Orchestrating a stateful application using Operator

Deekshith Nayak[1] and Dr. H. V. Ravish Aradhya[2]

[1,2] RV College of Engineering, Bengaluru, India

[1] deekshithnayak.ec17@rvce.edu.in , [2] ravisharadhya@rvce.edu.in

Abstract: *Containerization is a leading technological advancement in cloud native developments. Virtualization isolates the running processes at bare metal level but containerization isolates the processes at operating system level. Virtualization encapsulates all the new virtual instances with a new operating system but containerization encapsulates the software only with its dependencies. Containerization avoids the problem of dependency missing between different operating systems and their distributions. Concept of containerization is old but with the development of open-source tools like Docker, Kubernetes and Openshift accelerated the adaption of this technology. Docker builds container images and Openshift or Kubernetes is an Orchestrating tool. For stateful applications Kubernetes workload resources are not a better option to orchestrate the application, as each resource has its own identity. In such cases operator can be built to manage the entire life cycle of resources in the Kubernetes cluster. Operator combines human operational knowledge into software code in a systematic way. The paper discusses the default control mechanism in Kubernetes and then it explains the procedure to build the operator to orchestrate the stateful application.*

Keywords: Orchestration, Kubernetes, Openshift, Docker, Container, Workload, Controller, Resources, Operator.

1. Introduction

Extensive research in computing and low power industry has improved processing power and computing capacity of bare metal servers. Virtualization has been emerged as an option over bare metal servers for the efficient use of available resources on the single server bare metal system. Virtualization enables to run a greater number of virtual instances as a server on the single bare metal server. But all the new virtual instances will be encapsulated with a new operating system instance. Hence though virtualization makes efficient use of resources in single server system but it increases the resource overhead with respect to redundancy. In order to overcome the disadvantages of virtualization, the new technology that avoids the problem of redundant operating system overhead on each virtual instant, is containerization. Containerization binds only required dependencies with the software program and not the entire OS with the new virtual instance. In order to achieve isolation between running containers, containerization makes use of two kernel level features of Linux called cgroups and namespaces. Control groups, which is abbreviated as cgroups is a kernel level mechanism that controls and meassures the resources used by a group of running processes in Linux.

Namespaces also a kernel level mechanism that limits the visibility a group of processes have to the rest of the system. As part of containerization, each virtual instances are called containers.

Containers will carry only the software code with its required dependencies in the form of software image. Docker is a tool that helps in building software images that can be run as containers. Docker makes use of docker file to build software images. Kubernetes is a container orchestration tool, that helps in automating the entire life cycle of container deployment. Hence it avoids any problem that arises in life cycle management of a container. Along with container orchestration, it also helps in building clusters of nodes and managing them. Openshift is a PAAS platform, that adds additional functionality over Kubernetes.

Any application can be orchestrated through Kubernetes native workload resources but in case of stateful applications, where in each instance of software stores specific information regarding the user's previous activity. Thus, each instance of stateful service will have specific state, hence each request from the user has to reach a specific instance of the server. The particular control activity cannot be carried by Kubernetes workload resources. Operator is a custom controller which includes human operational intelligence on how to control such stateful applications and thus managing them for high availability. This paper discusses on the process of building operators to orchestrate stateful applications.

2. Containerization and Orchestration

2.1. Containerization and Docker

Containerization is a mechanism in which applications run in an isolated environment called containers. For an application to run inside a container, it should be in the form of software image. An application software along with its dependencies, binaries and configurations will be bundled together to form a software image and that image will be running inside containers. Containerization is achieved by using cgroups and namespaces in Linux. Container runs as an isolated unit from the host operating system. Containerization makes use of the concept of cgroups and namespaces. cgroups are also called as control groups which controls the resources, a process or group of processes can use. Namespaces are also control mechanisms, which controls the view, a container will have towards the host. These two mechanisms help containers to run in an isolated way. Docker is a tool that helps in building software images, thus container-based implementations. Docker has got a lot reputation that it has become standard in industries. The way docker helps to build portable and isolated applications in microservice architecture makes it suitable from small scale to large scale industries. Docker is a container engine that runs containers on operating system by making use of kernel features.

2.2. Orchestration

Orchestration is the process of automating most of the work in deploying a workload resource in container cluster. Such efforts include managing a container's life cycle, deploying the workload, provisioning resources, networking, scaling the number of instances, load balancing and many more things. Kubernetes and Openshift are the 2 major container orchestration tools.

Kubernetes is an open-source system for container orchestration developed at google. Kubernetes helps in managing containerized applications. By managing it means deploying, scaling and other networking efforts. It groups all the containers belonging to an application into a logical unit for easier infrastructure management. Kubernetes is also called as k8s or kube. Kubernetes can span across any premises that makes it ideal for cloud native application development. Currently Kubernetes is owned by Cloud Native Computing Foundation. Along with managing, Kubernetes adds additional features to orchestration technique that enhances container-based solutions. Kubernetes even helps in running stateful applications by allocating and mounting persistent volume to the running workloads. Periodic health check and self-healing techniques of Kubernetes helps in retaining the containers in the desired state. Kubernetes also controls the updates to applications with various techniques. To be precise main goal of Kubernetes is to ensure high availability of running application. Openshift is a open source Platform as a service development platform. It's a platform to develop and deploy application on cloud premise. Openshift is extensible and has capability to support various languages. Openshift is basically an upper set over Kubernetes and has all the functionality that Kubernetes provides. Along with that Openshift adds several features including security, which makes it a primary choice to run enterprise clusters. Openshift has its own source code version management and several database support. It provides a very responsive web condole. Along with web console Openshift has command line interface tools.

3. Stateful application

Stateful applications stores data of the existing session from client and uses that data to compute the current state of the application. The client data from the previous session will be used to serve the client in the next session. In containerization wherein multiple servers will be running to serve a service in microservice architecture, Stateful servers play a crucial role in maintaining consistency and low latency. Stateful applications have their importance in database-based application wherein databases contain data needed to process the request. The stateful applications can be majorly found in the cloud native application development industries, where computing near to storage area makes the service fast and decreases latency. State of an application is defined to be anything that it be at that time. For stateless applications it will be defined on the basis its activity, it can be running, restarting etc. But in the case of stateful application, the state is not only defined on activity but also on the data it stores. Load balancer is an application that sends client request to specific server based on some predefined algorithm which can be adaptive. Hence load balancers are the key software layer in stateful application, as each request from the client has to be delivered to a specific target and all server instances cannot be treated as same. Whenever the load balancer receives a request for service, it has to forward the request to specific instance of the server. Whereas in the case of stateless application the client request can be forwarded to any instance of the server. The difference exist in the case of stateful service is because each instance of the server will be having client metadata stored from the previous session. Hence only those

instances will be able to serve the specific request. However, the databases attached to several instances of the same kind of server will be consistent.

4. Operator as a custom controller

4.1 Custom controller

Kubernetes native controllers are capable of orchestrating the stateless application. In stateless application the order of deployment of pods doesn't matter. Because since the pods in the application doesn't have a specific state so any pod of a specific kind can receive the user traffic. But in case of a stateful application each and every pod of a specific kind will be attached to a specific database and thus has a defined state. Hence order of deployment of pods become important in case of a stateful application. To orchestrate stateful applications, engineer needs to apply the desired state and action in the form of software and thus indulging human operational knowledge in code. Operators are piece of code that will have human intelligence in terms of handling stateful applications. Hence operators help to deploy, pack and manage stateful application in Openshift. Engineer defines the desired state of each workload in the cluster with the code and builds the operator. Operator periodically keeps a health check on the workloads to see if they are in the desired state. Operator will get notified about any unusual change in the state of the workload. The process is called reconciliation loop in operators. Building operators is a tedious task. However, it can be simplified with the help of Operator-SDK. Operator-SDK is a part of the Operator framework project. It's an open-source tool to build operators. Various functionalities of Operator-SDK make the process of building operators easier. It helps in building, testing and packaging the operator's code. It helps in writing the operational logic more intuitively with its high-level Application Programming Interface (API)s. It creates scaffolding or boilerplate required to build an operator for the new project thus it cuts down the time spent in doing additional stuffs rather than writing the actual logic. It also has extensions to cover common use cases of operators.

4.2. Designing operator

The service built in the Openshift cluster is being watched by the Kubernetes native controllers, which has limited functionality in monitoring the resources. Each native controller is assigned with a specific control action. So, it cannot reflect human operational knowledge. Engineers have to keep on checking for the health of the resources to avoid any downtime and thus maintaining high availability. Operator SDK does the basic scaffolding required for the operator build.

4.3 Reconciliation in operator

Steps in writing reconciliation logic for operator

1. Define the resources that has to be watched by the controller.
2. Define the required state through specifications.
3. Store the current state through variables.
4. Compare the current state and the required state.

5. If current state and required state are different then the operator will run the logic defined in reconciliation loop.

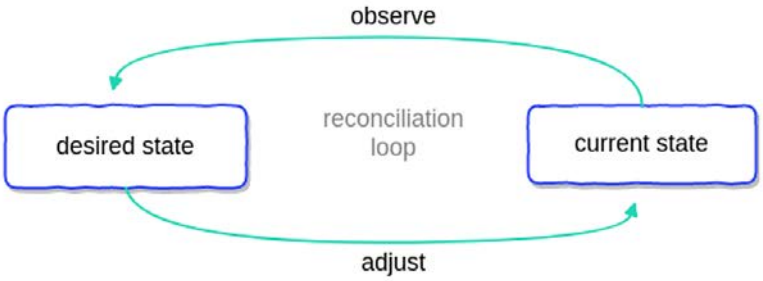

Figure 1 Reconciliation loop in Operator

5. Implementing stateful application with operator

Steps involved in building an operator for orchestration of stateful application with the help of operator-SDK are explained in this section.

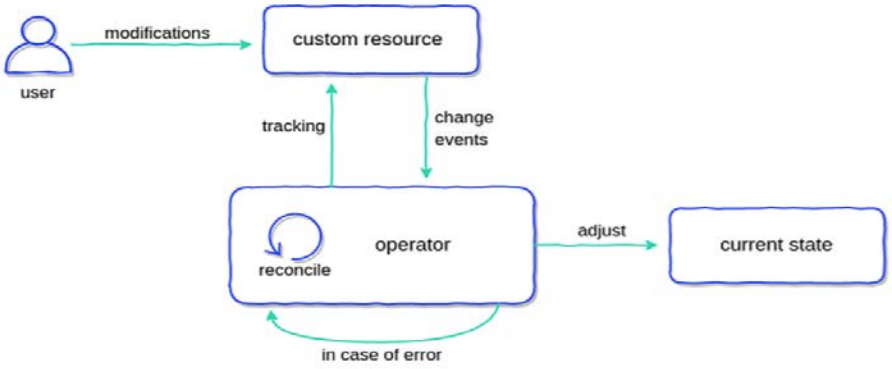

Figure 2 Operator as a custom controller

1. Install Operator Software Development Kit (SDK) with its dependencies such as Golang, Docker etc.
2. Prepare a basic scaffolding and API for the operator using operator-sdk init command with necessary tags. Tags required to keep track of versions in API.
3. Each API will be seen as an end point to the service in Openshift.
4. Define the API for the custom resource definition.
5. Each time whenever there is a modification to custom resource definition, run the make generate and make manifests to update the deep copy of Custom Resource Definition (CRD).
6. Write the controller file in a modular way for the required reconcile logic for the custom resource.

7. Deploy the custom resource definition into cluster by enabling web hooks.
8. Run the operator either as a local controller or as a pod in the cluster.
9. Deploy the custom resource YAML file into the cluster.
10. After the operator configured with required permissions in the namespace, mirror repository service will be up on the cluster.

6. Conclusion

Server computing technologies started with the development of bare metal servers. With the rapid enhancement in the capacity and efficiency of hardware resources, the bare metal computing techniques doesn't stand reliable. It won't be able to utilize full capacity of the hardware, in fact a very huge proportion of its capacity will be stringent. To overcome the disadvantages of bare metal server handling technique, by this to utilize full efficiency of the hardware resources, industries started looking towards virtualization technology. Virtualization helps in running several servers on a single hardware, by combining each server with its own instance of operating system. But virtualization adds overhead of carrying the entire operating system with each server the hardware has to run. To overcome such redundant usage of resources, Containerization developed as a new technique. Containerization binds application code with its dependencies and not with the entire operating system. Containerization adds advantage of avoiding redundant usage of resources. But Services running by containerization are difficult to handle for human operator. Hence Kubernetes workload resources help in minimal orchestration process. But that won't be enough for stateful applications because of the uniqueness of all the pods. In this paper steps in building an operator for orchestrating stateful applications. Operator-SDK has been used as a framework for development.

REFERENCES

[1] *V. Revuri, B. Ambika, D. Sravan Kumar, and C. Lokanatha Reddy, "High performance research implementations with third party cloud platforms and services," Materials Today: Proceedings, 2021, issn: 2214-7853. doi: https://doi.org/10. 1016/j.matpr.2021.01.755. [Online]. Available: https://www.sciencedirect. com/science/article/pii/S221478532100852X.*

[2] *C. Ramalingam and P. Mohan, "Addressing semantics standards for cloud portability and interoperability in multi cloud environment," Symmetry, vol. 13, no. 2, 2021, issn: 2073-8994. doi: 10 . 3390 / sym13020317. [Online]. Available: https : //www.mdpi.com/2073-8994/13/2/317.*

[3] *A. Iasio and E. Zimeo, "A framework for microservices synchronization," Software: Practice and Experience, vol. 51, Aug. 2020. doi: 10.1002/spe.2877.*

[4] *S. Laskawiec, M. Chora´s, R. Kozik, and V. Varadarajan, "Intelligent operator: Machine learning based decision support and explainer for human operators and service providers in the fog, cloud and edge networks," Journal of Information Security and Applications, vol. 56, p. 102 685, 2021, issn: 2214-2126. doi: https: //doi.org/10.1016/j.jisa.2020.102685. [Online]. Available: https://www. sciencedirect.com/science/article/pii/S2214212620308371.*

[5] *A. Hamo and H. I. AlNuri, "Cloud system for software testing," International Journal of Engineering and Computer Science, vol. 6, no. 12, pp. 22 288–22 293, Dec. 2017. [Online]. Available: http://www.ijecs.in/index.php/ijecs/article/ view/2565.*

[6] *P. P¨a¨akk¨onen, D. Pakkala, J. Kiljander, and R. Sarala, "Architecture for enabling edge inference via model transfer from cloud domain in a kubernetes environment," Future Internet, vol. 13, no. 1, 2021, issn: 1999-5903. doi: 10.3390/fi13010005. [Online]. Available: https://www.mdpi.com/1999-5903/13/1/5.*

[7] *L. A. Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, "A kubernetes controller for managing the availability of elastic microservice based stateful applications," Journal of Systems and Software, vol. 175, p. 110 924, 2021, issn: 0164-1212. doi: https://doi.org/10.1016/j.jss.2021.110924. [Online]. Available: https: //www.sciencedirect.com/science/article/pii/S0164121221000212.*

[8] *O. Arouk and N. Nikaein, "5g cloud-native: Network management automation," in NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium, 2020, pp. 1–2. doi: 10.1109/NOMS47738.2020.9110392.*

[9] *M. Bogo, J. Soldani, D. Neri, and A. Brogi, "Component-aware orchestration of cloud-based enterprise applications, from tosca to docker and kubernetes," Feb. 2020.*

[10] *E. Kristiani, C.-T. Yang, C.-Y. Huang, Y.-T. Wang, and P.-C. Ko, "The implementation of a cloud-edge computing architecture using openstack and kubernetes for air quality monitoring application," Mobile Networks and Applications, Jul. 2020. doi: 10.1007/s11036-020-01620-5.*

[11] *N. Kapo˘cius, "Overview of kubernetes cni plugins performance," Mokslas - Lietuvos ateitis, vol. 12, pp. 1–5, Feb. 2020. doi: 10.3846/mla.2020.11454.*

[12] *F. Rossi, V. Cardellini, F. Lo Presti, and M. Nardelli, "Geo-distributed efficient deployment of containers with kubernetes," Computer Communications, vol. 159, pp. 161–174, 2020, issn: 0140-3664. doi: https://doi.org/10.1016/j.comcom. 2020.04.061. [Online]. Available: https://www.sciencedirect.com/science/ article/pii/S0140366419317931.*

[13] *P. Kayal, "Kubernetes in fog computing: Feasibility demonstration, limitations and improvement scope: Invited paper," in 2020 IEEE 6th World Forum on Internet of Things (WF-IoT), 2020, pp. 1–6. doi: 10.1109/WF-IoT48130.2020.9221340.*

[14] *A. A. Khatami, Y. Purwanto, and M. F. Ruriawan, "High availability storage server with kubernetes," in 2020 International Conference on Information Technology Systems and Innovation (ICITSI), 2020, pp. 74–78. doi: 10.1109/ICITSI50517.2020. 9264928.*

[15] *A. M. Potdar, N. D G, S. Kengond, and M. M. Mulla, "Performance evaluation of docker container and virtual machine," Procedia Computer Science, vol. 171, pp. 1419–1428, 2020, Third International Conference on Computing and Network Communications (CoCoNet'19), issn: 1877-0509. doi: https : / /doi.org / 10* . 1016/j.procs.2020.04.152. [Online]. Available: https://www.sciencedirect. com/science/article/pii/S1877050920311315.