Integration Solution Design

|  |  |
| --- | --- |
| **Prepared by:** | Nitu Saksena |
| **Integration Architect:** | Nitu Saksena |
| **Project Manager** | Nitu Saksena |
| **Version:** | 1.1 |
| **Date:** | 25- Mar – 2020 |

TABLE OF CONTENTS

[1. Document Control](#_Toc36138433)

[1.1. Document History](#_Toc36138434)

[1.2. Document Milestones](#_Toc36138435)

[1.3. Reviewers](#_Toc36138436)

[1.4. Approvers](#_Toc36138437)

[1.5. Distribution List](#_Toc36138438)

[1.6. Related Documents](#_Toc36138439)

[4. Introduction](#_Toc36138440)

[4.1. Acronyms and Glossary](#_Toc36138441)

[5. Solution Architecture](#_Toc36138442)

[6. Functional Requirements](#_Toc36138443)

[6.1. Context.](#_Toc36138444)

[6.2. Actors](#_Toc36138445)

[6.3. Approaches Taken](#_Toc36138446)

[6.4. Challenges](#_Toc36138447)

[6.4.1 Setup Challenges](#_Toc36138448)

[6.4.2 Webservices challenge](#_Toc36138449)

[7. Design Decisions](#_Toc36138450)

[9. Solution Design](#_Toc36138451)

[9.2.3. Use Case Traceability Matrix](#_Toc36138452)

[9.5. Data Architecture](#_Toc36138453)

[9.6. Network Architecture](#_Toc36138454)

[9.6.1. Network Configuration](#_Toc36138455)

[9.6.4. Environment Mapping](#_Toc36138456)

[9.7. Security Architecture](#_Toc36138457)

[9.7.1. Authentication and Authorization](#_Toc36138458)

[9.8. Operations and Support](#_Toc36138459)

[9.8.1. Alerts](#_Toc36138460)

[9.8.2. Logs](#_Toc36138461)

[9.8.3. Monitoring and Dashboards](#_Toc36138462)

[9.8.4. API Analytics](#_Toc36138463)

[10. Appendix](#_Toc36138464)

[10.1 Release note](#_Toc36138465)

# 1. Document Control

## 1.1. Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Version Date** | **Updated by** | **Description of Changes** |
| 1.0 | 25/03/2020 | Nitu Saksena | Initial Version |
| 1.1 | 26/03/2020 | Nitu Saksena | EDR updated |
|  |  |  |  |
|  |  |  |  |

## 1.2. Document Milestones

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Doc Version** | **Date** | **Role/Person** | **Milestone Type** | **Information** |
| 1.0 | 25/03/2020 | Rebecca McDonald | Distributed for assessment |  |
| 1.0 | 25/03/2020 | Danica Ahlhaus | Distributed for assessment |  |

## 1.3. Reviewers

The following people have reviewed the document and signed off on the document.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name:** | **Title:** | **Org:** | **Role on Project:** | **Email Address:** | **Phone** | **Required / Optional:** | **Date Reviewed :** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

## 1.4. Approvers

The following people have approved and authorised the document

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name:** | **Title:** | **Org:** | **Role on Project:** | **Email Address:** | **Required / Optional:** | **Date Approved:** |
|  |  |  |  |  |  |  |

## 1.5. Distribution List

In addition to the Reviewer and approver lists the document is also distributed to the following people i.e. Inform in any RACI matrix

|  |  |
| --- | --- |
| **Name** | **Role** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## 1.6. Related Documents

|  |  |
| --- | --- |
| **Related Document** | **Link** |
| Repositories | * <https://github.com/nitusaksena/sys-weather-app.git> * <https://github.com/nitusaksena/test-docker> |
| Release Notes | Refer to section 10.1 |
|  |  |

# 4. Introduction

The purpose of this document is to track integration design considerations and high level implementation details of the weather app.

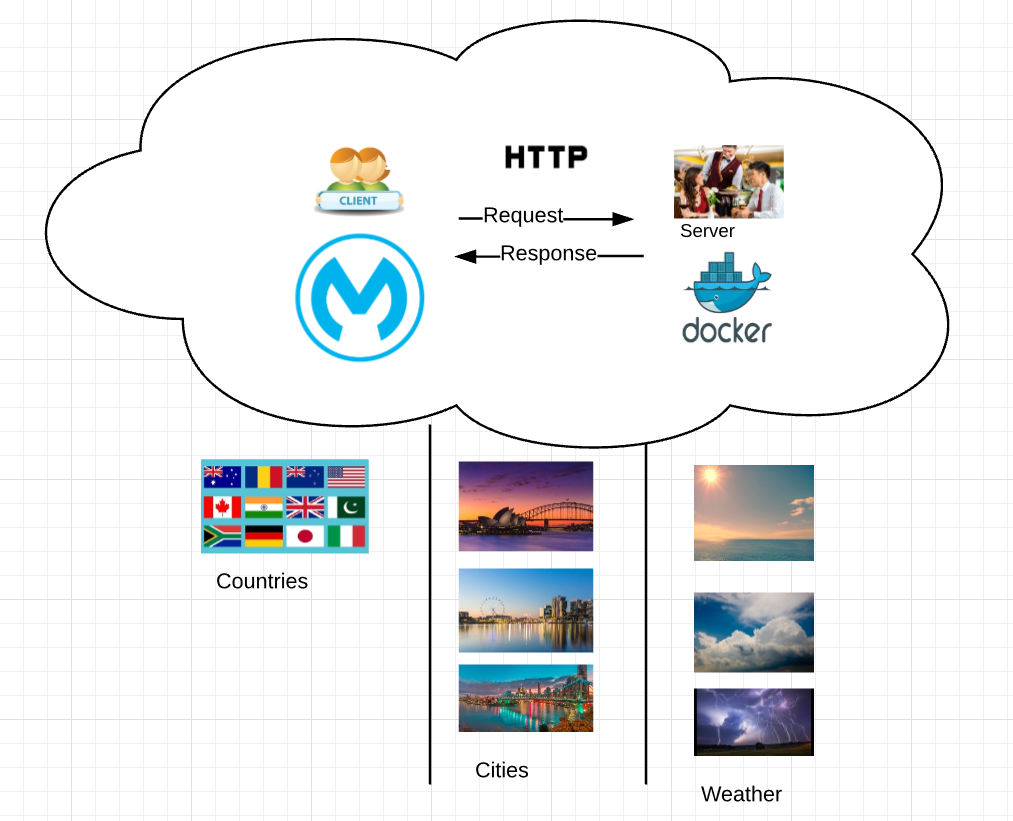
## 4.1. Acronyms and Glossary

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 |
| SOAP | Simple Object access protocol |
|  |  |
|  |  |
|  |  |
|  |  |

# 5. Solution Architecture

The high level integration strategy snapshot is captured in the self explanatory diagram below:



# 6. Functional Requirements

## 6.1. Context.

This code has been developed to complete technical challenge at Deloitte.

## 6.2. Actors

The actors of this system are the end users that are querying about the weather based on country and city/cities.

## 6.3. Approaches Taken

Mapped SOAP webservices with corresponding REST APIs. Used RAML for defining the REST specifications.

1. Rest API : /countries

Used XSLT transformation to select all distinct Countries from the webservice response.

1. Rest API : /countries/{country}/cities
   1. Used Dataweave transformation to select Cities from the webservice response.
   2. Used filter to select cities for a particular country.
   3. Validation added for path param: country
2. Rest API : /countries/{country}/cities/{city}/weather
   1. Used Dataweave transformation to select the webservice weather response to JSON.
   2. Used filter to select weather for a particular city.
   3. Validation added for path param: country, city

## 6.4. Challenges

### 6.4.1 Setup Challenges

* Need admin access for laptop.
* Anypoint studio version 4.xx is not compatible with DataWeave v1
* Could not enable Maven support as it requires Mulesoft Maven repo access.

### 6.4.2 Webservices challenge

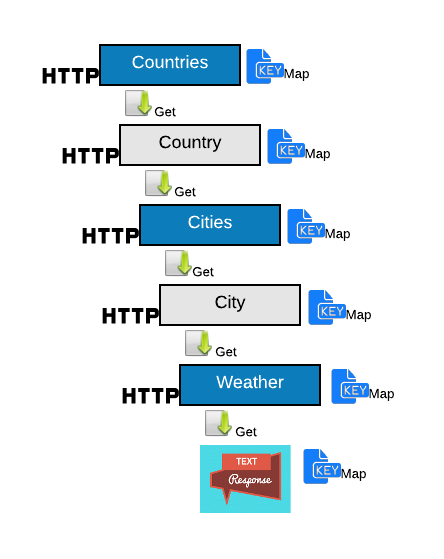
* The SOAP webservice at URL http://www.webservicex.com/globalweather.asmx?WSDL was unavailable. Needed to used Docker image for the service.
* Webservice provided with the docker image embedded the content in two CDATA elements (<![CDATA[ <![CDATA[ ]] ]]) which was wrong.
* Corrected the docker image. New docker sample is available in repo.
* Changed the npm package manager for the JavaScript runtime environment Node.js to openjdk.

# 7. Design Decisions

* Some endpoints like /countries/{country}, /countries/{country}/cities/{city} are left out as there was nothing unique to demonstrate there.
* Security aspects are not included or demonstrated as it was not scoped.
* MIME type application/json is used to return the output.
* Comprehensive use of HTTP status codes and headers is not demonstrated assumption would be to use the standard HTTP code where ever applicable.
* Unit tests are added only for happy path scenarios.

# 9. Solution Design

The section below describes high level design of the weather app.

The flow of the APIs is shown in the diagram.

MuleSoft's API driven approach infers there are three fundamental types of APIs that should be created.

* System API for system of record integration
* Process APIs for ESB orchestration
* Experience APIs for each consuming client

### 9.2.3. Use Case Traceability Matrix

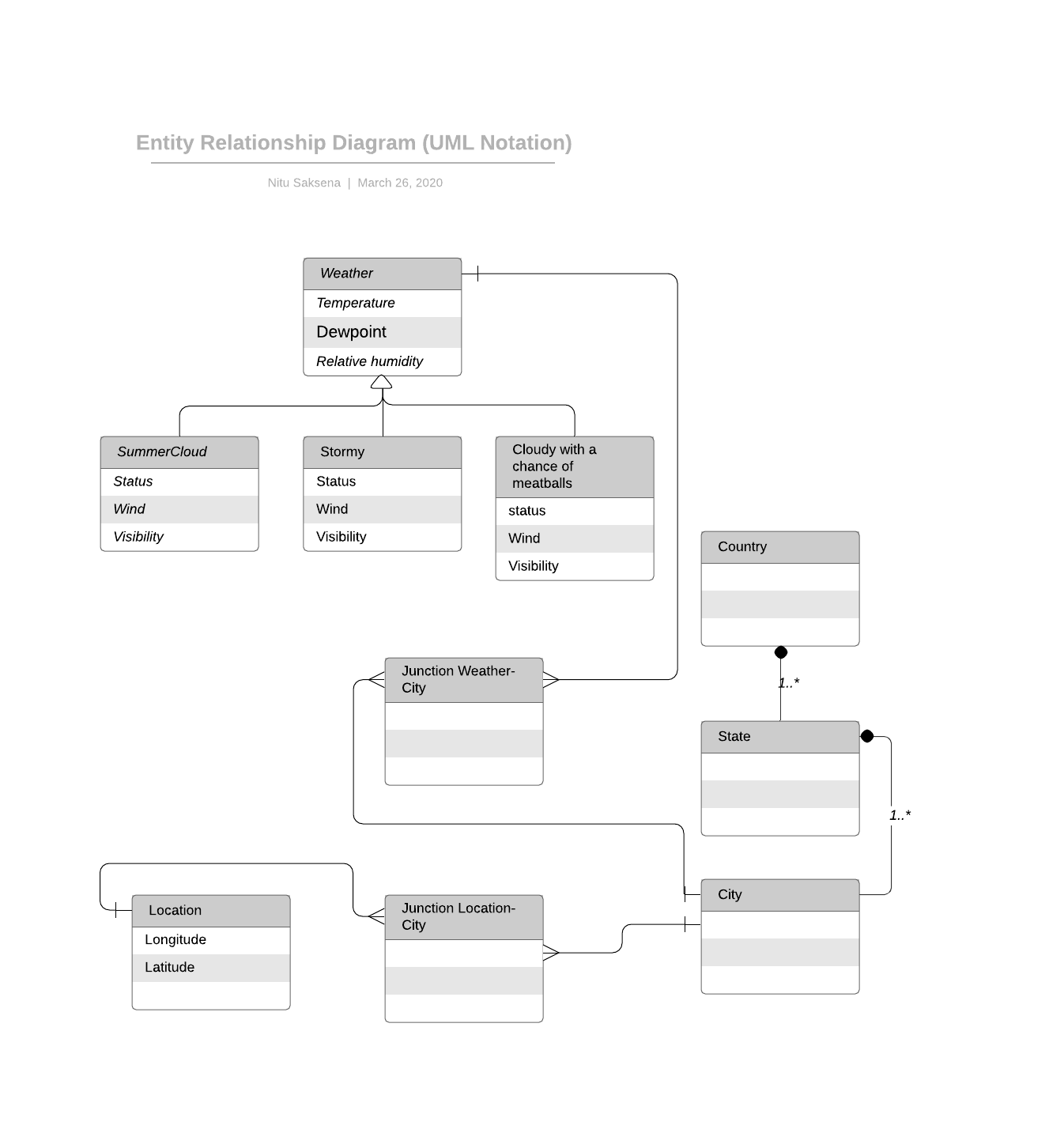
The following section describes the physical aspects of the solution. In particular, the engines (mule workers) that are going to execute the applications, where are they located, API type, ports etc.

Mule workers are deployed on cloudhub at port 8091 for private APIs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **API type** | **Private (8091)** | **Accessed from CLoud** | **Deployment name** | **new API Name** | **SIT API** |
| System API | Yes |  | sys-weather-app<env name> | <deployment name>/countries | http://mule-worker-internal- sys-weather-app-api-deloitte-sit.au-s1.cloudhub.io:8091/api/countries |
|
| System API | yes |  | sys-weather-app <env name> | <deployment name/countries/{country}/cities | [http://mule-worker-internal-sys-weather-app-api-deloitte-sit.au-s1.cloudhub.io:8091/api//countries/{country}/cities](http://mule-worker-internal-sys-weather-app-api-deloitte-sit.au-s1.cloudhub.io:8091/api//countries/%7bcountry%7d/cities) |
| System API | yes |  | sys-weather-app <env name> | <deployment name>/countries/{country}/cities/{city}/weather | http://mule-worker-internal- sys-weather-app-api-deloitte-sit.au-s1.cloudhub.io:8091/api/user-management//countries/{country}/cities/{city}/weather |

## 9.5. ERD- Data Architecture

Please see below entities in scope of this project. The ERD is intuitive and self-explanatory.

We require junction objects to cater to many-to-many relation between Weather & City and Location & City. There is a composition relation between country, state and city as indicated and generalization/specialization relation between Weather and weather types. 

## 9.6. Network Architecture

The section below describes the possible configurations that might be required to resolve any firewall or access issues on the internal private network.

### 9.6.1. Network Configuration

**VPN**

Use Anypoint VPN to create a secure connection between your MuleSoft Virtual Private Cloud (VPC) and your on-premises network. You can create multiple site-to-site VPNs if required.

**VPC**

The Anypoint Virtual Private Cloud (VPC) offering allows you to create a virtual, private, and isolated network segment in the cloud to host your CloudHub workers.

Connecting to your Anypoint VPC extends your corporate network and allows CloudHub workers to access resources behind your corporate firewall. You can connect on-premises data centers through a secured VPN tunnel, or a private AWS VPC through VPC peering, or by using AWS Direct Connect.

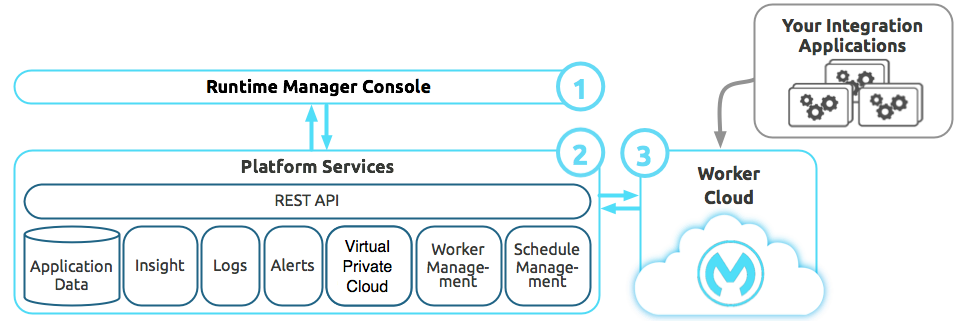
### 9.6.4. Environment Mapping

This table is meant to capture the systems in the columns and in rows the environment mapping from one system to other fro dev, test, SIT, UAT and Produdtion.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CLOUDHUB ENV** | **Environment** | **System 1** | **System2** | **System 3** |
| QA | SIT | To be provided by Customer | To be provided by Customer | To be provided by Customer |
| UAT | UAT | To be provided by Customer | To be provided by Customer | To be provided by Customer |
| Production | Production | To be provided by Customer | To be provided by Customer | To be provided by Customer |

## 9.7. Security Architecture

To understand CloudHub’s approach to security and availability, it’s important to understand the architecture behind CloudHub. It includes two major components—​Anypoint platform services, and the worker cloud. These two components and the Runtime Manager console through which you access them work together to run your integration applications.



|  |  |  |
| --- | --- | --- |
|  | [Integration Applications](https://docs.mulesoft.com/runtime-manager/cloudhub-architecture#integration-applications) | These are applications that you create and deploy to CloudHub to perform integration logic for your business. |
|  | [Runtime Management Console](https://docs.mulesoft.com/runtime-manager/cloudhub-architecture#runtime-management-console) | The Runtime Manager console is the face of CloudHub, allowing you to deploy and monitor integrations, and configure your account, among other things. |
|  | [Platform Services](https://docs.mulesoft.com/runtime-manager/cloudhub-architecture#platform-services) | This set of shared CloudHub platform services and APIs includes CloudHub Insight, alerting, logging, account management, virtual private cloud/secure data gateway, load balancing, and others. |
|  | [Global Worker Clouds](https://docs.mulesoft.com/runtime-manager/cloudhub-architecture#global-worker-clouds) | This is an elastic cloud of Mule instances that run integration applications. |

We have ensured the security of the APIs by deploying it on Mule private port unless required to be public.

When you configure your HTTP Listeners for your application on ${http.private.port} instead of using ${http.port} or ${https.port} for the port attribute, we use ports 8091 or 8092.

Then from inside your secure network, you can call your application with a DNS/URL similar to:

* [http://mule-worker-internal-{](about:blank)appname}.[cloudhub.io](http://cloudhub.io/):8091/path etc

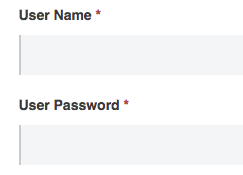
Ports 8091 and 8092 are special in cloudhub and only accept incoming requests on the private network. You cannot call them from public internet. You must use the 'internal' DNS addressing as described above.

This procedure applies at the per application level. This means you are able to define one application to have public listeners (i.e. 8081 or 8082) and another application in the same environment to be entirely private only (i.e. use only 8091 or 8092).

### 9.7.1. Authentication and Authorization

For this challenge we have not used authentication and authentication however basic -simple authentication policy can be used to protect the API by requiring a username and password when calling apps make a request. This policy is available only to an API you are managing in Mule 4 or later.

You can follow the general instructions for applying a policy, and enter the username and password to configure the credentials.



You can also configure resource-level policies and methods.

After applying this policy to the API, a request to that API must contain the following header:

Authorization: Basic <username:password>

where username:password is a base64-encoded string. In Mac OS X or Linux, for example:

echo '<Client Id>:<Client Secret>' | base64

## 9.8. Operations and Support

### 9.8.1. Alerts

Alerts were not scoped in the solution however it could be done. An API alert is an alarm that flags one of the following problems:

* The API request violates a policy.
* Requests received by the API exceed a given number within a period of time.
* The API returns a specified HTTP error code.
* The API response time exceeds a certain amount.

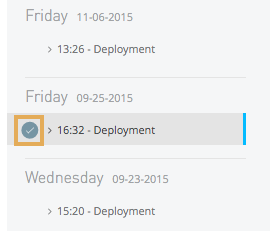
Runtime Manager alerts differ in nature from API alerts. Runtime Manager flags CPU usage, secure data gateway up/down, and deployment success/failure, to name a few. API alerts, on the other hand, relate to an API request or response.

You can add one of the following types of alerts to an API:

* Policy violation  
  An infraction of one or more policies that govern the API occurs.
* Request Count  
  Users request access to the API more times than allowed in a specified period of time.
* Response Code  
  The API returns one of these HTTP codes upon receiving a request: 400, 401, 403, 404, 408, 500, 502, or 503
* Response Time  
  The response time of the API exceeds a specified period.

### 9.8.2. Logs

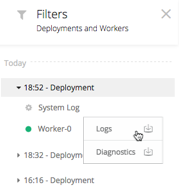
View logs for applications deployed to CloudHub.

1. In Runtime Manager, click **Applications** in the left menu, and then click the application that is deployed to CloudHub to open the management panel on the right.
2. Click **Logs**.  
   The application logs are displayed.
3. The **Deployments** section on the right of the logging screen lets you selectively view the output by day, deployment, system log, and workers. You can unapply a filter by unticking the checkbox next to a date.  
   

**Download Logs**

Each worker in each CloudHub deployment has separate logs you can download.

1. In the **Deployment** filters section on the right, find the date and worker you want to view.
2. Click the download icon next to the worker, and select **Logs**.



### 9.8.3. Monitoring and Dashboards

Configuring dashboard is not within the scope of this project. Customer could use the out of box dashboards. Here is some information from Mulesoft documentation about using the dashboard.To view dashboards, you must have either the Runtime Manager Read Applications permission or the Runtime Manager Read Servers permission for the resource.

Anypoint Runtime Manager provides these dashboards:

* [Application monitoring dashboards](https://docs.mulesoft.com/runtime-manager/monitoring-dashboards#application-monitoring-dashboards)Displays information about applications, including deployment target and status.  
  Runtime Manager provides two application dashboard interfaces: built-in and classic. The dashboard you see depends on the version, type, and deployment method of the selected application.
* [Server monitoring dashboard](https://docs.mulesoft.com/runtime-manager/monitoring-dashboards#server-monitoring-dashboard)Provides high-level information about the servers, including heap memory and CPU percentage, as well as detailed memory usage metrics.
* [Server group or cluster monitoring dashboards](https://docs.mulesoft.com/runtime-manager/monitoring-dashboards#server-group-or-cluster-monitoring-dashboard)Displays aggregated metrics for selected servers in a group or cluster.

Using the information gathered from these dashboards, you can:

* Set up alerts for applications and servers to notify you automatically when any of these metrics reaches a critical value for a sustained period.
* Use Insight to help analyze the root cause of failures, isolate performance bottlenecks, and test for compliance with company procedures.

### 9.8.4. API Analytics

API Analytics was not within the scope of this engagement. However API Analytics can provide insight into how your APIs are being used and how they are performing. From API Manager, you can access the Analytics dashboard, create a custom dashboard, create and manage charts, and create reports. From API Manager, you can get following types of analytics:

* API viewing analytics
* API events analytics
* Charted metrics in API Manager

You can view data from a policy perspective in the Analytics dashboard, Analytics chart, or API Manager by including policy violations. This action tracks traffic rejected, or not rejected, by a policy.

# 10. Appendix

## 10.1 Release note

# sys-weather-app

A rest API built using Mulesoft Anypoint platform that exposes a SOAP based service as REST APIs.

## Import

Import the project as a Mulesoft Anypoint studio project in the Anypoint studio version Version: 6.5.0

## Run

Run the src\main\app\weather.xml by right clicking and selecting option "Run as" -> Mule Application

## APIs

Following APIs are implemented:

- /countries ex. http://localhost:8081/weathermonitor/api/v1/countries/

- /countries/{country}/cities ex. http://localhost:8081/weathermonitor/api/v1/countries/Australia/cities

- /countries/{country}/cities/{city}/weather ex. http://localhost:8081/weathermonitor/api/v1/countries/Australia/cities/Melbourne/weather