International Journal of Latest Trends in Engineering and Technology
Special Issue NCRTCC 2017, pp.006-013
e-ISSN:2278-621X

SECURITY FOR USER INFORMATION ON CLOUD USING SSH KEY AUTHENTICATION

R.KanniyaSri[1], R.Kayalvizhi[2], M.Priyanka[3], R.Shobana Lakshmi[4]

Abstract – Most of the organization adapts cloud computing for their business use, for the benefit of low cost. An open source cloud implementation using open stack is mostly deployed as an Infrastructure-as-a-Services (IaaS). The user loses control of their personal information, because it is stored on a computer belonged to a cloud provider this happens when the owner of the remote server is a person or organization may want to take advantage. For that, we provides privacy to those information which is kept private by the user. This paper deals that issue for privacy by using Secure Shell (SSH) key authentication.

Keywords – Cloud Computing, Private cloud, Public cloud, Public Private Cloud Integration, Ubuntu instances.

I. INTRODUCTION

The nature of the Internet is changing from a place used to read web pages to an environment that allows the user to run software applications [1]. Cloud computing is a distributed computing paradigm that focuses on providing a wide range of users with distributed access to scalable, virtualized hardware and/or software infrastructure over the internet [2]. Cloud computing is hinting at a future in which we won't compute on local computers, but on centralized facilities operated by third-party compute and storage utilities [3]. The datacenter hardware and software is what we will call a cloud [4]. Most of the organization does not want to incur their annual expenditure for buying and maintaining the computational resources which are used only at the moment of situation or at a particular time because of those resources are in high cost and high power needed. Suppose if the organization has its own computational resources, even though this is not enough to tackle the high load due to this, it becomes unscalable. In case of light load, there is a wastage of resources this leads to inefficient utilization.

With cloud computing, resources are accessed over the internet, so additional compute nodes can be added easily in existing infrastructure to provide
scalability, efficient use of resources and a cost-effective solution [5]. Using this computing, the organization pay for the resources which can be utilized by them and their needs to be dynamically changed.

A private cloud is a model of cloud computing that is implemented within the corporate firewall under the control of the IT department with limited resources [5]. A public cloud is accessed by any user of the cloud which does not provides any limitation for the usage of resources, based on the usage they pay for that resources. To satisfy increasing demands in a cost-efficient way, the organization requires to have an integration of the private and a public cloud [5].

[1] *Sri Sairam Institute of technology Chennai, Tamil Nadu*
[2] *Sri Sairam Institute of technology Chennai, Tamil Nadu*
[3] *Sri Sairam Institute of technology Chennai, Tamil Nadu*
[4] *Sri Sairam Institute of technology Chennai, Tamil Nadu*

This paper describes an authentication for the integration of public private cloud using Secure Shell (SSH) key generation algorithm. For integrating two clouds, Openstack free and open source software to be used. Openstack provides Infrastructure-as-a-Service [5]. IaaS serves as the foundation layer for the other delivery models, and a lack of security in this layer will certainly affect the other delivery models, i.e., PaaS and SaaS that are built upon IaaS layer [6].

II. BACKGROUND

OpenStack is a software tool used for creating and managing cloud platforms like private and public clouds. OpenStack provides infrastructure which makes the user to quickly add a new instances. It allows users to deploy virtual machines as instances, which handle different tasks for managing a cloud environment [7]. It is a cloud operating system, which provides shared services such as compute, networking and storage and resources such as bare metal, virtual machines and containers. It is powerful and highly configurable integration engine.

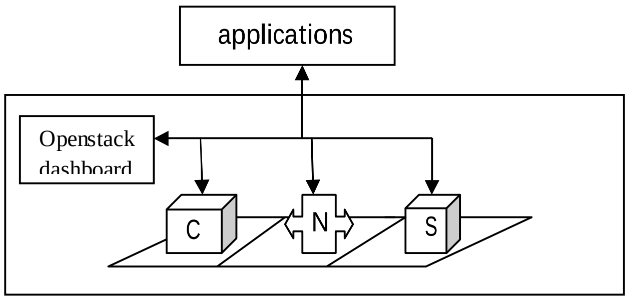


Fig. 1: OpenStack Cloud Operating System

In the Figure 1, the various services offered by OpenStack software to be represented as C- compute, N- networking, S- storage. In this, the important service is compute (Nova). Nova is a management layer which allows creation and deletion of virtual machines. It also identifies on which host to provision the customer virtual machine. It consists of several sub services such as nova-api, nova-compute, nova-scheduler, nova-network, etc [5].

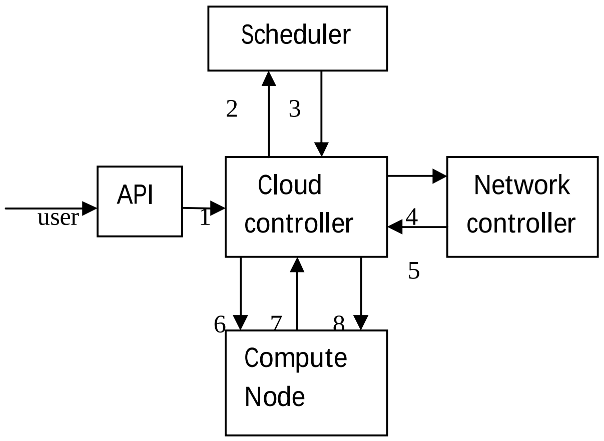


Fig. 2: Launch instance process

- (1) API server sends a message to a cloud controller.
- (2) After receiving the message, the controller communicates with nova scheduler.
- (3) Then the scheduler selects random compute nova and requests to start a new instance.
- (4,5) It sends a message to Neutron and requests to allocate a fixed IP to an instance.
- (6,7,8) Compute node continues with spawning a new instance.

Included with nova boot command, the nova package offers commands to handle the state of an instance. Those commands are *nova stop instance_name* to power off the instance, *nova start instance_name* to power on the instance, *nova suspend instance_name* to suspending the instance, *nova resume instance_name* to resuming the suspended instance, *nova reboot instance_name* to rebooting the running instance.

Nova allocates resources to an instance only if the enough resources are available, otherwise it denies that particular user request. If the resources are not enough to satisfies the user request, the cloud is overloaded and needs to transfer its load to other cloud. Cloud service providers provides some tools to connect and migrate instances to their networks. Some Cloud service providers are Amazon, Microsoft and IBM. All the tools provided by above three Cloud service providers provide a platform to the private cloud to connect to the public cloud for data storage and running applications.

While the integration of both private and public cloud requires some authentication to accessing the limitation of the informations. In our approach, by using SSH key authentication , provides privacy for the information which are kept private by the user, that are stored on the computer belonged to the provider. This approach would reduces the privacy issues.

III. INTEGRATION OF CLOUDS

In the organization, if it uses their own private cloud, there is a limited resources. If the workload of the private cloud goes increases when the expansion of the organization takes place. In that situation, addition of new resources to an existing cloud is not a feasible solution. For that integration takes place which is a better solution when compared to adding new resources. This integration of two clouds is a cost effective solution to overcome the resource limitations in a case where cloud gets overloaded for a small time interval but in case of longer duration, it is not suitable [5]. The integration of clouds allows migration and remigration of resources from one to another cloud and vice versa.

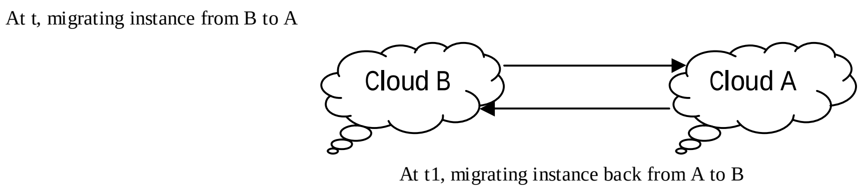

At t, migrating instance from B to A
Cloud B
Cloud A
At t1, migrating instance back from A to B

Fig. 3: Integration of Clouds

IV. COMPARISON

The comparison of SSH and without SSH authentication describes as follows:

In the SSH authentication, it creates two keys private and public keys by using those keys encryption and decryption takes place. With the creation of keys the user and the remote server transfer their information with a secured connection. The intruder arrival can be avoided.

In without SSH authentication, the user and the remote server transaction can be visualized by other users or intruders. They can take advantage in that transferred information to gain their own profit. And also they can misuse those information.

V. ARCHITECTURE

This architecture describes how the interaction takes place between the user with the integrated cloud. The reason behind to use our concept to require privacy between the user and the remote server, the remote server then interact with the cloud provider.

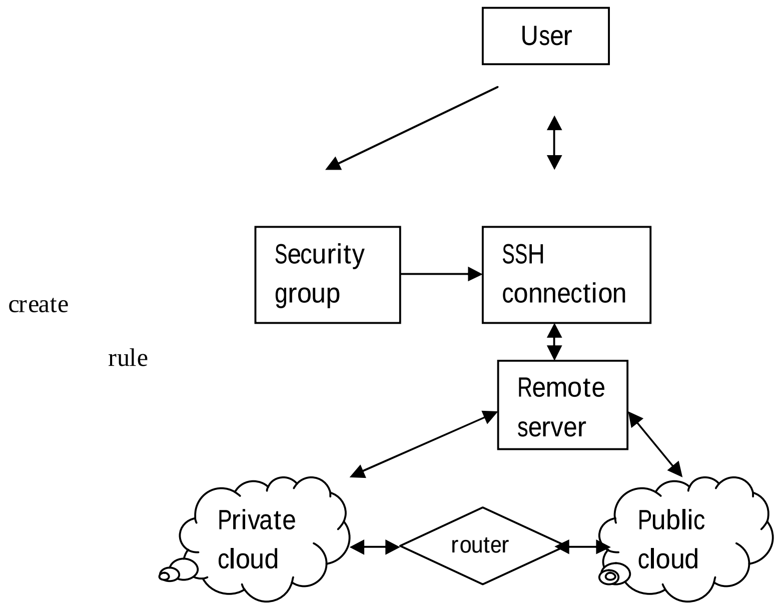


Fig. 4: Authentication on OpenStack

Step 1: The user creates security group for providing privacy to their information.

Step 2: From the security group the user choose their rule as SSH connection.

Step 3: The remote server and the user has to be provided with a secure connection

Step 4: Then the remote server get an information from the cloud based on the user demand.

Step 5: For integrating the public and private cloud, the router to be used to filter the traffic which to be allowed through it.

VI. IMPLEMENTATION

The nova package [8] provides a nova boot command to a user launch an instance. In this the user provide an image id, flavor id, network id and name of that instance. By using an image id, the user select an operating system to launch. Network id provides the subnet used for an instance. Flavor id provides memory,disk and virtual CPUs needed. The general nova instance-action command is shown below:

nova instance-action de1.pointtoserver.com *30faabced2017bhj100802*

In the nova instance action provide users insight into what has been done with their instance.

Algorithm 1 Instance placement on remote cloud with adding keypairs

Input Parameter:

Network_id: Subnet to launch an instance.

Image_id: Operating System to launch an instance.

flavor_id: Resource requirement specification.

Instance_name: Name of the instance which to be launched.

```
1: procedure INST_PLACE_WITH_KEYPAIR( )
2:    On execution of nova boot command, the create( ) function to be called.
```

```
3:  The create( ) function then calls createInst( ) function.
4:  The createInst( ) function uses the nova boot parameters passed by the create( ) function and extracts the corresponding image_ id.
5:  It rebuilds the nova boot command using the new parameters.
6:  if the user wants privacy then
7:      Call createKeypair( ) function to create public and private key.
8:  Correct the permissions to access the keypairs.
9:      Launch the instance and select the key pair just created providing with password.
10:     Call createSecurityGroups( ) function in which SSH to be a rule for providing secure connection.
11:         if the client connected with the server with the same IP address before then
12:             It shows warning message.
13:         else
14:             It continues
15:         end if
16:  end if
17:     Information of the launched instance is stored in a file, which is then extracted and is stored in the instance table.
18:     By using the instance_action table, the user can insight into their instance to check what has been done with their instance.
19: end procedure
```

The nova boot command in turn calls the create() function present in server.py file. Implementing the createInst() function in the base.py file, which calls the run_inst.sh script. This script passes nova boot parameters to the cloud controller. The user wants to kept their information privately then they create two keys private and public those can be downloaded, and stored in a file with extension .pem file. Save that file on a convenient location.

After that modifies the file permission i.e., .pem file. Now launch the instance with the key pair. Select the keypair that just created and providing with some password. Start the secure connection between the instance that to be launched by the user and the cloud controller. If the user / client connected with a server with the same IP address before used to launch the instance then it displays an warning and this warning is also possible that someone is doing something nasty. Otherwise it asks for continuing the further process then the user to be logged onto their instances. The information of the launched instance is stored in a file, which is then extracted and is stored in the instance table.

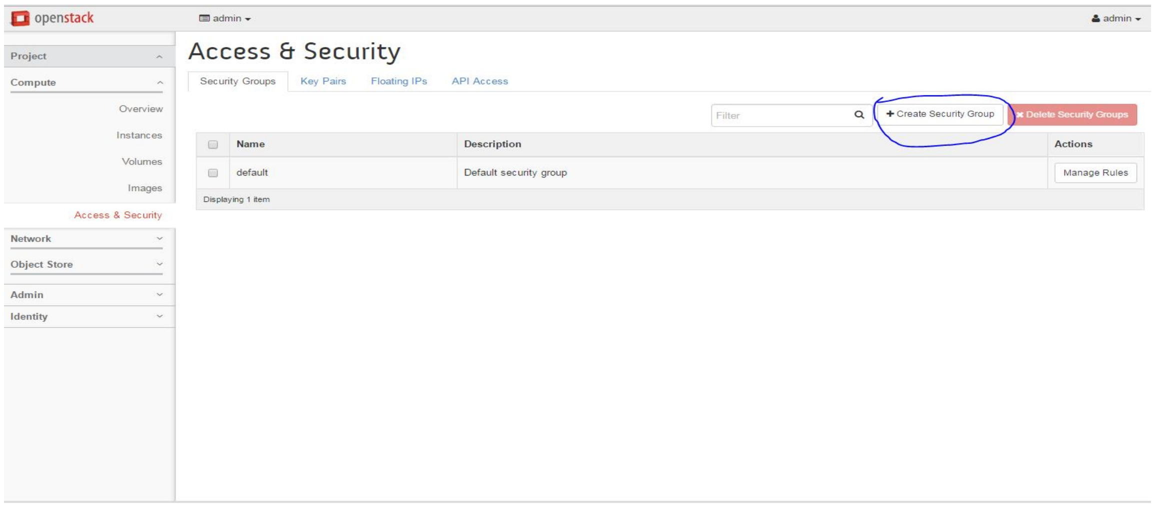
The user wants to check the activity carried on their instance to be verified by using the instance-action table. This is the functioning which is in the Algorithm 1. All the communications made between two clouds i.e., public and private cloud is done with secure connection. This connection to be made by creating security groups in which ssh to be provided as a rule to filter out the traffic which cannot be allowed to access the instance of the user.

The SSH connection consists of two stages. One is connection stage and another one is authentication. In the connection stages, opening a session. Once the session setup has been completed, a program is started to perform its operation. The size of the window, through which the data to be transferred should be mentioned. After that, the data to be transferred.in case there is a change in the window size at the user or client side, it may send a message to the server to indicate the altered window size.

For authentication there are several methods, we use public key authentication. In this method, if the requested user name does not exist then the session to be disconnected. Otherwise, sending an signature which can be created with private key. This created signature to be send to the server. The server verifies both the key and the signature which can be transferred. Once the verification can be successful then that user can be allowed to access their resources. The private key is stored in an encrypted form at the client host and also the user must supply a passphrase before the signature can be generated. By using this method,compared to other methods like username password authentication it gives little more secure for the information.

VII. RESULTS AND SNAPSHOT

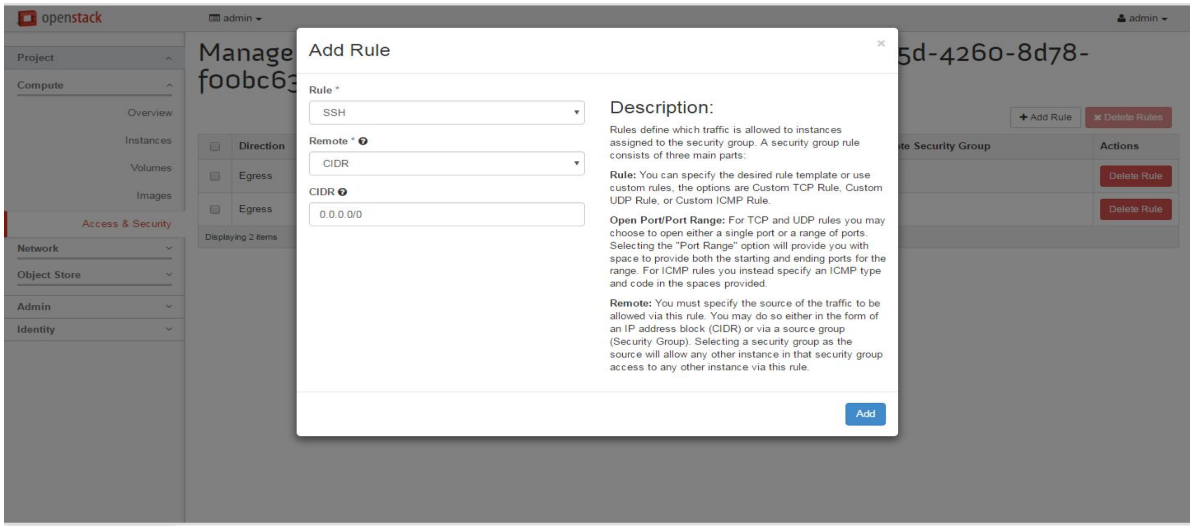
The following snapshot describes the creation of security group and adding the rules to the group to filter out the traffic which to be allowed by the router.



	Name	Description	Actions
	default	Default security group	Manage Rules

Displaying 1 item

Fig. 5: To Creating a security group



Add Rule

Rule *: SSH

Remote *: CIDR

CIDR: 0.0.0.0/0

Description:

Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:

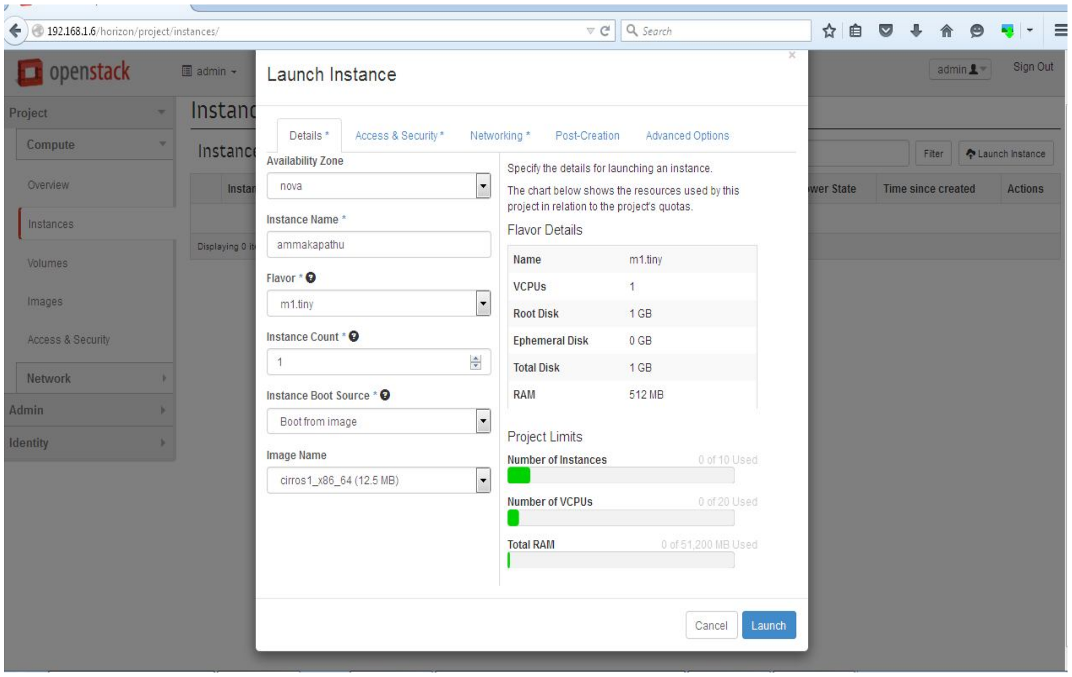
Rule: You can specify the desired rule template or use custom rules, the options are Custom TCP Rule, Custom UDP Rule, or Custom ICMP Rule.

Open Port/Port Range: For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the "Port Range" option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.

Remote: You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.

Fig. 6: Adding a ssh rule to the created group

In the first snapshot, the security group to be createdby clicking compute tab and then press access and security button in which click create security group to create a security group which are the set of IP filter rules applied to the network for the virtual machine.After that creating the security group adding rules to that group. The rule specifies the traffic which to be allowed to the instances which is assigned to the created security group.



Launch Instance

Availability Zone: nova

Instance Name *: ammakapathu

Flavor *: m1.tiny

Instance Count *: 1

Instance Boot Source *: Boot from image

Image Name: cirros1_x86_64 (12.5 MB)

Specify the details for launching an instance.

The chart below shows the resources used by this project in relation to the project's quotas.

Flavor Details

Name	m1.tiny
VCPUs	1
Root Disk	1 GB
Ephemeral Disk	0 GB
Total Disk	1 GB
RAM	512 MB

Project Limits

Number of Instances 0 of 10 Used

Number of VCPUs 0 of 20 Used

Total RAM 0 of 51,200 MB Used

Fig. 7: launch the instance in the cloud

In Figure.7, the instance created for the user to be launched in the cloud and check whether it is accessed by others.

VIII. CONCLUSION

In this paper, we provided a snapshot which contains the implementation of integrating public and private network and also an algorithm to maintain privacy for the instance of the user. This privacy is altered depending upon on the user needs. Future work can include providing more efficient algorithm for incorporating privacy for user information.

REFERENCES

[1] T. Surcel and F. Alecu, “Applications of Cloud Computing,” In International Conference of Science and Technology in the Context of the Sustainable Development, pp.177-180, 2008.

[2] G. Lewis, “Basics about Cloud Computing,” software Engineering Institute Carniege Mellon University, Pittsburgh, 2010.

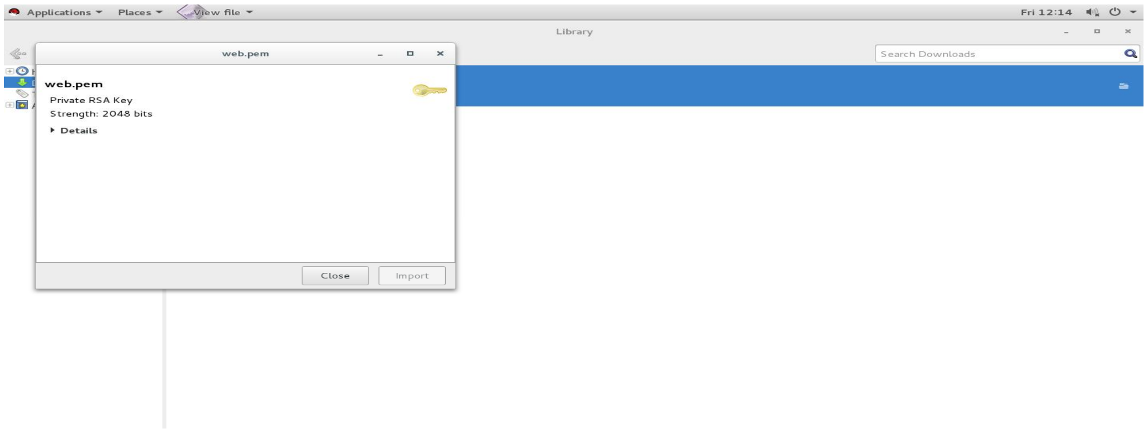
[3] I. Foster, Y. Zhao, I. Raicu and S. Lu, “Cloud computing and grid computing 360-degree compared,” In: IEEE Grid Computing Environments Workshop, pp.1-10, November, 2008.

[4] S. Goyal, “Public vs private vs hybrid vs community-cloud computing: A critical review,” International Journal of Computer Network and Information Security (IJCNIS), vol. 6, no. 3, p. 20, 2014.

[5] Geentanshu Mangal, Payal Kasliwal, Umesh Deshpande, Manish Kurhekar and Girish Chafle, “Flexible Cloud Computing by Integrating Public-Private Clouds using OpenStack,” In: IEEE International Conference on Cloud Computing in Emerging Markets, pp.146-152, 2015.

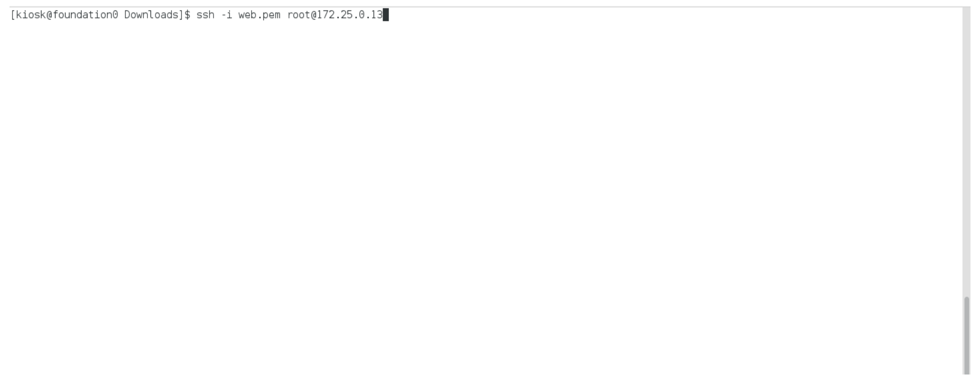
[6] Wesam Dawoud, Ibrahim Takouna, and Christoph Meinel, “Infrasturcture as a Service Security: Challenges and Solutions,” 2010.

[7] D. Dongre, G. Sharma, M. Kurhekar, U. Deshpande, R. Keskar, and M. Radke, “Scalable cloud deployment on commodity hardware using
openstack,” in Advanced Computing, Networking and Informatics- Volume 2, pp. 415–424, Springer, 2014.
[8] “Python apis: The best-kept secret of openstack.” http://www.ibm.com/developerworks/ cloud/library/cl-openstack-pythonapis/. Accessed:
2015-07-15.



web.pem
Private RSA Key
Strength: 2048 bits
Details

Fig. 8: creating the private key from the public key



```
[kiosk@foundation0 Downloads]$ ssh -i web.pem root@172.25.0.13
```

Fig. 9: Configuring passphrase from the private key