Minimizing Application Downtime During Upgrades Of Microservices coupled With Database Logic Operated In Kubernetes.

A.G.D.D.U Gunawardhana , E. Y. A. Charles & Isuru Siriwardhana (Independent Researcher)
Department of Computer Science, University of Jaffna, Sri Lanka
gunawardhanaudara@gmail.com

Introduction

Cloud computing has had such an impact on the IT industry that the majority of IT infrastructure is now shifting to the cloud. The success of cloud environments is due to the resources they enable for applications to grow easily and for free. The maintenance and updating of an application is critical. Every application must be updated on a regular basis.
When upgrading services deployed in Kubernetes settings, where the services are strongly tied with database objects, a strategy to minimize application downtime is required. In this case, the Kubernetes operator pattern is explored and used in conjunction with proxy servers and schema converters to alleviate the reshaping of service that occurs after a typical application upgrade.

Objectives

This research intends to design, implement, and evaluate an operator in order to maintain automated updates minimizing downtime and to make the system available to users at all times.

Methodology

Build an operator to minimize the application downtime performing the update deployed with microservices where tightly coupled with database and service layer. Operator watches the cluster and checks if the YAML and deployed versions are the same. If differ operator deploys the database and application without interrupting the user. The overall cluster setup is deprecated in Figure 1 and the proposed techniques are deprecated in Figure 2.

Experimental Setup

- ➢ To handle the update, First install the custom resource API(Operator) to the cluster.
- ➢ Configure the database manually. (The database size I tested 1M)
- ➢ Through the operator configure the application in the kubernetes cluster.
- ➢ Now, the operator watches the resource version and YAML version.
- ➢ GitHub : https://bit.ly/3kaGoZW

Operator Deployment & Application Setup

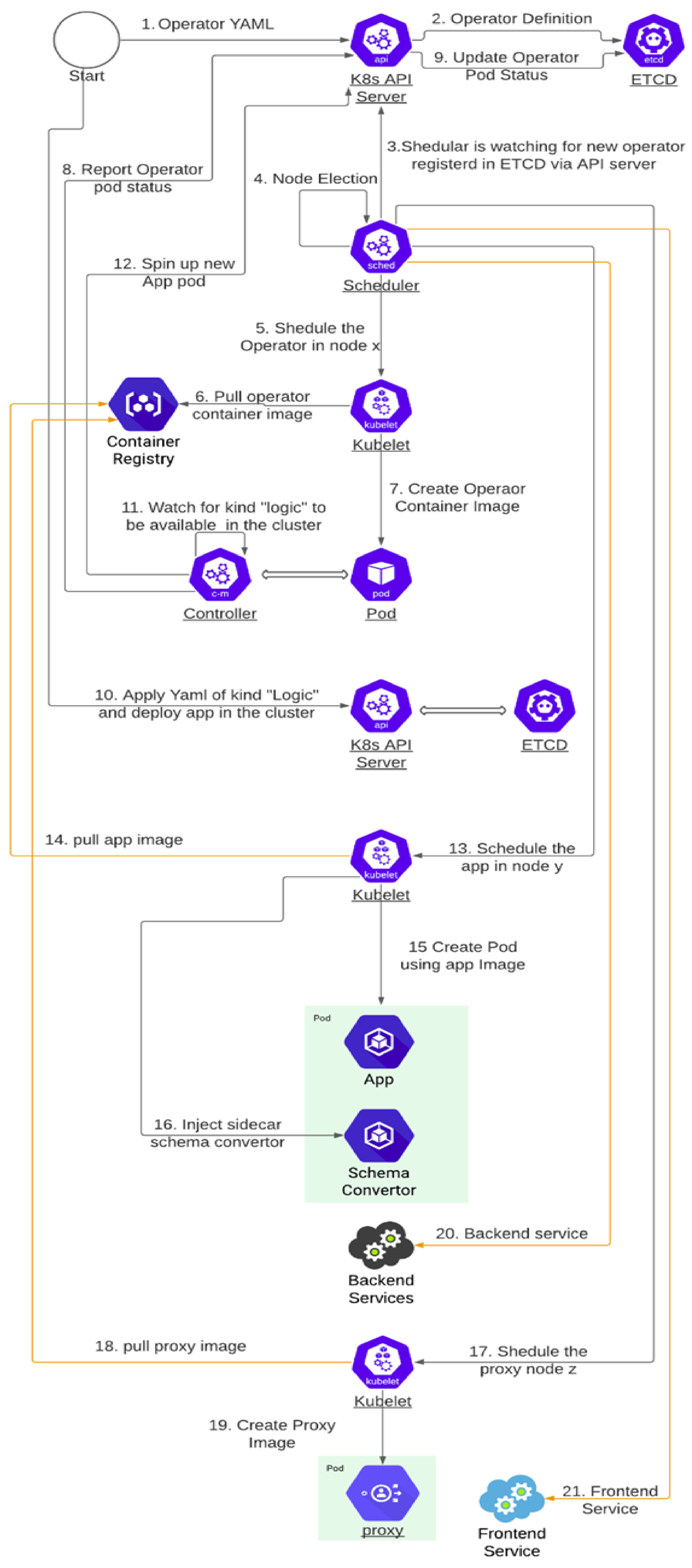
Start
1. Operator YAML
K8s API Server
2. Operator Definition
9. Update Operator Pod Status
ETCD
3.Shedular is watching for new operator registerd in ETCD via API server
8. Report Operator pod status
4. Node Election
Scheduler
12. Spin up new App pod
5. Shedule the Operator in node x
Container Registry
6. Pull operator container image
Kubelet
7. Create Operaor Container Image
11. Watch for kind "logic" to be available in the cluster
Controller
Pod
10. Apply Yaml of kind "Logic" and deploy app in the cluster
K8s API Server
ETCD
14. pull app image
13. Schedule the app in node y
Kubelet
15 Create Pod using app Image
Pod
App
16. Inject sidecar schema convertor
Schema Convertor
20. Backend service
Backend Services
18. pull proxy image
17. Schedule the proxy node z
Kubelet
19. Create Proxy Image
Pod
proxy
21. Frontend Service
Frontend Service

Figure 1

Proposed Methodology

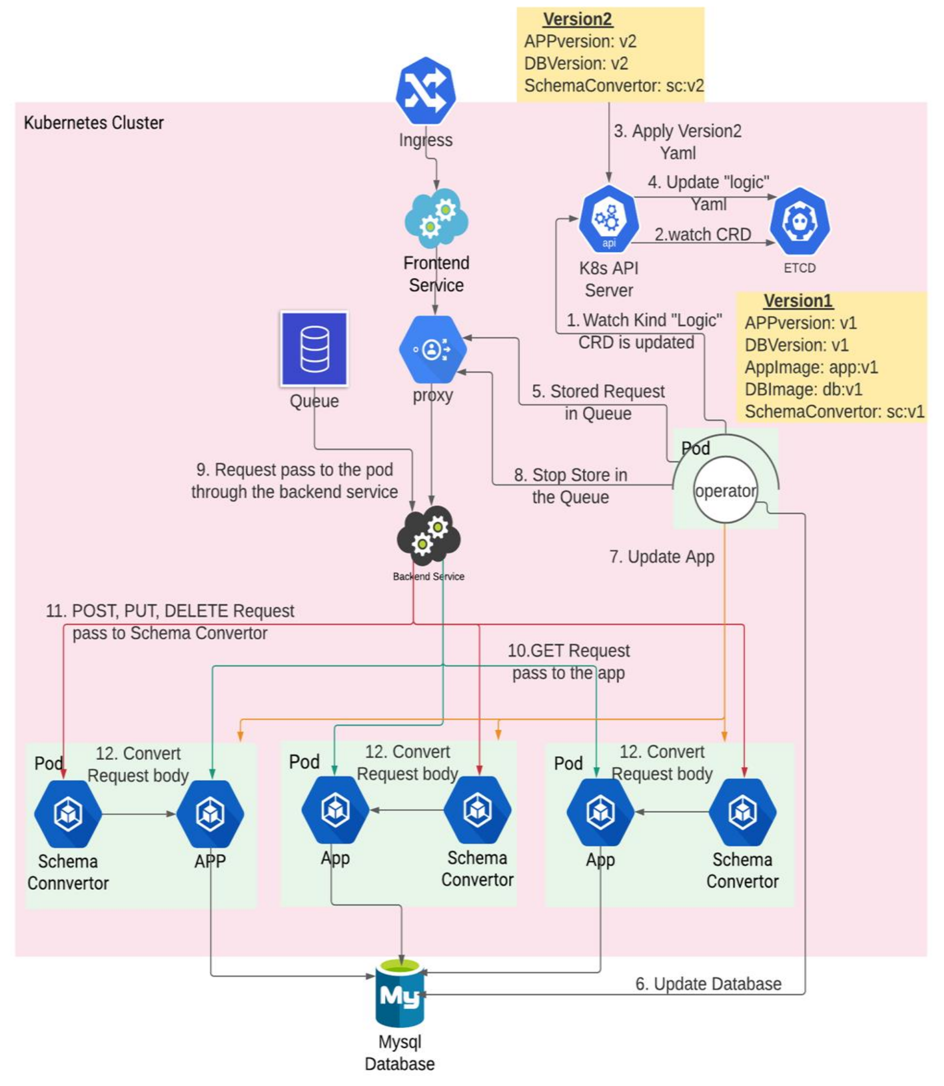

Figure 2

Testing

Testing of the system is conducted under a critical process of analyzing requirements implemented in the system, and this process will make sure that software quality is also considered in the development system. Software quality is measured under three methods.

1. Manual process. Before implementing the update, remove the current deploy version in the cluster. Next Update the database and then new version deploy in the cluster.
2. Manually update database and Rolling update the application. Here also use the manual update process. First update the database and then update the application pod according to the rolling update concept
3. Keep automatic updates using operators and handle the failure of the database and service layer.

Create new tables, rename columns, drop columns, drop tables, add columns, edit data types, and handle any database changes when upgrading the database.
The aforementioned method downtime is depicted in a pie chart(Figure3).

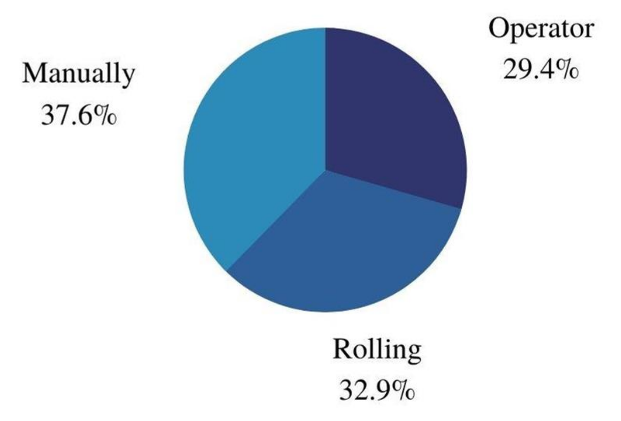

Figure 3

Conclusion

Update automatically tightly coupled with database and application service minimizing downtime using operator. Without user interruption can perform the update smoothly by the operator. This system was able to provide higher flexibility for developers to update their application in a cloud environment minimizing downtime.

References

[1] H. Saito, H.-C. C. Lee, and C.-Y. Wu, DevOps with Kubernetes accelerating software delivery with container orchestrators. Birmingham, UK: Packt Publishing, 2017.
[2] P. Mell and T. Grance, “The NIST Definition of Cloud Computing,” p. 7.
[3] Tomas Cerny,Michael J. Donahoo,Michal Trnka, Contextual Understanding of Microservice Architecture:Current and Future Directions
[4] “Kubernetes Cluster Architecture” kubernetes.io. [Online]. https://kubernetes.io/docs/concepts/architecture .
[5] “Operator SDK” operatorframework.io [Online]. https://sdk.operatorframework.io/docs/building-operators/golang/
[6] Michael de Jong , “Zero-Downtime SQL Database Schema Evolution for Continuous Deployment”
[7] David Jaramillo,Duy V Nguyen,Robert Smart, “Leveraging microservices architecture by using Docker technology”