Hybrid Testing Model for Cloud API Testing as a Service

[1]R. Jeba Gazelle and [2]M.A. Maluk Mohamed
[1]*Anna University, Chennai, India*
[2]*Department of Computer Science, MAM College of Engineering, Tiruchirappalli, India*

Key words: Application programming interfaces, quality assurance, testing as a service, software

Corresponding Author:
R. Jeba Gazelle
Anna University, Chennai, India

Page No.: 30-34
Volume: 12, Issue 3, 2019
ISSN: 1997-5422
International Journal of Systems Signal Control and Engineering Application

Abstract: Cloud computing provide infrastructures, development and deploying platforms and provides many application services over internet. As multiple systems, devices and applications interoperate in the today's cloud world, APIs play an important role providing the required functionalities by communicating with different sub systems without a compromise in the quality of delivery. An important area on this front is the testing of Application Programming Interfaces (APIs) which are direct links between the client code and the infrastructure it runs upon. In this study, a hybrid model for testing cloud platforms and infrastructures is suggested. The model is hybrid in the essence that it provides two options: one is that it leverages the API test cases either automated or manual test scripts which can be reused for testing various APIs and hence provides an option of testing as a service. Secondly, the model includes the various test environments that would be required to run the API testing, the plugins and adapters that would be required to execute the test cases such that it leverages the concept of infrastructure as a service for API testing, This option helps the development and cloud quality assurance engineers not to spend effort on setting up the required test environments but can use them as a service.

INTRODUCTION

Cloud Computing can be applications that reside in datacenters as hardware or software in the huge data centers and could be the products or applications. Li *et al*. (2011) explained the major characteristics of cloud computing as socialization, intensification and specialization. Out of which socialization is the behavior of cloud computing which showcase cloud computing as a computing model which provides various forms of cloud services resources like web services, Application Programming Interface (API) which are leveraged as Infrastructure as a Service (IaaS), Platform as a Service (Paas) and Software as a Service (SaaS), as cloud computing has started to gain momentum, architects are looking for more ways to integrate their traditional application with the cloud model. The service providers in today's age have come up with predefined list of cloud APIs which can help anyone who wants to integrate their services in the cloud. As such APIs are not a new word in computing world. APIs in general are defined as a set of programming instructions and standards for accessing a

web-based software application or web tool. A software company releases its API to the public, so that, other software developers can design products that are powered by its service. An API is not a user interface but a software-software interface. This enables applications to communicate with each other. In the connected world, although it may look like we are interacting with a single system; the main system would call or interact many other sub systems using API functional calls. An example to this would be when we purchase goods online, the only visible portal would be the online shopping portal and this would interact with any other systems in terms of either providing data or receiving response to make each transaction successful.

In essence APIs are one of main building blocks of non-standalone systems. As cloud computing deals with more of collaboration and socialization with different systems, Cloud APIs play a key role to integrate services in the cloud. In simple terms, the concept of APIs are nothing but a form of software as a service, in which a cloud provider enable users to leverage cloud computing in their systems through the list of APIs provided by the various cloud providers. With this introduction on cloud computing it would be the very important to ensure the APIs that are written or that which is used for integration does not compromise on quality and are completely validated before they are released to the market. Cretella and Di Martino (2012) in the paper "Semantic Web Annotation and Representation of Cloud APIs" have presented an analysis of techniques that can be used for semantic description of application programming interfaces exposed as web services through SOAP or REST based protocol. Wu and Lee (2013) bring out the importance of a strong API access control model for strengthening the security of cloud computing in the paper "Design and implementation of cloud API access control based on Oauth". Further, Jenkins *et al*. (2011) have proposed a new framework which by itself a cloud application which contains plugins for testing APIs of cloud platforms. This study focuses on explaining a hybrid model which can cater to API testing needs.

II NEED FOR CLOUD API TESTING-PROBLEM STATEMENT

Today's business revolution is not about developing or testing a system or an application for a requirement. The need has emerged out to be able to such that each APIs that is developed is going to be released to the market and hence could act as a standalone deliverable. Having said this, the quality of the designed or developed cloud API is very important for any API to have a longer shelf life. If the designed API for an application is not going to meet the required standards or the requirements of other applications any new application will opt for an API that would meet their requirement.

Each organization should take more caution in the API that they select as there may be a need for multiple different API models for your environment to function well. Also, cloud service model is very new and there are still some challenges to be ironed out. Specifically, platform and infrastructure compatibility have been an issue. There are times when applications just won't work with a cloud-based API platform. APIs do not belong to a single broad category such that validation and verification of them is less cumbersome, rather they can be classified in many ways which makes the testing or validation process a very detailed and an efficient one.

III CLASSIFICATION OF CLOUD APIS

Cloud APIs can be broadly classified based on the following types.

Classification based on the type of cloud services provided:

- PaaS APIs (Service-level): This APIs help in deployment of applications in Cloud
- SaaS APIs (Application-level): These APIs are designed in such a way that they can be used as a standalone service for building any cloud application
- IaaS APIs (Infrastructure-level): Commonly referred to as Infrastructure-as-a-Service, these APIs help in rapid provisioning or de-provisioning of cloud resources
- Cloud provider and cross-platform APIs: As today's environment does not limits is usage with a single cloud provider, these APIs play a greater role in providing cross platform capability

Classification based on the various initial conditions that can be subjected to an API: Each API will have a set of input conditions, based on that they can be classified as:

- Mandatory pre-setters.->requires few activities to be carried out before the API is called
- Behavioral pre-setters->There could be optional parameters which might be set or not set before the API is called

Classification based on the nature of the API:

- Operating system-API for MS Windows API for Apple Mac OS X (Cocoa)
- Application services API
- Web services API (REST or SOAP)

Classification based on the API declaration:

- APIs that do not belong to a class
- APIs that belong to a class but requires object to be created before API function call
- APIs that can be only by using the class reference but creation of object is optional

Classification based on the API invocation:

- Direct API call
- API call based on event trigger: e.g., mouse movement/mouse click
- API call based when an exception occurs, e.g.,

Classification based on the parameter passing:

- By value-the value is placed directly into the parameter list
- By reference->the pointer/address location of a variable is passed in the parameters list

Classification based on output of an API:

- Executes an operation an returns a value as an input for some other system
- Update/modify a registry or a resource
- Performs an operation but does not return anything

IV CLOUD API-STUDY AND ANALYSIS

To arrive at the proposed model for cloud API-testing as a service approach, a study (Anonymous, 2010, 2016) was conducted on Google app engine trial feature of application hosting and deployment.

First, the various cloud providers are analyzed to examine the APIs provided by each of them. Below are the cloud providers under the three main cloud service category.

Examples of IaaS providers include: Amazon EC2, Google Compute Engine, HP Cloud, Joyent, Linode, NaviSite, Rackspace, Windows Azure, Ready Space Cloud Services, Terremark and Internap Agile.

Examples of PaaS include: AWS Elastic Beanstalk, Cloud Foundry, Heroku, Force.com, Engine Yard, Mendix, Open Shift, Google App Engine, App Scale, Windows Azure Cloud Services, Orange Scape and Jelastic

Examples of SaaS include: Google Apps, Microsoft Office 365, Petrosoft, Onlive, GT Nexus, Marketo, Casengo

V API-STUDY-EXPERIMENTAL SETUP

The following setup was used to analyze the various APIs provided by GAE in designing the proposed model (Fig. 1).

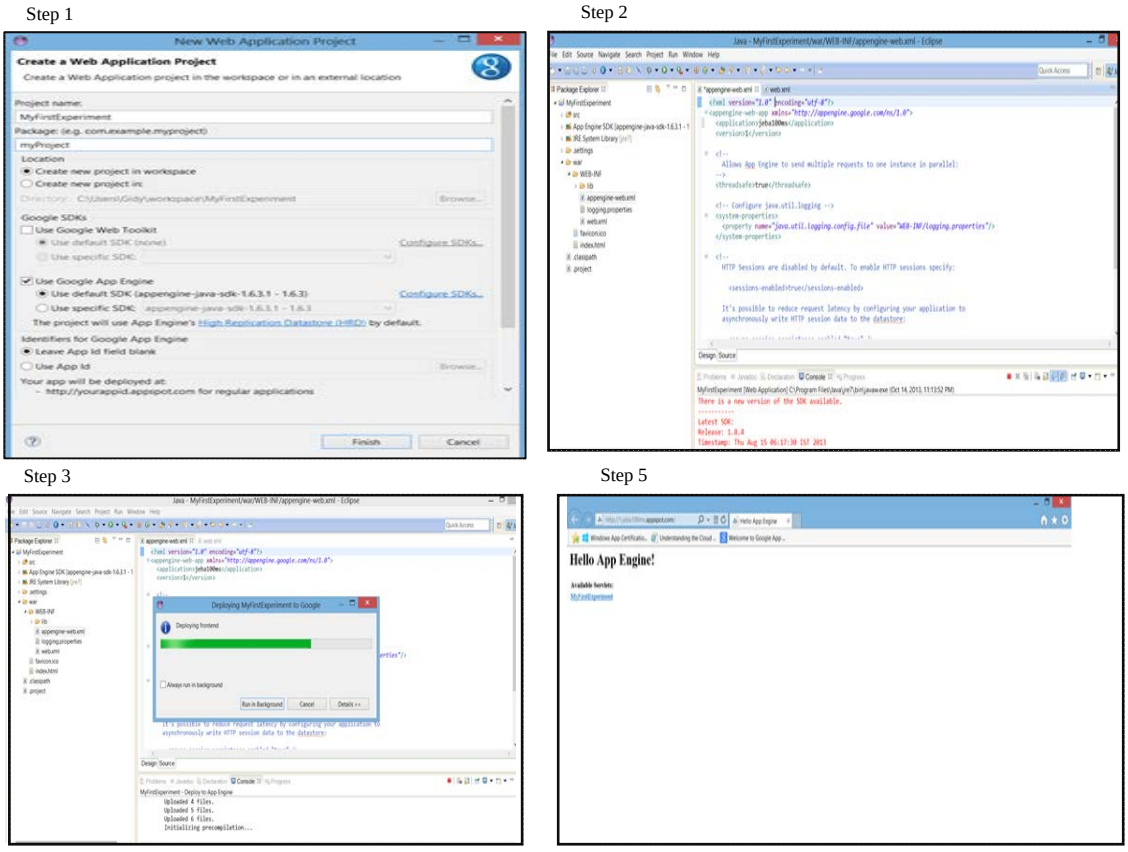

Fig. 1: Experimental setup

Step 1: Create a Web Application project in Eclipse IDE.

Step 2: In the created App Engine.xml file, supply the application identifier that was created using Google account trial period setup.

Step 3: After the application is run in the local host, it is time to deploy this in Cloud. Using the installed Eclipse plug-in for Google app engine, deploy the application to the cloud using the Google ID and password.

Step 4: The application is finally hosted. The following study was conducted by taking two infrastructures as a service provider, Google compute Engine and Amazon EC2 namely (Anonymous, 2010, 2016). There are various APIs provided by these service providers and hence the structure and request and response of the two operations were compared:

- VM instances related operation
- Image related operation

It was noted that the APIs format and structure used by the service provider Amazon EC2 and Google Compute Engine differ from each other. This proves that any model that can prescribed for providing cloud API testing platform or testing services should be easily customizable to suit the needs of different APIs that can be developed in various technologies.

VI PROPOSED MODEL FOR API TESTING AS A SERVICE

We have seen in the previous chapter on the different types of APIs available in today's world to suit different business needs. This section would provide an integrated architecture that can be used to leverage the concepts of cloud for testing cloud APIs and for providing a system which can help in configuring the required environment that is required for cloud testing. The main modules in the proposed model are discussed below (Fig. 2).

Test selector: The selection of functional or automated test cases based on the requirements is done by the test selector module. The test cases and automated scripts are stored in such a way that the model is highly reusable.

Test suite module: Static Code Analyzer or API specification reviewer: A detailed review of the API specification and any related use case documentation can be done in this module. This review of the API

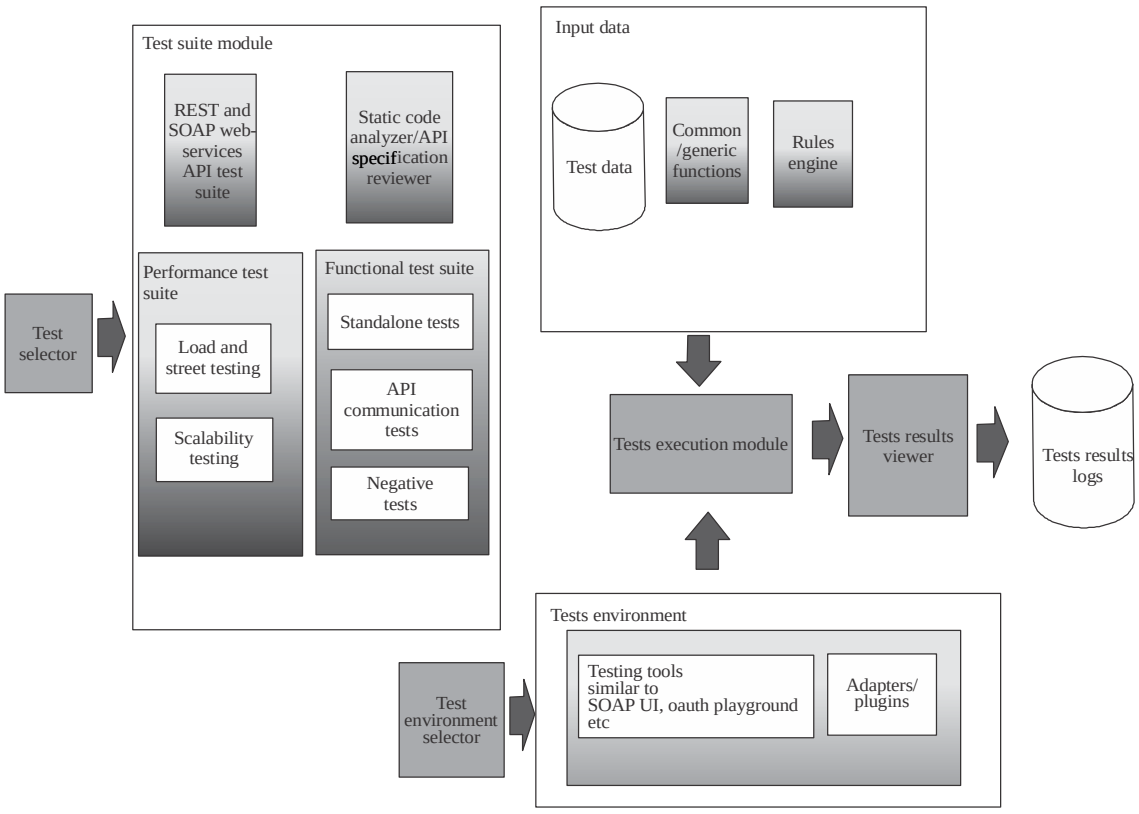

Fig. 2: The main modules in the proposed model

specification from a test perspective typically uncovers numerous errors in the implementation before a single test case is written.

Performance test suite: Performance testing an API includes creating a test environment, setting realistic performance targets, developing test scenarios, generating high-quality test input data, test execution and root cause analysis.

Functional test suite: REST and SOAP web services API Testing: The following tools can be integrated with the model to support the web services testing based on the requirement. Optimyz-Webservice tester is an end-to-end product offering automatic test generation; functional, regression and load testing; conformance testing against WS-I Profiles, BPEL-based orchestration testing; secure Web services testing and debugging and diagnostics.

Mercury (now HP)-"end to end" solution for Web services testing in the form of three offerings: Load runner, quick test professional and business process testing, its newest tool that sits on top of load runner.

Empirix Inc.-e-TEST: e-Manager enterprise, test management; e-Tester, functional testing; e-Load, scalability testing. Parasoft-SOAPtest, WSDL validation, unit and functional testing of the client and server, performance testing.

IBM rational software Co.-Test studio, unit, functionality, performance and load testing; PurifyPlus, runtime analysis tool for detects memory and performance bottlenecks early in the development cycle:

Standalone tests: Deals with the standalone behavior of the API without any functional call sequence.

Negative tests: Deals with providing the negative inputs and checking on the system behavior and stability.

API communication tests: Deals with tests that integrate one or more system using APIs.

Test execution module: Test execution module executes the selected test cases based on the input provided either using an automated test script on a given environment or by manually conducting the tests in the selected test environment.

Test environment selector: This module integrates the much need environment for the required testing, be it mobile devices or any particular server configuration in a dynamic way until the required testing environment is setup for the cloud APIs. For e.g., the Oauth play ground provided for validating the Oauth related specification is already available in today's world to meet this need. This architecture suggests the required plugins must be developed to ensure that these are available inside a single system.

Test results viewer: Test results play a major role during the quality assurance process. Storing of test results helps in identifying the root causes of major issues and helps in defect predication and prevention. Also, the logs help in reporting and in metrics derivation.

CONCLUSION

One of the biggest challenges in the Cloud Computing is to provide a most viable solution for the major risks or issues raised by its major components. In this paper we have made extensive study and have found that Cloud APIs as one of the major category of cloud architecture. As such APIs is not a new word in computing. However Cloud APIs need more study because of the mobility and huge data transfer required by cloud related service providers from the time they were invented. In the initial sections of this study, we have seen the use of APIs by all the cloud service providers as a way of enhancing their business and providing technological solutions for various business requirements. As we have seen there are various types of cloud APIs and as such testing has to be done in various forms to ensure that a quality system is erected. This paper concludes by providing a architecture to accommodate various types of cloud APIs which by its design can be easily extended for future needs.

REFERENCES

Anonymous, 2010. Cloud testing. United Technologies Corporation, Farmington, Connecticut.

Anonymous, 2016. Amazon elastic compute cloud API reference. Amazon Web Services, Inc, Seattle, Washington.

Cretella, G. and B. Di Martino, 2012. Semantic web annotation and representation of cloud APIS. Proceedings of the 2012 3rd International Conference on Emerging Intelligent Data and Web Technologies, September 19-21, 2012, IEEE, Bucharest, Romania, pp: 31-37.

Jenkins, W., S. Vilkomir, P. Sharma and G. Pirocanac, 2011. Framework for testing cloud platforms and infrastructures. Proceedings of the 2011 International Conference on Cloud and Service Computing, December 12-14, 2011, IEEE, Hong Kong, China, pp: 134-140.

Li, B., B.Q. Cao, K.M. Wen and R.X. Li, 2011. Trustworthy assurance of service interoperation in cloud environment. Int. J. Autom. Comput., 8: 297-308.

Wu, M.Y. and T.H. Lee, 2013. Design and implementation of cloud API access control based on OAuth. Proceedings of the IEEE 2013 Tencon-Spring Conference, April 17-19, 2013, IEEE, Sydney, Australia, pp: 485-489.