Technical Documentation

Deployment and Implementation of CloudLab OpenStack Profile to Support Docker Virtualization

Austin Bramley, Akash Kumar, Endri Koti, Simeon McGraw, Andrew Valenci

Computer Science Department

West Chester University

700 S High St, West Chester, PA 19382

ABSTRACT

The CloudLab OpenStack default profile contains the setup of 1 compute node that supports KVM-based virtual machines without the support for Docker. The primary task of this project is to augment the CloudLab OpenStack default profile to support the inclusion of a compute node type that provides support for Docker virtualization. Through the use of Zun, an OpenStack container service, we augmented a compute node to support the launching of a Docker container as an OpenStack managed resource. The coded bash scripts added to the default Openstack profile provide automated installation of systems necessary to support Docker virtualization. Currently we have augmented the default Openstack profile to install Etcd, Zun, and Kuryr in the controller node. Also, we provided the installations of Docker, Zun, and Kuryr in the compute node.

Keywords: Cloud Computing, Docker Virtualization, OpenStack, Zun, CloudLab, Automated Bash Script

1. INTRODUCTION

1.1. Purpose

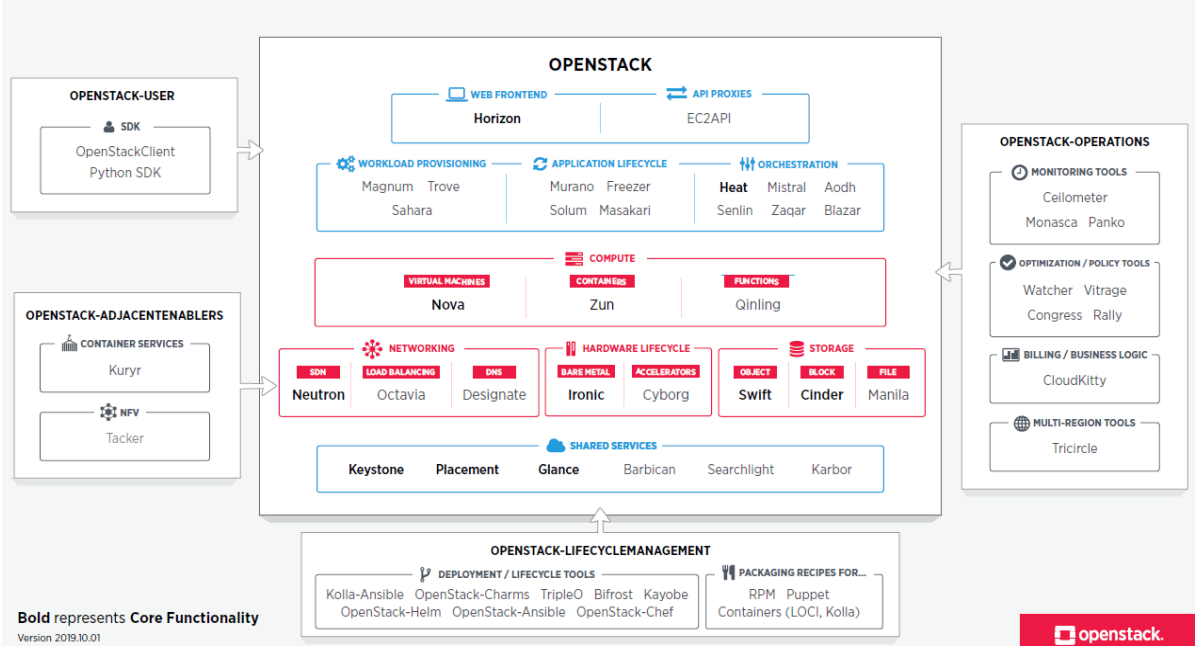
The purpose of this project is to modify the default Cloudlab Openstack profile to provide support for a compute node that allows for Docker virtualization managed by the Openstack dashboard. The current Openstack default profile provides support for Kernel-based Virtual Machines but lacks support for Docker virtualization. Our goal is to implement the Zun Container service to provide the default Openstack profile with the ability to manage containers such as Docker.

1.2. CloudLab

CloudLab is part of the National Science Foundation's NSFCloud program. CloudLab is a project of the University of Utah, Clemson University, the University of Wisconsin Madison, the University of Massachusetts Amherst, Raytheon BBN Technologies, and US Ignite [1]. CloudLab is a testbed designed to allow researchers to experiment with cloud architectures and the new applications that they enable. It is built for running experiments that will lead to new capabilities in future clouds, or to a deeper understanding of the fundamentals of cloud computing [1]. CloudLab is ideally suited to experiments that cannot be run in traditional clouds because they require control and/or visibility over parts of the system that would be “givens” in other clouds, such as the virtualization, storage, or network layers. CloudLab is built around profiles. A profile is a description of everything needed to build a cloud: the physical hardware (servers, disks, switches) and the software needed to transform it into a particular type of cloud. A profile is fully packaged and automated: building a cloud from scratch may take only minutes, depending on its size and complexity [1].

1.3. OpenStack

OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed and provisioned through APIs with common authentication mechanisms [4]. It is a free open standard cloud computing platform, mostly deployed as infrastructure-as-a-service (IaaS) in both public and private clouds where virtual servers and other resources are made available to users [3]. Users either manage it through a web-based dashboard, through command-line tools, or through RESTful web services. A dashboard is also available, giving administrators control while empowering their users to provision resources through a web interface [4]. Beyond standard infrastructure-as-a-service functionality, additional components provide orchestration, fault management and service management amongst other services to ensure high availability of user applications [3].
The history of OpenStack began in 2010 as a joint project of Rackspace Hosting and NASA. As of 2012, it is managed by the OpenStack Foundation, a non-profit corporate entity established in September 2012 to promote OpenStack software and its community. More than 500 companies have joined the project [4].



1.4. Zun

Zun is an OpenStack Container service. It provides OpenStack the ability to manage application containers such as Docker. Zun provides API endpoints for Openstack to integrate with other OpenStack Services, such as Keystone, Neutron and Glance [8]. The Zun Compute service allows OpenStack to manage container services, including Docker, and uses Kuryr to connect the containers to Neutron. Zun can also connect to Glance, an option for storing container images, much like DockerHub [9]. Zun also has the ability to connect to block storage through Cinder, OpenStack's block storage service.

1.5. Docker

Docker container technology was launched in 2013 as an open source Docker engine that leveraged existing computing concepts around containers [5]. Docker focuses on the requirements of developers and systems operators to separate application dependencies from infrastructure. Containers are used as an abstraction at the app later to package code and dependencies together with the ability to run multiple on the same machine and share the OS kernel [6]. Containers and virtual machines function differently because containers virtualize the operating system instead of hardware making them more portable and efficient.

2. PROJECT PLAN

2.1. Creating Experiment

The First step of our project was to launch the CloudLab OpenStack default profile and manually install the systems required for Docker virtualization. The control node requires the installation of Etcd, Kuryr and Zun services. Etcd provides a reliable way to store data that needs to be accessed by a cluster of machines [10], [12], [14]. Also the compute node required the installation of Docker, Kuryr, and Zun services [11], [13]. Applications can easily read from and write data to Etcd. Kuryr is designed to function as a bridge between container frameworks and OpenStack. Refer to 1.4 and 1.5 for Zun and Docker details respectively. Once the required services are installed on both the controller and compute nodes it is possible to launch Docker as an OpenStack managed resource [15].

2.2. Implementation and Deployment

The installation steps and guide of each of the services into the controller and compute nodes, helped us create two separate sets of bash scripts. One for the controller node and one for the compute node. The coded bash scripts help with the automated installation. The scripts are added to the setup scripts in the CloudLab OpenStack profile to install these services in the CloudLab experiment's setup phase. For a more detailed implementation and deployment of this project, please refer to the GitHube repository of this project: "*https://github.com/ab922530/496-cloud-project"* and its technical README file: "*https://github.com/ab922530/496-cloud-project/blob/master/README.md*."

2.3. Progress

We have successfully created and implemented in the CloudLab OpenStack default profile a bash script to install the services required to complete our task. The bash script runs successfully and installs all necessary services to both the controller and compute node. We are currently working through issues on the final phase of the project regarding launching Docker container.

3. PROJECT DELIVERABLE

3.1. Assessment

Considering that the team has worked on this project for over 30 hours collectively and also contributed X amount of hours independently on the following; properly setting up a CloudLab experiment from the CloudLab OpenStack default profile, completed the installation of Etcd, Docker, Kuryr, and Zun on the experiment, coded the bash scripts for automated installation of Docker in the compute node [19], wrote technical documentation, filmed the presentation video, and completed the README file of the project in GitHub repository [18], [17], we are proud to say that the team has completed 98% of the project.

Even though the team contributed a lot of technical expertise and effort in completion of this project, unfortunately we are not able to finalize the last step of the project which is launching the Docker container due to an "*Internal Server Error [http]: 500 Error*." We ran into other technical errors while setting up the experiment which the team was able to fix with their technical expertise. All the issues that we faced and the main issue of the project are explained more in detail in the section 3.2.

3.2. Technical Errors

Most of the technical errors that we faced were related with the installation of Zun and Docker containers.

The first error that the team faced was during the lunch of the "*zun-config file*" when we tried to generate a sample configuration file which helps to configure the components in the controller node [10]. The issue of this error was that the *"zun-config-generator.conf"* was not in the correct path.

```
root@ctl:/var/lib/zun/zun/zun# su -s /bin/sh -c "oslo-config-generator --config-file etc/zun/zun-config-generator.conf" zun
Traceback (most recent call last):
  File "/usr/bin/oslo-config-generator", line 10, in <module>
    sys.exit(main())
  File "/usr/lib/python3/dist-packages/oslo_config/generator.py", line 777, in main
    conf(args, version=version)
  File "/usr/lib/python3/dist-packages/oslo_config/cfg.py", line 2120, in __call__
    raise ConfigFilesNotFoundError(self._namespace._files_not_found)
oslo_config.cfg.ConfigFilesNotFoundError: Failed to find some config files: /var/lib/zun/zun/zun/etc/zun/zun-config-generator.conf
root@ctl:/var/lib/zun/zun# su -s /bin/sh -c "oslo-config-generator --config-file etc/zun/zun-config-generator.conf" zun
WARNING:stevedore.named:Could not load zun.conf
```

We were able to fix this issue by removing two zun folders "*/zun/zun*" and setting the file "*zun.conf*" in the correct path "*/var/lib/zun*" and executing the command "*su -s /bin/sh -c "oslo-config-generator --config-file etc/zun/zun-config-generator.conf" zun*" in that directory.
The second error that the team faced was when we edited the "*/etc/default/etcd file*" [14] for the Etcd Service.

```
root@cp-1:/etc/default# journalctl -xe
Apr 20 14:30:55 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us systemd[1]: Reloading.
Apr 20 14:30:55 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us systemd[1]: Reloading.
Apr 20 14:30:56 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us systemd[1]: Reloading.
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us systemd[1]: Starting etcd
-- Subject: Unit etcd.service has begun start-up
-- Defined-By: systemd
-- Support: http://www.ubuntu.com/support
--
-- Unit etcd.service has begun starting up.
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: recognized a
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: recognized a
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: recognized a
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: recognized a
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: recognized a
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: recognized a
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: recognized a
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: recognized a
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: recognized a
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: etcd Version
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: Git SHA: Not
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: Go Version:
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: Go OS/Arch:
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: setting maxi
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: found invali
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: listen tcp 1
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us systemd[1]: etcd.service:
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us systemd[1]: etcd.service:
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us systemd[1]: Failed to sta
-- Subject: Unit etcd.service has failed
-- Defined-By: systemd
-- Support: http://www.ubuntu.com/support
--
-- Unit etcd.service has failed.
--
-- The result is RESULT.
Apr 20 14:31:18 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us sshd[23868]: Received dis
Apr 20 14:31:18 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us sshd[23868]: Disconnected
lines 1652-1688/1688 (END)
```

```
Last login: Tue Apr 21 14:03:26 2020 from 108.52.44.133
ab922530@cp-1:~$ sudo su
root@cp-1:/users/ab922530# journalctl -xn -u etcd -fl
-- Logs begin at Mon 2020-04-13 10:35:30 CDT. --
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: etcd Version: 3.2.17
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: Git SHA: Not provided (use ./build instead of go build)
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: Go Version: go1.10
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: Go OS/Arch: linux/amd64
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: setting maximum number of CPUs to 40, total number of available CPUs is 40
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: found invalid file/dir default under data dir /var/lib/etcd (Ignore this if you
 are upgrading etcd)
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: listen tcp 10.11.10.1:2380: bind: cannot assign requested address
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us systemd[1]: etcd.service: Main process exited, code=exited, status=1/FAILURE
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us systemd[1]: etcd.service: Failed with result 'exit-code'.
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us systemd[1]: Failed to start etcd - highly-available key value store.
-- Subject: Unit etcd.service has failed
-- Defined-By: systemd
-- Support: http://www.ubuntu.com/support
--
-- Unit etcd.service has failed.
--
```

After evaluating the error we came to realize that in the Etcd file was not assigned the management IP address of the controller node for the ETCD_INITIAL_CLUSTER, ETCD_INITIAL_ADVERTISE_PEER_URLS, ETCD_ADVERTISE_CLIENT_URLS, ETCD_LISTEN_CLIENT_URLS.

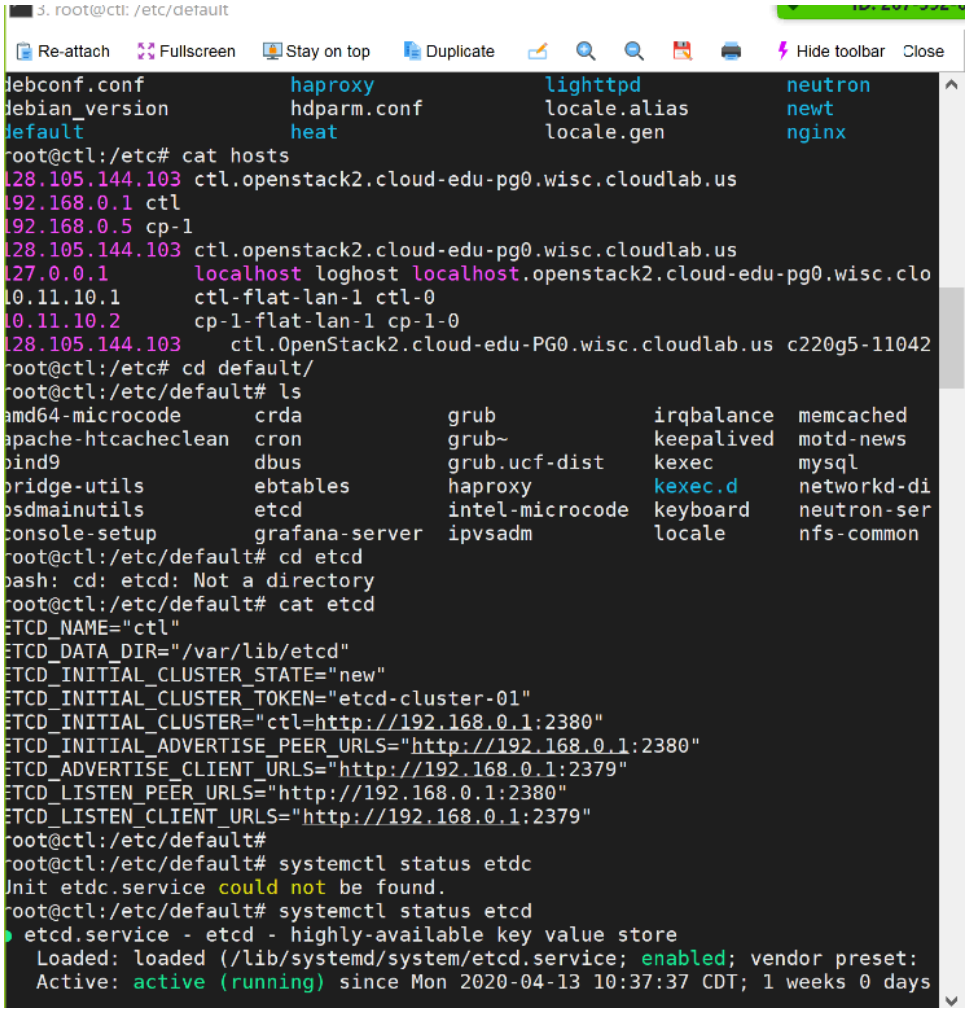
```
-- Unit etcd.service has begun starting up.
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: recognized and used environment variable ETCD_ADVERTISE_CLIENT_URLS=http://10.11.10.1:23
79
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: recognized and used environment variable ETCD_DATA_DIR=/var/lib/etcd
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: recognized and used environment variable ETCD_INITIAL_ADVERTISE_PEER_URLS=http://10.11.1
0.1:2380
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: recognized and used environment variable ETCD_INITIAL_CLUSTER=controller=http://10.11.1
0.1:2380
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: recognized and used environment variable ETCD_INITIAL_CLUSTER_STATE=new
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: recognized and used environment variable ETCD_INITIAL_CLUSTER_TOKEN=etcd-cluster-01
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: recognized and used environment variable ETCD_LISTEN_CLIENT_URLS=http://10.11.10.1:2379
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: recognized and used environment variable ETCD_LISTEN_PEER_URLS=http://10.11.10.1:2380
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: recognized and used environment variable ETCD_NAME=controller
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: etcd Version: 3.2.17
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: Git SHA: Not provided (use ./build instead of go build)
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: Go Version: go1.10
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: Go OS/Arch: linux/amd64
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: setting maximum number of CPUs to 40, total number of available CPUs is 40
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: found invalid file/dir default under data dir /var/lib/etcd (Ignore this if you are upgr
ading et
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us etcd[23839]: listen tcp 10.11.10.1:2380: bind: cannot assign requested address
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us systemd[1]: etcd.service: Main process exited, code=exited, status=1/FAILURE
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us systemd[1]: etcd.service: Failed with result 'exit-code'.
Apr 20 14:30:59 cp-1.openstack2.cloud-edu-pg0.wisc.cloudlab.us systemd[1]: Failed to start etcd - highly-available key value store.
-- Subject: Unit etcd.service has failed
-- Defined-By: systemd
-- Support: http://www.ubuntu.com/support
--
-- Unit etcd.service has failed.
```

We fixed this error by assigning the correct management IP address of the controller node "*ctl: 192.168.0.1*" in the ETCD_INITIAL_CLUSTER, ETCD_INITIAL_ADVERTISE_PEER_URLS, ETCD_ADVERTISE_CLIENT_URLS, ETCD_LISTEN_CLIENT_URLS to enable access by other nodes via the management network.

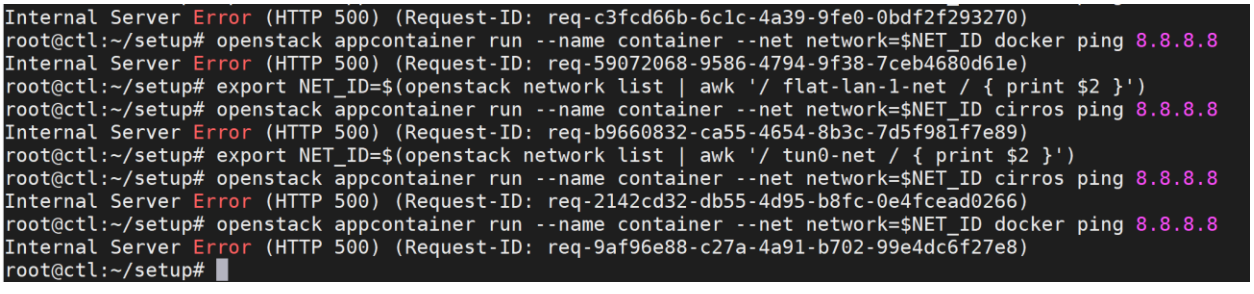
```
2. root@cp-1: /etc/default        3. root@ctl: /etc/default
  GNU nano 2.9.3                                  etcd


##### -client-cert-auth
## Enable client cert authentication.
## default: false
# ETCD_CLIENT_CERT_AUTH
ETCD_NAME="controller"
ETCD_DATA_DIR="/var/lib/etcd"
ETCD_INITIAL_CLUSTER_STATE="new"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster-01"
ETCD_INITIAL_CLUSTER="controller=http://192.168.0.1:2380"
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://192.168.0.1:2380"
ETCD_ADVERTISE_CLIENT_URLS="http://192.168.0.1:2379"
ETCD_LISTEN_PEER_URLS="http://0.0.0.0:2380"
ETCD_LISTEN_CLIENT_URLS="http://192.168.0.1:2379"
##### -trusted-ca-file
## Path to the client server TLS trusted CA key file.
## default: none
# ETCD_TRUSTED_CA_FILE
```

Running the Etcd service:

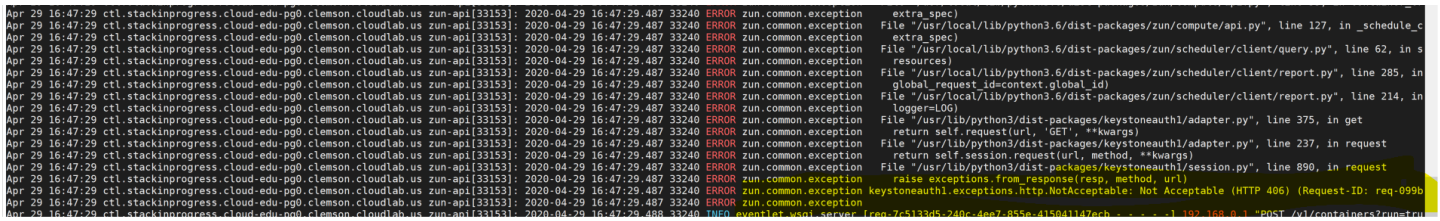


The final and the main issue that we faced on this project happened when we tried to launch a container after verifying that our app container service was operational. When attempting to launch a container on our Self Service network using the line "*openstack appcontainer run --name container --net network=$NET_ID docker ping 8.8.8.8*," this line resulted in the container being launched in an error state and we received an Internal Server 500 error as seen below.



In order to investigate further, we ran the command "*journalctl -u zun-api*" to get a detailed report regarding the error thrown during the launching of the Docker container. Further

investigation revealed that the error was due to our Keystone authorization (See below screenshot) which was configured at the beginning of our project inside the "*zun.conf file*." We returned to our keystone authorization configuration and ensured that everything was configured correctly. Then through many rounds of adjustments to the configuration and attempts at launching a container, we remained unsuccessful and continued to receive the same 500 error.



4. TEAM COLABORATION

Due to the advent of the 2019-2020 coronavirus pandemic, direct in-person communication between team members was not possible. So, we resorted to using online means of communication to collaborate and stay connected with each other and with our professor. Here are some of the online tools and platforms we used:

- *GitHub* - for collaboration and sharing of project code.
- *Zoom* - for face-to-face communication and screen sharing.
- *Piazza* - for asking the professor questions when encountering issues.
- *Slack* - for instant communication and team coordination.
- *Google Drive* - for editing and sharing documents together.
- *Trello* - for creating a project board to track project progress and key information.

In addition, we scheduled Zoom meetings three times a week Monday, Wednesday, and Friday from 2.p.m to 3:15.p.m to ensure that all team members are actively engaged with the work and keep the project on schedule.

5. CONCLUSION

In conclusion, being a cloud provider is not an easy job. It requires deep knowledge of cloud platforms, infrastructure, networking, containerization, virtualization, Linux, bash scripting, etc. As a team even though we were not able to accomplish our main goal, we learned a lot completing this project which broadened our knowledge in Cloud computing and Linux distribution.

6. REFERENCES

[1]. *CloudLab: https://www.cloudlab.us/#availability*
[2]. *CloudLab OpenStack default profile: https://gitlab.flux.utah.edu/johnsond/openstack-build-ubuntu*
[3]. *Intro to OpenStack: https://www.openstack.org/software/*
[4]. *OpenStack Wikipedia page: https://en.wikipedia.org/wiki/OpenStack*
[5]. *Intro to Docker: https://docs.docker.com/get-started/*
[6]. *Docker documentation: https://docs.docker.com/get-docker/*
[7]. *Installation Docker Engine on Ubuntu: https://docs.docker.com/engine/install/ubuntu/*

[8]. *Zun Wiki page: https://wiki.openstack.org/wiki/Zun*
[9]. *Zun documentation: https://docs.openstack.org/zun/latest/*
[10]. *Installation and configure Zun in controller node: https://docs.openstack.org/zun/latest/install/controller-install.html*
[11]. *Installation and configure Zun in the compute node: https://docs.openstack.org/zun/latest/install/compute-install.html*
[12]. *Installation and configure Kuryr in the controller node: https://docs.openstack.org/kuryr-libnetwork/latest/install/controller-install.html*
[13]. *Installation and configure Kuryr in the compute node for Ubuntu: https://docs.openstack.org/kuryr-libnetwork/latest/install/compute-install-ubuntu.html*
[14]. *Installation of Etcd in the controller node for Ubuntu: https://docs.openstack.org/install-guide/environment-etcd-ubuntu.html*
[15]. *Launch a container in Zun: https://docs.openstack.org/zun/latest/install/launch-container.html*
[16]. *OpenStack Forums: https://ask.openstack.org/*
[17]. *Project's GitHub Repository: https://github.com/ab922530/496-cloud-project*
[18]. *Project's README File: https://github.com/ab922530/496-cloud-project/blob/master/README.md*
[19]. *Bash Script: https://github.com/simeonjmcg/cloudlab-zun-docker*