



UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

A MULTI-TENANCY CLOUD TRUST MODEL USING QUALITY OF SERVICE
MONITORING: A CASE OF INFRASTRUCTURE AS A SERVICE (IaaS)

By

Pascal M. Mutulu

P53/10982/2018

Supervisor

Dr. Andrew M. Kahonge

A project report submitted in partial fulfillment of requirements for the award of Master of
Science in Distributed Computing Technology of the University of Nairobi.

2020

DECLARATION

This research project is my original work and has not been submitted for examination in any other university

Signature:

Date:

Pascal M. Mutulu

P53/10982/2018

This research report has been submitted in partial fulfillment of the requirements for the award of Master of Science in Distributed Computing Technology of the School of Computing and Informatics of the University of Nairobi, with my approval as the University supervisor.

Signature:

Date:

Dr. Andrew M. Kahonge

DEDICATION

This research project is dedicated to my lovely wife Grace, my two children Ariel and Blaise and my dear parents Japheth and Ruth.

ACKNOWLEDGEMENT

First, I would like to thank almighty God for bringing me this far and enabling me to complete this research project.

Secondly, to my supervisor, Dr. Andrew M. Kahonge for the dedication, guidance and encouragement throughout the project. Your patience and humility impacted me at a personal level, and I am so grateful for the lessons I have learnt from you.

I would also like to acknowledge my wife, Grace Musyoki and children, Ariel Ndanu and Blaise Mutulu for standing by me throughout the process. Your words of encouragement kept me moving. Though I worked for long hours and my interaction with you was affected to some extent, you still had hope that this will come to an end and our normal interactions would resume.

Finally, to my classmates Titus and Joseph. You were instrumental to me through giving me honest feedback and criticizing my work positively. For your constant reminders that we should keep progressing, I say thank you.

ABSTRACT

Digitization and changes in technological trends have necessitated the need for enterprises to start or have plans of migrating their services to cloud computing environments. This is to benefit from the many advantages that come with cloud computing. Third party providers whom majorly consume multi-tenancy architectures mainly offer the cloud platforms. This come with some challenges mostly when it comes to trust. The cloud consumers and cloud providers agree on some cloud service level agreements. Mostly the consumers have faith that they benefit from what they have agreed with the provider but lack a way of verifying the SLAs as well as doing QoS monitoring on their own.

This research project focuses on coming up with a multi-tenancy cloud trust model using QoS monitoring. Our key focus was the infrastructure as a service cloud model. It also involved developing a prototype to show case the proposed model. The overall research strategy employed was exploratory.

The model developed assists cloud consumers to be able to evaluate cloud services before they purchase services. This prevents them from leasing already congested clouds, or which do not meet their specifications. They also have the capability of continuous QoS monitoring of the cloud environment in real time when need be. On the other hand, cloud providers also benefit from the trust provided by our model because it might lead to good company reputation making them to sell more.

TABLE OF CONTENTS

| | |
|---|------|
| DECLARATION | ii |
| DEDICATION | iii |
| ACKNOWLEDGEMENT | iv |
| ABSTRACT..... | v |
| ABBREVIATIONS AND ACRONYMS | ix |
| LIST OF FIGURES | xi |
| LIST OF TABLES | xiii |
| CHAPTER ONE: INTRODUCTION..... | 1 |
| 1.1 Background of the research..... | 1 |
| 1.2 Problem statement..... | 1 |
| 1.3 Objectives..... | 2 |
| 1.4 Research questions | 3 |
| 1.5 Justification | 3 |
| 1.6 Scope | 3 |
| 1.7 Research Significance | 3 |
| CHAPTER TWO: LITERATURE REVIEW..... | 5 |
| 2.1 Cloud Computing (CC)..... | 5 |
| 2.1.1 Definition of cloud computing | 5 |
| 2.1.2 Key characteristics..... | 5 |
| 2.1.3 Service Models | 6 |
| 2.1.4 Deployment Models | 6 |
| 2.1.5 CC Reference Architecture..... | 6 |
| 2.1.6 Trust as key to adoption of cloud computing | 8 |
| 2.2 Multi-tenancy architecture and characteristics..... | 8 |
| 2.2.1 Multi-tenancy characteristics..... | 8 |
| 2.2.2 Multi-tenancy architecture..... | 9 |
| 2.3 Trust and its relation to multi-tenancy clouds..... | 9 |
| 2.3.1 The concept of trust | 9 |
| 2.3.2 Characteristics of trust | 10 |
| 2.3.3 Trust levels in the cloud..... | 10 |

| | |
|--|-----------|
| 2.4 Related works | 11 |
| 2.4.1 Chains of trust in the cloud..... | 11 |
| 2.4.2 Cloud Computing Service Security Strength Measuring Trust Model..... | 12 |
| 2.4.3 Collaborative cloud services Authorization models in multi-tenancy environments... | 13 |
| 2.4.4 Trusted computing environment model (MTCEM) | 13 |
| 2.4.5 Cross-tenant trust model (CTTM) | 14 |
| 2.5 QoS Monitoring..... | 15 |
| 2.6 Conceptual Model | 16 |
| 2.7 Summary | 17 |
| CHAPTER THREE: METHODOLOGY | 18 |
| 3.1 Research design strategy | 18 |
| 3.2 Delphi method | 18 |
| 3.2.1 Characteristics of Delphi method | 19 |
| 3.3 Population and Sample..... | 19 |
| 3.4 Data Collection..... | 19 |
| 3.5 Data Analysis | 20 |
| 3.6 Prototype Design..... | 20 |
| 3.7 Tools..... | 20 |
| 3.7.1 OpenStack cloud..... | 20 |
| 3.7.2 Prometheus | 21 |
| 3.7.3 Grafana | 23 |
| 3.8 Conclusion and justification of the methodology | 23 |
| CHAPTER FOUR: RESULTS AND DISCUSSIONS..... | 24 |
| 4.1 Data analysis | 24 |
| 4.2 The overall prototype architecture of the proposed model | 25 |
| 4.3 The cloud service provider environment..... | 26 |
| 4.3.1 The Cloud Platform (OpenStack)..... | 26 |
| 4.3.2 Actual provisioned resources for the OpenStack cloud..... | 27 |
| 4.3.3 OpenStack modules installed..... | 29 |
| 4.3.4 Broker | 33 |
| 4.3.5 Node exporter | 34 |

| | |
|---|----|
| 4.4 Third Party QoS Monitor | 35 |
| 4.3 The cloud consumer | 36 |
| 4.4 How the prototype works - End to End Integration of the components..... | 37 |
| 4.4.1 Low-level design | 37 |
| 4.4.2 Integration between node exporter and broker | 38 |
| 4.4.3 Integration between the broker and QoS monitor | 40 |
| 4.5 QoS metrics analysis | 41 |
| 4.5.1 Discrepancies between the actual and logical metrics assigned to the consumer. | 41 |
| 4.5.2 How the metrics are manipulated | 43 |
| 4.6 Evaluation of the model | 43 |
| 4.6.1 Formal testing..... | 43 |
| 4.6.2 Evaluation by the expert panelist..... | 44 |
| 4.7 The model shortcomings | 44 |
| 4.8 Discussions..... | 44 |
| CHAPTER FIVE: CONCLUSION AND RECOMMENDATIONS | 47 |
| 5.1 Conclusion..... | 47 |
| 5.2 Recommendations for further work | 47 |
| 5.3 Limitations | 48 |
| REFERENCES | 49 |
| APPENDICES | 52 |
| Appendix 1: Project Schedule | 52 |
| Appendix 2: Budget | 52 |
| Appendix 2: Hardware and Software requirements | 52 |

ABBREVIATIONS AND ACRONYMS

API - Application Programming Interface

AWS - Amazon Web Services

CC – Cloud Computing

CCTIM - Cross-Tenant Trust Model

CRTM - Core Root of Trust Measurement

CSA - Cloud Security Alliance

CSA - Cloud Security Appliance

CSP - Cloud Service Provider

CTA - Cloud Trust Authority

DC - Data Center

FSF - Free Software Foundation

GT - Grounded Theory

IaaS – Infrastructure as a Service

ISO - International Organization for Standardization

IT - Information Technology

MTAaaS - Multi-Tenant Authorization as a Service

MTC – Multi-Tenancy Cloud

MTCEM - Trusted computing environment model

NIST - National Institute of Science and Technology

NSE – Nairobi Security Exchange

OS - Operating System

PaaS – Software as a Service

QoS - Quality of Service

SaaS - Software as a Service

SME - Small and Medium Enterprises

SP – Service Provider

STAR - Security, Trust and Assurance Registry

TaaS - Trust as a Service

TC - Trusted Computing

TCG - Trusted Computing Groups

TCP - Trusted Computing Platform

LIST OF FIGURES

Figure 2: 1 CC Reference Architecture (Source - NIST, SP 500-292)

Figure 2: 2 Single versus multi-tenancy architectures

Figure 2: 3 Chains of trust in relation to cloud (Huang & Nicol 2013)

Figure 2: 4 MTCEM Model (Brown et al 2012).

Figure 2: 5 MTAaaS architecture (Tang & Sandhu, 2013)

Figure 2: 6 Conceptual model

Figure 3: 1 OpenStack landscape.

Figure 3: 2 Prometheus Architecture

Figure 3: 3 Project Schedule

Figure 4: 1 The overall prototype architecture of the proposed model

Figure 4: 2 The OpenStack login graphic user interface (GUI)

Figure 4: 3 Allocated disk space and partitioning.

Figure 4: 4 Allocated CPU.

Figure 4: 5 Allocated RAM

Figure 4: 6 OpenStack modules installed.

Figure 4: 7 Sample OpenStack dashboard created.

Figure 4: 8 Sample virtual machines created.

Figure 4: 9 Created Networks

Figure 4: 10 Identity service

Figure 4: 11 OpenStack images and storage

Figure 4: 12 Prometheus docker container

Figure 4: 13 Prometheus runtime and build information

Figure 4: 14 Node exporter.

Figure 4: 15 visualization of the cloud platform metrics.

Figure 4: 16 Different metrics of the cloud platform

Figure 4: 17 Sample cloud consumer view 38

Figure 4: 18 Low-level diagram of how the prototype works

Figure 4: 19 Node exporter service status

Figure 4: 20 Broker connection to the node exporter.

Figure 4: 21 Inter-connecting the Third party QoS monitor to the broker

Figure 4: 22 Sample metrics assigned to a cloud consumer

Figure 4: 23 Setting metric parameters

Figure 4: 24 QoS monitor showing actual resources.

LIST OF TABLES

Table 2.1: Definition of cloud Actors

Table 4.1: Metrics analysis

Table 4.2: Actual resources usage against what the consumer sees through the QoS monitor

CHAPTER ONE: INTRODUCTION

1.1 Background of the research

According to Ismail et al, 2017 most companies and organization are going through digital transformations by automating much of their traditional business processes. It states that companies not able to embrace the digitizing world may be victims of “digital Darwinism” and thus enterprises that cannot adapt to technological trends may not survive. With all this digitization, the various services need to reside in servers; building a data center (DC) is costly, consumes much time and requires huge capital to maintain. This has necessitated these clients to look for third party providers to offer them platforms where they can deploy their services.

The third-party providers commonly known as “cloud providers” consume virtualization to create different instances for the different clients. At a minimum, a virtualization technology has host hardware, hypervisor and the virtual machines. The client’s instances must not be able to interfere with each other but share resources (AlJahdali et al, 2014). They should be segregated and appear as a physical server to the customer. This necessitates the need for multi-tenancy technology.

On acquiring such services as compute, storage, networking the client and the Cloud Service Provider (CSP) agree on some service level agreements (SLAs) (Ansari, 2018). The cloud provider may commit that their cloud has all the cloud-computing characteristics such as redundancy, high availability, fault tolerance, optimum performance among others. Since the cloud provider has most of the control, depending on the service acquired by the client there is need for a way to confirm the provision of the agreements.

To enhance trust to the client that they are benefiting from all what they purchased for, there needs to be a third party means to confirm the same. This is the reason we came up with this trust model to address that problem. It makes it possible for the cloud consumer to confirm the cloud platform status in real time at any moment if they have access to the third party QoS monitor.

1.2 Problem statement

According to Odun-Ayo and Idoko, 2018 trust is a very complex belief. In cloud computing trust does not have a specific definition. One way is to describe it as the level of confidence the client has in the services a CSP offers. In some institutions it is required that their data is held following

some certain standards such as -: data should not move beyond a certain jurisdiction, replication should take place at a certain time, backups should be up to date, latency and response time should be within a certain threshold among many others. As discussed under the background information all these agreements are contained in the cloud service level agreements (SLAs).

Multi-tenancy cloud architecture is most commonly used by the CSPs. This enable them to offer different cloud consumers, services under the same hardware. The consumers, commonly known as tenants lease a space where they host their services. These consumers are abstracted from knowing each other and thus it might appear as if they are the only users. This brings a problem such that a tenant could be allocated logical resources, which are not physically available on the hardware.

Most cloud consumers depend on the same CSPs to offer them monitoring tools where they can confirm some of the metrics such as availability and latencies. Most of the other metrics the consumer has faith that the CSP will have them as agreed. This might be because bodies like International Organization for Standardization (ISO) certify the CSP (Huang and Nicol, 2013). This is not always the case as the design and architecture of these clouds could have changed leaving the cloud status not as at the point it was during the certification. In addition, other clouds may not be certified at all.

That is why we have proposed a third party QoS monitor, independent of the CSP that could be integrated to the cloud services acquired, fostering trust of the cloud consumer as they can verify resources according to the signed SLA. According Sen, 2013 there is lack of common industry standard that clients can scale their providers. Effective cloud-based monitoring tools would most likely need some integration of monitoring tools that are utilized by both consumer and the provider providing unambiguous identification of all actions carried out on client's cloud services.

1.3 Objectives

The main objective is to come up with a multi-tenancy cloud trust model using QoS monitoring for IaaS.

Sub objectives will be to achieve the following:

- i. To review the various trust models used in multi-tenancy clouds

- ii. To come up with a multi-tenancy cloud trust model using QoS monitoring.
- iii. To develop a prototype of the proposed model
- iv. To evaluate the model.

1.4 Research questions

- i. What is the match between the provisioned and actual cloud services offered to the consumer?
- ii. How can QoS monitoring enhance trust on multi-tenancy cloud in the IaaS cloud model?
- iii. How can increase in trust on multi-tenancy clouds contribute to its adoption?

1.5 Justification

Fully pledged cloud providers like Amazon Web Services provide a list of compliance reports from third party auditors known as Amazon artifact reports. The reports indicate whether the auditors have tested and verified Amazon's compliance with global, regional, and industry specific security standards and regulations. The released reports are publicly available in AWS artifact.

Since in Kenya most of the cloud providers do not publicly provide their compliance reports, then this research helps in providing a way for the cloud consumers in Kenya to benchmark and to ensure that the agreed metrics and agreements between them and the cloud service provider are achieved. It clears doubts on the clients and hence fosters their trust in the cloud providers and which in turn can boost cloud adoption.

Again, cloud is growing at a faster rate with the government and other institutions wanting to benefit from the advantages that come with cloud services. The research helps them in identifying the right requirements to assist in evaluating which cloud providers to adopt.

1.6 Scope

The scope of the project was to come up with a trust model of multi-tenancy clouds in the IaaS cloud model using QoS monitoring.

1.7 Research Significance

This research benefits cloud consumers by providing them with ways of doing cloud QoS monitoring to ensure that they are benefitting from all what they purchased from the CSP. The

model developed also helps in continuous monitoring of the cloud services as well as the overall cloud platform status in real time. This greatly helps them in evaluating cloud providers and choose the best depending on the quality of service they need.

On the other hand, cloud service providers benefit from the trust from the cloud consumers, such as leading to increase on cloud adoption making them to sale more. Trust brings with it good reputation, which may lead to more clients acquiring services from you as a result. Again, CSPs being aware that cloud consumers are sure of what they need and can confirm certain metrics using third party QoS monitor, make them ensure they follow the best practices as well as setting well planned and designed infrastructure in place.

CHAPTER TWO: LITERATURE REVIEW

This chapter shall discuss the following: - cloud computing, multi-tenancy clouds, trust, review of related literature, QoS monitoring, the conceptual model and then finally provide the summary.

2.1 Cloud Computing (CC)

2.1.1 Definition of cloud computing

CC is traced back to the mid 1990's when the grid-computing concept arose (Weinhardt et al 2009). Grid computing concept is a consequential model of the electrical power grid to put emphasis on features like reliability and simplicity (Foster and Kesselman 1999). It is evident that in the recent years, CC has turned into a trend to keep watch in the IT arena with features such as scalability, flexibility, availability among others. National Institute of Science and Technology (NIST) defines CC as a *“model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”*

NIST continues to give the most CC essential attributes, service and deployment models as below:

2.1.2 Key characteristics

- i. On demand self-service – Ability of users to spin services automatically lacking the necessity of assistance from the CSP.
- ii. Extensive network access - Services hosted in CC environment are reachable from various devices like tablets, laptops and mobile phone over the internet.
- iii. Resource pooling - CSPs computing resources share among multiple cloud users utilizing multi-tenant model.
- iv. Elasticity - Resources are adjustable automatically. This enables vertical or horizontal scalability to be rapid.
- v. Metered service – The cloud-based systems by design control, enhance resources through a means of a metering capability.

2.1.3 Service Models

The service models include the following, first is the Software as a Service (SaaS) where the user consumes commands running on the cloud provider's platform and does not control or manage any fundamental infrastructure. Second is Platform as a Service (PaaS). In this case the user can install onto the cloud infrastructure acquired applications programmed and supported by the provider. The consumer has control over the installed applications and may be can freely configure settings in the environment hosting the application. Lastly, we have Infrastructure as a Service (IaaS) where the consumer has the capability to control operating systems, installed applications and storage. They may also have some control on networking modules.

2.1.4 Deployment Models

Private cloud – This consumed by one organization. It might serve several consumers like business departments and managed by the organization, third party or even both parties. It might be located on either on or off premises.

Public cloud - Cloud infrastructure is open for consumption by the public

Community cloud - Used by a specified community who share the same interests such as policy or compliance. It can be under the control of one of the members or organization in the specific community or even a third party.

Hybrid cloud – It is a combination of two discrete cloud infrastructures or more bound by consistent technology allowing data and applications portability.

2.1.5 CC Reference Architecture

According to figure 2.1 below, NIST identifies five major cloud actors defined in table 2:1. These key players make the complete architecture cloud ecosystem. Each is an independent element that has its own structure and thus work together through the user of well-predefined technologies. Under the cloud provider is where service models reside. The IaaS is usually the bottom layer below the PaaS and SaaS and seats directly above the abstraction layer.

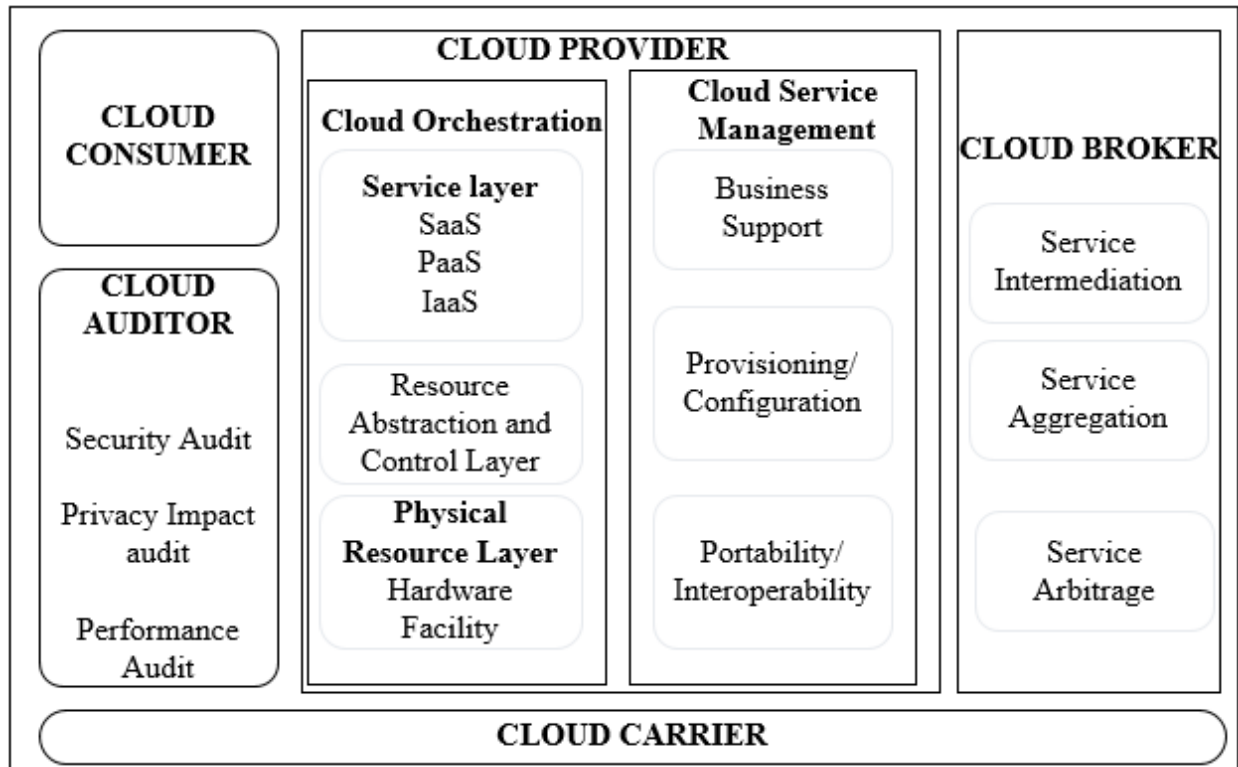


Figure 2: 1 CC Reference Architecture (Source - NIST, SP 500-292)

| Actor | Definition |
|----------------|---|
| Cloud Consumer | A person or organization that maintains a business relationship with, and uses service from, Cloud Providers. |
| Cloud Provider | A person, organization, or entity responsible for making a service available to interested parties. |
| Cloud Auditor | A party that can conduct independent assessment of cloud services, information system operations, performance and security of the cloud implementation. |
| Cloud Broker | An entity that manages the use, performance and delivery of cloud services, and negotiates relationships between Cloud Providers and Cloud Consumers. |

| | |
|---------------|---|
| Cloud Carrier | An intermediary that provides connectivity and transport of cloud services from Cloud Providers to Cloud Consumers. |
|---------------|---|

Table 2.1: Definition of cloud Actors (NIST, SP 500-292)

2.1.6 Trust as key to adoption of cloud computing

According to Meixner and Buettner, 2012, currently decisions on adoption of cloud related solutions; trust and security are the major obstacles to adoption and growth. They continue to state issues of security and trust as moderately solved so far. Clearly, literature about trust, security and cloud computing exist, though most of it is focused on IT. Human perspective, people’s anticipations, concerns as well as psychological aspects have less been examined and documented (Meixner and Buettner, 2012).

2.2 Multi-tenancy architecture and characteristics

NIST in the key characteristics 2.1.2 (iii) discussed above stated that resources sharing among multiple tenants is by use of multi-tenant architecture. It forms the basis of our literature in this section. Study done by Cloud Security Appliance (CSA) in 2017 on the top threats which organizations face in the adoption of cloud include shared technology vulnerabilities, abuse and disreputable consumption of cloud computing, intruders, loss of data among others. Multi-tenancy is acknowledged as one of the distinctive implications of security and privacy in CC.

2.2.1 Multi-tenancy characteristics

Multi-tenancy characteristics (Bezemer and Zaidman, 2010):

The first characteristics is hardware sharing. According to Wang et al, 2008 having numerous tenants on the same server improves utilization.

Secondly, highly configurable. In a multi-tenancy, tenants utilize the same application instance. In as such, the consumer requires it to appear as if they are using a dedicated one. As a result, a crucial obligation of multi-tenant application is the ability to design the application to a tenant needs the same way as in single tenancy (Mietzner, 2009).

And finally, application and database instance sharing. In multi-tenancy as the application is runtime configurable. Deployments such as updates are easy as few instances are affected.

2.2.2 Multi-tenancy architecture

The below figure 2.2 show the key difference between single-tenancy and multi-tenancy which have been discussed above in the multi-tenancy characteristics. In the single tenancy, each client has their own virtual server or even hardware while on multi-tenancy clients known as tenants share the same hardware or even the underlying infrastructure. This is abstracted from each other and thus from a specific tenant point of view it seems like its own physical server which is not shared.

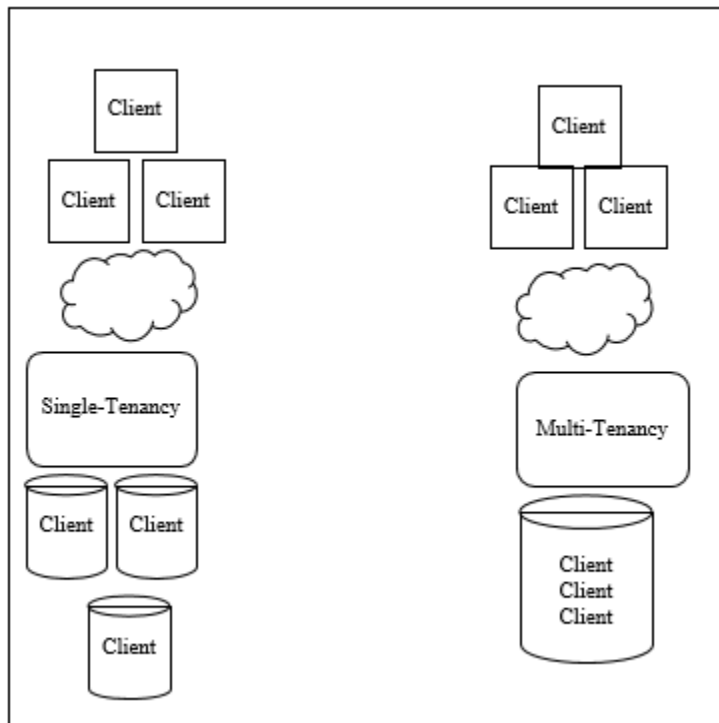


Figure 2: 2 Single versus multi-tenancy architectures

2.3 Trust and its relation to multi-tenancy clouds

2.3.1 The concept of trust

Trust has been known to people for a long time. It is old as the account of man and presence of human social relations. Classical disciplines such as economics, psychology and philosophy seem to have most of the literature and studies regarding trust. All of them focus on general understanding of trust. Philosophy for instance traces the trust concept back to the ancient Greek, where people-build trust in others only if they were confident that they feared detection and

corporal punishment, hence, that could prevent them from stealing or harming (Wang and Emurian, 2005).

Economics see trust in terms of organizational contexts while Psychology concentrations is on interpersonal trust and sees it as crucial to personality and development (Erikson, 1963). Based on social sciences we adopt the following definition by Huang and Nicol, 2013. It describes trust as a mental state that comprise three attributes. First is expectancy where the trustor anticipates some precise behavior from the trustee, second is belief where the trustor believes that the specific behavior happens centered on the confirmation of the trustee's competency, integrity and goodwill, and then thirdly is readiness that the trustor is willing to take risk because of the belief.

This research focuses on trust as it relates to multi-tenant clouds or largely CC.

2.3.2 Characteristics of trust

According to Wang and Emurian, 2005, the following characteristics of trust have mostly been researched and Meixner and Buettner, 2012, give their relationship with cloud computing.

Trustor and trustee – The relationship always depend on trusting party and the party to be trusted known as trustor and trustee respectively. In this case, cloud service provider becomes the trustee and the consumer becomes the trustor.

Vulnerability – Needed and works in areas where uncertainty, risk and vulnerability are involved. This might involve the large number of vulnerabilities consumers face with cloud computing.

Produced actions – Mostly yields actions that contains majorly risk-taking conducts. Consumers trust in cloud service might lead to client evening paying more and continuing to use it regularly.

Subjective matter – Observed differently by different parties. Each enterprise or individual has different preference in terms of technology that influence their level of trust towards cloud computing.

2.3.3 Trust levels in the cloud

Huang and Nicol, 2013 proposes several trust mechanisms and the aspect of trust they address. They are as below but not limited to the mentioned:

Reputation based trust – Typically, higher reputation translates into much trust by many entities in a society. The reputation of a CSP determines how cloud user consume the cloud services provided.

SLA verification-based trust – A good way for cloud service providers and cloud users to relate is to “Trust but verify”. Quality of Service (QoS) monitoring and SLA verification is critical in trust management for cloud computing.

Cloud transparency mechanisms – Cloud providers should use transparency and accountability to obtain trust. For instance, “Security, Trust and Assurance registry (STAR) launched by Cloud Security Alliance (CSA) as a program to be used freely by the public to broadcast security controls self-assessment for users to assess their security services.

Trust as a service – The same way we can have third party professionals manage SLA verification and QoS monitoring, RSA introduced the use of Cloud Trust Authority(CTA), a cloud service known as Trust as a service (TaaS) to deliver a single point use for configuration as well as management of security of cloud services offered by numerous providers.

Evidence based – Metrics such as performance could be measured and thus evidence provided fostering trust.

Others include opinion from peer users, attribute assessment and certification, statements from the cloud service provider, assessment of the cloud auditor among others

2.4 Related works

This section critically evaluates some of the related works that have been done in relation to trust on the multi-tenant clouds.

2.4.1 Chains of trust in the cloud

Huang & Nicol 2013 propose a model, which focuses on customer verification of services through third party professionals to foster trust through QoS monitoring and SLA verification. They suggest chain of trusts between the cloud provider, auditor, broker, cloud user and the cloud service as depicted in the figure 2:3 below. It does not give control to the end customer to monitor any aspects on their own as most of the control in relations to trust is bestowed on the cloud auditor,

such that as long as they have certified the cloud broker, cloud provider and the cloud service then the cloud user will also trust them. Architecture of the cloud might change quite often. This might mean that the audit done by the cloud auditor might need some changes as soon as the architecture changes. This is may not always be the case as most audits are done yearly or after any major changes.

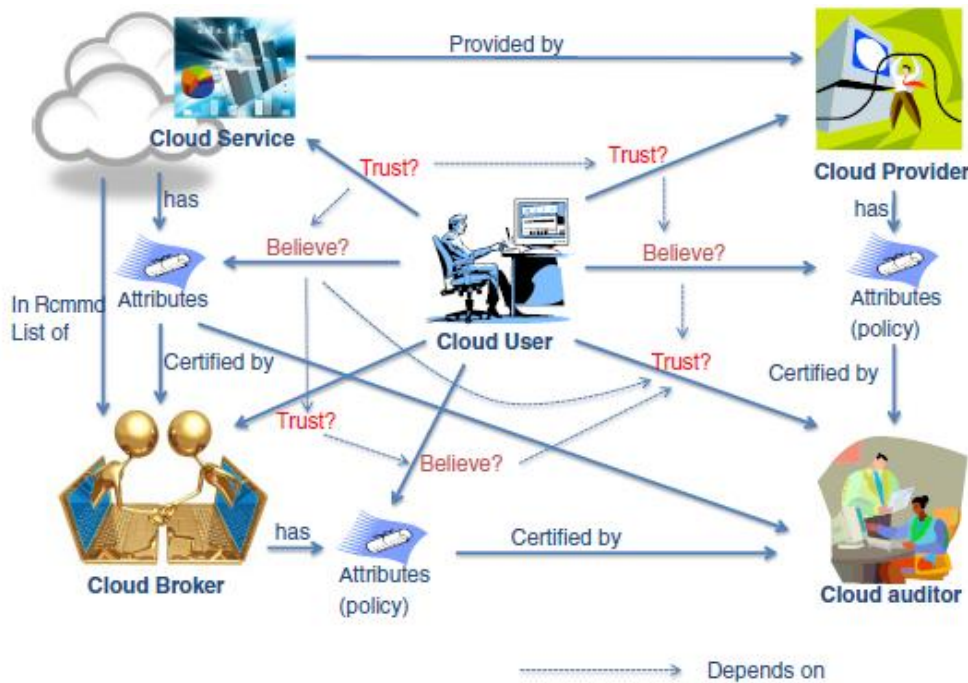


Figure 2: 3 Chains of trust in relation to cloud (Huang & Nicol 2013)

2.4.2 Cloud Computing Service Security Strength Measuring Trust Model

Shaikh & Sasikumar, 2015 proposes a trust model that measures the cloud security and establishes a trust value. It uses some considerations such as identity management, authorization, authentication, confidentiality among others to come up with the value. It only focuses on security on the cloud environment and the parameters are evaluated through interaction with the cloud environment. The trust value is consumed by customers to evaluate the cloud vendor they ought to purchase. This model does not give the customers some level of monitoring or a way of verifying the trust once they have acquired the cloud service.

2.4.3 Collaborative cloud services Authorization models in multi-tenancy environments

Tang et al 2015, in their research about collaborative cloud services authorization models in multi-tenancy environments suggest that trust between CSPs and cloud users is like the trust relations between organizations and their contracted outsourcing vendors. They identify three independent organizations namely the enterprise, the outsourcing company and the auditing firm whom are responsible for storage services, service coding and reporting respectively. They propose a mathematical model for authorization as a service. They do not seem to provide any tools employable to directly foster trust to the cloud consumers.

2.4.4 Trusted computing environment model (MTCEM)

As a counter measure to cloud security risks Brown et al 2012, discusses the proposed Multi-tenancy trusted computing environment model (MTCEM) by Li et al, 2010 that implements the trusted computing groups (TCG). According to Anderson, 2003 Trusted Computing Platform (TCP) a set of principles, standards and technologies that makes a data owner to trust as well as holding accountable the underlying computing infrastructure where applications that create, store and changes their data runs. TCP comprises two assertions discussed below, and the architecture shown in the figure 2.4 below:

Transitive trust – This suggests that computing platforms might only adjust from a Core Root of Trust Measurement (CRTM). This includes hardware or even encrypted firmware certified by a certified body of specialists and thus deemed trustworthy. Implicitly trust is implied such that one level of initialization trusts the previous.

Platform attestation – A computing policy displays to a third party that it is trusted. The systems trustworthiness is attested by the other systems it interacts with and thus in turn considered reliable by further systems. Main challenge is how to express conventional reasonable and quantifiable metrics useful to show how trustworthy the system is.

A critical look at the trusted computing (TC) model discussed above presents some limitations and has some drawbacks as presented by the internet community. Professor Anderson (University of Cambridge) claims that it is more of Information Technology (IT) industry than for people. It might give providers much power making them come up with unfair policies. Again, Stallman, 2018 the GNU project founder and Free Software Foundation (FSF) president, says that trusted

computing may expose free operating software as well as free application to a risk that users may not have the capability to run them anymore. Such criticisms raise some critical issues with trusted computing that may make it impossible to be implemented with actual technology.

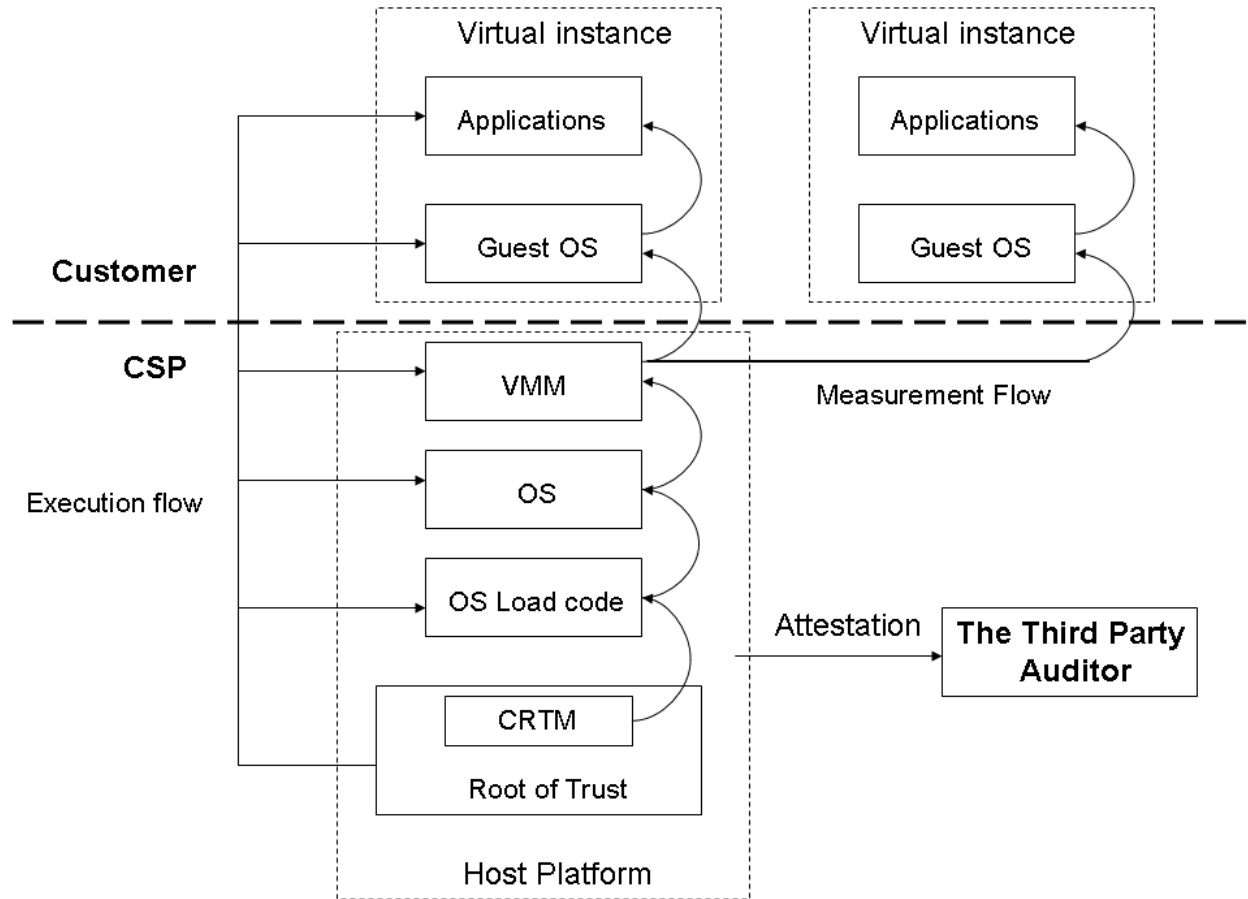


Figure 2: 4 MTCEM Model (Brown et al 2012).

2.4.5 Cross-tenant trust model (CTTM)

Tang & Sandhu, 2013 suggest a cross-tenant trust model (CTTM) in CC. The model consists of unilateral trust relations that reflect access control needs by two different tenants namely the trustor and the trustee. They suggest a multi-tenant authorization as a service (MTAaaS) to enable the implementation as shown in figure 2:5.

Their key contribution is proposing a cloud based MTAaaS where different tenants communicate to the MTAaaS platform with the use of application programming interface (API) which then gets the policies specific to the tenant and thus providing application centric security.

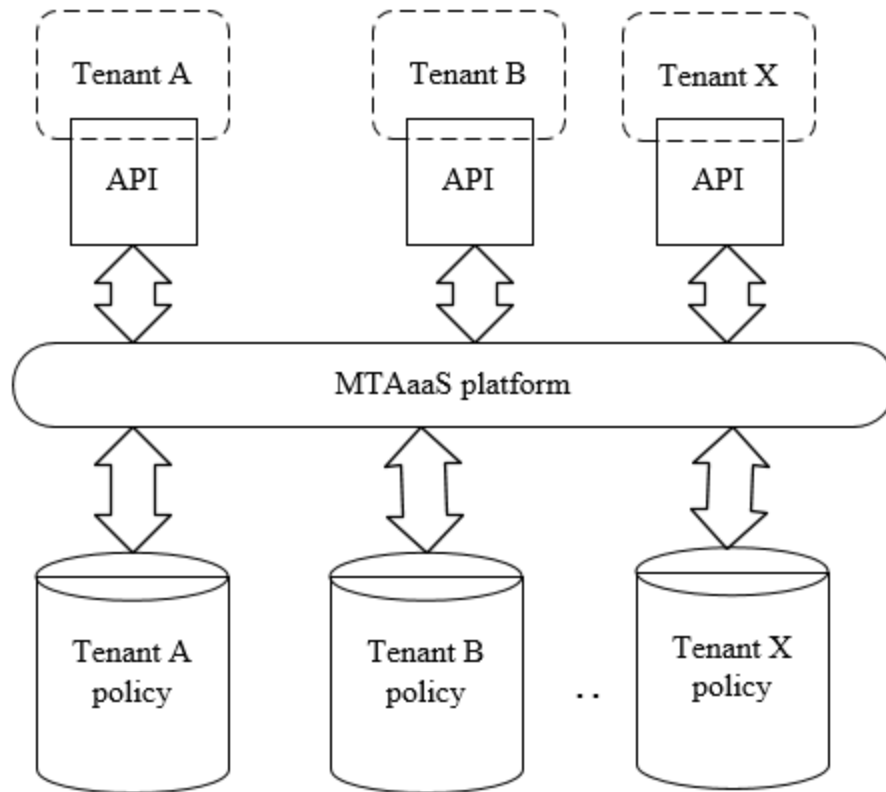


Figure 2: 5 MTAaaS architecture (Tang & Sandhu, 2013)

Their work mostly focusses on the trust between the different tenants. Different tenants in a multi-tenant architecture need not to know each other as a basic requirement, actualizing such a model could be difficult and does not contribute much trust between the CSP and the consumer perspective. Cloud hardware providers already provide abstraction between tenants that is a basic requirement for such architectures.

2.5 QoS Monitoring

According to Odun-Ayo, Ajayi, and Falade, 2018 provisioning of the appropriate resources to cloud workloads depends on the QoS requirements of such workloads. In the IaaS compute and storage resources are offered at a fee. The resources may include and not limited to CPU, memory, disk, storage, networks, and bandwidth among many others. Cloud consumers selects a cloud service that can offer services with adequate QoS guarantee.

From the above introduction, one way to define QoS could be stated the overall performance of a service such as cloud computing service. In cloud, QoS may entail the level of performance, availability, and reliability obtainable by an application, platform or the infrastructure that hosts

it. Sample QoS metrics or parameters of cloud SLA include availability, throughput, response time, memory utilizations, processing capacity among others.

2.6 Conceptual Model

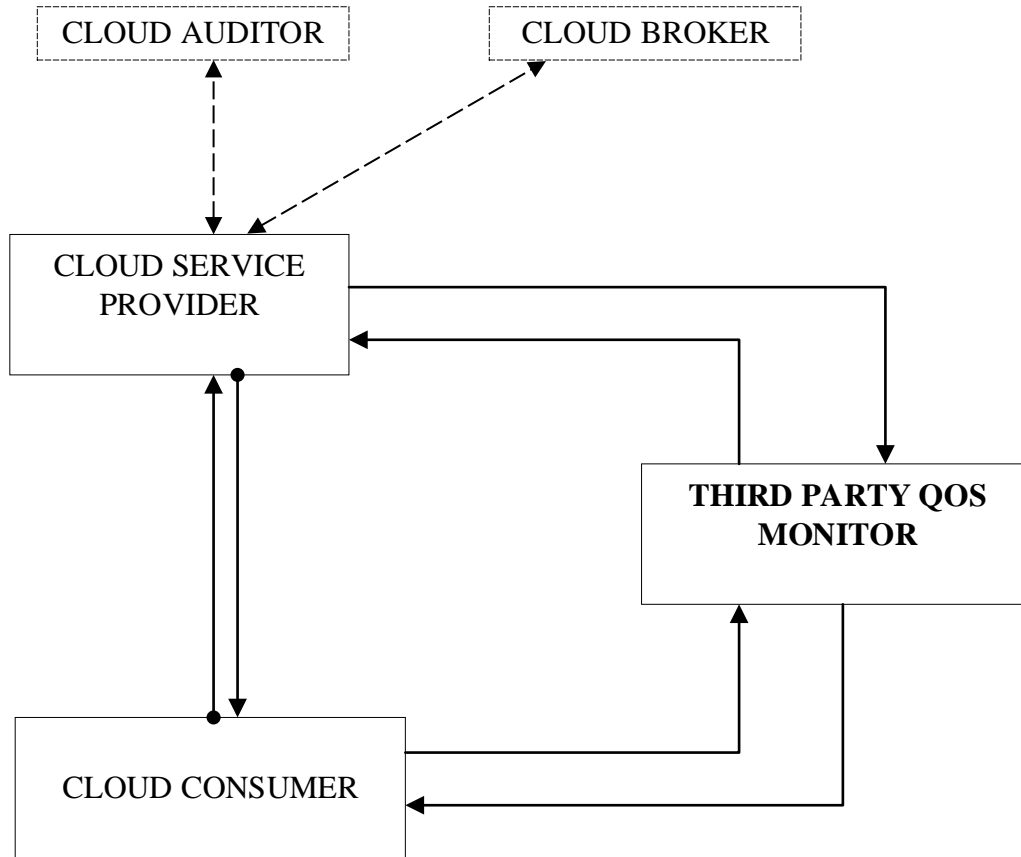


Figure 2: 6 Conceptual model

Figure 2.6 above represents the conceptual model of the model developed. In section 2.1.5, we discussed the cloud computing reference architecture model and the various actors involved. We had planned to introduce the third party QoS monitor who can keep watch of the cloud service provider platform in real time as compared to the cloud auditor who audits the cloud occasionally.

Cloud consumers wishing to join the cloud in addition to the audit reports done by the cloud auditor can benefit from the real time cloud overall performance from the QoS monitor. The third party QoS monitor is built by third party and connects to the cloud through secure, high through put direct connect link.

2.7 Summary

The literature was analyzed critically discussing what has been done so far and the gaps that exist. It was established that cloud computing consumes multi-tenant architecture and thus it is a key characteristic. Again, technically trust may not have a specific definition but has two key actors who include the trustor and the trustee. In addition, most cloud trust models developed do not seem to suggest third party QoS monitoring tools or parties who can carry QoS monitoring as a way of enhancing trust. The project was aimed at bridging that gap.

CHAPTER THREE: METHODOLOGY

According to Oates, 2005 this chapter discusses the strategies, methodological approach used, methods of data collection, analysis of the methods used as well as the justification of the methodological choice. In this case, methodology means the framework used to assemble, plan and direct the research process. It is evident that multiplicity of such frameworks formulated over the years, each has its own documented merits and demerits. Each single methodology designed to suit a specific kind of a project.

3.1 Research design strategy

Research design comprises the research strategy carried out which involves the study one is about to do. It also extends to justifying one's choice of research design. It further describes where to get data, population and even steps of the activities to carry out (Oates, 2005).

Exploratory research design is the overall research design strategy employed in this research. Delphi method is utilized to do data collection and analysis, population and sampling among other activities. The research design perfectly helps in exploring the concepts and techniques around trust in multi-tenancy clouds and thus able to come up with a well-thought and researched conceptual framework from where a model is developed.

3.2 Delphi method

According to Skinner et al, 2015 Delphi method was developed in the 1950's by the Rand Corporation. It is a methodical and interactive research procedure used to get opinion of a panel of independent experts in relation to a specific task. They highly recommend the use of this method while conducting information systems research more so for a qualitative research study, though it might not be suitable to all scenarios.

Rowe & Wright, 2001 suggest Delphi method to be most effective when statistical method is unfavorable, several experts are available, and then one can simply average the input of the several individuals or use a group. It is specifically appropriate for acquiring expert inputs while resolving an information systems related issue.

3.2.1 Characteristics of Delphi method

Skinner et al, 2015 suggest some generic characteristics that Delphi method should possess:

- i. Use of experts – To best fit the panel, an individual need to be technically knowledgeable and interested in their fields.
- ii. Panel – The panel should comprise of a group of selected experts without any size limitations
- iii. Anonymity – The individuals need not to beware of official position of the other panelist so that their opinions are not affected
- iv. Rounds – Execution is in a series of rounds with two rounds considered minimum. Three structured rounds are generally sufficient.
- v. Iteration and feedback – Analysis of opinions collected from the panelists is done and answers shared back for comments and/or basis for the next round.

3.3 Population and Sample

The population from which sample is drawn consist of cloud experts or individuals who have been supporting cloud consumers technically. This is because such individuals know how the multi-tenancy cloud works and due to the support, they do to the cloud consumers they know some pain points of these customers. A target of ten cloud experts either managing clouds directly or supporting cloud formed the population.

Delphi method does not conclusively define how many individuals should form a panel. Some researchers suggest a minimum of two individuals with a maximum of up to one hundred or even more. Generally, there is no accepted number of panelist or groups.

Purposive sampling was used in selection of these individuals as we needed to focus on their technicality and experience about cloud or using cloud. One panel consisting of five individuals was to be deemed enough.

3.4 Data Collection

This being a qualitative research and depending on the methodology the data collection technique consumed was interviews through open-ended questions and literature review. Since the

respondents must remain anonymous to each other to avoid bias then each was contacted independently.

The data collected, due to ethical issues is treated with uttermost confidentiality and only publishable or distributed to a third party upon permission from the data owner. The participants attended to the interview voluntarily. This means during the analysis the companies the respondents work for are not be disclosed.

3.5 Data Analysis

According to Oates (2005), data analysis idea is to look for patterns within data and draw conclusions. Data provided by the panelist was analyzed by comparing the common patterns among the various respondents. This helped in determining the QoS metrics to focus on as well as the best way to architect the model.

3.6 Prototype Design

According to Kim, 2019, a prototype can be described in two ways: As an original model of something that serves as a basis for other thing or as an early sample including the functions of tests, created to find a design solution. A prototype design of the proposed model is developed to describe our proposed model.

3.7 Tools

3.7.1 OpenStack cloud

It is an open source standard cloud computing platform mainly written in Python deployed in both private and public cloud mostly as infrastructure as a services (IaaS). It controls a large pool of compute, networking as well as storage resources and is manageable through dashboard or OpenStack API. It works well with other open source and enterprise technologies making it ideal for heterogeneous infrastructure. Many world's largest brands use OpenStack to run their businesses. Some of the platinum members include AT&T, Ericsson, Huawei, Intel, Rackspace, Redhat, Suse and Tencent cloud. This makes it to be the most favorable cloud platform for this project. More information can be found on the following URL: <https://www.openstack.org/>.

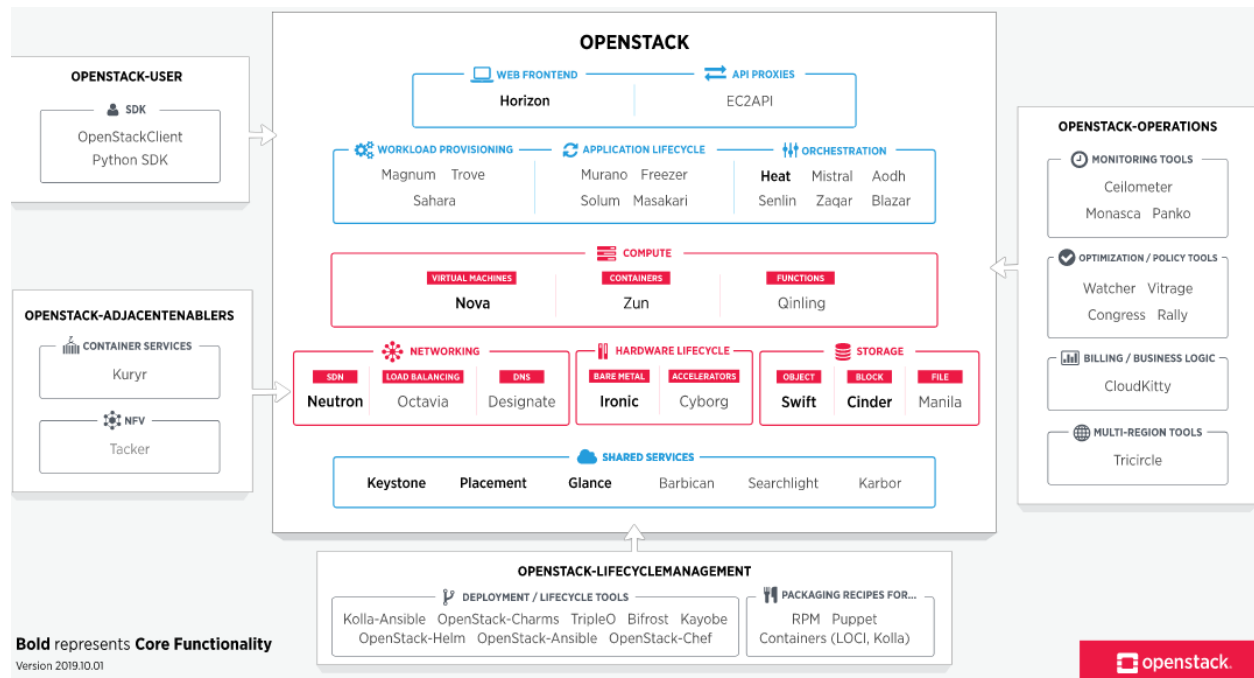


Figure 3: 1 OpenStack landscape.

As per the above figure 3.1 OpenStack has several components, which we shall consume in this project. The components to be utilized include:

Nova – Used for compute service which handle the virtual machines

Swift – Used for object storage

Cinder – Used for block storage

Neutron – used for networking

Keystone – used for identity service

Glance – used for image service

Horizon – used for dashboard

3.7.2 Prometheus

Prometheus is an open source monitoring and alerting toolkit maintained independently of any company. Most components are built with Go, making them easy to build and deploy as static

binaries. It works best in recording any purely numeric time series and fits both machine-centric monitoring as well as monitoring of highly dynamic service-oriented architecture.

Prometheus Architecture (source: <https://prometheus.io/docs/introduction/overview/>)

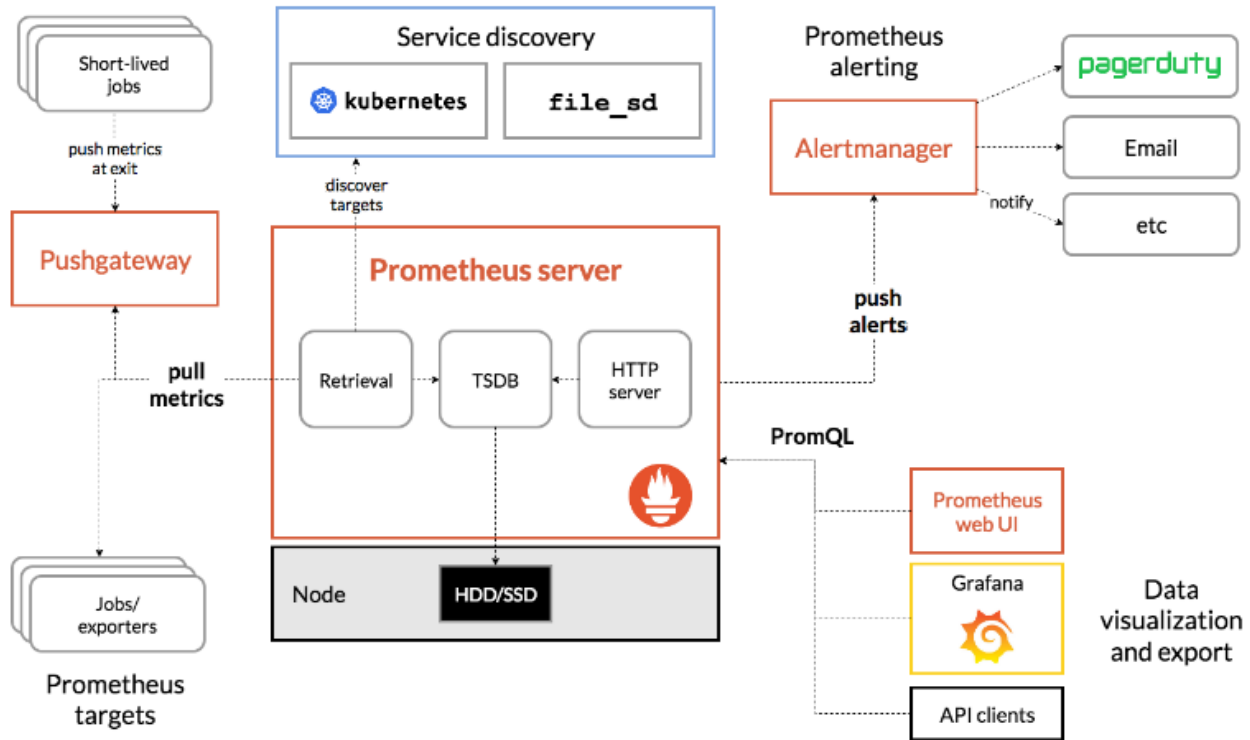


Figure 3: 2 Prometheus Architecture

As per the above figure 3.2 Prometheus consist of several optional components as discussed below

Prometheus server – Scarp and stores times series data

Push gateway – for supporting short-lived jobs

Alert manager – Handles alerts

Special service exporters – for services like HA Proxy, StatsD, and Graphite among others.

It uses PromQL – A flexible query language to leverage dimensionality.

On our project, we shall consume Prometheus to act as the broker where external third parties connect to and hence acts as the endpoint. More information about Prometheus can be found in the following url <https://prometheus.io/>.

3.7.3 Grafana

Grafana is multi-platform and open source analytics and interactive visualization software. Written in Go programming language, it provides charts, graphs and alerts for web when connected to supported data sources. In our prototype, Grafana will be used as the third party QoS monitor with the capabilities of visualizing metrics from the OpenStack cloud and sharing the same with the cloud consumer in a way they can understand.

3.8 Conclusion and justification of the methodology

Cloud Multi-tenancy being the underlying architecture for cloud computing and with limited research done on trust specifically in Kenya, the use of Delphi methodology fits in. This enabled us to get expert opinions in the relation to cloud status in Kenya from industry practitioners. As a result, we were able to come up with a realistic model, which can benefit both the cloud consumer and the cloud provider in building a stronger relationship.

CHAPTER FOUR: RESULTS AND DISCUSSIONS

In this chapter, we describe the system prototype in detail. We discuss the interconnection between the various components used, how it works and how it brings in the trust model aspect. Most tools used are open source and hence we can use them for research and even modify them to fit the purpose we intend to.

4.1 Data analysis

From the data collected through the open-minded interview sessions with the panelists several conclusions were drawn. The first round of interview was conducted before the prototype development began. It sought to establish cloud consumers pain points in term of QoS monitoring and acquisition of cloud services, institutions majorly consuming cloud services hosted locally, the best ways in which the trust model could be designed and if any of them knew existence of such a trust model in Kenya. The general conclusions drawn were as below:

- i. Most of the cloud customers consuming cloud services within Kenya are mostly institutions, which do not want their data to go outside the country. This is majorly contributed by policies within such institutions. This was according to four out of six respondents.
- ii. According to two out of three panelist who were major cloud consumers, such customers in (i) choose to remain with a certain cloud consumer despite poor services as there were no enough options in Kenya or they lacked insights about the other companies and feared they might experience the issues as with their current provider.
- iii. Most cloud consumers lack visibility of their cloud service provider platform, making it difficult to make some decisions.
- iv. None of the individuals interviewed knew of any existing platform in any Kenya's CSP where they could verify the QoS metrics using a third party QoS monitor.
- v. Three of the cloud consumers said that, such a trust model would be very beneficial in helping cloud consumers make some decisions related to cloud hosting services.
- vi. They also expressed that CSP might not want direct connection to their cloud platform by third parties as it might be a way of exposing them.

4.2 The overall prototype architecture of the proposed model

We shall start by discussing the overall prototype architecture of the proposed model. How data flow from the cloud service provider to the cloud consumer. We shall also put into perspective how the various tools used are utilized to accomplish the solution. We shall refer to the below figure 4.1.

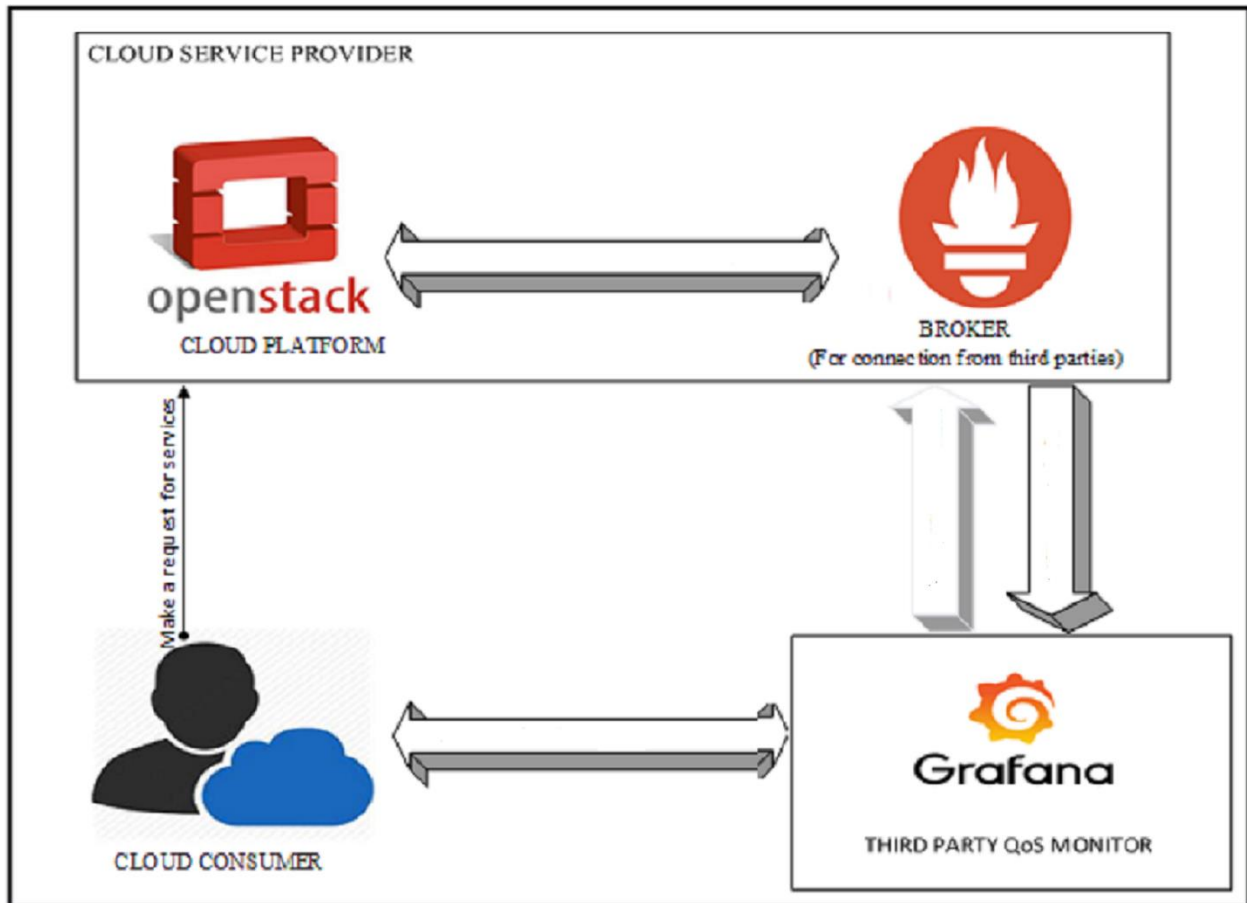


Figure 4: 1 The overall prototype architecture of the proposed model

At the cloud service provider, OpenStack cloud platform is set up. It serves as an infrastructure as service cloud model platform. A virtual data center (vDC) is created for cloud consumers from where they can then create their own virtual machines, images as well as their own virtual private clouds (VPC). In addition, a broker is set up where third party QoS monitors can connect to avoiding direct connection to the core platform. The broker serves the sole purpose of mediating between the cloud platform and external networks. The broker has a module installed on the cloud

platform from where it scraps metrics. The module is referred to as node exporter. The tool used to work as the broker is Prometheus.

The third party QoS monitor once allowed connects to the cloud service broker through a secure connection by use of application programming interfaces (APIs) and can get the metrics scraped from the cloud platform by the broker. The tool used to represent the third party QoS monitor in this project is Grafana. It has the capability of visualizing the cloud platform status by displaying the various metrics. This is then shared with the cloud consumers. Various ways can be used to share the data such as: provisioning them and giving them a web portal to log into or connecting through APIs or manually share the report with them inform of excel pdf or any other formats.

The cloud consumer uses the information from the third party QoS monitor to make trustworthy decisions related to service level agreements with the cloud provider. If allowed they could also connect to the third party QoS monitor with APIs.

4.3 The cloud service provider environment

This part will discuss the cloud platform in depth by describing how and where it has been setup, resources allocated, modules installed and how it collects data and shares with the third party QoS monitor.

4.3.1 The Cloud Platform (OpenStack)

This is the most crucial part of the setup as it where the metrics are scraped from and it is the part where the cloud services are being hosted. It is installed in Ubuntu 18.08 LTS server. As per the below figure 4.2 it accessible over the internet through a web browser. Once you insert your username and password it takes you to your panel from where you can manage your resources.

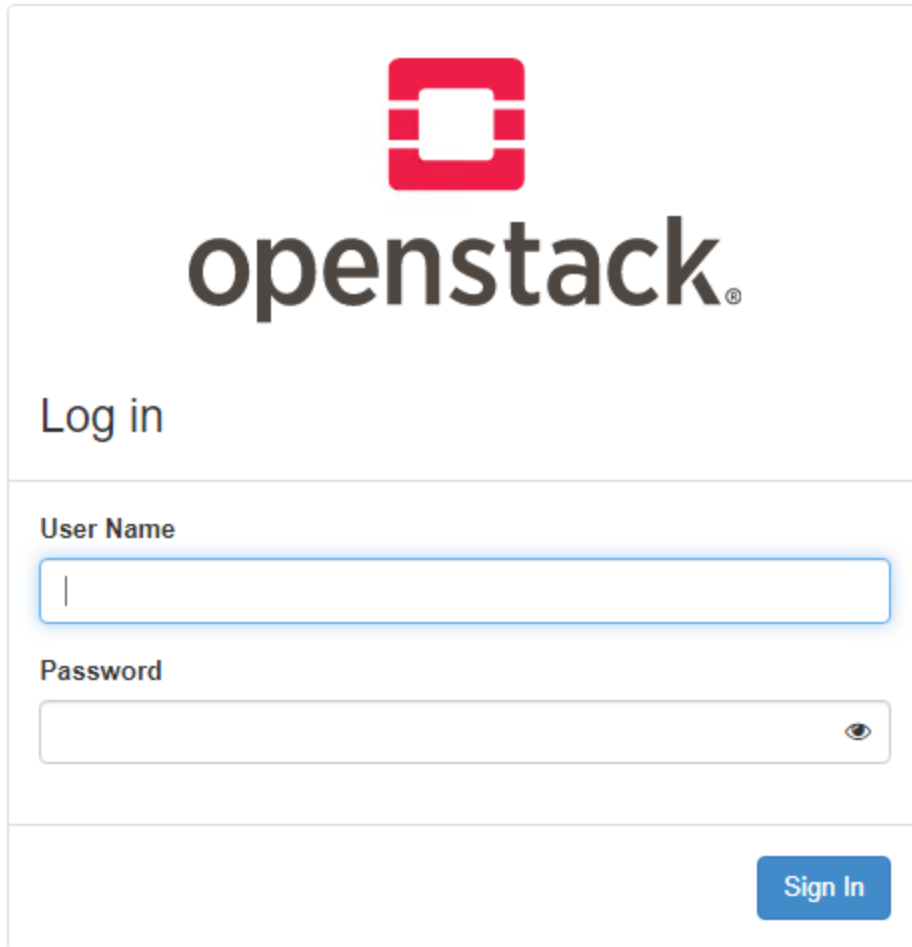
The image shows the OpenStack login graphical user interface (GUI). At the top center is the OpenStack logo, a red square with a white square inside, and the word "openstack" in a bold, black, sans-serif font. Below the logo is the text "Log in". Underneath, there are two input fields: "User Name" and "Password". The "User Name" field is a simple text box with a vertical cursor. The "Password" field is a text box with a small eye icon on the right side, indicating a password toggle. At the bottom right of the form is a blue button with the text "Sign In".

Figure 4: 2 The OpenStack login graphic user interface (GUI)

4.3.2 Actual provisioned resources for the OpenStack cloud.

The actual resources provisioned for use by the platform is as below:

Hard disk – 393 GB (Partition where OpenStack is installed)

RAM – 31 GB

CPU – 16

This are the most crucial metrics, as we shall focus on them to derive crucial decisions in our trust model. Figure 4.3 below show the hard disk size and portioning done. The total amount of allocated space is 450 GB.


```

stack@openstack:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            16G   0    16G   0% /dev
tmpfs           3.2G  1.4M  3.2G   1% /run
/dev/sda2       393G  35G  338G  10% /
tmpfs           16G   0    16G   0% /dev/shm
tmpfs           5.0M   0   5.0M   0% /run/lock
tmpfs           16G   0    16G   0% /sys/fs/cgroup
/dev/loop0      94M   94M   0 100% /snap/core/8935
/dev/loop1      94M   94M   0 100% /snap/core/9066
tmpfs           3.2G   0   3.2G   0% /run/user/1000
stack@openstack:~$ █

```

Figure 4: 3 Allocated disk space and partitioning.

The figure below 4.4 show the allocated CPU, model and architecture of the hardware.

```

stack@openstack:~$ lscpu
Architecture:      x86_64
CPU op-mode(s):   32-bit, 64-bit
Byte Order:       Little Endian
CPU(s):           16
On-line CPU(s) list: 0-15
Thread(s) per core: 1
Core(s) per socket: 2
Socket(s):        8
NUMA node(s):     8
Vendor ID:        GenuineIntel
CPU family:        6
Model:            62
Model name:       Intel(R) Xeon(R) CPU E5-4650 v2 @ 2.40GHz
Stepping:         4
CPU MHz:          2400.000
BogoMIPS:         4800.00
Hypervisor vendor: VMware
Virtualization type: full
L1d cache:       32K
L1i cache:       32K
L2 cache:        256K
L3 cache:        25600K
NUMA node0 CPU(s): 0,1
NUMA node1 CPU(s): 2,3
NUMA node2 CPU(s): 4,5
NUMA node3 CPU(s): 6,7
NUMA node4 CPU(s): 8,9
NUMA node5 CPU(s): 10,11
NUMA node6 CPU(s): 12,13
NUMA node7 CPU(s): 14,15
Flags:            fpu vme de pse tsc msr pae mce cx8 apic sep mtr:
topology tsc_reliable nonstop_tsc cpuid aperfmperf pni pclmulqdq sss
stack@openstack:~$ █

```

Figure 4: 4 Allocated CPU.

The allocated RAM is shown in the below figure 4.5

```
stack@openstack:~$ top
top - 19:03:28 up 46 min,  1 user,  load average: 1.64, 1.43, 1.09
Tasks: 388 total,  1 running, 250 sleeping,  0 stopped,  0 zombie
%Cpu(s):  1.8 us,  1.1 sy,  0.0 ni, 96.2 id,  0.8 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 32937268 total, 25072268 free,  6691044 used, 1173956 buff/cache
KiB Swap: 8388604 total, 8388604 free,    0 used. 25529684 avail Mem
```

Figure 4: 5 Allocated RAM

4.3.3 OpenStack modules installed

These are the modules described in section 3.7.1. This part shall focus on the key modules installed in our environment. The folders containing the installed modules can be shown the below figure 4.6. If one needs to modify the configuration related to a certain module then they navigate to the associated folder.

```
stack@openstack:~$ ll
total 108
drwxr-xr-x 19 stack stack 4096 Mar 16 17:13 ./
drwxr-xr-x  3 root  root  4096 Mar 12 16:56 ../
-rw-----  1 stack stack  175 Apr 13 23:41 .bash_history
-rw-r--r--  1 stack stack   220 Apr  4 2018 .bash_logout
-rw-r--r--  1 stack stack 3771 Apr  4 2018 .bashrc
drwxr-xr-x  2 root  root  4096 Mar 12 17:16 bin/
drwxr-xr-x  4 stack stack 4096 Mar 12 17:14 bindep-venv/
drwxr-xr-x  3 stack stack 4096 Mar 12 17:13 .cache/
drwxr-xr-x 13 stack stack 4096 Mar 12 17:23 cinder/
drwxr-xr-x  8 stack root  4096 Mar 12 18:02 data/
drwxrwxr-x 16 stack stack 4096 Apr 13 23:35 devstack/
-rw-r--r--  1 stack stack   44 Mar 12 18:04 devstack.subunit
drwxr-xr-x 13 stack stack 4096 Mar 12 17:21 glance/
drwxr-xr-x 14 stack stack 4096 Mar 12 17:44 horizon/
drwxr-xr-x 17 stack stack 4096 Mar 12 17:18 keystone/
drwxr-xr-x  2 stack stack 4096 Mar 12 17:35 logs/
-rw-----  1 stack stack   52 Mar 12 17:15 .my.cnf
drwxr-xr-x 16 stack stack 4096 Mar 12 17:25 neutron/
drwxr-xr-x 15 stack stack 4096 Mar 12 17:30 nova/
drwxr-xr-x  4 stack stack 4096 Mar 12 18:08 .novaclient/
drwxr-xr-x 10 stack stack 4096 Mar 12 17:28 noVNC/
drwxr-xr-x 13 stack stack 4096 Mar 12 17:32 placement/
-rw-r--r--  1 stack stack  807 Apr  4 2018 .profile
drwxr-xr-x 10 stack stack 4096 Mar 12 18:01 requirements/
drwxr-xr-x 12 stack stack 4096 Mar 12 18:04 tempest/
-rw-----  1 root  root  1892 Mar 12 17:05 .viminfo
-rw-r--r--  1 stack stack  165 Mar 12 17:56 .wget-hsts
stack@openstack:~$
```

Figure 4: 6 OpenStack modules installed.

Below are few module snapshots at work.

i. Horizon (Dashboards)

As shown in the below figure 4.7 this is the module concerned with the graphical user interface. It is used to create the specific dashboards.

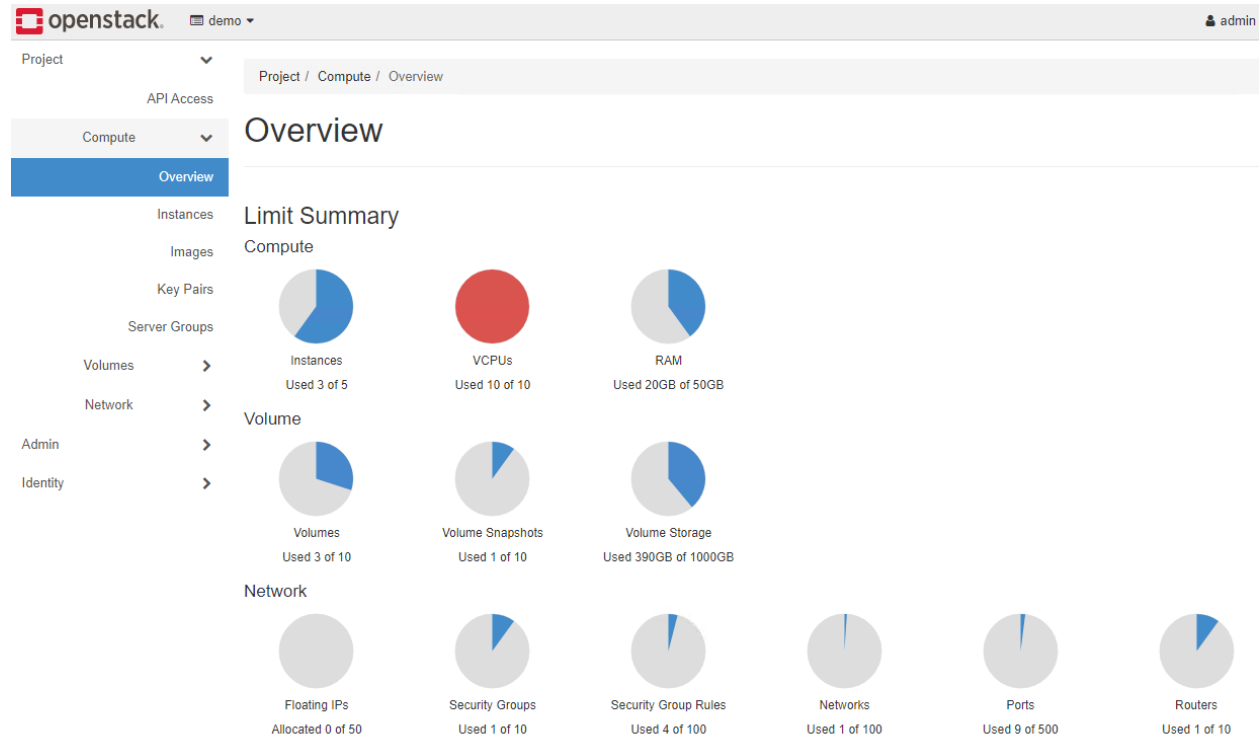


Figure 4: 7 Sample OpenStack dashboard created.

ii. Nova (Compute services)

This is the module used to handle compute services like virtual machines as seen in the below figure 4.8. In our case the demo user has three instances (virtual machines created).

The screenshot shows the OpenStack 'Instances' page. The left sidebar contains navigation options: Project, API Access, Compute, Overview, Instances (selected), Images, Key Pairs, Server Groups, Volumes, Network, Admin, and Identity. The main content area displays a table of instances with columns: Instance Name, Image Name, IP Address, Flavor, Key Pair, Status, Availability Zone, Task, Power State, Age, and Actions. Three instances are listed:

| Instance Name | Image Name | IP Address | Flavor | Key Pair | Status | Availability Zone | Task | Power State | Age | Actions |
|-------------------------------|--------------------------|--|-----------|----------|---------|-------------------|------|-------------|------------------|----------------|
| pascal-dem-o-ubuntu | pascal-ubuntu | shared 192.168.233.137 private 10.0.0.12, fd44:f22d:a041:0:f816:3eff:fe1d:6445 | m1.small | - | Shutoff | nova | None | Shut Down | 1 month, 2 weeks | Start Instance |
| cirros-pascal-test-prometheus | cirros-0.4.0-x86_64-disk | private 10.0.0.25, fd44:f22d:a041:0:f816:3eff:fe46:baa8 shared 192.168.233.122 | m1.xlarge | - | Shutoff | nova | None | Shut Down | 1 month, 3 weeks | Start Instance |
| cirros-demo | cirros-0.4.0-x86_64-disk | private 10.0.0.40, fd44:f22d:a041:0:f816:3eff:fe46:3384 shared 192.168.233.160 | m1.small | - | Shutoff | nova | None | Shut Down | 1 month, 3 weeks | Start Instance |

Figure 4: 8 Sample virtual machines created.

iii. Neutron (Networks)

This is module used to create the networking components such as private or public networks. As per the figure, 4.9 below three networks have been created which consist private, shared and public subnets.

The screenshot shows the OpenStack 'Networks' page. The left sidebar contains navigation options: Project, API Access, Compute, Volumes, Network (selected), Network Topology, Networks (selected), Routers, Security Groups, Floating IPs, Admin, and Identity. The main content area displays a table of networks with columns: Name, Subnets Associated, Shared, External, Status, Admin State, Availability Zones, and Actions. Three networks are listed:

| Name | Subnets Associated | Shared | External | Status | Admin State | Availability Zones | Actions |
|---------|---|--------|----------|--------|-------------|--------------------|--------------|
| private | ipv6-private-subnet fd44:f22d:a041::/64 private-subnet 10.0.0.0/26 | No | No | Active | UP | nova | Edit Network |
| shared | shared-subnet 192.168.233.0/24 | Yes | No | Active | UP | nova | Edit Network |
| public | public-subnet 172.24.4.0/24 ipv6-public-subnet 2001:db8::/64 | No | Yes | Active | UP | nova | Edit Network |

Figure 4: 9 Created Networks

iv. Keystone (identity service)

This the identity service which deals with creation of users as well authentication and authorization. It is the enables a user to be directed to their project or virtual data center while they

login in through the GUI. It also controls the API integrations. Figure 4.10 shows various users created.

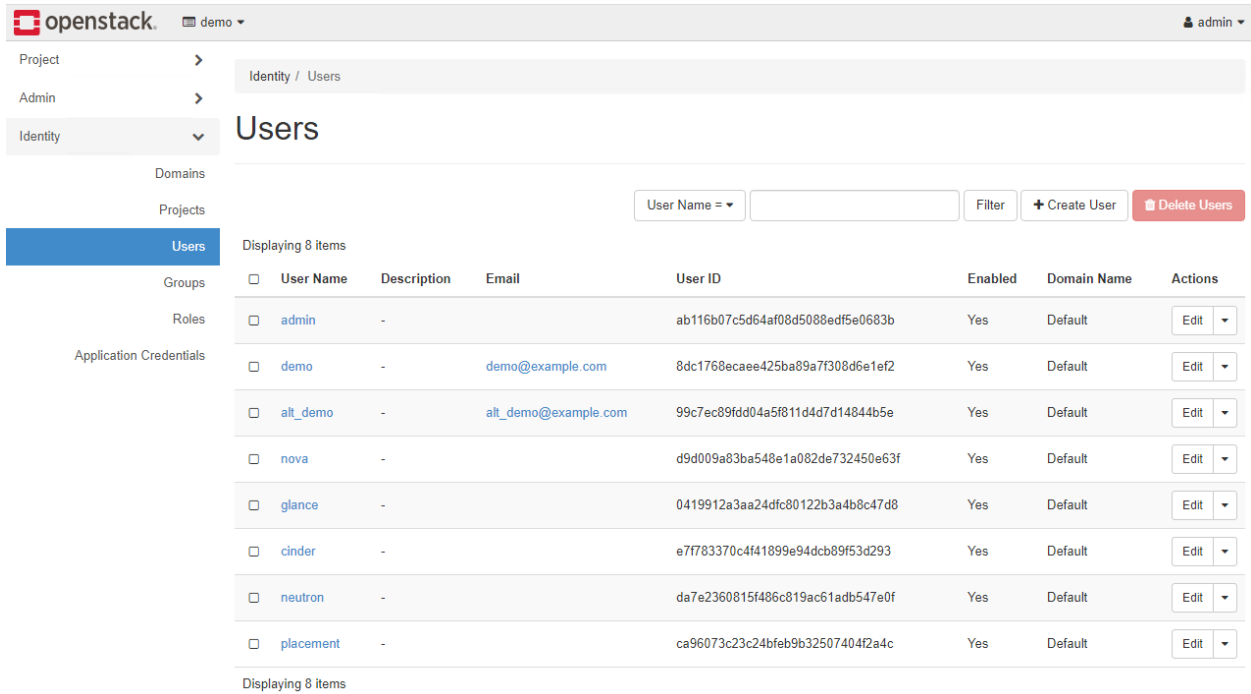


Figure 4: 10 Identity service

v. Glance and Swift (Images and Storage)

Swift is used for object storage while Glance is used for images services. As seen the below figure 4.11 we were able to create images from which instances could be deployed and storage done in the shown volumes.

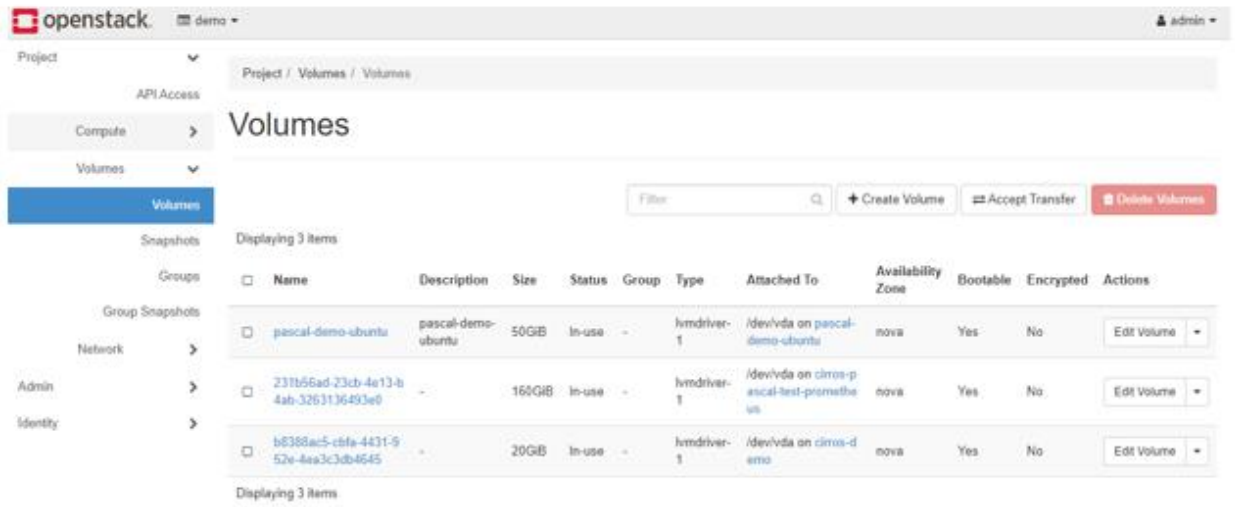
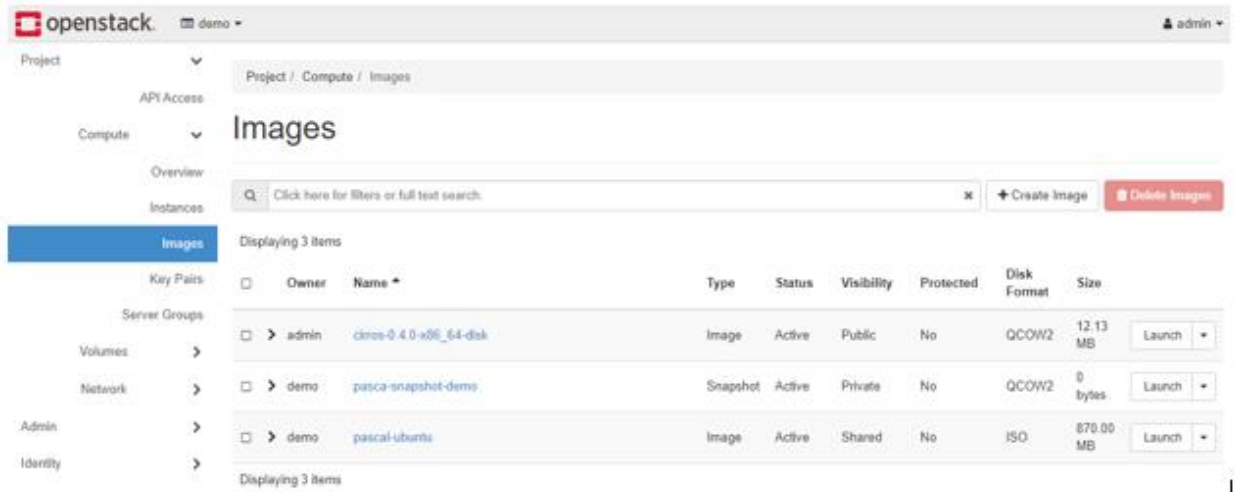


Figure 4: 11 OpenStack images and storage

4.3.4 Broker

The broker is where the third party QoS monitor connects. This acts like an abstraction layer preventing direct connections to the cloud platform. In our case, Prometheus has been setup in a containerized environment as shown in figure 4.12. It run on Ubuntu server. Prometheus has a component installed in the cloud platform that it uses to scrap metrics. The below figure 4.13 shows the Prometheus run time and build information.

```
[root@webservices ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS                               NAMES
620de104163b      grafana/grafana    "/run.sh"          7 weeks ago        Up 7 weeks         0.0.0.0:3000->3000/tcp              grafana
47f13493f683      ubuntu:14.04      "/bin/bash"        7 weeks ago        Up 7 weeks         0.0.0.0:9090->9090/tcp              prometheus1
[root@webservices ~]#
```

Figure 4: 12 Prometheus docker container

Runtime Information

| | |
|---------------|---|
| Uptime | 2020-03-16 16:24:41.650217265 +0000 UTC |
|---------------|---|

Build Information

| | |
|------------------|--|
| Version | 1.3.1 |
| Revision | be476954e80349cb7ec3ba6a3247cd712189dfcb |
| Branch | master |
| BuildUser | root@37f0aa346b26 |
| BuildDate | 20161104-20:24:03 |
| GoVersion | go1.7.3 |

Figure 4: 13 Prometheus runtime and build information

4.3.5 Node exporter

This is an application packaged and running as a service on the OpenStack platform as shown in the below figure 4.14. Node exporter is part of the broker and which makes it hard for any manipulations to be done within the cloud platform and the broker.

```
stack@openstack:~$ id node_exporter
uid=999(node_exporter) gid=999(node_exporter) groups=999(node_exporter)
stack@openstack:~$ systemctl status node_exporter.service
â node_exporter.service - Prometheus Node Exporter
   Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; vendor
   Active: active (running) since Wed 2020-05-06 18:17:49 EAT; 1h 31min ago
   Main PID: 1691 (node_exporter)
   Tasks: 35 (limit: 4915)
   CGroup: /system.slice/node_exporter.service
           ââ1691 /usr/local/bin/node_exporter
stack@openstack:~$ ss -altnp | grep 91
LISTEN 0      100          0.0.0.0:8775          0.0.0.0:*
LISTEN 0      100          127.0.0.1:32969     0.0.0.0:*
fd=4), ("uwsgi",pid=1091,fd=4))
LISTEN 0      128          *:9100              **
stack@openstack:~$
```

Figure 4: 14 Node exporter.

4.4 Third Party QoS Monitor

This forms the intermediary between the cloud service provider and the cloud consumer. Cloud consumers can verify the status of the CSP cloud platform through the third party QoS monitor. It is installed in Ubuntu server 18.04 LTS server and is set up in third parties' environment. It connects to the broker from where in can request and get metrics of the cloud platform. To display this Grafana is used and displays the actual metrics of the cloud platform as shown in the figure 4.15 below.



Figure 4: 15 visualization of the cloud platform metrics.

Different metrics can be visualized using the tool such as CPU, memory, disk utilization as well as availability as shown in the below figure 4.16

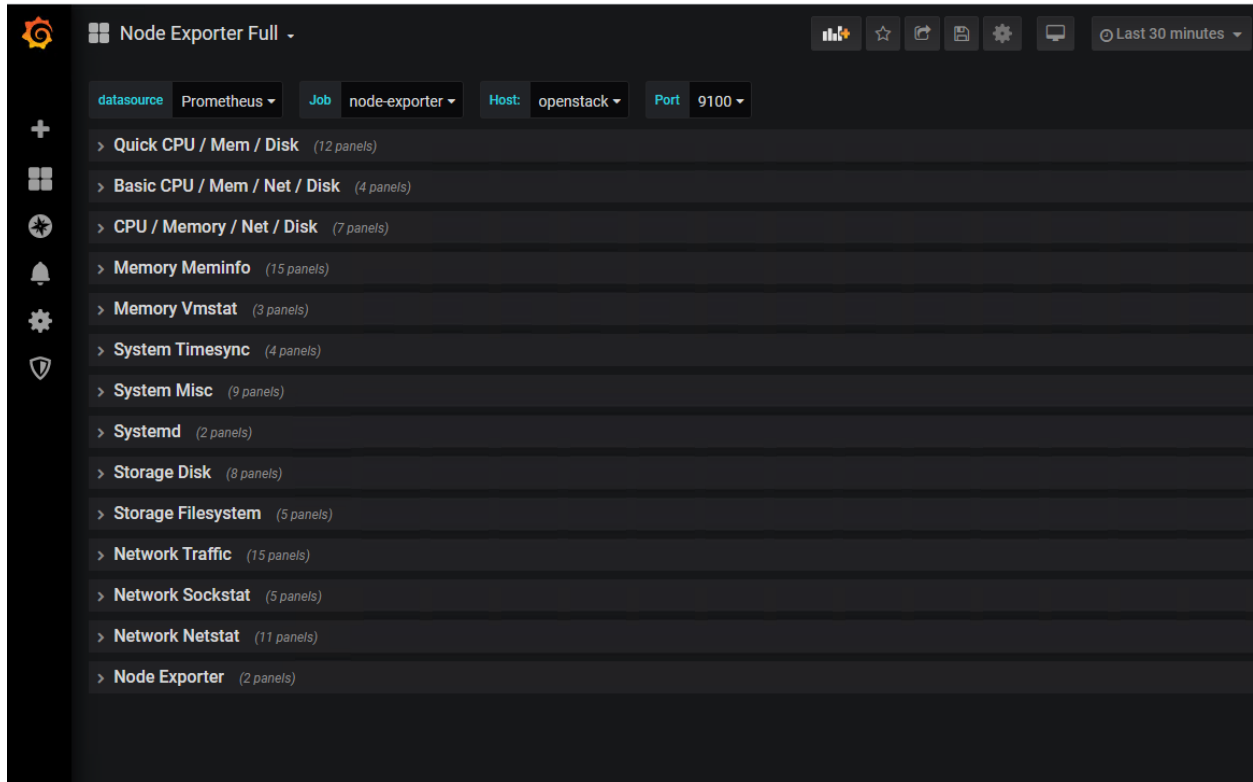


Figure 4: 16 Different metrics of the cloud platform

4.3 The cloud consumer

Once provisioned by the third party QoS monitor, cloud consumers can view the cloud status through a web interface. Through this, it is possible for cloud consumers to verify the status of a certain metrics of the cloud even before they purchase and hence boosting their trust and confidence. Figure 4.17 below show a sample cloud consumer view.

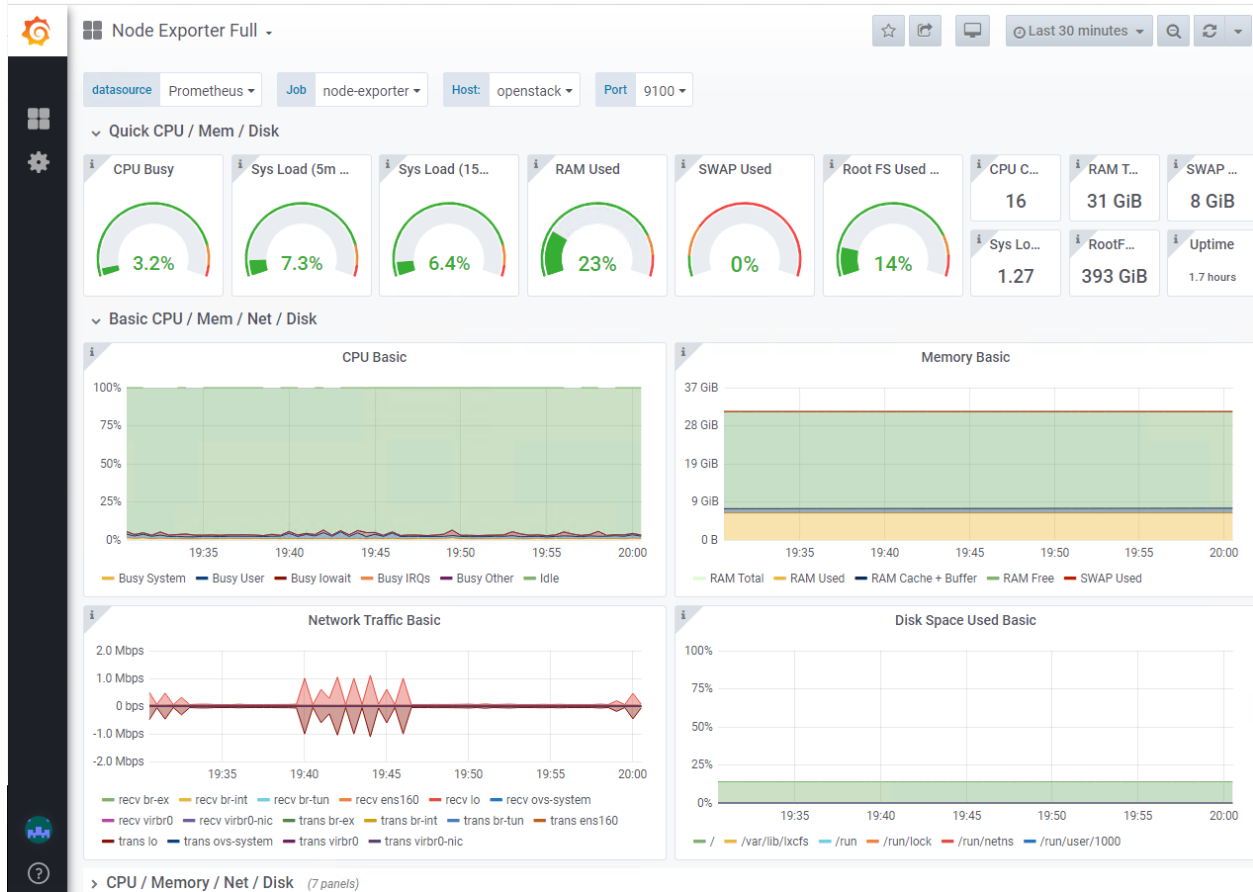


Figure 4: 17 Sample cloud consumer view

4.4 How the prototype works - End to End Integration of the components

4.4.1 Low-level design

Figure 4.18 below represents the actual low-level diagram of the developed prototype with the actual connections. It clearly depicts how the different components integrate with each other from the cloud service provider to the cloud consumer. The ports used can be change to any other so long as the right configuration is done.

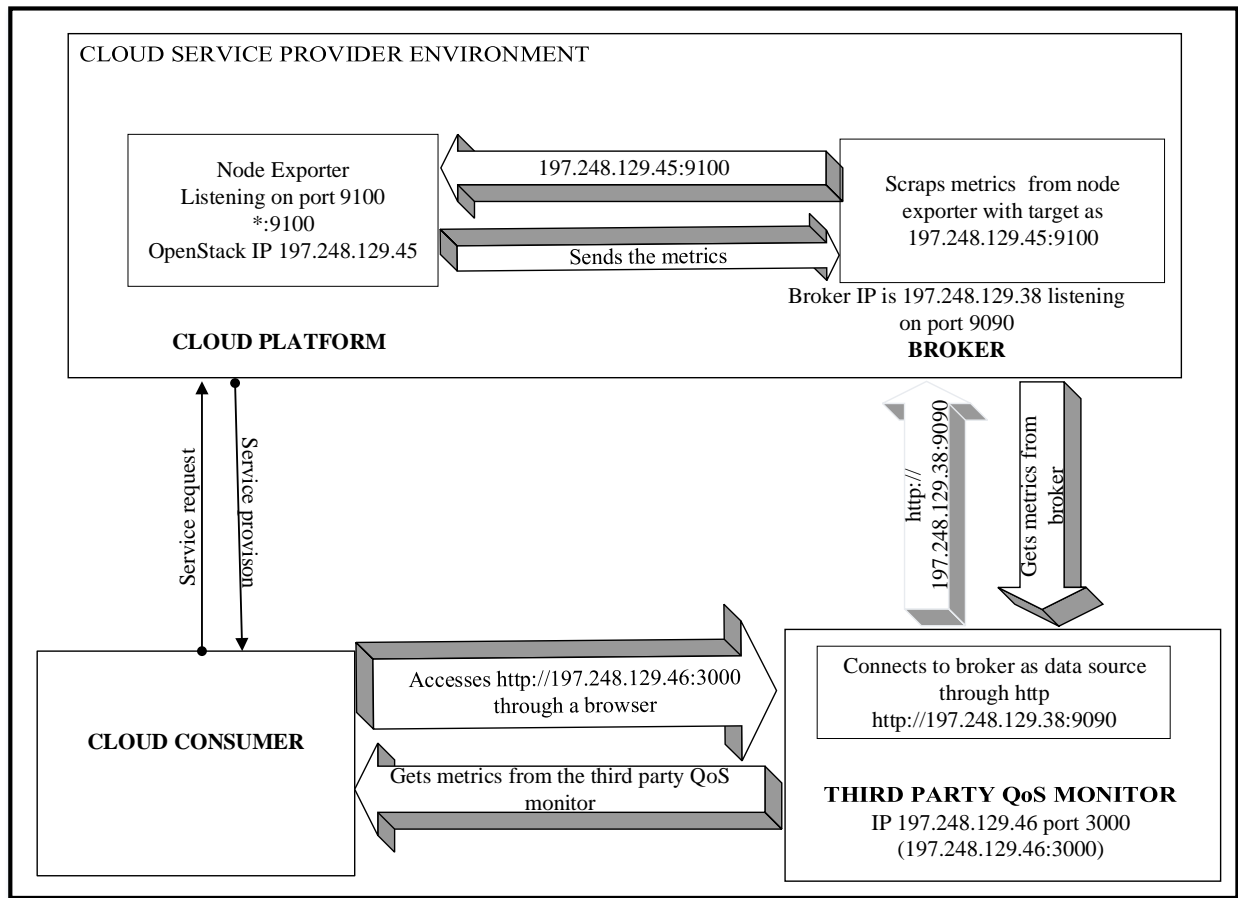


Figure 4: 18 Low-level diagram of how the prototype works

4.4.2 Integration between node exporter and broker

The below figure 4.19 shows the exact service status running on the cloud platform together with the port it is listening to.

```

stack@openstack:~$ id node_exporter
uid=999(node_exporter) gid=999(node_exporter) groups=999(node_exporter)
stack@openstack:~$ systemctl status node_exporter.service
● node_exporter.service - Prometheus Node Exporter
   Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; vendor
   Active: active (running) since Wed 2020-05-06 18:17:49 EAT; 1h 31min ago
   Main PID: 1691 (node_exporter)
     Tasks: 35 (limit: 4915)
    CGroup: /system.slice/node_exporter.service
            └─1691 /usr/local/bin/node_exporter
stack@openstack:~$ ss -altnp | grep 91
LISTEN 0      100          *:*          0.0.0.0:8775      0.0.0.0:*
LISTEN 0      100          *:*          127.0.0.1:32969   0.0.0.0:*
fd=4), ("uwsgi",pid=1091,fd=4)
LISTEN 0      128          *:*          *:9100           *:*
stack@openstack:~$

```

Figure 4: 19 Node exporter service status.

The figure 4.20 below show how the broker (Prometheus in our case) connects to the node exporter to scrap the metrics.

```
root@47f13493f683:/prometheus-1.3.1.linux-amd64# ll
total 71068
drwxrwxr-x.  5 1000 1000    4096 Mar 17 13:44 ./
drwxr-xr-x. 22 root root    4096 Mar 12 20:18 ../
-rw-rw-r--.  1 1000 1000   11357 Nov  4 2016 LICENSE
-rw-rw-r--.  1 1000 1000    2552 Nov  4 2016 NOTICE
drwxrwxr-x.  2 1000 1000     38 Nov  4 2016 console_libraries/
drwxrwxr-x.  2 1000 1000    4096 Nov  4 2016 consoles/
drwx-----. 262 root root    8192 May  6 20:21 data/
-rwxr-xr-x.  1 1000 1000 62829079 Nov  4 2016 prometheus*
-rw-r--r--.  1 root root    1282 Mar 16 17:20 prometheus.yml
-rw-r--r--.  1 root root    1058 Mar 12 20:21 prometheus.yml.original
-rwxr-xr-x.  1 1000 1000 9890020 Nov  4 2016 promtool*
root@47f13493f683:/prometheus-1.3.1.linux-amd64# vi prometheus.yml
# my global config
global:
  scrape_interval:      15s # By default, scrape targets every 15 seconds.
  evaluation_interval: 15s # By default, scrape targets every 15 seconds.
  # scrape_timeout is set to the global default (10s).

  # Attach these labels to any time series or alerts when communicating with
  # external systems (federation, remote storage, Alertmanager).
  external_labels:
    monitor: 'codelab-monitor'

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first.rules"
  # - "second.rules"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  #PAS - job_name: 'prometheus'

    # Override the global default and scrape targets from this job every 5 seconds.
    #PASscrape_interval: 5s

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

  #PAS static_configs:
    #PAS- targets: ['localhost:9090']

# 197.248.129.45 (node exporter) replace with .47 to pull from the node exporter only

- job_name: 'node-exporter'
  scrape_interval: 5s
  static_configs:
  - targets: ['197.248.129.45:9100']
```

Figure 4: 20 Broker connection to the node exporter.

4.4.3 Integration between the broker and QoS monitor

Figure 4.21 below shows how interconnection between the third party QoS monitor in our case Grafana and broker is happening.

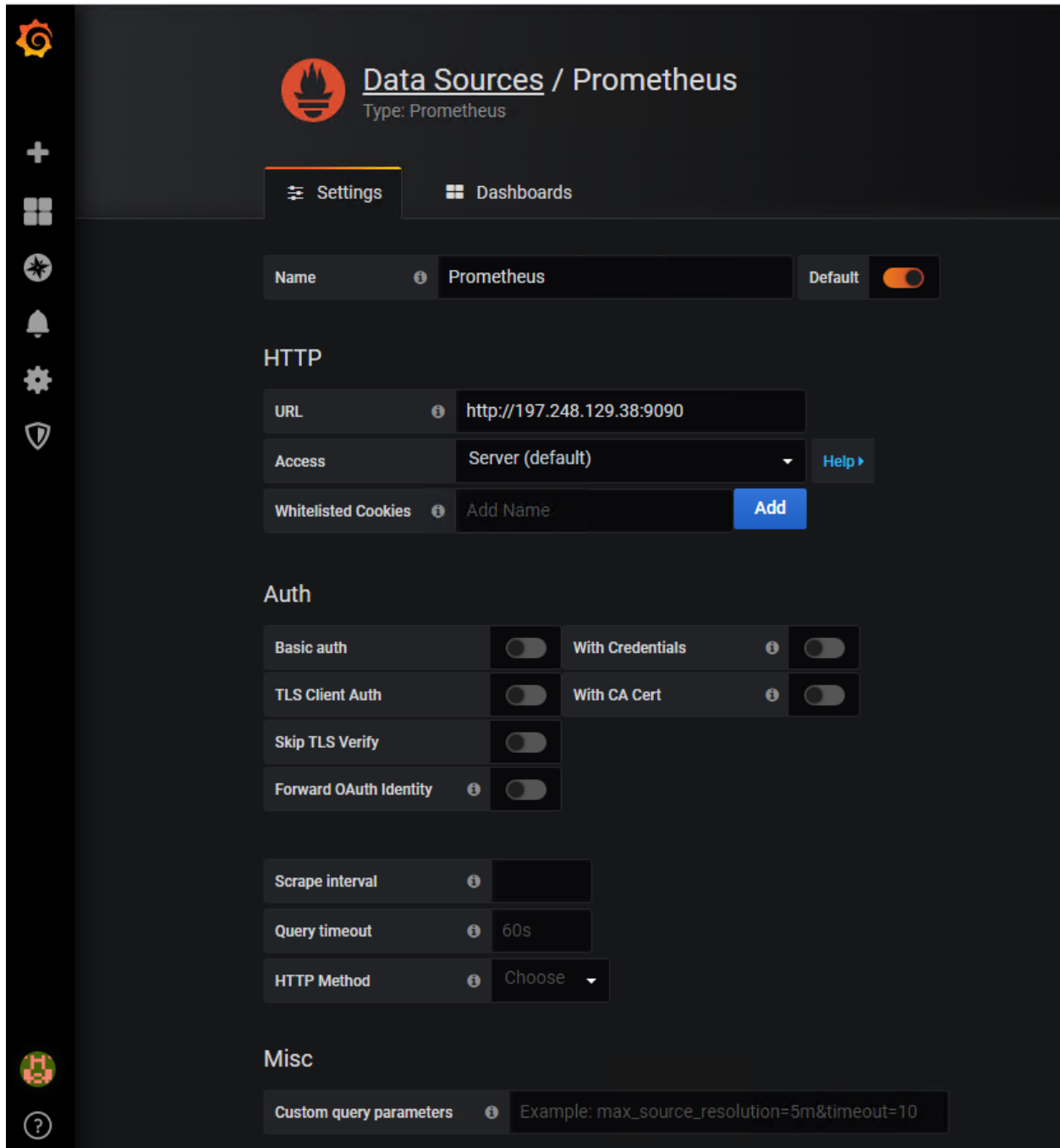


Figure 4: 21 Inter-connecting the Third party QoS monitor to the broker

4.5 QoS metrics analysis

4.5.1 Discrepancies between the actual and logical metrics assigned to the consumer.

Bearing in mind, that we know the resources allocated to physical cloud platform; the cloud could be configured in such that we could assign logical capacity to cloud consumers than the cloud could physically accommodate. This means that we could assign fake resources like virtual CPU, ram and hard disk to a consumer during provisioning and which would be visible in their virtual data center. Unless they know how to confirm then they would believe the right capacity has been provisioned.

| | |
|---------------------------|---------|
| Physical RAM | 31 GB |
| Physical HDD | 500GB |
| Tenant Demo allocated RAM | 50 GB |
| Tenant Demo allocated HDD | 1000 GB |

Table 4.1: Metrics analysis

The figure 4:22 below and table 2 above shows sample metrics assigned to a cloud consumer (demo). Doing some analysis, we realize that the user is assigned a RAM of 50 GB whereas the physical cloud platform has a total of 31 GB RAM. Again, the user is allocated a storage of 1000 GB, but the cloud platform has a total of 500 GB. With the QoS monitor such illegalities are easily noticed.

Also, table 3 below shows the status of the actual cloud resources and how the QoS monitor displays them to the consumer. This shows that the consumer can view the actual status of the cloud. The small deviations are as a result of the workloads in the server. For hard disk, RAM and CPU usage we do not expect any changes.

| | Hard disk | RAM | CPU | CPU Busy | Sys Load |
|---|-----------|-------|-----|----------|----------|
| Actual Resources | 393 GB | 31 GB | 16 | 5.2 % | 4.7% |
| Visible to consumer through QoS monitor | 393 GB | 31 GB | 16 | 3.2% | 6.4% |

Table 4.2: Actual resources usage against what the consumer sees through the QoS monitor

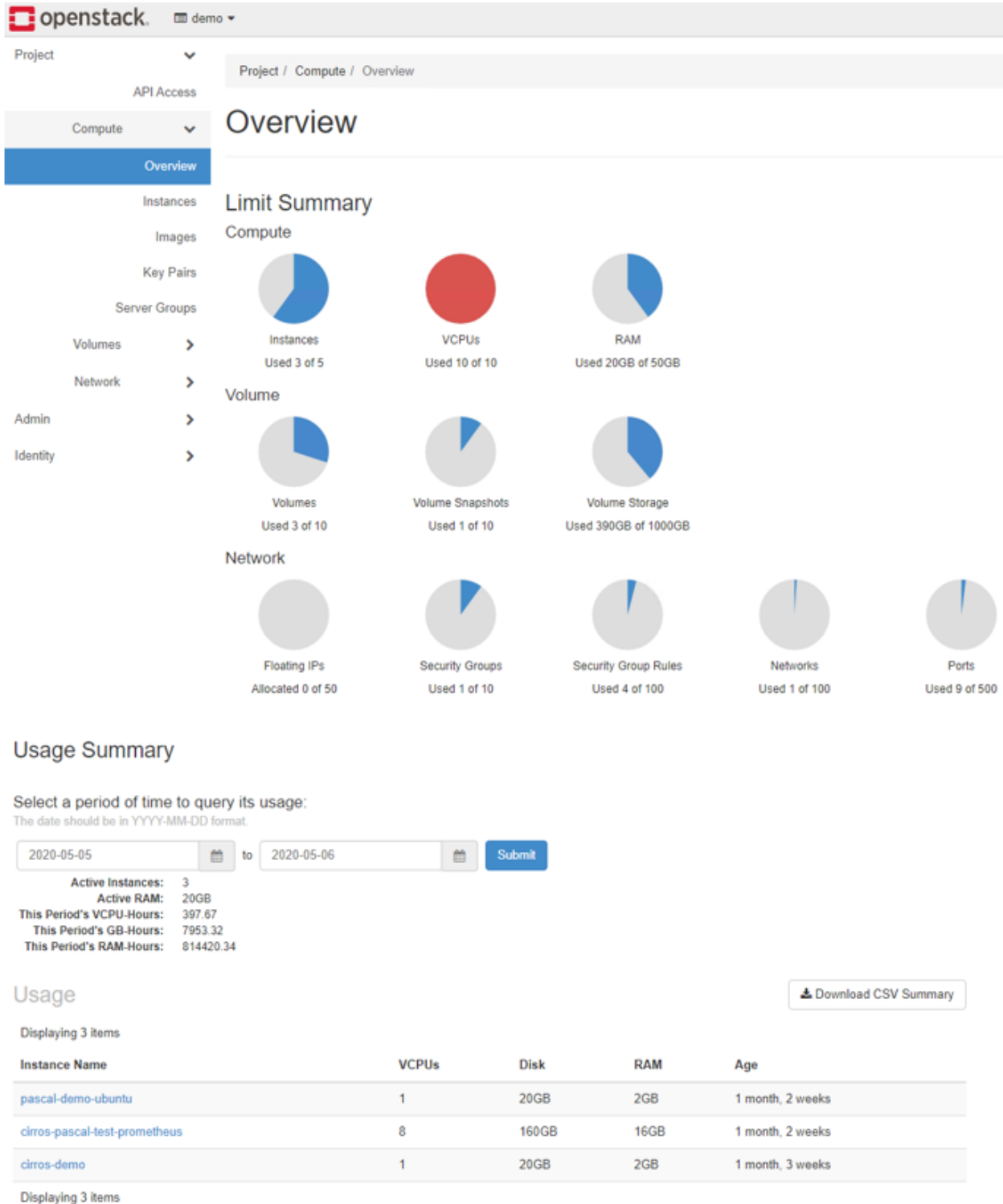


Figure 4: 22 Sample metric assigned to a cloud consumer

4.5.2 How the metrics are manipulated

Since open source cloud platform are highly configurable, it is possible to tune it to serve the purpose you need. Rogue CSPs cloud program it such that they steal small portions of resources from cloud consumer. The portions might almost be negligible, but they might affect one applications efficiency. For instance, in our cloud platform you can edit consumer metrics by just clicking button and updating as shown in the below figure 4.23. A simple solution to this would be to set threshold such that you cannot provision more than the cloud platform can physically accommodate.

| Quota Name | Limit |
|------------------------------|-------|
| VCPUs | 80 |
| Injected File Content Bytes | 10240 |
| Length of Injected File Path | 255 |
| Injected Files | 5 |
| Instances | 10 |
| Key Pairs | 100 |
| Metadata Items | 128 |
| RAM (MB) | 51200 |
| Server Group Members | 10 |
| Server Groups | 10 |

Figure 4: 23 Setting metric parameters

4.6 Evaluation of the model

4.6.1 Formal testing

From the above section 4.5 so that metrics could be altered logically on the cloud platform. Despite the metric allocated on the cloud platform, the QoS monitor shows the right metric since it is getting them directly from the hardware. For instance, the RAM and hard disk allocated in the figure 4.22 which surpass the actual hardware resources does not affect the result of the QoS monitor as shown in the below figure 4.24. If the cloud consumer had access to the QoS monitor

realizing that more RAM and hard disk space allocated was more than is available in the physical cloud would have been faster. This shows that model is effective

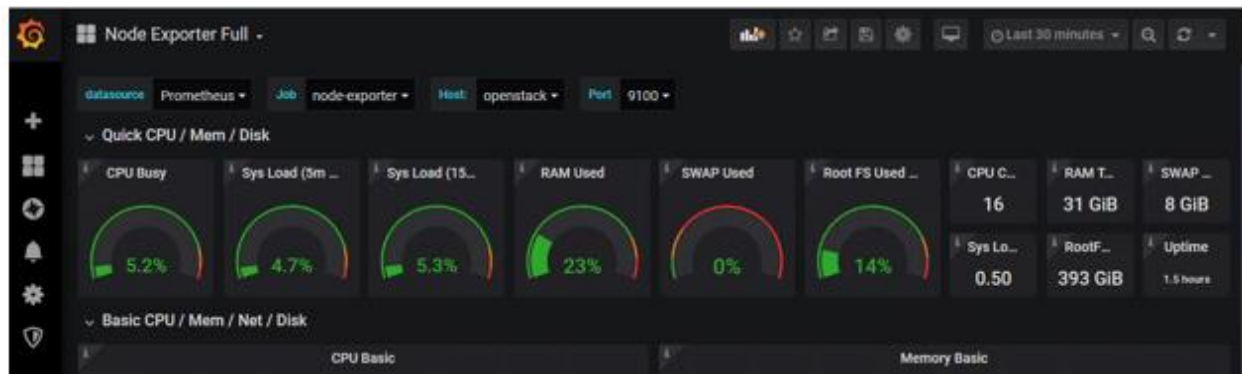


Figure 4: 24 QoS monitor showing actual resources.

4.6.2 Evaluation by the expert panelist

The general feedback after evaluation by the expert panelists was that the model served the purpose and was a true representation of how third party QoS monitoring ought to be done. It was suggested that the QoS monitor should keep history of the metrics to realize unusual trends occurring due to cloud service provider actions such as tampering with the API scraping the metrics. The developed model already had that capability.

4.7 The model shortcomings

High latencies because of traffic congestion in the connection between the QoS monitor and the broker could lead to inconsistency values at the QoS monitor side. This should be remediated by the use of high throughput secure direct connect link between the two parties. This would ensure that always the capacity allocated is proportionate to the actual traffic pushed by the cloud platform.

4.8 Discussions

This section shall discuss the results. As seen in the above results, we realized it is possible for CSPs to allocate logical resources to cloud consumers that even exceed what the physical infrastructure can accommodate. We were able to test that using various users in our prototype and were able confirm that. The QoS monitor comes in to help cloud consumers confirm whether what

they have been provisioned is available. This answers our first research question on the match between the provisioned and actual capacity provisioned.

Comparing our proposed model with the other models within our related works we find out that it is possible to give control to the cloud consumer to verify QoS metrics in real time without having to go through the cloud auditor. One of the models, which closely relates to our work in section 2.4.1 proposes QoS monitoring by third party professionals mainly the cloud auditor without giving the cloud consumers ability to view metrics from any tool as audit reports might not be up to date. This makes our model a more viable solution.

Again, comparing our proposed model with the conceptual model we can conclude that it was realized. The only modification done to the conceptual model to actualize the proposed model was the broker, which abstracts direct connection to the cloud platform. This means that the information acquired from the literature review and which contributed much to the development of the conceptual model was helpful in identifying the gap and hence the need of finding a solution leading to this proposed model.

This proposed model could be commercialized in two ways. The first way would be the cloud consumer pays the third party QoS monitor then the QoS monitor pays the cloud service provider. This way it might be somehow difficult to penetrate the market as it is dependent on the cloud consumer. Our assumption is that only big institutions will go that way since they know the benefit whereas small enterprises or individuals would choose to go directly to the cloud service provider, as they might not worry much about metrics.

The second model is where the cloud service provider pays the third party QoS monitor and discerning cloud consumers or already existing user acquire it for free for a certain period. This would boost sales on the CSPs side because it would be easier for clients to verify cloud status before even, they purchase the service and hence boosting their confidence and trust. In addition, the third party QoS monitor would act the cloud platform monitoring operations center and able to share unbiased feedback to the CSP where there are anomalies.

The principle of “Trust but Verify” in the cloud would best fit in this model and thus making it a source of trust using the QoS monitor. This is according to the trust levels in the cloud discussed in section 2.3.3 (iii) which emphasis on SLA verification-based trust.

CHAPTER FIVE: CONCLUSION AND RECOMMENDATIONS

This chapter discusses our conclusion, recommendations for further works and limitations

5.1 Conclusion

Our first objective was to explore the various trust model used in the multi-tenancy clouds. We were able to get several of them discussed in the literature review. Most of which focused on trust among the different tenants or trust through third party professionals. None of them was focusing on a third party QoS monitor where cloud consumers could get metrics in real time. This was a great opportunity for us to think through and come up with the developed model

The second objective was to develop a multi-tenancy clouds trust model using QoS monitoring. The objective was achieved, and we were able to come up with a model that enabled cloud consumer to go through the third party QoS monitor to confirm in real time the status of the cloud service provider platform status. This would help greatly once making the initial decision of which cloud platform to adopt.

The third objective was to develop a prototype of the proposed model. We were able to come up with a prototype, which worked almost the same as our proposed model. It was able to capture the anomalies in relation to assignment of resources to cloud customers. The prototype was developed using readily available tools, which made it easier and cheaper for us to consume. We were able to configure and program the tools to accomplish what we intended to.

5.2 Recommendations for further work

We recommend future works to implement a granularity to the information given to the cloud consumer. This means that after the cloud consumer has already acquired the cloud service then they can continue using the third party QoS monitor capable of showing only their virtual data center that hosts their resource. This will enable then not only to see the overall status of the cloud but also have more visibility to the resources allocated to them only.

We also recommend that cloud service providers to adopt the model. They could replicate a similar model within their environment making them more transparent to their discerning cloud consumers. As discussed, this might go a long way in helping them strengthen their reputation which in turn may lead to more sales and hence revenue growth.

5.3 Limitations

This system works well where cloud consumer is constrained to using CSP within a country. Since most of the full-fledged public cloud companies such as Microsoft Azure and Amazon web services do not have presence in all the countries then such consumers are left with few choices to select from and hence a need to scrutinize what they have locally before they can leap in. Also, the model works best where the third QoS monitor and the CSPs trust each other first. CSP negotiations should be from a management level where the QoS monitor also acts a monitoring entity for the said CSP and hence they see it as a benefit and not a way of exposing them.

REFERENCES

- AlJahdali, H., Albatli, A., Garraghan, P., Townend, P., Lau, L. and Xu, J., 2014, April. Multi-tenancy in cloud computing. In *2014 IEEE 8th International Symposium on Service Oriented System Engineering* (pp. 344-351). IEEE.
- Anderson, R., 2003. 'Trusted Computing' Frequently Asked Questions, <https://www.cl.cam.ac.uk/~rja14/tcpa-faq.html> accessed on 21st December 2019.
- Ansari, A., 2018. Service Level Agreement Governance For Cloud Computing. 10.13140/RG.2.2.11361.15206.
- Bezemer, C.P. and Zaidman, A., 2010, September. Multi-tenant SaaS applications: maintenance dream or nightmare?. In *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE)* (pp. 88-92). ACM.
- Brown, W.J., Anderson, V. and Tan, Q., 2012, September. Multitenancy-security risks and countermeasures. In *2012 15th International Conference on Network-Based Information Systems* (pp. 7-13). IEEE.
- Erikson, E.H., 1963. *Childhood and society* (2nd Eds) New York. NY: Norton.
- Foster I, Kesselman C (1999) *The grid: blueprint for a future computing infrastructure*. Morgan Kaufmann, San Mateo
- Huang, J. and Nicol, D.M., 2013. Trust mechanisms for cloud computing. *Journal of Cloud Computing: Advances, Systems and Applications*, 2(1), p.9.
- Ismail, M.H., Khater, M. and Zaki, M., 2017. *Digital Business Transformation and Strategy: What Do We Know So Far. Cambridge Service Alliance, November.*
- Kim, D.Y., 2019. A Design Methodology Using Prototyping Based on the Digital-Physical Models in the Architectural Design Process. *Sustainability*, 11(16), p.4416.
- Li, X.H., Liu, T.C., Li, Y. and Chen, Y., 2008, December. SPIN: Service performance isolation infrastructure in multi-tenancy environment. In *International Conference on Service-Oriented Computing* (pp. 649-663). Springer, Berlin, Heidelberg.
- Li, X.Y., Zhou, L.T., Shi, Y. and Guo, Y., 2010, July. A trusted computing environment model in cloud architecture. In *2010 International Conference on Machine Learning and Cybernetics* (Vol. 6, pp. 2843-2848). IEEE.
- Meixner, F. and Buettner, R., 2012. Trust as an integral part for success of cloud computing. In *The Seventh International Conference on Internet and Web Applications and Services, ICIW.*
- Mietzner, R., Metzger, A., Leymann, F. and Pohl, K., 2009, May. Variability modeling to support customization and deployment of multi-tenant-aware software as a service applications. In *Proceedings of the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems* (pp. 18-25). IEEE Computer Society.

- Odun-Ayo, I., Ajayi, O. and Falade, A., 2018. Cloud Computing and Quality of Service: Issues and Developments.
- Oates, B.J., 2005. Researching information systems and computing. Sage.
- Odun-Ayo, I. and Idoko, B.E., 2018. Cloud Trust Management–Issues and Developments.
- Odun-Ayo, I., Misra, S., Abayomi-Alli, O. and Ajayi, O., 2017, December. Cloud multi-tenancy: Issues and developments. In *Companion Proceedings of the 10th International Conference on Utility and Cloud Computing* (pp. 209-214). ACM.
- Padhy, R.P., Patra, M.R. and Satapathy, S.C., 2011. Cloud computing: security issues and research challenges. *International Journal of Computer Science and Information Technology & Security (IJCSITS)*, 1(2), pp.136-146.
- Povedano-Molina, J., Lopez-Vega, J.M., Lopez-Soler, J.M., Corradi, A. and Foschini, L., 2013. DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant Clouds. *Future Generation Computer Systems*, 29(8), pp.2041-2056.
- Rowe, G. and Wright, G., 2001. Expert opinions in forecasting: the role of the Delphi technique. In *Principles of forecasting* (pp. 125-144). Springer, Boston, MA.
- Sen, J., 2013. Security and Privacy Issues in Cloud Computing: Innovation Labs, Tata Consultancy Services Ltd., Kolkata, INDIA
- Shaikh, R. and Sasikumar, M., 2015. Trust model for measuring security strength of cloud computing service. *Procedia Computer Science*, 45, pp.380-389.
- Skinner, R., Nelson, R.R., Chin, W.W. and Land, L., 2015. The Delphi Method Research Strategy in Studies of Information Systems. *Cais*, 37, p.2.
- Stallman, R., 2018. Can You Trust Your Computer?, <https://www.gnu.org/philosophy/can-you-trust.en.html> accessed on 21st December 2019.
- Tang, B. and Sandhu, R., 2013, August. Cross-tenant trust models in cloud computing. In *2013 IEEE 14th International Conference on Information Reuse & Integration (IRI)* (pp. 129-136). IEEE.
- Tang, B., Sandhu, R. and Li, Q., 2015. Multi-tenancy authorization models for collaborative cloud services. *Concurrency and Computation: Practice and Experience*, 27(11), pp.2851-2868.
- Wang, Y.D. and Emurian, H.H., 2005. An overview of online trust: Concepts, elements, and implications. *Computers in human behavior*, 21(1), pp.105-125.
- Wang, Z.H., Guo, C.J., Gao, B., Sun, W., Zhang, Z. and An, W.H., 2008, October. A study and performance evaluation of the multi-tenant data tier design patterns for service oriented computing. In *2008 IEEE International Conference on e-Business Engineering* (pp. 94-101). IEEE.

Weinhardt, C., Anandasivam, A., Blau, B., Borissov, N., Meinl, T., Michalk, W. and Stöber, J., 2009. Cloud computing – a classification, business models, and research directions. *Business & Information Systems Engineering*, 1(5), pp.391-399.

APPENDICES

Appendix 1: Project Schedule

The following schedule was followed.

| ID | Task Name | Start | Finish | Duration | Jan 2020 | Feb 2020 | Mar 2020 | |
|----|---|-----------|-----------|----------|----------|----------|----------|-----|
| | | | | | | 2/2 2/9 | 3/1 3/8 | 4/5 |
| 1 | Preparing the semi-structured interview | 1/20/2020 | 1/24/2020 | 1w | ■ | | | |
| 2 | Data collection and analysis | 1/23/2020 | 1/31/2020 | 1.4w | ■ | | | |
| 3 | Interpreting the data | 2/3/2020 | 2/7/2020 | 1w | | ■ | | |
| 4 | Designing the prototype | 2/6/2020 | 2/19/2020 | 2w | | ■ | | |
| 5 | Developing the prototype | 2/17/2020 | 3/6/2020 | 3w | | | ■ | |
| 6 | Writing the Project report | 2/17/2020 | 4/3/2020 | 7w | | | ■ | |

Appendix 2: Budget

| id | Item | Cost (KSh) |
|----|---|---------------|
| 1 | Internet connectivity | 10,000 |
| 2 | Since most of the tools used are open source no capital has been used | 0 |
| 3 | Power (running on premise server) | 5,000 |
| 4 | Miscellaneous | 1,000 |
| | TOTAL | 16,000 |

Appendix 2: Hardware and Software requirements

The below hardware and software requirement were tested to work in deploying the respective tools. This can vary depending on usage.

OpenStack cloud

- i. Ubuntu server 18.04 LTS
- ii. 8 GB RAM

- iii. 2 vCPUS
- iv. Hard disk capacity of 250GB
- v. Internet connection
- vi. User with sudo privileges
- vii. Python, Django framework, Apache and MYSQL programming language skills

Prometheus

- i. Ubuntu 16.04 server or higher
- ii. A non-root user with sudo privileges
- iii. Go and Time series database skills
- iv. Docker for deploying containers

Grafana

- i. Ubuntu server 18.04 LTS
- ii. 4 GB RAM
- iii. 1 CPU
- iv. Apache, SQLite database and Go programming language
- v. Docker for deploying the ubuntu container