논문 21-46-11-21 The Journal of Korean Institute of Communications and Information Sciences '21-11 Vol.46 No.11
https://doi.org/10.7840/kics.2021.46.11.1950

Kubernetes용 YAML 템플릿 관리 위한 애플리케이션

레 반끄엉◆, 유 명 식°

Application for Managing YAML Template for Kubernetes

Van-Cuong Le◆, Myungsik Yoo°

요 약

Kubernetes(K8s)는 Cloud Native Computing Foundation(CNCF)에서 개발한 선도적인 컨테이너 오케스트레이션 툴이다. K8s는 단일 클러스터에서 애플리케이션 컨테이너 운영부터 다중 클러스터를 통한 복잡한 서비스의 자동화에 이르는 일련의 기능을 제공한다. 컨테이너형 애플리케이션은 StatefulSet, Deployment 과 같은 K8s 리소스 개체를 서술해주는 K8s YAML을 통해 K8s 클러스터 내에서 운영된다. YAML을 활용하여 서비스 구현을 가속화할 수 있지만, 이를 위해서는 K8s의 속성 및 관계에 대한 지식이 요구된다. 이전 연구들에서는 K8s에 대한 YAML 파일 자동 생성에 대한 내용을 다루었지만, YAML 파일 저장, 검색 및 수정과 같은 핵심 관리 기능은 다루어지지 않았다. 이로 인하여 여러 사용자가 여러 개의 YAML 파일을 관리하는 데 어려움이 있었다. 제안된 애플리케이션은 K8s YAML 파일의 효율적인 관리를 제공하는 것을 목표로 한다. 구현 및 평가를 통하여 제안된 애플리케이션이 효율적으로 YAML 파일을 관리할 수 있음을 보여주고자 한다.

Key Words : NFV, Container, Kubernetes, YAML, Workload Resource

ABSTRACT

Kubernetes (K8s) is the leading container orchestrating tool managed and developed by Cloud Native Computing Foundation (CNCF). It provides a set of capabilities spreading from operating an application container on a single cluster to automating the maintenance of a complex service across a set of clusters. The containerized applications can be operated on K8s clusters via K8s YAML to describe K8s resource objects such as StatefulSet, Deployment, etc. One can accelerate service deployment by utilizing YAML, but it requires knowledge on K8s properties and relationships. Although previous works support generating YAML files automatically for K8s, they miss key management functions such as storing, retrieving and modifying the YAML files. It leads to difficulties in managing multiple YAML files for multiple users. The proposed application aims to provide the efficient management of K8s YAML files. By implementation and evaluation, we show that the proposed application can efficiently manage YAML files.

※ This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2021-2017-0-01633) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation).

◆ First Author : Department of Information Communication Convergence, Soongsil University, Seoul, South Korea, cuonglv@soongsil.ac.kr

° Corresponding Author : School of Electronic Engineering, Soongsil University, Seoul, South Korea, myoo@ssu.ac.kr, 정회원

논문번호 : 202107-176-D-RN, Received July 21, 2021; Revised July 31, 2021; Accepted July 31, 2021

Ⅰ. Introduction

Kubernetes (K8s) is an extensible, portable, and open-source system built originally by Google. It has been developed and maintained by Cloud Native Computing Foundation (CNCF)[1]. K8s allows for the management of containers across a cluster of nodes.

In recent years, big technology companies such as Google, Facebook, Netflix have successfully changed from monolithic architecture to microservice architecture[2,3]. The container is considered an enabler technology for microservice architecture. The K8s, which is a leading orchestration tool for containers, also supports realizing microservice architecture. Typically a microservice is deployed in a Pod that is the smallest deployable unit in Kubernetes. A complete application is a group of microservices. To provide a complete application, a group of Pods has to be managed and orchestrated. It is a challenge when deploying hundreds or thousands of applications based on microservices on a cluster. To end this, Kubernetes provides various workload resources to describe allocated resources and relationships between microservices in an application, such as ReplicaSet, DaemonSet, and StatefulSet, to manage Pods effectively. The workload resource is described and maintained with a YAML file for deploying an application in production[4]. Although many efforts [5,6] have been made to minimize the complexity of describing workload resources on YAML files and support generating the YAML file automatically, they lack of management functions such as storing, retrieving and modifying the K8s YAML files for multiple users. We propose the application that can help users to create, store, retrieve, modify and deploy their YAML files to the K8s clusters effectively.

The rest of this paper is organized as follows. Section II describes the background of resource objects of K8s and the YAML language. Details of the architecture and the workflow of the proposed application are described in Section III. Section IV shows the details of the experiment. Finally, section V concludes the paper.

Ⅱ. Background

This section provides the backgrounds of workload resource in K8s and the YAML language.

2.1 Workload Resource in Kubernetes

Instead of running a sequence of commands, K8s is a declarative system. It will execute the required actions to create and maintain the desired state over time[7]. K8s determine these states as a set of resources with configurable properties described in a YAML template file.

As mentioned earlier, Pod is the most basic deployable unit in Kubernetes. In K8s, workload resources manage the running, replication, and monitoring of pods in the K8s cluster. By using workload resources, it helps to manage sets of pods of an application considerably easier.

By default, K8s provides several types of workload resources as follows:

- ReplicaSet is used to manage a stateless application with a set of different Pods that can be replaced if needed[8]. By default, this workload resource allows number of the application replicas to be set. By changing the number of replicas in deployment, it will change the number of the application replicas in a cluster. K8s then schedule and create new containers when the number of running replicas less than the required. Conversely, when these containers more than the required, K8s will kill containers until the number of containers meets the number of replicas.
- StatefulSet is used to manage stateful applications running one or more related Pods. It maintains the uniqueness and ordering of its Pods[9]. Typically, each Pod in a StatefulSet workload resource should be mounted a persistent volume to maintain state data if any Pod is replaced.
- DaemonSet defines Pods that provide node-local facilities. When adding a new node to a cluster that matches the specification in a DaemonSet, the Kube-scheduler schedules a Pod for that DaemonSet onto the new node[10].

- Job and CronJob define tasks that run to completion and then stop. CronJobs repeats a job periodically on a given schedule[11]. A job creates a set of Pods and will continue to retry execution of the Pods until a specified number of them successfully terminate[12].

The proposed application provides the ability to generate and manage the above K8s workload resources with YAML files.

2.2 YAML Language

YAML stands for Yet Another Markup Language. It is a data serialization language that is often used for writing configuration files[13]. YAML was created specifically for common use cases such as:

- Object persistence
- Log files
- Inter-process messaging
- Configuration files
- Complex data structures
- Cross-language data sharing

YAML files are easy to implement, extensible, and read by humans. They are expressive, easily portable between programming languages. It is easy to transform between JSON and YAML format.

Building blocks of a YAML file is described as the following:

- Key-Value Pair: The basic type of entry in a YAML file
- Dictionary/Map: A more complex type of YAML file
- Arrays/Lists: Lists would have several items listed under the name of the list.

When creating a workload resource in K8s, we must describe its desired state. We can use K8s API directly to create a workload resource, but it may be quite complex in practice. Typically, we describe the desired state of a workload resource in a YAML file. Workload resources in K8s use information in a YAML template file to create pods and ensure that running pods match the desired state in the YAML template file. An example of YAML file for a K8s Deployment workload resource was shown in Figure 1.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 4
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.20.1-alpine
        resources:
          requests:
            memory: "256Mi"
            cpu: "200m"
          limits:
            memory: "1Gi"
            cpu: "500m"
        ports:
        - containerPort: 80
```

Fig. 1. The example of YAML file

In the example in Figure 1, a Deployment workload resource named nginx-deployment is created, which has four replicated Pods. The Deployment uses the spec.selector field to find which Pods to manage. The template field contains the following sub-fields: the Pod's label, the Pod template's specification.

Ⅲ. Proposed Architecture and Workflow

3.1 Proposed Architecture

The proposed application is built using many open-source projects such as the yq tool[14], React framework[15], and MongoDB[16] as shown in Figure 2.

The FrontEnd component of the application is implemented using the React framework. Users easily interact with the visual interface provided by FrontEnd to generate a YAML file that describes desired state of their application. The data described

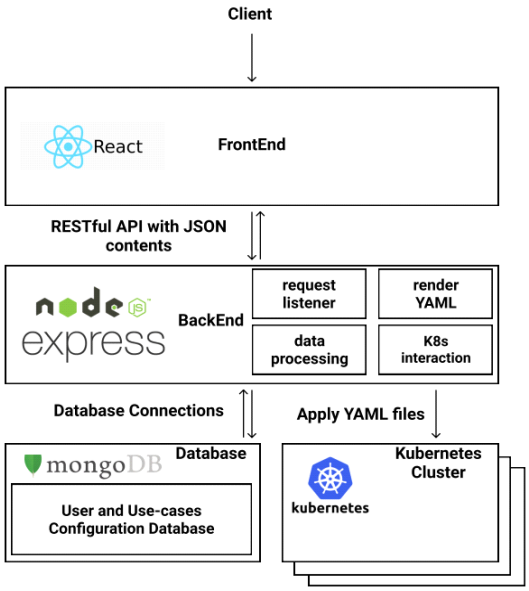

Fig. 2. The proposed architecture

by the user is sent to the BackEnd under JavaScript Object Notation (JSON) format.

The BackEnd component of the proposed is based on the Express framework to build core functions as follow:

- The request listener component waits to process the request attached data from the FrontEnd component.
- The render YAML component returns the expected result to the FrontEnd to show it.
- The data processing component manages data using MongoDB. The data is stored in JSON format. A YAML file stored in the database can be used by the data processing component to adapt quickly to an application's new deployment requirement.
- The K8s interaction component is used to verifying, applying, and destroying a K8s YAML file on a K8s cluster. By integrating kubectl, the K8s command-line tool, and many K8s configurations, the proposed application easily interacts with many K8s clusters.

We use MongoDB as a database in the proposed application. MongoDB, a non-SQL database, appropriates to store JavaScript Object Notation (JSON) data. It is used to store user YAML data for the proposed application.

3.2 Workflow of Proposed Management Application

For YAML file management, one needs to manage multiple K8s clusters for multiple customers. YAML files describe workload resources that can be used in all environments with a few modifications. The workflow of the proposed application provides a mechanism to generate, store, and reuse YAML files effectively and reliably. The details of the workflow are described in the following two parts.

3.2.1 Generating, Storing and Deploying YAML files

The workflow of generating, storing, and modifying YAML files is shown in Figure 3.

- A web graphic user interface (GUI) of the proposed application is built using React, a popular front-end framework. Users can describe a workload resource on it easily (1). Additionally,

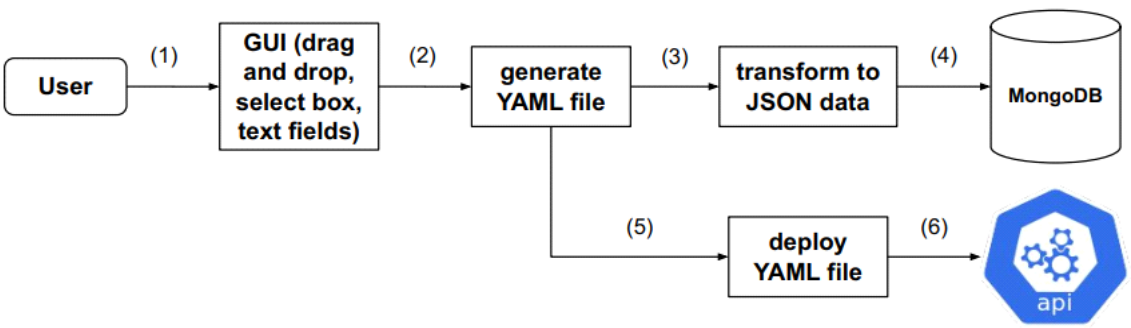

Fig. 3. The workflow of generating, storing, and deploying YAML files

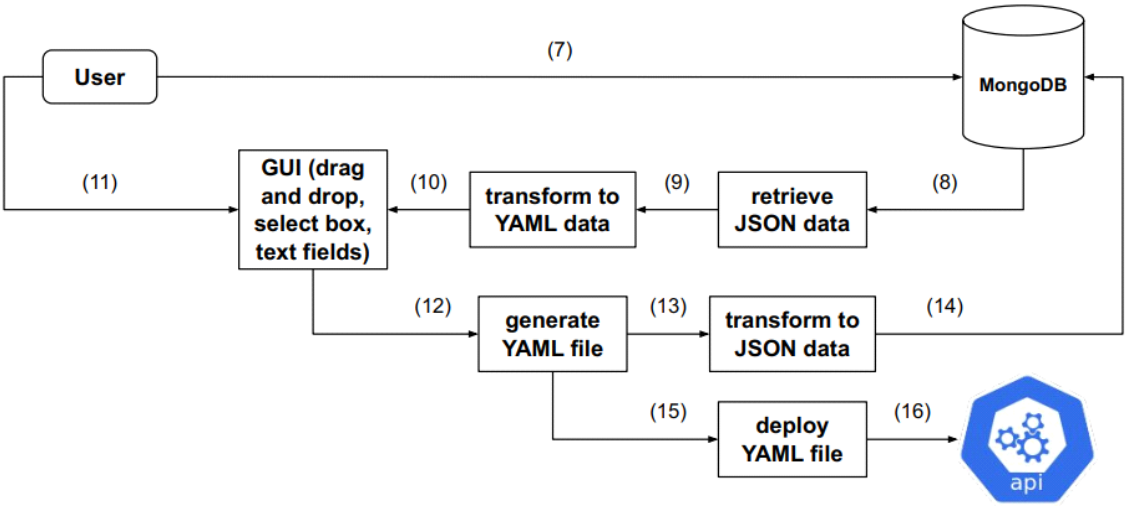

Fig. 4. The workflow of retrieving and modifying YAML files

the GUI also is responsible for visualization workload resource description in YAML format.

- Using users' data from the GUI component, the proposed application generates a YAML file using the yq tool (2). The yq tool is a smart command-line YAML tool that works with YAML files along with JSON.
- Because data in a YAML is difficult to store, workload resources are stored in JSON format into a database (3) (4).
- At the same time, the kubectl command-line tool is used to verify the constraint and syntax of the YAML file if the generated YAML file can work correctly in the K8s (5). The kubectl is a default tool provided by K8s, which helps users to control K8s clusters easily.
- The K8s interaction component deploys the generated YAML file from the GUI to a K8s cluster with the respective K8s configuration (6).

3.2.2 Retrieving and Modifying YAML files

The workflow of retrieving and modifying YAML files is shown in Figure 4.

- Users also can get the stored YAML file to create a new one with less effort (7).
- The JSON data of the stored YAML file is retrieved (8) and transform into YAML data (9).
- The YAML data then displayed on the GUI (10) so users can modify it for the new case (11).
- After the stored YAML file is modified, the proposed tool will generate a new YAML file for the new case (12).
- The new YAML file is transformed to JSON data (13) for storing in the MongoDB database (14).
- At the same time, the new YAML file also is verified the constraint and syntax according to k8s policies (15).
- Once all the above is finished, the new YAML is deployed to a cluster (16).

With the proposed tool, users can deploy the stored YAML file to a new cluster or environment with different requirements such as a number of replicas, etc. User also can easily modify their previous deployed YAML files and deploy them to a new cluster or environment in a fast way. Thus the proposed management application helps saving time and effort for users to deploy applications in new clusters and environments.

Ⅳ. Experiment

The experiment was performed on a server that has the system configuration as shown in Table 1.

The experiment environment is designed for multiple K8s clusters to evaluate the proposed

Table 1. Specifications of the Experiment

Entity	Details
Server hardware	Intel(R) Xeon(R) CPU E3-1240 V3 @ 3.40GHz, 8 cores
Operation System	Ubuntu 18.04 LTS, 64 bit
Software	qemu-kvm v4.0.0

application. First of all, we deploy three virtual machines (VMs) using the kernel-based virtual machine (KVM). After that, we deploy the Minikube, a local K8s cluster, on each virtual machine to simulate a K8s cluster. All of these components are connected in a LAN. In the experiment, we evaluate the effectiveness of the proposed application in generating and storing a deployment YAML file. The experiment also shows that we can retrieve, and modify a stored YAML file for a new context of deployment.

We created a StatefulSet workload resource named kymo-deployment. It has three replicated Pods. The label for identifying the set of Pod of this StatefulSet is named kymoexport: KymoExport. The updating strategy is RollingUpdate that allows StatefulSets' update to take place with zero downtime by incrementally updating Pods instances with new ones. The Pod management policy is defined OrderedReady that tells the StatefulSet to respect the ordering of Pods. In the template part, we describe details of each Pod of the StatefulSet. The Pod's labels are consist of "app: web" and "kymoexport: KymoExport" to filter Pods in hundreds and thousands of Pods running on the cluster. The Pod template's specification defined the name and image of the container running in the Pod and the port is opened on the Pod. The affinity field describes the node's information in which pods of the workload resource will be deployed. The weight field is to choose a Node that qualifies to schedule the Pod. The pod affinity field is to identify what pod can deploy in the Node with the Pod. Figure 5 shows a K8s YAML template for generating, storing, and deploying a StatefulSet workload resource.

The example shows the capability of the proposed

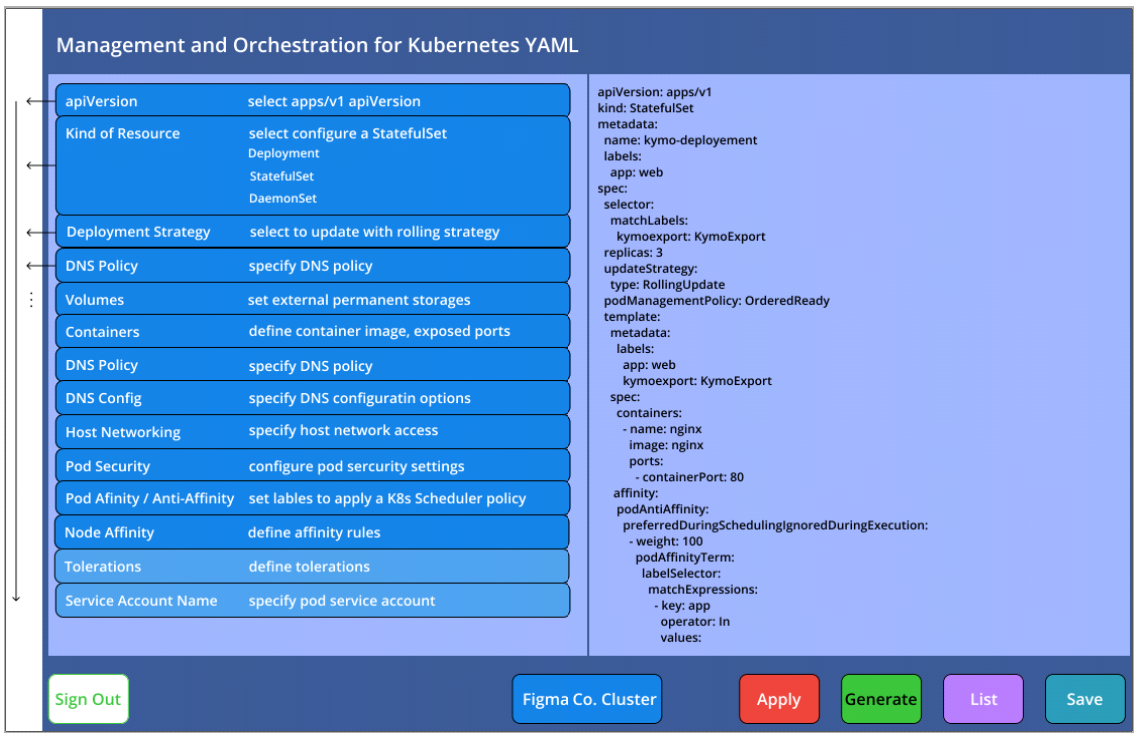

Fig. 5. K8s YAML file template for a StatefulSet workload resources

application to support generating and managing YAML files for K8s environment. The result also confirms that the YAML template generated by the proposed application can work with the existing K8s cluster.

Ⅴ. Conclusion

In practice, hundreds to thousands of containerized applications need to be effectively managed using a YAML file. Generating and managing a hundred to thousands of YAML files also is a challenge. This paper presented the way to manage the applications in the K8s cluster using the YAML files. We propose the management application tool to generate and manage the YAML files. The ability of the proposed application tool is verified through the experiments.

References

[1] *Kubernetes*, [Online] Available: https://kubernetes.io/home

[2] *Google The Site Reliability Workbook*, [Online] Available: https://sre.google/workbook/table-of-contents/

[3] Josh Evans, *Mastering Chaos - A Netflix Guide to Microservices* [Online] Available: https://www.infoq.com/presentations/netflix-chaos-microservices/

[4] *Declarative Management of Kubernetes Objects*, [Online] Available: https://kubernetes.io/docs/tasks/manage-kubernetes-objects/declarative-config/

[5] *Kubernetes YAML Generator - Powered by Octopus* [Online] Available: https://k8syaml.com/

[6] T.-H. Nguyen and M. Yoo, "A VNF descriptor generator for tacker-based NFV management and orchestration," *ICTC*, Jeju, Korea, 2018.

[7] K. Kaur, S. Garg, G. Kaddoum, S. H. Ahmed, and M. Atiquzzaman, "KEIDS: Kubernetes-based energy and interference driven scheduler for industrial IoT in edge-cloud ecosystem," *IEEE Internet of Things J.*, vol. 7, no. 5, May 2020.

[8] *Kubernetes Deployment*, [Online] Available: https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

[9] *Kubernetes StatefulSet*, [Online] Available: https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

[10] *Kubernetes DemonSet*, [Online] Available: https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/

[11] *Kubernetes CronJob*, [Online] Available: https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/

[12] *Kubernetes Job*, [Online] Available: https://kubernetes.io/docs/concepts/workloads/controllers/job/

[13] *YAML basic in Kubernetes* [Online] Available: https://www.redhat.com/en/topics/automation/what-is-yaml

[14] *The yq tool*, [Online] Available: https://mikefarah.gitbook.io/yq/

[15] *ReactJS framework*, [Online] Available: https://reactjs.org/

[16] *MongoDB*, [Online] Available: https://www.mongodb.com/

레 반끄엉 (Van-Cuong Le)

Van-Cuong Le received the B.Eng. degree in software engineering from the University of Science & Technology, University of Da Nang, Da Nang City, Vietnam, in 2018. He is currently pursuing a master's degree with Soongsil University. His research interests include Software-defined Network/Network Function Virtualization.

유 명 식 (Myungsik Yoo)

Myungsik Yoo received his B.S. and M.S. degrees in electrical engineering from Korea University, Seoul, Republic of Korea, in 1989 and 1991, and his Ph.D. in electrical engineering from State University of New York at Buffalo, New York, USA in 2000. He was a senior research engineer at Nokia Research Center, Burlington, Massachusetts. He is currently a professor in the school of electronic engineering, Soongsil University, Seoul, Republic of Korea. His research interests include visible light communications, sensor networks, Internet protocols, control, and management issues.

[ORCID:0000-0002-5578-6931]