Framework for validation of cloud management based Micro Services

[1]Venkata Prasanna Verma N, [2]Vinay Kumar H S
[1]Department of Computer Science and Engineering, Visvesvaraya Technological University, Bengaluru, India
[2]Department of Electronics and Communication Engineering, Visvesvaraya Technological University, Bengaluru, India

Abstract – Software updates are crucial in determining availability of the software. Validating an upgrade, post development assures better software uptime by ensuring that a upgrade is valid and successful .The framework discussed in this paper addresses Docker services. On manually performing the validation operation, there will be a delay in the release of the software along with huge scope for failures. The proposed validation framework automates the validation of cloud management software by performing a set of pre- upgrade, upgrade and post upgrade tasks. Once an equivalence is achieved between the results of pre and post upgrade tasks, then the upgrade is termed as valid. The system developed automates the task of software upgrade validation ensuring an effective and faster software upgrade release. It is highly extensible and can be used to automate the validation of various end to end use cases like software migration (moving software from one platform to another), software upgrade (includes patches, newly added features etc.) and many more. The proposed validation framework provides a performance efficiency of almost 90% when compared to performing the validation manually.

Keywords– Cloud, Software Validation, Software migration, software framework, upgrade, Uptime.

I. INTRODUCTION

Cloud computing refers to the practice of using a network of remote servers hosted on the Internet to store, manage, and process data rather than using a local server or a personal computer. Computer hardware/software that we access through desktop is provided by cloud service providers over the internet. Its relevance is increasing by the day as it enables developers to focus on just building their applications rather than working on underlying infrastructure and the much needed development platform. Its key feature is that it stays managed i.e. customers only use the services that is provided and managed by someone else on their behalf. Cloud services are available on demand and follow the pay as you go model. There are 3 types of cloud deployments- private, public and hybrid. Private cloud is exclusive to a single company. Hardware and software are owned by the company which runs the services needed for that company. Public cloud is managed by service providers and its implementation is abstracted from the public users. User or company takes part of compute, network and storage to run its applications. Hybrid cloud is a popular deployment that combines the benefits of both private and public cloud.

With the increase in application of agile methodologies in software development, a number of patches and minor features get added to the developed software on a regular basis. This is most importantly done to ensure that the software meets the current customer needs. This methodology also ensures that developed software is flexible and secure in operating on the changing IT infrastructure. Software updates include new or enhanced features and compatibility enhancements. These updates ensure a better user experience. Recently, most of the companies are focusing on security updates which are very critical in keeping the users data safe. Release of faulty upgrades adversely affects the software availability.

Bugs in the released software affects the availability of the software which is unacceptable especially in cases like medical devices, nuclear reactors etc. This in turn results in huge financial loss, which in turn develops a conservative attitude to changes and failure to exploit advancement in technology. Validation of the upgrade plays a crucial role in ensuring the security, availability and functionality of the stays unaffected.

Validation of an upgrade can be done in many ways as required by the developer community. Validation involves performing a set of tasks before the upgrade and checking if the result of the tasks performed stays unchanged. The tasks involves creating data dependencies, populating of data, making some changes to the core elements of the software and verifying if the changes made remain intact. If the pre and post upgrade tasks produce same results on the core elements that are tampered upon then we can say that the upgrade is valid. Once the upgrade is confirmed as valid, it is then released to the customers. The software developed can be both upgraded and downgraded if needed, automatically with the help of the developed framework.

In this paper, we consider CSA - a cloud management software with which the framework interacts. CSA is a web-application which is hosted by a JBoss server. Here the pre upgrade task mainly involves setting up a certain environment and populating data into the existing Cloud Service Automation (CSA) application of the Hybrid Cloud Management (HCM) suite. To do so Representational State Transfer (REST) calls are made to the CSA application. REST API is an application programming interfaces which uses both HTTP/HTTPs request to GET, PUT, POST and DELETE data. These Application Programming Interfaces (API) are used in web service development for making calls to remote servers either to perform certain tasks like creating, updating data or getting some data from the servers. REST is simple and uses the features of HTTP to achieve its objectives rather than creating new standards.

A. REST URI's, LOG4J and URLs:

Any request made to the server via a REST API is as simple as URI call. The most commonly used data exchange formats are Java Script Object Notation (JSON) and Extensible Markup Language (XML). Data can be interchanged between the server and the client in the above mentioned formats. The software architect is given freedom to choose any data exchange format too like exchanging binary data. The operations used by default while Performing a REST action include GET to retrieve details from the server, DELETE to delete resource in the server, PUT for saving data on the server and for much complex tasks other than just reading, saving and deleting, POST method is used.

Basic constraints of a Rest based system are:

1. Use of Uniform Interface (UI):- The resources in the server should be accessible only through GET, POST, DELETE, PUT methods via A URL only.
2. Client-server based: - There should be loose coupling between the client and server so that both can be developed and enhanced independently.
3. Stateless operations:- All the operations between the client and server should be stateless and state management should occur on the client side.
4. RESTful resource caching:- In order to save time, reduce latency and improve performance in multiple consecutive retrieval of similar resource the resources are to be cached unless and until explicitly indicated not to do so.

An important requirement for good software development is adding logging functionality to the code. It not only helps in debugging but also in easy maintenance and structure storage of applications runtime information.Log4j is used to achieve logging action.Log4j is a reliable logging framework from Apache Software. It can be configured through external files at runtime. The Log4j framework is depicted in figure 1.1. It has 3 major components:

1. Loggers- they capture logging information
2. Appender - they publish logging information to preferred destination.

3. Layouts - used for formatting the logged information.

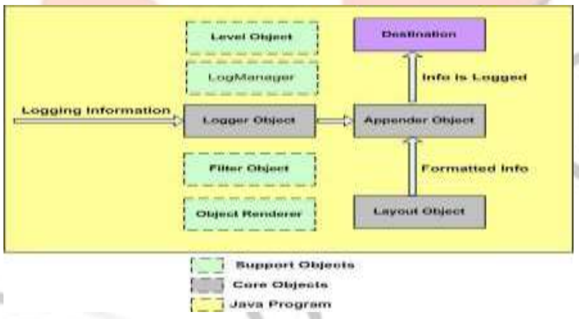

Figure .1. Components of Log4j Framework

The proposed framework is developed which uses all the concepts briefly discussed in this paper to help in validating the upgrade of say CSA application thereby reducing the lapse time between the development and release of the software. As a result of which, better management of the cloud services can be achieved due to faster and stable release of the cloud management software. Framework is a layered structure consisting of abstract classes with functionalities that can be used for a particular operation or extended if additional functionalities need to be added. The extensibility feature of framework is what can be used to build something useful in short time span. Some frameworks include interfaces, some programs, and some program usable library files. By thereby building a framework that automates the various tasks to be performed for validating use cases like upgrade /migrate of CSA, a proper upgrade can be released in no time as the manual task performed by testers is automated through various class files which are a part of the framework.

II. LITERATURE REVIEW

Software verification and validation are important phases in application lifecycle. Validation of a software consumes a lot of time which delays the release of a software even though it was developed earlier. Hence it is important to validate a software as soon as possible.

There are a lot of methods and techniques that can be used for software verification and validation. Shaoying Liu [1] studied and analyzed existing techniques and methods for software verification. He explains their strengths and weaknesses through model checking, formal proof, software testing, software review and program analysis. Verification and validation of large scale software with complex data structures are still a challenge to the research and application communities. As a result of the

significant progress made by extensive research over the last few decades, several major categories on OS techniques have been proposed and established such as model checking, formal proof, software testing, software review and program analysis.
As described by Somayeh Asadollahi, Vahid Rafe, Reza Rafeh [2], Design and Development of modern software systems includes a large collection of complex software artefacts, therefore designing them before implementing is an important task. It's very important to have software systems without bugs or defects. Authors [2] presented an efficient solution for verifying software systems by layered graph transformation systems,where static part of the system is represented by graphs and dynamic part of the system is defined by graph transformation rules.

III. METHODOLOGY

The proposed framework automatically validates an upgrade and ensures that only correct upgrade is released to the customers. The proposed framework performs the following:

1. It populates data on the existing version of the software which involves populating data in various sections of the software like creating organizations, importing designs(Designs are the blue prints for creating resources like new VM's) etc.
2. It initiates an upgrade process automatically by interchanging necessary container images and restarting the changed Docker pods.
3. It validates data set after the upgrade by performing a series of validation tasks which involves verifying that the data populated and dependencies created prior to upgrade remain unaffected.
4. If the data equivalence between the results of the pre upgrade and post upgrade is achieved,then the upgrade is considered valid and released to the customers else the upgrade is modified to achieve the desired results.

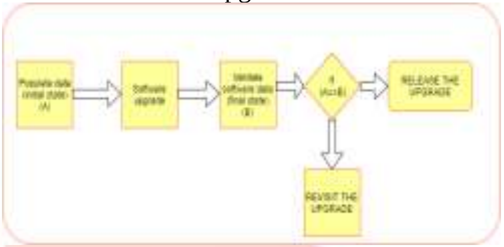

Figure .2. Components of Validation Framework

As in figure 2 with the proposed framework we are able to effectively automate the task of Software upgrade validation which fastens the process of CI (Continuous integration) thereby ensuring an effective and efficient update process with a better software release at the end.

IV. SYSTEM ARCHITECTURE

System architecture gives an overview on the working of the developed framework. in achieving automation of validating certain use cases like software update, migration and many more by using CSA.
The System architecture mainly discusses about the framework developed to automate validation of tasks like software upgrade, software migrate and similar use cases. To perform validation and automate the corresponding task, it has to interact with external systems as shown in the figure 3. Input to framework is given through the external test cases programed by the respective software developers. These class files contain functional calls to access CSA to perform some specific tasks. The framework in between helps the class files in communicating and performing the required task on the CSA. Framework internally contains the following
Data Handler class - Used to structure the data to be sent during Http request to the CSA.
HttpHelper class - Helps in creating a configured Http Object and managing with the Login credentials.
Utility classes - Help in making the necessary REST calls to CSA with the appropriate data.
The external test classes just contain a function call to perform a specific task and functional parameters as input to perform required task. Utility class makes a respective functional call to the CSA API. Prior to making the REST call, it makes a functional call to the Data Handler class to prepare the data to be sent to CSA and the HttpHelper class to get the HTTPS object which is used to make a call to the CSA server's which then sends response to the framework which in turn processes the response and sends a valid output to external Test files.

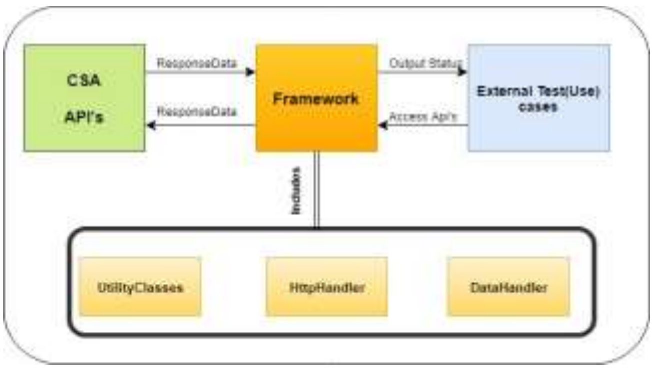

Figure .3. System Architecture inclusive of the Validation Framework

A. *Structural Representation of the Validation Framework*

The structure chart as seen in figure 4 shows the control flow among the modules in the system. It explains all the identified modules and the interaction between the modules. It also explains the identified sub-modules. The structure chart explains the input for each modules and output generated by each module.

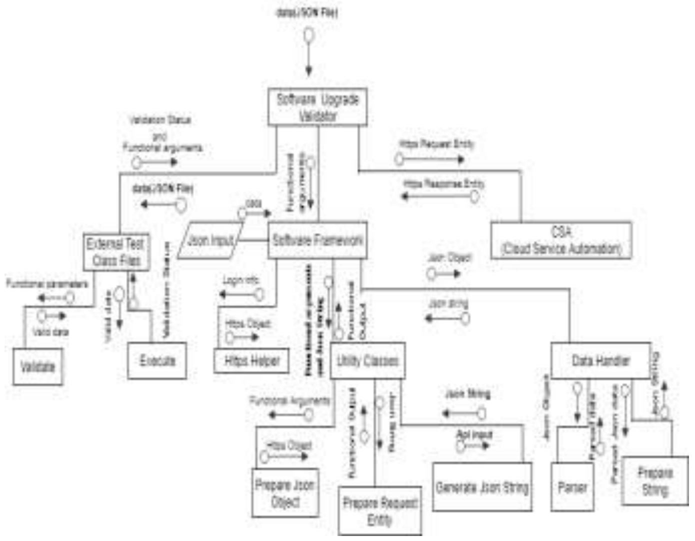

Figure .4. Structure Chart of proposed Framework

A. *Sequential Diagram*

A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

Sequential interaction of the framework modules with CSA is depicted in figure 5. As seen in figure 5, there are four objects in interaction which are: External Test cases, Utility classes, Data Handler class and CSA application. The interaction is initiated by the external test classes wherein they make a functional call corresponding to the required task that is to be achieved by specifying function name and input parameters .This call is made to the utility classes which manage making REST call to CSA. The Data Handler sends the formatted Json string to Utility class which then makes a Rest call to CSA. The Utility class then sends the resulting response after processing CSA's response to the external test cases. In case call to CSA fails the whole process gets terminated immediately with the exception details and cause of failure written in the log file.

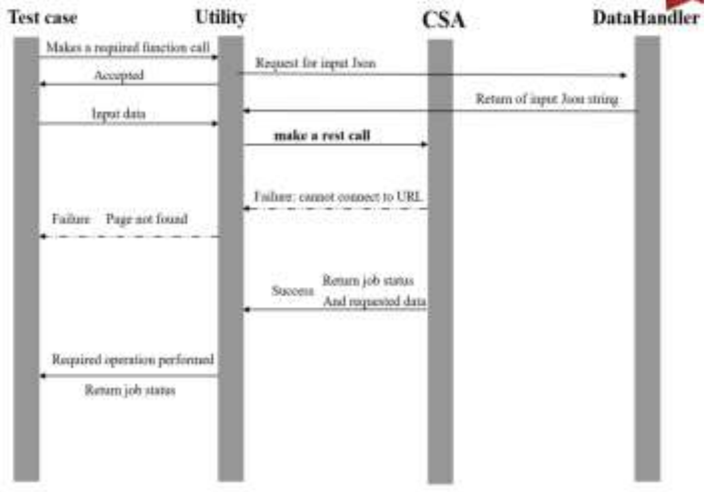
Test case
Utility
CSA
DataHandler
Makes a required function call
Accepted
Request for input Json
Return of input Json string
Input data
make a rest call
Failure: cannot connect to URL
Failure Page not found
Success Return job status And requested data
Required operation performed
Return job status

Figure .5. Sequential interaction involving framework modules

IV. EXPERIMENTAL ANALYSIS AND RESULTS

Analysis of a process, experiments are commonly used to evaluate the inputs of which process will have a significant impact on the output of the process, and how much the target level of those inputs should be to achieve a desired result. The output obtained from the system is compared with the truth to verify the correctness of the system. There are several metrics for comparison. Analyzing the experimental output is verifying whether the evaluation metrics are satisfied.

A. *Evaluation Metrics*

In the following discussion, Validation framework's performance is analyzed by taking Micro Focus's CSA software deployed in a containerized environment as an example.

In this paper, the outputs that are obtained from the different input parameters given to the system are compared with the expected outcomes to check whether the metrics are satisfied. The required metrics of evaluation as per which the validation framework should be evaluated against are:

1. Percentage of errors in validating an upgrade
2. Time taken to validate an upgrade
3. Remote access to CSA

B. Performance Analysis of Results

This section explains the experimental results of this project. The accuracy of around 90% is achieved in the test cases of the project in real time scenarios. Performance achieved by automating a variety of tasks required to perform validation of an upgrade is analyzed by comparing the time taken to complete a variety of tasks to be performed before the upgrade, during the upgrade and after the upgrade manually and on using the framework. Column Graphs are used to depict the performance variation of the tasks when performed manually and when automated on the three different stages of automation achieved by the framework. The X-axis depicts the external test class files to be executed and the Y-axis depicts the time taken to complete the task in **Seconds**.

Figure 6 represents the time taken to complete a lists of tasks performed during the pre-upgrade stage, the upgrade stage, post-upgrade stage when performed manually and when automated by invoking the developed framework The graph also represents time taken to perform the complete validation tasks when performed manually and upon being automated.

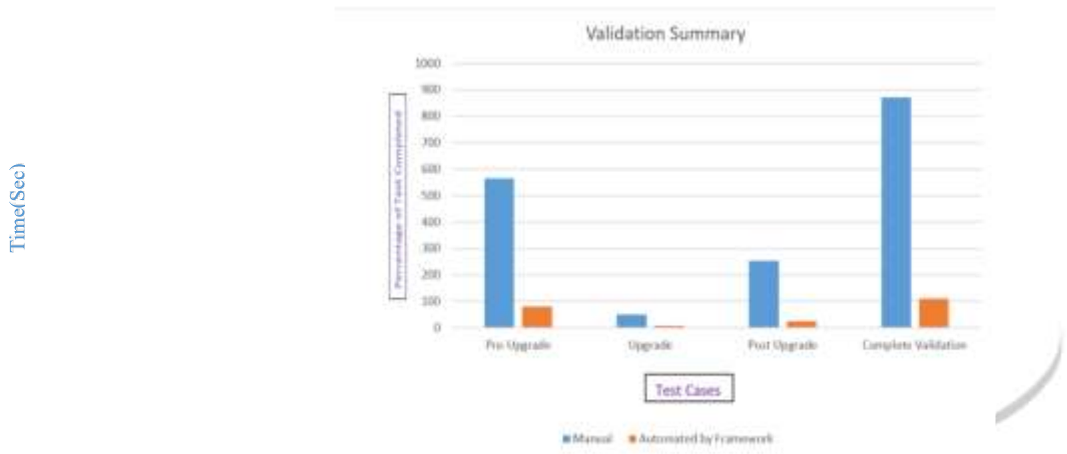

Figure 6. System Analysis on various tasks performed for upgrade validation

The results depicted in the above graph is based on data collected on running the framework and performing the tasks manually multiple times to get the average time taken to complete the tasks.
The efficiency achieved in terms of performance with time as a base parameter is depicted in the table 1:-

Table .1. System testing (Overall Efficiency)

Tasks performed	**Efficiency achieved on using the framework**
Pre-Upgrade tasks	85.85129%
Upgrade task	84.503%
Post-Upgrade tasks	90.113%
Total System	**87.116%**

Hence the framework developed helps in increasing the performance efficiency by almost 90%.

V. CONCLUSION AND FUTURE WORK

Ensuring that faulty updates are not released is very important for technological businesses to run successfully. So the quick release of updates with proper validation done prior to release in an automated environment helps in better cloud management and increases the reliability and availability of the management software.
The developed validation framework ensures that a proper upgrade for the cloud management software is released on time. It automates the manual task of upgrade validation which involve tasks like pre upgrade setup ,running the upgrade followed by post upgrade checks which saves a lot of time in the release of the software upgrade thereby making the off period of the cloud management services minimum ,which finally ensures that the management of cloud resources by the software is happening with complete efficiency which leads to nothing but customers satisfaction, which is the required ultimate fruit in the end. The framework provides an increase in performance efficiency of almost 90% with respect to the use case of software upgrade validation.

Future Work

Enhancements are a necessary key to continuous improvement and increased efficiency. Stated below are a few enhancements or future work that can be incorporated/ achieved in this project:

- Achieving better accuracy with different input set.Presently the validation framework works with only Json files as an input. Its performance is to be tested when developed using input in alternate formats like xml.
- Adding additional Utility classes to the framework covering a broad spectrum of functionalities. The utility classes which provide diverse inbuilt functionalities to the developer are presently limited to a few tasks for demonstration purpose and they are to extended based on the need.
- Adding GUI to the framework developed. The validation framework is now run on the command prompt and does not support a GUI interface.

VI. REFERENCES

[1] S. Liu, "*Testing-Based Formal Verification for Algorithmic Function Theorems and Its Application to Software Verification and Validation*," 2016 International Symposium on System and Software Reliability (ISSSR), Shanghai,2016,pp.1-6.doi: 10.1109/ISSSR.2016.010

[2] S. Asadollahi, V. Rafe and R. Rafeh, "*Towards Automated Software Verification and Validation*," 2009 International Conference on Computer Technology and Development,Kota,Kinabalu,2009,pp.206-210. doi: 10.1109/ICCTD.2009.164

[3] T. Berling and M. Host, "*A case study investigating the characteristics of verification and validation activities in the software development process*," 2003 Proceedings 29th Euromicro Conference, 2003, pp. 405-408.doi: 10.1109/EURMIC.2003.1231623

[4] D. R. Wallace and R. U. Fuji, "*Software verification and validation: an overview*," *in* IEEE,vol.6,no.3,pp.1017,May1989.doi.10.1109/52.28119

[5] D. G. Bell and G. P. Brat, "*Automated Software Verification & Validation: An Emerging Approach for Ground Operations*," *2008 IEEE Aerospace Conference*, Big Sky, MT, 2008,pp.1-8.doi: 10.1109/AERO.2008.4526648.

[6] G.R.N.,"*Testing Processes in Business-Critical Chain Software Lifecycle*," 2009 WRI World Congress on Software engineering,Xiamen2009,pp 238-242.doi: 10.1109/WCSE.2009.424

[7] YingHui Wang, XiuQing He and QiongFang Wang, "*Lifecycle based study framework of software evolution*," 2010 International Conference on Computer Application and System Modelling (ICCASM 2010),Taiyuan,2010,pp. V3-262-V3-266.doi: 10.1109/ICCASM.2010.5620014

[8] R. Torkhan, J. Laval, M. Derras and N. Moalla, "*Conceptual interoperability framework for software development and maintenance*," 2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC), Funchal, 2017, pp. 1327-1332.doi: 10.1109/ICE.2017.8280034

[9] R. Torkhan, J. Laval, M. Derras and N. Moalla, "*Conceptual interoperability framework for software development and maintenance*," 2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC), Funchal, 2017, pp. 1327-1332.doi: 10.1109/ICE.2017.8280034

[10] M Lehman, D. Perry, J. Ramil, "*Implication of evolution metrics on software maintenance*", *Proceedings on the International Conference on Software Maintenance IEEE computer soc.*, pp. 208-217, 1998.

[11] S. R. Balakrishnan, S. Veeramani, J. A. Leong, I. Murray and A. S. Sidhu, "*High Performance Computing on the Cloud via HPC+Cloud software framework*," 2016 Fifth International Conference on Eco-friendly Computing and Communication Systems (ICECCS), Bhopal, 2016, pp.48-52.doi: 10.1109/Eco-friendly.2016.7893240

[12] De Lucia, A.R. Fasolino, E. Pompelle, "*A decisional framework for legacy system management*", IEEE International Conference on Software Maintenance, pp. 642-651, 2001.