CROSS CLOUD WITH DOCKER AWARE SOFTWARE DEFINED NETWORKING

K. Janarthanan - IT15051608

Peramune P.R.L.C - IT14009532

Theviyanthan K - IT14004414

Ranaweera A.T. - IT15026644

Group ID – 18-015

Final Report

Bachelor of Science (Hons.) in Information Technology Specializing in Computer System and Network Engineering

Department of Information Systems

Sri Lanka Institute of Information Technology

Sri Lanka

October 2018

CROSS CLOUD WITH DOCKER AWARE SOFTWARE DEFINED NETWORKING

K. Janarthanan - IT15051608
Peramune P.R.L.C - IT14009532
Theviyanthan K - IT14004414
Ranaweera A.T. - IT15026644

Dissertation submitted in partial fulfillment of the requirements for Bachelor of Science Special (Honors) Degree in Information Technology

Department of Information Systems

Sri Lanka Institute of Information Technology

Sri Lanka

October 2018

1 Declaration

We declare that this is our own work and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Name	**Student ID**	**Signature**
K. Janarthanan	IT15051608	
Peramune P.R.L.C	IT14009532	
Theviyanthan K	IT14004414	
Ranaweera A.T.	IT15026644	

The above candidates are carrying out research for the undergraduate Dissertation under my supervision.

Signature of the supervisor: Date

2 Abstract

In today's world, more and more organizations are adopting the cloud services mainly because of the reliability and affordability provided by them. However, there are several drawbacks faced by the cloud users and cloud service providers. Apart from the security perspective, the cloud users are facing challenges in control and visibility, lack of standard service interfaces, difficulty in deploying applications across multiple clouds and vendor lock-in. Also, cloud service providers are facing challenges in degradation of the quality of service provided because of the distance between cloud datacenter and the end user and unexpected interruption of services etc. The above problems can be reduced to a greater extent by adopting Multi Cross Cloud Infrastructure. A Cross Cloud Infrastructure is termed as using multiple cloud service providers services. This benefits the cloud users to receive the best quality services in order to increase their productivity. Furthermore, there is a necessity to allocate user services on demand and optimizing the resource allocation for cloud service providers. So, the main aim of this research is to build a common platform to manage a cross-cloud environment. Cross cloud platform can be implemented within an organization or Enterprise and is used by 3rd level support team or Infrastructure team to provide multiple services (E.g. – Delivering application containers, VM connection for development purposes) to end users based on some SLA (policies) with more control and visibility. Software Defined Network (SDN) is a pathbreaking technological innovation that is expected to complement the rapidly growing cloud infrastructure. SDN separates the control plane from the data plane of a network and centralizes the control plane in a single device allowing network developers to dynamically propagate different networking paradigms. However, the centralization of the control plane, while affording its fair share of benefits, makes SDN vulnerable to several security threats such as denial of service, and man-in-the-middle attacks. This research proposes a radical way to address such limitations by distributing the control plane across the network using the blockchain algorithm while allowing authorized applications to write to the blockchain to communicate with the data plane.

3 ACKNOWLEDGEMENT

We like to express our appreciation and gratitude to our supervisor – Mr. Lakmal Rupasinghe (Senior Lecturer, Department of ISE, SLIIT-Malabe, Sri Lanka) to make this study successful and for encouraging us in this whole experimentation period.

K. Janarthanan, Peramune P.R.L.C, Theviyanthan K and Ranaweera A.T.
8th October 2018

TABLE OF CONTENT

4 LIST OF TABLES

5 LIST OF FIGURES

6 LIST OF ABBREVIATIONS

VM	Virtual Machine
AWS	Amazon Web Services
SLA	Service Level Agreements
IT	Information Technology
VPN	Virtual Private Network
SDN	Software Defined Networking

7 INTRODUCTION

7.1 Overview

In today world, more and more organizations are adopting the cloud services mainly because of the reliability and affordability provided by them. However, there are several drawbacks faced by the cloud users and cloud service providers. Apart from the security perspective, the cloud users are facing challenges in control and visibility, lack of standard service interfaces, difficulty in deploying applications across multiple clouds and vendor lock-in. Also, cloud service providers are facing challenges in degradation of the quality of service provided because of the distance between cloud datacenter and the end user and unexpected interruption of services etc. The above problems can be reduced by adopting Multi Cross Cloud Infrastructure. A Cross Cloud Infrastructure is termed as using multiple cloud service providers services. This benefits the cloud users to receive the best quality services to increase their productivity. Furthermore, there is a necessity to allocate user services on demand and optimizing the resource allocation for cloud service providers. So, the main aim of this research is to build a common platform to manage a cross-cloud environment.

Cross cloud platform can be implemented within an organization or Enterprise and is used by 3rd level support team or Infrastructure team to provide multiple services (E.g. – Delivering application containers, VM connection for development purposes) to end users based on some SLA (policies) with more control and visibility.

Migration of containers among VMs running in the same cloud environment or in between different cloud environments, to save the cloud cost and for some testing purposes. Migration is based upon policy rules/parameters such as pre-configured resource reservation on containers made by the users, policies that define which containers should always co-exist with each other (For E.g. – Application containers and DB containers must co-exist inside a single VM for better performance), policies that define which containers cannot co-exist with each other (For E.g. – If two containers are used for redundancy, then they cannot co-exist within inside a single VM) and policies related to VM specific container. For this purpose, a suitable allocation algorithm will be developed and used in the backend so by that containers could be migrated based on configured parameters/policies to another VM.

There are several advantages of running containers with in the cloud. They are increased portability, heavy layer of isolation between guest environment and host server, increased security and increased stability. This platform can be thought as a system implemented within an organization or Enterprise and is used by 3rd level support team or Infrastructure team to provide multiple services (E.g. – Delivering application containers, VM connection for development purposes) to end users based on some SLA (policies) with more control and visibility.

Using the Cross-Cloud platform that built on top of Multi-Cloud, benefits the end user in following ways,

• Performance guarantee – Performance can be increased by deploying workloads across different clouds

• Availability – Geographically distributed cloud infrastructure can reduce the latency and unexpected outages

• Convenience – One central platform means, easy management

• Dynamic distribution of workload – Workload can be moved to a cloud location that is closer to the customer

Now let’s look, what are the motivations behind this cross-cloud platform architecture,

• Resource provisioning – Cloud resources can be expanded as needed. This property is called elasticity. By using two or more clouds the user can able to switch his/her cloud deployment according to the need and demand.

• Reduced Costs – This architecture brings the services closer to the cloud users with lower costs. Thus, providing a way to migrate the workloads/containers and services from one VM to another VM and powering down the redundant VMs.

• Avoid Vendor Lock-In – The end cloud users are not locked to a certain cloud service provider. They have the freedom to move the workload between different cloud environment.

• Legal regulation – There are some government regulations in the world that avoid the data to be moved outside of their countries. By adopting the cross-cloud architecture, other cloud service providers can host their data/applications in a cloud data center which situate in that country.

Some other features of this cross cloud platform is managing VMs (on/off/restart) from one place, monitoring the resource consumption of deployed VMs, migration of the containers from VM to VM within the same cloud or different cloud, retrieving the containers details in single pane, giving information about the best VM in the cloud to deploy new containers, ping tool to verify the connectivity among different VMs deployed in the cloud , deploying containers based on different policies and executing remote commands in VMs without individually logging into each.

The most commonly used networking architectures have undergone little change during the last two decades. Creating computer networks using devices such as switches and routers have been adequate to address the computing and networking needs hitherto. However, with the wide adoption of cloud computing, the proliferation of Internet of Things devices and the big data that stem from such devices, changing business models and needs, and the growth of multinational companies that want to cater to customers all over the world, the contemporary networking paradigm, it could be argued, is fast approaching its glass ceiling.

The traditional network architecture employs networking devices with a tightly coupled control and data plane encompassed within them. Ergo, the networks besot by such architectures are very rigid in nature, thus failing to provide a flexible solution to dynamic networking requirements. Amending the network to fit changing business needs and use cases mean that each and every networking device should be individually configured. This is fast proving to be a limiting factor in adapting to the needs of modern-day data centers and cloud infrastructures.

Software Defined Networks (SDN) is a pioneering idea that separates the control plane from the data plane of the network allowing the manipulation of the control panel to take place through a single portal. Though the initial seeds of this idea were sown in 2004 in the form of Forwarding and Control Element Separation Framework[22], it failed to gain traction owing to fears about the issues that may arise from the failure of the control plane, and vendors dreading increased competition among themselves.

OpenFlow protocol was created in 2008 paving the way for more practical implementations of SDN. The Open Networking Foundation was inaugurated in 2011 to accelerate the development of SDN and OpenFlow. In 2012, Google acknowledged its first use of OpenFlow in their datacenters.

At present, OpenFlow-enabled switches are available in the market, with many commercial implementations of SDN Controllers available.

7.2 Background

Currently, there is a shortage of a platform to manage both AWS and AZURE together to cater the different cloud user needs. Only limited number of research was conducted on cloud federation topic and SDN over containers. In 2014, four members Cuadrado, Navas, Duenas, and Vaquero carried out a research on topic “Research Challenges for Cross-Cloud Applications”. In that paper, they have addressed the main challenges for potential cross-cloud applications explaining what kind of support is needed from Infrastructure and networking side [1]. In 2015, Gianluca Zangara, Diego Terrana, Pietro Paolo Corso, Marco Ughetti and Guido Montalbano proposed a sample cloud federation architecture based on open technologies to accept a different type of cloud service model and to bill the service to the customer and to get details about the health status of the cloud [2]. In 2016, M.R.M. Assis and L.F. Bittencourt wrote an article on the topic “A Survey on Cloud Federation Architectures: Identifying Functional and Non-Functional Properties”. There they have specified the functional and non-functional properties that belong to cloud federation architecture [3]. In 2010, Antonio Celesti, Francesco Tusa, Massimo Villari and Antonio Puliafito proposed an authentication agent module, which can handle secure communications within the cross-cloud federation environment. To address that problem, they proposed a solution based on Idp/SP with SAML technology [4].

VMWare cloud services provide some plug ins which support movement of VMWare related products between different clouds. The main purpose of this is to move VMs and other data storage between cloud and on-premise VMWare environment. VMWare also implemented a technology called distributed resource scheduling (DRS) which migrate VMs from one ESXI host to another ESXI host depending on the resource allocation. Thus, it can make sure of load balancing on physical servers. Migration of containers is included with cloudify using kubernetes, but it only involves with pods. Kubernetes is a product from google which is used in the orchestration of containers enabling central, easy and efficient management. OpenVZ provides containers which are little bit different from standard docker containers which is used for live migration of containers from one host to another.

Live migration of containers is currently provided by Virtuozzo who have first created this live migration technology based on OpenVZ containers. “runC” also provides containers engine working based on CRIU technology. CRIU is check and restore in user space technology which helps to freeze the application runtime/memory and transfer it to another host and restart it again. Jelastic provides solution of migration of containers between different hardware, data centers and cloud providers.

VMware affinity and anti-affinity rules are used in the distributed resource scheduler (DRS) mechanism to load balance the virtual machines running on top of the physical servers. Affinity rules are included, to place a specific group of virtual machines (VMs) on the single physical server. This is done, to improve the performance of multi-tier applications running inside the VMs. Anti-affinity rules are used to place specific group of VMs, to run on different physical hosts. This is done, to improve the redundancy of the applications running inside those VMs to ensure the high availability (HA) of applications [16].

There are only few platforms available in the market for container management in large scale. Kubernetes and Docker Swarm are among them. Kubernetes, which is an enterprise container orchestration platform deals with the automation of container deployment and the management of containers across physical hosts or virtual machines(VMs) [38]. Containers are deployed in pods (smallest deployable unit) with in nodes. Nodes form a cluster and the cluster is managed by master node. Docker swarm which is also an orchestration and clustering tool for docker containers used to schedule and deploy containers across worker nodes [39]. But in this research, policies-based containers deployment within the best virtual machine across Multi-Cloud (AWS/AZURE) is discussed. So, the 3rd level team member can easily use this feature to identify the best VM to deploy new containers according to the requirement

Researches to create programmable networks have been afoot since the early 90s as the active networks research program surveyed pathbreaking methodologies to overcome the limitations of the then prevalent networking approaches [23]. The concept of separating the control plane from the forwarding plane has been discussed in Signaling System 7, Ipsilon Flow Switching, and Asynchronous Transfer mode even prior to 1998 [24].

One of the very first proposals to standardize programmable networks was the IEEE P1520 Standards Initiative for Programmable Networks Interfaces. It elucidated about the need to create a higher-level abstraction of the network and the requirement of a programming interface to define a network. The SoftRouter architecture provided a dynamic binding between the forwarding plane and the software-based control plane. The Forwarding and Control Element Separation that was created by the Internet Engineering Task Force (IETF) attempted to standardize the way the controllers communicated with the forwarding planes [25].

The need to stay abreast of the rapid pace at which virtual cloud computing was progressing and the burgeoning of big data, sired the OpenFlow protocol in 2008. Until 2011, when the Open Network Foundation took over the OpenFlow protocol, it was managed and developed by a group of individuals who met physically at Stanford University.

Developed with the goals of expediting network innovations, and automating the management of large networks, Software Defined Network's and OpenFlow's use cases run the gamut. Their application range from enabling dynamic, flexible communication with virtual cloud machines that make up data centers to ensuring Quality of Service in enterprises to aid effective and efficient video conferences and remote desktop access, which is onerous to implement in large networks since QoS should be employed end-to-end between two communicating devices [26].

The blockchain algorithm was devised in 2008 by a person or a group of people with the pseudonym Satoshi Nakamoto to enable trustless financial transactions. Originally meant to decentralize how financial transactions take place digitally, in blockchain, the tech-community has recently found various use cases that range from liquid democracy to decentralized domain name systems.

The blockchain algorithm enables two entities unknown to each other transfer funds without a central authority verifying the transaction. Even though when done physically by means of cash and coins, transactions avoid the double-spend problem since the cash paid to one party cannot be paid to another, when digitally transferred, it can lead to double-spend problems. In digital transactions, currency is a virtual value that is added and deducted from an account during transactions. Since individual ownerships of such accounts can lead to lack of consensus among the transacting parties, a centralized authority is needed to evade the double-spend problem.

Be that as it may, a centralized authority leads to many practical constraints such as the parties involved in the transaction having to pay a processing fee to perform a transaction. Withal, a centralized financial authority may lead to increased state surveillance on economic activities, economic censorship, and at worst, government institutions embezzling citizens’ money in the pretext of social justice and nation-building [27].

The blockchain solves this problem by providing a distributed ledger to store every transaction. Transactions between two parties take place with the use of public-private key pairs, and each participating individual can verify the transaction by parsing the transaction history. A chain of blocks that serves as a ledger orders the transactions chronologically, allowing participating entities to avoid the double-spend problem [28].

Albeit the algorithm was meant to power the bitcoin currency, the algorithm has found many uses in various fields. Blockchain algorithm is now employed in decentralizing proof of existence documents, decentralizing file storage, and enabling trustless communication between Internet of Things devices that are booming [29].

7.3 Research Gap

There is no any or lack of a platform to manage both AWS and Azure cloud environment on a central point which gives more freedom and flexibility to cloud users. Also, currently there is a lack of an efficient model to migrate the containers that are running inside the VM on top of the cloud. The main purpose of this research part is migration of containers based on policies between VMs in the same cloud or between different cloud according to user need. There is no currently any system available for these kinds of requests. Also, in this system containers are placed inside the VMs running in cloud. There are several advantages of running containers inside the VM. They are, increased portability, heavy layer of separation between the guest environment and host environment, increased security and increased stability. This part is mainly used by 3rd level team in an organization such as Infrastructure team to handle migration of containers based on policies in the same cloud or between different cloud according to user need. For this user manageable interface will be created and the user has the freedom to select which VM to VM the migration of container takes place. In order for the communication and migration of containers between VMs in two different clouds, AZURE and AWS, VPN based tunnel is used. By using VPN based tunnel more protection is provided for containers.

Irrespective of OpenFlow being an actively pursued technology by both network device manufacturers and researchers, the protocol entails various constraints and limitations. These limitations pose a great problem in the practical implementation of Software Defined Networks.

Despite Transport Layer security protocol being a mandatory requirement in the original implementation of the OpenFlow protocol, this has been made optional since OpenFlow version 1.30. This stems from various problems and difficulties that arise when implementing TLS in a Software Defined Network. Configuring TLS necessitates generating a site-wide certificate, generating controller certificates, signing those certificates with a private key, and installing the right keys and certificates on all the network devices in the network.

The aforementioned complications have resulted in popular OpenFlow switch vendors forgoing support for TLS in their products. Off the majority of the OpenFlow controller manufacturers, only one provides support for TLS.

The absence of TLS begets security threats such as Man-in-the-Middle attacks. Since the communication among controllers and switches are not secured by an encrypted tunnel, it makes the network susceptible to unauthorized individuals with malicious intentions to both eavesdrop on the communication and modify messages exchanged [31].

Even though this might not be a huge threat in networks that are physically secure such as the ones that are most likely to be found in data centers, as far as campus-style networks and remote deployments are concerned, the lack of TLS poses great threats [32].

Such security issues that arise from the non-existence of a secure channel result in worse consequences in in-band OpenFlow networks than when the same issues arise in traditional network environments [30].

The absence of a mechanism by which switches could be authenticated allows attackers, depending on the topology discovery logic, receive sensitive traffic from the controller by faking attached host credentials [33]. This also allows an attacker to perform network reconnaissance and make a portion of a network to crash [34].

OpenFlow communication from controllers to switches take place over a certain port, and the listener mode supported by most OpenFlow switches let them listen to unauthenticated connections from any network source. This allows an attacker to send spurious flow-table entries that can bring down certain parts of the network, and be used as a proxy to perform further attacks in the future [30].

Tampering the state of the flow-table in OpenFlow switches can lead to mismatches between how the controller sees the rule-state of a network and the way each switch sees the rule-state of a network. This can pave the way for network outages and access-control failures [30].

Moreover, since most implementations of OpenFlow controllers use only one controller, the network is prone to single point of failures [31]. In addition, this also makes it easy for an attacker to perform a Denial of Service attack on the network. Although different solutions have been proposed to rectify this problem by means of enabling multiple controllers to control the network, such solutions complicate matters by creating east-west communication channels in addition to the north-south communication conduits [35].

Software Defined Networking and blockchain are recent phenomena that are revolutionizing their respective fields of application. However, a cursory appraisal of the problems faced by Software Defined Networks, and the advantages the blockchain algorithm bring into financial transactions lays bare the possible applications of the blockchain algorithm in Software Defined Networking.

Problems such as creating consensus among controllers in a multi-controller environment, authenticating controllers and switches, securing communication between switches and controllers can all be accomplished by the use of distributed ledgers, consensus algorithms, and private-public key pairs- all of which combine to form the blockchain algorithm.

Only a few studies and researches on SDN have tried to exploit the benefits accorded by the blockchain algorithm. A research carried out by the University of Patras tried to make the Internet of Things more secure by using SDN along with a Security Gateway for sensors that made authentication decisions by parsing blockchains [36]. A paper published by Sadhu Ram Basnet and Subama Shakya discusses about securing file-sharing using a distributed ledger over an SDN network [37].

7.4 Research Problem

Cloud users are affected with problems such as lack of dynamic resource provisioning, increased cost and legal regulation when adopting Cloud solutions for an Enterprise Organization. Also, Currently the 3rd level teams of IT organizations are facing difficulties to support different user requests regarding containers migrations between cloud. Through this research project, above problems are minimized or eliminated by adopting cross cloud approach and building a common interface to manage both AWS and AZURE environments together from single place. Also, policies based containers migration helps to efficiently migrate the containers between AWS and AZURE.

This research also, would investigate the unprecedented idea of improving Software Defined Network with the use of the blockchain algorithm. This study aims to explore the possibility of distributing the control-plane logic while providing portals through which the entire network could be manipulated. By employing a blockchain based solution, feasibility of authenticating controllers and switches, and creating an immutable history of flow-table state changes would also be examined.

7.5 Research Objectives

Building a cross-cloud platform which gives more visibility and control to the 3rd level team who provides different services to end users such as developers and testers. Main objectives of this research project is, enabling dynamic resource provisioning - By using multi-clouds approach, the user can able to move his/her cloud deployment according to the need and demand, Reduced costs – The proposed cross-cloud platform brings the services closer to the cloud users with lower costs by providing a way to migrate the containers and services from one VM to another VM and powering down the redundant VMs in the cloud, To avoid single vendor Lock-In – The end cloud users are not restricted to a certain cloud service provider. They have the freedom to move the workload between different cloud environment depending on the request and Handling legal regulation -There are some government/regional regulations in the world that avoid the data to be moved outside of their countries. By adopting cross-cloud, the problem can be mitigated.

Additional Features of the proposed Cross-Cloud Platform are,

1. Monitoring the resource consumption of deployed VMs across Azure/AWS in a single plane at the real-time
2. Managing the VMs (On/Off/Restart) from one place
3. Migrating the containers (dockers) from one VM to another VM between AWS and AZURE based on different policies
4. Retrieving the containers details in a single plane which are deployed inside the VMs
5. Giving information about the best VM in the cloud to deploy new containers
6. Ping tool to verify the connectivity between VMs in inter/intra cloud
7. Execute remote command in VMs without logging into individual VMs
8. VPN based connection between the AWS and AZURE cloud environments

The other part of research is to demonstrate how a distributed ledger along with the other elements of the blockchain algorithm could help rectify some of the problems found in SDNs. The problems that would be addressed in this research are:

1. Allowing multiple controllers to manipulate the SDN using a distributed ledger.
2. Authenticating controllers to avoid listener-mode vulnerability.
3. Creating an immutable history of flow-table state changes.

4. Ensuring that the controller view of the rule-state of the SDN is not different from those of the switches.

Even though a more practical implementation of an SDN network using blockchain would require a complete transformation and adaptation of the OpenFlow protocol to suit the paradigm of decentralized applications, the scope of this research work shall be restricted to, owing to time constraints, merely demonstrating a working solution that integrates the blockchain algorithm with specific parts of the OpenFlow protocol to solve the aforementioned problems. Ergo, this research shall only demonstrate the benefits the integration of the blockchain algorithm shall bequeath to Software Defined Networks. Practically implementing such an idea would require further work. This research is expected to be a precursor to further researches on the proposed idea.

7.5.1 Policy based containers migration in cloud

Different policies such as co-existing policy, cannot co-existing policy, resource reservation policy and host specific containers policy are used and validated before migrating the containers between virtual machines in the same cloud or different cloud (AWS and AZURE). The main reasons for doing container migrations are,

- High availability (HA)
- To do hardware/System upgrade in the VM which host the containers
- Cost reduction by moving container from one VM to another VM in different cloud
- For testing and evaluation purposes for different application running in containers across different cloud
- To improve the workload balance

This helps the 3rd level team (Infrastructure Team) of an organization, to manage the migration of containers efficiently from one place another place in controlled manner.

7.5.2 Setting QoS to Docker Containers

Classful priority queueing discipline is applied to the docker bridge interface and the quality of services to docker containers are set. By using the cross-cloud management platform the 3rd level team can easily scale up and scale down the service offering to containers. Mainly 3 service offering are provided to docker containers. They are,

- Gold offering – High priority
- Silver offering – Medium priority
- Standard offering – Low priority

7.5.3 Policy based containers deployment across multi-cloud

Best virtual machine for the deployment of container in the cloud is selected according to the new container requirement such as memory and disk capacity and after analyzing to the resource usage inside the virtual machines.

There are several advantages of running containers inside the VM. They are, increased portability, provides a heavy layer of separation between the guest environment and host environment, increased security and increased stability. The policies involved in the best VM selection are,

- Default VM policy – Best VM search is expanded across all the VMs in the selected cloud
- Include VM policy – Best VM search is limited within the VMs in the policy in the selected cloud
  (Example – The user wants to search the best VM within the production VMs list or development VMs list only)
- Exclude VM policy – Policy to exclude some VMs from the search of best VM to deploy new container
  (Example – The user wants to exclude the Temporary VMs list from the search)

Policies are passed through cross cloud platform and the best VM selection is done through the controllers placed in each cloud.

7.5.4 Decentralizing an SDN controller using blockchain

Software Defined Network (SDN) is a pathbreaking technological innovation that is expected to complement the rapidly growing cloud infrastructure. SDN separates the control plane from the data plane of a network and centralizes the control plane in a single device allowing network developers to dynamically create different networking paradigms. However, the centralization of the control plane, while affording its fair share of benefits, makes SDN vulnerable to several security threats such as denial of service, and man-in-the-middle attacks. This research proposes a radical way to address such limitations by distributing the control plane across the network using the blockchain algorithm while allowing authorized applications to write to the blockchain to communicate with the data plane.

8 METHODOLOGY

This cross-cloud platform is mainly designed for an organization where different kind of users are involved. Typically, the end user will request a service through a ticket to the 3rd level team for the purposes such as, to deploy new container in the VM for development purposes or testing purposes and for the migration of container from one VM to another VM with in same cloud or different cloud for different purposes. Thereby, the 3rd level team of an organization can efficiently use this management platform to provide different services for the end users. The 3rd level team can be thought as a service provider within an organization providing services to end users such as developers and testers in the organization. Some features in the platform are tied with policies, for example container migration. Policies are similar to rules that are used to control the containers placement during the migration process. The policies are configured using the cross-cloud management platform and they are stored in the text files. Before doing any migration of containers, the configured policies will be checked out and the request will be validated. Then only if it permits, migration is allowed between the VMs in same cloud or different cloud.

This test cross-cloud management platform uses the AZURE cloud and AWS cloud, to achieve the multi-cloud approach. For this, free-tier services from AZURE and AWS clouds are utilized. VMs in AZURE are created based on the operating system ubuntu 16.04, 1vCPU and 1 GB RAM (B1s type). In AWS, all EC2 instances are based on the operating system ubuntu 16.04, 1vCPU and 1 GB RAM (t2.micro type). Containers are deployed from Docker images. This platform is made of using AZURE APIs, AWS APIs, ASP .NET with C#, PowerShell scrips and Shell scripts

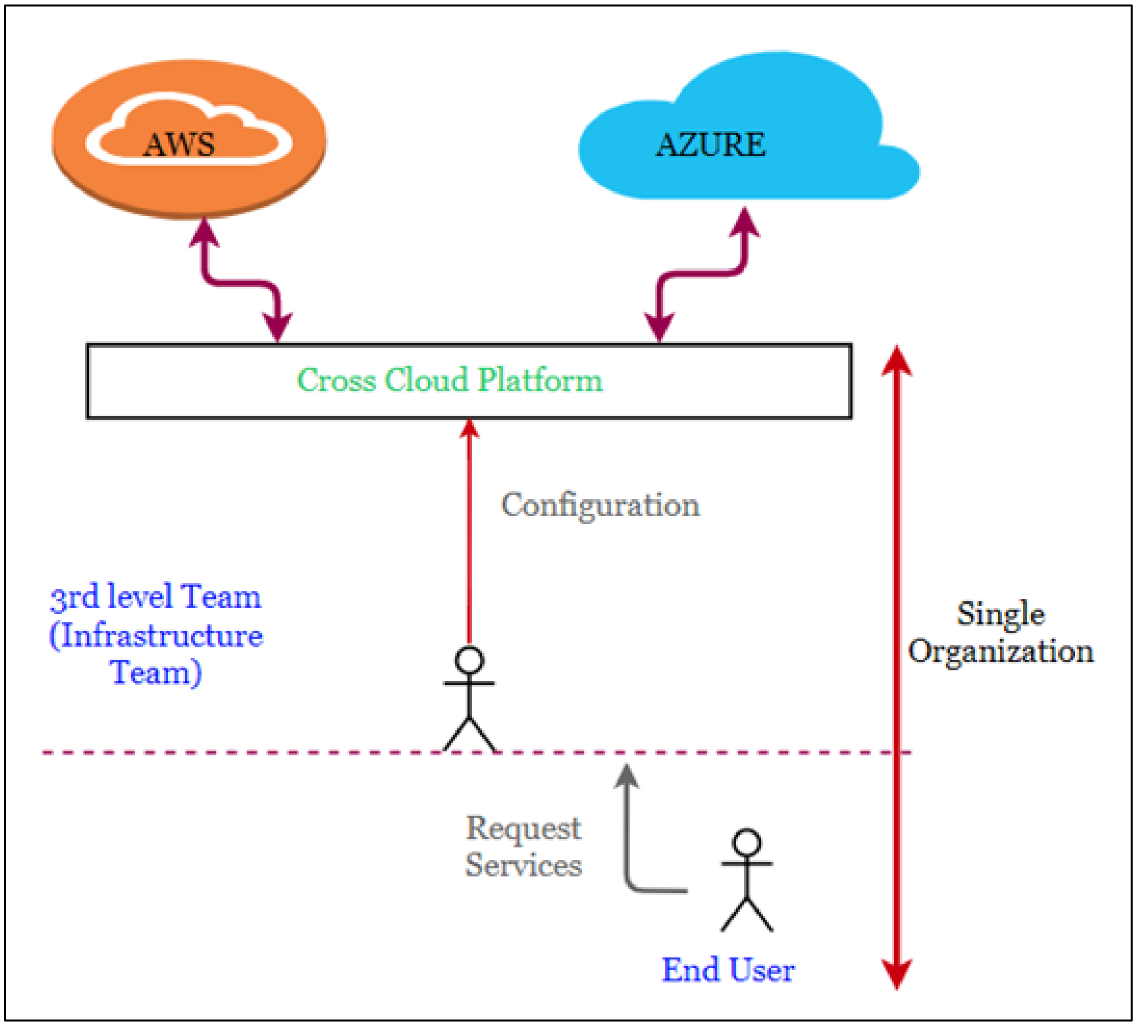
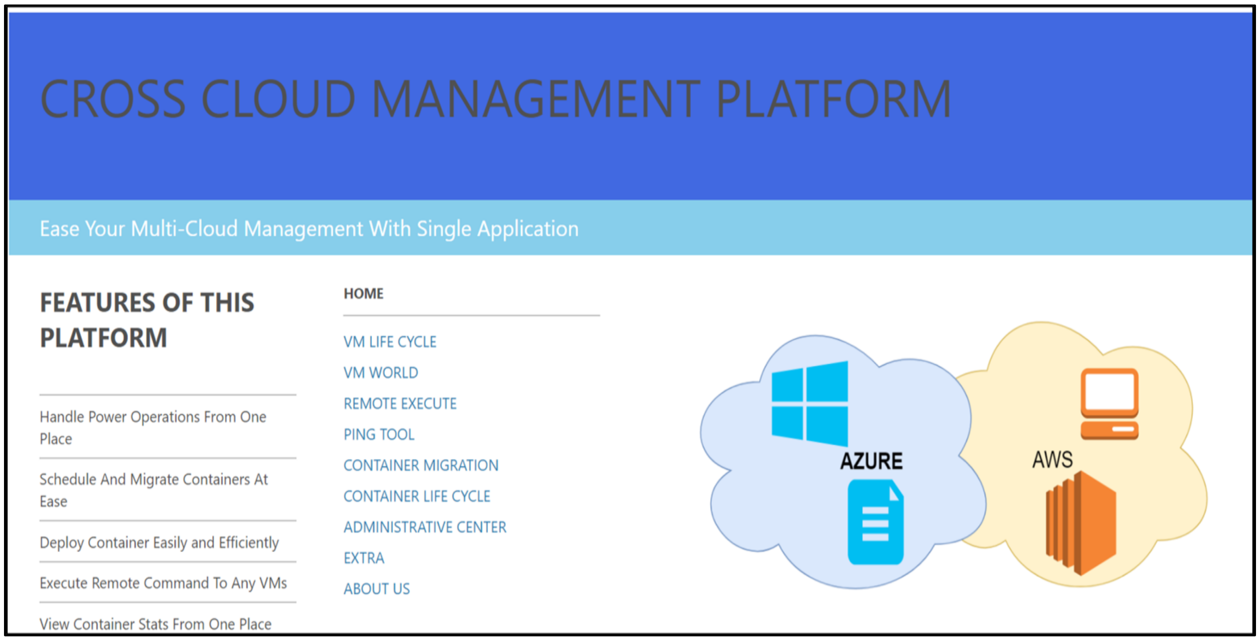


Figure – 9.0.1 - Architecture of cross cloud platform

Power operations such as starting, restarting and stopping any VMs running in AWS and AZURE clouds is controlled from the single plane using the appropriate Azure and AWS APIs that are in-built with PowerShell. Ping tool is provided to check the connectivity among the VMs in the same cloud or different cloud from the platform itself, rather than logging into each VMs individually and checking the connectivity. Remote execution of commands and the live output from the console is provided by the platform, to reduce the overhead of managing each VMs provisioned in different clouds individually. This is achieved using the PowerShell with SSH called "POSH" scripts. Information related to containers (Dockers) that are deployed across in multiple VMs and their resource usage statistics can be viewed in the single plane in this cross-cloud management platform.



CROSS CLOUD MANAGEMENT PLATFORM

Ease Your Multi-Cloud Management With Single Application

FEATURES OF THIS PLATFORM

- Handle Power Operations From One Place
- Schedule And Migrate Containers At Ease
- Deploy Container Easily and Efficiently
- Execute Remote Command To Any VMs
- View Container Stats From One Place

Figure 9.0.2 – Home Page of cross cloud platform

8.1 Policies used in container migration

The main purpose of this part is the migration of containers based on policies between VMs in the same cloud or between different cloud (AWS/AZURE) according to the user need. There is a shortage for these kind of systems or platforms in the market, to handle different user requests regarding container migration. Also, in this system containers are placed inside the VMs running in the cloud. There are several advantages of running containers inside the VM. They are, increased portability, provides a heavy layer of separation between the guest environment and host environment, increased security and increased stability. This part is mainly used by 3rd level team in an organization such as Infrastructure team to handle the migration of containers based on policies in the same cloud or between different cloud according to end user need. For this user manageable interface is created in the cross-cloud management platform and the user has the freedom to select which VM to VM, the migration of container takes place. For the communication and migration of containers between VMs in two different clouds (AZURE and AWS), a VPN based tunnel is used. By using VPN based tunnel more protection is provided for container migration as traffic is encrypted.

Reasons for running container within VMs are,

- Increased portability
- Heavy layer of isolation between guest environment and host server
- Increased security
- Increased stability

There are several reasons for the migration of containers they are for,

- High availability (HA)
- To do hardware/System upgrade in the VM which host the containers
- Cost reduction by moving container from one VM to another VM in different cloud
- For testing and evaluation purposes for different application running in containers across different cloud
- To improve the workload balance

The policies are,

1. Co -Exist policy

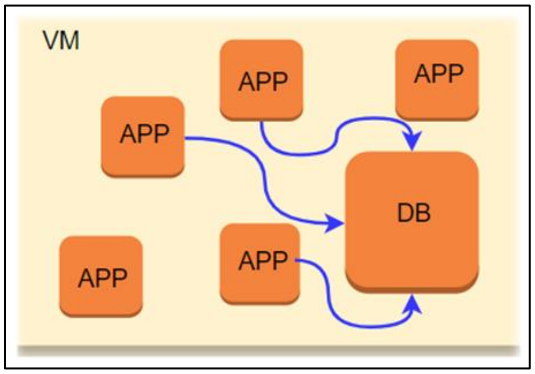


Figure 9.1.1 - Containers that are existing together within a VM

Some containers should co-exist with other containers for efficient performance, for E.G – DB containers and App containers must co-exist within the same VM to increase the performance. If they are moved to different VMs then performance may be degraded

2. Cannot-Co Exist Policy

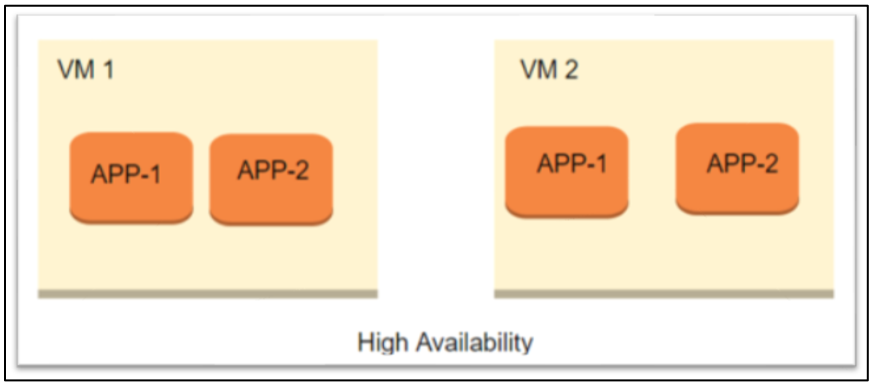


Figure 9.1.2 - Redundant containers that are running in different VMs

Cannot Co-Exist policy defines that certain containers cannot be located within the same VM, for E.G – two VMs that are created for redundancy purposes cannot be migrated to same VM, so that redundancy purpose will be lost if one VM is down.

3. VM specific containers policy

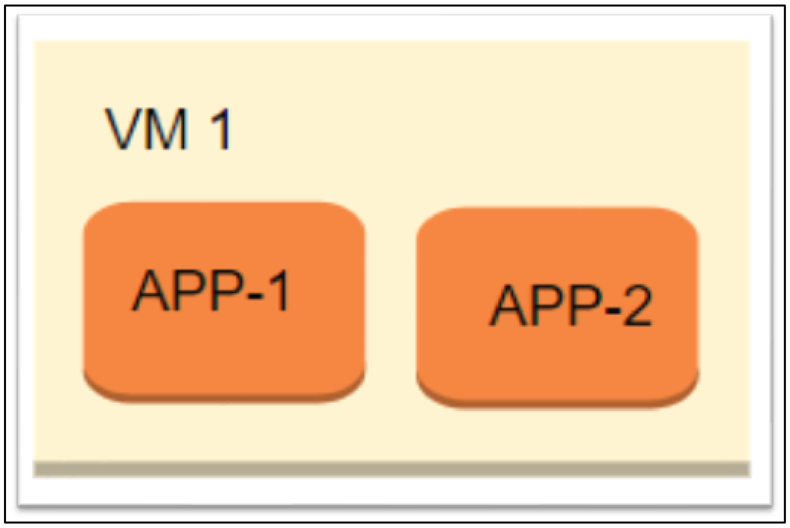


Figure 9.1.3 - Containers that are tied to a specific VM

VM specific containers policy makes sure that particular container always exists within that particular VM only, to meet its requirements, so the container cannot be migrated to another VM.

4. Resource reservation policy

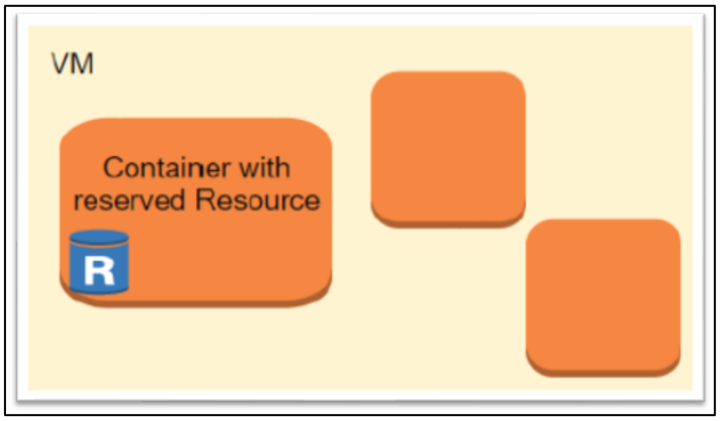


Figure 9.1.4 - Container with reserved resource tag

Some containers can be reserved for required resources already. Therefore, before the migration the target host should be validated against the required resources, if its meets the requirement only migration will take place

5. Scheduling policy

It helps to automate the migration of the containers in a pre-defined schedule configured by the user

8.2 Container Migration

Docker images are stored in the read-only layer and the docker containers that are created based on that images are stored in the read-write layer (Empty layer) on top of the read-only layer. This file system is called as Union file System. Whatever the changes or modifications made to the original container, will be copied from read-only layer and stored in the read-write layer. Once the running container is removed, all the modified data also will be lost with it. To overcome this, Docker introduced the concept of named volumes. Named volumes, help to map the internal folder of a container to a folder in the host[9].

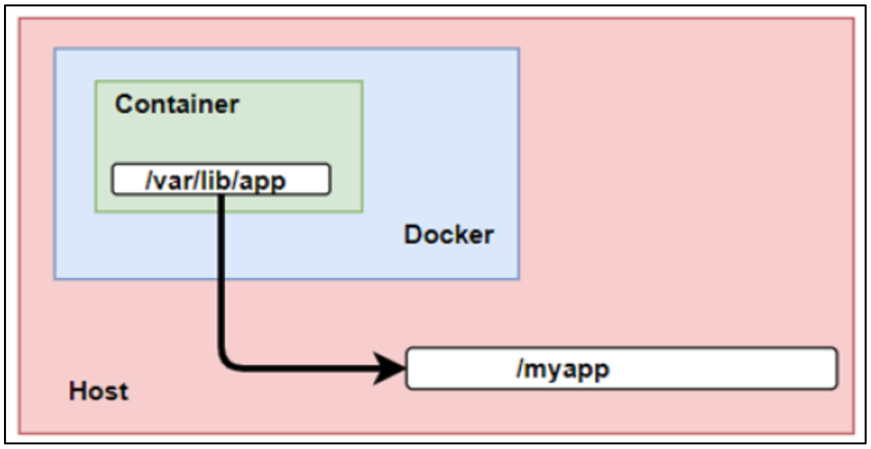


Figure 9.2.1 – Volume in Docker container

Even if the running container is removed, data will be persisted in the volume and it can be reused by another container, whenever needed. For the migration of containers between VMs in the same cloud and different cloud in this platform, uses the named volume approach. All the migration related policies are stored inside the text file within a folder identified by the container name in the host where this cross-cloud management platform is installed. Even though, the policies are stored in the text format, unauthorized people cannot gain access to it, as these text files are stored along with this cross cloud management platform in a secured Windows operating system environment, where the only 3rd level team members of an organization have access to it. All the containers and VMs are uniquely named while they are created. “Co-Exist” policy specify the containers names, that are need to be kept along with the source container (container which is going to be migrated). Before migration is initiated, this policy is used to check whether there are any containers specified in this policy are running along with the source container inside the source VM. If there are any, then migration process will be terminated, and a log entry will be created. If not, next policy in the line will be validated.

“Cannot Co-Exist” policy specify the containers names, that should not be kept along with the source container. This policy is used to check whether there are any containers specified in this policy are running in the destination VM where migration has to be done. If there are any specified containers, then migration process will be cancelled, and log entry will be created. VM specific container policy is used to check whether the source container is tied to certain VMs only. If the destination VM is not in the list, then migration will be terminated. Resource reservation policy is used to ensure the container meets its minimum computing requirements. Destination VMs are checked for the RAM and HDD and compared with the configured resource policy of the source container, if it satisfies the requirement, then other policies will be checked, if not migration will be cancelled.

Scheduling policy is introduced to automate the migration process of containers whenever required according to the convenience. All the pre-request policy will be check out and if they are validated, then “cron job” will be created on the source VM to automate the migration of container from it to destination VM, on the scheduled time. It’s also possible for the containers to exist without any configured policies, for example stand-alone containers, which are not bound to any specific VMs or to the other containers. In that case, such containers are migrated without checking any pre-requests. All the mentioned processes are carried out by PowerShell scripts and Shell scripts to control the remote Linux VMs. In this cross-cloud management platform, following additional capabilities are also provided with the container migration, they are,

Removing the container completely from the source VM and migrating it to the destination VM and starting it up

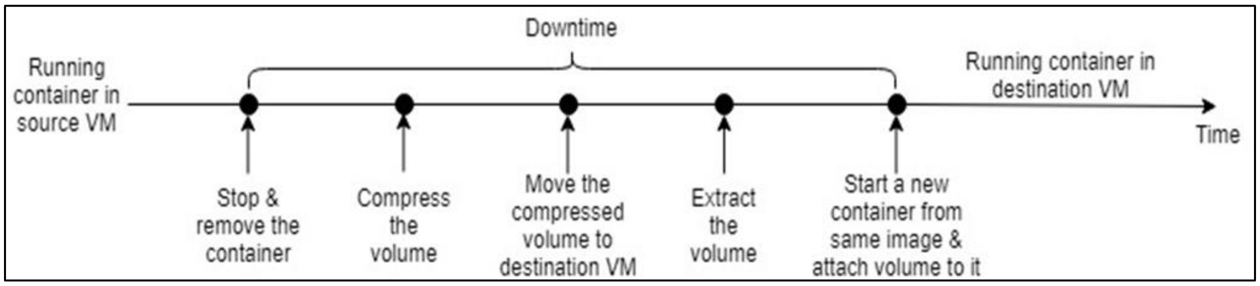

Downtime
Running container in source VM
Running container in destination VM
Time
Stop & remove the container
Compress the volume
Move the compressed volume to destination VM
Extract the volume
Start a new container from same image & attach volume to it

Figure 9.2.2 – Phases of Container migration

Keeping the source container as it is and migrating it to the destination VM and starting it with the different name

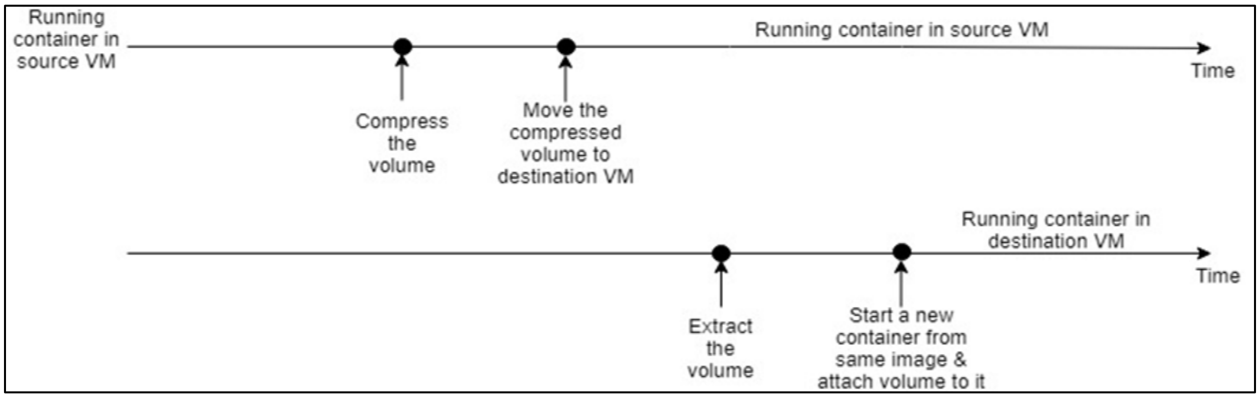

Running container in source VM
Running container in source VM
Time
Compress the volume
Move the compressed volume to destination VM
Running container in destination VM
Time
Extract the volume
Start a new container from same image & attach volume to it

Figure 9.2.3 – Phases of Container migration while keeping the source container intact

If all pre-requests are validated and if the container is eligible for the migration, then the volume of the container is compressed in the source VM and copied to the destination VM over the network and extracted there. After that, a new container from the same source image is created with the same name or different name (According to the options selected) with the migrated volume attached to it to the same location as in the previous container. Log files are stored in the format of text file within a folder identified by the container name in the host where this cross-cloud management platform is installed (Same folder where the policies related to a container are stored). Log files are named according to the initialization time of migration of the containers to easily identify them. For example, if the migration is initiated on a container at 07-29-2018 at 4.05 AM, then log file will be named as "Log_07_29_2018-4-05-00-AM"

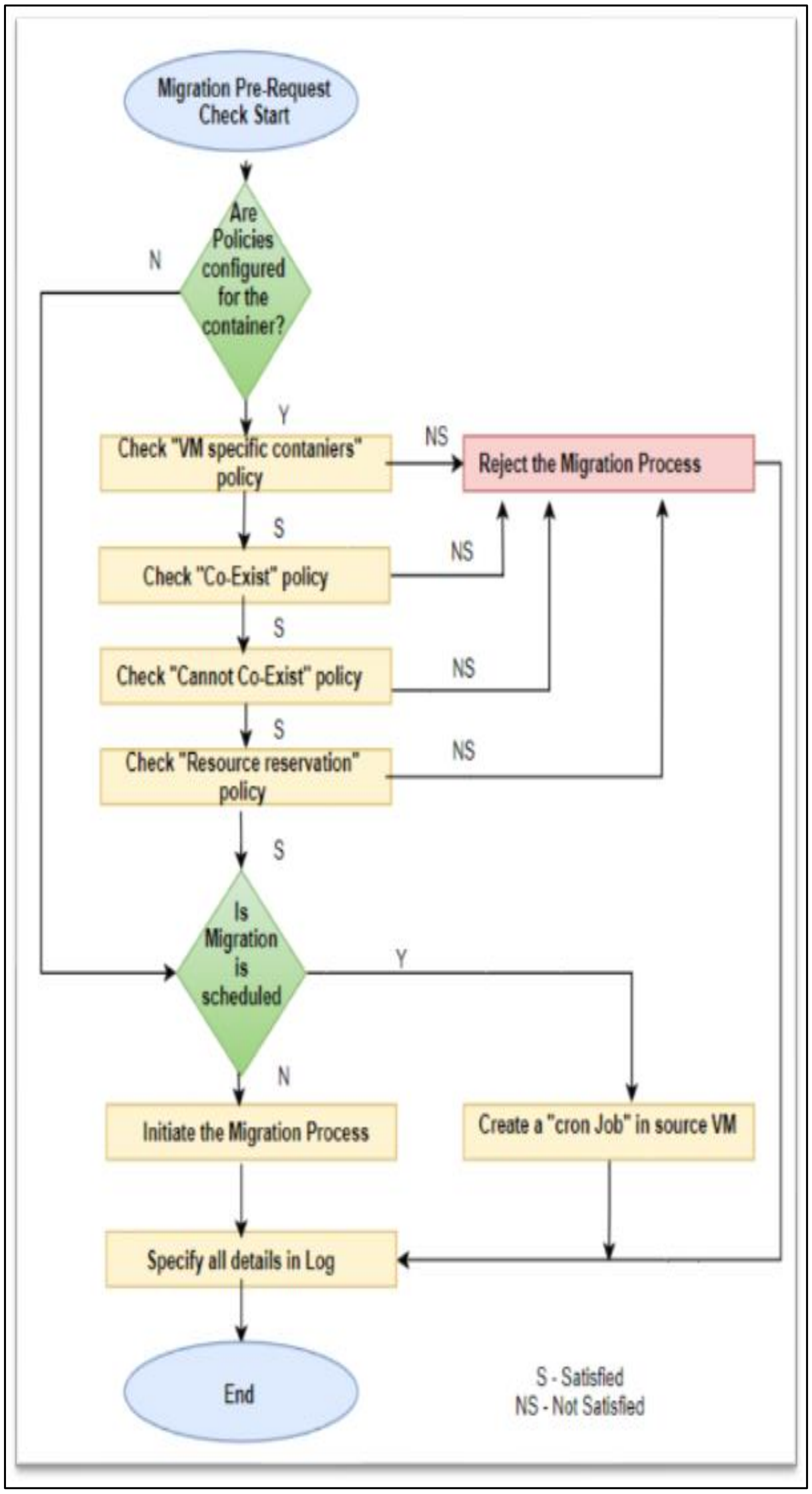
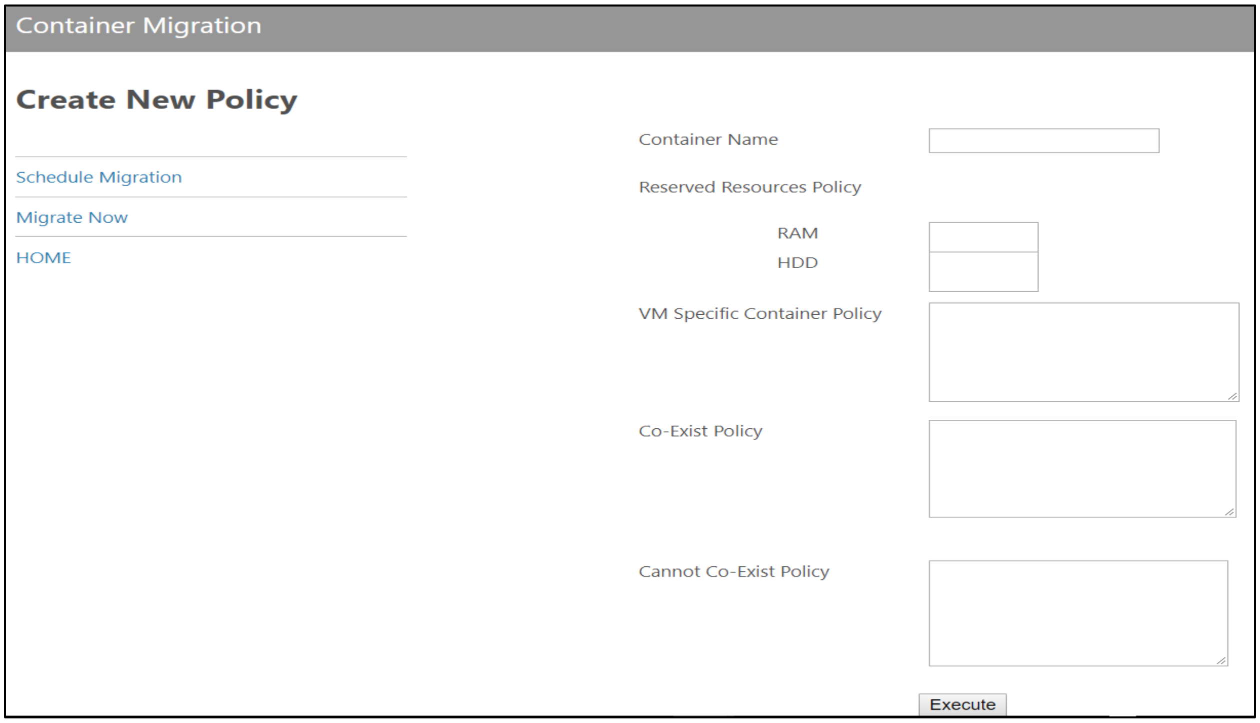

Migration Pre-Request Check Start
Are Policies configured for the container?
N
Y
Check "VM specific contaniers" policy
NS
Reject the Migration Process
S
Check "Co-Exist" policy
NS
S
Check "Cannot Co-Exist" policy
NS
S
Check "Resource reservation" policy
NS
S
Is Migration is scheduled
Y
N
Initiate the Migration Process
Create a "cron Job" in source VM
Specify all details in Log
End
S - Satisfied
NS - Not Satisfied

Figure 9.2.4 – Flow chart of handling policies


Container Migration

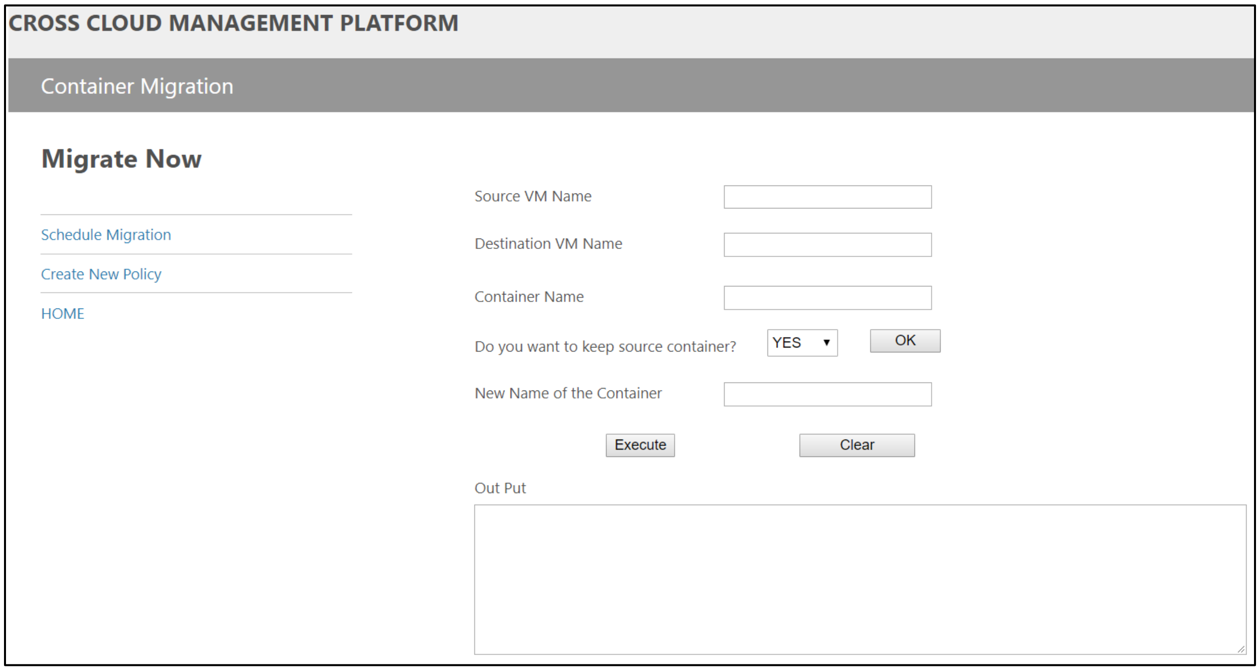
Create New Policy

- Schedule Migration
- Migrate Now
- HOME

Container Name	
Reserved Resources Policy	
RAM	
HDD	
VM Specific Container Policy	
Co-Exist Policy	
Cannot Co-Exist Policy	

Execute

Figure 9.2.5 – Interface to handle new policy creation


CROSS CLOUD MANAGEMENT PLATFORM

Container Migration

Migrate Now

- Schedule Migration
- Create New Policy
- HOME

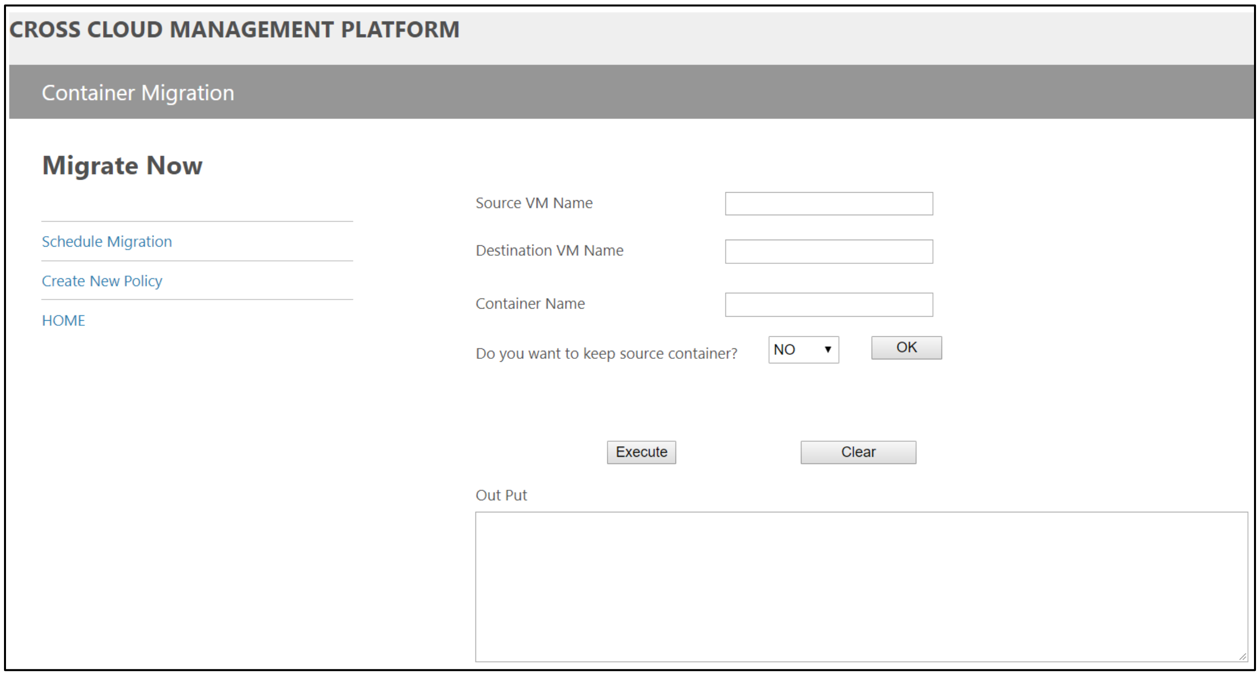
Source VM Name	
Destination VM Name	
Container Name	
Do you want to keep source container?	YES
New Name of the Container	

OK

Execute Clear

Out Put

Figure 9.2.6 – Interface to handle container migration between cloud (Cloning the container)


CROSS CLOUD MANAGEMENT PLATFORM

Container Migration

Migrate Now

Schedule Migration

Create New Policy

HOME

Source VM Name

Destination VM Name

Container Name

Do you want to keep source container? NO

Figure 9.2.7 – Interface to handle container migration between cloud

8.3 Connection between AWS and AZURE Cloud

Inter-cloud communication between AWS and AZURE cloud is created based on VPN tunnel. Using VPN tunnel provides extra layer of protection through encryption. A virtual private network is created in AZURE composing the required VMs (all VMs are in private range IP address network) and that network is connected to the internal part of the VPN gateway provided by the AZURE. The outside of AZURE VPN gateway is connected to the other part (AWS) using public IP. Since AZURE VPN devices are not compatible with AWS cloud VPN devices directly, a 3rd party VPN software named “strongSwan” is used in the other end as VPN gateway. “strongSwan” is an open source IPsec-based Virtual Private Network (VPN) solution for Unix based operating systems. It is used in this platform mainly for its simple configuration, strong authentication and strong encryption method [17]. “strongSwan” is installed in an EC2 instance (Edge VM) that will act as AWS VPN gateway and it helps to route traffic between AWS EC2 instances and AZURE VMs. In AWS cloud, a virtual private cloud (similar as virtual private network in AZURE) is created with the required VMs (all VMs are in private range IP address network). After the connection is created, all AZURE and AWS VMs have end to end connectivity to each other even in they are in different clouds and in different private subnets.

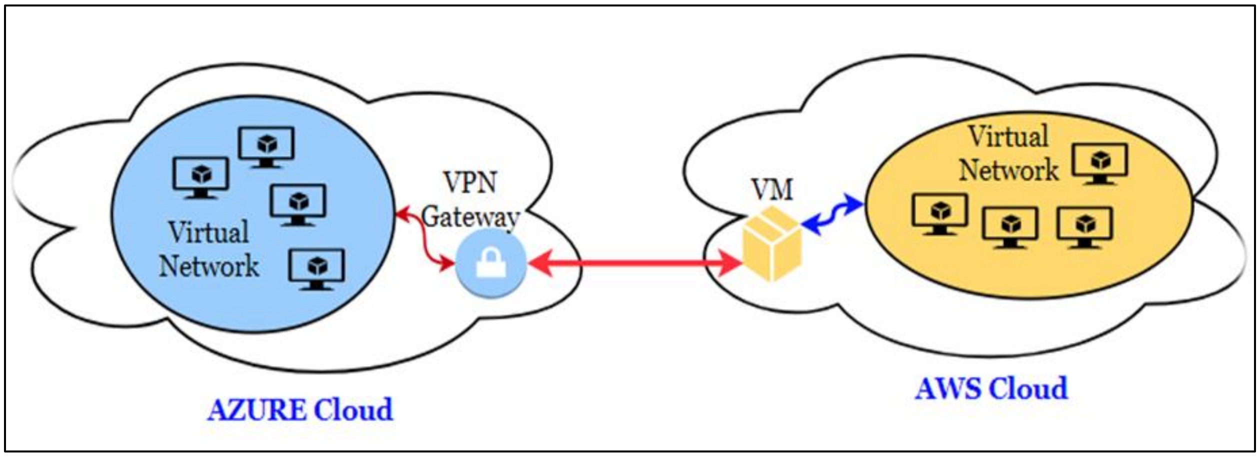


Figure 9.3.1 – Inter cloud communication between AWS and AZURE

8.4 Service offering for Docker Containers

Service offering to different containers is applied using Traffic Control (TC) in Linux. Service offering are,

- Gold (High Priority)
- Silver (Medium Priority)
- Standard (Low/Default Priority)

Every Docker containers that are deployed within the VM is interconnected through Docker Bridge. By applying traffic shaping policy to the Docker Bridge, the traffic can be controlled to each of the container.

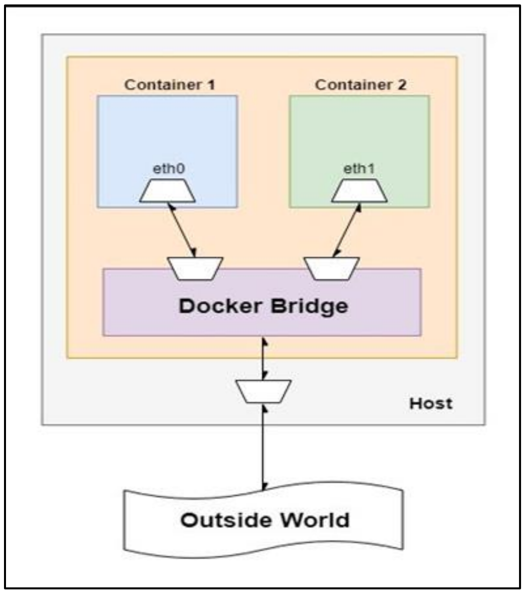


Figure 9.4.1 – Docker Bridge

Priority queuing discipline is attached to Docker bridge. Priority queuing discipline is a classful queuing discipline where subclasses can be added. Priority queuing discipline will ultimately create 3 classes and stochastic fairness queuing (SFQ) is added to 3 classes created under it. SFQ is used to ensure that different traffic in each traffic class get its fair share.

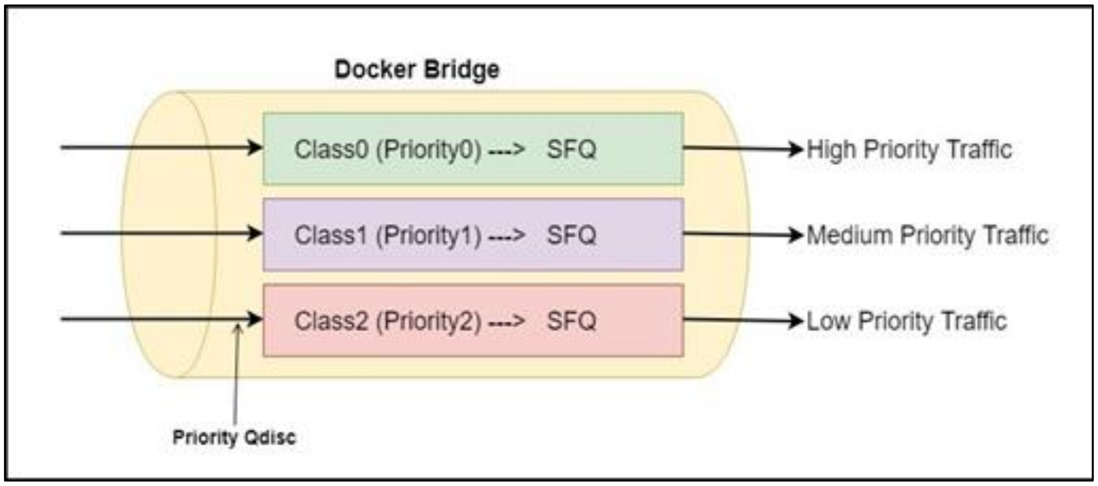
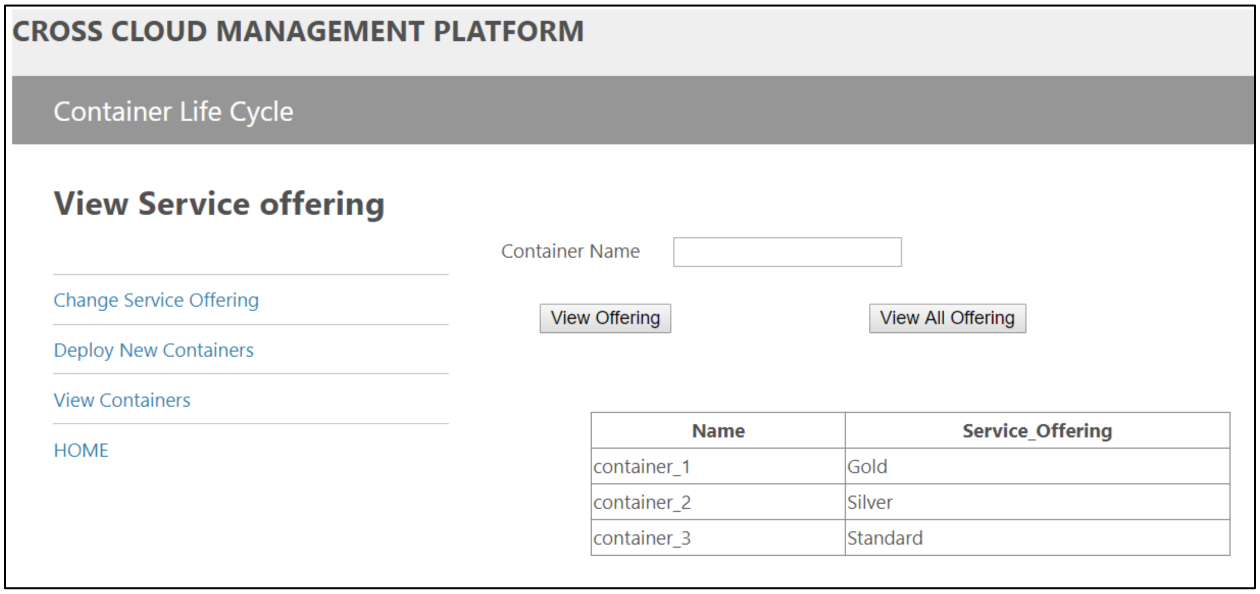


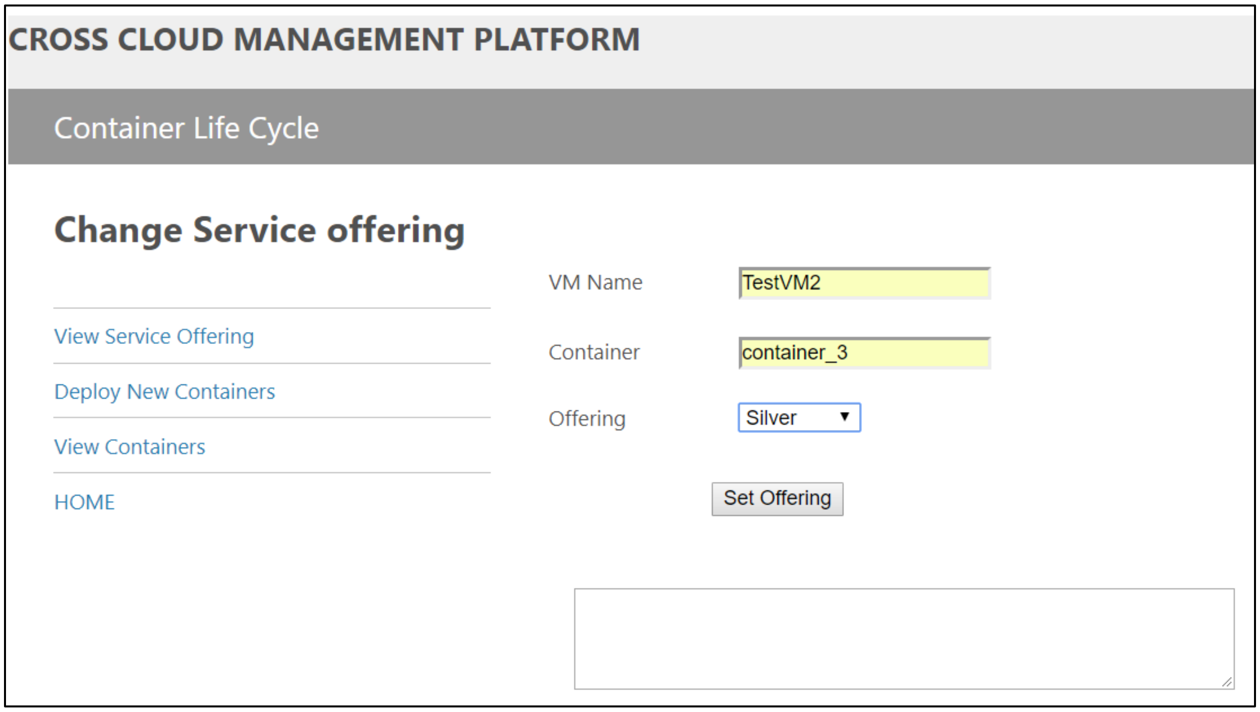
Figure 9.4.2 – Queuing disciplines to categorize the priority of traffic

Priority 0 is used for Gold level service offering, Priority 1 is used for Silver level service offering and Priority 2 is used for Standard level service offering. Each Docker container traffic is identified by adding a mark to its network packet. Then traffic is filtered/shaped by using the mark in the network packet.



Name	Service_Offering
container_1	Gold
container_2	Silver
container_3	Standard

Figure 9.4.3 – Interface to view all service offering of containers


CROSS CLOUD MANAGEMENT PLATFORM

Container Life Cycle

Change Service offering

VM Name: TestVM2

Container: container_3

Offering: Silver

Figure 9.4.4 – Interface to set QoS to different containers running in cloud

8.5 Policies based best VM selection to deploy new container

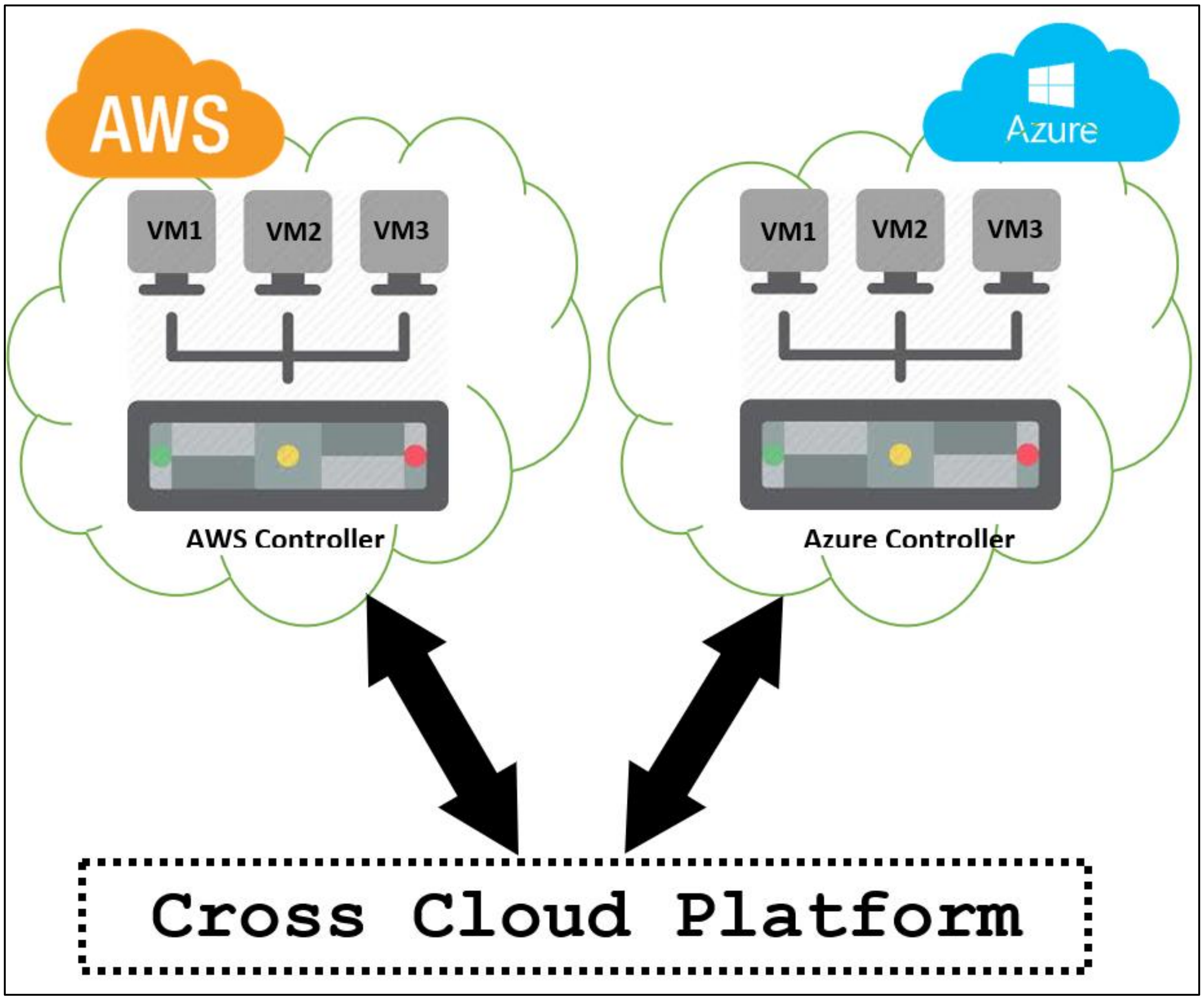


Figure 9.5.1 – Controllers in Cloud

Both cloud environments may run many virtual machines. Among those virtual machines one specific virtual machine reserved as controller VM which having capability of communicating other running virtual machines. This specific controller will be placed in both cloud environments. The main purpose to use stated controller is to communicate with other running virtual machines.

The best VM selection process is handled in 2 stages. The first part is handled within the system where the cross cloud management platform is installed, and the other part is handled in the controller VM inside the specified cloud. The user (3rd level team member) provides the required memory (in MB) and the disk capacity (in GB) for the container that he/she needs to deploy. Then the user will select the policy to be used in the search of best VM. Depending on the policy selected, the VM list is filtered and passed to the controller VM in the specified cloud with the required memory, disk capacity, container image name and the container name. Then by using the VMs list received, the controller VM will collect the required details such as memory, disk capacity, Network I/O usages of the respective VMs in that cloud and filter the VMs based on the passed parameter and choose the VM with lowest Network I/O as the best VM to deploy container. Since the memory, CPU usages are dynamically changing values, a shell script is scheduled to run on all deployed VMs continuously, to collect the average values of memory, CPU and disk capacities of the VMs for every 10 minutes. These average values only used by the controller for the selection of best VM. In order to collect the Network I/O details of a VM, "iftop" tool is used.

The main polices used in container deployment are,

1. Default Policy
2. Include VM Policy
3. Exclude VM Policy

Default policy means that the controller will search for the best VM from all the available VMs in the cloud. Include VM policy ensures that the search is span in the specified VM list only. Exclude VM policy is to exclude some VMs from the search for the best VM. These policies are really helpful in the environment, where different categories of VMs are found. For example, database VMs, production VMs, development VMs and training VMs. Service offering policy is used to configure the service level of a container such as gold, silver and standard that are running/deployed within VMs.

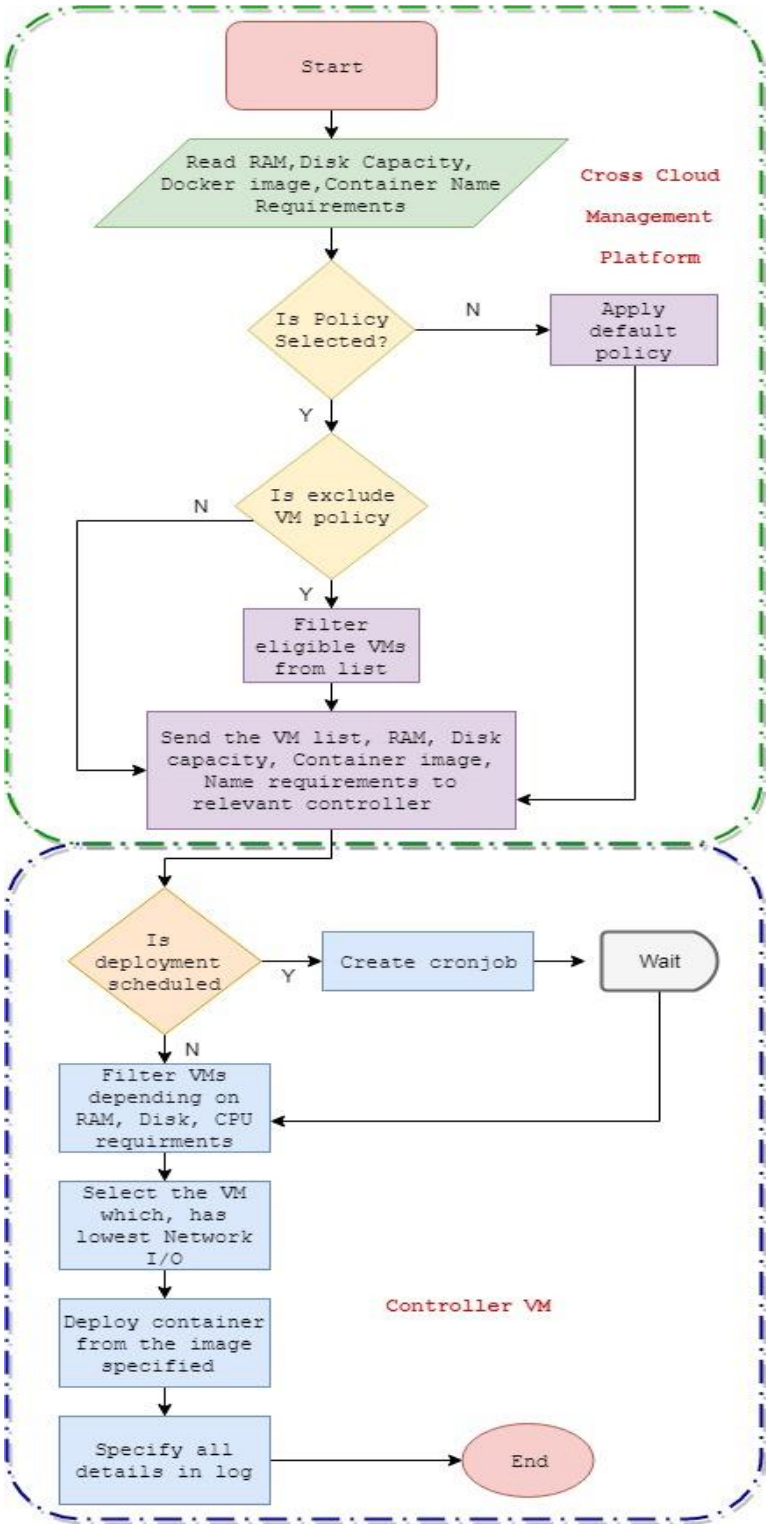
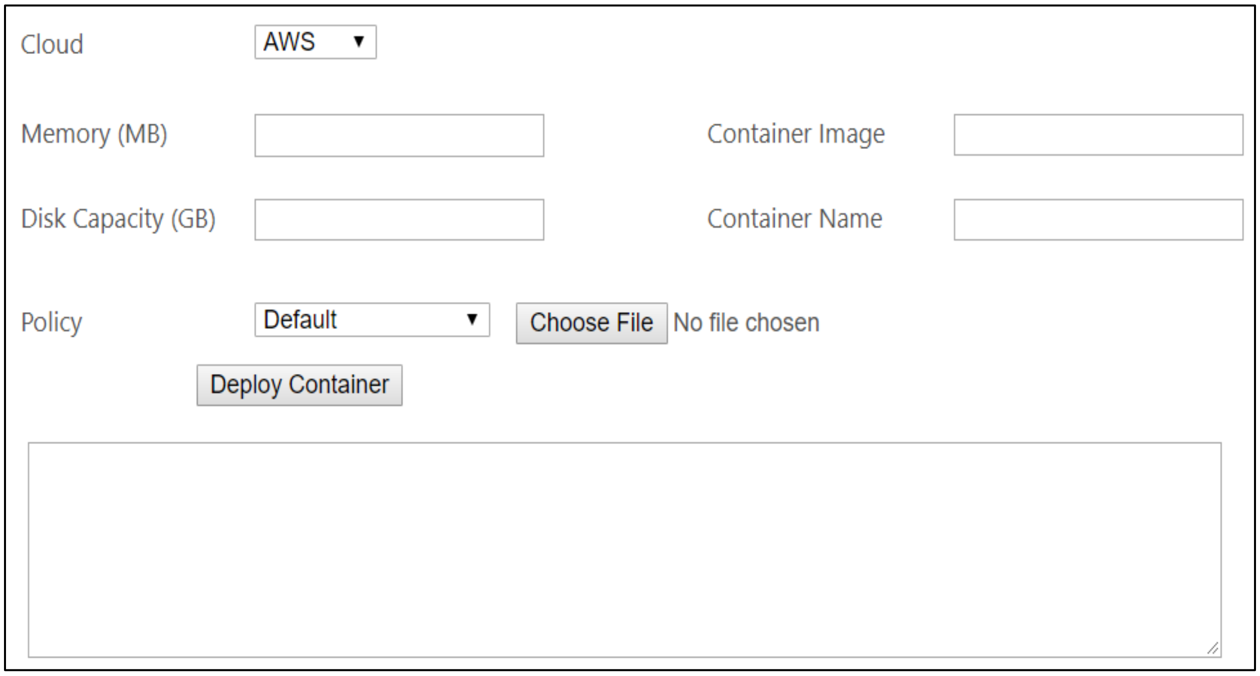

Start
Read RAM, Disk Capacity, Docker image, Container Name Requirements
Cross Cloud Management Platform
Is Policy Selected?
N
Apply default policy
Y
Is exclude VM policy
N
Y
Filter eligible VMs from list
Send the VM list, RAM, Disk capacity, Container image, Name requirements to relevant controller
Is deployment scheduled
Y
Create cronjob
Wait
N
Filter VMs depending on RAM, Disk, CPU requirments
Select the VM which, has lowest Network I/O
Controller VM
Deploy container from the image specified
Specify all details in log
End

Figure 9.5.2 – Flow chart to find the best VM in the cloud and to deploy new containers



Cloud	AWS		
Memory (MB)		Container Image	
Disk Capacity (GB)		Container Name	
Policy	Default	Choose File No file chosen	

Figure 9.5.3 – Interface to search for best VM in the cloud depending on user parameters

8.6 Blockchain Based SDN controller

A typical SDN architecture would consist of three layers, namely:

1. The application layer.
2. The control layer.
3. The infrastructure layer.

The application layer consists of applications that are meant to either control the network or manipulate the network as a part of accomplishing a bigger goal such as remote desktop connections.

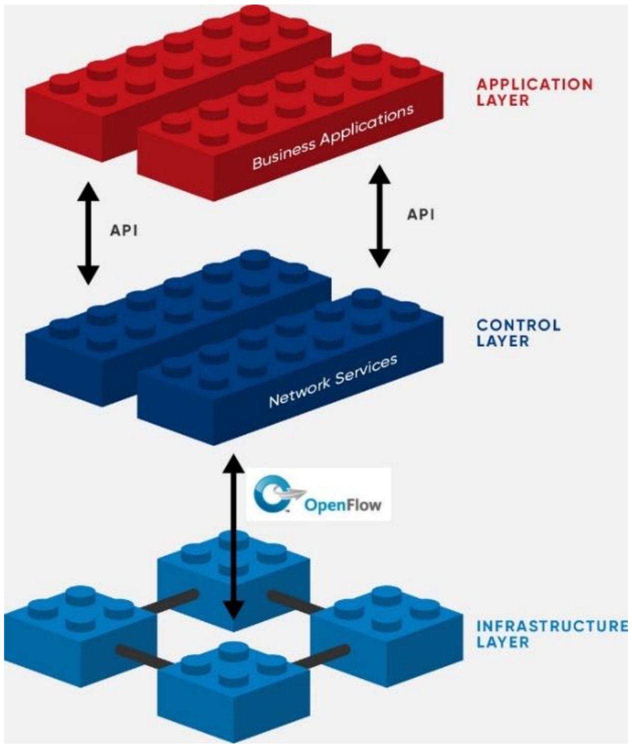


Figure 9.6.1 - The architecture of a Software Defined Network

The control layer typically contains an SDN controller. There are several OpenFlow controllers such as NOX, POX, OpenDayLight, FloodLight and Cisco ACI. The infrastructure layer is composed of OpenFlow enabled switches such as the Open vSwitch.

This application layer is allowed to control the infrastructure layer through the control layer. The control layer provides two APIs.

1. Northbound APIs
2. Southbound APIs

The Northbound APIs allow the application layer to interact with the control layer, whereas the Southbound APIs facilitate the communication between the control layer and the infrastructure layer.

To achieve the objectives mentioned in the antecedent section, the control layer would be replaced with a private blockchain. The application layer and the infrastructure layer will both interact with the blockchain.

Using private-public key pairs, the interacting parties can be authenticated, thus, rectifying vulnerabilities that arise from the lack of switch and controller authentication. The blockchain will also serve as an immutable ledger storing the history of the state changes of the flow-table, making the task of performing security auditing easier.

Applications that want to control the infrastructure layer can write to the blockchain and the switches that the applications intended to communicate with can read from the blockchain. The reverse shall be true when the switches want to interact with the applications.

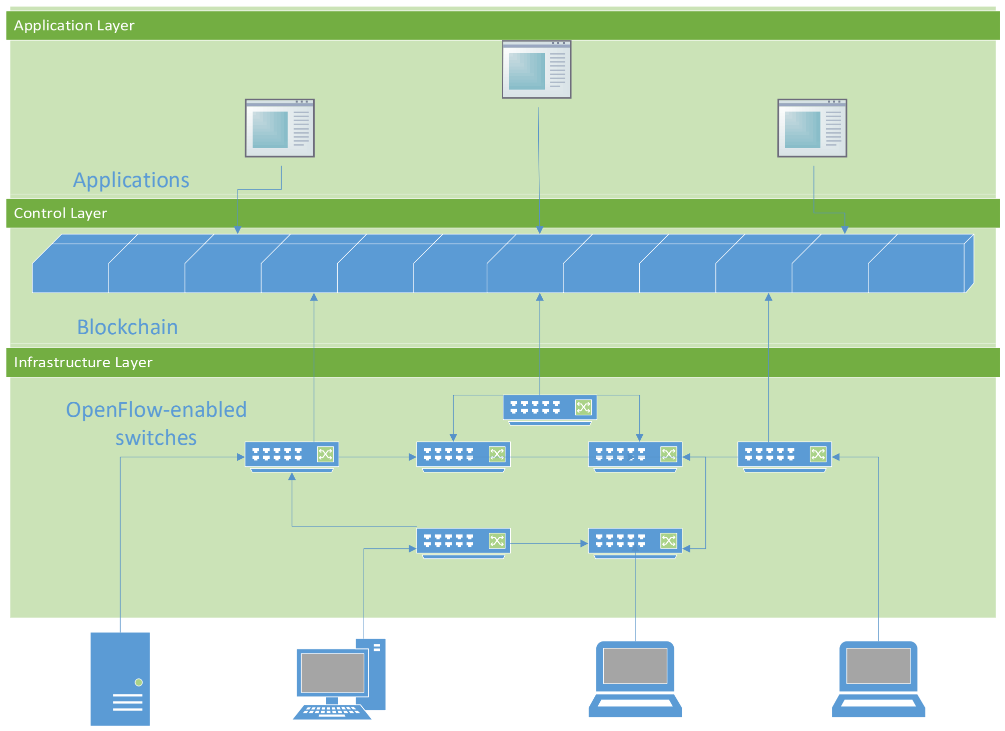


Figure 9.6.2 – Proposed architecture

Employing this technology offers many benefits. First and foremost, since the blockchain is immutable, the flow-table changes written to the blockchain would provide a history of the state of the flow table changes. Hence, security auditing can be done easily and in case of a security attack on the network, the attacker will not be able to erase his/her traces, providing the security engineers much-needed information about the attack.

Second, since a private-public key pair is needed to communicate with the blockchain, the confidentiality of the communication is ensured while confirming the authenticity of the sender. A writer needs to use its private key to initiate a communication. Thus, the authenticity of the communicator is confirmed. When a writer sends a message to a certain switch, the public key of the recipient is used. Therefore, only the intended recipient will be able to read the message, warding off any possibilities of a Man-in-the-Middle attack.

Third, a blockchain allows multiple parties to communicate. This allows multiple applications to avail themselves of the distributed control layer. Although even a traditional SDN environment allows multiple applications to manipulate the network, it must be pointed out that, such communications will have to take place through a single controller. This makes the network vulnerable to a single point of failure. When blockchain is used, since a blockchain is distributed and each participating node has a copy of the blockchain, the entire network will have to be taken down to cause a failure, which is, of course, a paradoxical scenario since the blockchain would be purposeless without a network.

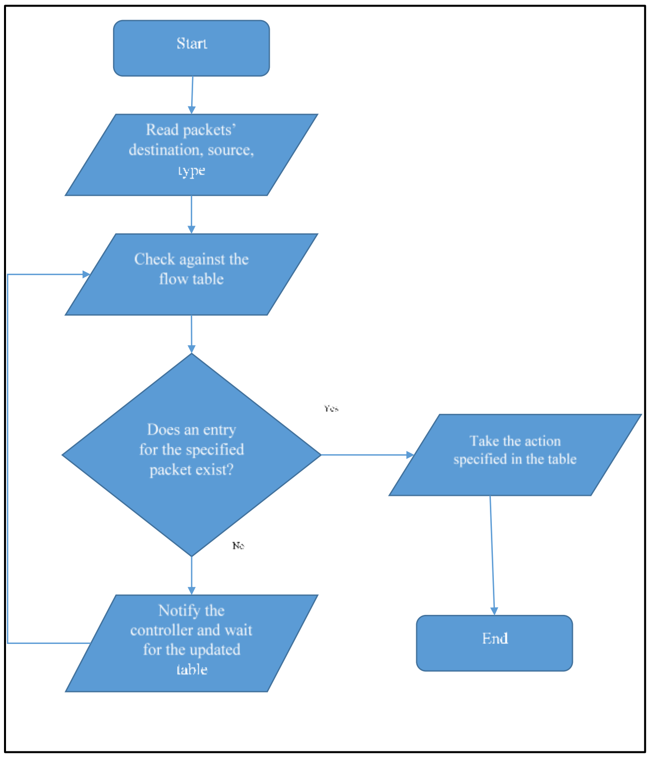

Start
Read packets' destination, source, type
Check against the flow table
Does an entry for the specified packet exist?
Yes
Take the action specified in the table
No
Notify the controller and wait for the updated table
End

Figure 9.6.3 – The flow used by switches in handling packets

The other advantage of this control layer is that since a blockchain is immutable, both the controller and the switches read from and write to the blockchain, and the storage of flow-table states is distributed across the blockchain, the controller's view of the rule-state is the same as those of the switches. Hence, the controller and the switches will always be in sync.

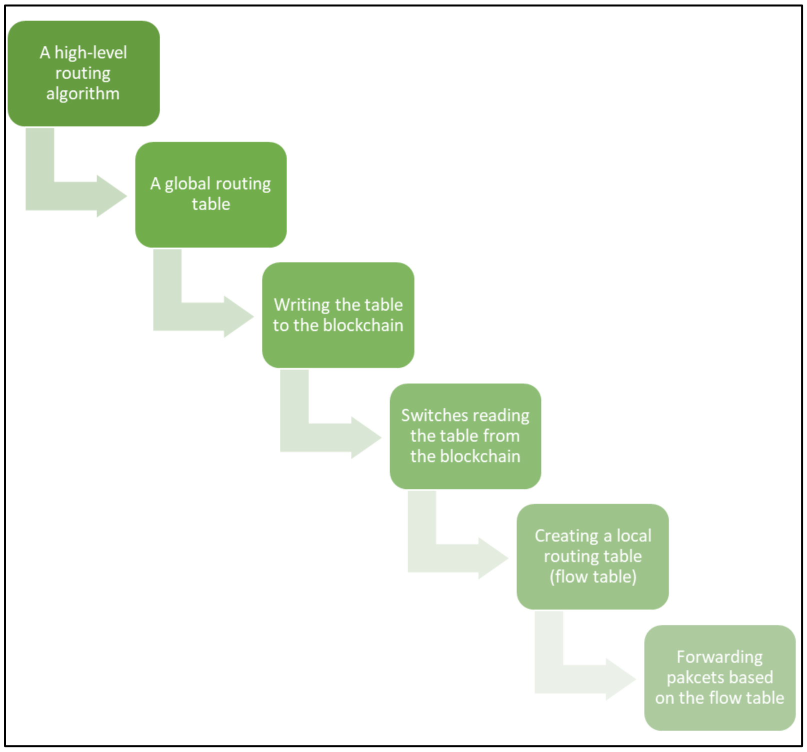

A high-level routing algorithm
A global routing table
Writing the table to the blockchain
Switches reading the table from the blockchain
Creating a local routing table (flow table)
Forwarding pakcets based on the flow table

Figure 9.6.4 - The process flow proposed by the architecture

Ethereum is a blockchain platform for building decentralized applications. This platform allows a developer to deploy an application in the Ethereum's public blockchain or in a privately run blockchain. The control layer of the SDN network can be built using Ethereum.

Open vSwitch is an OpenFlow enabled virtual switch that is primarily used in networking virtual machines. However, Open vSwitch also provides an application that can run on top of Linux. Hence, this can be used to develop switches that can interact with the blockchain.

Web3.js is a node module compatible with Ethereum that can be used a sample application that would interact and manipulate our SDN network. Since the Node programming environment

8.7 Testing and Implementation

8.7.1 Unit Testing

Every feature of this cross-cloud platform is thoroughly tested individually. Especially the scripts which handle the power operations of AWS and AZURE VMs, Display VM lists, Display container details are tested using PowerShell. The shell scripts to migrate the containers between AWS – AWS, AWS – AZURE, AZURE – AZURE and AZURE – AWS are also checked individually, and the desired output is obtained. Also, individually each policy in migration is checked out and the desired result is obtained. Further the functionality of controllers in each cloud is checked whether they are handling the communication between other VMs properly and selecting the best VM at that moment. Also, QoS handling scripts are checked with the different user selection and its confirmed that the containers are getting proper service offering according to the user selection

8.7.2 Integration Testing

All individual scripts are embedded into the cross-cloud platform with the relevant user interfaces. All functionalities such as power on/off operation, migration of containers, setting QoS to containers, finding best VM in cloud are checked and validated. After integration testing, some test cases are carried out as follows,

Test Case 01	Power On/Off Operation
Input	Cloud Name, VM Name
Output on success	Successful power off message
Output on failure	VM Name cannot be found

Table 9.1 – Test Case 01

Test Case 02	Migration of Containers between Cloud based on policies
Input	Source and Destination VM Name, Container Name
Output on success	Migration successful message
Output on failure	Policies which were failed to validate will be displayed

Table 9.2 – Test Case 02

Test Case 03	Setting QoS to containers
Input	Container Name, VM Name, Service offering
Output on success	Service offering updated message will appear
Output on failure	VM/Container name not found and QoS update failed error

Table 9.3 – Test Case 03

Test Case 04	Finding best VM in the cloud
Input	Cloud name, Disk space, RAM and the policy
Output on success	Best VM to deploy new container and the deployed container image id
Output on failure	Cannot find best VM message

Table 9.4 – Test Case 04

Test Case 05	Ping tool
Input	Cloud, Source IP, Destination IP
Output on success	Successful Ping output between two VMs
Output on failure	Failed Ping output

Table 9.5 – Test Case 05

Test Case 06	Remote Execute
Input	IP address, Command
Output on success	Successful output according to the command supplied
Output on failure	VM not found or fail message according to the command supplied

Table 9.6 – Test Case 06

8.7.3 Functional Testing

Functional testing ensured that each expected functions of the cross-cloud platform worked without any interruption or failure

8.7.4 Performance Testing

Migration time for containers based on policy validations are compared and it showed that the obtained values are closer to the expected values. Executing of remote commands using remote execute feature of the cross cloud platform also took the expected time range to return the output

9 RESULTS AND DISCUSSION

9.1 Container Migration using Cross Cloud platform

For testing purposes, containers are created from ‘MySQL’ image and a sample database called ‘sakila’ is restored inside the container [7]. All containers are created inside the VMs in both clouds (t2.micro type EC2 instances in AWS and B1s type VMs in AZURE). Before migrating the container, sample insert statements were executed against the database and the migration process is initiated from the source VM. After that it was possible to see the same container is up and running in the destination VM. When the select query is executed against the migrated container in the destination VM, all the last executed statements were retrieved. By this it is proved that the database container can be migrated between the 2 clouds without any data loss.

Following table illustrates, the time taken for the migration of the container from a source VM to destination VM in same cloud and between different cloud after all the policies are validated. Size of the container is approximately 220 MB.

Source VM Cloud	Destination VM Cloud	Average Time Elapsed For Migration
AZURE	AWS	28.5 s
AZURE	AZURE	45.5 s
AWS	AZURE	42 s
AWS	AWS	11 s

Table 10.0.1 – Migration time for containers

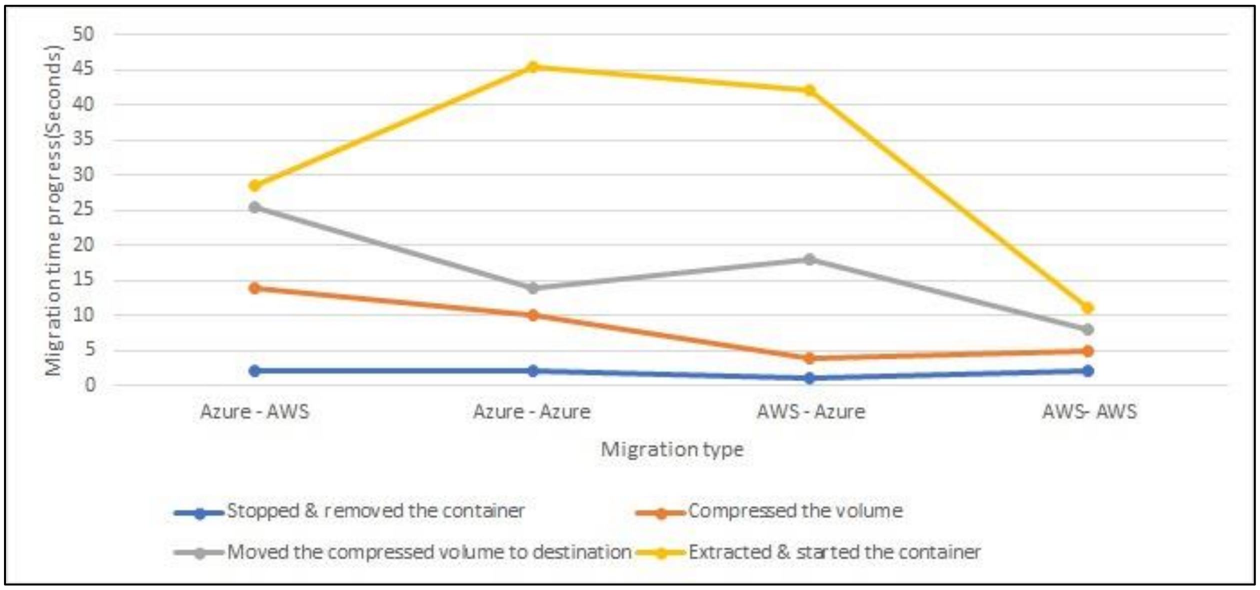
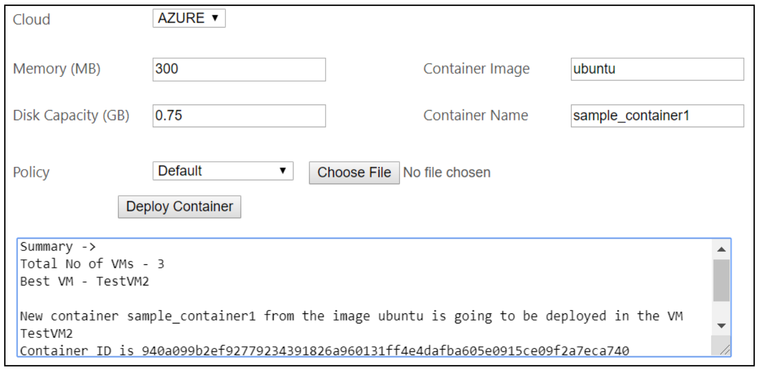


Figure 10.0.1 - Migration progress of container in different environments

In all the above VMs, ‘MySQL’ docker image is previously downloaded. If not, additional time required to download the ‘MySQL’ image needs to be considered. Average time spent to download the ‘MySQL’ image in AZURE is 59.33 s and in AWS is 16.35 s. All the above calculated time periods will vary depending on the service offering of the AZURE VMs and EC2 Instances in the cloud (For this experimentation Free-tier services only utilized) and the datacenter location where they are deployed.

9.2 Policy based containers deployment across multi-cloud

According to the figure 4, the values are passed into cross cloud platform. “Ubuntu” image is used as the deployment container image. Default policy is selected as the search policy.



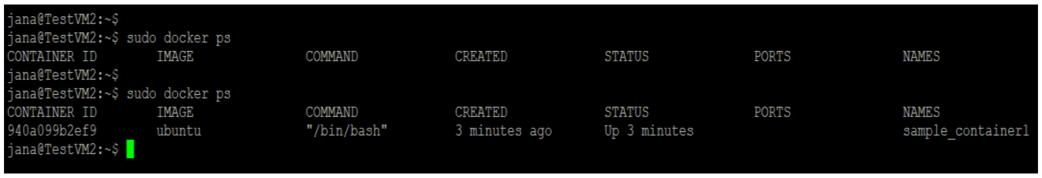
Cloud	AZURE ▾		
Memory (MB)	300	Container Image	ubuntu
Disk Capacity (GB)	0.75	Container Name	sample_container1
Policy	Default ▾		

```
Summary ->
Total No of VMs - 3
Best VM - TestVM2

New container sample_container1 from the image ubuntu is going to be deployed in the VM
TestVM2
Container ID is 940a099b2ef92779234391826a960131ff4e4dafba605e0915ce09f2a7eca740
```

Figure 10.2.1 – Best VM is found out in the cloud depending on the container requirement

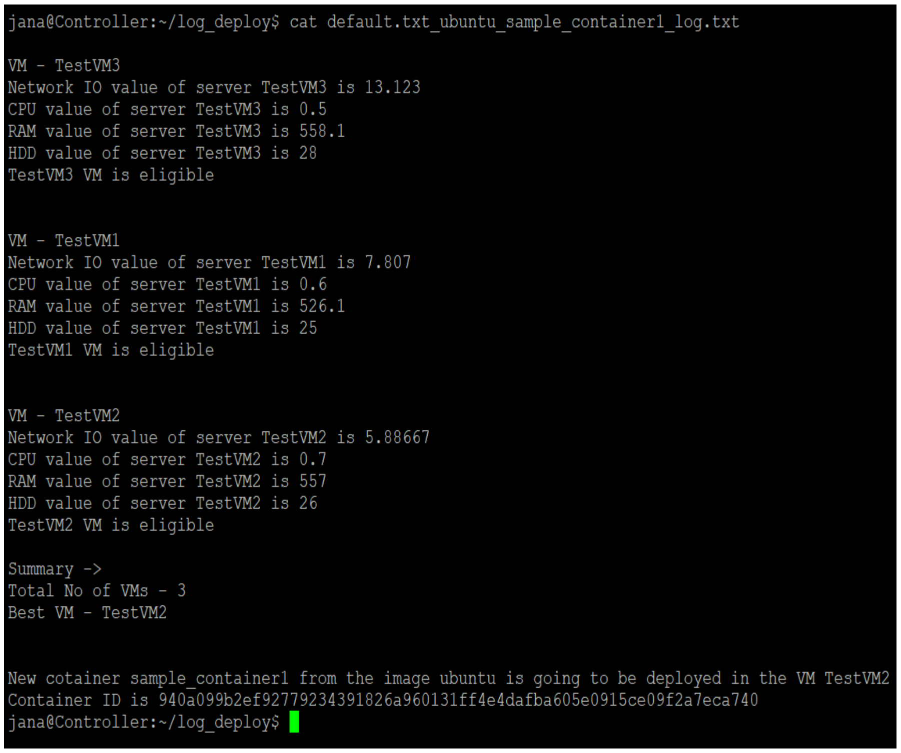
It is observed after few seconds period, the container named “sample_container1” from the image “ubuntu” is deployed in the best VM at that period in the AZURE cloud.



```
jana@TestVM2:~$
jana@TestVM2:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
jana@TestVM2:~$
jana@TestVM2:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
940a099b2ef9        ubuntu              "/bin/bash"         3 minutes ago       Up 3 minutes                            sample_container1
jana@TestVM2:~$
```

Figure 10.2.2 – New container is deployed in the best VM at that moment

Below figure 6, represents the log file from the controller and it is clearly identified that the container is deployed in the eligible VM with the lowest Network I/O, that is “TestVM2”.



```
jana@Controller:~/log_deploy$ cat default.txt_ubuntu_sample_container1_log.txt

VM - TestVM3
Network IO value of server TestVM3 is 13.123
CPU value of server TestVM3 is 0.5
RAM value of server TestVM3 is 558.1
HDD value of server TestVM3 is 28
TestVM3 VM is eligible


VM - TestVM1
Network IO value of server TestVM1 is 7.807
CPU value of server TestVM1 is 0.6
RAM value of server TestVM1 is 526.1
HDD value of server TestVM1 is 25
TestVM1 VM is eligible


VM - TestVM2
Network IO value of server TestVM2 is 5.88667
CPU value of server TestVM2 is 0.7
RAM value of server TestVM2 is 557
HDD value of server TestVM2 is 26
TestVM2 VM is eligible

Summary ->
Total No of VMs - 3
Best VM - TestVM2


New cotainer sample_container1 from the image ubuntu is going to be deployed in the VM TestVM2
Container ID is 940a099b2ef92779234391826a960131ff4e4dafba605e0915ce09f2a7eca740
jana@Controller:~/log_deploy$ 
```

Figure 10.2.3 – Log file inside the controller VM

By adopting this feature, the following advantages can be gained,

- Cloud wise search (AWS/AZURE) for the best VM to deploy new containers
- Configurable parameters (Memory/Disk Capacity), according to the container requirements
- Dynamic mapping between the user request and nodes (VMs) selection to ensure the reduced the number of messages & communication traffic in the cloud environments

9.3 Blockchain based SDN controller

Since the expected outcome of this research is not quantitative, the outcomes of the research work would be evaluated qualitatively using the summative and formative evaluation model. The research would be broken down to the following components and each component would be divided into modules as follows for the purposes of module testing:

1. Migrating containers to a virtual machine with redundant resources and shutting down unnecessary virtual machines to save power.

    1.1.Getting an accurate measure of the resource usage of virtual machines

    1.2. Finding virtual machines with redundant resources
    1.3. Migrating a container from one virtual machine to another
    1.4. Shutting down the extraneous virtual machines

2. Deploying a container across virtual machines running on different clouds.
    2.1. Analyzing the requirements of the developer and the resource usage of the virtual machines and suggesting the best virtual machines to the developer to deploy their container on.
    2.2. Allowing the developer to deploy a container on virtual machines across various cloud services.
    2.3. Providing a common user interface and higher-level abstraction that will help the developer accomplish the above easily.
3. Establishing a programable network in the form of a Software defined Network with a decentralized control plane.
    3.1. Creating a private blockchain to store the state of the flow-tables.
    3.2. Creating a north-bound API to allow applications to write to and read from the blockchain.
    3.3. Creating a south-bound API to allow switches to write to and read from the blockchain.
    3.4. Building a module that will allow an OpenFlow-enabled switch to communicate with the blockchain.
    3.5. Establishing a route using flow-tables to virtual machines running on different cloud services.
4. Creating an SDN based load balancer.
    4.1. Getting the chronological resource usage of virtual machines.
    4.2. Getting the user's location and finding the virtual machine that is in close proximity to the user.
    4.3. Finding the appropriate virtual machine based on the above details.
    4.4. Building flow-tables to route the user to the appropriate virtual machine.

10 COMMERCIALIZATION ASPECTS OF THE PRODUCT

The main benefits of this cross cloud platform for an organization are,

- Performance guarantee – Performance can be increased by deploying workloads across different clouds
- Availability – Geographically distributed cloud infrastructure (AWS and AZURE) can reduce the latency and unexpected outages
- Convenience – One central platform means, easy management
- Dynamic distribution of workload – Workload can be moved to a cloud location that is closer to the customer
- Resource provisioning – By using two or more clouds the user can able to switch his/her cloud deployment according to the need and demand.
- Reduced Costs – This architecture brings the services closer to the cloud users with lower costs. Thus, providing a way to migrate the workloads/containers and services from one VM to another VM.
- Avoid Vendor Lock-In – The end cloud users are not locked to a certain cloud service provider. They have the freedom to move the workload between different cloud environment.
- Legal regulation – There are some government regulations in the world that avoid the data to be moved outside of their countries. By adopting the cross-cloud architecture, other cloud service providers can host their data/applications in a cloud data center which situate in that country.
- Makes the life of a 3rd level team member much easier
- Secure VPN based communication between AWS and AZURE

11 INDIVIDUAL CONTRIBUTION

Student ID – IT15051608

Name – K. Janarthanan

Sub Title – **Policy based containers migration between virtual machines placed in same cloud or different clouds**

The main objective is to migrate the containers between cloud based on policies. So that it will make the life of the 3rd level team of an organization much easier. Migration is based on user defined policies such as co-existing policy where containers of some type must co-exist within the same virtual machine in order to get maximum performance, cannot co-exist policy where containers of some type cannot co-exist within the same virtual machines, resource reservation policy where containers minimum resource requirements are specified and host specific containers policy where certain containers are tied to specific virtual machines only.

Containers are created with the named volume approach, and during the migration the volume is compressed at source virtual machine and transferred to destination virtual machine in the same or different cloud. Then finally the volume is extracted, and a new container is started with the same image. This is provided with two options. First option is similar to cloning the container, where a container similar to source container is started at destination. Second option is stopping the source container and running at destination. Also, in order to manage everything in a central place a cross-cloud platform is built using relevant AZURE and AWS APIs. The platform is able to handle the operations such as, Policies based container migration between clouds, Handle power operations of VMs from one place, View all deployed VMs and Containers in a single plane, Ping tool to verify the connectivity among VMs (Inter/Intra cloud), View Containers stats from a VM deployed in any cloud, Execute remote command in VMs without logging into individual VMs, VPN based connection between the AWS and AZURE cloud environments and provide service offering to containers

Student ID – IT14009532

Name - Peramune P.R.L.C

Sub Title - **Setting QoS to Docker Containers**

The main objective is to set the QoS to containers according to the end user requirements. There are 3 level offered for the containers as follows,

- Gold offering – High priority
- Silver offering – Medium priority
- Standard offering – Low priority

In order to achieve the QoS, classful priority queuing discipline is used. SFQ is used to ensure that different traffic in each traffic class get its fair share. Each Docker container traffic is identified by adding a mark to its network packet. Then traffic is filtered/shaped by using the mark in the network packet.

Student ID – IT15026644

Name - Ranaweera A.T

Sub Title - **Policy based containers deployment across multi-cloud**

Best virtual machine for the deployment of container in the cloud is selected according to the new container requirement such as memory and disk capacity and after analyzing to the resource usage inside the virtual machines.

The policies involved in the best VM selection are,

Default VM policy – Best VM search is expanded across all the VMs in the selected cloud

Include VM policy – Best VM search is limited within the VMs in the policy in the selected cloud

Exclude VM policy – Policy to exclude some VMs from the search of best VM to deploy new container

Policies are passed through cross cloud platform and the best VM selection is done through the controllers placed in each cloud.

Student ID – IT14004414

Name - Theviyanthan K

Sub Title - **Decentralizing an SDN controller using blockchain**

SDN separates the control plane from the data plane of a network and centralizes the control plane in a single device allowing network developers to dynamically create different networking paradigms. However, the centralization of the control plane, while affording its fair share of benefits, makes SDN vulnerable to several security threats such as denial of service, and man-in-the-middle attacks. This research proposes a radical way to address such limitations by distributing the control plane across the network using the blockchain algorithm while allowing authorized applications to write to the blockchain to communicate with the data plane.

12 CONCLUSION

By implementing this cross-cloud platform more control and flexibility can be gained by a 3rd level team in an organization to deliver services to the end users. Thereby, it makes the life of 3rd level team members much easier. Some of the main features that are included in this platform are,

- Managing VMs (on/off/restart) from one place
- Monitoring the resource consumption of deployed containers across the clouds from one place
- Policies based containers migration across the clouds
- Executing remote commands in all VMs situated in AZURE and AWS
- Ping tool to verify the connectivity among VMs in the inter/intra clouds

Further by adopting the policies based container migration, it helps to reduce the complexity and increase the efficiency in managing the containers that are deployed across AZURE and AWS. Since these policies are configurable according to the need, it provides more flexibility.

Since a private-public key pair is needed to communicate with the blockchain, the confidentiality of the communication is ensured while confirming the authenticity of the sender.

A writer needs to use its private key to initiate a communication. Thus, the authenticity of the communicator is confirmed. When a writer sends a message to a certain switch, the public key of the recipient is used.

Therefore, only the intended recipient will be able to read the message, warding off any possibilities of a Man-in-the-Middle attack.

A blockchain allows multiple parties to communicate. This allows multiple applications to avail themselves of the distributed control layer. Although even a traditional SDN environment allows multiple applications to manipulate the network, it must be pointed out that, such communications will have to take place through a single controller.

This makes the network vulnerable to a single point of failure. When blockchain is used, since a blockchain is distributed and each participating node has a copy of the blockchain, the entire network will have to be taken down to cause a failure, which is, of course, a paradoxical scenario since the blockchain would be purposeless without a network.

13 REFERENCES

[1] Cuadrado, Navas, Duenas and Vaquero, “Research Challenges for Cross-Cloud Applications”, 2014

[2] Gianluca Zangara, Diego Terrana, Pietro Paolo Corso, Marco Ughetti and Guido Montalbano, “A Cloud Federation architecture”, 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing

[3] M.R.M. Assis and L.F. Bittencourt, “A Survey on Cloud Federation Architectures: Identifying Functional and Non-Functional Properties”, 2016 Journal of Network and Computer Applications

[4] Joy Su, " Hybrid Cloud vs. Multi-Cloud: What’s the Difference?" [Online]. Available: http://www.bmc.com/blogs/hybrid-cloud-vs-multi-cloud-whats-the-difference/.

[5] David Bernstein, "Intercloud" - Not all the same! Federation versus Multicloud"[Online].Available:http://cloudstrategypartners.blogspot.com/2013/04/intercloud-not-all-same-federation.html.

[6] "Cross-Cloud: A Common Operating Environment Across Private and PublicClouds"[Online].Available:https://www.vmware.com/radius/cross-cloud-introduction/.

[7] MySQL, “Sakila Sample Database / Installation” [Online]. Available: https://dev.mysql.com/doc/sakila/en/sakila-installation.html

[8] Ashraf Sharif, “MySQL Docker Containers: Understanding the basics” [Online]. Available: https://severalnines.com/blog/mysql-docker-containers-understanding-basics

[9] Docker docs, “Use volumes” [Online]. Available: https://docs.docker.com/storage/volumes/

[10] “AWS Tools for PowerShell Cmdlet Reference” [Online]. Available: https://docs.aws.amazon.com/powershell/latest/reference/items/pstoolsref-welcome.html

[11] Microsoft Azure, “Common PowerShell commands for creating and managing Azure Virtual Machines” [Online]. Available: https://docs.microsoft.com/en-us/azure/virtual-machines/windows/ps-common-ref

[12] Cloudyn: “Cloud Management Platform & Cloud Cost Optimization Tool”. [online] Available at: https://www.cloudyn.com/

[13] Dynatrace.com. [online] Available at: https://www.dynatrace.com/

[14] ScienceLogic. “Modern Operations Management – ScienceLogic”. [online] Available at: https://sciencelogic.com/

[15] Unigma.com. “Performance Monitoring, Automation and Cost Analytics for Public Clouds”. [online] Available at: https://www.unigma.com/

[16] Raffic . “DRS affinity and anti-affinity rules in VMware vSphere”. [online]. Available: https://4sysops.com/archives/drs-affinity-and-anti-affinity-rules-in-vmware-vsphere/

[17] Strongswan.org. “strongSwan – About”. [online] Available at: https://www.strongswan.org/about.html

[18] Tetiana Fydorenchyk, “Docker Multi-Containers Orchestration with Live Migration and High-Availability for Microservices in Jelastic” [Online]. Available: https://dzone.com/articles/docker-multi-containers.

[19] Adi Shachar, “Migrating Pods With Containerized Applications Between Nodes In The Same Kubernetes Cluster Using Cloudify” [Online]. Available: https://cloudify.co/2018/01/31/migrating-pods-containerized-applications-nodes-kubernetes-cluster-using-cloudify/.

[20] Ihor Kolodyuk, “Live Containers Migration Across Data Centers: AWS and Azure Integration” [Online]. Available: https://jelastic.com/blog/live-containers-migration-across-data-centers-aws-and-azure-integration/

[21] Jelastic. “Adding Docker Container to Jelastic Environment”. [online] Available: https://docs.jelastic.com/dockers-management#wizard

[22] R. Dantu, T. A. Anderson, R. Gopal, and L. L. Yang, “Forwarding and Control Element Separation (ForCES) Framework.”

[23] N. Feamster, J. Rexford, and E. Zegura, “The Road to SDN: An Intellectual History of Programmable Networks,” ACM Sigcomm Comput. Commun., vol. 44, no. 2, pp. 87–98, 2014.

[24] E. E. Haleplidis, Ed. University of Patras, K. Pentikousis, Ed., “Software-Defined Networking (SDN): Layers and Architecture Terminology.”

[25] A. Lara, A. Kolasani, and B. Ramamurthy, “Network Innovation using OpenFlow: A Survey,” IEEE Commun. Surv. Tutorials, vol. PP, no. 99, pp. 1–20, 2013.

[26] J. Tourrilhes, P. Sharma, S. Banerjee, J. Pettit, and – Vmware, “The Evolution of SDN and OpenFlow: A Standards Perspective.”

[27] T. Tanaka, “Possible economic consequences of digital cash,” First Monday, vol. 1.

[28] Research Handbook on Digital Transformations - Google Books. .

[29] M. Crosby, “BlockChain Technology: Beyond Bitcoin,” Appl. Innov. Rev. Issue, no. 2, 2016.

[30] K. Benton, L. J. Camp, and C. Small, “OpenFlow vulnerability assessment,” Proc. Second ACM SIGCOMM Work. Hot Top. Softw. Defin. Netw. - HotSDN ’13, p. 151, 2013.

[31] A. Feghali, R. Kilany, and M. Chamoun, “SDN security problems and solutions analysis,” in International Conference on Protocol Engineering, ICPE 2015 and International Conference on New Technologies of Distributed Systems, NTDS 2015 - Proceedings, 2015.

[32] K. Benton, L. J. Camp, and C. Small, “OpenFlow Vulnerability Assessment.”

[33] A. Akhunzada, E. Ahmed, A. Gani, M. K. Khan, M. Imran, and S. Guizani, “Securing software defined networks: Taxonomy, requirements, and open issues,” IEEE Commun. Mag., 2015.

[34] jarlah, “Scenario based security evaluation: Generic OpenFlow network Title: Scenario based security evaluation: Generic OpenFlow network,” 2014.

[35] O. Blial, M. Ben Mamoun, and R. Benaini, “An Overview on SDN Architectures with Multiple Controllers,” J. Comput. Networks Commun., vol. 2016, 2016.

[36] C. Tselios, I. Politis, and S. Kotsopoulos, “Enhancing SDN security for IoT-related deployments through blockchain,” in 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2017, pp. 303–308.

[37] S. R. Basnet and S. Shakya, “BSS: Blockchain security over software defined network,” in 2017 International Conference on Computing, Communication and Automation (ICCCA), 2017, pp. 720–725.

[38] "Learn Kubernetes Basics" [Online]. Available: https://kubernetes.io/docs/tutorials/kubernetes-basics/.

[39] "Get Started, Part 4: Swarms "[Online]. Available: https://docs.docker.com/get-started/part4/