INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

भारतीय प्रौद्योगिकी संस्थान खड़गपुर

1951

योगः कर्मसु कौशलम्

Distributed Cloud Storage for Health Monitoring Data

Cloud Computing (CS60118)
Spring 2020-2021
Group Number
Instructor: Prof. S. Misra

K Snehal Reddy (17CS30020)
Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, West Bengal, India

P Amshumaan Varma (17CS30025)
Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, West Bengal, India

Sandeep Kothapalli (17CS10021)
Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, West Bengal, India

Robin Babu Padamadan (17CS10045)
Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, West Bengal, India

Contents

1. Introduction

With an increase in the number of smart wearable devices that have the ability to collect health information in real-time, we are sitting on a gold mine of health data corpus. The data could be very useful in analysing the health of a user and could potentially warn the users before few health conditions inflict them. However, the data size could potentially be huge given the ever-increasing number of users whose metrics are being monitored in real-time.

We aim to build software to aid in health monitoring using real-time biometrics of users from health sensors. The possibilities in terms of the processing that could be done on top of the data are endless, and we aim to build a pipeline that could facilitate these operations on the real-time corpus. We employ cloud computing technologies whilst making the best use of its scalability and privacy.

2. Feasibility Study

a) Technical feasibility :

   i) Here, we evaluate the current technologies, which are needed to accomplish customer requirements within the time and budget. As far as the client-facing application is considered, this is a fairly small scale project. Python flask with SQLite database was used to build the backend and HTML + Bootstrap was used to build the frontend; details of whom will be discussed in section 3. But deploying the project has been done on a dedicated architecture built using OpenStack, details of which will be discussed in Section 4.2. Even though this does not take a humongous time to design and deploy, the resource overhead is very high. Multiple virtual machines had to be created to get the project working. To conclude, a similar project when scaled could prove to be less budget-friendly.

b) Economic feasibility:

   i) Here, we decide whether the software can generate financial profits for an organization. If the software is commercialized, after a burn-in period, it is highly likely that the health monitoring app scales rapidly hence generating revenue. The targeted market is huge and everyone needs their health monitored. But penetrating the market could prove to be challenging considering the vast majority of applications already doing this.

3. Technology Used

3.1 Hardware

For the course of developing and testing our project, we used 6 VMs with the following configuration -

- Ubuntu 16.04
- 2GB Ram
- 50GB Memory

3.2 Software

The web application was made primarily with the help of Python based Flask library. The database made use of SQLite (via SQLAlchemy) for the database management system. Necessary front-end for web pages like login, signup, dashboard and user health data interface were written using HTML5 and Bootstrap v5.0. SQLCipher was used for encrypting the database.

3.3 Cloud Services

The web application was deployed on a VM instance deployed on a private cloud built using Openstack. We used the virtual machines (due to resource constraints to communicate among our personal systems) with above configurations simulated using Oracle VirtualBox as individual nodes to deploy the Openstack architecture. We used an opensource tool called Fuel which provides an interface for setting up communication and networking between the nodes to deploy and maintain the Openstack environment on this cluster.

4. Project Description

4.1 Web Application

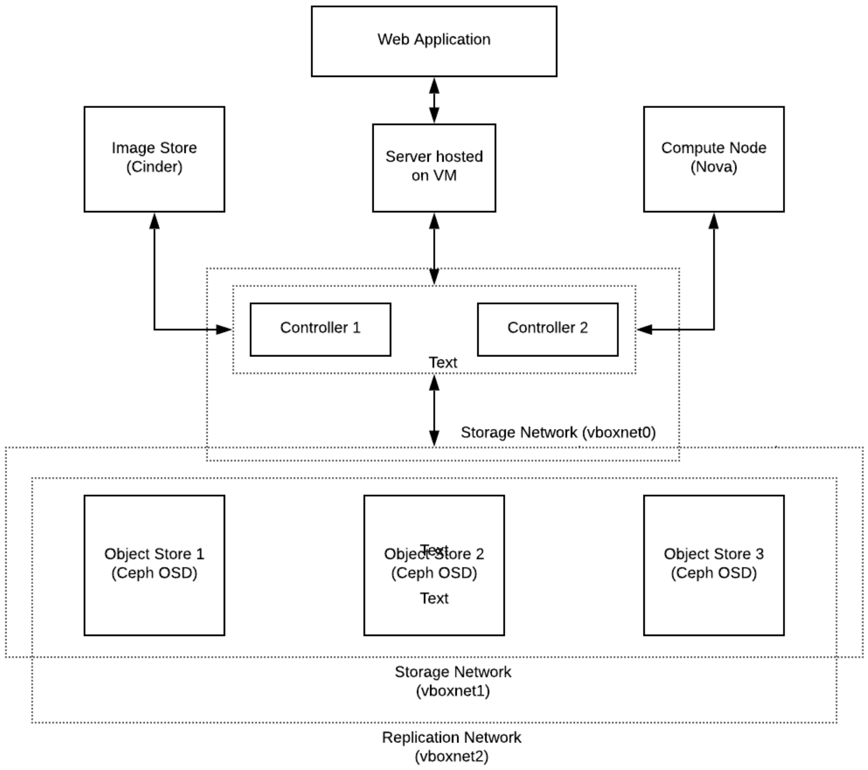
The user signup/login details is the first interface between a user and the web application. After logging in, the user has the functionality to start uploading their data in either exercise mode or sleep mode. The data that we plan to collect, store and analyse are as follows -

- Heart rate
- Blood oxygen levels
- Blood pressure
- Calories burnt

All these are indexed with a timestamp and user id. This data is stored in SQLite databases. The user data tends to be private and hence they are read and written in an encrypted format using SQLCipher.

We will present the minimum, average and maximum data values of the person's heart rate, oxygen saturation levels and blood pressure. This will let the user understand if his vitals are within healthy limits or not. The exercise mode possesses additional information regarding calorie consumption. The sleep mode contains additional information regarding the variance of heartbeat rate which is used to understand the sleep conditions of the person. These operations are very basic and only meant to form a base for more sophisticated processing techniques that could be performed in a similar fashion.

4.2 Deploying on Cloud (Openstack)

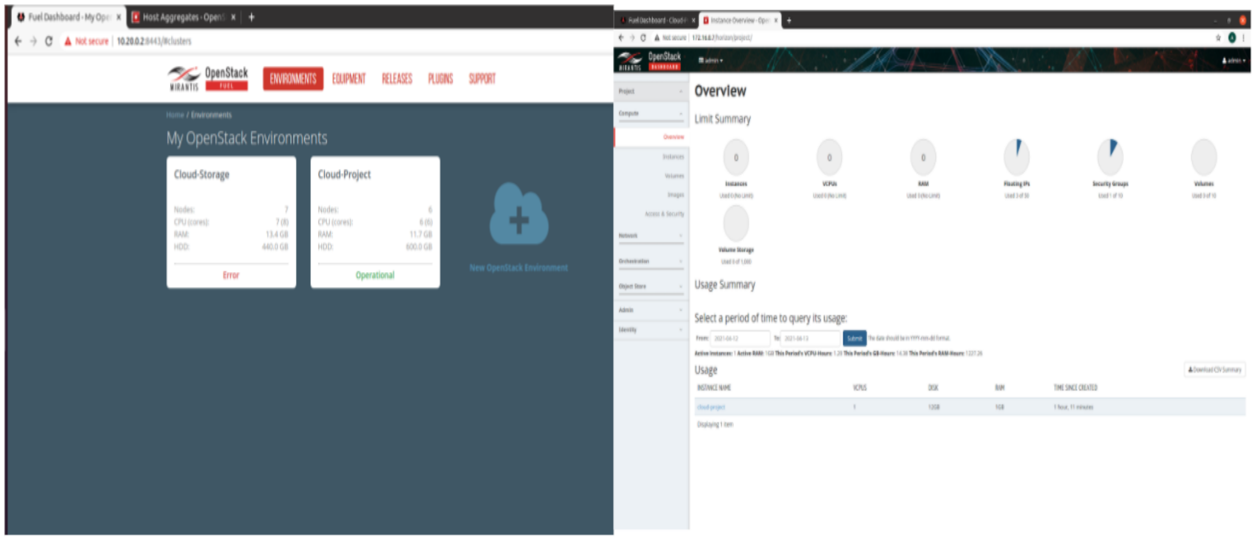

Web Application
Image Store (Cinder)
Server hosted on VM
Compute Node (Nova)
Controller 1
Controller 2
Text
Storage Network (vboxnet0)
Object Store 1 (Ceph OSD)
Object Store 2 (Ceph OSD)
Text
Text
Object Store 3 (Ceph OSD)
Storage Network (vboxnet1)
Replication Network (vboxnet2)

The above figure shows our deployed architecture for our OpenStack environment. The deployed nodes are as follows -

- 2 Controller nodes (Keystone, Neutron etc)
- 1 Compute node (Nova)
- 1 Compute node (Nova)
- 3 Object Storage (Ceph OSD / Swift)

We deployed 3 object storage nodes for fault-tolerance i.e the virtual instance created has its disk replicated across these 3 nodes for it to still be accessible even in case of failure of a few nodes. The Cinder node stores the boot images for the virtual instances and the Nova node maps the virtual resources to the physical hardware. The controller node runs the Identity service, Image service, management portions of Compute, management portion of Networking, various Networking agents, and the dashboard. We used Fuel UI to integrate these nodes into the environment. There is a single master node that has a full deployment of fuel and OpenStack and detects these other nodes in the same network to install necessary OpenStack components as per the node specifications.

Only the master node has a NAT adapter with DHCP enabled so that it has open access across the internet given a public static IP. Other nodes have only local networks and only the master node can interact with them. The object storage nodes also have a replication network setup among themselves to handle the disk replication among the nodes.



My OpenStack Environments

Cloud-Storage	
Nodes:	7
CPU (cores):	7 (8)
RAM:	13.4 GB
HDD:	440.0 GB
Error	

Cloud-Project	
Nodes:	6
CPU (cores):	6 (6)
RAM:	11.7 GB
HDD:	600.0 GB
Operational	

Overview

Limit Summary

Instances	VCPUs	RAM	Floating IPs	Security Groups	Volumes	Volume Storage
0	0	0				
Used 0 (No Limit)	Used 0 (No Limit)	Used 0 (No Limit)	Used 3 of 50	Used 1 of 10	Used 0 of 10	Used 0 of 1,000

Usage Summary

Select a period of time to query its usage:

From: 2021-04-12 To: 2021-04-13 The date should be in YYYY-mm-dd format.

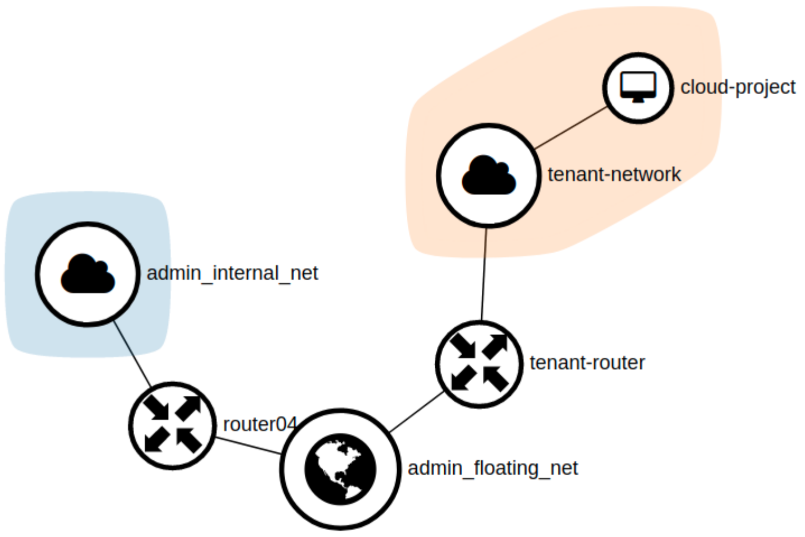
Active Instances: 1 Active RAM: 1GB This Period's VCPU-Hours: 1.20 This Period's GB-Hours: 14.38 This Period's RAM-Hours: 1227.26

Usage

INSTANCE NAME	VCPUS	DISK	RAM	TIME SINCE CREATED
cloud-project	1	12GB	1GB	1 hour, 11 minutes

Displaying 1 item

We can manage the node deployment to the OpenStack cloud from the Fuel UI accessible through the provided public IP address from a system on the same local network. It can detect the other nodes in the same network to install necessary OpenStack components as per the node specifications. The deployed OpenStack model can be accessed directly from the fuel UI and we can also deploy multiple instances of OpenStack on the cluster.



We first create networks, subnets and routers and generate floating ips to assign to the VM. Then we generate security rules and key pairs generated by keystone for ssh into the VM after launching. We then create a ubuntu disk image using glance and cinder to store the image and launch the VM on this network and assign a public floating ip. We then ssh into the VM from localhost and deploy our web application on the openstack VM. The deployed website can be accessed from the client browser using the VM's floating ip address.

5. References

- Jay A. Kreibich. 2010. Using SQLite (1st. ed.). O'Reilly Media, Inc.
- Grinberg, M., 2018. Flask web development: developing web applications with python, " O&#x27;Reilly Media, Inc."
- Alok Shrivastwa, Sunil Sarat, Kevin Jackson, Cody Bunch, Egle Sigler, and Tony Campbell. 2016. OpenStack: Building a Cloud Environment. Packt Publishing.
- https://docs.openstack.org/fuel-docs/mitaka/userdocs/fuel-install-guide/intro/intro_fuel_intro.html
- https://www.mirantis.com/software/mcp/openstack/
- Brian Walters. 1999. VMware Virtual Platform. Linux J. 1999, 63es (July 1999), 6–es.
- Wenying Zeng, Yuelong Zhao, Kairi Ou, and Wei Song. 2009. Research on cloud storage architecture and key technologies. In Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human (ICIS '09). Association for Computing Machinery, New York, NY, USA, 1044–1048. DOI:https://doi.org/10.1145/1655925.1656114