Title of the Project

[Times new Roman: Bold face, Font size: 24]

Technique For Load Balancing with Kubernetes and Python

A Project report submitted in partial fulfilment

of the requirements for the degree of B. Tech in Electrical Engineering

[Times new Roman: Italic, Font size: 12]

By

Name of the Students (Roll No.)

Abhro Roy (11701619034)

Deepanjan Mondal (11701619025)

Anish Chakraborty (11701619015)

Under the supervision of --

Dr. Shilpi Bhattacharya (H.O.D)

Department of Electrical Engineering

श्रमम् बिना न किमपि साध्यम्

Department of Electrical Engineering

RCC INSTITUTE OF INFORMATION TECHNOLOGY

CANAL SOUTH ROAD, BELIAGHATA, KOLKATA – 700015, WEST BENGAL,Maulana Abul Kalam Azad University of Technology (MAKAUT)© **2023**

ACKNOWLEDGEMENT

It is my great fortune that I have got opportunity to carry out this project work under the supervision of **(Name of the Project Guide/Supervisor)** in the Department of Electrical Engineering, RCC Institute of Information Technology (RCCIIT), Canal South Road, Beliaghata, Kolkata-700015, affiliated to Maulana Abul Kalam Azad University of Technology (MAKAUT), West Bengal, India. I express my sincere thanks and deepest sense of gratitude to my guide for his constant support, unparalleled guidance and limitless encouragement.

I wish to convey my gratitude to Prof. (Dr.) Shilpi Bhattacharya, HOD, Department of Electrical Engineering, RCCIIT and to the authority of RCCIIT for providing all kinds of infrastructural facility towards the research work.

I would also like to convey my gratitude to all the faculty members and staffs of the Department of Electrical Engineering, RCCIIT for their whole hearted cooperation to make this work turn into reality.

Signature of the Students

Place:

Date:

Department of Electrical Engineering
RCC INSTITUTE OF INFORMATION TECHNOLOGY
CANAL SOUTH ROAD, BELIAGHATA, KOLKATA – 700015, WEST BENGAL

CERTIFICATE

To whom it may concern

This is to certify that the project work entitled (**write the Title of your Project here)** is the bona fide work carried out by **(write your Name and Roll number here)** , a student of B.Tech in the Dept. of Electrical Engineering, RCC Institute of Information Technology (RCCIIT), Canal South Road, Beliaghata, Kolkata-700015, affiliated to Maulana Abul Kalam Azad University of Technology (MAKAUT), West Bengal, India, during the academic year 2021-22, in partial fulfillment of the requirements for the degree of Bachelor of Technology in Electrical Engineering and this project has not submitted previously for the award of any other degree, diploma and fellowship.

______________________ ______________________

Signature of the Guide

Name:

Designation

Signature of the HOD, EE

Name:

Designation

____

Signature of the External Examiner

Name:

Designation:

: Table of Contents:

	Page no.
List of Nomenclature (if any)	*i*
List of Acronyms (if any)	*ii*
List of Tables (if any)	*iii*
List of Figures (if any)	*iv*
Abstract	*v*
1. Introduction	**1**
2. Theory	**3-5**
2.1 Working Principle	**4**
2.2 Circuit Diagram	**5**
2.3	**6**
3. Mathematical Model	**7-10**
3.1......................	**.**
3.3..................	**.**
4. Hardware Model	**16-18**
5. Algorithm and Software Program	**.**
6. Observations and Results	**.**
7. Conclusions	**.**
Appendix A	**25-30**
A.1 Specifications of the Hardware components	**25**
A.2	**27**
References	**40**

[It is standard practice, not to use serial number for ***Nomenclature, Acronyms, List of Tables, List of Figures, Abstract, Reference.*** *For* ***Appendix*** *follow serial no. as* ***A.1, A.2*** *or* ***B.1, B.2*** *....etc.]*

Technique For Load Balancing with Kubernetes and Python

Abhro Roy, Deepanjan Mondal, Anish Chakraborty, Shilpi Bhattacharya
Electrical Engineering
RCC Institute of Information Technology, Kolkata
e-Mail: abirabhroroy@gmail.com

ABSTRACT

Often we find servers across organizations and institutes having servers which have minimal to no load on normal days but peak during certain times of the year and are slow and overload rendering the site to little use. Kubernetes enables versatile deployment of applications which are auto-scaled, auto-maintained, and auto-managed depending on variety of parameters including peak load.

Keywords: **Kubernetes, Python, Docker, Linux**

1. INTRODUCTION

Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

The name Kubernetes originates from Greek, meaning helmsman or pilot. K8s as an abbreviation results from counting the eight letters between the "K" and the "s". Google open-sourced the Kubernetes project in 2014. Kubernetes combines over 15 years of Google's experience running production workloads at scale with best-of-breed ideas and practices from the community.[1]

2. LITERATURE SURVEY

The cost of deploying a Kubernetes cluster is close to nothing, since it is an open source code available to anyone anywhere. However, one does need a relatively mediocre bare-metal system to host this architecture without hitches.[2]

3. TECHNOLOGIES AND METHODOLOGIES USED IN THIS PROJECT

The resources used to deploy this entire architecture are less in number, but implemented in depth. They are as follows:

A. KUBERNETES

The actual platform which hosts our code, application and all the bells and whistles required to seamlessly expose the application as a service outside Kubernetes.

B. PYTHON

The application is written in this language along with Python-Flask server to host the application inside the container.

C. DOCKER

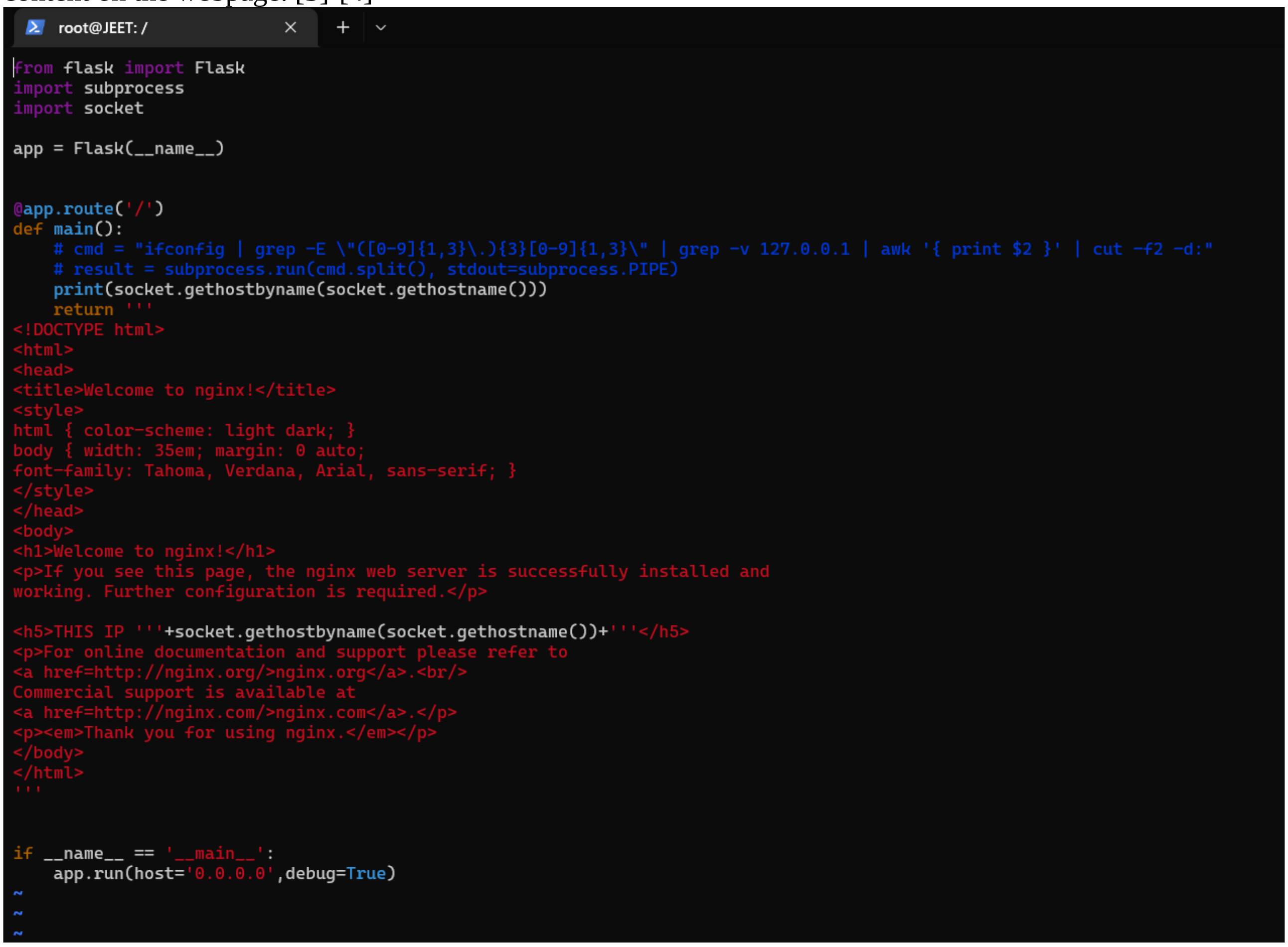
Docker is a containerization platform used to deploy containers, which are nothing but a shell of a base image of an OS, on which one can install all the resources one installs on a normal bare-metal OS. Containers are surprisingly light and fast and are used during mass deployment of similar applications for milking out the most of the resources available at hand.

4. TECHNIQUES USED FOR REAL-LIFE IMPLEMENTATION AND WORKING

A. Writing the python script

Write a python code with Flask web server to print a basic html page with ip of the host it is running on and push the image to DockerHub.

Below is the python code implemented. A rather simple code stating our application be broadcast on all IPs (ref. Last line ‘host=0.0.0.0’). On going to our default route which is ‘/’ in ‘webgen.americaniche.com/’, the below code is executed. The code gets the host IP the application is running on and displays it has HTML content on the webpage. [3]-[4]



```
root@JEET: /

from flask import Flask
import subprocess
import socket

app = Flask(__name__)


@app.route('/')
def main():
    # cmd = "ifconfig | grep -E \"([0-9]{1,3}\.){3}[0-9]{1,3}\" | grep -v 127.0.0.1 | awk '{ print $2 }' | cut -f2 -d:"
    # result = subprocess.run(cmd.split(), stdout=subprocess.PIPE)
    print(socket.gethostbyname(socket.gethostname()))
    return '''
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<h5>THIS IP '''+socket.gethostbyname(socket.gethostname())+'''</h5>
<p>For online documentation and support please refer to
<a href=http://nginx.org/>nginx.org</a>.<br/>
Commercial support is available at
<a href=http://nginx.com/>nginx.com</a>.</p>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
'''


if __name__ == '__main__':
    app.run(host='0.0.0.0',debug=True)
~
~
~
```

B. Creating the custom image with Dockerfile

We first build a custom Docker image with base OS ‘ubuntu:latest’ and we setup the image to do the following:
Expose port 5000 outside container where our server will be running.
Copy our Python code from host os to image os.
Install python and its dependencies.
Create entry-point script to run our python code whenever the image is deployed.[5]

```
from ubuntu:latest

EXPOSE 5000

COPY /app.py /

RUN apt update

RUN apt install -y python3 && \
    apt install -y python3-pip && \
    pip3 install Flask && \
    apt install -y net-tools && \
    apt install -y vim && \
    apt install -y cron

#RUN echo "until python3 /app.py; do  echo "Server 'flask' crashed with exit code $?.  Respawning.." >&2;  sleep 1; done" >> /keepalive.sh
#RUN chmod 644 /keepalive.sh

#RUN touch /var/log/cron.log

#RUN echo '3 * * * * kill -9 $(ps ax|grep "app.py"|head -1|awk "{print $1}"' >> /etc/cron.d/runka
#RUN echo '* * * * * python3 /app.py '>> /etc/cron.d/runka
#RUN chmod 644 /etc/cron.d/runka

# Apply cron job
#RUN crontab /etc/cron.d/runka
ENTRYPOINT ["python3", "/app.py"]
~
~
```

C. Setting up the Kubernetes cluster

Using Vagrant and Ansible we set up a 5 node Kubernetes cluster of which one is the master node (responsible for managing worker nodes and for deploying apps and services) and four are the worker/slave nodes.

The Vagrantfile is responsible for spinning up the required number of virtual machines and naming them, provisioning resources, disk space, assign IP etc. , and the ansible playbook, which is embedded within the Vagrantfile, and is responsible for installing required applications, dependencies, features etc. The first image below is the Vagrantfile and the second image is the trimmed ansible playbook for the VM that will act as our master node in our Kubernetes cluster.[6]

```
Vagrant.configure("2") do |config|
    config.ssh.insert_key = false

    config.vm.provider "virtualbox" do |v|
        v.memory = 4096
        v.cpus = 2
    end

    config.vm.define "k8s-master" do |master|
        master.vm.box = IMAGE_NAME
        master.vm.network "private_network", ip: "192.168.56.10"
        master.vm.hostname = "k8s-master"
        master.vm.provision "ansible" do |ansible|
            ansible.playbook = "master-playbook.yaml"
            ansible.extra_vars = {
                node_ip: "192.168.56.10",
            }
        end
    end

    (1..N).each do |i|
        config.vm.define "node-#{i}" do |node|
            node.vm.box = IMAGE_NAME
            node.vm.network "private_network", ip: "192.168.56.#{i + 10}"
            node.vm.hostname = "node-#{i}"
            node.vm.provision "ansible" do |ansible|
                ansible.playbook = "node-playbook.yaml"
                ansible.extra_vars = {
                    node_ip: "192.168.56.#{i + 10}",
                }
            end
```

```
[abhro@backendserver vaganskube]$ cat master-playbook.yaml
---
- hosts: all
  become: true
  tasks:
  - name: Install packages that allow apt to be used over HTTPS
    apt:
      name: "{{ packages }}"
      state: present
      update_cache: yes
    vars:
      packages:
      - apt-transport-https
      - ca-certificates
      - curl
      - gnupg-agent
      - software-properties-common

  - name: Add an apt signing key for Docker
    apt_key:
      url: https://download.docker.com/linux/ubuntu/gpg
      state: present

  - name: Add apt repository for stable version
    apt_repository:
      repo: deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial stable
      state: present

  - name: Install docker and its dependecies
    apt:
      name: "{{ packages }}"
      state: present
      update_cache: yes
    vars:
      packages:
      - docker-ce
      - docker-ce-cli
      - containerd.io
    notify:
      - docker status

  - name: Add vagrant user to docker group
    user:
      name: vagrant
      group: docker

  - name: Remove swapfile from /etc/fstab
    mount:
      name: "{{ item }}"
```

D. Assigning Node and Pods

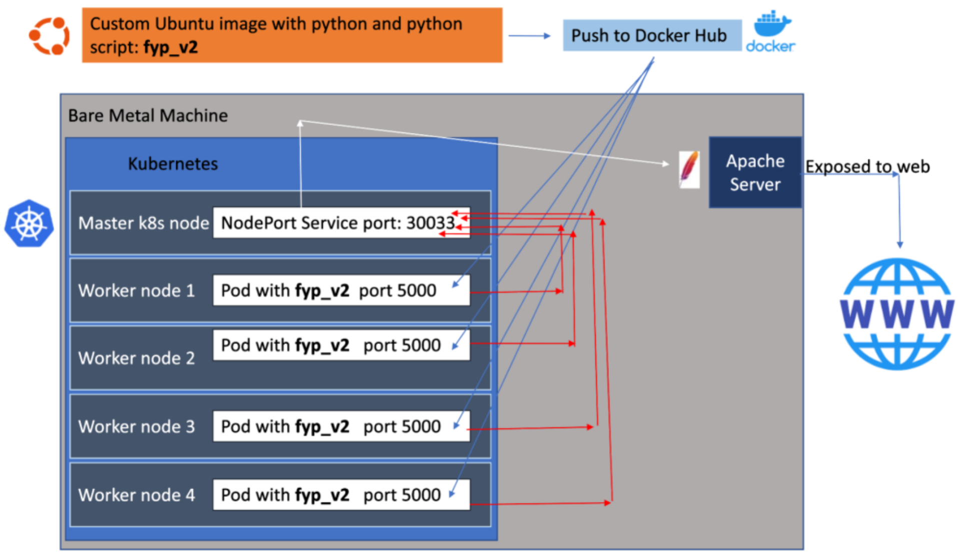
We pull our custom image from DockerHub and setup a pod(docker container). Kubernetes assigns an IP to the pod and a node where the pod will be running. We can directly access this IP from inside Kubernetes to check whether our pod is up.
Upon successful deployment, a NodePort type service is created which is bound to our pod and is exposed outside Kubernetes using a port opened from the node our pod is hosted on.
We then create a DaemonSet which clones our Pod hosting our python script almost entirely and deploys one copy on each node, hence making it highly available.[7]-[9]

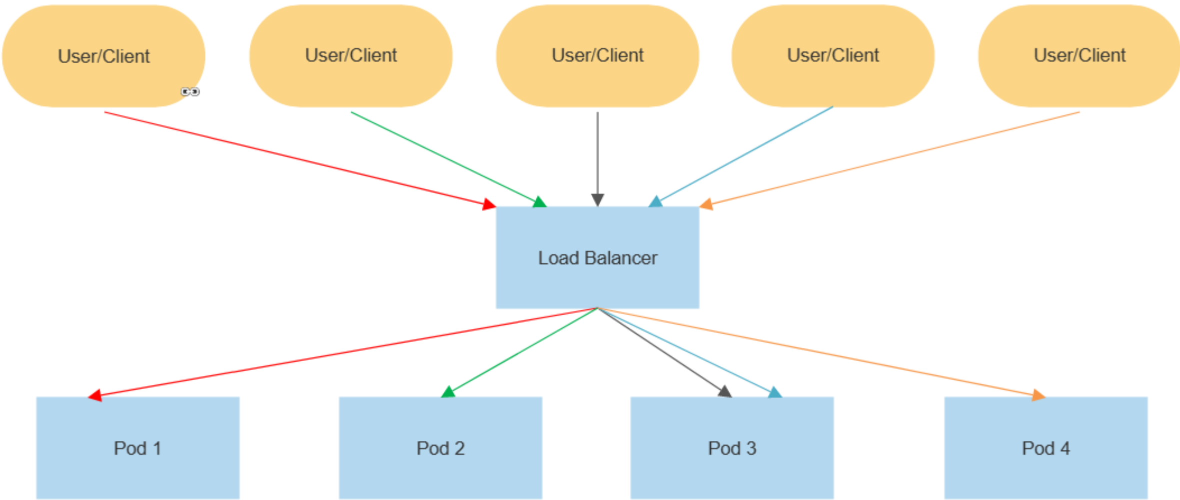
E. Exposing outside Kubernetes

Using an Apache server virtual host with reverse proxy (to prevent leaking of port) and a pre-configured AWS Route53 domain allows us to access this Pod from anywhere in the world, in this case http://webgen.americaniche.com. The virtual hosting configuration is depicted below.
‘<VirtualHost *:80>’ states that we are creating a virtual host for every request coming on the port 80.
Our FQDN is ‘webgen.americaniche.com’ set by the ‘ServerName’ parameter. The Proxypass configuration is the most important in this part since it masks our port and prevents exposing where our apps are running, and minimize the possibility of being compromised/attacked.[10]

```
<VirtualHost *:80>
    Servername webgen.americaniche.com
Proxypass / http://webgen.americaniche.com:30034/
ProxypassReverse / http://webgen.americaniche.com:30034/
</VirtualHost>
```

5. VISUAL REPRESENTATION





6. FUTURE WORK

Our scope to this project ends here and goes on to show the capabilities offered, however, it can be extended to include sticky sessions to bind to users to save login data or session data, other enhancements and whatnot. For this we would need to find a requirement to satisfy, and that supports this development.[6]-[9]

7. CONCLUSION

Thus with this implementation we prove the ease of usability, low cost and setup of a deployment platform that can host entire infrastructures with the proper resources anywhere in the world, for a fraction of the cost required by mainstream cloud platforms like AWS and Azure.

REFERENCES

[1] D. Ashley, "Using Flask and Jinja," Foundation Dynamic Web Pages with Python, pp. 159–181, 2020, doi: 10.1007/978-1-4842-6339-6_5.

[2] D. Ashley, "Introduction to Web Servers," Foundation Dynamic Web Pages with Python, pp. 1–27, 2020, doi: 10.1007/978-1-4842-6339-6_1.

[3] O. Yilmaz, "Introduction," Extending Kubernetes, pp. 1–19, 2021, doi: 10.1007/978-1-4842-7095-0_1.

[4] M. Lukša, "Kubernetes erweitern," Kubernetes in Action, pp. 553–578, Jul. 2018, doi: 10.3139/9783446456020.018

[5] Q. Li and B. Moon, "Distributed cooperative Apache web server," Proceedings of the tenth international conference on World Wide Web

[6] M. Shahinpoor, Y. Bar-Cohen, T. Xue, J.O. Simpson and J. Smith, "Ionic Polymer-Metal Composites (IPMCs) as Biomimetic Sensors, Actuators and Artificial Muscles: A Review", Proceedings of SPIE's 5th Annual International Symposium on Smart Structures and Materials, 1-5 March, 1998, San Diego, CA DOI: 10.1088/0964-1726/7/6/001

[7] Zheng Chen, Yantao Shen, Jason Malinak, Ning Xi, Xiaobo Tan,"Hybrid IPMC/PVDF Structure for Simultaneous Actuation and Sensing", Proceedings of SPIE Vol. 6168, Smart Structures and Materials, pp. 61681L 1 – 61681L9, 2006. DOI: 10.1109/ICIEV.2014.6850840

[8] R.K. Jain, S. Datta and S. Majumder, "Design and Control of an EMG Driven IPMC Based Artificial Muscle Finger", Chapter 15, of Computational Intelligence in Electromyography Analysis: A Perspective on Current Applications and Future Challenges" ed. by Ganesh R. Naik, pp. 362 – 390, 2012. DOI: https://doi.org/10.1007/978-3-319-05582-4_82

[9] S. Bhat, "Introduction to Containerization," Practical Docker with Python, pp. 1–8, 2018, doi: 10.1007/978-1-4842-3784-7_1.

[10] K. Jangla, "Docker Compose," Accelerating Development Velocity Using Docker, pp. 77–98, 2018, doi: 10.1007/978-1-4842-3936-0_6.