Society for Science and Education United Kingdom
TMLAI Transactions on Machine Learning and Artificial Intelligence
Volume 5 No 4, 2017
ISSN 2054-7390

Implementation of the Flexible "Private - Public" Cloud Solution based on OpenStack

M. Elmahouti, O. Achandair, S. Khoulji, M.L. Kerkeb
Laboratory ISERG, Abdelmalek Essaadi University, Tétouan, Morocco
mouradelmahouti@gmail.com, o.achandair@gmail.com, khouljisamira@gmail.com, kerkebml@gmail.com, acmlis.conference@gmail.com

ABSTRACT

Cloud computing is a model that facilitates access to and manipulation of resources on demand. It is a technology that is unique today to meet the needs and demands of customers by guaranteeing a high quality of service rendered. This new model provides convenience to reorganize the current revolt in the information technology industry by ensuring cost effective and less costly solutions to meet the constraints of technical capabilities and their extension.

This article will present the technical implementation of a flexible "private - public" cloud computing, based on the Openstack solution, to ensure the business needs in terms of performance, and a response time of treatments tailored to customer demand with a Availability.

This approach, followed by a "private-public" flexible cloud, will be able to communicate two clouds, the Cloud A, which includes the entire physical infrastructure of the company, while cloud B will be provided by a service provider that does not Will be called once the configurable load threshold is exceeded on the cloud A, and as soon as the resources on the private cloud A are released, the instances migrated to the Cloud B will be again remigrated to the Cloud A to minimize even the times Allocation.

Keywords—Cloud computing, Private cloud, Public cloud Information resources, Information security, OpenStack

1 Introduction

Today, cloud computing technology has marked an intrinsic transformation in improving the quality of services rendered, minimizing response times and processing on exponential quantities of dynamic data.

It is not easy today to master the growing difficulties for the proper functioning of the different architectures, which have to adapt with the required load to meet a certain level of performance within the company, this new concept of Cloud computing offers fascinating proposals to companies to choose the optimal configuration to outsource some or the entire infrastructure.

The article addresses the different concerns of information systems managers who want to join this new cloud computing technology and have slowed down because of the significant investment in their current infrastructure by offering a smart and flexible solution based on free software "OPENSTACK", which

DOI: 10.14738/tmlai.54.3334
Publication Date: 15th August, 2017
URL: http://dx.doi.org/10.14738/tmlai.54.3334

consists of creating two clouds, a private cloud that supports the company's current platform, and a public cloud that will be auxiliary once the load threshold exceeds the capabilities of the internal platform.

The article is considered a continuation of the previous article [19], or we will detail the technical implementation of this flexible private cloud solution "private - Public", giving the necessary screenshots.

2 Proposed Solution: Private-Public Cloud Model Based On Openstack

To encourage companies to adopt this new cloud computing technology, minimizing budgets in the future and taking advantage of the state of the existing and existing platforms, our proposal is to create two models "Private : Cloud A"And" public : Cloud B "cloud based on a free solution that is the OpenStack[1].

Our approach is to create a "private - public" cloud: a private cloud (Cloud A) that supports the existing platform, and once the number of requests exceeds the capacity of the private platform, these requests will be refused internally by routing them to the Cloud to satisfy them by the public cloud (Cloud B) by paying just the rental costs, And once the internal resources are released, automatically some of the applications running on the public cloud will be supported again by the private cloud by minimizing rental costs[2].

2.1 Architecture proposed for the cloud "Private - Public"

The “Figure 1”shows the mechanism to be used by proposing the three algorithms to follow in order to pass the two-way passage between the two clouds, private and public [3], to ensure this proposition:

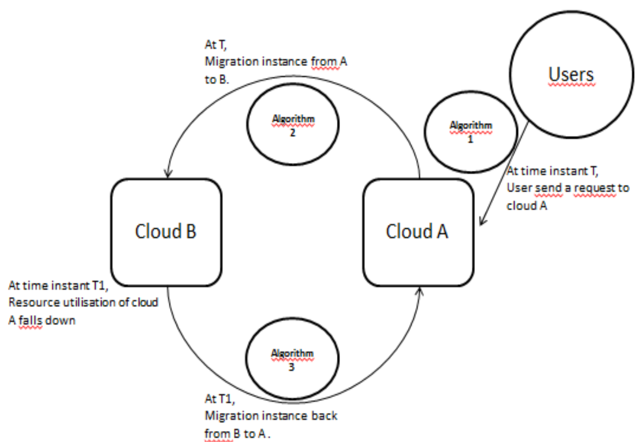

Figure 1: Cloud integration

2.2 Presentation of the three algorithms used:

The main brick of Openstack is Nova. Its purpose is to manage the resources of calculation of the infrastructure used, using the command NOVA BOOT to attack our platform [4], with its syntax below:

nova boot --image *imageID* --flavor *flavorID* --nic net-id=*nicID*

`imageID` :To select the operating system.

`net-id=nicID` : To tell the cloud which subnet is used for an instance.

`flavorID` :Reflects memory, disk and virtual process requirements.

To check the actual load on the cloud A, an implementation of the load analysis function is done, the functioning of the "analyze load" function will be explained in the algorithm 1 below[16].

Once the "analyze load" function is called, all information from all compute nodes of a controller on cloud A is received, if the load on the A cloud is below the configurable threshold, the instance will be created On cloud A, but if the load is greater than the threshold, the instance will be created on the cloud B[5].

Algorithm 1 Analyzing the load

Input Parameter:
flavorid: flavor id of requested instance

Output Parameter:
val: 0 if cloud is overloaded, 1 otherwise.

procedureAnalyzing the load (f lavor id)
Oracle database connection is made to the nova database.
Extract user's requirements using flavor id from the instance types table.
totalmemory for all compute and total disk for all compute is initialized to 0.
freememory for all compute and free disk for all compute is initialized to 0.
for each compute node c do
Find the total memory and the free memory of c from the compute nodes table.
Find the total disk space and the free disk space of c from the compute nodes table.
totalmemory for all compute =total memory for all compute + total memory of c.
total disk for all compute =total disk for all compute + total disk of c.
freememory for all compute =free memory for all compute + free memory of c.
free disk for all compute =free disk for all compute + free disk of c.
end for
if (total disk for all compute * disk threshold percentage>= users disk requirement + (total disk for all compute-free disk for all compute) then
if (total memory for all compute * ram threshold percentage) >= users memory requirement + (total memory for all compute-free memory for all compute) then
Return 1
end if
end if
Return 0
end procedure

Note that setting up an instance on another cloud requires some information on the Cloud B: Image-ID, Network-ID, which is used by algorithm 2 to position the instance on the cloud B[6].

Algorithm 2 Position the instance on the cloud B

Input parameter:
Network id: Subnet to start instance. Image id: OS launching an instance.
Tasting Id: Specify required resources
Instance Name: The name of the instance to be launched.
Procedure INSTANCE positioning ()
When running the nova boot command, the function creation ()
calls up the load to check the load
If Not exceeding threshold then
The instance is created on cloud A.
Other

It calls the createInstance () function. // Creating the instance on the cloud B
CreateInstance () uses the new startup parameters
We call the function creation () and extract the
Id of the image for the cloud B.
It Execute the NOVA command with the new parameters.
This nova boot command is executed
On the B cloud using the SSH connection.
All instance information is stored in a file, retrieved later, and stored again in the array of migrating instances to the NOVA database
End if
End of procedure.

After each migration to the cloud B, the load on the cloud A is verified, once the load is below the configurable threshold, a remigration is made again to the cloud A, the instances that will be migrated back to the cloud A follow A FIFO order, algorithm 3 below explains the function of the remigration operation [7]:

Algorithm 3 Remigration

Input Parameter:
Migratedinstancestable: Table regrouping the instances of remigration
procedure REMIGRATION( )
The Analyzing the load() for remigration is Performed at
Of the parameterized intervals.
if the resource utilization falls below a configurable
low threshold then
do
It finds the instance to be remigrated from the
migratedinstances table using the FIFO : rule.
It launches a new instance on cloud A using
the information stored in the migrated instances table.
It copies the disk image of the migrated
instance from the cloud B to the newly launched instance on the cloud A, to: restore the current state.
It finished the remigrated instance from thecloud B to free resources.
It cleans the entry of the table of instances that have been migrated.
whileResource usage is below the migration threshold and the migrated instance table is not empty
end if
end procedure

The communication between the two clouds (Cloud A and Cloud B) is secured with the use of an SSH public key, as noted in algorithm 2, [8].

3 The OpenStack SOLUTION

3.1 OpenStack Architecture

OpenStack is a set of open source software that enables deployment and manipulation of cloud computing infrastructures. This technology has a modular architecture that consists of several correlated sub-projects (Nova, Swift, Glance ......), see "Figure 2", whose purpose is to control all the resources of virtual machines, such as computing power, storage or network [9].

The Openstack project is supported by the Openstack Foundation, a non-commercial organization whose goal is to promote the Openstack project by providing support and support to the entire OpenStack community. [10]

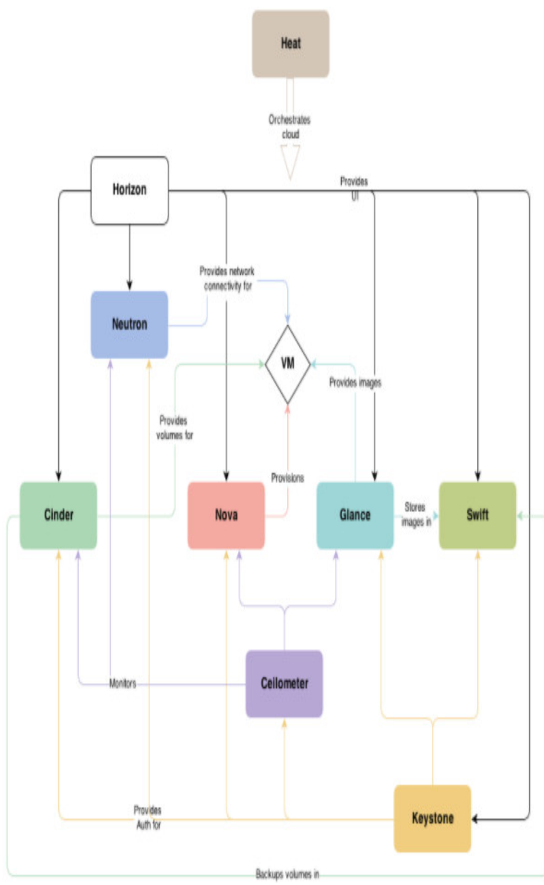

Figure 2: OpenStack Architecture and Components

3.2 The components of OpenStack

- Calculation / Nova: Nova is one of the main components of Openstack. Its role is to manage infrastructure calculation resources [11].

Nova's architecture is designed to evolve horizontally by adding hardware. Among the strengths of Nova is that it works with non-specialized hardware, which makes it possible to reuse existing servers for example [12].

- Object Storage / Swift: It is a dedicated system for storing redundant and scalable data, files are written to multiple hard disks spread across multiple servers, responsible for replication and data integrity within the cluster [13].
  The Swift cluster evolves horizontally by adding new servers, once a server or disk fails, Swift takes care of replicating its content from active nodes.
  As Swift is based on an application logic, it allows the use of inexpensive and non-specialized equipment [14].
- Block storage / Cinder:
  This is the OpenStack block storage service; its role is to provide block-based persistent devices to OpenStack instances.
  It manages the creation, attachment, and detach operations of these devices on servers.
  This bulk storage mode is used for high-performance scenarios such as database storage, but also to provide the server with low-level access to the storage device [15].
- Network / Neutron:
  The Neutron service allows you to manage and manipulate networks and IP addressing within OpenStack, allowing users to create their own networks by controlling traffic and connecting their instances to one or more networks.
  Neutron also provides different types of network deployment based on the target infrastructure.
- Dashboard / Horizon: This is a WEB application that allows users and administrators to have a dashboard to manage their clouds through a GUI [16].
- Identity service / Keystone: This service provides a central directory containing the list of services and the list of OpenStack users as well as the roles and permissions.
  Within OpenStack, Keystone is used by all users and services to authenticate with each other.
- Image service / Glance: This is the OpenStack image service, which allows the discovery, sending and distribution of disk images to instances. The glance service also allows you to store backups of these disks. Glance can store these disk images in several ways: in a folder on server, but also through the object storage service of OpenStack or in decentralized storage [17].
- Telemetry / Ceilometer: It is the service that collects different metrics on cloud usage. For example, the number of instances launched in a project and for how long.
  These metrics can be used to provide information needed for a billing system, for example. These metrics are also used in applications or other Openstack components to define actions based on certain thresholds as with the orchestration component.
- Orchestration / Heat: It is the component that ensures the orchestration of OpenStack, it allows to describe an infrastructure in the form of models. It can also use the metrics provided by Ceilmeter to decide to create additional instances according to the load of an application for example [18].

4 Implementation of the Solution: Private-Public Cloud Model Based on Openstack

To implement the proposed solution of creating two clouds, a private cloud A that supports the company's internal infrastructure, and a public cloud B at our service provider, which will be called as an ancillary platform, Once the load threshold is exceeded in the cloud A, the newly launched instances will be migrated to the cloud B for execution, once the load threshold on the cloud A arrives at its configured value, executing instances and Loaded by the public cloud B will be migrated back to the private cloud A.

An automatic load metering is initiated every 20 seconds to evaluate the load threshold in the cloud A to make the right decision on new instances arriving at cloud A and those running on public cloud B to ensure Either a migration to the B cloud when the threshold is exceeded or a remigration to the private cloud A.

In order to technically ensure this solution, two DELL Poweredge 2950 servers were used (one server is bought by our company, the other is leased since it is provided by our service provider, the same range was required to ensure efficient compatibility Between services), each having a memory of 1.8 TB, which acts as one of the compute nodes.

100 machines were assigned which function as calculation nodes for the two clouds, respectively. Knowing that the two clouds belong to a single subnet.

The OpenStack solution was installed on both servers, an installation based on Ubuntu 16.04, and then two clouds, a private cloud on the first server, all the internal servers in the company, and a second public cloud at the provider that will be used after the configurable threshold is exceeded.

The private cloud A is used by all production departments to execute their instances, they can create instances with different dimensions of 20 GB, 40 GB or 80 GB. Once 80% of the disks are allocated and 75% Of memory is busy, Cloud A is said to be overloaded, and all other requests will be satisfied by the cloud B.

Since both clouds belong to the same subnet, migrating an instance of cloud A to cloud B takes almost 17 seconds (for example, for an instance of 20GB). This time varies depending on the size of the instance.

The “Figure 3” shows the initial state of the public cloud B which initially contains four instances.

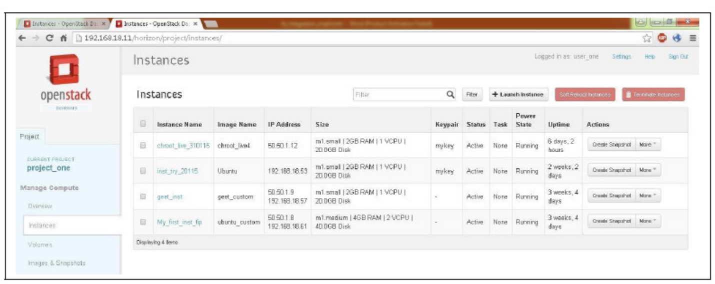

Figure 3: Initial Public Cloud

The “Figure 4” depicts the placement process of an instance; the terminal at the bottom of the “Figure 4” shows the execution of the new start command on the private cloud A.

The dashboard on the right side of the figure shows multiple instances that run on the private cloud.

The private cloud is overloaded because of these instances, so an instance named inst6 will be created on public cloud B as shown in the dashboard on the left side of the figure.

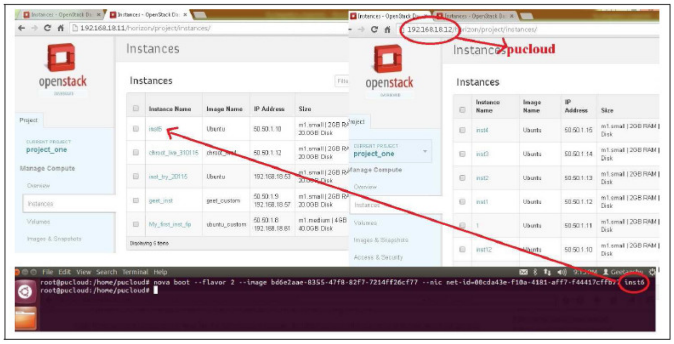

Figure 4: Placement of the instance on public cloud B.

The “Figure 5” shows that once the private cloud is overloaded with reaching the configurable threshold, the creation of the four instances (from inst6 to inst9) is supported by the public cloud.

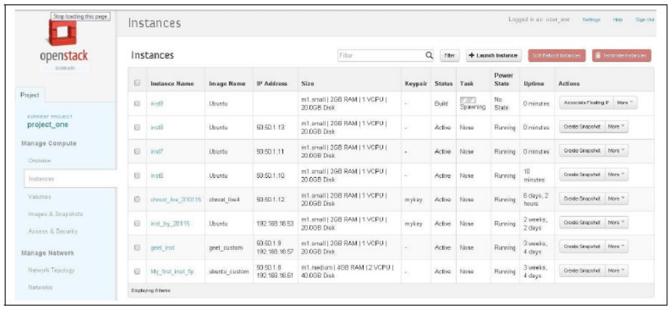

Figure 5: The public cloud after overloading condition of private cloud.

Some instances from private cloud are terminated to reduce the load on it, till it reaches the configurable low threshold as show in the “Figure 6”.

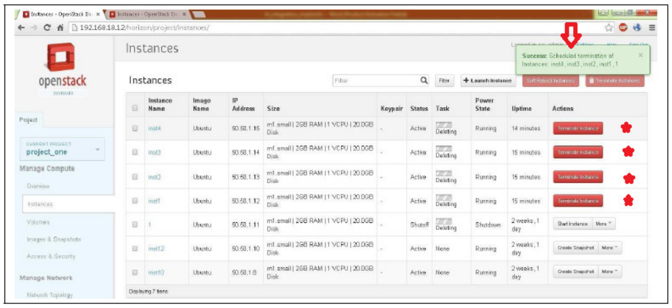

Figure 6: Reducing load on the private cloud.

Once the load is reduced on the private cloud, and we arrive below the reconfigurable threshold, the migration process is triggered.

The “Figure 7” shows the remigration of the instance inst5.

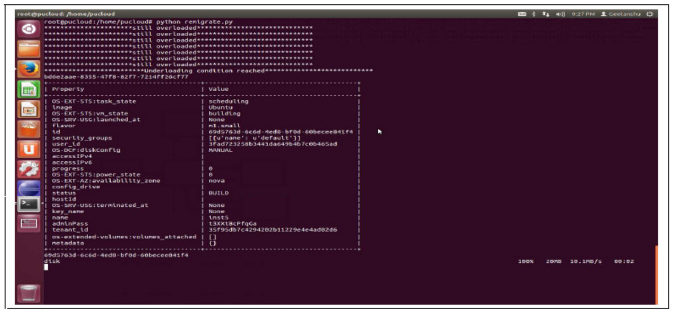

Figure 7: The remigration process.

The “Figure 8” and “Figure 9” show the state of both private and public clouds after re-migration.

Once the load is reduced on the private cloud, four instances (De inst6 to inst9) have been remigrated to the private cloud by releasing the public cloud.

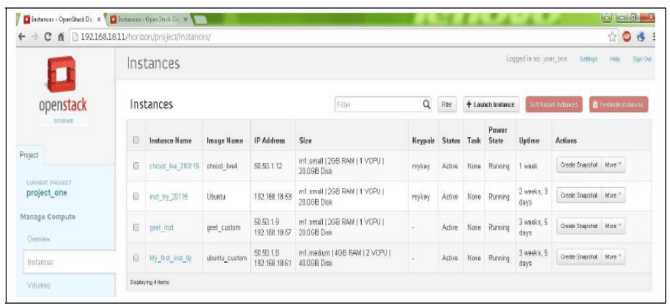

Figure 8: The public cloud after remigration.

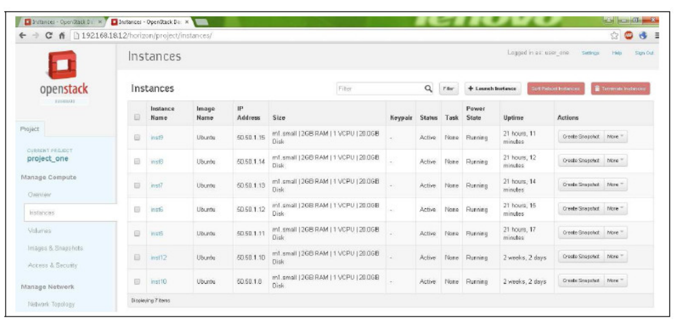

Figure 9: The private cloud after remigration.

5 Conclusion

In this paper, we proposed an effective solution to accompany the change in information systems management, especially to encourage the adoption of new cloud computing technology based on the free OpenStack solution to create a private cloud Which consolidated the entire enterprise infrastructure and another public cloud at the service provider that is called whenever internal capabilities are exhausted and the only configurable is outdated. This solution has been definitively deployed and has shown these added values in terms of performance, quality of service rendered and associated costs.

The current implementation is considered the first pilot site, the next step will be to generalize the same architecture for all the subsidiaries.

ACKNOWLEDGMENT

This work was carried out with the support of our company, for the implementation of the project.

We thank all members of the project team who contributed to the success of this proposal.

REFERENCES

[1] Haifeng Li, Huachun Zhou, Hongke Zhang, Bohao Feng, "EmuStack: An OpenStack-Based DTN Network Emulation Platform", 2016 International Conference on Networking and Network Applications (NaNA), vol. 00, no. , pp. 387-392, 2016, doi:10.1109/NaNA.2016.24

[2] Dinkar Sitaram, Sudheendra Harwalkar, K.V. Suchith Kumar, "Standards Based Integration of Intercloud for Federation with OpenStack", 2016 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), vol. 00, no. , pp. 113-118, 2016, doi:10.1109/CCEM.2016.028

[3] Pooya Musavi, Bram Adams, Foutse Khomh, "Experience Report: An Empirical Study of API Failures in OpenStack Cloud Environments", 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), vol. 00, no. , pp. 424-434, 2016, doi:10.1109/ISSRE.2016.42

[4] M. Abhishek Sharma, Monica O. Joshi, "Openstack Ceilometer Data Analytics & Predictions", 2016 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), vol. 00, no. , pp. 182-183, 2016, doi:10.1109/CCEM.2016.045

[5] Ali Babar, Ben Ramsey, "Tutorial: Building Secure and Scalable Private Cloud Infrastructure with Open Stack", 2015 IEEE 19th International Enterprise Distributed Object Computing Workshop (EDOCW), vol. 00, no. , pp. 166, 2015, doi:10.1109/EDOCW.2015.23

[6] Dario Bruneo, Francesco Longo, Giovanni Merlino, Nicola Peditto, Carmelo Romeo, Fabio Verboso, Antonio Puliafito, "A Modular Approach to Collaborative Development in an OpenStack Testbed", 2015 IEEE Fourth Symposium on Network Cloud Computing and Applications, vol. 00, no. , pp. 7-14, 2015, doi:10.1109/NCCA.2015.12

[7] Vojtech Cima, Bruno Grazioli, SeÃ¡n Murphy, Thomas Michael Bohnert, "Adding energy efficiency to Openstack", 2015 Sustainable Internet and ICT for Sustainability (SustainIT), vol. 00, no. , pp. 1-8, 2015, doi:10.1109/SustainIT.2015.7101358

[8] D. Thomas Erl, D. Zaigham Mahmood, D. Ricardo Puttini, “Cloud Computing: Concepts, Technology & Architecture” September 2015

[9] Yun Zhang, Farhan Patwa, Ravi Sandhu, Bo Tang, "Hierarchical Secure Information and Resource Sharing in OpenStack Community Cloud", 2015 IEEE International Conference on Information Reuse and Integration (IRI), vol. 00, no. , pp. 419-426, 2015, doi:10.1109/IRI.2015.71

[10] Robayet Nasim, Andreas J. Kassler, "Deploying OpenStack: Virtual Infrastructure or Dedicated Hardware", , vol. 00, no. , pp. 84-89, 2014, doi:10.1109/COMPSACW.2014.18

[11] David Bernstein, "Cloud Foundry Aims to Become the OpenStack of PaaS", IEEE Transaction on Cloud Computing, vol. 1, no. , pp. 57-60, July 2014, doi:10.1109/MCC.2014.32

[12] Juan Angel Lorenzo del Castillo, Kate Mallichan, Yahya Al-Hazmi, "OpenStack Federation in Experimentation Multi-cloud Testbeds", , vol. 02, no. , pp. 51-56, 2013, doi:10.1109/CloudCom.2013.103

[13] Yang Wang, Xiaomin Wang, Jianyong Chen, "On-Demand Security Architecture for Cloud Computing", IEEE Computer, vol. 45, no. , pp. 73-78, July 2012, doi:10.1109/MC.2012.120

[14] V.KRISHNA REDDY1 , Dr. L.S.S.REDDY, "Security Architecture of Cloud Computing",International Journal of Engineering Science and Technology (IJEST), ISSN : 0975-5462 Vol. 3 No. 9 September 2011 pp.7149-7155.

[15] A Beloglazov, J Abawajy, R Buyya - Future generation computer systems, Future Generation Computer Systems, Elsevier, Volume 28, Issue 5, May 2012, Pages 755–768

[16] Ian Foster, Rubén S. Montero, Borja Sotomayor, Ignacio M. Llorente, "Virtual Infrastructure Management in Private and Hybrid Clouds", IEEE Internet Computing, vol. 13, no. , pp. 14-22, September/October 2009, doi:10.1109/MIC.2009.119

[17] Wen-Hwa Liao, Shuo-Chun Su, "A Dynamic VPN Architecture for Private Cloud Computing", 2011 IEEE 4th International Conference on Utility and Cloud Computing (UCC 2011), vol. 00, no. , pp. 409-414, 2011, doi:10.1109/UCC.2011.68

[18] Sumit Goyal, "Public vs Private vs Hybrid vs Community - Cloud Computing: A Critical Review", I.J. Computer Network and Information Security, 2014, 3, 20-29.