An Automated Deployment Engine Based on a PaaS Cloud Platform of the Micro-service Architecture System

Chen Zhong[1], Xin Yuan[2] and Jun Yang[3]
[1]*Department of Computer Science and Technology, Tsinghua University, Beijing 100083*
[2]*North China Institute of Computing Technology, Foundation 4, Beijing 100083*
[3]*Department of Electronic Engineering, Tsinghua University 100083*

Abstract. In view of the deployment management problems caused by the service atomization, architecture complexity and large-scale cluster of micro-service architecture system, a PaaS cloud platform automation deployment engine is designed and implemented based on the Docke PaaS platform so as to provide a simple, flexible, efficient, full-stack, full-process deployment solution to the micro service architecture system.

1 Introduction

In order to cope with the changing needs of users and the rapid growth of user size, it is needed for information systems to have the ability of rapid deployment, reliable operation and flexible expansion. With the development of cloud computing technology, the idea of "de-centralization" is adopted in micro-service architecture, which advocates dividing a single application into a group of small services. The services coordinate and cooperate with each other to solve the rapid changes that traditional single-block architecture can not adapt to.

Micro-service architecture distributes system functions among discrete services, which are more atomic and less autonomous. High-density deployment is adopted to support on-demand expansion. However, while bringing benefits, micro-services are becoming more and more difficult to deploy and manage due to their complex architecture and large-scale cluster. The emergence of Docker-based PaaS platform provides the possibility to build a simple, flexible, efficient, full-stack, full-process micro-service architecture system deployment solution to solve the above problems.This paper proposes a PaaS cloud platform deployment engine for the micro-services architecture system to solve the deployment problems of complex micro-services architecture systems in an automated manner.

2 Analysis of System Deployment Architecture for Micro-service Architecture

2.1 System Deployment Architecture of the Micro-service Architecture

Traditional application systems generally divide the C/S architecture or the B/S architecture. Regardless of the C/S architecture or the B/S architecture, in order to ensure the reliability and performance of the application system, the service is generally deployed on multiple servers and these servers are built as service clusters. Access requests on all clients must pass through the load balancing device to reach the service cluster. The services of application system generally use databases and files as their data support. Also for reliability and performance considerations, database servers and file servers are also built into clusters to provide data support for services, but the data itself is generally not directly stored on the database servers and file servers, it is stored on a dedicated centralized storage device.

At present, the cloud platform-based application system deployment is divided into client, business service, data, resource and platform services. The deployment mode is shown in Diagram 1.

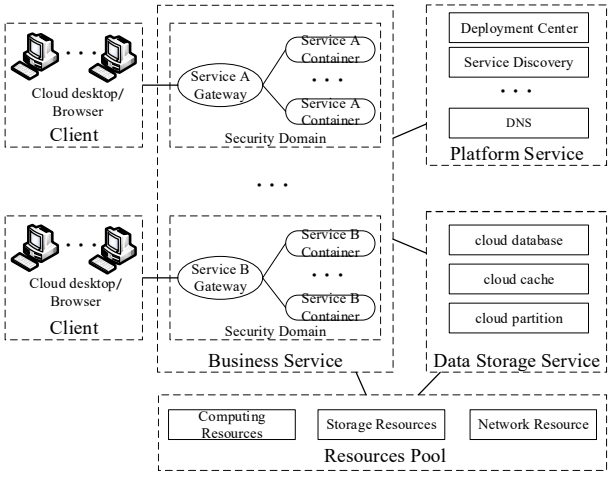

Figure 1. The cloud system-based application system deployment mode

The cloud platform-based application system is characterized by following a micro-service architecture and splitting logically complex business services into a large number of business-simple micro-services and data storage objects; the application system is defined by describing the dependencies between these micro-services and data storage objects. All micro-services that make up the application are deployed on the compute nodes in the form of service containers. For reliability and performance consideration, each micro-service needs to be registered with the service gateway, and one or more service containers are mounted under the service gateway. Cloud partitions, cloud caches, and cloud databases supported by data storage services provide data support for business services. The cloud platform virtualizes resources such as computing, storage, and network into resource pools, and provides support for the deployment of business services and the creation of data storage objects such as cloud partitions, cloud caches, and cloud databases.The cloud platform also provides a series of platform services, such as deployment center, service discovery, etc. to provide unified deployment and service calls for business systems.

2.2 Dependencies of Micro-service Architecture System Deployment

Micro-service architecture system deployment needs to deal with three kinds of dependencies: software dependencies, topology dependencies, and resource dependencies.

(1) Software dependencies

The software of the PaaS cloud platform node usually consists of multiple layers of software stacks, and the installation of the upper layer software depends on the installation of the underlying software. The entire software stacks can be divided into four layers from bottom to top: the operating system, the platform software, the technical framework, and the application. The operating system is usually a Linux-like operating system; the platform software is usually a database, a web server, a cache service, etc.; the technical framework is attached to the platform software, providing a framework or a class library that provides feature language-related software functions, such as SSH (Struts) +Spring+Hibernate; the application is a program or software grouping that implements business logic and provides business-related functionality.

(2) Topology dependencies

Large-scale complex information systems based on micro-service architecture often involve multiple clusters. Each cluster is composed of several nodes or sub-clusters. The nodes communicate with each other through the server interface. The system deployment needs to establish the topology between system services so as to deal with the relationship between nodes and nodes, nodes and clusters, and clusters and clusters. The relationship between a node and a node is generally a communication relationship. The source node establishes a communication connection with the target node through the service port. When the platform is deployed, the service port (such as IP, port, URL, etc.) of the target node needs to be deployed on the source node, and ensure the Layer 3 link between the source node and the target node is accessible. The relationship between a node and a cluster is usually a dependency, including network dependencies and node dependencies. Network dependencies refer to the node of the cluster and the cluster belong to the same subnet. The node dependencies refer to the node of the cluster is managed or controlled by a certain master node of the cluster. The cluster needs to be accordingly deployed according to the dependencies relationship between the master node of the network and the cluster. At the same time, the nodes of the entire cluster are batch-operated through the dependencies, such as batch deployment and batch restart.

(3) Resource dependencies

Cloud platforms are typically built on shared cloud resource pools, and application deployments need to estimate resource requirements for the platform. Virtual machines, virtual storage, and virtual network resources with the appropriate specifications and quotas are applied for from the cloud resource pools based on resource requirements. For the PaaS platform, the

deployment of the micro-services architecture system needs to estimate and apply for certain platform service resources such as cloud databases, application server clusters, and cloud caches from the cloud resource pools.

3 Requirement Analysis and Architecture Design of Automated Deployment Engine for Micro-service Architecture System

3.1 Deployment Requirements of Micro-service Architecture System

The automated deployment of the micro-services architecture system is to solve the problem of system dependencies as automatically as possible during the deployment process. The deployment of the engine automatically completes the deployment and distribution of the micro-services system, and realizes the allocation, installation and deployment of various resources, sub-cluster, nodes, and software so that the application package submitted by the developers is loaded onto the application node of the PaaS platform, and finally the cluster of the micro-service system runs accurately.

(1) Automatically processing software dependencies of container nodes

The installation, deployment and startup of the software of each layer in the entire software stack can be automatically completed according to the hierarchical structure of the software stack, and the installation steps of the same or similar hierarchical software stack can be reused.

(2) Automatically processing topology dependencies of micro-service system

It is convenient to establish a topology model of the micro-service architecture system, describe the affiliation, the communication relationship and the dependency relationship among the cluster/sub-cluster, the nodes of the platform, and recursively complete the deployment of the entire cluster according to the affiliation and the dependency relationship, automatically complete related parameter deployment of nodes and networks according to the communication relationship and reuse cluster topology and deployment process with the same structure.

(3) Automatically processing cloud resource dependencies

The computing, storage, and network resource dependencies in the cloud platform resource pools can be automatically completed according to resource requirements, including resource dependencies such as container clusters, cloud partitions, and distributed file systems, and the creation of basic resources and related parameters deployment are completed according to the dependency relationship. The creation and deployment process of cloud resources with the same requirements can be reused.

(4) Improving the deployment efficiency of the micro-service system on the PaaS cloud platform

It is needed to reduce the workload of manual deployment, compress the software stack of the nodes to improve the deployment efficiency of a single node, increase the parallelism of platform node deployment, and improve the deployment efficiency of the entire micro-service architecture system cluster.

3.2 Automated Deployment Engine Design of the Micro-service Architecture System

The automated deployment engine of the micro-services architecture system is built based on open source Docker container platform and scheduling system, distributed file system, cloud cache, cloud database and other cloud infrastructures, including PaaS platform management portal, PaaS automated deployment engine, container cluster, cloud warehouse, container cluster, deployment center and other templates, and the system architecture is shown in Diagram 2. Among them, the cloud infrastructure service provides application cluster computing, application, configuration, scheduling and destruction services of storage and network resources in the form of API. The cloud repository stores container images such as the operating system, basic software, technical framework, and application software required for the application cluster. The deployment center stores various deployment parameters, domain names, port numbers, and data source addresses required for deployment. The PaaS automated deployment engine provides the core functions of automated deployment such as micro-services architecture system deployment model resolution, deployment process implementation, resource control, and elastic scheduling. The container cluster node implements the loading and running of the application software stack by pre-installing the Docker engine. The PaaS automated deployment engine implements the deployment process of the application cluster by scheduling the process engine and the process orchestration framework, and calls the management interface of the cloud infrastructure through the management agent to complete the deployment of the entire container cluster.

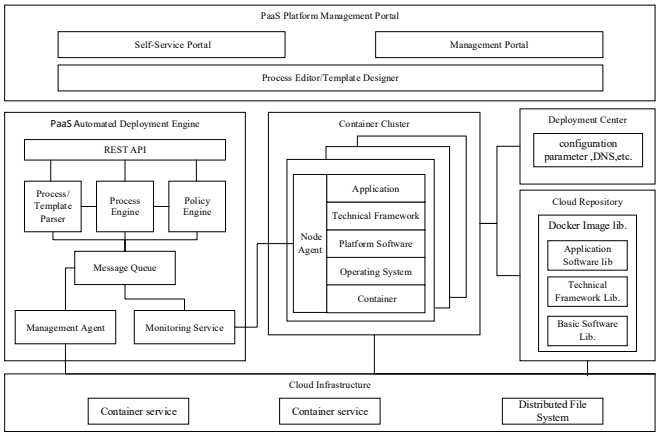

Figure 2. Architecture of PaaS cloud platform deployment engine

4 System Deployment Process Based on Automated Deployment Engine

4.1 Design of Deployment Template

The application system of micro-services architecture is composed of business service micro-services. Therefore, both the application system and the business micro-services need to define deployment templates. The business micro-service deployment template is shown in Diagram 3.

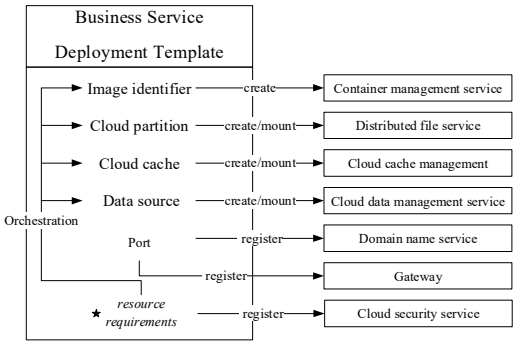

Figure 3. Business micro-service deployment template

Business micro-service deployment templates include statement for image labels, cloud partitions, cloud caches, data sources, ports, and resource requirements. Among them, the resource requirement is the only dynamic item in the business service template that needs to be temporarily specified by the users during deployment. The other items are static items whose content has been determined during the process of deploying the template, but the content of some static items is allowed to be arranged and adjusted according to the content of the resource requirements item at the time of deployment. The resource requirements item states the deployment device architecture option set and service features that the users expect, and its content is used as the allocation basis of other items except the port during deployment. The schema type of the business service adaptation and the image identifier corresponding to the architecture type are declared in the image label item. The deployment computing device architecture type, the amount of computing resources, and the image ID corresponding to the deployed computing device architecture type are determined by referring to the resource requirements during deployment, and are used in the process of creating a container for calling the container management service. The cloud partition item declares all the cloud partition paths used by the business service. These paths will be built on the same cloud partition and used in the process of calling the distributed file service to create/mount the cloud partition. If the item is empty, indicating the business service does not need to use the cloud partition. The cloud cache item declares the computing resources needed by the cloud cache in the business service, such as the number of CPU cores and memory capacity, and is used in the process of calling the cloud cache management service to create/mount the cloud cache. If the item is empty, indicating the business service does not need to use the cloud cache. The data source item declares all the data source information used by the business service, and is used in the process of calling the cloud database management service to create/mount the cloud database. If the item is empty, indicating the business service does not need to use the cloud database. The port item declares all service port information for the business service to be developed externally, and is used when calling the gateway to register the service cluster and calling the domain name service to register the domain name.

The application system deployment template based on the micro-service architecture is shown in Diagram 4.

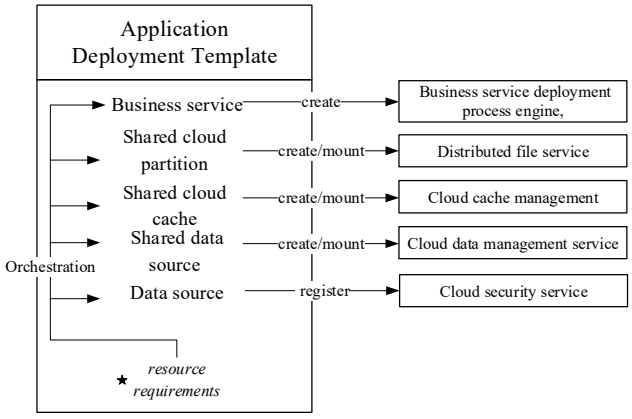

Figure 4. Application system deployment template based on the micro-service architecture

Application deployment templates include claims for business services, shared cloud partitions, shared cloud

caches, shared data sources, security domains, and resource requirements. The resource requirement is the only dynamic item in the business service template that needs to be temporarily specified by the user during deployment. The other items are static items whose content has been determined during the process of deploying the template, but the content is allowed to be allocated and adjusted during deployment according to the content of the resource requirements item. The resource requirements item declares the deployment device architecture option set, service characteristics, isolation requirements, etc. that the user expects, and its content is used as the allocation basis of other items during deployment. The business service item declares the identifier and version number of all business services of the application systems. The identifier and version number can uniquely identify a business service deployment template. When deploying, the deployment resource requirements of each business service will be finalized with reference to the resource requirements. The shared cloud partition item declares the shared cloud partition group used by multiple business services in the application system, each group represents a shared cloud partition, and the business service identifier in each group represents the business service using the shared cloud partition. If the item is empty, indicating the application system has no shared cloud partition. The shared cloud cache item declares a shared cloud cache packet used by multiple business services in the application system, each group represents a shared cloud cache, and the business service identifier in each group represents a business service using the shared cloud cache. If the item is empty, indicating the application system has no shared cloud cache. The shared data source item declares a shared cloud database grouping used by multiple business services in the application system, each group represents a shared cloud database, and the business service identifier and the JNDI name in each group collectively represent a JNDI used by the business service under the shared cloud database. If the item is empty, indicating the application system has no shared cloud database.

4.2 Process of system deployment

A deployment on the PaaS cloud platform generally requires some major steps such as allocating resources, creating data storage objects, deploying service containers, registering service clusters, and registering domain names. The specific deployment process is shown in Diagram 5.

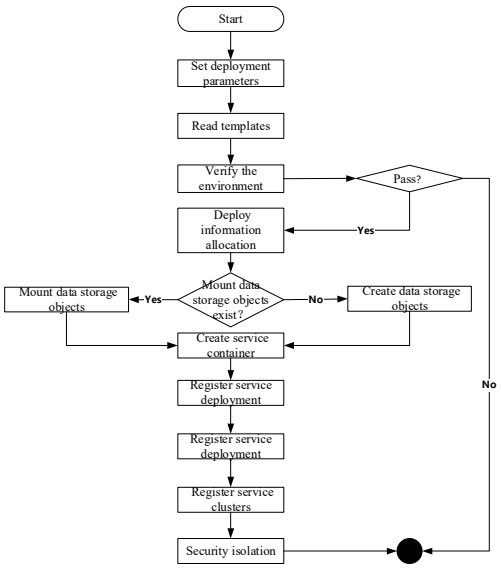

Figure 5. Process of system deployment

Firstly, the user sets the deployment parameters, and the cloud platform reads the corresponding deployment template from the cloud repository, performs an environment verification for this deployment based on deployment parameters, statements in the deployment template, and current computing resources and data storage objects. If the environment verification passes, the deployment process continues, otherwise the deployment process is aborted; secondly, according to the result of the environment verification and the deployment parameters, the deployment information in the template is allocated, and if the allocated data storage object does not exist, it is created, otherwise it is mounted; thirdly, the user applies for resources from the resource scheduling service, creates a service container, and registers related service deployment information, service clusters, and domain names. Finally, security domains and isolation measures are set to complete security isolation.

According to the general steps of the above deployment, the deployment of the micro-service deployment engine is performed as shown in Diagram. 6.

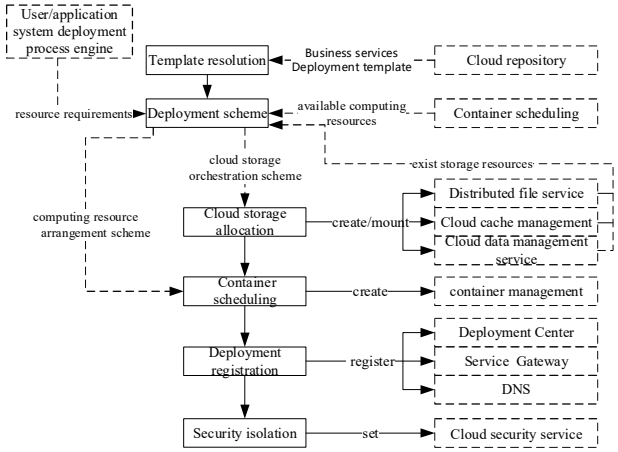

Figure 6. Deployment process of micro-services

Firstly, the micro-service deployment engine is limited to read the business service deployment template from the cloud repository and parse it, and submit the parsed deployment data items to the resource orchestration template, and at the same time, the business service computing resource requirements submitted by the user or the application system deployment engine are also submitted to the resource allocation template; Secondly, the resource orchestration template queries the currently available computing resources from resource scheduling service, and generates the business service computing resource allocation scheme for this deployment according to the adapted computing resource architecture type declared in the deployment template, the service characteristics declared in the computing resource requirements, the computing resources amount, and the currently available computing resources. The resource allocation template also obtains the currently existing cloud storage object from the cloud storage management service, and generates the cloud storage orchestration scheme of the current deployment according to the mounted cloud storage identifier declared in the computing resource requirements; The cloud storage orchestration scheme is submitted to the cloud storage allocation template to invoke the corresponding cloud storage management service and create a cloud storage object or a cloud storage object; the business service computing resource allocation scheme is submitted to the container scheduling template to create a service container; then, the deployed registration template registers the service container deployment information, the service cluster information, and the domain name information to related deployment services (for example, deployment center, gateway, domain name services, etc.); finally, the security isolation template calls the cloud security service to set up isolation measures according to the deployed security requirements.

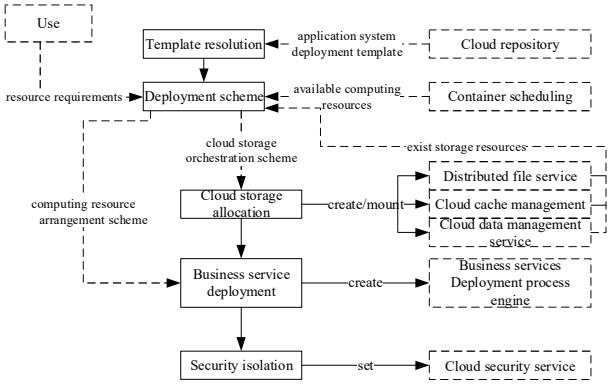

Figure 7. Deployment process of application system

The deployment of the application system is shown in Diagram 7, and the application system deployment is triggered by the user. The process of system deployment is similar to the process of micro-service deployment. After the cloud storage resources are allocated to the system, the business service deployment engine is invoked to deploy the business micro-services.

5 Conclusions

The system automated deployment engine of the PaaS cloud platform built based on Docker container and micro-service architecture technology achieves a good balance between the deployment efficiency and deployment flexibility of the micro-service architecture system. The software stack supported by any node through the Docker container, the mode deployed through the customized micro-service architecture system supports the complex cluster architecture, supports the full-stack, full-process automation deployment of large-scale complex micro-service systems, and supports the PaaS cloud platform to better respond to the changes in user demands and rapid growth in user scale.

References

1. Zhen-wei He, Li-yun Yan, Hui-yun Li, Lin Zhang, Gang Lu. An automatic deployment plan for Internet service platform based on open source PaaS Technology, Telecommunications Science 2015, 31(10)
2. Miglierina M. Application deployment and management in the cloud. Proceedings of the 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, 2014
3. Hong-Liang Sun. Docker source code analysis (1): Docker architecture.http://www.infoq.com/cn/articles/docker-source-code-analysis-part1, 2014 Sun H L.Docker source code analysis(I):Docker

architecture.http://www.infoq.com/cn/articles/docker-source-codeanalysis-part1, 2014
4. Cloudify 3.2 architecture overview. http://getcloudify.org/guide/3.2/overview-architecture.html, 2014 2014
5. Peng Xu, Si Chen, Sen Su. Internet application PaaS platform architecture, Journal of Beijing University of Posts and Telecommunications 2012(01)
6. Zhang H R. Design and implementation of the automatic deployment module for cloud platform(master dissertation). Harbin: Harbin Institute of Technology, 2013