Design of a Small-Scale and Failure-Resistant IaaS Cloud Using OpenStack (Supplementary Material)

Samuel A. Heuchert, Bhaskar P. Rimal, Martin Reisslein, and Yong Wang

***Abstract*—This supplementary materials document accompanies the article "Design of a Small-Scale and Failure-Resistant IaaS Cloud Using OpenStack" with additional details on the implementation and configuration of the proposed small-scale and failure resistant IaaS cloud.**

I. Implementation

A. All Node Preparation

Each of the five nodes is prepared from a clean, minimal install of CentOS 7 with a single network interface on the local LAN that provides Internet access. Each node is configured with a mirrored storage array to provide redundancy at the lowest level in the event of physical disk failure. After each node has completed a clean, minimal install of CentOS 7, the following preparation tasks are completed before configuring the PackStack configuration file. It is assumed that each node can be accessed via SSH with the root password configured during installation.

Step 1: Updates are installed on each node and each node is rebooted.

```
$ sudo yum -y update
$ sudo reboot
```

Step 2: Static IP addresses are configured on the LAN interfaces. PackStack configures all OpenStack internal networking through the use of Linux Bridges. In the steps below, appropriate interface names and IP addresses are substituted.

```
$ sudo vi /etc/sysconfig/network-scripts/
    ifcfg-eth0

ONBOOT = yes
IPADDR = x.x.x.x
PREFIX = x.x.x.x
GATEWAY = x.x.x.x
DNS1 = x.x.x.x
BOOTPROTO = none
$ systemctl restart network
```

Step 3: Hosts files are updated to reflect each node's IP and hostname mapping. Each line in the hosts file must contain a unique entry. After the entries are committed, each node can ping each other using hostnames.

```
$ sudo vi /etc/hosts
x.x.x.x controller.local controller
x.x.x.x networking.local networking
x.x.x.x compute1.local compute1
x.x.x.x compute2.local compute2
x.x.x.x compute3.local compute3
```

Step 4: SELinux and NetworkManager are disabled.

```
$ sudo vi /etc/sysconfig/selinux
SELINUX=disabled
$ sudo su
$ systemctl stop NetworkManager
$ systemctl disable NetworkManager
$ reboot
```

B. Node 1 Preparation: Controller

The controller node acts as the hub for installation and must access each of the other nodes with root permissions. Because of this, root user authentication without passwords is required for each of the other nodes. The 'ssh-copy-id' command must be used for each of the target nodes. After it is configured, remote access using ssh without a password prompt is available.

```
$ sudo su
$ ssh-keygen
$ ssh-copy-id -i /root/.ssh/id\_rsa.pub
    root@Networking\-NodeIP
$ ssh-copy-id -i /root/.ssh/id\_rsa.pub
    root@Compute1\-NodeIP
$ ssh-copy-id -i /root/.ssh/id\_rsa.pub
    root@Compute2\-NodeIP
$ ssh-copy-id -i /root/.ssh/id\_rsa.pub
    root@Compute3\-NodeIP
```

Following the root ssh authentication change, the RDO repository is enabled to install PackStack on the controller node. The root user is still used.

```
sudo su
$ yum install -y https://www.rdoproject.
    org/repos/rdo-release.rpm
$ yum install -y centos-release-openstack
    -train
$ yum install -y openstack-packstack
```

Funding: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

S. A. Heuchert, B. P. Rimal and Y. Wang are with the Beacom College of Computer and Cyber Sciences at Dakota State University, Madison, SD, 57042 USA, e-mail: sam.heuchert@trojans.dsu.edu; bhaskar.rimal@dsu.edu; yong.wang@dsu.edu

M. Reisslein is with the School of Electrical, Computer, and Energy Engineering, Arizona State University (ASU), Tempe, AZ 85287 USA, e-mail: reisslein@asu.edu

The answer file for PackStack is now generated. The answer file is used as the configuration file to install OpenStack on top of CentOS 7 on all nodes.

```
$ sudo su
$ packstack --gen-answer-file=/root/
    answer.txt
```

Once generated, the answer file is edited to specify the target IP addresses of the networking and compute nodes. In addition, other items, such as public NTP servers and default passwords are changed, alongside disabling demonstration mode and enabling SSL access for Horizon. The full list of lines to change in the answer file is below.

```
$ sudo su
$ vi /root/answer.txt
CONFIG\_CONTROLLER\_HOST=x.x.x.x
CONFIG\_COMPUTE\_HOSTS=x.x.x.x,x.x.x.x
CONFIG\_NETWORK\_HOSTS=x.x.x.x
CONFIG\_PROVISION\_DEMO=n
CONFIG\_HORIZON\_SSL=y
CONFIG\_NTP\_SERVERS=132.163.97.6
CONFIG\_KEYSTONE\_ADMIN\_PW=PasswordHere
```

While more parameters can be changed, they are out of the scope of this paper since the focus is on a small-scale IaaS deployment. Elasticity and resiliency can be fully demonstrated with the above parameter changes.

After the parameters are changed, the installation is started using the PackStack command while specifying the answer file. The installation takes about two hours depending on the performance of the hardware.

```
$ packstack --answer-file=/root/answer.
    txt
```

The installation uses PackStack-specific puppet modules and manifests to install all the required dependencies on each node. Fig. 1 shows a screenshot of the installing process as it appears on the command line of the controller node and Fig. 2 shows the Horizon Dashboard login page that is available following the completion of the installation. It is accessible at https://x.x.x.x/dashboard using the admin username and password as specified in the answer file.

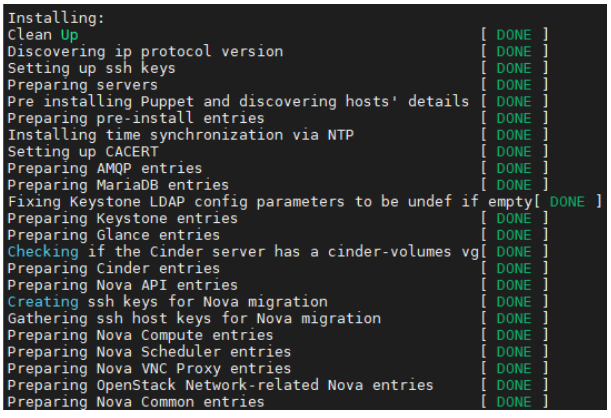

Fig. 1. A snapshot of the installation process.

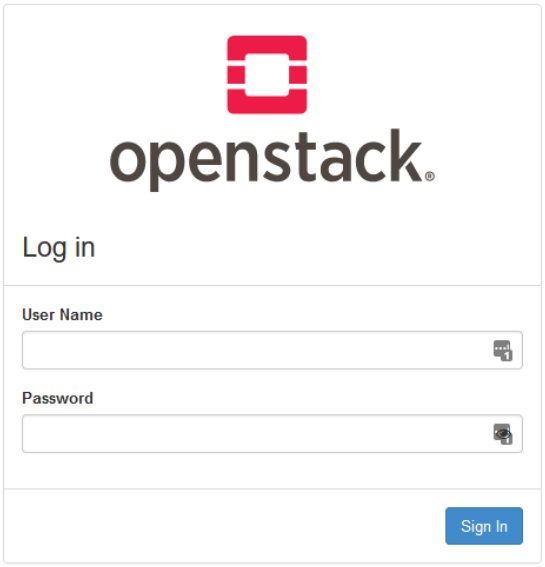

Fig. 2. A snapshot of the Horizon Dashboard login page.

C. Networking Node Interface Changes

Following the successful installation, the networking node is configured in order that the physical LAN-facing interface is added to the Open vSwitch Bridge interface that PackStack created called 'br-ex'. This process starts by copying the physical interface configuration to the bridge interface configuration and assigning the LAN IP to the bridge, while adding the physical interface to the Bridge. Once completed, the networking node is rebooted. The configuration is now tested by verifying that the physical interface's IP address is now assigned to the Open vSwitch interface, while still being able to ping LAN devices. The configuration changes are outlined below.

```
$ cd /etc/sysconfig/network-scripts
$ cp ifcfg-eth0 ifcfg-br-ex
$ vi ifcfg-eth0
DEVICE=eth0
TYPE=OVSPort
DEVICETYPE=ovs
OVS\_BRIDGE=br-ex
ONBOOT=yes
$ vi ifcfg-br-ex
DEVICE=br-ex
TYPE=OVSBridge
BOOTPROTO="none"
DEVICETYPE=ovs
IPADDR=x.x.x.x
NETMASK=x.x.x.x
GATEWAY x.x.x.x
DNS1=x.x.x.x
ONBOOT=yes
```

D. Node Verification

Each node is verified by logging into the Horizon Dashboard. Once logged in, the System Information section is chosen under the Admin tab. From this screen, the different services, such as Glance, Swift, Cinder, Nova, Keystone, or Neutron, are verified. Fig. 3 is an example of the different service menus that are checked.

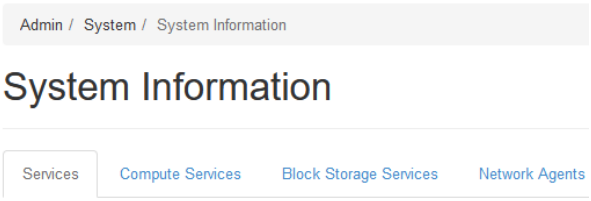

Fig. 3. The system information menu.

Under the Compute Services menu in the same System Information section, the three compute nodes show as nova-compute service hosts located in the nova zone. The nova-conductor and nova-scheduler services are hosted on the controller node and shown in the internal zone. Fig. 4 shows the compute services status with all three compute nodes active.

Name	Host
nova-conductor	controller.local
nova-scheduler	controller.local
nova-compute	compute2.local
nova-compute	compute1.local
nova-compute	compute3.local

Fig. 4. The compute services information menu.

Finally, the Network Agents menu is verified that an OVN Controller Agent is hosted on the networking node and the three compute nodes also have a metadata agent worker. Fig. 5 shows the verification.

Type	Name	Host
OVN Controller agent	ovn-controller	networking.local
OVN Controller agent	ovn-controller	compute2.local
OVN Metadata agent	networking-ovn-metadata-agent	compute2.local
OVN Controller agent	ovn-controller	compute3.local
OVN Metadata agent	networking-ovn-metadata-agent	compute3.local
OVN Metadata agent	networking-ovn-metadata-agent	compute1.local
OVN Controller agent	ovn-controller	compute1.local

Fig. 5. The network agents information menu.

II. Project Preparation

A. Project and Project Member Creation

Fig. 6 shows the project quota settings menu. Once the

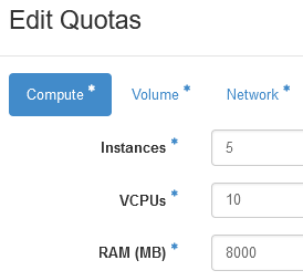

Fig. 6. The quota settings.

project is created, quotas are set, the user is assigned, and appropriate permissions are configured. The newly created user can sign into the Horizon dashboard with the configured credentials. When signing in, project users will only have access to the dedicated project that the administrator assigns.

B. Image and Flavor Creations

Flavors and images are defined on the Admin side of OpenStack as follows. The Admin section is selected, and the Compute tab is expanded. From here, an image is created. For this project, the following image and settings are used while setting the permissions to Public in order that all projects, including the newly formed project1, can use it.

- Name: cirros
- Disk Format: qcow2
- Source: http://download.cirros-cloud.net/0.5.1/cirros-0.5.1-x86_64-disk.img

Fig. 7 shows how the image appears in the Admin console after it is created. Since the visibility permissions are set to public, all projects in the OpenStack deployment are able to use it.

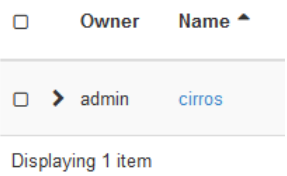

Fig. 7. The newly created image.

After the image is created, a flavor is defined and assigned to the new project. Although there are defined default flavors, a custom flavor that defines vCPUs, RAM, Root Disk Size, Ephemeral Disk Size, and permissions is created and assigned to the project as to adhere more closely to the previously assigned quotas. In this case, a flavor called `a1.small` with 1 vCPU, 512 MB of RAM, and a 1 GB Root Disk provides a lightweight instance option to demonstrate scalability. Another flavor called `a2.medium` with 2 vCPUs, 1024 MB of RAM, and a 1 GB Root Disk provides a higher performing option

that can be used to scale up an instance. Fig. 8 shows the two flavors that are used by project1 to demonstrate elasticity through live editing of the flavors. In production, this would allow developers to scale up instances as long as they have not hit their quota as defined by the project.

Flavor Name	VCPUs	RAM	Root Disk
a1.small	1	512MB	1GB
a2.medium	2	1GB	2GB

Fig. 8. The newly created flavors.

C. Network Creation

In order for the instances to have network connectivity, both internally and externally, a network is created. To do this, the project1 user signs into the new project and creates two networks: an internal network to be used by the instances for local connectivity and an external network to be used for external connectivity. Fig. 9 shows the two newly created networks prior to a router being created that will bridge the gap between the two and provide Network Address Translation.

Name	Subnets Associated
external	**sub-external** 10.2.1.0/24
internal	**sub-internal** 172.25.1.0/24

Fig. 9. The internal and external networks.

After the networks are created, a router is created in the project by clicking on the Create Router option under the Routers section. Following the creation of the router, the external network is marked as 'external' from the admin panel by the admin user. Once the external network is set, the router is set as the gateway from the project panel to this newly marked external network. In addition, the router is then configured to have an internal interface to route the instance traffic to the external network.

Once created and functioning, the external network is assigned a floating IP from the external network, giving the instances an isolated LAN while using Network Address Translation through the networking node to access the Internet. This allows for full connectivity in and out of the internal network.

III. Results and Experiments

A. Instance Creation

The two created instances are load balanced across the compute1 and compute2 nodes as shown by the instance overview properties in Fig. 10.

Project	Host	Name
project1	compute2.local	instance2
project1	compute1.local	instance1

Fig. 10. The automatic distribution across nodes.

B. Instance Connectivity

With two instances running on the same subnet, each instance is able to ping each other with minimal latency as shown by Fig. 11.

```
$ ping 172.25.1.101
PING 172.25.1.101 (172.25.1.101): 56 data bytes
64 bytes from 172.25.1.101: seq=0 ttl=64 time=3.050 ms
64 bytes from 172.25.1.101: seq=1 ttl=64 time=1.017 ms
64 bytes from 172.25.1.101: seq=2 ttl=64 time=0.583 ms
```

Fig. 11. A snapshot of the internal connectivity.

C. Compute Node Addition

Fig. 12 shows the Horizon Dashboard before the node addition and Fig. 13 shows after the node addition.

Displaying 3 items

Hostname	Type	VCPUs (used)	VCPUs
compute1.local	QEMU	2	24
compute2.local	QEMU	2	24
compute3.local	QEMU	0	8

Displaying 3 items

Fig. 12. Before compute node addition.

Displaying 4 items

Hostname	Type	VCPUs (used)	VCPUs (total)
compute1.local	QEMU	2	24
compute2.local	QEMU	2	24
compute3.local	QEMU	0	8
compute4.local	QEMU	0	24

Displaying 4 items

Fig. 13. After compute node addition: compute4 node has been added.

D. Stress Test

Fig. 14 shows a comparison of what is assigned and what is actually available in the OpenStack deployment. In this case, no cap was set on the quotas.

IV. Acknowledgment

The authors would like to thank the OpenStack Foundation.

Stress Test Quota		
	Actual	Assigned
vCPU	80	∞
RAM	46	∞

Fig. 14. Stress test quota.

Samuel A. Heuchert will be attending the University of the Cumberlands to pursue a Ph.D. in IT while specializing in Blockchain research. He completed his Master of Science degree in Cyber Defense from Dakota State University in 2020. He received his Bachelor of Arts (BA) in Information Technology from the University of Jamestown in North Dakota and holds the CCNP certification. In 2020, he contributed to the Software-Defined Perimeter (SDP) and Zero Trust publication by the Cloud Security Alliance and is an active participant in industry topic research. His research interests include fog computing, network security, blockchain, and automation techniques to increase the efficiency of computing, both in the cloud and on-premises.

Bhaskar P. Rimal is an Assistant Professor of Computer and Cyber Sciences at Dakota State University, Madison, USA. He was a Visiting Assistant Professor and Academic Specialist at the Department of Computer Science at Tennessee Tech. University, Cookeville. He was a Visiting Scholar with the Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA. He currently serves as a Technical Editor for the *IEEE Network Magazine*, Associate Editor for the *IEEE Access*, Associate Editor for the *EURASIP Journal on Wireless Communications and Networking*. Further, he served as an Associate Technical Editor for the *IEEE Communications Magazine*, and an Editor for the *Internet Technology Letters*.

Martin Reisslein received the Ph.D. degree in systems engineering from the University of Pennsylvania, Philadelphia, in 1998. He is currently a Professor with the School of Electrical, Computer, and Energy Engineering, Arizona State University (ASU), Tempe. He serves as an Associate Editor for the *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Network and Service Management*, *IEEE Transactions on Education*, *IEEE Access*, and *Computer Networks*. He is also Area Editor for Optical Communication of the *IEEE Communications Surveys and Tutorials* and a Co-Editor-in-Chief of *Optical Switching and Networking*.

Yong Wang is an Associate Professor at Dakota State University. He received his B.S. and M.S.E degrees in Computer Science from Wuhan University (China) in 1995 and 1998 respectively. He received his Ph.D. degree in Computer Science from the University of Nebraska-Lincoln in 2007. His research focuses on the Internet of Things and data security and privacy. Since he joined DSU in 2012, Dr. Wang has received five awards from National Science Foundation. He has published 70+ peer-reviewed papers in prestigious journals/conferences and served as reviewers and technical program committee members for many journals and conferences.