

1. Reusable API's - As solutions will not be B2B, there will be some API's (mainly System and Process API's) which can be reused in the new client opportunities.

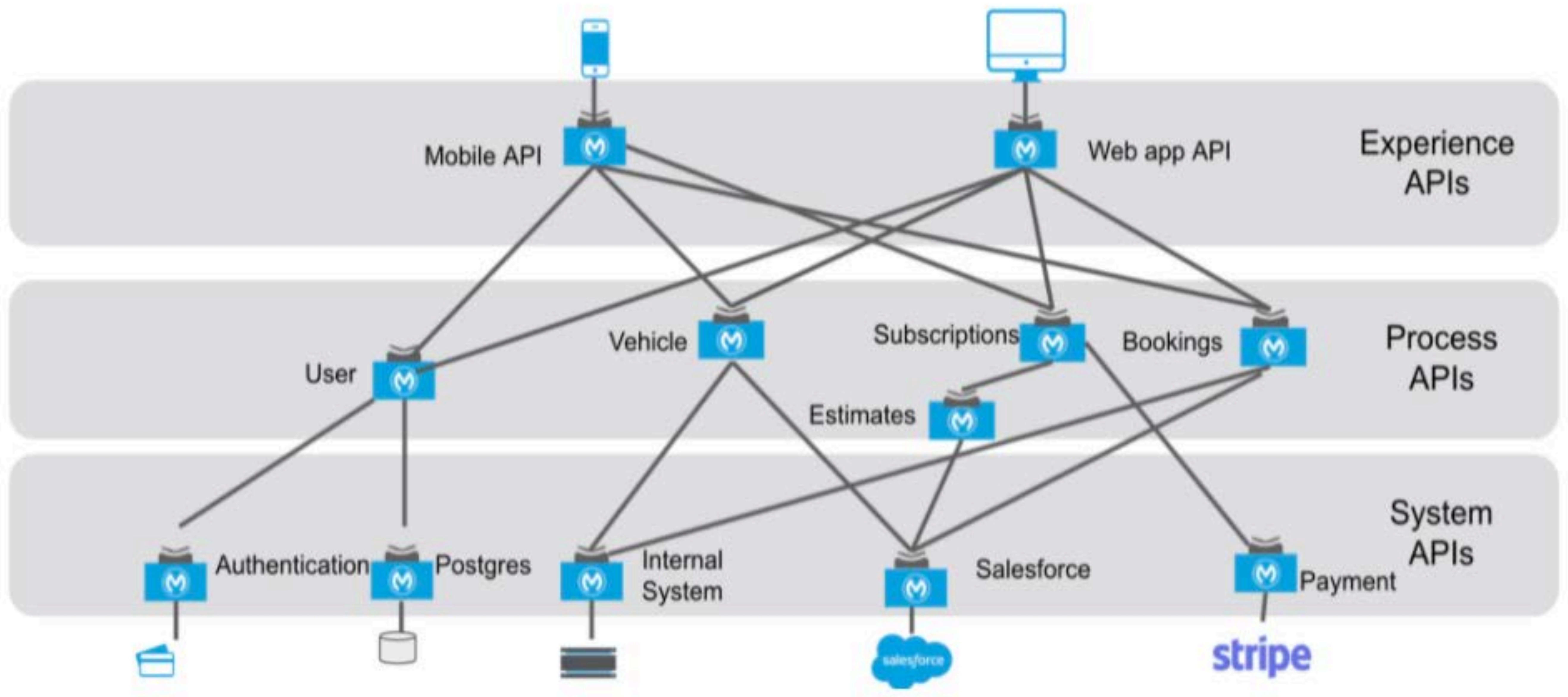
2. Faster delivery - Having reusable API's will reduce the development time to large extend and help in deliverities the API's quickly.

3. Easy enhancements and predictable changes - As there will be separate API's, any changes and enhancements will affect only the specific API in which the changes are needed without having to change the entire application.

How Showoff uses Mulesoft's API Led Connectivity Approach

Sonali Shah Mulesoft Developer

Below is an example of how Showoff has used API led Connectivity Approach in order to realise one of its customer demands.



Let's start with the steps of how we can realise the use case into the customer success story.

Steps for Successful Integration Strategy Realization

Sonali Shah Integration Architect

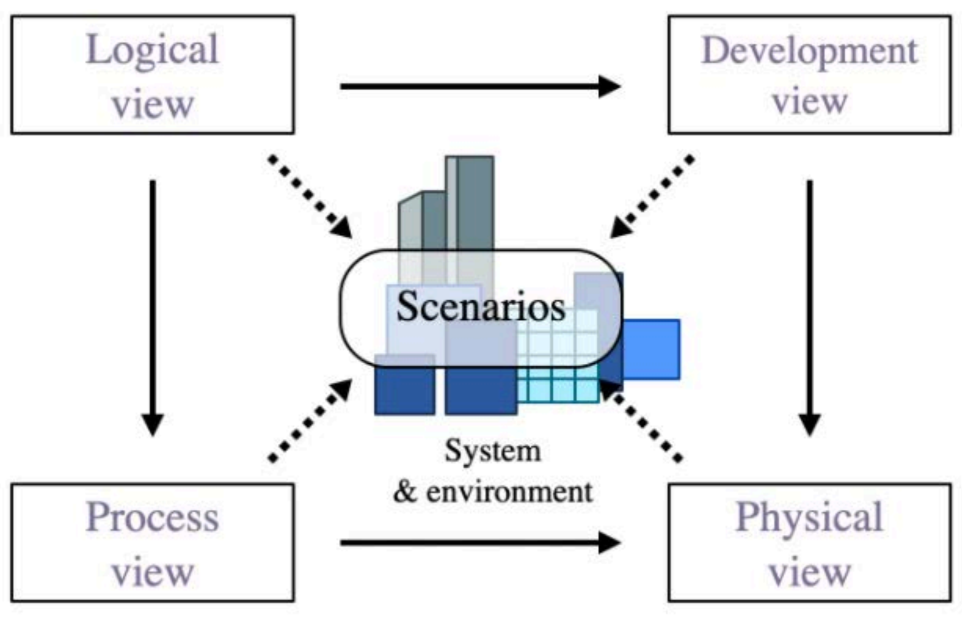
Identifying the Stakeholders and describing the objective of the use case

As soon as the requirements of the case study are understood it is important that the stakeholders of the use case are identified and as the life cycle of the development is started they are being provided with the necessary documents which help them to understand that the requirements are understood well and there is no discrepancy between the requirement gathering and implementation. Usually the important stakeholders are:

1. Project Sponsors
2. Architects
3. Integration Developers
4. Auditors
5. Users

Identifying the type of documentations in the integration solution architecture

As each stakeholder has a different role, providing the same set of documents to them is not considered the best practice and hence Mulesoft suggests the 4+1 approach to illustrate the views of the software system.



Let's talk about each view, scenarios and the stakeholders related to it.

Logical view - Describes the end to end functionality. It defines and documents the system, stakeholder, interface and their relationship. Logical views can be represented using the data models and relationship diagrams. Project sponsors and architects are the main stakeholders using the logical view diagrams.

Process View - Illustrates the runtime behavior of the system. Activity and State diagrams are used to represent the process views. It is more technical as it shows the relationship between the different systems while still skipping the integration details. The architects and the developers are the main stakeholders of the process view documents.

Development View - Describes the system from the developers perspective. It shows the components and the packages which can be used to code the solution. Development views have different categories of documents starting from the testing strategy to be used, resource information of the reusable libraries and standards and quality related information. Developers and Testers are the main stakeholders for the development view documents.

Physical View - Documents the system deployments and deployment diagrams are used to show the physical view of the system. Deployment Diagram shows the different components (systems), the artifacts (applications) and the connections between them. These documents include the high level topology, licensing, infrastructure details. Auditors, developers and the operations team are the stakeholders for these documents.

Scenarios - Help in illustrating the behavior of the system from the end user perspective. It helps validate the architecture design and requirements with the key stakeholders (users and project sponsors).

Identifying the Integration Pattern

Integration solutions are basically connecting one or more systems together. Integration patterns are proven solutions which easily solve specific problems. There are about 5 patterns which are commonly used to solve an integration problem:

Migration Patterns: Migration patterns are mainly used for migrating/syncing the data from one system to another. Batch processing and using the parallel for each component are the 2 main ways to solve a migration scenario.

Broadcast Patterns: Broadcast pattern is moving the data from one source system to multiple target systems. Using Parallel processing will help in reducing the processing time as the same data is to be sent across multiple systems. Mulesofts Scatter Gather component can be used to send the data across the multiple systems as it allows parallel processing.

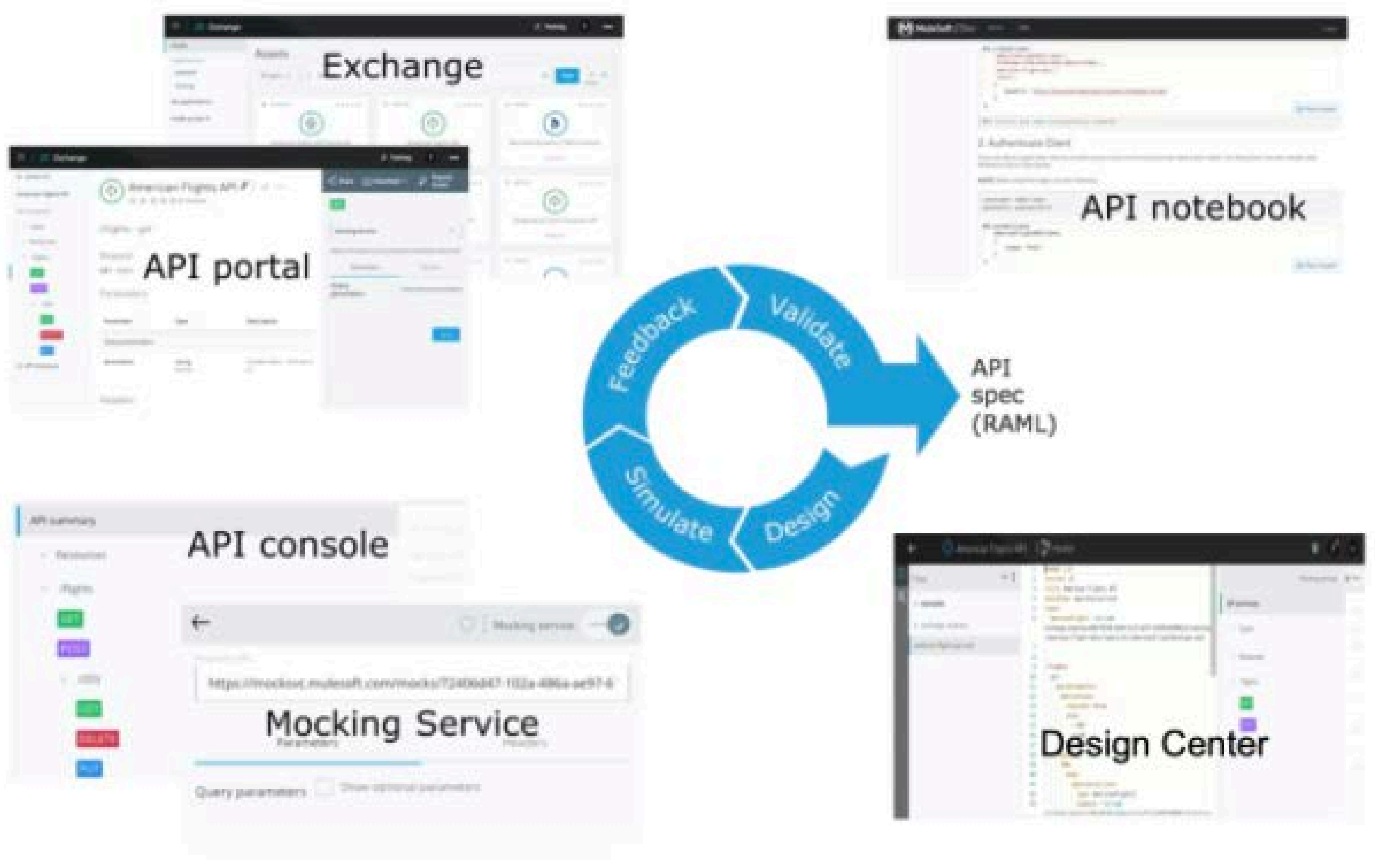
Bi-Directional Sync Pattern: This pattern is used when there is source to target and target to source. The requirement can be accomplished by using the asynchronous processing either by using the polling or webhooks. Webhook is a HTTP Callback which allows the target system to complete its processing and then call the data requester on the address given to the webhook.

Aggregation Pattern: In this pattern the data is to be fetched from multiple sources and given back to the caller. The best way to handle this requirement is to have multiple system API for each system and have a common process api to combine all the data and provide the result to the experience API.

Reliability Pattern: When there is a need to store the transactional data as the source system may lose the data after it has provided it to the Mulesoft API, the reliability pattern is used. Here the data is stored in the queues (JMS, VM). So basically as soon as the data is received it is published in the queue and then dequeued and then the processing is started.

Defining the API Specification and publishing to exchange

Once the pattern for the integration case study is decided, the next step is to create the API Specification which can be done using the API Design Center. Mulesoft recommends using the design first approach so that the requirements are well documented and then feedback can be obtained from stakeholders. Anypoint's Mocking service can be used to understand the request response from each of the functionalities defined in the specification.



Implementation and defining the standard support provided by each API

It is important to have a functional design document (FDD) specifying the functional and nonfunctional requirements to be implemented in the API. Developers should follow the FDD and perform a set of unit tests before deploying the application. For Showoff to consider an API implementation complete, we must make sure all its core API's have the following standards and supports provided:

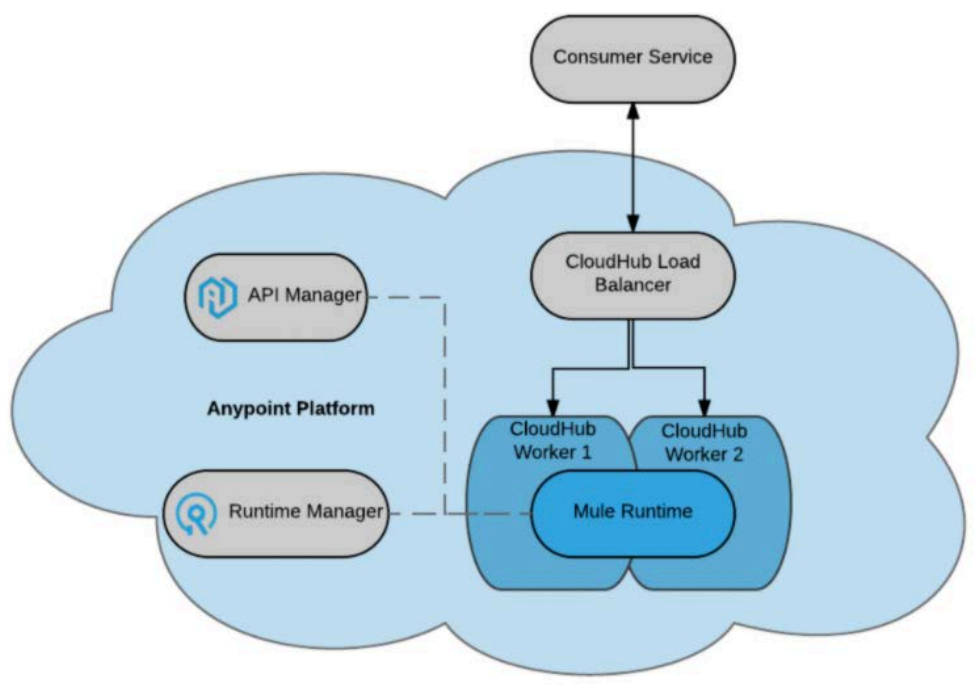
1. Having a proper authentication mechanism for accessing the API's
2. Using database/object store to have the persistent application data
3. Using caching strategies like HTTP caching or using 3rd party services like Redis
4. Error handling to be implemented and having standard error response defined
5. Tracking the application data by logging it into 3rd party applications like Splunk, Honeybadger.

Identifying the deployment strategy

Mulesoft provides various deployment options for its customers each having their own pros and cons. Here we will be speaking about each option along with the deployment option which Showoff considers suitable for its clients based on their requirements. Mulesoft manages the API Manager, Runtime Manager, Design Center etc in its control plane and it runs the applications specific to the user in the runtime plane.

1. Mulesoft Hosted Control and Runtime Plane with CloudHub load balancer

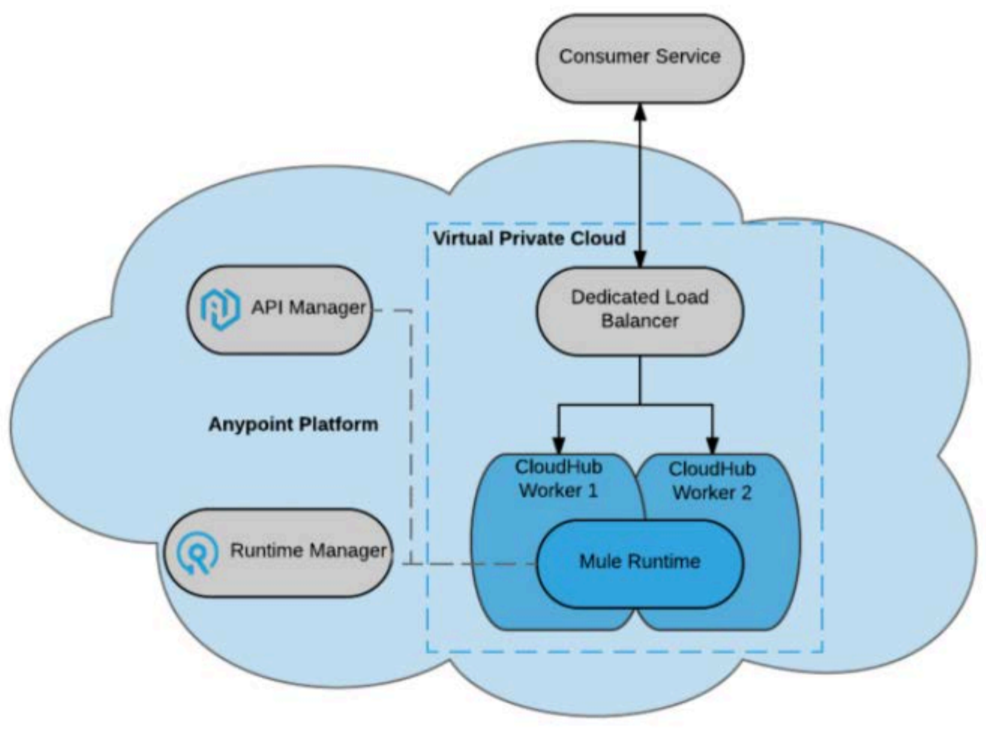
Anypoint Platform is a IPaas platform which provides a facility to deploy the applications directly into the Runtime Manager without the need to configure the runtime environment. The features provided by the API Manager like security for the API's etc can be configured in this deployment strategy option. Some of the disadvantages of using this strategy is that the applications do not have dedicated network range. As the Cloudhub load balancer is controlled by Mulesoft Customized DNS , URL Mapping options are not available.



2. Mulesoft Hosted Control and Runtime Plane with dedicated load balancer

In this model both the control plane and the runtime plane are managed by Mulesoft, along with the user's dedicated virtual private cloud where the user can have a dedicated load balancer to manage its applications. As the user will have a VPC, it can identify and whitelist the applications and the clients which can access its services and there will also be an option for having the customised domain name.

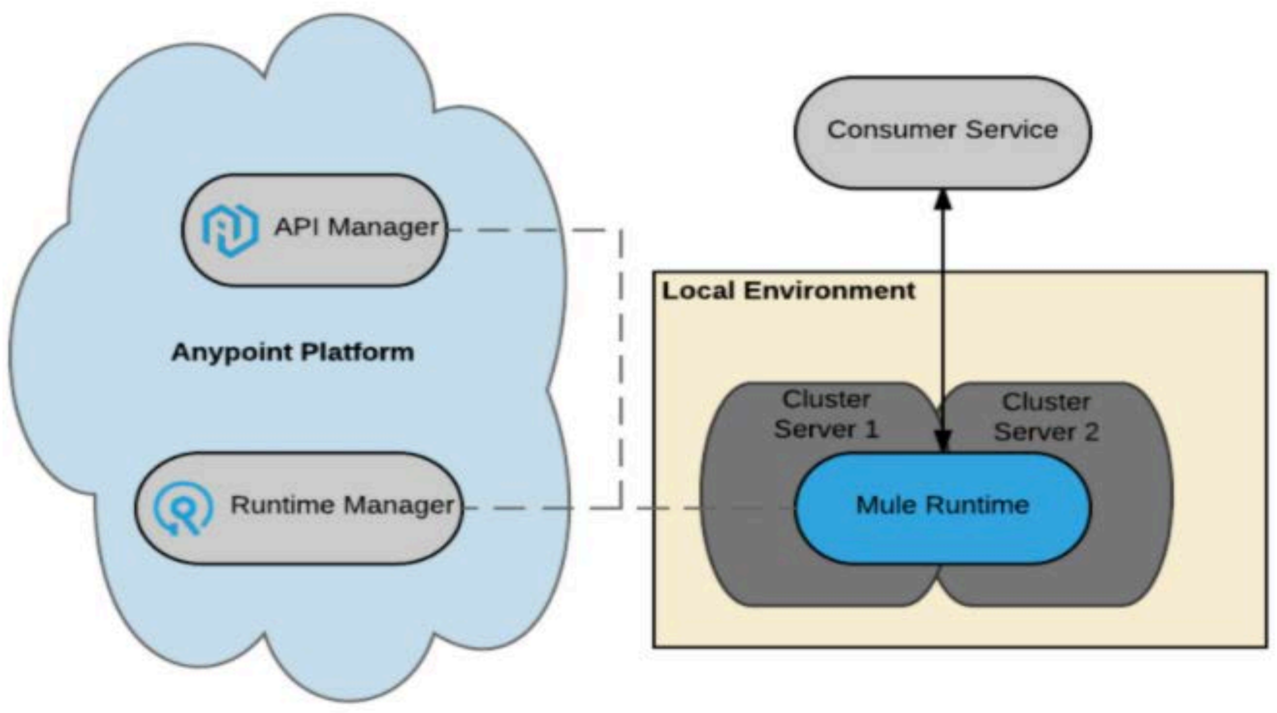
.



3. Mulesoft Hosted Control Plane with Customer Hosted Runtime Plane

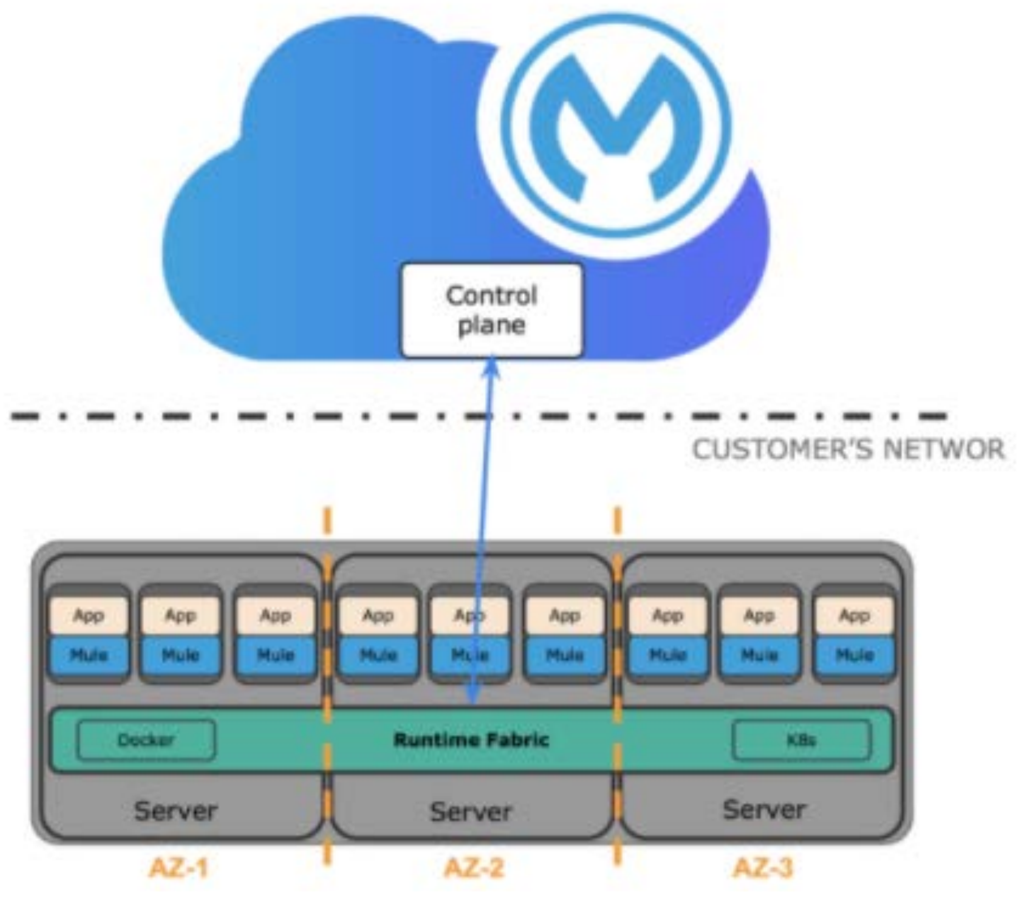
This model is usually preferred by customers who are using their own servers for deploying the applications. In the runtime manager there is the option of registering the server on which the mule application is to be deployed. The API manager can still help in providing limited security to the applications. The options of monitoring and troubleshooting are only available in hybrid approach. Load balancing has to be done on the customer side and a lot of network level expertise is required for this model.

.



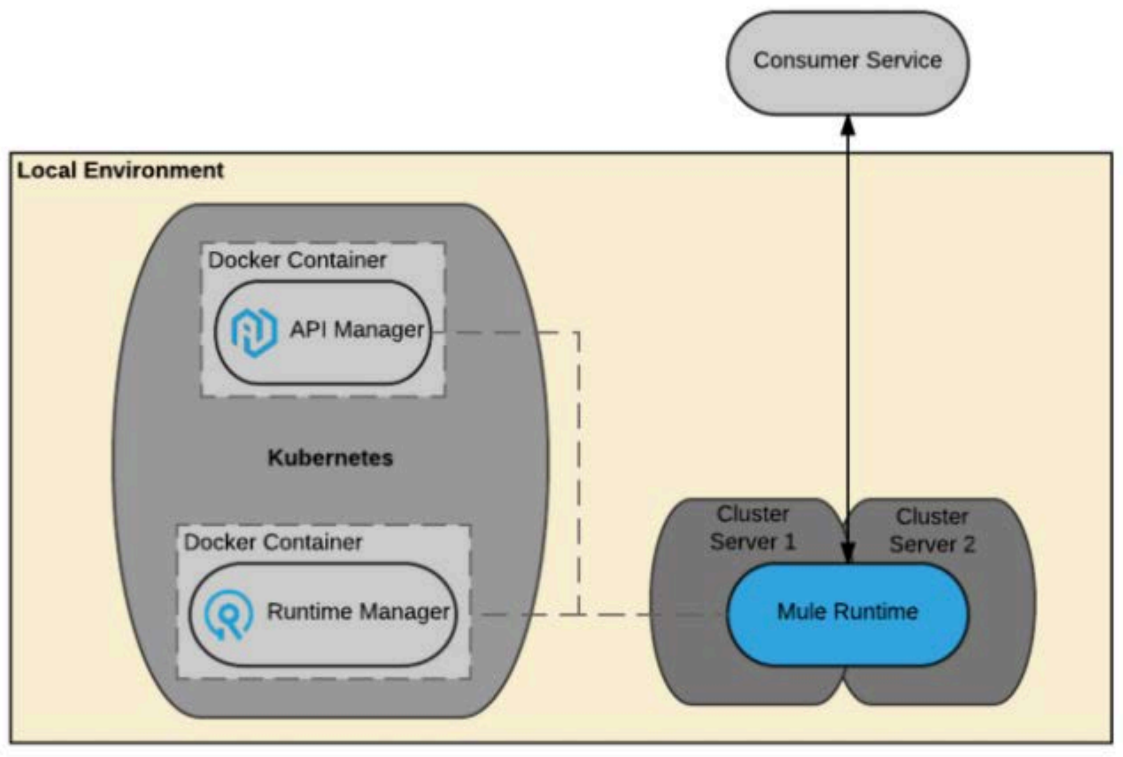
4. Mulesoft Hosted Control Plane with Customer Hosted Runtime Plane with Anypoint Runtime Fabric

This model is similar to the model 3, the only difference here is it uses Runtime Fabric which is a software appliance that provides customer-hosted iPaaS functionality comparable to CloudHub. It leverages a Kubernetes cluster to do so and executes Mule applications on Mule runtimes within Docker containers. Connection to the control plane is via AMQP/TLS initiated by Anypoint Runtime Fabric. Contains a load-balancer for HTTP traffic to replicated Mule applications (performs TLS termination).



5. Customer Hosted Runtime Plane and Control Plane.

In this model everything is managed in the customer environment. In this private cloud edition, Mulesoft provides containerised distribution management and engagement capabilities. This option is mainly used when there are strict compliance requirements to have the data and metadata both in the company's network.



Choosing a deployment model depends on the following:

Data isolation requirement - where the data and metadata has to be only in the company's network.

Security and compliance requirements from the client - whether the data has to be in specific region, private network etc

Company size and its infrastructure - whether the company already has a cloud service for deployment of its applications, budget to buy specific models mentioned above etc.

Implementing the Non Functional Requirements

It is important to gather the following non functional requirements for any application being created:

Performance

Average response time for the application is the important discussion to be carried out in the requirements gathering meeting with the client architects.

Availability

Whether the application has to be available the entire time and what steps need to be taken for its recovery has to be decided.

Security

Clients data is the most important thing and protecting it from cyber attacks is equally as important as its development, hence the various security-related non functional requirements for each layer can be strategized.

Scalability

Sometimes the API request for the application can vary and reach a breakdown point, which is why it is necessary to have the feature of auto scaling the application in such scenarios. Anypoint platform helps to achieve that goal.

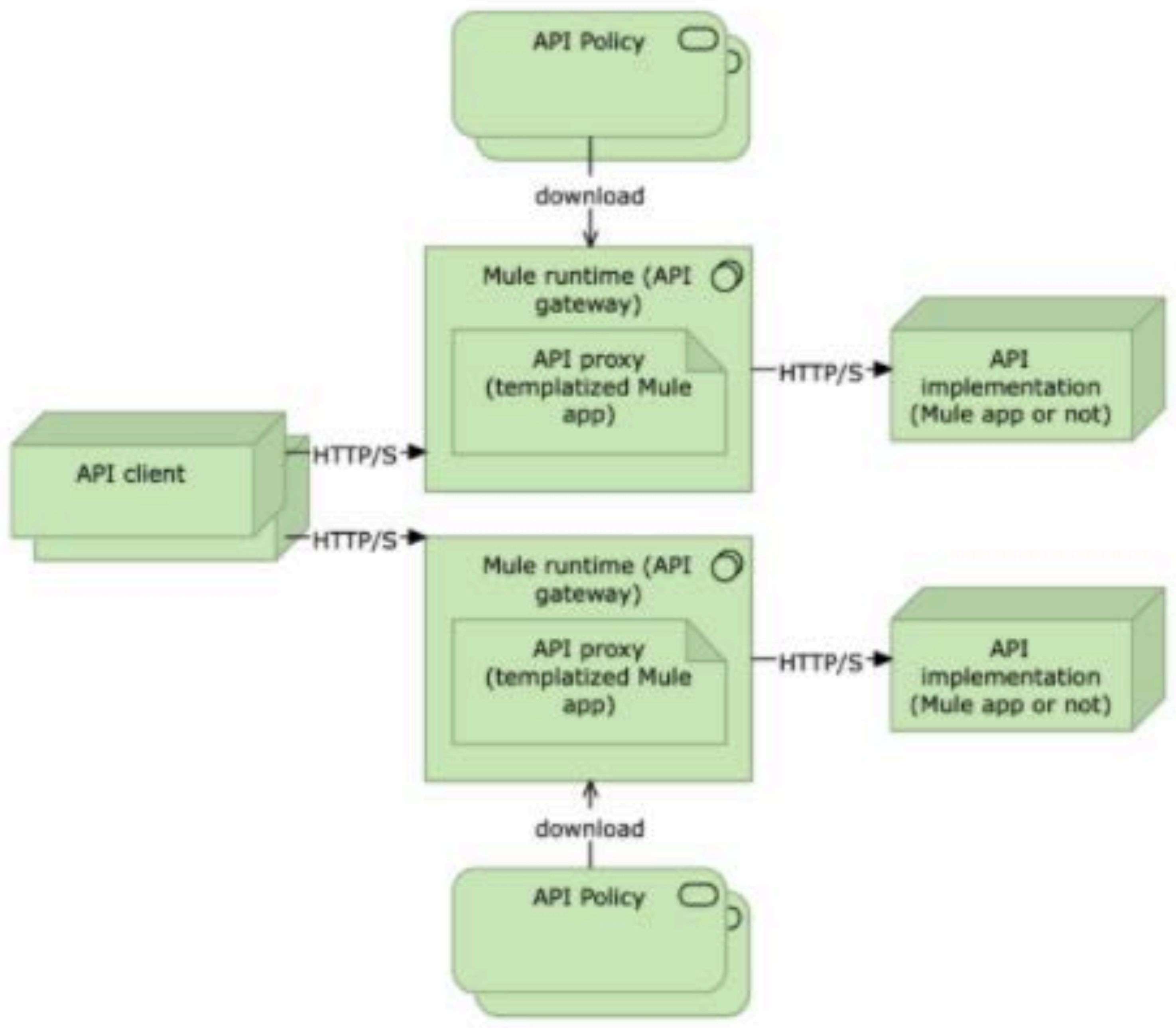
Based on Mulesoft's standard guidelines Showoff has found out the list of non-functional requirements that must be applied to its applications based on each layer.

Experience Layer: As experience layer is the single point of contact for the outside applications to contact, access to experience layer can be limited by allowing the specific list of IP's to contact it. Hence IP whitelisting can be considered as an option but it cannot be used when mobile applications want to connect to the experience layer. Also to protect the data entering the API XML/Json protection policies can be applied to it. Rate limiting and client identity enforcements can also be applied to experience layer API's.

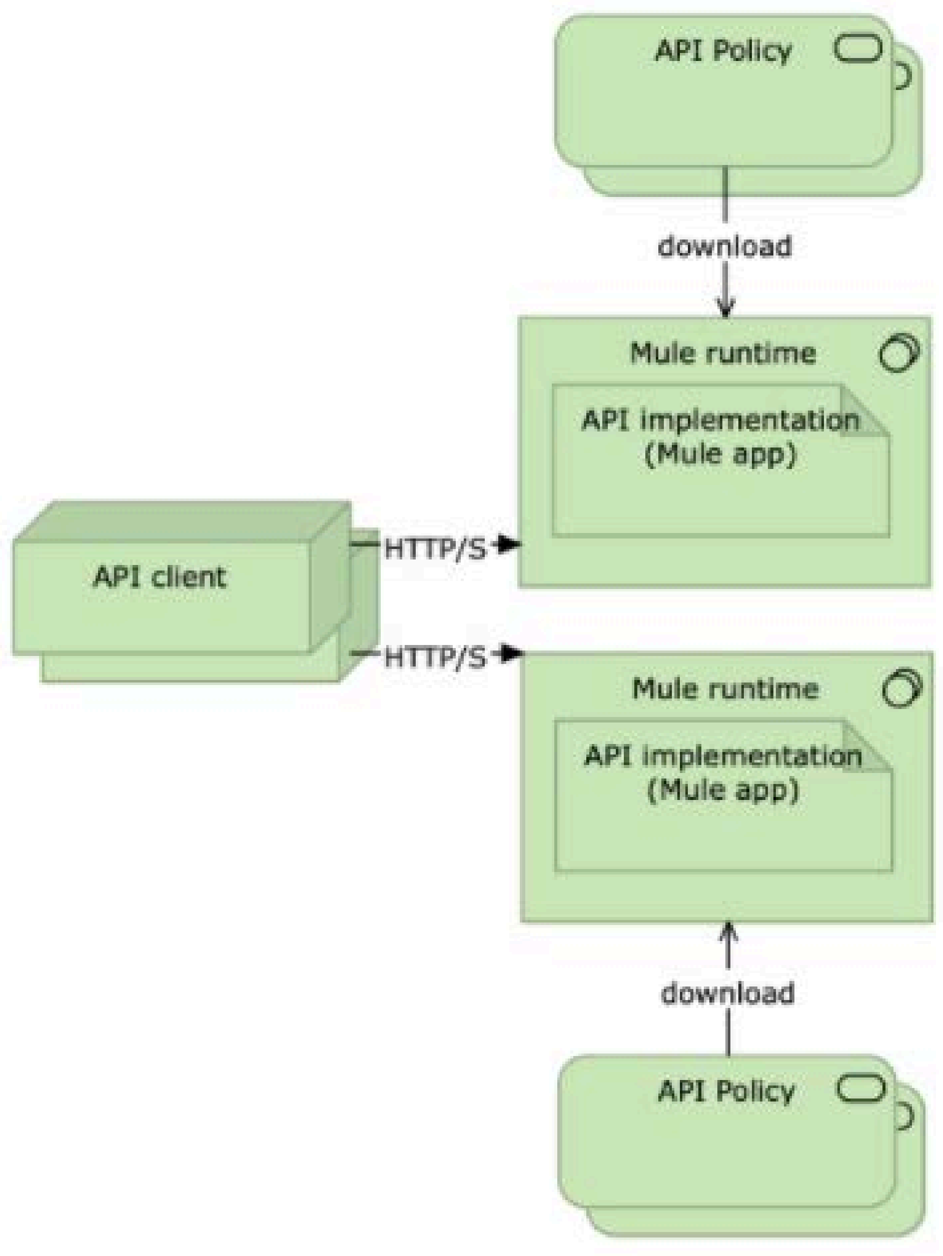
Process and System Layer: IP Whitelisting, Spike control and Rate limiting SLA based as the major non-functional requirement policies applied to these layers.

Mulesoft's Anypoint Platform provides the capabilities of implementing the above non-functional requirements by just applying the API policies to its applications. The API policies are listed by the API Manager. The API Policies can be applied to the applications using 2 different ways:

1. By creating a proxy application and applying the policies to this application which internally connects to the main application.

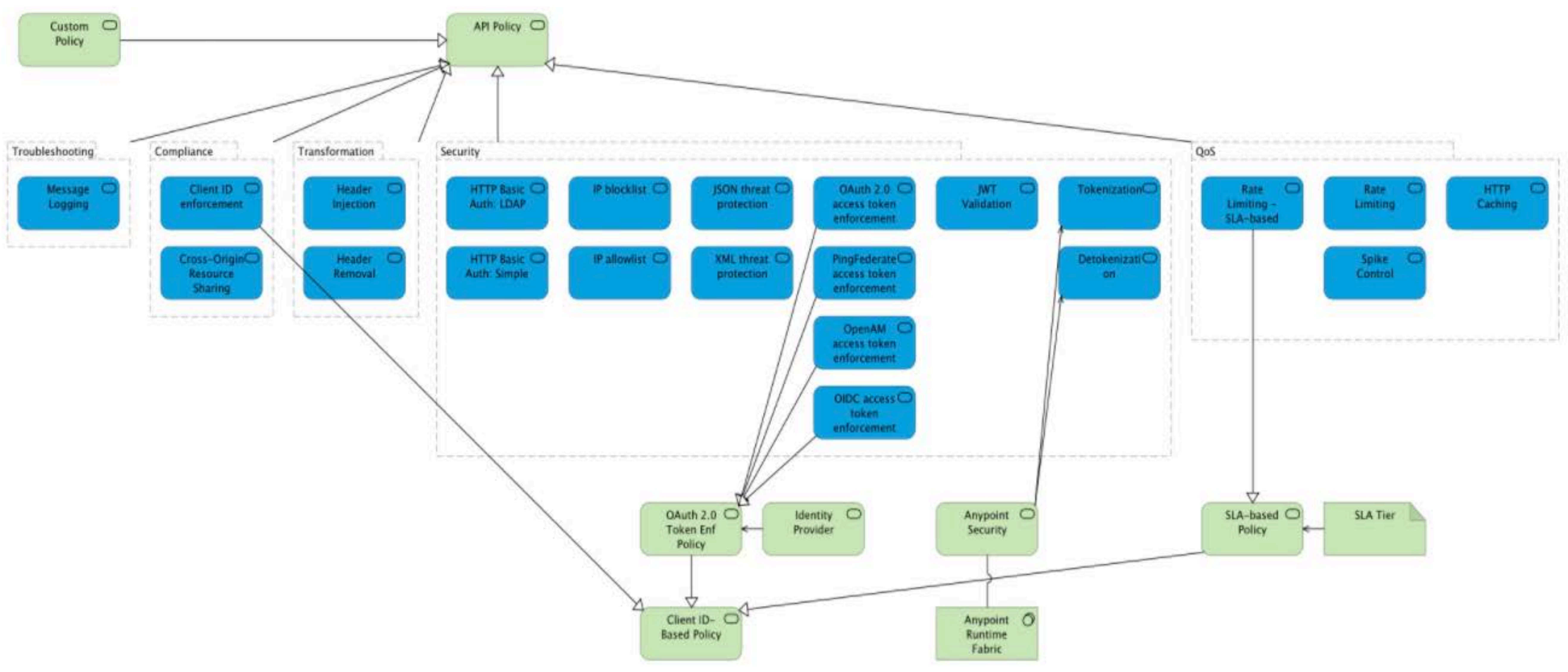

API Policy
download
Mule runtime (API gateway)
API proxy (templatized Mule app)
HTTP/S
API implementation (Mule app or not)
API client
HTTP/S
HTTP/S
Mule runtime (API gateway)
API proxy (templatized Mule app)
HTTP/S
API implementation (Mule app or not)
download
API Policy

2. By directly applying the API policies to the application using the feature of API Autodiscovery.



Selection of the above option is usually dependent on the API, i.e for experience API it is usually considered best practise to have a separate proxy application to avoid making the experience API bulky. It is also best practise to use the API policy within the application itself for system and process API's as this helps in saving the vcores and they will usually be protected by the experience API's.

The following diagram represents the policy options available in the API Manager for implementing the Non-Functional Requirement



Implementing the alerting and monitoring strategy in Anypoint Cloudhub

Developing and deploying the application is just one part of the work done but to maintain it is another part of it. Anypoint platform provides the capability to maintain the deployed applications by usings its services like Anypoint Visualizer, API Monitoring and Alerts.

Anypoint Visualizer is mainly used to visually represent the different applications present in the application network. It can act as a starting point for any senior developer or architect to get to know the company's API network. It also helps in identifying the duplicate dependencies between different applications in the network.
It is considered as a best practice to assign the applications as nodes in the network and then assign the nodes to the different layers i.e experience, process and system to which the application belongs.

API Monitoring helps in understanding the API's behavior based on the various metrics like:

1. Input response time
2. Output response time
3. Infrastructure usage
4. Average response time
5. Failures

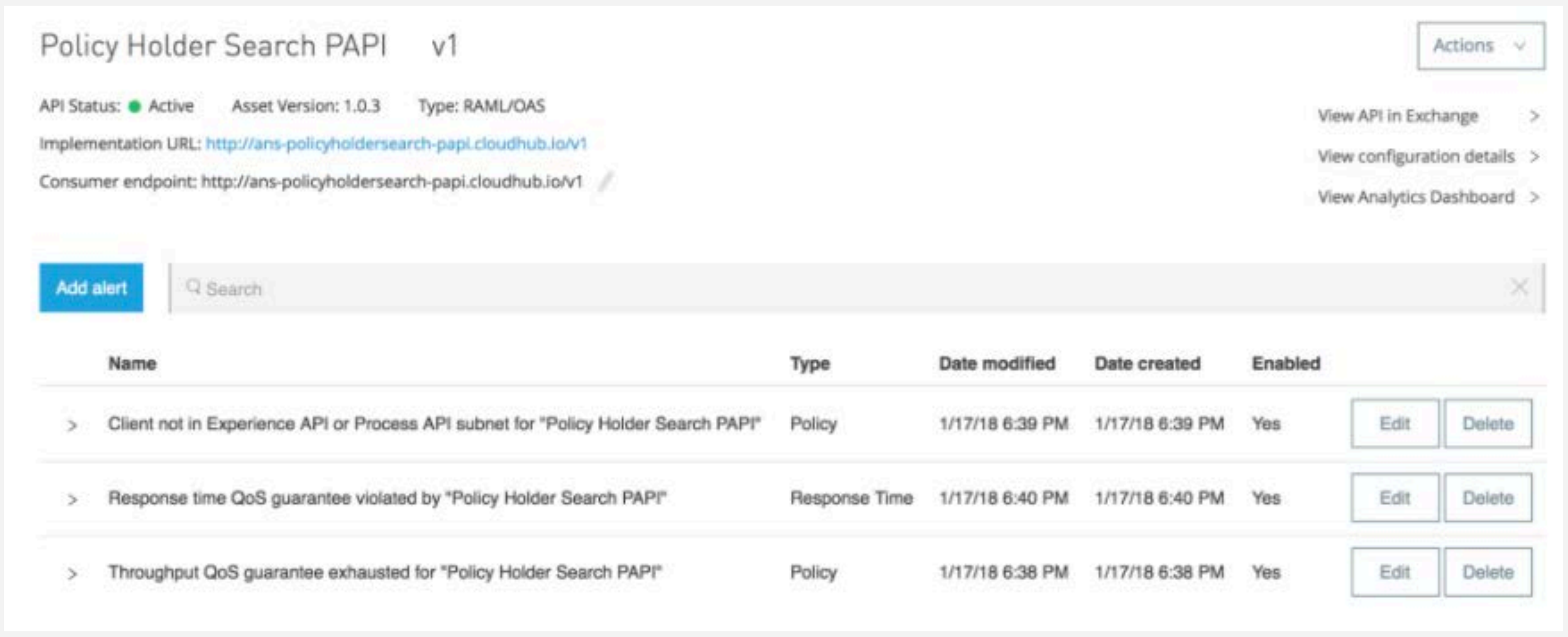
Built-in dashboards are created based on the above metrics and can be used by the support team to understand the behavior of the API's. It is considered as best practice to have a consolidated copy of all the dashboards emailed to the relevant team as a health check process or all the dashboards thoroughly checked every day.

API Monitoring is not just generating the dashboards related to the different API's but also to raise the alerts when there are any unusual or failures generated by the API's.

Alerts can be raised based on a number of reasons as listed below:

1. Violations to the policy
2. Failure error codes
3. Response time exceeding a given threshold
4. Number of API invocations

Below is the sample example of how alerts can be defined in the Anypoint platform.



Policy Holder Search PAPI v1

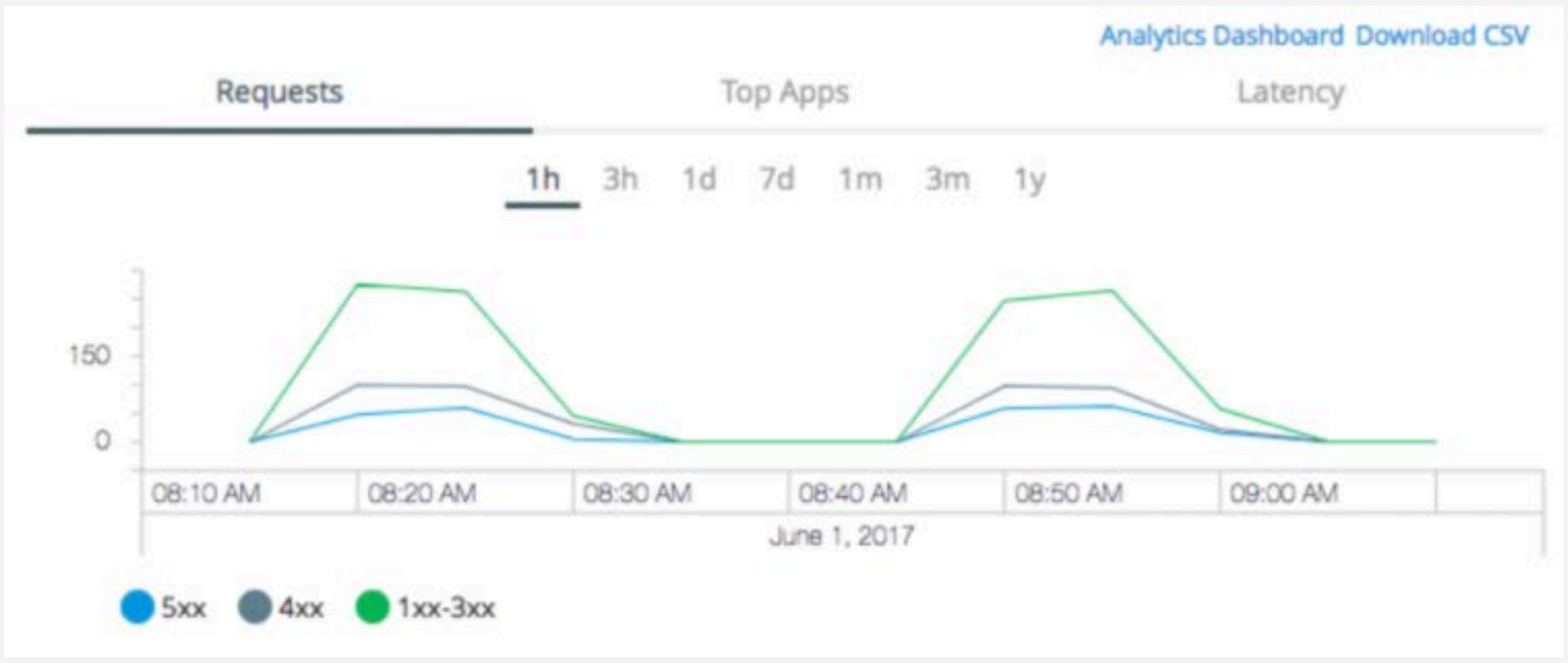
API Status: ● Active Asset Version: 1.0.3 Type: RAML/OAS

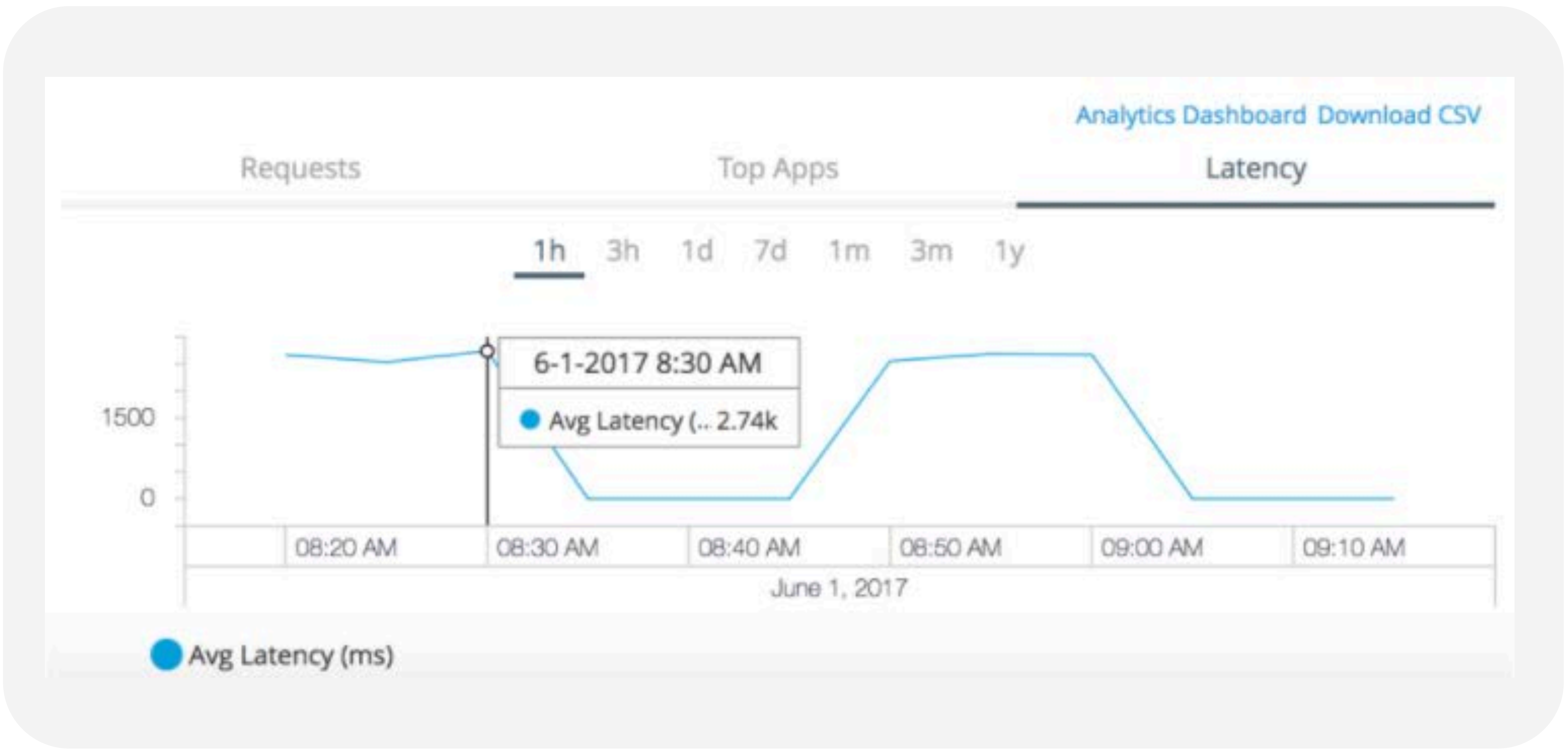
Implementation URL: http://ans-policyholdersearch-papi.cloudhub.io/v1

Consumer endpoint: http://ans-policyholdersearch-papi.cloudhub.io/v1

	Name	Type	Date modified	Date created	Enabled		
>	Client not in Experience API or Process API subnet for "Policy Holder Search PAPI"	Policy	1/17/18 6:39 PM	1/17/18 6:39 PM	Yes	Edit	Delete
>	Response time QoS guarantee violated by "Policy Holder Search PAPI"	Response Time	1/17/18 6:40 PM	1/17/18 6:40 PM	Yes	Edit	Delete
>	Throughput QoS guarantee exhausted for "Policy Holder Search PAPI"	Policy	1/17/18 6:38 PM	1/17/18 6:38 PM	Yes	Edit	Delete

The following diagrams represent the different dashboards which can be created using Analytics for monitoring the API's.





It is a mandatory practice at Showoff to create and consistently check the following 3 alerts for each API:

1. When the threshold for the API failure response code crosses a specific limit

2. When the API is down for more than the specific amount of time

3. When the CPU utilization is more than 70%

Conclusion

Sonali Shah Integration Architect

To conclude, using Mulesoft's API led connectivity approach helps Showoff to create microservice based API's which can act as assets and can be used by various clients, helping Showoff deliver faster time to market.

Following the best practice guidelines discussed in the article helps us to guarantee successful delivery to our clients. To summarise, the best practise guidelines once again are:

- Strict documentation of the requirements in the form of functional design documents for Developers
- High level project architecture documents for the sponsors and architects etc
- Identifying the correct architecture pattern
- Following the correct implementation guidelines for functional and non functional requirements
- Identifying the deployment models and setting up the correct monitoring and alerting strategies.

Ultimately following the above proper guidelines ensures better understanding of customer requirements and a reduction in issues which leads to faster, scalable and cost effective delivery of top quality secure solutions.

Showoff

Showoff is a Data Solutions company. With our partners at Salesforce, Mulesoft and Stripe we focus on three main areas for our customers:

- Liberating & transforming data from disparate and siloed systems
- Supporting rapid digital transformation in established enterprises customers
- Bringing new revenue streams into existing businesses with a particular emphasis on customer retention and growth

Founded in 2012, Showoff has grown to be a key partner for our customers and partners alike, delivering premium solutions in an expedited and clinical fashion. Funded by €4,000 & 80 Sheep, Showoff has grown to be a key player for businesses seeking to innovate around their actionable data. With over 300 projects delivered, we are focused on growing our market share in the Automotive, Transport, Retail space, FinServ and Professional services sectors.

MuleSoft®

Mulesoft, A Salesforce Company

Mulesoft, the world's #1 integration and API platform, makes it easy to connect data from any system - no matter where it resides - to create connected experiences, faster. Thousands of organizations across industries rely on Mulesoft to realize speed, agility and innovation at scale. By integrating systems and unifying data with reusable APIs, businesses can easily compose connected experiences while maintaining security and control. Through API-led connectivity, customers unlock business capabilities to build application networks that deliver exponentially increasing value. Mulesoft is the only unified platform for enterprises iPaaS and full lifecycle API management, and can be deployed to any cloud or on-premises with a single runtime.

If you would like to get a deeper understanding of how the API led approach can help your business please get in touch with the Showoff team. https://www.showoff.ie/contact