Private Cloud Containerization using Raspberry Pi Network

Sahana M P[1], Sonali Joyce Lobo[2]

Asst.professor
ISE, DSCE Bengaluru, India

Abstract

The existing FOSS IAAS cloud systems target enterprise as a primary user, students and developers usually cannot afford to maintain their own cloud and technical knowledge required to maintain is still very high. Raspberry Pi and ARM based devices is a series of little single board computers which is very affordable and caters to all the requirements. Raspberry Pi is a accomplished little device that enables people of all ages to explore cloud computing. It's capable of doing everything we'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheet, word-processing, and playing games. Students who wants to learn cloud should be able to span cloud servers on demand for free in their premise. This would definitely make the life of a student much better by building pocket cloud on demand easily. Raspberry Pi devices are officially supported by Docker and Kubernetes which can be used to create and orchestrate cloud containers. ARM based devices can also be used to spawn Virtual Machines using cloud stack. In this paper, Raspberry Pi and affordable device is configured with Docker and Kubernetes with a supporting Operating System to spawn on demand containers and Virtual Machines.

Keywords-*Docker; Raspberry Pi; Cloud Computing; Kubernetes; Containers;*

I. INTRODUCTION

Raspberry Pi: The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to assist the teaching of basic computer science in schools and in developing kingdom. Containers: Using containers, everything required to make a piece of software run is packaged into isolated containers. Unlike VMs, containers do not bundle a full operating system - only libraries and settings required to make the software work are needed. This makes for efficient, lightweight, self-contained systems and guarantees that software will always run the same, regardless of where it's establish.

Docker: Docker is the world's leading software container dais. Developers use Docker to remove "works on my machine" problems when collaborating on code with co-workers. Operators use Docker to run and control apps side-by-side in isolated containers to get better compute thickness.

Kubernetes: Kubernetes is an open-source platform for automating deployment, scaling, and operations of application containers across collection of hosts, providing container-centric configuration.

With Kubernetes consumer demand can be met and responded quickly and effectively. Applications deployments happen rapidly and predictably. Applications can be scaled on the fly. New features can be seamlessly rolled out. Use of hardware can be optimized by using only the resources needed.

Major Components of the system:

1) Master Components: Essentially, master is the brain of cluster. Here, there is a core API server, which maintains RESTful web services for querying and defining the desired cluster and workload state. It's important to note that the control pane only accesses the master to initiate alter and not the nodes directly.

Additionally, the master includes the scheduler, which works with the API server to schedule workloads in the form of pods on the actual minion nodes. These pods include the different containers that make up application stacks. By default, the basic Kubernetes scheduler spreads pods across the clusters and uses various nodes for matching pod replicas. Kubernetes also allows specifying necessary resources for each container, so scheduling can be altered by these additional factors.

2) Node (Minion) Components: A node is a worker machine in Kubernetes, previously known as a minion. A node may be a VM or physical machine, depending on the cluster. Each node has the services necessary to run pods and is controlled by the master components. The services on a node include Docker, kubelet and kube-proxy.

Node status describes current status of HostIP, Node Phase and Node Condition. Node is created by cloud providers or from physical or virtual machines. Kubernetes only creates a representation and after creation, it will examine whether the node is valid or not. Node Controller manages Node objects, cluster-wide synchronization (create/delete representation), single node life-cycle management.

Kubernetes Core Architecture:

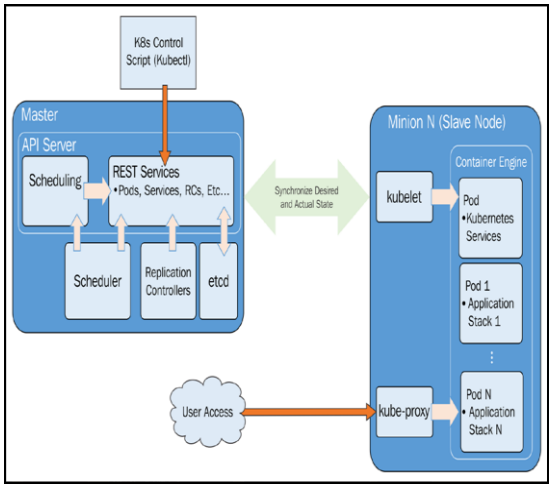

Figure 1: Kubernetes Architecture

Figure 1 shows a general block diagram describing the activities performed by this system with architecture in the forefront the entire architecture has been implemented in four modules which will be seen in high level design and low-level design in later chapters.

II. CURRENT CONCERNS AND SOLUTIONS

A. Concerns :

The existing FOSS IaaS systems target Enterprise as the primary user. This leaves a big hole where small/medium scale companies, universities, students can't install, or maintain a cloud of their own. The amount of technical knowledge required to maintain any of the available cloud is still very high. This tool will fit in this hole where it will give the minimal features of any existing cloud service.

Till now a cloud network with physical access to the servers was not available, only AWS/Digital Ocean catered for building cloud and renting our servers from different parts of the world which would question the security of the network. Developers had to build apps /websites for every platform to deploy them, but now a cross platform container technique called Docker can also be spawned and any app/ images can be made to run on any platform. This would enable developers to build an app for one platform and deploy it on all the platform easily using Docker.

Raspberry Pi is a capable little device that authorize people of all ages to explore cloud computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything we'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

Students who wants to learn cloud should be able to span cloud servers on demand for free in their premise. This would definitely make the life of a student much better by building pocket cloud on demand easily.

Students or Amateur Developers should be able to build their own cloud in their home using raspberry pi network and test how their app is working on cross platform deployed on the server. Developers should be able to deploy their apps on cloud easily without a deployment team, thereby reducing the cost of the project.

B. Scope and Objective of this paper:

Build Virtual Machines and Containers to enable students to understand what it means and the difference between them. To use less expensive hardware for doing the same, minimizing the cost and make it a one-time investment. To help people understand about the "CLOUD" and Providing a way to help teaching and experiencing cloud.

A steady system would require some basic necessary features such as spawning Cloud instances (the basic VM(s) and containers that can be created on a cloud) and assigning elastic IP to the VM to get access from network and use Command line tools and web interface to manage the instances. A profile where User/Groups with quotas can be added or edited and a very simple object store.

C. Solutions and Literature Review:

Zhan Ying, Sun Yong has inscribed in their work about Analyzing and discussing data pooling structure, online data pooling partition container model, structure model of enterprise Cloud Storage System, proposes the design plans of data control and Cloud Storage.

Dimitrios Kelaidonis, Angelos Rouskas, Vera Stavroulaki, Panagiotis Demestichas, Panagiotis Vlacheas has discussed in their paper about Federated architecture combined by distributed Edge Cloud-IoT platforms, that enables the semantic-based service combination and provisioning. The 5G technological features, such as low-latency, zero-losses, high-bitrates are studied and initiate for the empowerment of the federation of the Edge Cloud-Io architectures.

Donggang Cao, Peidong Liu, Wei Cui, Yehong Zhong, and Bo An has described Virtualizing the cluster environment for distributed application frame. Most applications can directly run in the virtual cluster environment without any modification, which is a great advantage. Based on lightweight containers, implement of a real system of ClaaS named Docklet to prove the feasibility of this service model.

Víctor Medel , Omer Rana , José Ángel Bañares, Unai Arronategui has experimented on containers that are rapidly replacing Virtual Machines (VMs) as the compute instance of choice in cloud-

based deployments. The significantly lower overhead of deploying containers (compared to VMs) has often been cited as one reason for this. Analyze the performance of Kubernetes system and develop a Reference net-based model of resource management within this system. Their model is characterized using actual data from a Kubernetes deployment, and can be used as a basis to design scalable applications that make use of Kubernetes.

Sachchidanand Singh, Nirmala Singh has described that Container-based virtualization uses single kernel to run multiple instances on an operating system and virtualization layer runs as an application within the operating system. It is also called operating system virtualization and in this approach, the kernel of operating system runs on the hardware node with different isolated Visitor virtual machines (VMs) called containers.

III. METHODOLOGY

Module 1: Porting of a "libvirt" enabled Operating System to ARM based devices and setting up a static IP for all.

Module 2: Installing Docker containers and managing it with Kubernetes by configuring the nodes to the master

Module 3: Writing utilities (scripts) to automate the process and manage it in a controlled environment

Module 4: Implementing a command line tool and a web interface for the client

A raspberry Pi device has an ARM based architecture and it requires the use of "libvirt" library for virtualization and Containerization support. Only few Operating Systems support this library function and that should be ported to Raspberry Pi device. Since the use of IPs are regular all over the project, a static IP should be set up for each of the Raspberry Pi devices.

Once the OS is installed and static IP is set, Docker should be installed to all the Raspberry Devices for deploying containers. After a cluster of Docker is created by connecting multiple Docker installed Raspberry Pi(s), Kubernetes is installed to manage the cluster. One of the Raspberry Pi devices is made a master node and the rest of them are made pods.

After all the configurations are carried out and the nodes are connected and tested, the installation process is automated by pushing the commands used during the process to install and configure. Separate scripts and techniques are used to automate the process of adding and deleting the containers.

There has to be an easy and effective way of managing the containers and Kubernetes. To achieve that a web GUI is required which acts as a dashboard to manage Kubernetes and all the pods. A CLI tool is also an easy and faster way of managing and using the services of the project.

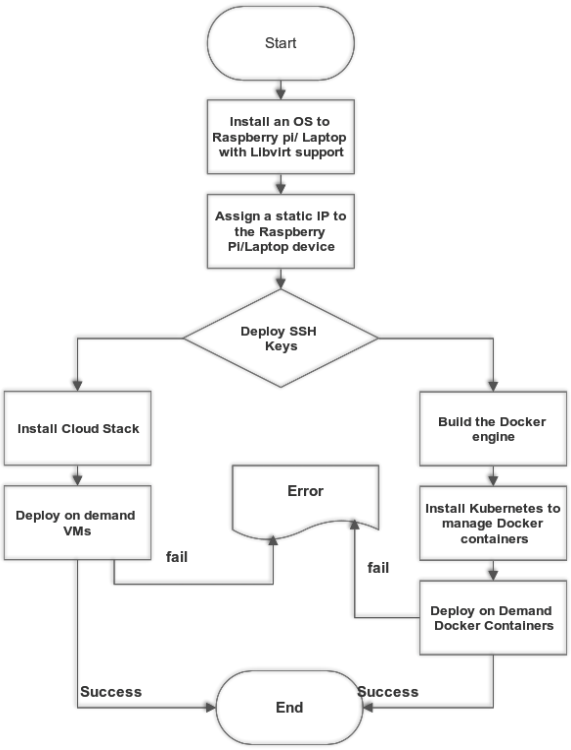

Figure 2: Flowchart of the implementation

According to the flowchart given in figure 2 above, initially an OS is installed on raspberry pi/laptop with libvirt support. A "libvirt" is a type of library which is supported only by some of the Operating Systems and is used for virtualization and containerization. A static IP is assigned to raspberry pi/laptop device and then SSH keys are deployed with which the 1st part of the project i.e. installation of cloud stack can be done remotely and on demand VMs can be spawned. If the VM is spawned then a success message is sent else an error message is sent. The latter part of the setup is installing Docker and building the Docker Engine. After a cluster of Docker engines are setup, a Kubernetes is required to manage these Docker clusters. Kubernetes sends commands to the Docker to deploy on demand Docker Containers with the given container application image. On failure, an error message is displayed else a success message is displayed and the containers are deployed.

The installation and configuration of this dashboard are as below:

$***kubectl get pods --all-namespaces | grep dashboard***

To check if the dashboard is already installed or not.

$ ***kubectl create -f https://git.io/kube-dashboard***

To create the dashboard if it is missing

$ ***kubectl proxy***

Easiest way to access the dashboard

IV. RESULTS AND PROOF OF CONCEPT

Description	Input	Expected	Actual O/P	Result
Test OS	Port a supported OS	Containerization Success	Containerization Success	Pass
Test Network Access	Static IP Configured	SSH Succeeded with the given IP	SSH succeeded with the given IP	Pass
Master Node Availability	Kubernetes command to check master	Live, up and Running	Live, up and Running	Pass
Slave Nodes Availability	Deploy a container	Container Successfully deployed on Nodes	Container Successfully deployed on Nodes	Pass
Test Replica pods	Check the availability of pods	Pods Running	Pods Running	Pass
Dashboard Availability	Check Web Dashboard	Dashboard Running	Dashboard Running	Pass
Operations	Add a Pod	Pod is Added	Pod is Added	Pass

Table 1: Proof of Concept

Figure 3: A Webpage Deployed on One of the Pods Using Kubernetes

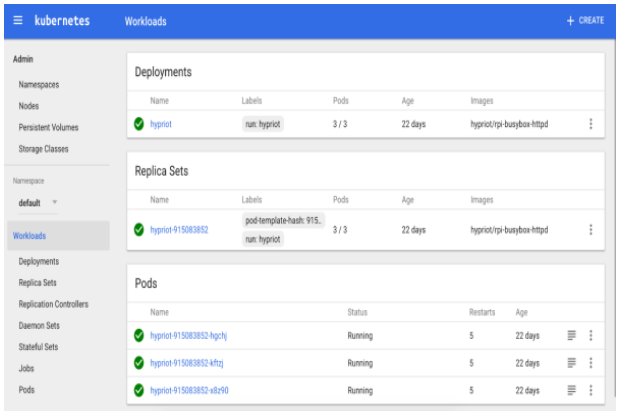

Figure 4: Kubernetes Workload

The Kubernetes dashboard has some advanced features as follows:

1. Shows the Workloads of Kubernetes – Deployments, ReplicaSets,Pods
2. Shows the Memory usage and CPU usage of the setup
3. Shows the number of Pods and its status
4. Allows the creation of new container application
5. User can edit the existing Pods
6. Services and important details are furnished in dashboard
7. Tokens and certificates used to keep the system secured can be monitored from the dashboard
8. Admin access to control the Nodes and Namespaces

V. CONCLUSION AND FUTURE WORK

A.Conclusion

Cloud computing is a vast field and requires a lot of technical knowledge to use it. The existing IAAS system's limitations can be overcome by using a cheap device called raspberry pi which serves the purpose of a pocket cloud. This setup doesn't require much of technical knowledge, yet allows students to work on cloud. This paper helps students and developers to create a cluster of Docker containers and manage it by Kubernetes easily through running scripts. Users get access to the fully equipped Kubernetes Web GUI Dashboard and will be able to perform various tasks/operations on the dashboard without worrying about ever changing commands. Users can also view real time data and a well-presented data from the dashboard. The main problem users face is of setting up the system initially, which now can be achieved by running automated scripts. Kubernetes installation and Master-Slave node configuration is done through automated scripts. Since all the Raspberry pi devices has a maximum memory of 1GB, a cluster of Docker devices can be formed and a Kubernetes master controls all of them efficiently to allow the user to deploy container applications. There is also a load balancing technique to manage the traffic / requests and the user will be

served by any of the slave nodes connected to the system. The user will not know the exact node. In this project, some of the techniques to port "libvirt' to one of the OS and automate the installation and configuration of Raspberry Pi Kubernetes clusters are used. This is limited to only containerization and doesn't have a full-fledged virtualization technique involved, thereby saving the resources and using it efficiently. The user can perform operations from the CLI and look into its reflections in the Web GUI Dashboard. The user can add/delete pods, deploy application on pods, check the status and credentials of the containers. It is possible to deploy an entire OS into a container and run it, but it is recommended to use a virtualization technique instead of containerization technique.

B.Future Work:

This system currently supports containerization and deployment of applications and Operating Systems on containers. A virtualization solution for Raspberry Pi is to be developed which can give all the system resources to the Virtual Machine environment. There is no official support from VMware or Oracle Virtual Box for Raspberry Pi devices. But this can be achieved by implementing some virtualization techniques to the Raspberry Pi. Users should be able to select between Virtualization/Containerization before deployment of their apps. A single system which is a one stop solution of virtualization and containerization will be the future enhancements of this project.

REFERENCES

[1] Spencer Julian, Michael Shuey, Seth Cook "Containers in Research: Initial Experiences with Lightweight Infrastructure" - Proceedings of the XSEDE16 Conference on Diversity, Big Data, and Science at Scale – July 2016

[2] ZHAN Ying, SUN Yong "Cloud Storage Management Technology" - 2009 Second International Conference on Information and Computing Science – November 2009

[3] Dimitrios Kelaidonis, Angelos Rouskas, Vera Stavroulaki, Panagiotis Demestichas, Panagiotis Vlacheas "A Federated Edge -IoT Architecture Enabling semantic-based service provisioning in Future Internet" - University of Piraeus, Department of Digital Systems, Piraeus, Greece – June 2016 Cloud

[4] Donggang Cao, Peidong Liu, Wei Cui, Yehong Zhong, and Bo An "Cluster as a Service: A Resource Sharing Approach for Private Cloud" - TSINGHUA SCIENCE AND TECHNOLOGY, Volume 21, Number 6 - December 2016

[5] Jheng-Jhe Sie, Shang-Chen Yang, Zih-Yun Hong, Chien-Kai Liu, Jen-Jee Chen, and Simon Cimin Li "Integrating Cloud Computing, Internet-of-Things (IoT), and Community to Support Long-Term Care and Lost Elderly Searching" - 2016 International Computer Symposium – March 2016

[6] (March 27th 2017) Raspberry Pi Documentation [Online]. Available: https://www.raspberrypi.org/documentation/

[7] (March 27th 2017) Raspberry Pi Wikipedia [Online] Available: https://en.wikipedia.org/wiki/Raspberry_Pi

[8] (March 27th 2017) What is Docker? [Online] Available: https://www.docker.com/what-docker

[9] (March 27th 2017) Docker (Software) Wikipedia [Online] Available: https://en.wikipedia.org/wiki/Docker_(software)

[10] (March 27th 2017) Cloud Computing Wikipedia [Online] Available: https://en.wikipedia.org/wiki/Cloud_computing