IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

NCAIT 2017

JSS Academy of Technical Education

Vol. 4, Special Issue 8, May 2017

# Cloud Integration of Applications and Services

**Manoj N Bisarahalli[1], Shreyas M[2], Pradeep Nagendra Urala[3]**

Dept. of ISE, SJBIT Bengaluru, India[1,2,3]

**Abstract:** The paper deals with the development of a cloud integration service that involves building multiple integration components. An integration component is a particular, metamorphic piece of functionality that consistently represents a way to communicate with one organization and/or API. Sharing information and providing visibility into both frontend and backend systems using cloud integration increases productivity and simplifies business process. With third-party cloud service integration, organizations can focus more on driving new business and less on the hassles of trying to make data available. Moreover, synchronizing data to provide updated information whenever a change is made in either of the cloud services can be automated, with valuable data from cloud service systems available to the right people at the right time, all through a single interface. The system provides generic interface between components.

**Keywords:** cloud integration; iPaaS; OAuth2.0; RESTful APIs; Accounting; e-Commerce.

## I. INTRODUCTION

The rapid enhancement and wide adoption of cloud services has allowed the enterprises to remodel their businesses and increased innovation processes popularized by the new paradigm. By depending on the component reuse and supplying the advantages of economy of margin to limited enterprises, cloud based approaches lead to the creation of new organizations and service models making the dependencies and relations between various organizations more robust than ever. An important ramification of the current adoption of cloud based services among organizations is the growing scale of outsourcing of business and technological functions to third parties.

Advanced approaches such as cloud integration platforms (iPaaS) promote this trend further. Building and deployment of integrations in the cloud is done using iPaaS which acts as a platform. In this case users' resources shared over various cloud systems are retrieved, distributed and managed thoroughly in the cloud, at third party premises effectively transferring the execution of entire business processes to the cloud. An integration platform catering the specific needs to different software solutions as well as allowing the integration of solutions helps in the progression of business processes. In the long run, a platform like this is vital to get the explosion of software solutions in control. This ensures a constant management of dataflow, databases, reporting and monitoring tools.

As cloud and cloud related services draw progressive traction in the enterprise technology realm, proposing a software solution or the platform not shifting to the web is not feasible. Scalability, reliability and ease of implementation are provided to all businesses by web based instances whereas on-site hosting is unable to do so. Although it is new to the world of technology, web based integration platforms are an impending addition to the warehouse of cloud software tools. This is chiefly relevant as more vendors are transitioning to SaaS and they require a cost effective and simple solution to integrate the systems. Integration hosted on the web will strengthen the assets of integration as key information and processes are synced. Once the syncing is complete the information can be accessed from anywhere. The acceleration of cloud-based enterprise technology hints at web-based integration establishing the landscape of interconnected enterprise in the coming days.

In this work we concentrate on the prospect of resource owners which sanctions automated, structured discovery of resources hosted at different cloud premises. We also consider the prospect of integration clients that retrieves and manages distributed resources in support of resource owners providing the model for discovery, integration and other specific parameters of sharing requests.

## II. BACKGROUND AND RELATED WORK

### *A.* Integration Platoforms

Integration platform as a service allows the implementation of application, process, service, data integration and other suitable administration in the cloud. Coexistence and union of iPaaS facilities along with on-premises will be conventional in large organizations. Since the emergence of iPaaS, many analysts have tried to establish and define this service model. Pezzini et al. has contributed in view of this heading, characterizing iPaaS as a collection of cloud management advocating progress, execution and management of fields relating any combination of on-premises and cloud based approaches, applications and information within single or multiple organizations[1]. Currently Elsatic.io is one of the leading micro services-based hybrid integration platform as a service (iPaaS), built to link different cloud based and on-premise data easier and quicker. The plan is a positive fit for a bimodal IT corporation.[2] Principally it gives information on popular implementation of iPaaS and about the elastic.io cloud based integration platform as a service in specific.

*B.* Protocols and Architectures for Web Authorization

In the conventional client server validation model, the client demands an access-restricted asset (ensured asset) on the server by validating with the server, utilizing the asset proprietor's accreditations. Keeping in mind the end goal to give outsider applications access to confined assets, the asset proprietor shares its certifications with the outsider. Hardt.D et al. contributed in this regard, one of the major approaches to protect interfaces is OAuth 2.0, a web authorization protocol that enables clients to access protected resources on behalf of a resource owner[3]. The OAuth 2.0 authorization framework enables a mediator application to gain restricted access to an HTTP service, either on behalf of a resource proprietor or by allowing the mediator application to gain access on its own behalf.
These HTTP services can be accessed by making the corresponding API calls. API is a substitute to a graphical user interface. Software requires an interface that facilitates it for the easy consumption of the data. Interfaces are typically built according to RESTful architectural style which currently represents one of the major approaches for web APIs. This can be observed by analyzing the directory maintained at Programmable Web where the vastmajority of registered web APIs are based on RESTful style [4].

*C.* Problem Statement

A quickly developing enterprise can instantly be intertwined with a complex application scenario. In the pursuit of gaining new clients, companies employ independent opportunity management system, with different systems for rectifying customer support problems. Inventory management, order management and fulfilment systems are directed with contrasting software and spreadsheets. Additional business software that aids the processes can be standalone applications provided by various vendors, original applications or a blend of spreadsheet workarounds. Businesses that evolve their systems over time in this style can find themselves with an underdeveloped architecture that handles limited skilful needs sub-optimally while holding the enterprise back from scaling efficiently over the long term. The main issues faced by this approach are wasted employee productivity, lack of real-time visibility, integration complexity and increased customer churn.

In this work, we contribute a model which deals with the development of a cloud integration service that involves building multiple integration components. The elastic.io integration platform as-a-service serves as a middleware platform in this model. Implementing connectors and developing customised integrations is cost-effective and automates the real-time data transfer and synchronisation between the systems.

## III. SYSTEM ARCHITECTURE

A system architecture is a conceptual model that defines the structure, behaviourand more views of a system. The architecture diagram in Figure (1)represents the limitless connectivity provided by the system.

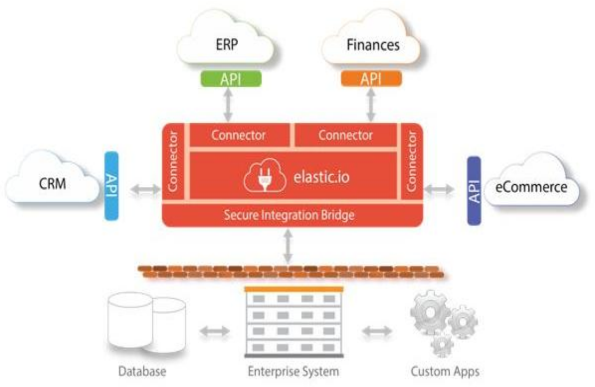


Figure 1: System Architecture

A system architecture can comprise system components that will work together to implement the overall system.

The various components which compose the architecture of this system are as follows:

- Business Software:

  Business applications are used to increase productivity and to perform business functions such as managing finances or accounting, eCommerce and customer relationship management (CRM). Most of the business applications today are SaaS. SaaS removes the need for organizations to install and run applications on their own computers or in their own data centres instead makes it available over the internet.
- APIs:

  A cloud API constitutes a portal or an interface that provides unambiguous and in-direct cloud framework and software services to users. A cloud API is the core component behind any public cloud solution and is based primarily on the REST and SOAP frameworks.
- Connectors:

  In this integration scenario connectors are built using triggers and actions where they either try to post or retrieve data from one business software to another. It can either be a trigger or an action depending on the direction of the data flow.
- Elastic.io:

  Elastic.io is a software vendor specializing in Data Integration. This platform provides micro services-based integration software for connecting applications, databases, and APIs, cloud-to-cloud or cloud-to-ground. It is a part of the hybrid integration platforms category, as it can be deployed in the cloud as well as on-premise.

IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

NCAIT 2017

JSS Academy of Technical Education

Vol. 4, Special Issue 8, May 2017

## IV. METHODOLOGY

We are building the connectors between the business applications to be integrated to provide real time data flow between the same and implementing OAuth 2.0 Framework which is being used for authentication and authorisation by the applications.

### *A.* Building the Connectors

Connectors can be built as either triggers or actions depending on the direction of data flow. A trigger is a module which makes use of various GET HTTP requests to obtain information from the server. An action is also a module which uses HTTP requests to exchange information with the server. A response to the GET HTTP request is the data in the form of JSON object only after successful completion of the authentication and authorization process.

A POST HTTP request needs to have a body which is a JSON object, in its request. The data obtained from a server through GET requests are converted into a standard Aplynk format which is basically a standard naming convention. Once the names are standardized other connectors can use the convention to map the data without any hassle.

### *B.* Authentication and Authorisation Process using OAuth 2.0

Both the business applications- Shopify and Exact Online make use of OAuth 2.0 Framework for authentication and authorisation as shown in Figure (3).

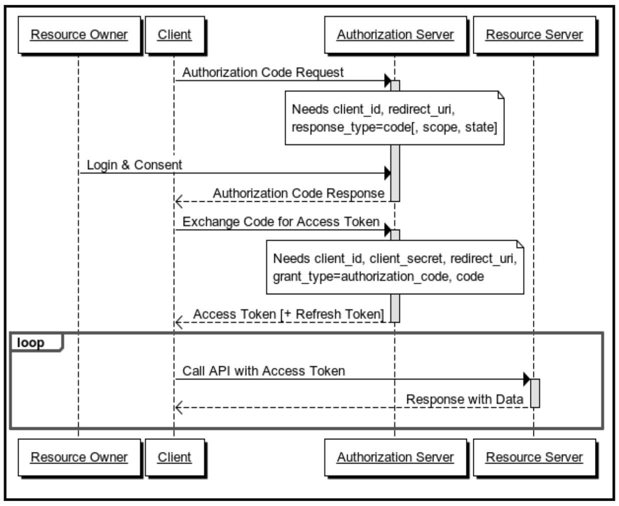

Resource Owner
Client
Authorization Server
Resource Server
Authorization Code Request
Needs client_id, redirect_uri, response_type=code[, scope, state]
Login & Consent
Authorization Code Response
Exchange Code for Access Token
Needs client_id, client_secret, redirect_uri, grant_type=authorization_code, code
Access Token [+ Refresh Token]
loop
Call API with Access Token
Response with Data
Resource Owner
Client
Authorization Server
Resource Server

Figure 2: Sequence Diagam of OAuth 2.0

The OAuth 2.0 sequence illustrated in Figure (3) describes the interaction between the four roles and includes the following steps:

- To access service resources, the application requests authorization from the user.
- Upon the validation of request by the user, an authorization grant is received by the application.
- Authorization server receives a request of access token (API) from the application after being presented with its authentication and authorization grant.
- After the verification of application's identity and validation of authorization grant an access token is issues to the application by the authorization server (API). Once this is done, authorization is complete.
- Resource is requested from the resource server (API) by the application and the access token is presented for authentication.
- The resource server(API) serves the resource to the application if the access token is valid.
- "Invalid Token Error" is resulted if a request is made after the token expires. A refresh token can be included so that in situations like these a fresh access token can be requested.

## V. IMPLEMENTATION

### *A.* Details of the language used

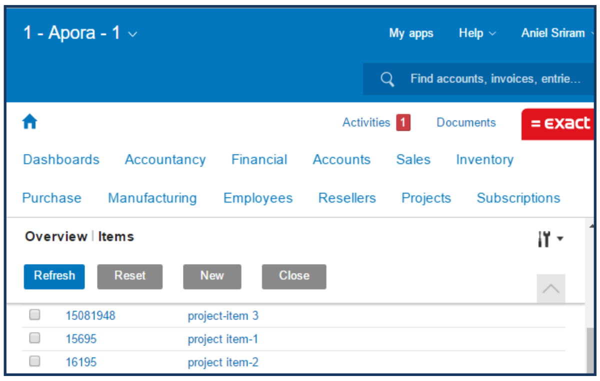
The complete implementation of the design is in java language. Java is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems Java platform.

We have majorly used the concepts of Retrofit. Retrofit turns HTTP API into a Java interface. The retrofit class generates an implementation of the interface. Each call from the created interface can make an HTTP request to the remote web server. Retrofit is a type safe REST client. Retrofit provides a powerful framework for authentication and interactions with APIs. OKHTTP is an important underlying library of Retrofit which is used for sending and receiving HTTP requests. Retrofit provides a straightforward method for downloading XML/JSON data. This data is then parsed with the help of POJO (Plain Old Java Objects).To send out network requests to an API, we need to use the Retrofit builder class and specify the base URL for the service. GSON library is used to serialize and de-serialize data.

Gradle also plays an important role It is an open source build automation structure that develops upon the concepts of Apache Ant and Apache Maven. Gradle supports the automatic download and configuration of dependencies or other libraries. It has the impression of projects and tasks and also supports the automatic download of the Java library dependencies. The current project makes use of Gradle wrapper which allows user to run build with a predefined version without local Gradle installation. Wrapper acts as a batch script on Windows where specified version of Gradle is automatically downloaded and used to run the build.

### *B.* Snapshots

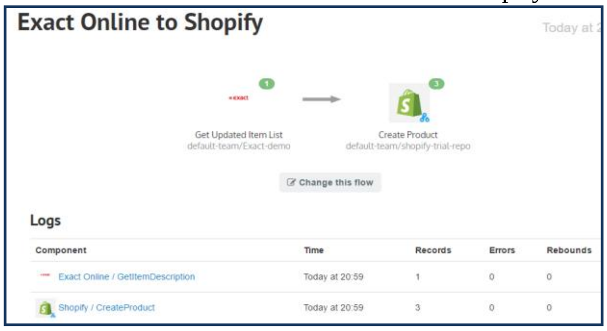
Figure 3 is a snapshot which shows us the Items which are being added into the Exact online App. of the store owner.



| | |
|---|---|
| 15081948 | project-item 3 |
| 15695 | project item-1 |
| 16195 | project item-2 |

Figure3: Snapshot for adding items on Exact

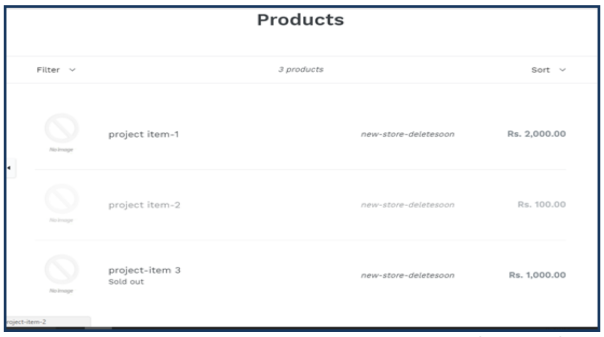
Figure 4 is a snapshot of the Aplynk portal where the data flows from one component to another. In this case the 3 records are transferred from Exact online to Shopify.



| Component | Time | Records | Errors | Rebounds |
|---|---|---|---|---|
| Exact Online / GetItemDescription | Today at 20:59 | 1 | 0 | 0 |
| Shopify / CreateProduct | Today at 20:59 | 3 | 0 | 0 |

Figure 4: Transfer of records from Exact to Shopify

Figure 5 is a snapshot where we can see the items which being displayed on the Shopify storefront.



| | | |
|---|---|---|
| project item-1 | new-store-deletesoon | Rs. 2,000.00 |
| project item-2 | new-store-deletesoon | Rs. 100.00 |
| project-item 3<br>Sold out | new-store-deletesoon | Rs. 1,000.00 |

Figure 5: Item catalogue displayed on Shopify storefront

## VI. CONCLUSION

In this paper we have presented our work on cloud integration of e-commerce and accounting software using iPaaS as a platform. This approach of cloud integration provides accurate and consistent data as Aplynk keeps Shopify store orders, products & accounting data up to date. Real time integration takes place as every selected update made on Shopify will be synced to Exact Online account in real time. The setup is easy and simple as the web based integrator is instantly more scalable and customizable than an on premise project. Integration in the cloud can take days and accommodate the needs much more cost-effectively. Integration of new systems or an increase in storage space requires little more than some work on current integration platform online, not an on-site customization visit. Clients can spend less time and money on administrative tasks. Hence, the current integration process saves time and is cost effective.

## ACKNOWLEDGMENT

We thank our guide, Mrs.Yamuna U, for her valuable guidance, continuous support and sharing the domain knowledge. We also thank Apora Technologies Private Limited, which has provided the necessary infrastructure and support.

## REFERENCES

[1] Pezzini, M. and Lheureux, B. ”Integration platform as a service: moving integration to the cloud.” Gartner, 2011.
[2] Elastic.io:” Overview of the elastic.io: Integration Platform as a Service”.2016
[3] Hardt, D. ”The OAuth 2.0 authorization framework”. 2012
[4] Programmable Web. http://www.programmableweb.com (last accessed 2016)