IJCAT - International Journal of Computing and Technology, Volume 6, Issue 4, April 2019
ISSN (Online) : 2348-6090
www.IJCAT.org
Impact Factor: 0.835

Deploying A Multi Container application in Kubernetes

[1] Arun Kumar K; [2] Vinayaditya B V; [3] Vinutha B S

[1] Assistant Professor, School of CS & IT, Jain University, Bangalore, Karnataka, India

[2] PG Scholar – School of CS & IT, Jain University, Bangalore, Karnataka, India

[3] PG Scholar – School of CS & IT, Jain University, Bangalore, Karnataka, India

Abstract - Containers have been helping teams of all sizes to solve issues with consistency, scalability, and security. Using containers such as Docker allow you to separate the application from the underlying infrastructure. Gaining that separation requires some new tools in order to get the most value out of containers, and one of the most popular tools used for container management and orchestration is Kubernetes. So in this paper we are using Docker Networking Technology to communicate between different containers.

Keywords - *Docker, Container, Kubernet.*

1. Introduction

Kubernetes, at its basic level, is a system for running and coordinating containerized applications across a cluster of machines. It is a platform designed to completely manage the life cycle of containerized applications and services using methods that provide predictability, scalability, and high availability. Kubernetes is a system designed to manage applications built within containers across clustered environments. It handles the entire life cycle of a containerized application including deployment and scaling.

As a Kubernetes user, you can define how your applications should run and the ways they should be able to interact with other applications or the outside world. You can scale your services up or down, perform graceful rolling updates, and switch traffic between different versions of your applications to test features or rollback problematic deployments. Kubernetes provides interfaces and compo sable platform primitives that allow you to define and manage your applications with high degrees of flexibility, power, and reliability.

Containers have been helping teams of all sizes to solve issues with consistency, scalability, and security. Using containers such as Docker allow you to separate the application from the underlying infrastructure. Gaining that separation requires some new tools in order to get the most value out of containers, and one of the most popular tools used for container management and orchestration is Kubernetes.

Kubernetes allows you to deploy and manage containers at scale. It was designed by Google, based on their years of running containers in production.

2. Deployment

In this paper, we used PHP programming language because, PHP is a general - purpose scripting language that is especially suited to server-side web development, in which case PHP generally runs on a web server. Any PHP code in a requested file is executed by the PHP runtime, usually to create dynamic web page content or dynamic images used on websites

I. Version: PHP 7.0
II. Java Script with bootstrap
III. Back end: kubectl v1.3

2.1 Deployment Platform

For deploying of this application, we used amazon web Services. Amazon web services provide servers on rent to deploy application. Amazon web services is the one Provide on-demand cloud computing platforms to Individuals, companies and governments, on a paid Subscription basis.

Platform: amazon web services (EC2 instance – Ubuntu 16.4 servers).

2.2 Terminology

- *Images* - The blueprints of our application which form the basis of containers. In the demo above,

we used the docker pull command to download the **busybox** image.

- *Containers* - Created from Docker images and run the actual application. We create a container using docker run which we did using the busybox image that we downloaded. A list of running containers can be seen using the docker pscommand.
- *Docker Daemon* - The background service running on the host that manages building, running and distributing Docker containers. The daemon is the process that runs in the operating system to which clients talk to.
- *Docker Client* - The command line tool that allows the user to interact with the daemon. More generally, there can be other forms of clients too - such as Kitematicwhich provide a GUI to the users.
- *Docker Hub* - A registry of Docker images. You can think of the registry as a directory of all available Docker images. If required, one can host their own Docker registries and can use them for pulling images.

2.3 Existing System

Before the use of the multiple containers, we were using single containers to host our applications. Here are some of the limitations of the same:

I. Containers can't communicate with each other
II. Containers can't exchange data between each other.

2.4 Problem Statement

Maintaining consistent environments is hard without virtualization technologies, applications are not guaranteed to work exactly the same way as they work in one environment

Containers and virtual machines solve this problem and save you the stress by isolating your application and its dependencies into a single self-contained unit that can reliably run anywhere.

3. System Architecture

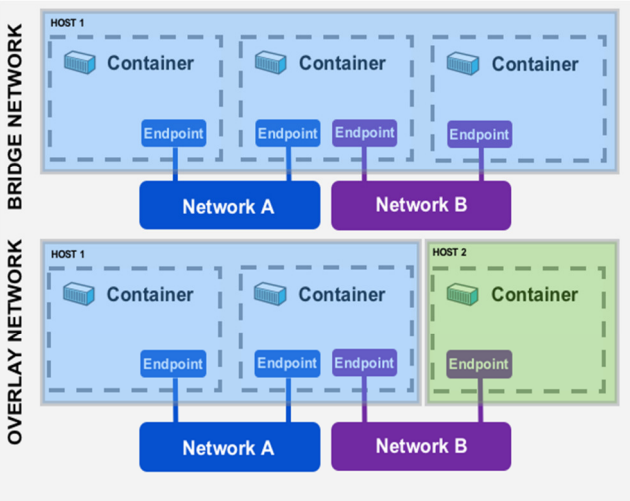

Fig.3.1 Multi - Container Architecture.

Docker takes care of the networking aspects so that the containers can communicate with other containers and also with the Docker Host.

This adapter is created when Docker is installed on the Docker Host. This is a bridge between the Docker Host and the Linux Host.

By default Docker server creates a default bridge but you can create your own bridge network and assign your container to that custom bridge on start. Or you can use docker network command which seems to be an easier path. That way you can specify the IP range to match with your DHCP settings.

1. Create your own bridge

```
docker network create --driver=bridge \
--subnet=192.168.127.0/24 --gateway=192.168.127.1 \
--ip-range=192.168.127.128/25 yourbridge
```

2. Run the container using your custom bridge;

```
docker run -d --net=yourbridge .
```

IJCAT - International Journal of Computing and Technology, Volume 6, Issue 4, April 2019
ISSN (Online) : 2348-6090
www.IJCAT.org
Impact Factor: 0.835

4. Implementation

The implementation involves
1. Create Docker Image

- Install Docker
- Install docker.io with this apt command
- Create Dockerfile
- Edit the 'Dockerfile' with vim
- Save the file and exit

2. Use Terraform to Launch Instance

- Using Terrafarm launch the Cluster of machines that u need to run your containerized application

3. Enable Auto scaling and Load balancer for High availability

- After launching instance enable auto scaling and load balancer for high availability of server

4. Containerized Application Support

- Specify dynamic ports in the ECS container task definition
- When a new task is added to the fleet, the ECS schedule auto-assigns it to the ALB using that port
- Share the ALB amongst multiple services using path-based routing
- Improve cost efficiency by running more components of your application per EC2 fleet.

5. Implementation

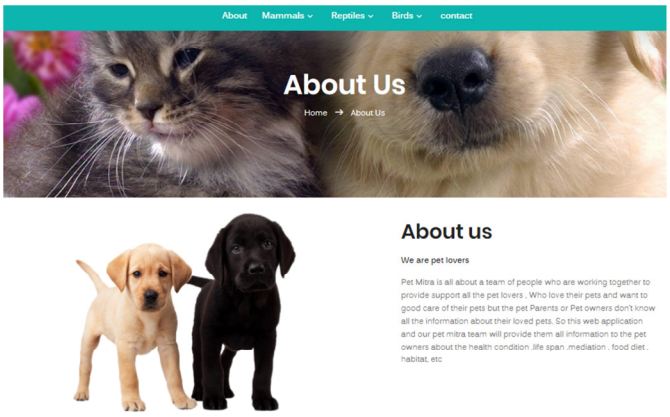

Fig 5.1 WEB Application front Page 1

This page consists of information about Pet Mitra is all about a team of people who are working together to provide support all the pet lovers,

```
*** System restart required ***
Last login: Sat Apr 27 06:11:33 2019 from 157.45.108.10
ubuntu@ip-10-0-1-122:~$ history
    1  sudo apt-get update
    2  sudo apt-get install git
    3  git
    4  git --version
    5  sudo apt-get install docker.io
    6  sudo usermod -aG docker $(whoami)
    7  sudo service docker start
    8  mkdir docker
    9  mkdir sourcecode
   10  ls
   11  cd sourcecode/
   12  ls
   13  unzip petprojectmaster.zip
   14  sudo apt-get install unzip
   15  unzip petprojectmaster.zip
   16  ls
   17  rm -rf petprojectmaster.zip
   18  cp -r Pet-Mainproject-kubernetes-master/* .
   19  rm -rf Pet-Mainproject-kubernetes-master/
   20  ls
   21  git init
   22  git add *
   23  git status
   24  git commit -m "project code"
   25  git remote add origin https://github.com/vinayaditya91/PetProject-Kuberne
tes.git
```

Fig 5.2 Server Machine information

The above figure contains information about the server machine which is consisting of Git how git is created git version how code is pushed to git

```
version: "2.0"

services:
  achi-php-apache:
    image: vinayaditya1391/petproject-kubernetes:1.0
    container_name: achi-php-apache
    ports:
      - "8081:80"
    environment:
      WORDPRESS_DB_HOST: achi-mysql
      WORDPRESS_DB_USER: root
      WORDPRESS_DB_PASSWORD: 123456
      WORDPRESS_DB_NAME: wp-dbname
#    volumes:
 #     - /home/achi/workspace/web/wordpress-template:/var/www/html:rw
    depends_on:
      - achi-mysql
    networks:
      - wp-net
  achi-mysql:
    image: mysql:5.7
    container_name: achi-mysql
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: 123456
      MYSQL_DATABASE: wp-dbname
    volumes:
      - wp-mysql-data:/var/lib/mysql
    networks:
      - wp-net

networks:
  wp-net:
    driver: bridge

volumes:
  wp-mysql-data:
~
```

Fig 5.3 Server machine

This figure consists of information about the 2 containers one for PHP application and another one for MySql and the bridge networking used to communicate them

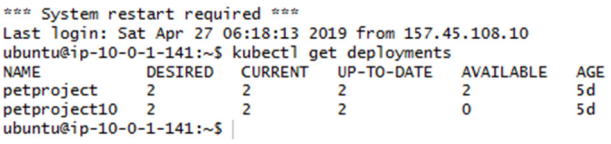

Fig 5.4 Kubernetes server information

This figure consists information about kubernetes server and the deployment information how they are deployed and how many containers are running

Fig 5.5 Pods information

This figure consists of information about the number of pods running in the kubernetes server machine.

6. Future Enhancement

For future enhancement we suggests to make this work with a n number of more than 100-200 container so that they can communicate with each other and host any size huge web application For High Availability.

7. Conclusions

This project is designed to deploy a Multi-containerized application on to a Kubernetes cluster using EKS. We have created the micro services of our web application using Docker. The web application is running inside one container and the database in another container and these containers are successfully deployed in a Kubernetes cluster which has a master and a worker node and this web application is successfully deployed and running inside the node and both the containers are communicating using Docker Networking.

References

1. Vinayaditya B V, “Kubernet Dockerize application on amazon web service using kops” International research journal of engineering and technology (IRJET), Vol 6, issue 2, 2019.
2. P. Mell and T. Grance, The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology, NIST Special Publication 800-145, 2011.
3. U. Thakrar, “Introducing Right- Scale Cloud Appliance for vSphere,” blog, 10 Dec. 2013;
4. B. Kepes, “VoltDB Puts the Boot into Amazon Web Services, Claims IBM Is Five Times Faster,”Forbes,2014; azon-web-services -claimsibm-5-faster.
5. J. Petazzoni, “Containers & Docker: How Secure Are They” blog, 21 Aug. 2013;
6. J. Petazzoni, “Linux Containers (LXC), Dockers and Security,” 2014;
7. https://github.com/kubernetes/kops/blob/master/docs/cli/kops_create_secret_encryptionconfig.md
8. https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/
9. https://github.com/kubernetes/kops/blob/master/nodeup/pkg/model/kube_apiserver.go#L61
10. https://github.com/georgebuckerfield/kops/blob/master/pkg/apis/kops/cluster.go#L162