Research on the model of SaaS platform based on Docker container technology

Weiwei Zhang[a*], Lin Zhao[b], Lin Bi[a], Haibo Zhang[a]
[a] Drug Research and Development Center, Shandong Drug and Food Vocational College, Weihai 264200, China; [b] Dalian Neusoft University of Information school of software, Dalian 116000, China

ABSTRACT

Although the rapid development of information technology makes rooms for many small and medium-sized enterprises on program deployment through virtual machines (VMs), the physical resources is not fully occupied since some programs are not as large as the virtual systems. Therefore we applied the Docker container technology, which is an application level isolation. It is more flexible and smaller than virtual machines. This paper studies the standardized migration, automatic deployment and application cluster monitoring of Docker container technology and applies them to the Saas platform. It provides visual operation function for enterprise project management and reduces the steps of system deployment, operation and maintenance.

Keywords: Docker container technology, SaaS, virtual machine

1. INTRODUCTION

Due to the rapid development of information technology, in order to solve the problem of low utilization of physical resources caused by virtual machine deployment, many Internet companies are making efforts towards container technology. Under the particularly prominent influence of Docker container technology, traditional deployment cannot be compared with Docker container technology in terms of easy accessibility, operability, resource utilization, or performance problems. There are already many systems similar to Docker container technology visualization operation in the same category, but most of them simply solve the problem of visualization operation. The development goal of the Docker container system SaaS platform is to provide small enterprises and individuals with visual operation functions based on Docker container technology as the core[1]. In addition, it also provides single tenant and multi tenant SaaS modes, so that both individual users and small and medium-sized enterprises can use it, and it also provides a strict permission mode[2].

This paper studies the standardized migration, automatic deployment and application cluster monitoring of Docker container technology and applies them to the Saas platform. It provides visual operation function for enterprise project management. In terms of container management, you can manage, view and modify different container instances, and use DockerHub image and local upload image functions to upload and package your own images, and monitor the container running instances. In terms of permission management, multi tenant and multi container instances can be provided without affecting each other. Through the study of this model, the tedious steps of developer deployment, operation and maintenance are reduced, and Docker container technology is used for standardized migration, unified parameter configuration, automatic deployment, application cluster monitoring, etc.[3].

2. MODEL REQUIREMENT ANALYSIS

After preliminary analysis, this model is applicable to all software developers, especially small enterprise companies. Its core is to solve the use scenario of multi tenant mode for small enterprise companies[4]. The development goal of using the Docker container system SaaS platform is to provide small and medium-sized enterprises with visual operations based on Docker container technology[5].

2.1 Business requirements

An image can create multiple containers. A project contains multiple containers. A container instance can only mount

[*] vivian2088@163.com

one container volume. A service can have multiple user container instances. A project can contain multiple images. A project can have multiple container volumes mounted. The domain model is shown in Figure 1.

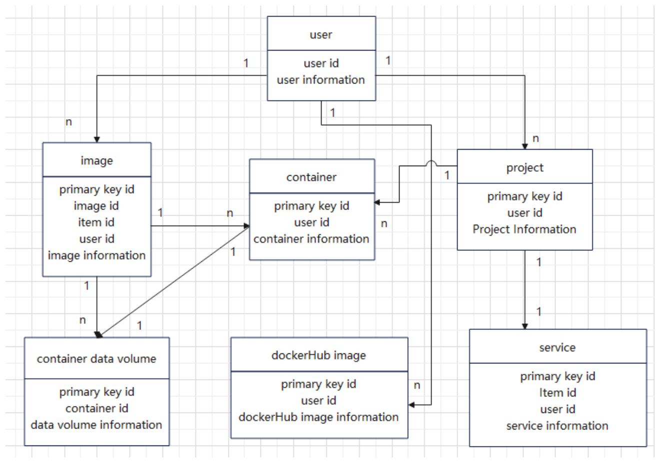

Figure 1. Domain model.

2.2 Functional requirements

All functions of the model are placed in the following high-level use case diagram, which is divided into two categories: all permission functions of ordinary users and all permission functions of super administrators. The high-level use case diagram of the project is shown in Figure 2.

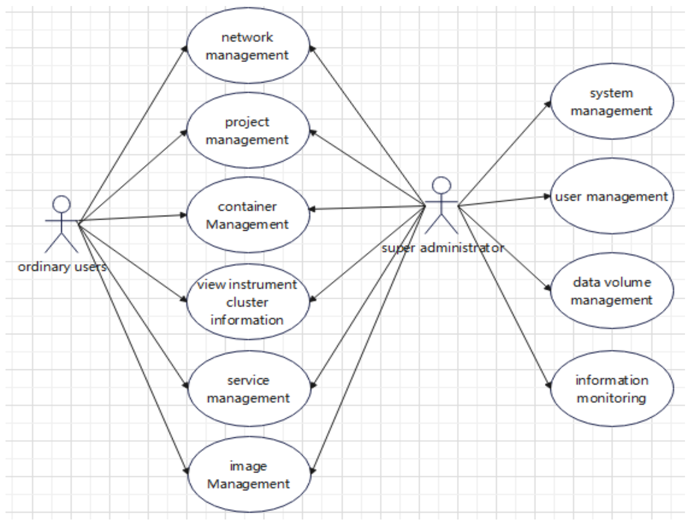

Figure 2. High level use case diagram.

The use case summary is described in Table 1.

2.3 Non functional requirements

Correctness: Business errors are not allowed. Robustness: In case of system failure, the user's login data will not be lost. The system will use the RDB snapshot mode in Redis to ensure data backup of the user's data and password[6]. Performance: The system processing time shall not exceed 2 seconds under normal traffic conditions, and the latest time under busy traffic conditions shall not exceed 5 seconds. Security: user password information can be encrypted with MD5 to prevent user information from being disclosed. Compatibility: can run Linux kernel 3.10.0-1062.1.2.el7.x86_ 64 and above.

Table 1. Use case summary description.

Case	Abstract description
View instrument cluster information	You can view the utilization of all containers, projects, and images by viewing the dashboard information.
View third-party monitoring information	The user can see the running status information of the container running instance.
Manage user information	The administrator can view all ordinary user information, freeze the ordinary user account, view the data of the container that the user runs, and check whether the Token information logged in by each user is expired.
Manage project information	The administrator can query the information details and project log information of a project through time, user name and project name.
Manage Container Information	The administrator can query specific containers by container name and container status, start, pause, delete, restart, kill the container, and view the monitoring information of the running status of the container.
Manage service information	The administrator can query the list information of the service and enter the specific service.
Manage image information	Administrators can view local public images, images uploaded by local users, and images uploaded to DockerHub. You can query the specific user image through the image name, and you can select to delete the image and export it to a local file. You can also import from a local image file to a local user image. You can also pull the image you want to use from DockerHub to the local image library.
Manage network information	Administrators can view several kinds of network information in Docker, including public network and non-public network information, and can also customize and create network information, and can keep the network information synchronized with the server.
Manage data volume information	The administrator can view the mount information of the container volumes of all containers mounted on the container, and can clean up the useless data volume information of the stopped container.
Management system information	The administrator can view the running information log of the entire SaaS service and the operation log information of some incorrect operations.

3. MODEL FRAME DESIGN

3.1 Technology roadmap

The core key technology of this model in the implementation process is based on Docker container technology. This technology can break the limitations of traditional deployment projects and running projects, and it is more convenient and simple to complete the project[7]. You only need to package the project into an image, and it can run everywhere on the server, with high startup efficiency, easy maintenance and iteration.

Docker is an open source application container engine. Just let developers package their applications and dependency packages into a portable container, and then publish them to any mainstream Linux machine to achieve virtualization.

3.2 Model frame design

The system client is implemented as a SPA single page Web application. It uses AJAX requests to interact with the server based on HTTP protocol. The page uses Vue+JQuery+JavaScript to build a componentized page[8]. The Vue component is used to split the interface module. The variable part is split into separate subcomponents. Each module has a common component to reduce the development effort. The parent component uses different subcomponents in different scenarios through page logic control, Display the user interfaces under different logics. Because Vue is only responsible for processing part of the UI logic and does not provide a communication scheme with the server, it uses the AJAX implementation of JQuery or the popular Fetch to handle the communication with the server. The second level data is required in the data monitoring interface, so the mainstream WebSocket communication technology is used to ensure that the monitoring data of the container is updated in real time. The client system architecture is shown in Figure 3.

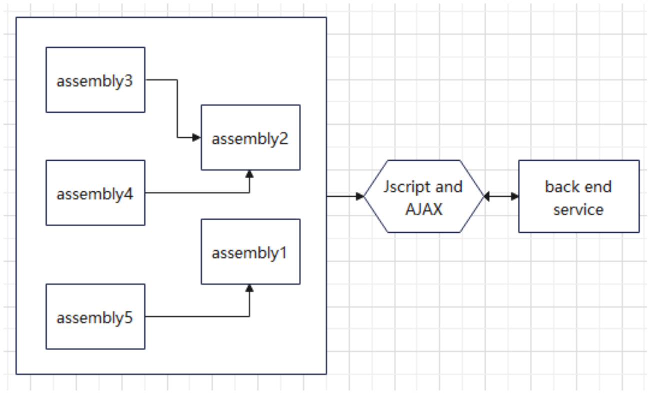

Figure 3. Client system architecture diagram.

The server side system architecture is mainly divided into four layers to realize the division of different responsibilities, ensure that each layer has a single function, and follow the principle of unity[9]. The display layer is mainly used by the front end to show users the requirements for completing user functions. The control layer is responsible for distributing and processing client requests, verifying the Tokens of user identity information, permission control, data type conversion, view analysis, session management, and business logic layer is responsible for handling specific business logic, The data access layer is mainly used to provide necessary data support for the business logic layer (for example, if a user logs in, he needs to put the generated token in Redis for verification to see whether it is a legitimate user, and if he needs to query the user’s permission from the MySQL database, the database service layer will provide data persistence support). The data access layer is responsible for the persistence of business entities[10]. The server system architecture is shown in Figure 4.

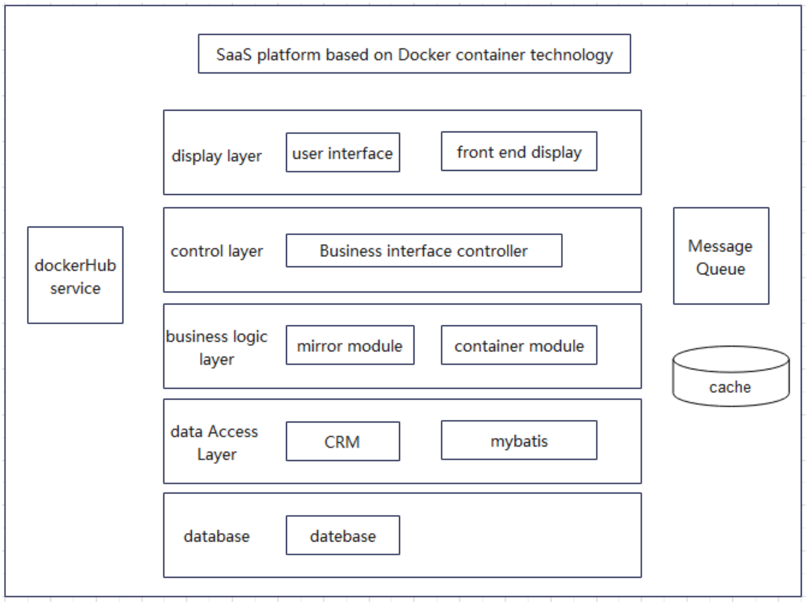

Figure 4. Server system architecture.

3.3 Database design

The database involves the implementation of 14 tables corresponding to different functions. The conceptual model (ER diagram) is shown in Figure 5.

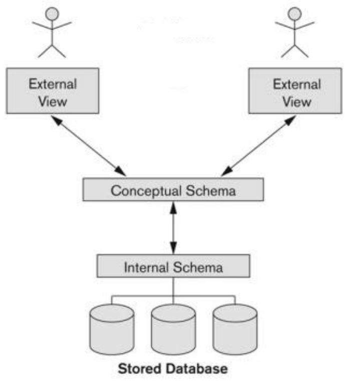

Figure 5. ER diagram.

3.4 System use case realization

3.4.1 User create container process. The user enters the project, operates on the Create Container page, and returns the corresponding prompt after the operation. The case of creating a container instance is shown in Figure 6.

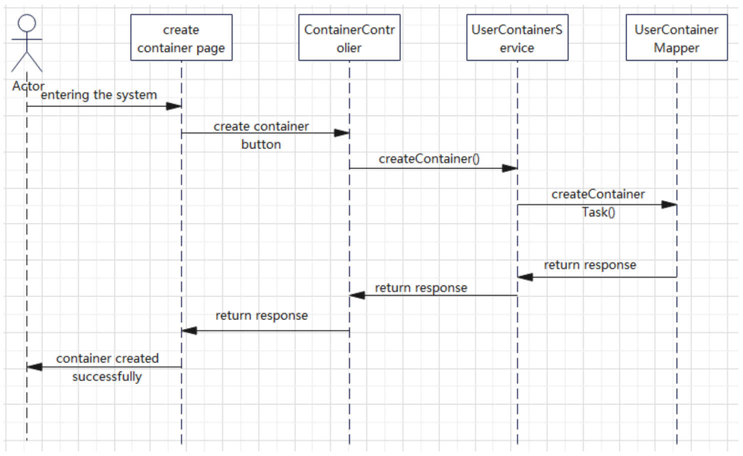

Figure 6. Sequence diagram of create container instance use case.

3.4.2 User created project function. The user creates the project on the Create Project page, and returns the corresponding prompt after the operation. The implementation sequence diagram of creating project use cases is shown in Figure 7.

3.4.3 User management run container. The case of running container instance creates a container successfully for management. Select the container to run and return the corresponding prompt after the operation. The implementation sequence of running container instance is shown in Figure 8.

3.4.4 User export image function. The user enters the project to export the image, and returns the corresponding prompt after the operation. The implementation sequence of the export image is shown in Figure 9.

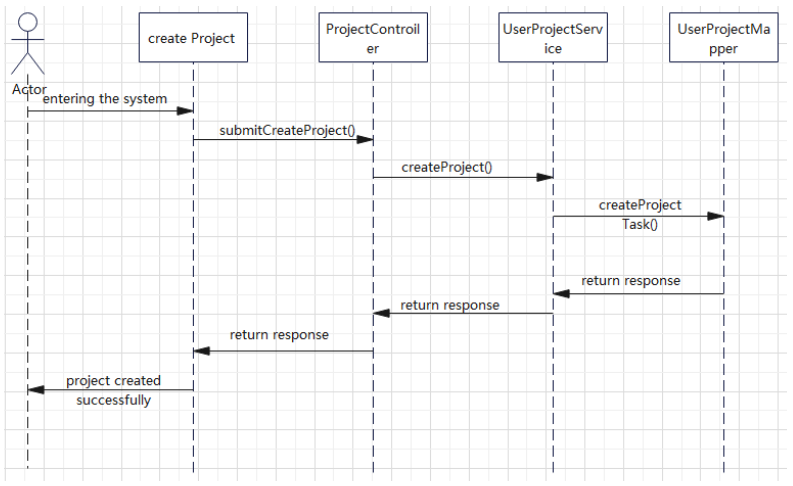

Figure 7. Sequence diagram for creating project use cases.

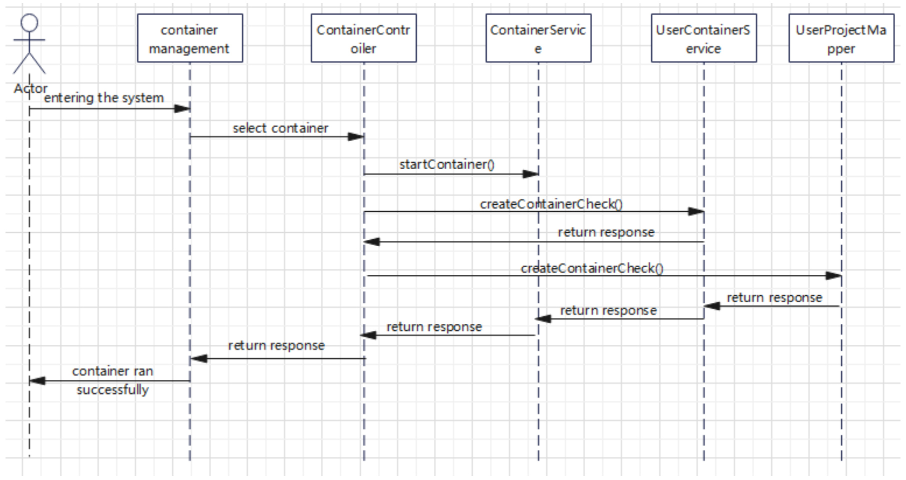

Figure 8. Sequence diagram of use cases of running container.

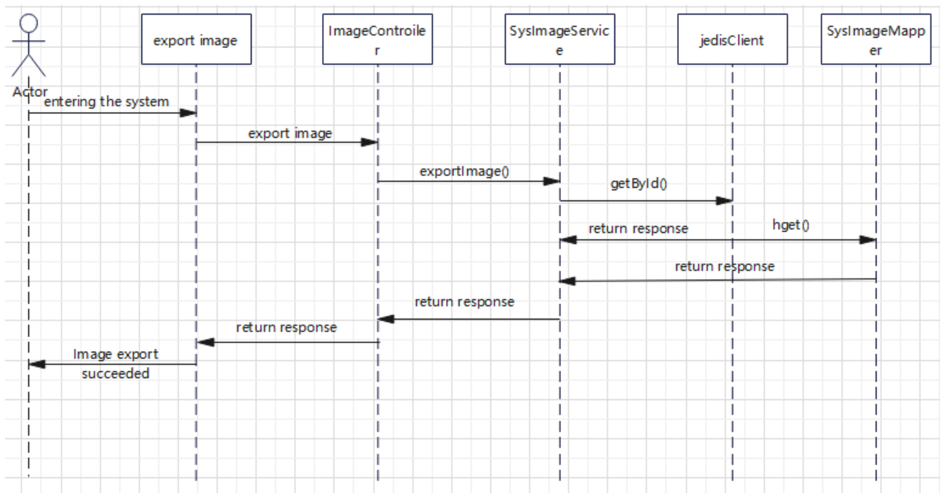

Figure 9. Sequence diagram of export image use case.

4. MODEL IMPLEMENTATION AND DISCUSSION

The core business of the model is to create a container instance. The user needs to configure some parameters required for creating a container instance from the front end and send the parameters to the back end through the request. The back end will request the DockerClientAPI to interact with the Docker daemon through the ActiveMQ message queue task asynchronously, and then create a container instance. If the creation is successful, the result will be sent to the ActiveMQ message queue, Return the prompt information to the user.

The logic code of the back-end core business part of the container creation is shown below.

```
if(portMap != null) {
        Set<String> realExportPorts = new HashSet<>();
        Map<String, List<PortBinding>> portBindings = new HashMap<>(16);
        for(Map.Entry<String, String> entry : portMap.entrySet()) {
           String k = entry.getKey();
           int v = Integer.parseInt(entry.getValue());
           realExportPorts.add(k);
```

Importing an image is to upload the image file locally to DockerHub for direct use, transfer the file through the file byte stream, and then create a container instance object through the uploaded image. Exporting an image is to download the image file on the remote DockerHub.

Because the system is a token authentication mechanism of single sign on based on Redis technology, and the system uses the Spring Security permission control framework, and encrypts the user's password plaintext information, it improves the security of the user's personal information using the system. Therefore, the key technical difficulty is to correctly verify the Token information when generating and logging in the Token based on the Spring Security framework, and put the Token information into the Redis cache.

```
JwtService jwtService = SpringBeanFactoryUtils.getBean(JwtService.class);
  String token = jwtService.genToken(username);
  response.addHeader("Authorization", token);
  response.setHeader("Access-Control-Expose-Headers","Authorization");
  UserVO userVO = jwtService.getUserInfo(token);
  String json = JsonUtils.objectToJson(ResultVOUtils.success(userVO));
  response.getWriter().write(json);
```

5. CONCLUSIONS

The development goal of the Docker container system SaaS platform is to provide small and medium-sized enterprises with the visual operation function based on Docker container technology for customized processing of requirements. For the import and export of images, container instance monitoring indicators, the overall performance of the model can meet the expected goals before the system design. It reduces the tedious steps of developers' deployment, operation and maintenance, and has the advantages of standardized migration, unified parameter configuration, automatic deployment, application cluster monitoring, etc.

REFERENCES

[1] Nigel, P., [Understanding Docker], 04:20(2019).

[2] Meng, J., [Research and Application of Docker Based PaaS Platform], North China Electric Power University (Beijing), (2017).
[3] Chen, Q. and He, J., "Vue+Springboot+MyBatis technology application analysis," Computer Programming Skills and Maintenance, 2020(01), 14-15+28(2020).
[4] Xu, M., Wang, F. and Zhang, M., "Application and research of Redis technology based on large data," Information Technology and Informatization, 2019(11), 228-230(2019).
[5] Wang, Y., [Research on Docker based SQL Parallel Query Optimization], Jilin University, (2019).
[6] Xiang, Z., Shi, C. and Han, L., "Docker container technology in MES system deployment," Manufacturing Automation, 42(4), 116119(2020).
[7] Liu, S., "Linux based private cloud and container deployment design and implementation," China New Communications, 21(18), 53(2019).
[8] Li, N., "Research on the development and application of Docker container technology," Word Technology and Application, 36(11), 95-96(2018).
[9] Shu, R., Gu, X. and Enck, W., "A study of security vulnerabilities on Docker Hub," Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, ACM, Scottsdale, 269-280, (2017).
[10] Ying, Y., Liu, Y. and Yu, Y., "Using Docker container technology to build a large Data Lab," Laboratory Research and Exploration, 37(2), (2018).