Design on Deployment of Microservices on Container-based Cloud Platform

Jiqing Cao1,*

1Department of Information Engineering, Suzhou SIP Institute of Service Outsourcing

*Corresponding author email: George.cao@siso.edu.cn

Abstract. The resource allocation and monitoring functions provided by container technology can meet the independent deployment principle of microservices and support the rapid deployment of microservices. The deployment of microservices on container-based cloud platform can realize the automatic deployment and continuous integration of DevOps, thus improving the efficiency of application development and maintenance. This paper designs the deployment process of microservices on the cloud platform based on container, including the construction of image, several optimization schemes to reduce the size of image, image management, container orchestration and other activities, which further simplifies the deployment process and can provide useful reference for enterprise practice.

Keywords: Microservice; Container; Microservice deployment; Container deployment optimization.

1. Introduction

Different from single application, microservice is an architecture that splits large application systems from large to small scale. It is divided into several independent small service modules according to business functions. Each of them has its own data storage system, development mode and runs on its own environment, and then communicates between services through some communication protocols, such as http-based restful API [1]. The container realizes the isolation between applications by "packing" the applications. Docker is an open source container engine. Based on the Linux containers (LxC) technology, it further encapsulates some operation interfaces of the container to provide developers with management and use of the container [2]. The resource allocation and monitoring functions provided by container technology can meet the independent deployment principle of microservices and support the rapid deployment of microservices.

In the container-based cloud platform environment, microservice deployment usually adopts the mode of one service instance per host, that is, one service instance per container. In this mode, each service instance runs in a container, and the container, as a lightweight application, has extremely fast image construction and operation speed. In addition, the container cluster management tools such as Kubernetes can be used to manage the container, which further simplifies the deployment process. This paper consists of four parts. It first describes and designs the deployment process of microservices on the cloud platform based on container, then it introduces the methods of the construction of container image, especial proposes several optimization schemes to reduce the size of image in the third part, the fourth part focuses on how to manage many varies of container images in a unified way, finally, this paper discusses the container orchestration tool needs to be introduced to manage and schedule the containers during the deployment of microservices.
2. Deployment Design of Microservices

2.1. Deployment Process of Microservices
Conditioner is a lightweight, operating system level virtualization technology that allows us to run applications and their dependencies during resource isolation. The whole life cycle of the deployment of microservices on container-based cloud platform can be divided into three activities: container image building, container image management and container choreography. When the development of microservice applications are completed by the developers with the microservices-supported development framework, each microservice is packaged to be built as a container image. Then, the image is pushed to the created private image warehouse for image storage management (refer to figure. 1). For each subsequent deployment, you only need to pull the relevant image from the image warehouse. The last step is to run the container on different virtual machines or physical machine clusters by selecting container orchestration tools such as Kubernetes, configure the container, and monitor the running state of the container cluster in real time, so as to ensure the normal operation of the whole application. The following three steps are described in detail.

![Figure 1. Docker Image Creation Process.](image)

2.2. Microservice Container Image Construction
Image building is the process of generating a container image of the developed microservices that can run. The docker platform has two ways to build a new image. The first is to submit a container as a new image through the command-line mode, and the second is to write a dockerfile specifically to build a container image. The former method solidifies the container layer (a read-write file system at the top of the image with an initial value of null). Each time you modify the image, there will be another container layer for the image, and the image size will increase accordingly. In the second method, dockerfile is a configuration file in text format. Through the configuration of four parts: basic image information, maintainer information, image operation instructions, and execution instructions when the container starts, you can use the docker build command to complete the creation of the image by the server [3]. This paper used the latter one and it proved to be convenient and efficient.

2.3. Microservice Container Image Optimization
Microservice image includes microservice source code, middleware and many operating environments it depends on. In this chapter, several schemes and methods are discussed below as a complex solution. Sometimes, the scale of the image built is too large and takes up a lot of resource space, so it takes more time to deploy and pull the image every time. Therefore, this paper attempts to simplify the container image, so as to reduce the container construction time, start-up time, and improve the deployment efficiency of microservices. The following are two optimization schemes to reduce the image size:

---

2
2.3.1. Using Lightweight Image Tools. At present, there are many image tools, which have great differences in the efficiency of creating images. Alpine is a lightweight image distribution tool for Linux. The basic image size created by Alpine is only about 4-5M. It supports many languages and frameworks to create the basic image. For example, Java's Spring boot applications include open JDK: 8-jdk-alpine, open JDK: 8-jre-alpine and other tools. According to the author's test, the image size produced by a microservice using JRE Alpine is twice that of JDK alpine, which is more than 100 megabytes. So, using the right lightweight mirror tool is a very important optimization method.

2.3.2. Optimizing the Integrated Dockerfile. Each instruction of dockerfile will generate a file layer. We can combine instructions by some methods, such as operators, to maximize the use of cache to reduce the size of image. Through experiments, it is found that the image size can be significantly reduced by this method. [4]

2.4. Microservice Container Image Management

Generally, an application system includes multiple microservices, that is, there are multiple container images, so it is necessary to manage these container images in a unified way. Many tools and frameworks have this management function. For example, Dockerhub is the image management tool of docker official authentication, which has the management functions of image authentication, workgroup structure, workflow tool and build trigger [5]. When the container of a microservice application is created, you can first create a corresponding private warehouse on the Dockerhub, and then push all the container images built by the microservice under the application to the warehouse for storage and management. In the subsequent development process, it is convenient to pull the corresponding image to build a new development and test environment.

2.5. Orchestration of Microservice Containers

Applications typically consist of individually containerized components and must be organized sequentially at the network level to enable them to run as planned. The process of organizing multiple containers in this way is called container choreography. When the image is instantiated as a container, the microservice will run in the newly created container process. Because an application system includes multiple microservice containers, the start sequence of containers, the communication between running containers, and the running state of container instances need to be configured and managed. If these tasks are manually configured and operated, the workload and complexity of deployment personnel and maintenance personnel will be greatly increased. Therefore, the container orchestration tool needs to be introduced to manage and schedule the containers during the deployment of microservices.

At present, Docker Compose, Docker Swarm and Kubernetes are the most popular orchestration tools. Among them, docker-compose needs to write a file, docker-compose.yml, which defines the service of the application, declares the container to be started, configures some parameters, and then runs the docker-compose instruction. However, docker-compose can only manage dockers on the current host, and cannot start docker containers on other hosts [6]. At present, many application services are multi host clusters, and Docker Compose cannot adapt to this situation.

Kubernetes is an open-source project of Google. Based on its large-scale container management technology for many years, Kubernetes has complete cluster management capabilities, including transparent service registration and discovery mechanism, scalable resource automatic scheduling mechanism, strong fault discovery and self-healing capability, multi-granularity resource management capability, and load balancing, covering the several activities as development, deployment, testing, operation and maintenance monitoring [7]. It can realize large-scale, distributed and highly available docker cluster.

Therefore, it is important to select the appropriate container orchestration tools according to the situation of the application system, so that the microservice instances can run normally in their own containers and maintain good interaction, and the whole application system can run normally and complete the deployment of microservices on the container based cloud platform.
3. Conclusion

Container based cloud platform is an ideal deployment environment for microservices. By using container technology, it will be easier to deploy application systems with multiple microservices. For large-scale application systems, one-click deployment can be realized by using appropriate orchestration tools. During the whole development and operation and maintenance phase, the container-based cloud platform is the best practice of DevOps. By means of automatic deployment and continuous integration, integration cost and bug rate can be effectively reduced. Container deployment microservices have played an important role in enterprise development, deployment and maintenance of applications.

Acknowledgments

In this paper, the research was sponsored by the Service Outsourcing Foundation of Suzhou Industrial Park Institute of Service Outsourcing (Project No. KY-XJ803).

References

[1] YUY.A Microservice-based Reference Architecture Model in the Context of Enterprise Architecture [A]. Proceedings of 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC2016) [C]. IEEE Beijing Section, Global Union Academy of Science and Technology, Chongqing Global Union Academy of Science and Technology, 2016:5.

[2] Yang Baohua, Dai Yujian, Cao Yalun. Docker Introduction and practice of Technology [M]. Beijing: Mechanical Industry Press, 2014.

[3] Redhat. Introduction to control groups [EB/OL]. https://access.Red-hat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Resource_Management_Guide/ch01.html, 2016.

[4] Fu Linlin, Zou Suwen. Research on Container Deployment of Microservices [J]. Computing Technology and Automation, 2020(04)

[5] Li Hongjian. Application Analysis of Microservice Architecture and Container Technology[J]. Wireless Internet technology, 2018, 15(08):134-135.

[6] Zhou Li. Spring Cloud and Docker Microservice Architecture [M]. Beijing: Electronic Industry Press, 2017.

[7] MARKUSL. Docker Compose for the Simple Deployment of an Integrated Drug Target Screening Platform[J]. Journal of Integrative Bioinformatics, 2017, 14(2):98-102.