

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the
award of Bachelor of Technology Degree in Information
Technology

By
Karri Dundy Nagaveera Sai Pranay
REG. NO.40120043



DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND
TECHNOLOGY
JEPPIAARNAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119, TAMILNADU

APRIL - 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI

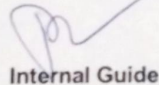
CHENNAI- 600119

www.sathyabama.ac.in

SCHOOL OF COMPUTING

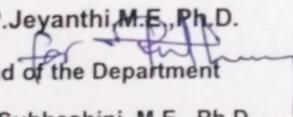
BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Karri dundy Nagaveera Sai Pranay (Reg. No: 40120043)** who carried out the project entitled **"Graphical interface to identify unaccessed VMS"** under supervision from FEBRUARY 2023 to APRIL 2023



Internal Guide

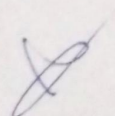
Dr.P.Jeyanthi, M.E., Ph.D.



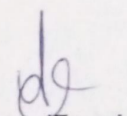
Head of the Department

Dr.R.Subhashini, M.E., Ph.D.

Submitted for Viva voce Examination held on 28/4/23



Internal Examiner



External Examiner

DECLARATION

I, Karri Dundy Nagaveera Sai Pranay hereby declare that the project report entitled "Graphical interface to identify unaccessed VM's " done by me under the guidance of DR.P.Jeyanthi submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Information Technology.

DATE: 28/4/23

PLACE: CHENNAI

Pranay
SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D, School of Computing** and **Dr. R. Subhashini M.E., Ph.D., Head of the Department of Information Technology** providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. P. Jeyanthi, M.E., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE

Ref.No.: 20 /FSPCC/BC001/04/2023



इलेक्ट्रॉनिक्स एवं
सूचना प्रौद्योगिकी मंत्रालय
MINISTRY OF
ELECTRONICS AND
INFORMATION TECHNOLOGY
सायमेन बसो



COURSE COMPLETION LETTER

It is certified that

Karridundy Nagaveerasaipranay

Sathyabama Institute of Science and Technology

has successfully undergone the Bridge Course Training Program on

Cloud Computing Technology

Under "FutureSkills PRIME" Programme

commenced from 11 Mar, 2023

Shri. V. Sridharan

Chief Investigator

Cloud Computing Technology

Lead Resource Centre

L R PRAKASH

Digitally signed by L. R. PRAKASH
Date: 2023.04.26 02:02:33
+05'30'

Cdr.(Retd.) L.R. Prakash

Senior Director

Centre Head

C-DAC, Chennai

ABSTRACT

The OpenStack dashboard provides a user-friendly graphical interface that allows users to access virtual machines quickly and easily. They can power on an unaccessed virtual machine by clicking on the instance name and then clicking the "Start" button, which makes it accessible again.

The Instances tab provides a table that shows various details about each virtual machine, including its name, status, IP address, and more. The "Status" column is particularly useful for identifying unaccessed virtual machines as it shows the current status of each virtual machine, such as "Shutoff" or "Paused."

Users can filter the virtual machines by their status to view only the ones that are not accessible. Once they have identified an unaccessed virtual machine, they can select it to view more details about it, such as its name, IP address, and more.

TABLE OF CONTENTS

CHAPTER NO:	TITLE	PAGENO:
	ABSTARCT	i
	LIST OF FIGURES	iv
	LIST OF ABBREVIATIONS	v
1	INTODUCTION	1
2	AIM AND SCOPE	2
	2.1Aim	2
	2.2Scope	2
3	METHODOLOGY	3
	3.1Installation of ubuntu	3
	3.2Installation of open stack	4
	3.3System Architecture	5
	3.4Graphical Interface	6
	3.5System Requirements	8
	3.5.1 Hardware Requirements	8
	3.5.2 Software Requirements	8
	3.5.3 Programming Skills	8
	3.5.4 Open stack API Knowledge	8
4	RESULT	9

5	SUMMARY AND CONCLUSION	10
6	APPENDIX AND SNAPSHOTS	11
	6.1Appendix	11
	6.2Snapshots	14
7	REFERENCES	19

LIST OF FIGURES

FIGURE NO:	FIGURE NAME:	PAGE NO:
3.1	System Architecture	5
4.1	Graphical Interface	9
6.1	Ubuntu Version	14
6.2	Ubuntu Commands	15
6.3	Open Stack Interface	16
6.4	Launch Instance	16
6.5	Instance Details	17
6.6	Instance Launched	17
6.7	Inactive Instances	18
6.8	Graphical Interface	18

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
VM	Virtual Machine
CLI	Command Line Interface
LTS	Long Term Support
LPS	Lightweight Portable Support
API	Application programming Interface
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheet

CHAPTER 1

INTRODUCTION

OpenStack is a widely used open-source cloud computing platform that provides a scalable and flexible infrastructure for running various types of workloads. It allows users to create and manage virtual machines (VMs) using a command-line interface (CLI) or a web-based dashboard.

One of the key features of OpenStack is the ability to identify unaccessed virtual machines using a graphical interface. This feature is essential for cloud administrators and users to manage their cloud infrastructure efficiently.

To identify unaccessed virtual machines in OpenStack using a graphical interface, users need to log in to the OpenStack dashboard, which is accessible via a web browser. From there, they can go to the "Instances" tab, which displays a list of all the virtual machines running in the cloud.

The Instances tab provides a table that shows various details about each virtual machine, including its name, status, IP address, and more. The "Status" column is particularly useful for identifying unaccessed virtual machines as it shows the current status of each virtual machine, such as "Shutoff" or "Paused."

Users can filter the virtual machines by their status to view only the ones that are not accessible. Once they have identified an unaccessed virtual machine, they can select it to view more details about it, such as its name, IP address, and more.

The OpenStack dashboard provides a user-friendly graphical interface that allows users to access virtual machines quickly and easily. They can power on an unaccessed virtual machine by clicking on the instance name and then clicking the "Start" button, which makes it accessible again.

In conclusion, the graphical interface to identify unaccessed virtual machines in OpenStack is a powerful tool that helps cloud administrators and users manage their cloud infrastructure efficiently. It provides a user-friendly way to identify and access virtual machines, enabling them to operate their workloads seamlessly.

CHAPTER 2

AIM AND SCOPE

2.1 AIM

The aim of the project “Graphical interface to identify un accessed virtual machines” is to create a graphical web interface to identify virtual machines that are not accessed over a certain period of time (Last usage since 1 week,1 month, 3 months, 6 months, more than one year).Access refers to login to the VMs. The timeline should be in the dropdown list and all virtual machines in the Cloud satisfying the condition should be listed dynamically.

2.2 SCOPE

The scope of the graphical interface to identify un accessed virtual machines in OpenStack is to provide cloud administrators and users with an easy-to-use tool for managing their cloud infrastructure efficiently. This tool allows them to quickly identify un accessed virtual machines and take appropriate actions to make them accessible again, thereby minimizing downtime and ensuring optimal performance of their workloads.

The graphical interface allows users to view a list of all the virtual machines running on the OpenStack cloud and filter them based on their status. This feature helps them to identify un accessed virtual machines quickly and take appropriate action to make them accessible again. By providing a user-friendly way to manage virtual machines, the graphical interface helps to reduce the learning curve and streamline the management of OpenStack clouds. Moreover, the graphical interface enhances the overall user experience by providing a visual representation of the virtual machines, making it easier to navigate and manage the cloud infrastructure. It also allows users to access the virtual machines from any device with an internet connection, making it possible to manage the cloud infrastructure remotely. Overall, the objective of the graphical interface to identify un accessed virtual machines in OpenStack is to provide users with a user-friendly and efficient way to manage their cloud infrastructure, thereby improving the overall performance and availability of their workloads.

CHAPTER 3

METHODOLOGY

3.1 INSTALLATION OF UBUNTU

First we need to install the Oracle VM Virtual box.

Then we need to download ISO file for Ubuntu. For that, all the versions of Ubuntu are available on the official site ubuntu.com.

After the downloading is over, you can install Ubuntu on VirtualBox with the help of following instructions:

Step 1: Open VirtualBox and click on the New button.

Step 2: Give a name to your Virtual Machine and select the location for it to install.

Step 3: Assign RAM size to your Virtual Machine.

Step 4: Create a Virtual Hard disk for the machine to store files.

Step 5: Select the type of Hard disk. Using VDI type is recommended.

Step 6: Either of the physical storage type can be selected. Using Dynamically allocated disk is by default recommended.

Step 7: Select disk size and provide the destination folder to install.

Step 8: After the Disk creation is done, boot the Virtual Machine and begin installing Ubuntu.

Step 9: Proceed with the installation file and wait for further options.

Step 10: Click on the Install Ubuntu option, this might look different for other Ubuntu versions.

Step 11: Select Keyboard layout, if the defaults are compatible, just click on the continue button and proceed.

Step 12: Select installation type. By default, it is set to Normal installation, which is recommended, but it can also be changed to Minimal installation if there is no need for all Ubuntu features.

Step 13: Click on the Install Now button and carry on with the installation.

Do not get worried with the Erase disk option, it will only be effective inside the virtual machine, other system files outside the VirtualBox remain intact.

Step 14: Click on the continue button, and proceed with writing changes on the disk.

Step 15: Select your location to set the Time Zone.

Step 16: Choose a name for your computer and set a password to secure login info.

Step 17: Wait for the installation process to complete.

Step 18: Once the installation process is over, reboot your Virtual Machine.

Step 19: You're finished with the installation process. Now you can use Ubuntu along with the Windows, without creating a dual boot.

3.2 INSTALLATION OF OPENSTACK

- After the installation of ubuntu we need to install the open stack using micro stack in your ubuntu with the help of terminal.
- The following process shows how to install open stack.
- open the terminal in ubuntu.
- First we need to install the micro stack and this is the beta version and the requirement for this particular micro stack would be 8 GB of memory 100 GB hard disk and it was actually tested on ubuntu 1804 LTS and 2004 LPS.
- Use the command `sudo snap install micro stack --beta` to install the micro stack.
- After sometime the micro stack will be installed. We can check the version of snap micro stack using the command `snap list micro stack` and the version will be displayed.
- Next comes the initialization of all the open stack services and the deployment of all the services so for that we have to use micro stack command `sudo micro stack init --auto --control`.
- And in the log line you will see multiple services getting created , deployed and configured using the micro stack command line.
- For example we can see Horizon dashboard and subsequently rabbit mq etc.
- Let us wait for all the services to get started and initialized by the micro stack and then once it is done we can check the information regarding the open stack.

- At last we can see microstack as initialized in log line.
- Now we can use the typical open stack command for example `microstack.openstack --version` to check the version of open stack installed(5.2.0).
- we can check image list by using command `microstack.openstack image list` and also flavors.
- so for more details regarding the available commands you can type `microstack.openstack --help` | more to get details.
- And move to the dashboard to access the dashboard you will have to use the IP address or your Nic the IP address configured on your Nic.
- And to get the password for your dashboard you will have to run the command `sudo snap get micro stack config.credentials.keystone-password` to get the password for dashboard and a password will be appear in log line.
- opened use the password which was already generated and username as admin.
- Then the open stack will be opened.

3.3 SYSTEM ARCHITECTURE

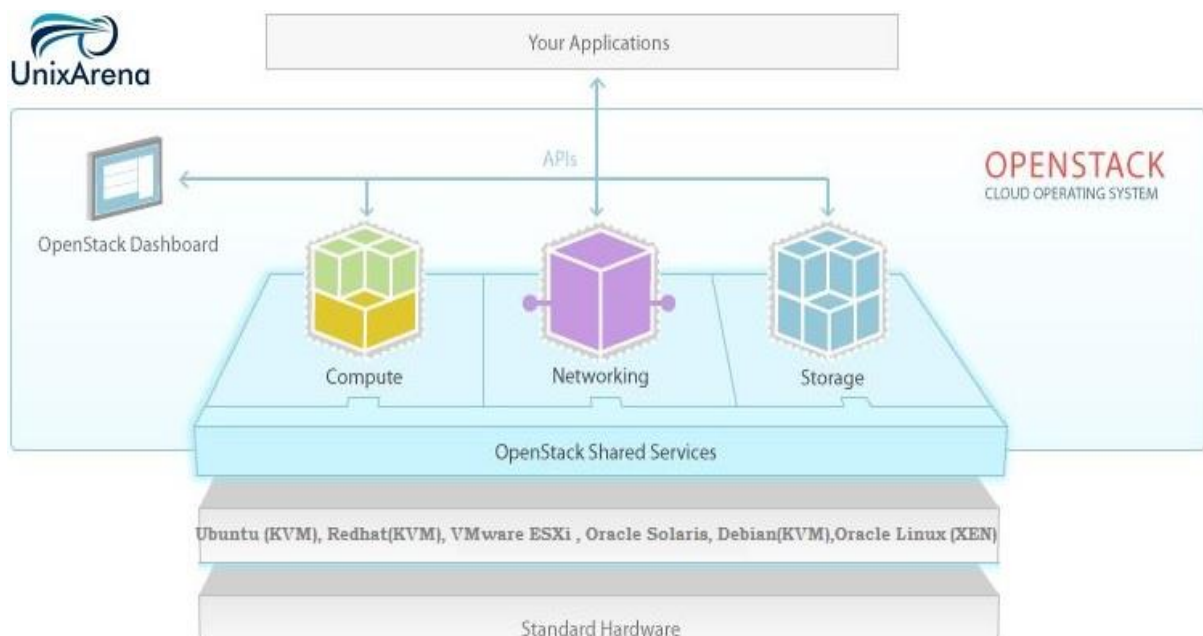


Fig 3.1 Architecture Diagram

The system architecture for the graphical interface to identify un accessed virtual machines in OpenStack consists of several components working together to provide a user-friendly and efficient tool for managing the cloud infrastructure.

The architecture starts with the OpenStack cloud, which is installed and configured on one or more physical servers. The OpenStack cloud provides a scalable and flexible infrastructure for running various types of workloads, including virtual machines.

Next, the dashboard component provides a web-based interface for managing the OpenStack cloud infrastructure. It allows users to view a list of all the virtual machines running on the cloud, filter them based on their status, and take appropriate action to make them accessible again.

The dashboard component communicates with the OpenStack API to retrieve information about the virtual machines, such as their name, status, and IP address. It also communicates with the web server component, which is responsible for serving the virtual machine information to web clients in a web-friendly format.

The web server component can be any web server software, such as Apache or Nginx, that is configured to read the virtual machine information from the OpenStack API and serve it to clients in a web-friendly format. It requires some programming and web development knowledge to configure the web server to read the virtual machine information from the OpenStack API and display it in a user-friendly manner.

3.4 GRAPHICAL INTERFACE

The methodology for creating a graphical interface to identify un accessed virtual machines in OpenStack involves several steps. These steps include installation and configuration of the OpenStack cloud, development of the web server, and implementation of the dashboard component.

The first step in the methodology is to install and configure the OpenStack cloud on one or more physical servers. This involves following the official OpenStack documentation for the installation process. Once the cloud is set up, virtual machines can be created using the OpenStack CLI or dashboard. Next, the web server component needs to be developed to serve the virtual machine information to web clients. Any web server software, such as Apache or Nginx, can be used for this purpose. The web server needs to be configured to read the virtual machine information from the OpenStack API and serve it to clients in a web-friendly format.

The dashboard component is then implemented to provide a graphical interface for managing the virtual machines. The dashboard can be accessed via a web browser and provides a list of all the virtual machines running on the OpenStack cloud. The dashboard communicates with the OpenStack API to retrieve information about the virtual machines and displays it in a user-friendly manner.

To identify unaccessed virtual machines, the dashboard provides a filter by status option. Users can filter the virtual machines by their status, such as "Shutoff" or "Paused," to view only the ones that are not accessible. Once an unaccessed virtual machine is identified, the user can select it to view more details about it, such as its name, IP address, and more. From there, the user can click the "Start" button to power on the instance and make it accessible again.

Overall, the methodology for creating a graphical interface to identify unaccessed virtual machines in OpenStack involves installation and configuration of the OpenStack cloud, development of the web server, and implementation of the dashboard component. It provides a user-friendly and efficient tool for managing the cloud infrastructure and ensuring optimal performance and availability of workloads.

3.5 SYSTEM REQUIREMENTS

3.5.1 HARDWARE REQUIREMENTS:

The hardware requirements for the OpenStack cloud depend on the size of the workload and the number of virtual machines needed. A minimum of two physical servers is required to set up an OpenStack cloud. The servers should have sufficient resources, such as CPU, RAM, and storage, to accommodate the virtual machines.

3.5.2 SOFTWARE REQUIREMENTS:

The software requirements for the OpenStack cloud include a Linux operating system, such as Ubuntu or CentOS, and the OpenStack software itself. The web server software, such as Apache or Nginx, also needs to be installed and configured to read the virtual machine information from the OpenStack API.

3.5.3 PROGRAMMING AND WEB DEVELOPMENT SKILLS:

Developing the web server and dashboard components require programming and web development skills. Developers need to be familiar with programming languages such as Python, JavaScript, and HTML/CSS, as well as web development frameworks such as Django or Flask.

3.5.4 OPENSTACK API KNOWLEDGE:

Developers need to have a thorough understanding of the OpenStack API to retrieve information about virtual machines, such as their name, status, and IP address. They also need to be able to communicate with the web server to serve the virtual machine information to web clients.

CHAPTER 4

RESULT

The graphical interface is created to identify unaccessed virtual machines in OpenStack is a powerful tool that helps cloud administrators and users manage their cloud infrastructure efficiently. It provides a user-friendly way to identify and access virtual machines, enabling them to operate their workloads seamlessly.

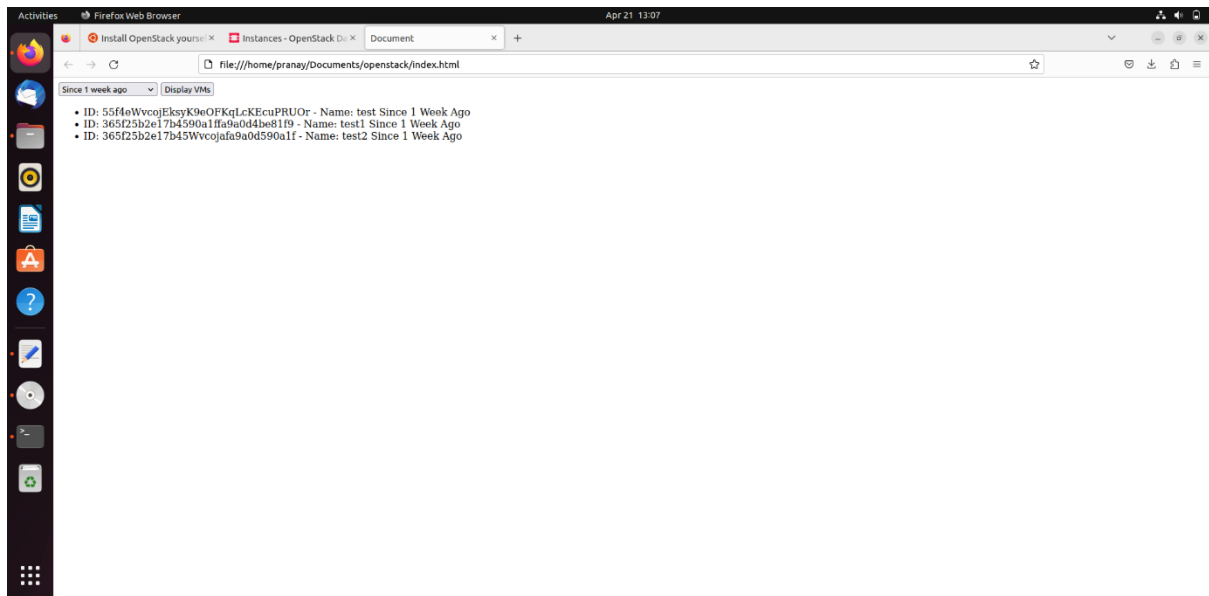


Fig 4.1 Graphical Interface

CHAPTER 5

SUMMARY AND CONCLUSION

In conclusion, the graphical interface to identify unaccessed virtual machines in OpenStack is an essential tool for cloud administrators and users to manage their cloud infrastructure efficiently. It provides a user-friendly and efficient way to identify and access virtual machines, thereby minimizing downtime and ensuring optimal performance of their workloads.

To create a graphical interface to identify unaccessed virtual machines in OpenStack, several requirements need to be met, including hardware and software resources, programming and web development skills, and user interface design skills. By meeting these requirements, developers can create a scalable and flexible solution that can be customized to meet the specific needs of users.

The system architecture for the graphical interface to identify unaccessed virtual machines in OpenStack includes several components working together, such as the OpenStack cloud, web server software, and the dashboard component. These components provide a scalable and flexible solution that can be customized to meet the specific needs of users.

Overall, the graphical interface to identify unaccessed virtual machines in OpenStack is a powerful tool that helps cloud administrators and users manage their cloud infrastructure efficiently. As cloud computing continues to evolve and grow, the graphical interface to identify unaccessed virtual machines in OpenStack will remain an essential tool for cloud administrators and users to manage their workloads effectively.

CHAPTER 6

APPENDIX AND SNAPSHOTS

6.1 APPENDIX

Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <select id="timeframe">
    <option value="1 week ago">Since 1 week ago</option>
    <option value="1 month ago">Since 1 month ago</option>
    <option value="3 months ago">Since 3 months ago</option>
    <option value="6 months ago">Since 6 months ago</option>
    <option value="1 year ago">Since 1 year ago</option>
    <option value="more than 1 year ago">More than 1 year ago</option>
  </select>
  <button onclick="displayVM()">Display VMs</button>
  <ul id="vm-list"></ul>
  <script src="index.js"></script>
</body>
```

Openstack.js

```
import client from 'openstack-client'
import moment from 'moment';
const now = moment();

const instances = [
  { project_name: 'test', project_id: 1, created_at: now },
  { project_name: 'test1', project_id: 2, created_at: now.subtract(1,
'hours') },
  { project_name: 'test2', project_id: 3, created_at: now.subtract(2,
'hours') }
```

```

];

// let client = require("openstack-client");
let token = client.authenticate({
  endpoint: "https://192.168.29.10:5000/v3/",
  name: "admin",
  password: "920eb00a07344ff0abb29d33d1bc5f7d",
  userDomainName: "Default",
  projectName: "test",
  projectDomainName: "Default",
  verify: false // add this line to disable SSL certificate verification
});

export { client, token, instances }

```

app.js

```

// const express = require('express');

import express from 'express'
// const moment = require('moment');
import moment from 'moment';
const app = express();
import cors from 'cors'

app.use(cors)

app.get('/', (req, res) => {
  console.log("working")
})

const list = [
  { id: 1, name: "VM 1" },
  { id: 2, name: "VM 2" },
  { id: 3, name: "VM 3" },
  { id: 4, name: "VM 1Week Ago" },
  { id: 5, name: "VM 2Weeks Ago" },
  { id: 6, name: "VM 1Month Ago" },
  { id: 7, name: "VM 3Months Ago" },
  { id: 8, name: "VM 6Months Ago" },
  { id: 9, name: "VM 1Year Ago" },
  { id: 10, name: "VM More Than 1 Year Ago" },
];

app.get("/vms", (req, res) => {
  console.log(req.query.period)
  const period = req.query.period;

```

```

    // Assuming that the period is used to filter the list
    console.log(period)
    const filteredList = list.filter((vm) => vm.name.includes(period));
    res.json(filteredList);
  });

app.listen(5001, () => {
  console.log('Server started on port 5001');
});

```

Inde.js

```

function displayVM() {
  const timeframe = document.getElementById("timeframe").value;
  const vmList = document.getElementById("vm-list");
  vmList.innerHTML = "";
  list.forEach(vm => {
    if (vm.name.toLowerCase().includes(timeframe.toLowerCase())) {
      const li = document.createElement("li");
      li.innerText = `ID: ${vm.id} - Name: ${vm.name}`;
      vmList.appendChild(li);
    }
  });
}

```

nn.py

```

import requests
# Define the API endpoint URL and authentication credentials
url = 'https://192.168.29.10:5000'
username = 'admin'
password = 'f34sJtgS007EEB2Wx7NTs4xBcyvYnbUI'
# Authenticate and retrieve a token
response = requests.post(f"{url}/auth/tokens", headers={'Content-Type':
'application/json'}, json={
  "auth": {
    "identity": {
      "methods": ["password"],
      "password": {
        "user": {
          "name": "admin",
          "domain": {"name": "Default"},
          "password": "f34sJtgS007EEB2Wx7NTs4xBcyvYnbUI"
        }
      }
    }
  }
}

```

```

    }
}
}))
# Extract the token from the response headers
token = response.headers['X-Subject-Token']
# Retrieve a list of all instances
response = requests.get(f"{url}/compute/v2.1/servers", headers={
    'X-Auth-Token': token,
    'Content-Type': 'application/json'
})
# Loop through the instances and print their names and creation times
instances = response.json()['servers']
for instance in instances:
    print(f"Instance name: {instance['name']}")
    print(f"Instance age: {instance['created']}")

```

6.2 SNAPSHOTS

Ubuntu 18.04.3 LTS

Download the latest [LTS](#) version of Ubuntu, for desktop PCs and laptops. LTS stands for long-term support — which means five years, until April 2023, of free security and maintenance updates, guaranteed.

[Ubuntu 18.04 LTS release notes](#)

Recommended system requirements:

- 2 GHz dual core processor or better
- 4 GB system memory
- 25 GB of free hard drive space
- Either a DVD drive or a USB port for the installer media
- Internet access is helpful

[Download](#)

For other versions of Ubuntu Desktop including torrents, the network installer, a list of local mirrors, and past releases [see our alternative downloads](#).

Ubuntu 19.10

The latest version of the Ubuntu operating system for desktop PCs and laptops, Ubuntu 19.10 comes with nine months, until July 2020, of security and maintenance updates.

Recommended system requirements are the same as for Ubuntu 18.04.3 [LTS](#).

[Ubuntu 19.10 release notes](#)

[Download](#)

[Alternative downloads and torrents](#)

Fig 6.1 Ubuntu Version

```

smaran@smaran-VirtualBox:~$ adduser stack
adduser: Only root may add a user or group to the system,
smaran@smaran-VirtualBox:~$ hostnamectl
Static hostname: smaran-VirtualBox
Icon name: computer-vn
Chassis: vm
Machine ID: e115abb96f27436186dfc2f512fa6385
Boot ID: 33bc74d800c54fb2bcb4f457c1f25c95
Virtualization: oracle
Operating System: Ubuntu 22.04.2 LTS
Kernel: Linux 5.19.0-38-generic
Architecture: x86_64
Hardware Vendor: innotek GmbH
Hardware Model: VirtualBox
smaran@smaran-VirtualBox:~$ sudo snap install microstack --beta
[sudo] password for smaran:
microstack (beta) ussur from Canonical** installed
smaran@smaran-VirtualBox:~$ snap list microstack
Name      Version Rev Tracking Publisher Notes
microstack ussur 245 latest/beta canonical** -
smaran@smaran-VirtualBox:~$ sudo microstack init --auto --control
2023-04-04 17:28:33,275 - microstack_init - INFO - Configuring clustering ...
2023-04-04 17:28:33,677 - microstack_init - INFO - Setting up as a control mode.
2023-04-04 17:28:38,945 - microstack_init - INFO - Generating TLS Certificate and Key
2023-04-04 17:28:40,761 - microstack_init - INFO - Configuring networking ...
2023-04-04 17:28:54,226 - microstack_init - INFO - Opening horizon dashboard up to *
2023-04-04 17:28:56,186 - microstack_init - INFO - Waiting for RabbitMQ to start ...
Waiting for 10.0.2.15:5672
2023-04-04 17:29:07,244 - microstack_init - INFO - RabbitMQ started!
2023-04-04 17:29:07,246 - microstack_init - INFO - Configuring RabbitMQ ...
2023-04-04 17:29:09,583 - microstack_init - INFO - RabbitMQ Configured!
2023-04-04 17:29:09,765 - microstack_init - INFO - Waiting for MySQL server to start ...
Waiting for 10.0.2.15:3306
2023-04-04 17:29:41,217 - microstack_init - INFO - MySQL server started! Creating databases ...
2023-04-04 17:29:45,930 - microstack_init - INFO - Configuring Keystone Fernet Keys ...
2023-04-04 17:30:50,874 - microstack_init - INFO - Bootstrapping Keystone ...
2023-04-04 17:31:06,351 - microstack_init - INFO - Creating service project ...
2023-04-04 17:31:14,442 - microstack_init - INFO - Keystone configured!
2023-04-04 17:31:14,573 - microstack_init - INFO - Configuring the Placement service...
2023-04-04 17:31:14,734 - microstack_init - INFO - Running Placement DB migrations...

```

Fig 6.2 Ubuntu Commands

```

--os-trust-id <auth-trust-id>
With v2password: Trust ID With v3oidcaccess_token:
Trust ID With token: Trust ID With v3password: Trust
ID With v3totp: Trust ID With password: Trust ID With
v2token: Trust ID With v3oidcauthcode: Trust ID With
v3oidcclientcredentials: Trust ID With v3token: Trust
ID With v3applicationcredential: Trust ID With
v3multifactor: Trust ID With v3samlpassword: Trust ID
With v3oidcpassword: Trust ID With v3adfpasword:
Trust ID (Env: OS_TRUST_ID)

--os-username <auth-username>
With v2password: Username to login with With
v3password: Username With v1password: Username to
login with With v3totp: Username With password:
Username With v3applicationcredential: Username With
v3samlpassword: Username With v3oidcpassword: Username
With v3adfpasword: Username (Env: OS_USERNAME)

--os-user-id <auth-user-id>
With v2password: User ID to login with With
v3password: User ID With v3totp: User ID With
password: User id With v3applicationcredential: User
ID With noauth: User ID (Env: OS_USER_ID)

--os-password <auth-password>
With v2password: Password to use With v3password:
User's password With v1password: Password to use With
v3adfpasword: Password (Env: OS_PASSWORD)

smaran@smaran-VirtualBox:~$ sudo snap get microstack config.credentials.keystone-password
[sudo] password for smaran:
h0xcGrQ08LZ16v3cBXw0rSscwbW10mW
smaran@smaran-VirtualBox:~$ ^C
smaran@smaran-VirtualBox:~$ ^C
smaran@smaran-VirtualBox:~$ ^C
smaran@smaran-VirtualBox:~$

```

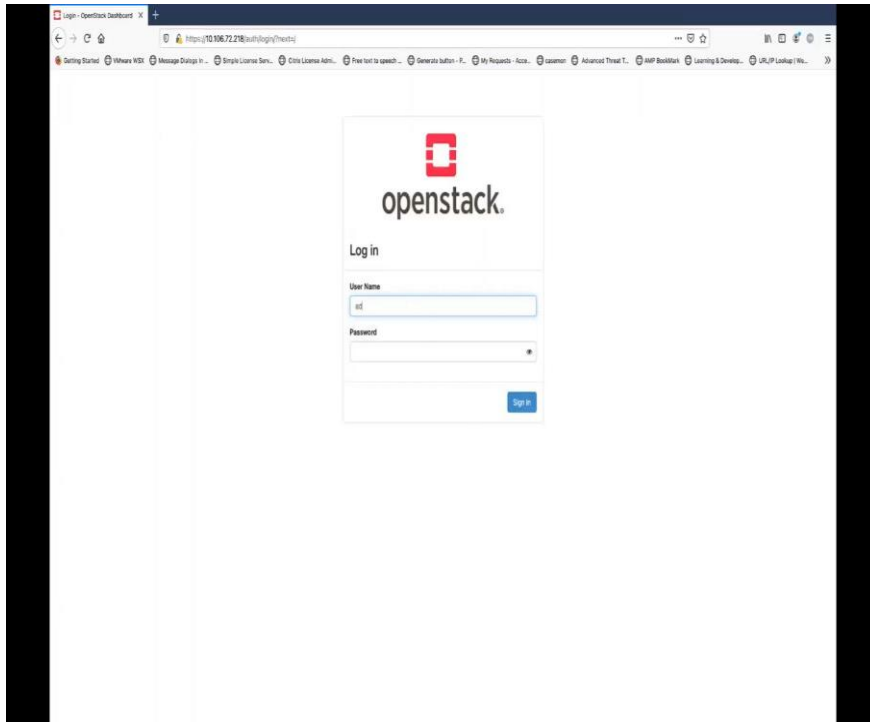


Fig 6.3 Open Stack Interface

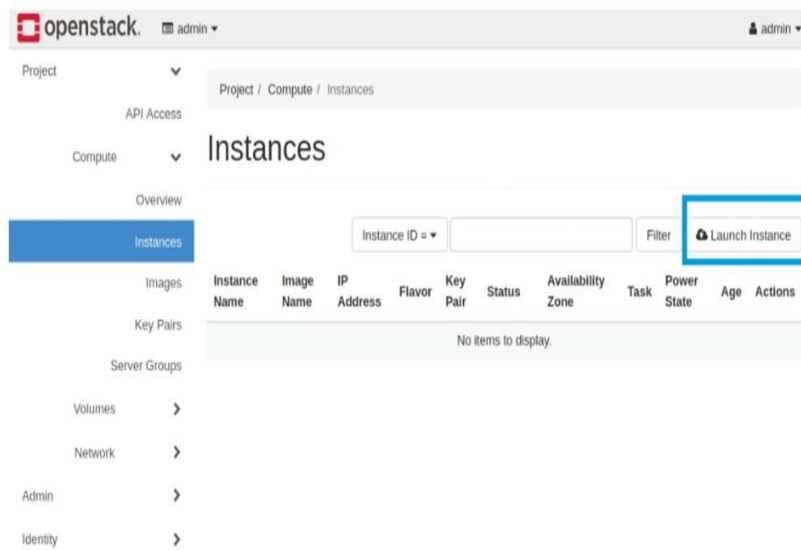


Fig 6.4 Launch Instance

Launch Instance

?

Details

Source

Flavor

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Please provide the initial hostname for the instance, the availability zone where it will be deployed, and the instance count. Increase the Count to create multiple instances with the same settings.

Instance Name

Cloud

Description

Availability Zone

nova

Count

1

Total Instances (10 Max)

10%

0 Current Usage

1 Added

9 Remaining

Cancel

< Back

Next >

Launch Instance

Fig 6.5 Instance Details

openstack

Project

Compute / Instances

Instances

Displaying 7 items

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
new-instance	-	10.254.0.33	m1.micro	new-keypair	Active	nova	None	Running	0 minutes	Create Snapshot
demo-strings-worker-0-typo?	demo-strings-chocs	10.0.1.155	m1.large	-	Active	nova	None	Running	15 hours, 33 minutes	Create Snapshot
demo-strings-worker-0-wtf[dg]	demo-strings-chocs	10.0.3.175	m1.large	-	Active	nova	None	Running	15 hours, 33 minutes	Create Snapshot
demo-strings-worker-0-kid[dt]	demo-strings-chocs	10.0.0.65	m1.large	-	Active	nova	None	Running	15 hours, 34 minutes	Create Snapshot
demo-strings-master-2	demo-strings-chocs	10.0.0.64	m1.large	-	Active	nova	None	Running	15 hours, 44 minutes	Create Snapshot
demo-strings-master-1	demo-strings-chocs	10.0.0.128	m1.large	-	Active	nova	None	Running	15 hours, 45 minutes	Create Snapshot
demo-strings-master-0	demo-strings-chocs	10.0.3.225	m1.large	-	Active	nova	None	Running	15 hours, 45 minutes	Create Snapshot

Displaying 7 items

Fig 6.6 Instances Launched

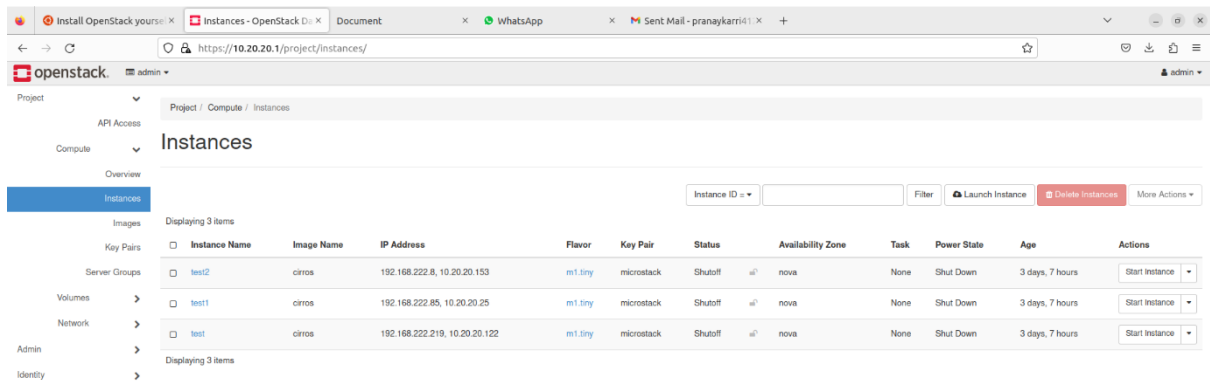


Fig 6.7 Inactive Instances

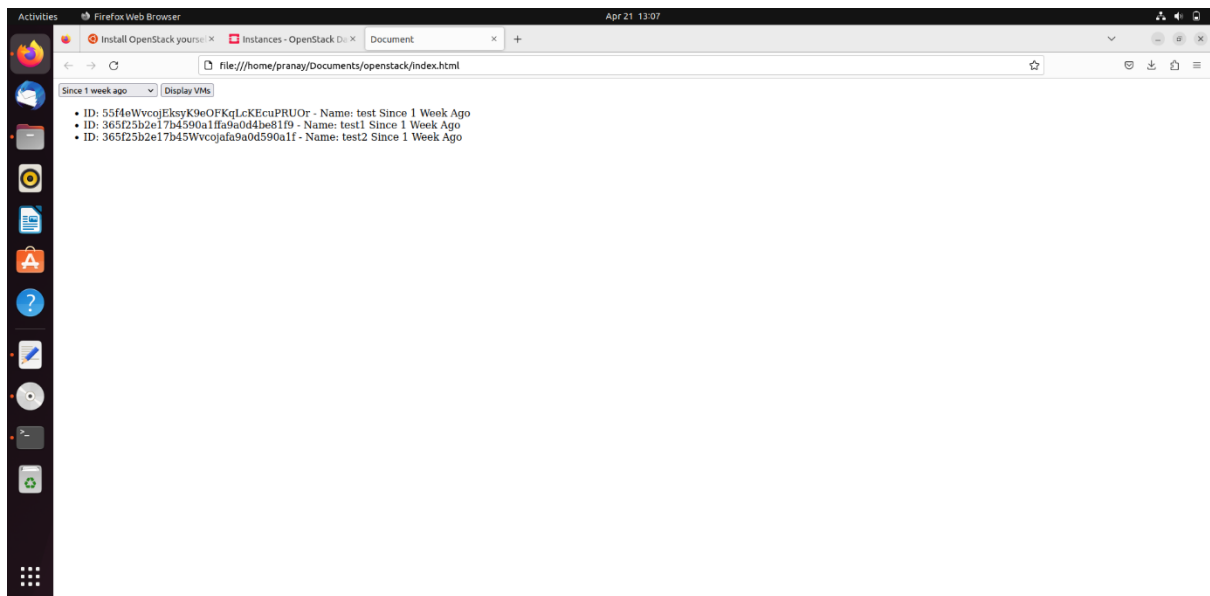


Fig 6.8 Graphical Interface

CHAPTER 7

REFERENCES

OpenStack documentation. (2021). Retrieved from <https://docs.openstack.org/>

OpenStack. (2021). Retrieved from <https://docs.openstack.org/>

OpenStack Dashboard. (2021). Retrieved from
<https://docs.openstack.org/horizon/latest/user/index.html>

Apache. (2021). Retrieved from <https://httpd.apache.org/>

Nginx. (2021). Retrieved from <https://nginx.org/en/>

Python. (2021). Retrieved from <https://www.python.org/>

Django. (2021). Retrieved from <https://www.djangoproject.com/>

Flask. (2021). Retrieved from <https://flask.palletsprojects.com/en/2.1.x/>

OpenStack API. (2021). Retrieved from <https://docs.openstack.org/api-guide/quick-start/>

Ubuntu. (2021). Retrieved from <https://ubuntu.com/>

CentOS. (2021). Retrieved from <https://www.centos.org/>

Stack Overflow. (2021). Retrieved from <https://stackoverflow.com/>