**MLC Solution Architecture – Technical Exercise**

MLC Architecture Design for the Rocket Launch API

**Prepared by:**

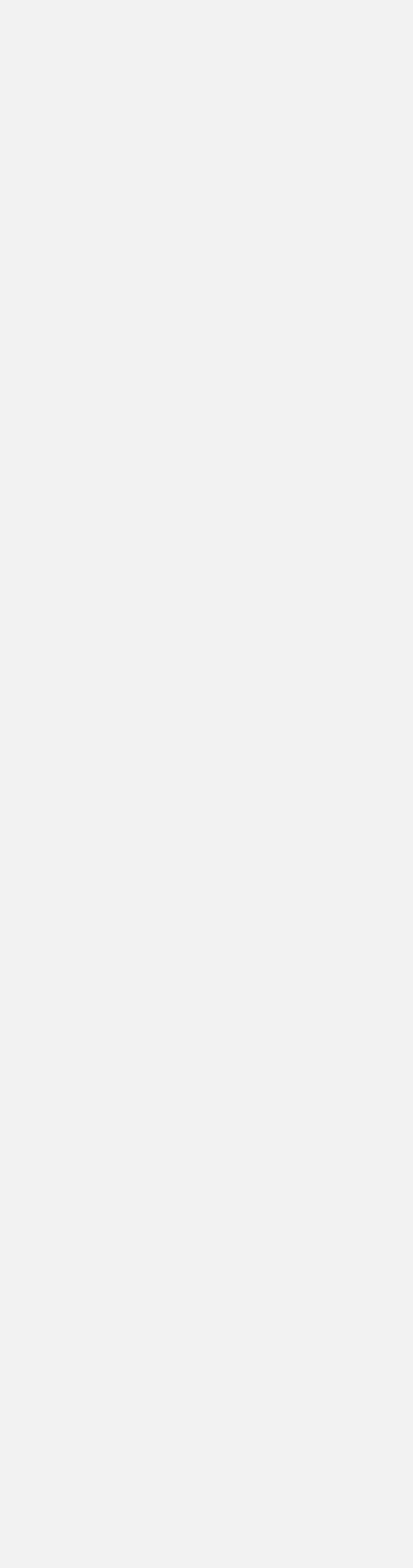
**Solutions Architect: Paul Dominguez**

**Enterprise Architect:**

**Business Analyst:**

**Version:** 0.1

**Date:** 17th March 2018

TABLE OF CONTENTS

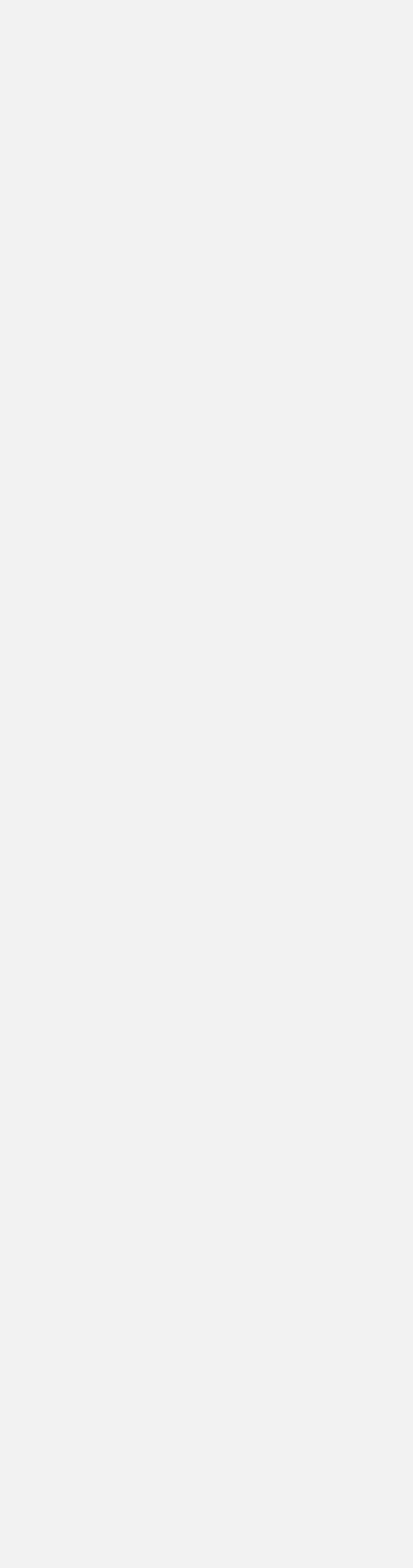
1. Document Control

1.1. Document History



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Version** | **Version Date** |  | **Updated by** |  |  | **Description of Changes** |
|  |  |  |  |  |  |  |
| 0.1 | 18/03/2019 |  |  | Initial Version | | |
|  |  |  |  |  | | |



1.2. Document Milestones

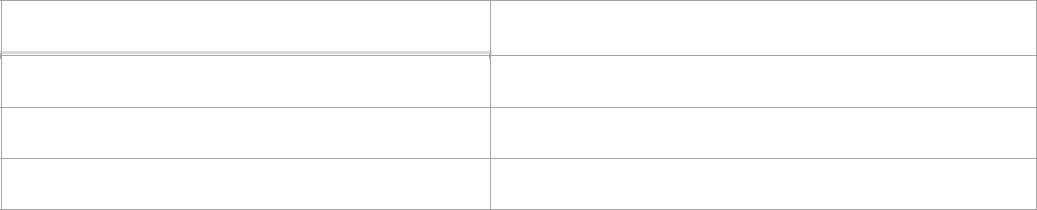
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Doc Version** |  | **Date** | **Role/Person** |  | **Milestone** | **Information** |
|  |  | **Type** |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  | Distributed |  |
|  |  |  |  |  | for review |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  | Signed off |  |
|  |  |  |  |  |  |  |



1.3. Distribution List

|  |  |
| --- | --- |
|  |  |
|  | Enterprise Architect |
|  | Solution Architect |
|  | Business Analyst |

1.4. Related Documents



|  |  |
| --- | --- |
| **Related Document** | **Link** |
|  |  |

2. Introduction

The following document describes the Solution Architecture for the Rocket Launch API Application. This project will be implemented the using severless Azure Logic Apps and Azure Functions and will be using the native API Management.

2.1. Acronyms and Glossary

The following table describes some of the acronyms and terms used along the document:

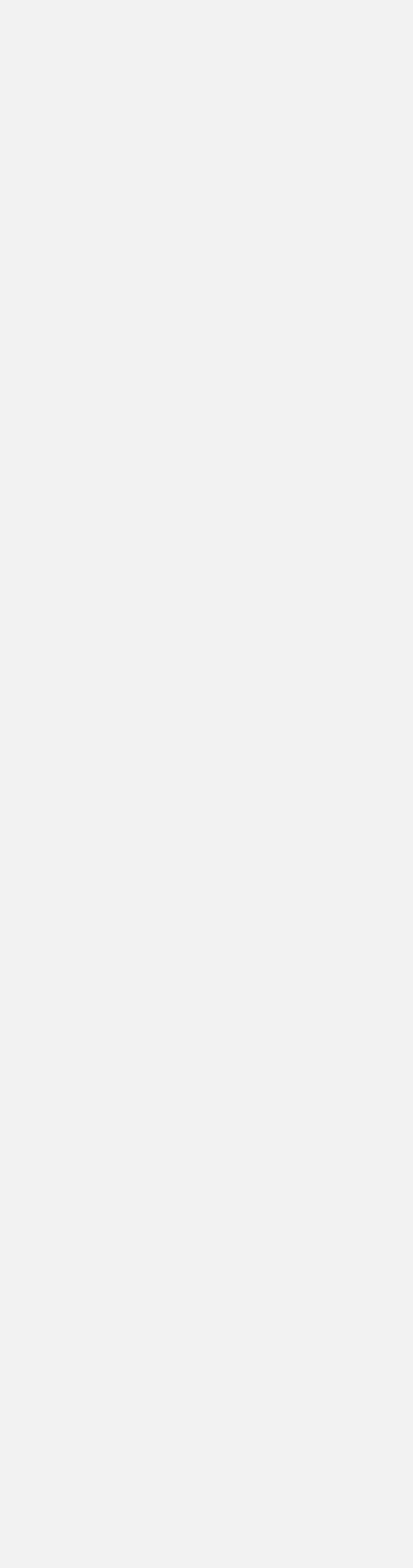
|  |  |
| --- | --- |
|  |  |
| HLD | High Level Design |
|  |  |
| API | Application Programming Interface |
|  |  |
| REST | Representational state transfer (also known as RESTful) |
|  |  |
| HTTP | Hypertext transfer Protocol |
|  |  |
| HTTPS | Secured Hypertext Transfer Protocol |
|  |  |

3. High Level Design

The High Level Design (HLD) describes a very high level of the serverless architecture of the solution, as well as the main systems and information flows that implement the use cases of the solution for the Rocket Launch



1. The Rocket Launch API is managed by the Azure API Gateway to enforce governance and policies
2. The API will resolve to a callable Logic App that will trigger the workflow
3. Lastly, Several components will be using serverless functions for mappings and further manipulation where logic app components are not fit for purpose

.

Functional Requirements

4.1. Context

The API should be able to provide the top five(5) time and location suitable for Rocket Launching. The information will include an environment score based on the weather conditions. A certain threshold per location is used to validate appropriate conditions for launching activities

4.2. In-Scope Use Cases

The solution presented in this document covers the integration that supports the following use cases:

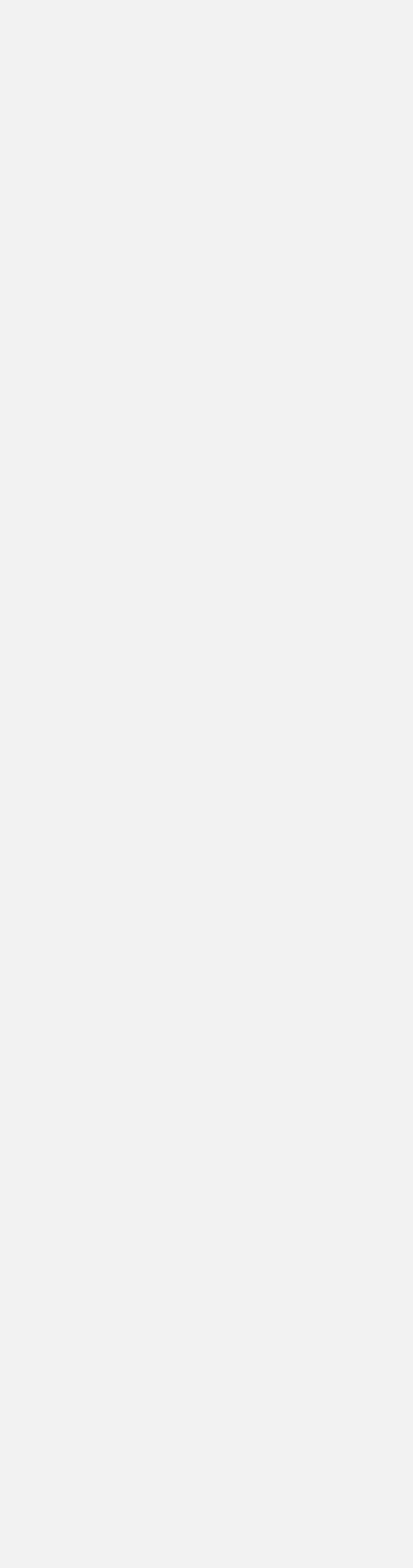
* Consumers are able to get the list of top 5 rocket launch window per selected location
* Consumers should be able to get list top 5 rocket launch window for all available location

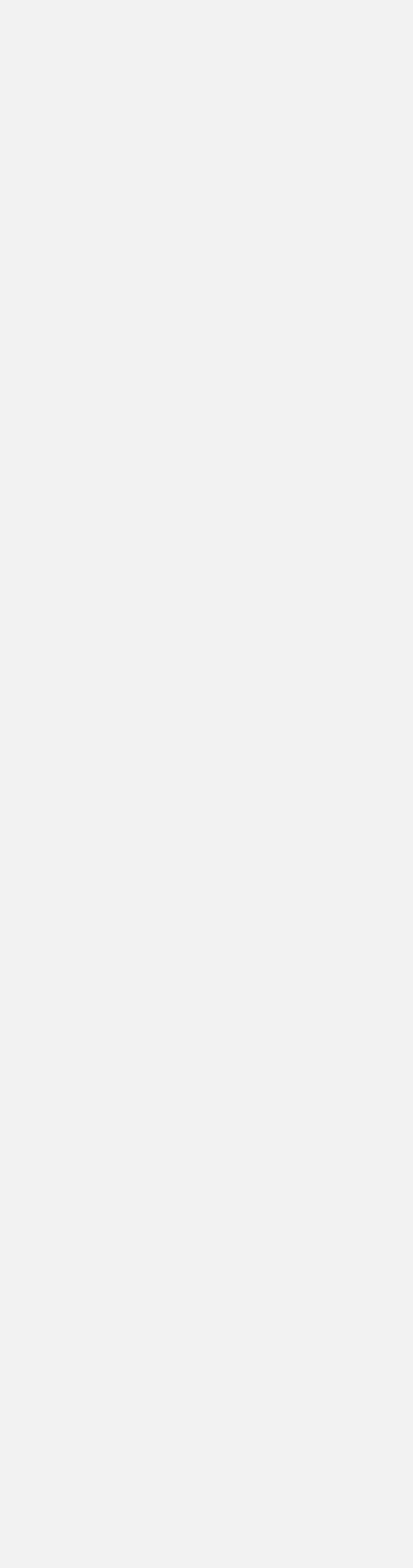
The following table summarizes the functional requirements that are going to be implemented in the solution. Every use case has a unique identifier that will be used later for traceability.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case # | Use Case Scenario | Process Summary | Data Sets (payloads) required/format | Integration Endpoints Involved |
| API-1 | Process API | As a consumer, I would like to get a list the top 5 rocket launch window for a certain location | JSON | RESTful API |
| API-2 | Process API | As a consumer, I would like to get a list the top 5 rocket launch window for all available location | JSON | RESTful API |

1. Design Decisions

This section describes the design decisions made by the team during the meetings and discussions while architecting the solution. These decisions cover many different domains of the solution: Business, Application, Data, Technology, Security and Operations. Many of these decisions are also detailed in the rest of the document.

* **Estimated $ Value calculated for the API should be minimal or free**
* **Multiple public APis will be consumed therefore there is no expected cost** /2017 3:56 PM
* **Networking Topology:** There will be an Azure subscription instance on free tier that will host the logic apps and azure functions
* The resource path of the APIs will be kept simple, the query parameters will be used to filter the response.
* **Security**: is going to be implemented on Azure API Management for the API
* **More decisions in point 5.1.**

6. Non functional Requirements

6.1. Reusability

As a principle and a requirement that will shape this solution and other future solutions based on the Serverless Microservices approach, the logic apps and functions should be callable and delivered should be “reusable”. By “Reusable” it is understood that the number of consumers of those APIs/functions are multiple, and, therefore, the Functional Granularity of the APIs should be quite generic and agnostic to a particular usage of the API.

6.2. Security

The following security requirements will be applicable after testing:

* Authentication: To enforce an authentication policy in the APIs to make sure that consumers are identified.
* Authorization: To enforce an authorization policy to guarantee that consumers of the API have the right to use a particular functionality of the API.
* Integrity: Data cannot be changed in transit.
* Confidentiality: Data in transit cannot be seen by a peer, which is external of the communication endpoints.

6.3. Volumes

Average number of requests will be determined on testing and historical data.

6.5. Auditing

For the POC, there hasn’t been identified any auditing requirements. If needed in the future the auditing requirement will be implemented via a log aggregation application.

6.6. Scalability

The solution should be scalable and it will allow an increase of processing power if the load conditions of the system requires it. As we are using serverless, it has the autoscaling capability

6.7. Monitoring

The solution should be able to provide monitoring capabilities and alerting, in order to identify anomalies and malfunctions.

Please see “Operations and Support” section where the solution to the monitoring solution is presented.

6.8. Exception Management

Due to the nature of all the interactions described in the Use Cases (all of them follow the synchronous request-response message pattern), the Exception Handling required is to reply an error message back to the consumer. The error message should contain information about:

* The http status code
* The http Error Code

6.9. Archiving

Since there is no data that needs to be persisted. No Archiving will be provided

6.10. Disaster Recovery

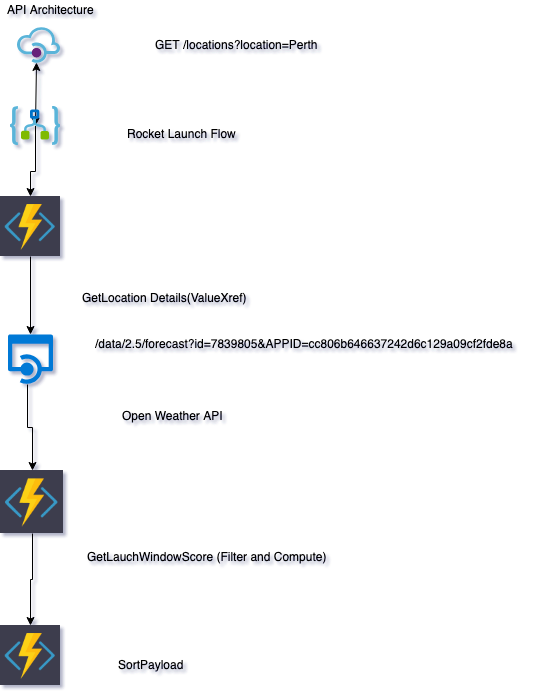
The solution will use the standard Azure disaster recovery mechanism.

7. API Architecture

The Rocket Launch API will consume all the Azure function as callable API as needed

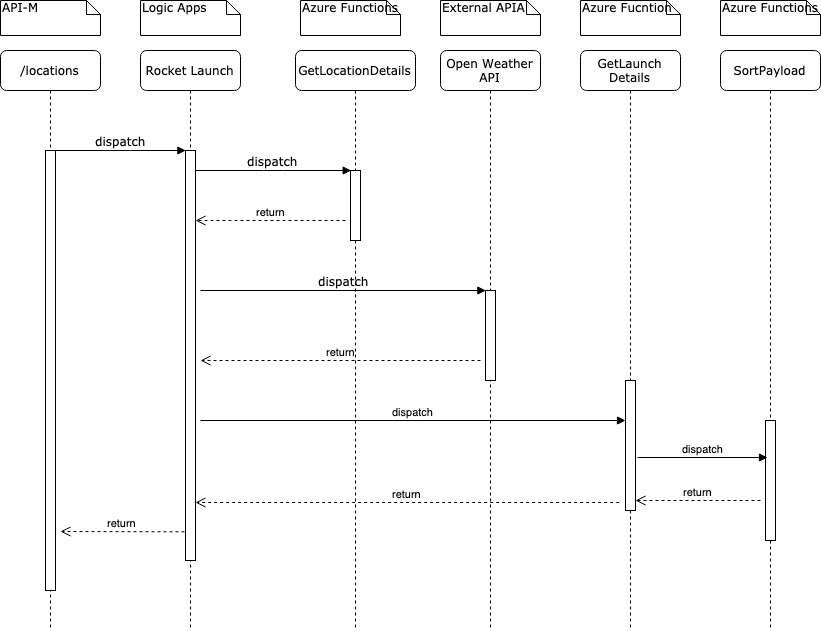
The following diagram shows the individual APIs that each application is going to expose via logic app flows, their URL and the APIs that the flows are going to consume:

|  |  |  |  |
| --- | --- | --- | --- |
| API Code | Method Endpoint | Flow | Data |
| API 1.1 | GET /locations | API 2.1 | QueryParameters = Location |
| API 1.2 | GET /locations | API 2.2 | No PArameters |



7.2.3. Application Interaction View

The following diagram shows a generic call and the flow that the different systems follow to get the data back to the API.



**7.3. Technology Architecture**

The following section describes the physical aspects of the solution, in particular, the engines that are going to execute the applications, where are they located, the networking design, ports, protocols, etc.

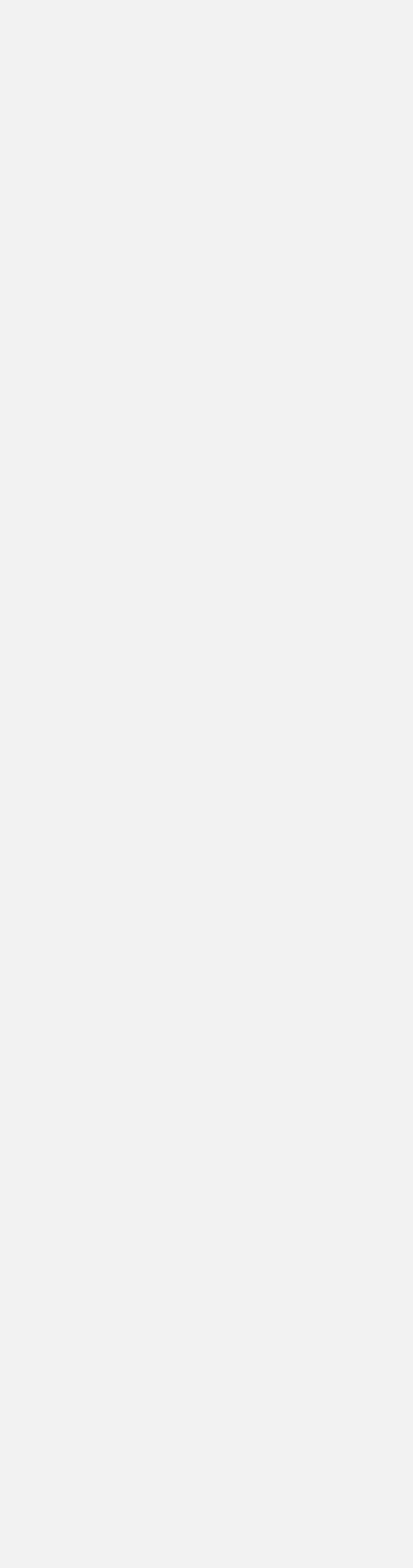
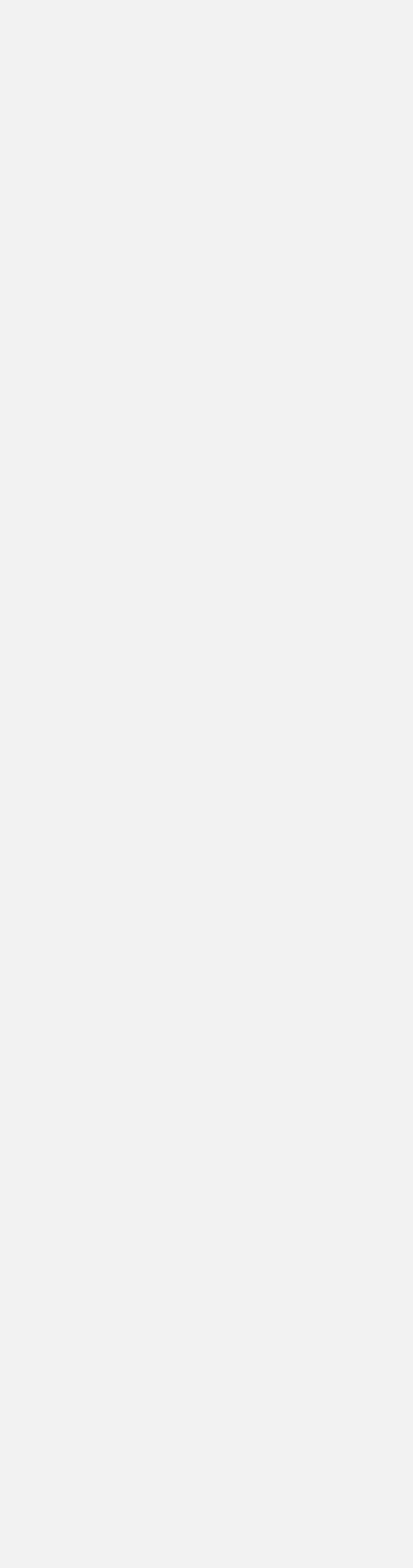
7.4. Network / Deployment View -refer to the HLD

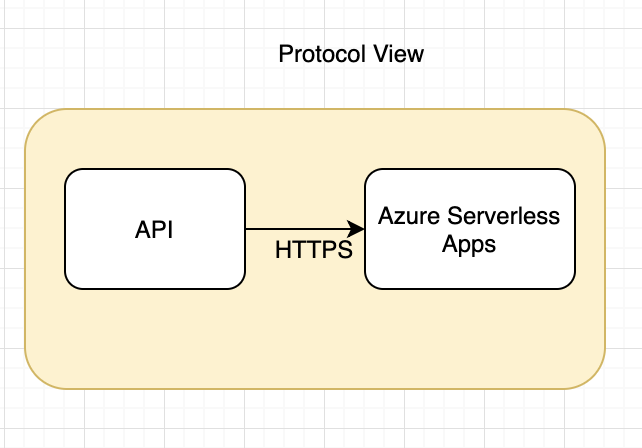


Azure subscription on Free Trial

**High Availability and Scalability:**

- Azure Serverless Auto Scales and 99.99% Available

7.4.1. Transports and Protocols View



As can be seen in the diagram below, we will be using the following protocols in the solution:

* HTTP and HTTPS if needed

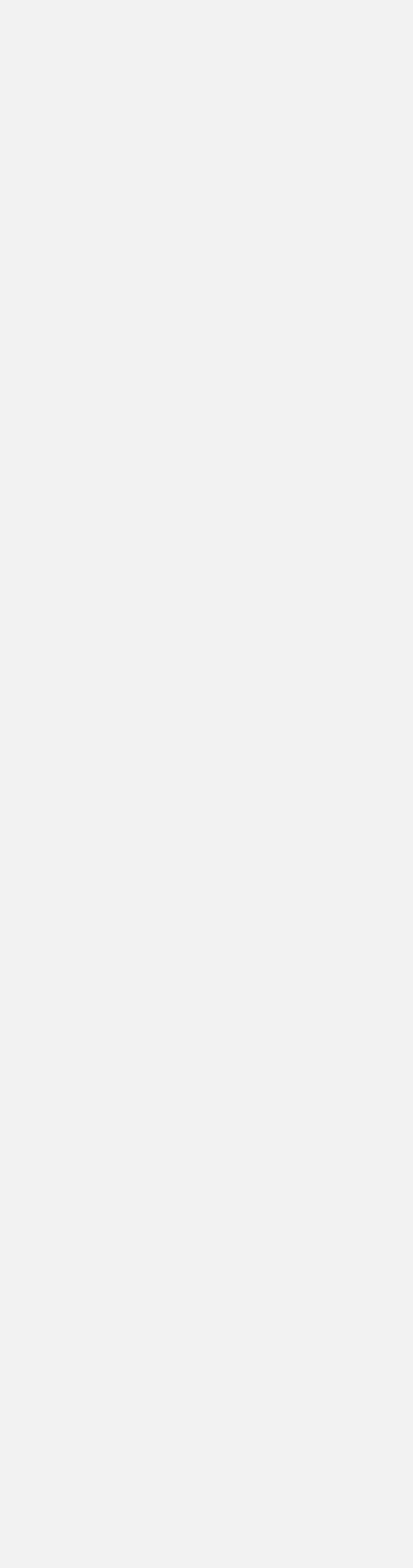
7.5. Security Architecture

The following section describes and addresses the concerns related with how secure is the solution. It covers the 4 main basic security requirements (Authentication, Authorization, Confidentiality and Integrity) and the solution to implement them.

Every azure application exposing an API will be secured using Azure API Management, which enables to manage in real-time the security policies applied to the applications:

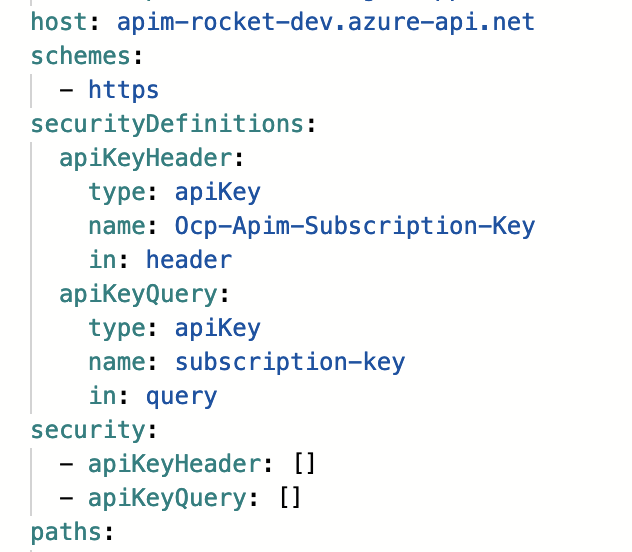
7.5.1. Authentication and Authorization

Every API call to any application settled in any layer will be authenticated. The method selected to proceed with the authentication for Rocket Launch API is the usage of subscription keys for secured connections. Security Policies of the API can be manage using the Azure API -M



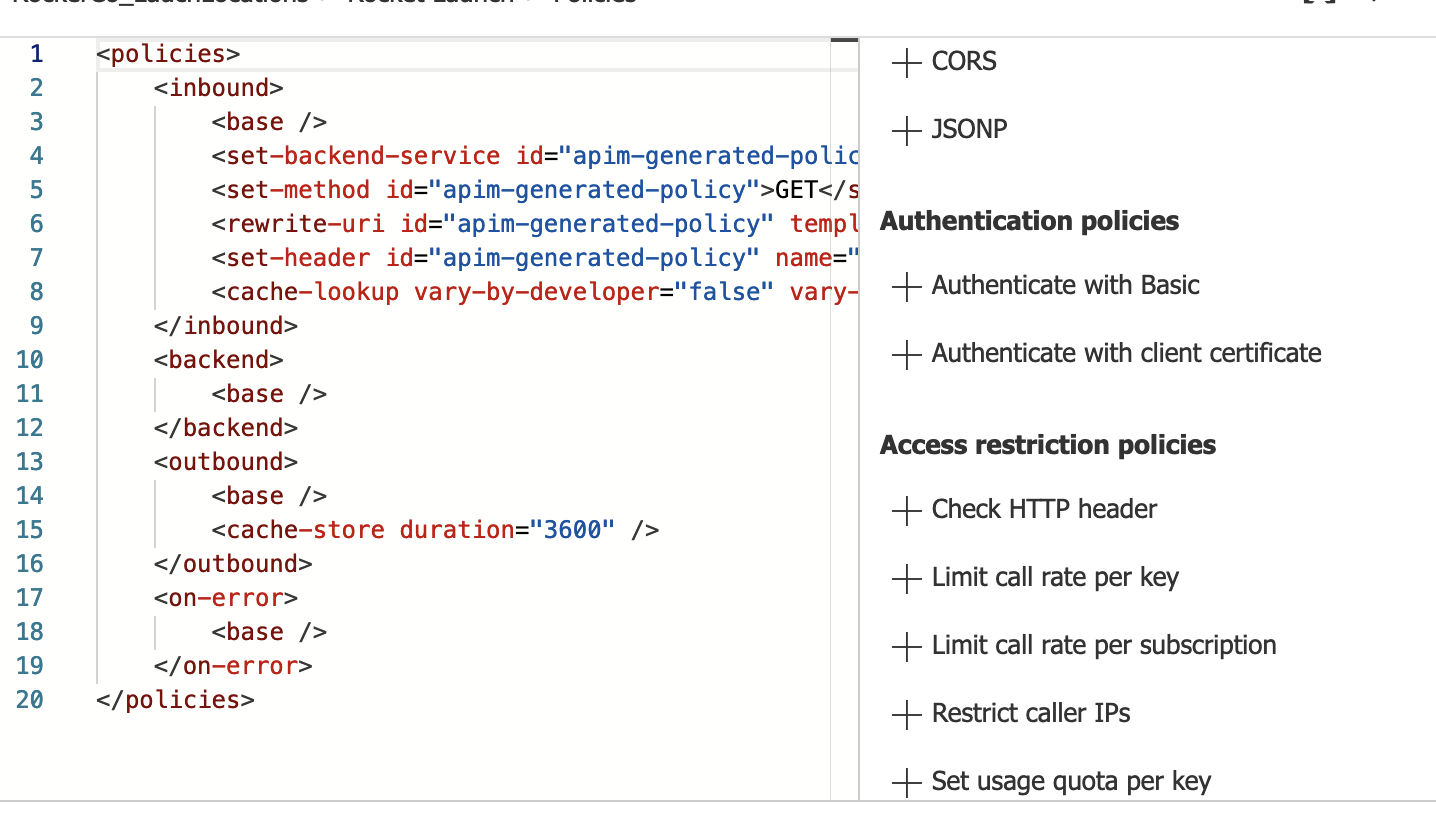
7.5.2. Authentication and Authorization policy in API Manager

The policy configured in API Manager to enforce authentication and authorization will be “Client Subscription



7.5.3. Other Policies that can be applied

There are other policies available in the API Manager to apply to the applications if there are other security related or SLA related requirements associated: Note that some of these policies require an external identity manager that, for the pilot, won’t be available but it will be for the strategic roadmap.



7.6. Operations and Support

7.6.1. Alerts

We will use the out-of-the-box alerting framework that comes with any Azure application managed in Azure Services

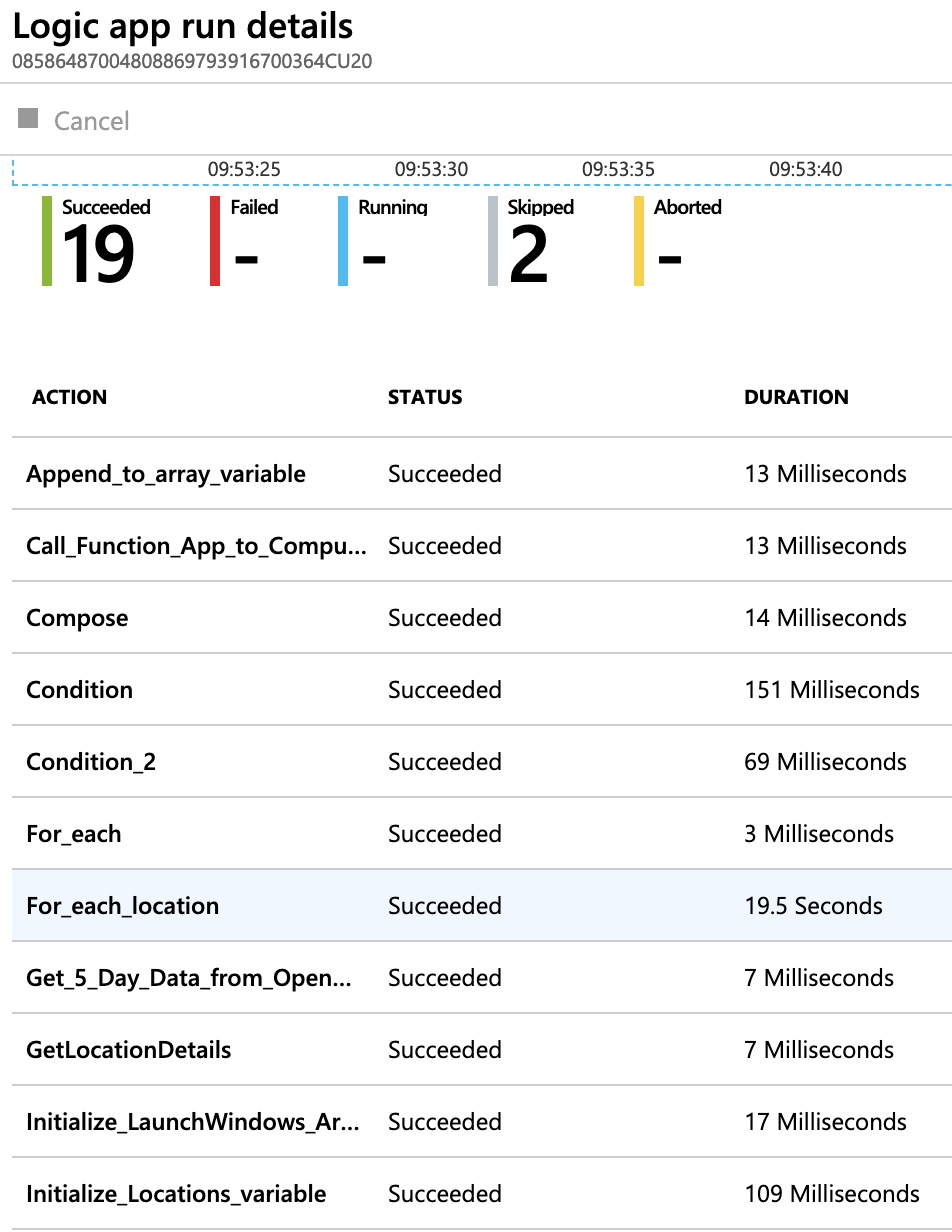
Azure alerting allows the operations team to define their own application-related alerts, or system-related alerts following an easy configuration screen.

Here you can see an example of that configuration screen, to reflect that multiple alerts can be defined and configured

7.6.2. Logs

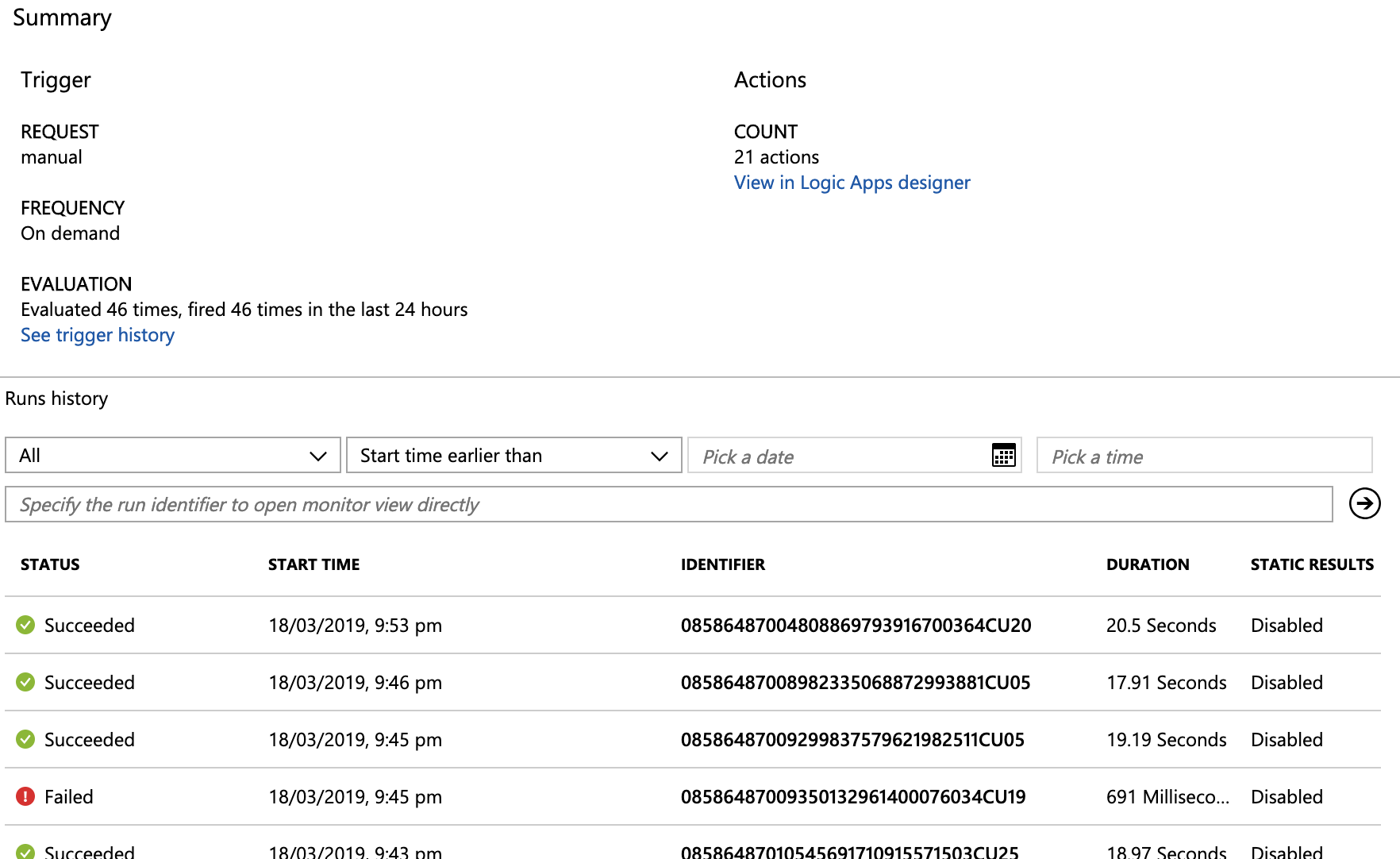
Any application will create a log with meaningful messages to be used in the event of any particular requirement (troubleshooting, normal supervision of the application, etc..). The log entries are part of the Logic flow and they are OOTB.

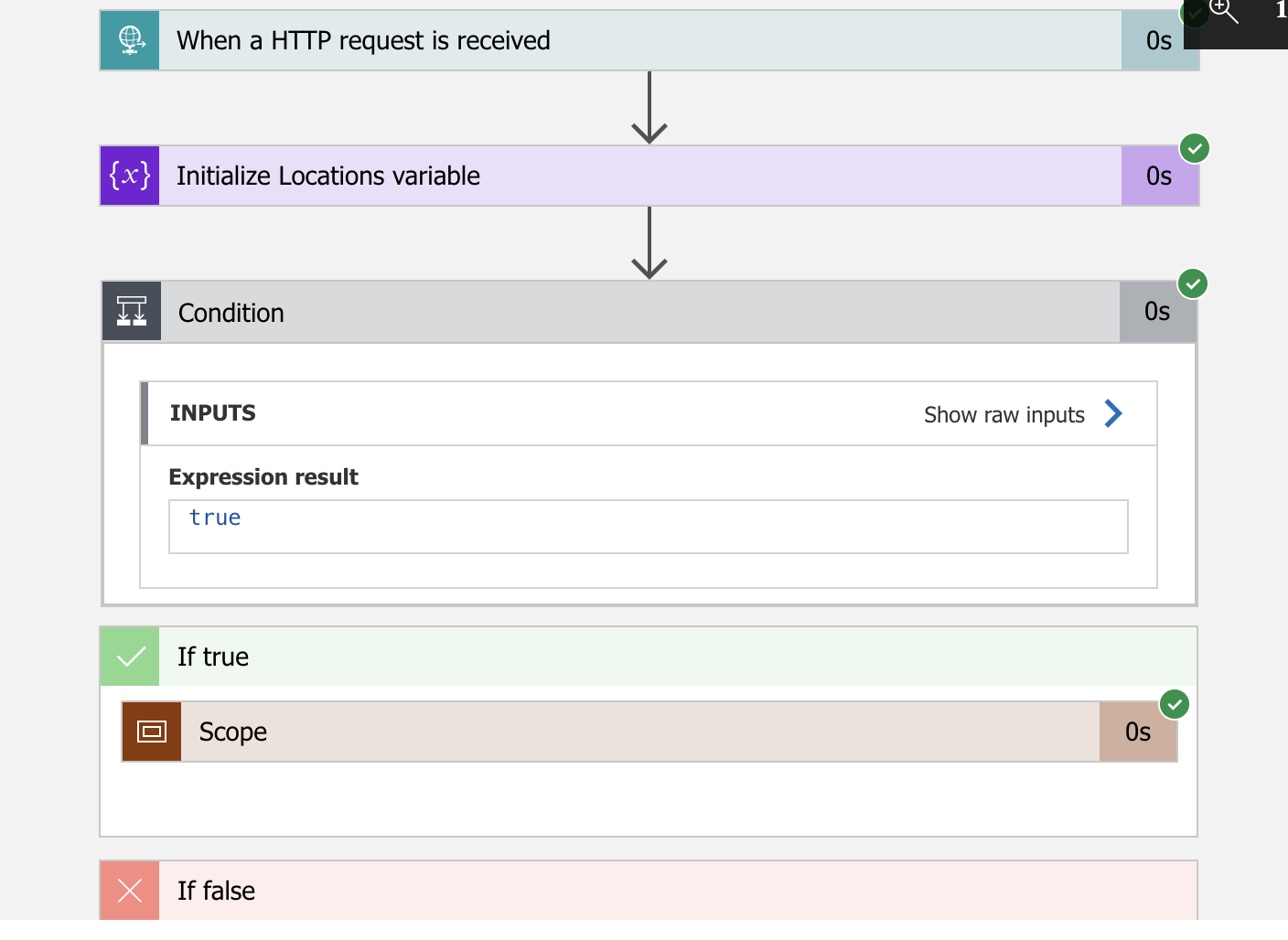
Application logs can be seen using Run History Overview under the Logic Apps management of the application

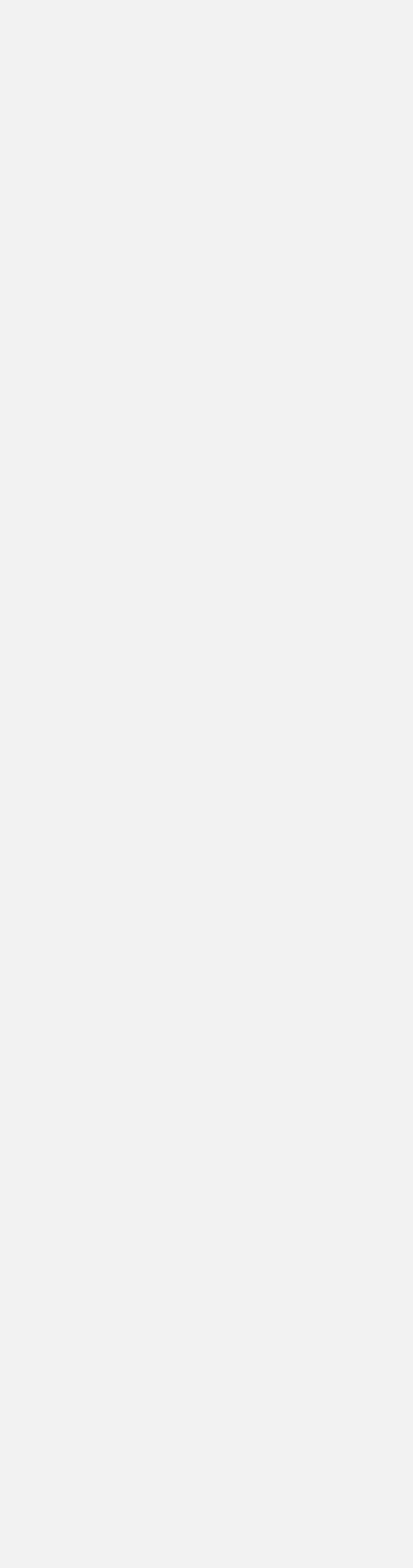


7.6.3. Monitoring and Dashboards

Azure applications will provide with extensive data to the operations team in order to monitor their performance and identify any possible problem.





7.6.4. API Analytics

Independently of the Application Monitoring provided by API Management , the APIs also can be monitored from the API perspective. This information is gathered by the API Manager and it has the ability to provide custom reports and export these data charts.



API Endpoints

https://apim-rocket-dev.azure-api.net/locations?location={location}