2021 International Conference on Computing, Communication and Green Engineering (CCGE)
JSPM's RSCOE, Pune, India. Sep 23-25, 2021

Creating Helm Charts to ease deployment of Enterprise Application and its related Services in Kubernetes

Shivani Gokhale
Dept. of Computer Engineering
MKSSS's Cummins College of Engineering for Women, Pune
Pune, India
shivani.gokhale@cumminscollege.in

Reetika Poosarla
Dept. of Computer Engineering
MKSSS's Cummins College of Engineering for Women, Pune
Pune, India
reetika.poosarla@cumminscollege.in

Sanjeevani Tikar
Dept. of Computer Engineering
MKSSS's Cummins College of Engineering for Women, Pune
Pune, India
sanjeevani.tikar@cumminscollege.in

Swapnali Gunjawate
Dept. of Computer Engineering
MKSSS's Cummins College of Engineering for Women, Pune
Pune, India
swapnali.gunjawate@cumminscollege.in

Aparna Hajare
Dept. of Computer Engineering
MKSSS's Cummins College of Engineering for Women, Pune
Pune, India
aparna.hajare@cumminscollege.in

Shilpa Deshpande
Dept. of Computer Engineering
MKSSS's Cummins College of Engineering for Women, Pune
Pune, India
shilpa.deshpande@cumminscollege.in

Sourabh Gupta
Veritas Technologies LLC
Pune, India
Sourabh.Gupta@veritas.com

Kanchan Karve
Veritas Technologies LLC
Pune, India
Kanchan.Karve@veritas.com

***Abstract*—Modern day software applications are required to have high availability and performance capabilities to ensure highly productive features and a smooth user experience. It becomes increasingly difficult for organizations to innovate with rapid building, testing and deployment of systems in static, monolithic environments. In order to ascertain the development of resilient applications, Kubernetes is widely used for distributed systems for workload scalability and orchestration of containers. The management of the system using Kubernetes becomes progressively inconvenient with increasing size and complexity. In order to make the process of Kubernetes configuration simpler and faster, Helm charts are used to preconfigure applications and automate the processes of development, testing and production. This paper proposes a method to ease the deployment of the enterprise application in Kubernetes using Helm charts. Our study shows that deployment of Kubernetes resources is simplified using Helm such that applications can be defined as a set of components in the minikube Kubernetes cluster. The experimental results of the proposed method show that there is 6.185 times speed improvement in the deployment process by using Helm. This makes it extremely influential for DevOps teams to improve their cluster management.**

Keywords—Authentication service, Backup service, Cloud Computing and Containerization, Docker, Front end service, Helm Charts, Kubernetes

I. Introduction

A business' IT systems keep growing as their operations go on increasing. This becomes challenging when new features need to be added, changes have to be made or functions have to be scaled. The IT industry is constantly under great pressure to enhance agility as well as enable faster delivery of new functionality to the business domain. A specific focus of importance is the deployment of new or improved application programs at the immediate frequency that any digital transformation demands [1]. Using containers, complex programs with their configuration and deployment related dependencies are encapsulated and run in an isolated environment making applications portable, hence adopted in High Performance Computing (HPC) clusters [2]. Containerization is a type of virtualization of the operating system, by means of which applications can be executed in isolated, independent user spaces called containers, while all of them make use of the same shared operating system (OS) [3]. Containerization provides the benefits of: increased portability, improved scalability, simple and fast deployment, enhanced productivity and improved security. Kubernetes is a container orchestration solution [4] which simplifies the management of multiple containerized applications. It facilitates the automation of the deployment, scaling and networking of containers and encompasses portability and extensibility in its platform for management of containerized workloads and services, allowing declarative configuration as well as automation [5]. Kubernetes achieves reliability and availability of services during server downtime and physical disk damage [6]. The application performs backup of user data upon authentication of the user's identity. It is deployed in Kubernetes with a front end service monitoring the cluster and its pods. However, it becomes a tedious and complex process to write configuration files for each module of the application in the Kubernetes setup. In order to solve this problem, it is necessary to automate the process of configuration of individual Kubernetes services and deployments.

This paper presents a method to simplify and self-regulate the deployment process. The proposed method focuses on writing Helm charts to ease the entire deployment of Backup related services and front end services. Helm charts are a kind of deployment configuration of YAML files. These files will then be used to maintain and monitor the lifecycle of the front end service and deployment of the application in Kubernetes.

The rest of the paper is organized as follows. Section II reviews related work about microservice based applications using Docker as a containerization tool and Kubernetes for

2021 International Conference on Computing, Communication and Green Engineering (CCGE) | 978-1-6654-1509-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CCGE50943.2021.9776450

orchestration. In Section III, we propose an efficient method for simplifying the deployment of microservice based applications in Kubernetes using Helm charts. In section IV, we present an implementation of a Helm chart to deploy our application in a Kubernetes cluster. The analysis of the results is discussed in section V. Section VI highlights the conclusions and future work.

II. Related Work

Several efforts have been devoted to improve scalability and availability of large scale enterprise applications and to ease the deployment of such applications using some orchestration tools like Kubernetes. With today's era of Cloud Computing and Containerization, research based on scheduling of containers and the need to automate deployment of such containerized applications has drawn increasing attention. Kubernetes Container Scheduling Strategy [7] (KCSS) is proposed by Tarek Menouer [8] which provides a global vision about the state of cloud and user's needs. Yanghu Guo et al. [9] presented a study of container scheduling based on neighborhood division. In order to ensure high availability, scheduling of containers is essential which can be achieved through some orchestration tool like Kubernetes having inbuilt features of self-healing in order to manage the desired state of the cluster. Kubernetes enables self-healing capability through failure recovery operations to improve availability of large scale applications [9].

The architectural style of microservices has emerged primarily from industry. Dragoni et al. in their work [10] propose the definition of a Microservice as a small and unconstrained process that interacts by messaging. According to their work, microservice based architecture comprises multiple microservices which provides remarkable effect in improving quality attributes of an application [11]. Along with performance and maintainability, they emphasize their work specifically on availability as a quality attribute which has a significant effect on Microservice based architecture. Khazaei et al. in [12] propose a cloud based platform including relevant microservices that uses Docker containerization technology to allocate containers based on requests of microservice users [13]. One of the significant differences between such a type of a platform and an orchestration tool like Kubernetes is that this platform has the ability to ask for more virtual machines from infrastructure when needed while Kubernetes does not. Apart from Helm there are some tools that ease the deployment of microservices in Kubernetes as an orchestration tool. One of them is Terraform which acts as an alternative to Helm that simplifies deployment and provides support for various cloud solutions. However, it doesn't provide automatic rollback and upgrade operations on Kubernetes resources [14].

In this paper, we propose a method which uses a Helm chart such that a versioned history of each deployed release is automatically maintained by Helm. This ensures that rollback and upgrade operations are performed in an efficient manner in order to maintain the state of the cluster and manage Kubernetes resources [15].

III. Proposed Method

The applications that are developed using Kubernetes as an orchestration tool contain resource files in declarative format for varied Kubernetes objects like Deployments, Services, ConfigMaps, PersistentVolumeClaims and so on. Distributing and managing such large scale enterprise applications based on Kubernetes is difficult. Packaging of all Kubernetes resource files is done by Helm in the form of "charts". The Helm chart mainly follows "Push Button Deployment". A chart can be considered as a Kubernetes package which bundles all the dependencies, metadata and information about resource files into a single folder and deployed as a single unit. Manual deployment in Kubernetes requires that each service and deployment YAML file needs to be applied separately using kubectl commands. In industry oriented business applications which contain a huge number of configuration related YAML files there is a need to automate such deployment processes. In order to ease the deployment of the application along with its related services, Helm is used which acts as package manager for Kubernetes [12]. This kind of deployment provides capabilities for performing operations to manage the life cycle of containerized applications using Helm install, upgrade, rollback and uninstall commands [15].

IV. Implementation

The sample application has been developed comprising three microservices: the authentication, backup and frontend services. The entry point for our Application Programming Interface is the front end service [16]. This service continuously monitors the authentication and backup services. The user sends HTTP requests from the User Interface of the application. This request contains the user credentials for verifications and user data to be backed up. The front end service then sends a request to the authentication service to authenticate the user. Data validations have been performed wherever necessary and appropriate errors are returned in case of invalid user credentials input by the user. Upon successful authentication of the user, his data is sent to the backup service as a subsequent request from the front end service for backup to the database. There are two versions of the backup service. The first version of the service allows repeated data entry for backup, whereas the second version discards them [7]. Fig. 1 shows the diagrammatic representation of the process.

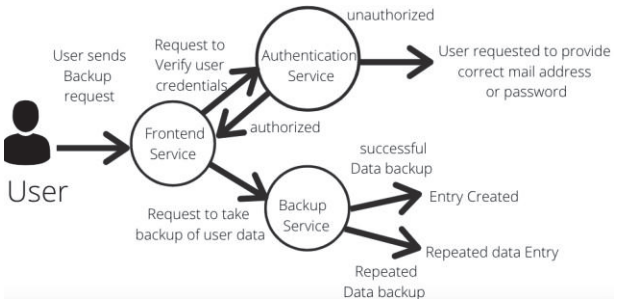

Fig.1. User Activity Diagram

Microservices of the application have been developed using the Flask framework of Python. The minikube tool has been used to run Kubernetes (k8s) on the local machine. It creates a single node cluster contained in a virtual machine (VM).

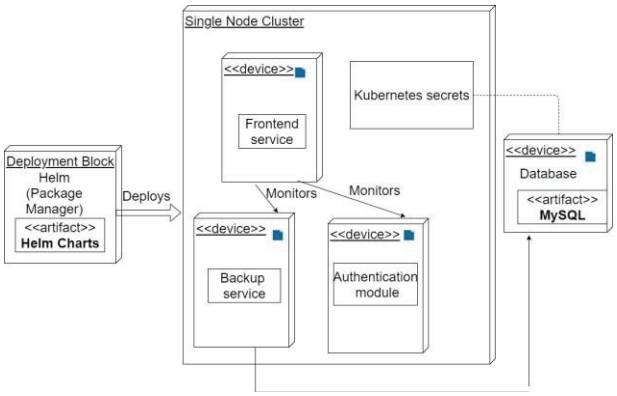

Fig. 2. Module Diagram

Fig. 2 shows the Module diagram of the application which comprises three microservices such that each microservice is containerized [17] and represents a running instance in the Kubernetes cluster for each of the services. A persistent storage is used for backup to ensure quality attributes like availability and consistency. A MySQL Server has been used as the persistent storage which runs in a separate pod within the same cluster. A Helm chart is used to deploy these microservices and MySQL database in the Kubernetes cluster.

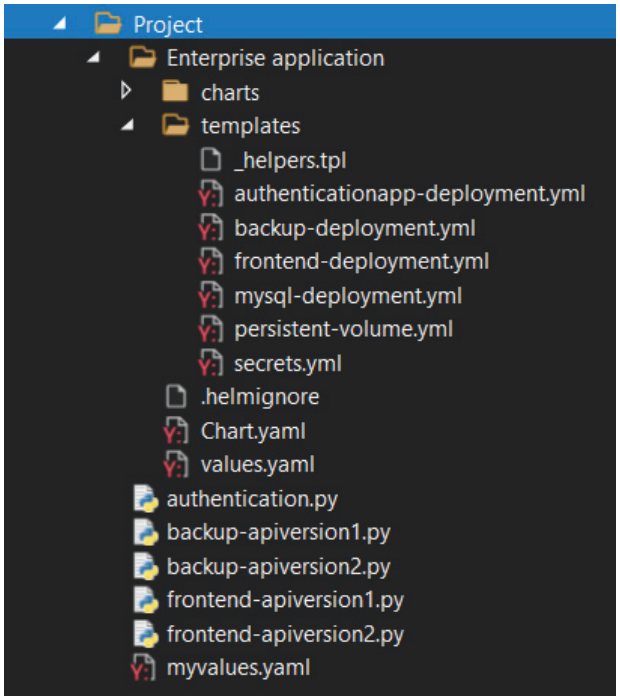

Fig. 3. Project Structure

YAML is a text based serialization standard which specifies configuration related information in the form of key-value pairs. YAML files have been used to set the configurations of each deployment module of the application within Kubernetes. Fig. 3 shows the project structure which represents the Helm charts directory.

A. YAML files description

1) Chart.yaml: It acts as an entry point in which the apiVersion, name of the Helm chart's directory, the chart version and version of the application being deployed (which is appversion) has been mentioned.

2) Persistent-volume.yaml: A persistent volume is a resource containing data storage which will remain intact irrespective of the pod's lifecycle. In stateful applications like a MySQL database, in order to persist the data even if the claim is deleted a PersistentVolumeClaimPolicy is used such that the data can be backed up or retrieved manually from the database.

3) Secrets.yaml: Kubernetes has an inbuilt secret object for dealing with sensitive information. The database password is specified in encrypted format in this YAML file.

4) Mysql-deployment.yaml: In this YAML file information has been specified to spin up a single pod. It will manage all the pods having their label as 'db'. The Mysql container will obtain the database credentials (database name, database password) from the Secrets.YAML. The port number that the container exposes and which path to be mounted to the persistent-volume have also been mentioned.

5) Deployment YAML files: There are three deployment YAML files for authentication, backup and front end service modules each having attributes for apiVersion, type of deployment, metadata, image details, number of replicas of image and configuration of database details.

6) Service YAML files: There are three service YAML files for each authentication, backup and front end service module. Here the service type and port number on which service is exposed have been mentioned.

7) Values.yaml: Values is a built-in object that the Helm template includes. This object offers access to attributes provided within a chart. Common attributes can be listed in this file so that it can be easily referred to in the other YAML files.

8) Myvalues.yaml: It is outside the Helm charts directory and has been used to mention attributes that users want to upgrade in order to move the application to a newer version so that these attributes will get overridden with attributes in the Values.YAML file.

B. Helm commands

Helm has been used to manage the entire lifecycle of all the pods of the application deployed in the Kubernetes cluster.

1) Helm install: is used to deploy the application and the front end service in the Kubernetes cluster along with its related services and the persistent storage[13]. Hence authentication service, backup service, front end service and mysql-persistent storage are deployed using a single command of "Helm install". Fig. 4 shows the Helm install command by which the entire application gets deployed in the Kubernetes cluster with the status of all the pods as 'running'.

```
user@user-Inspiron-15-3567:~/Documents$ helm install ente
rprise-application ./enterpriseApplication -n=enterprisea
pplication
NAME: enterprise-application
LAST DEPLOYED: Sat Jul 10 20:31:28 2021
NAMESPACE: enterpriseapplication
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Fig. 4. Helm Install

2) Helm upgrade: is used to upgrade a release to a newer version of a chart. The first version of the backup service allows storing repeated entries. In the second version a check is made for any user having the same backup information as that of the already existing data in the database and then the repeated entry is discarded. This versioning is performed using Helm upgrade. Fig. 5 shows the Helm upgrade command by which the application is moved to the newer version.

```
user@user-Inspiron-15-3567:~/Documents$ helm upgrade ente
rprise-application enterpriseApplication -f myvalues.yaml
 -n=enterpriseapplication
Release "enterprise-application" has been upgraded. Happy
 Helming!
NAME: enterprise-application
LAST DEPLOYED: Sat Jul 10 20:34:04 2021
NAMESPACE: enterpriseapplication
STATUS: deployed
REVISION: 2
TEST SUITE: None
```

Fig. 5. Helm Upgrade

3) Helm rollback: is used to restore the application back to its earlier state. If the user's requirement for backup functionality changes then the entire application can be moved to the earlier version in which repeated entries for backup information are allowed. Fig. 6 shows the Helm rollback command by which the application is moved to the earlier version.

```
user@user-Inspiron-15-3567:~/Documents$ helm rollback ent
erprise-application 1 -n=enterpriseapplication
Rollback was a success! Happy Helming!
```

Fig. 6. Helm Rollback

4) Helm uninstall: is used to clean up and undo everything done at the time of installation. Fig. 7 shows the Helm uninstall command by which deployment of applications can be completely deleted from the Kubernetes cluster.

```
user@user-Inspiron-15-3567:~/Documents$ helm delete enter
prise-application -n=enterpriseapplication
release "enterprise-application" uninstalled
```

Fig.7. Helm Uninstall

V. Analysis and Discussion

In order to deploy containerized applications with inbuilt Kubernetes functionality using "kubectl commands", all the Kubernetes resources, deployment files and service files will have to be deployed in the Kubernetes cluster. To deploy each of these resources a separate kubectl command will have to be executed respectively. To solve this problem we proposed an innovative solution which will automate the deployment process.

Table I and Table II show the experimental results which demonstrate the outstanding performance using the proposed Helm based method, in terms of "speed improvement" of the deployment process. The method used to perform this analysis is the Kubernetes inbuilt command - "kubectl get pods". On executing the above command the name and age of the pods can be known; the age shows the time required to bring the pods into running state. According to this hypothetical analysis (Table I), deployment in Kubernetes without using Helm requires deploying all the pods separately which takes 14 seconds for bringing each pod up and running; it additionally takes 83 seconds to pull the MySql image for the database. Table II shows that using Helm, each of the microservices along with the MySql deployment, Kubernetes secrets resource and persistent volume will be applied to the cluster with a single command. 27 seconds are required to bring all pods of the application in running state using "Helm install command" whereas deployment using "kubectl apply" command requires 167 seconds to bring all pods of the application in running state.

TABLE I. Execution Time for Microservices with "kubectl apply" Deployment

Actions	Kubernetes deployment
Authentication service deployment	14 seconds
Backup service deployment	14 seconds
Kubernetes secrets deployment	14 seconds
Front end service deployment	14 seconds
Mysql deployment	14 seconds
Persistent volume deployment	14 seconds
Mysql image pull	83 seconds
Total time	167 seconds

TABLE II. Difference in Deployment Time

Deployment using Helm	Deployment without using Helm
27 seconds	167 seconds

In Fig. 8 this difference has been plotted; the bar graph shows 6.185 times speed improvement in deployment of the application upon using Helm charts. For large scale enterprise applications there will be a large amount of service and deployment files which will result in an even more significant speed improvement.

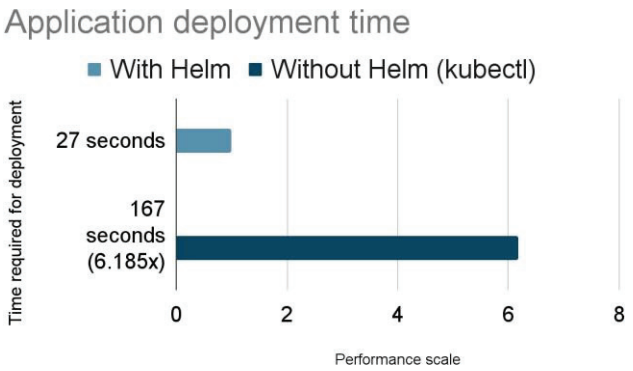

Fig. 8. Performance improvement using Helm

Additional advantages of Helm charts:

- Applications' deployment process is simplified and API versioning can be easily performed with Helm.
- Since the deployment is automated, it is less prone to human error.
- Helm helps to manage the lifecycle of containerized applications.
- A single file manages all the configurations.

VI. Conclusion and Future Work

In this paper we proposed a method to ease the deployment of containerized applications in Kubernetes using Helm charts. The proposed method not only eases the deployment process but also achieves quality attributes like performance, availability and maintainability.
Experimental results show that:

- There is a 6.185 times speed improvement in deployment using Helm charts.
- Lifecycle management of applications is eased by automating the deployment process with Helm.

This paper signifies how Helm can notably reduce the efforts required for deploying large scale applications.

In the future these Helm charts can be optimized to get efficient results.

References

[1] Nathan C. Sheffield, "Bulker: A Multi-container Environment Manager", Department of Public Health Sciences, University of Virginia,2019

[2] Wing-Kwong Wong, Cheng-Sheng Lee, "An Implementation of Face Recognition with Deep Learning based on a Container-Orchestration Platform", 2020 Indo – Taiwan 2nd International Conference on Computing, Analytics and Networks (Indo-Taiwan ICAN)

[3] Sachchidanand Singh, Nirmala Singh "Container and Docker: Emerging roles and Future of cloud technology",2016 International Conference (iCATccT)

[4] https://kubernetes.io [Accessed on: 1st July, 2021]

[5] Naweiluo Zhou, "Container Orchestration on HPC Systems through Kubernetes- Journal of Cloud Computing-22 February 2021

[6] Anqi zhao "Research on Resource Prediction model based on Kubernetes Container-auto scaling technology IOP Conference 2019

[7] Tarek Manouer, "Kubernetes Container Scheduling Strategy"Journal of SuperComputing 24th September 2020

[8] Jinze Lv, Mingchag wei,Yang Yu," A Container Scheduling Strategy based on machine learning",2019 IEEE International Conference

[9] Yanghu Guo, "A Container Scheduling Strategy Based on Neighborhood Division in Microservice," IEEE ICC 2017 SAC Symposium Cloud Communications and Networking Track, May 2017

[10] Leila Abdollahi Vayghan "Deploying Microservice based applications with Kubernetes:Experiments and lessons learned 2018 IEEE international conference on Cloud Computing

[11] N. Dragoni et al., "Microservices: Yesterday, Today, and Tomorrow," in Present and Ulterior Software Engineering, M. Mazzara and B. Meyer, Eds. Cham: Springer International Publishing, 2017, pp. 195–216

[12] Leila Vayghan, Mohmed Aymen Saied " Kubernetes as an Availability Manager for Microservice Applications",2019

[13] H. Khazaei, C. Barna, N. Beigi-Mohammadi, and M. Litoiu, "Efficiency Analysis of Provisioning Microservices," in 2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2016, pp. 261–268

[14] https://www.trustradius.com/products/hashicorp-terraform/reviews?qs=pros-and-cons [Accessed on 4th July, 2021]

[15] https://helm.sh [Accessed on: 7th July, 2021]

[16] https://towardsdatascience.com/how-to-deploy-a-flask-api-in-kubernetes-and-connect-it-with-other-micro-services-af16965b67fe [Accessed on: 8th July, 2021]

[17] https://www.docker.com[Accessed on: 10th July, 2021]