Research and Implementation PaaS platform based on Docker

Xugang Yin[1, a], Yanlei Shang[2, b]

[1] The State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China

[2]The State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China

[a]email: yinxugang2009@126.com, [b]email: shangyl@bupt.edu.cn

Keywords: PaaS; Docker; Kubernetes; Cloud Computing

Abstract. With the development of Internet technology and the popularity of cloud computing technology, the organizations or companies try to build a private cloud platform. PaaS is a form of cloud computing service resources and provides application development environment. As the existing PaaS platform hosting and virtual machine environment simplification of the problem of excessive consumption of resources, we have conducted in-depth research on the PaaS platform in this paper. We propose a mechanism to create PaaS platform based on Docker [8]. Docker provides a running application solutions and is built on a lightweight virtualization LXC container [3]. We also develop a dashboard to facilitate users operations. Even though users do not know the professional knowledge of cloud, they can easily deploy their application.

Introduction

Cloud computing is a form of shared computing resources, and make the centralizing computing resources fully utilized by a specific form. Cloud computing platform based on the sharing of resources is divided into three levels of cloud computing [1]: IaaS (Infrastructure as a Service), PaaS (Platform as a Service) and SaaS (Software as a Service). PaaS is an important service mode in the cloud computing, PaaS is between SaaS and IaaS [4]. PaaS provides a much easier operation and deployment environment for application software [2].

However, there are some deficiencies in the existing PaaS platform [9]. Firstly, PaaS platform of the application hosting environment is single and provide only the operating environment particular programming language or scripting language. Secondly, the components of PaaS platform is closed. Lastly, virtual machines consume excessive resources. So the PaaS platform proposed internet applications will focus on and address the following issues. Firstly, the platform should provide the runtime environment for a variety of applications. It not only supports popular programming and scripting languages and also provide stronger compatibility and more versatile operating environment. So the virtual machine is also provided as a runtime environment for application. Secondly, the platform can not only provide the ability that providing an open assembly mechanism to users and allow a third party to provide capabilities based on this platform. Lastly, we should find a more lightweight virtualization solutions to reduce resource consumption [7]. This paper is to research and implement a lightweight PaaS platform to meet the individual needs of users and simplify users' work.

System Design

This paper is designed and implemented PaaS platform based on Docker and divided into the following areas.

Environment Deployment.

The PaaS platform is deployed on the OpenStack and use the OpenStack virtual machine to deploy. It is master/slave architecture in this cluster. The apiserver is deployed in the master node as the entry system. It encapsulates the Docker container add, delete, change and other operations and is provided to external customers and internal components to call.

Web Management Interface.

Web management interface is written by Spring MVC framework. It is mainly through separate model, view and controller role in the application business logic decoupled from the interface. Typically, the model is responsible for encapsulating application data displaying in the View layer. View just show these data, it does not contain any business logic. The controller is responsible for receiving requests from the user and calling back services to handle business logic. After processing, the background business layer might return some data and show in the view layer. The controller collects the data and prepared model and display in the View layer. The core idea of the MVC pattern is to separate business logic from the interface and allow them to change independently without affecting each other. In Spring MVC application, the model usually consists of POJO objects. It is treated in the business layer and persistent in the persistence layer. View usually is JSP template written with JSP Standard Tag Library. Controller section is the responsibility of the dispatcher servlet. GET, PUT, DELETE and other methods of RESTful API [6] is called on the web interface. We can view the status of pods and nodes in the cluster and also achieve the creation, delete and other operations of the pod on the web interface. Graphical interface is more convenient for users.

Docker Cluster Management.

Docker cluster communication uses flannel network configuration mode. Flannel allows Docker container created on different nodes in the cluster to have cluster-unique virtual IP address. So that we can be able to communicate with Docker in different containers directly through the IP network.

VNCserver can be used to achieve web access to Docker container. VNCserver is to meet the distributed users to share server resources. NoVNC is a VNC client based on HTML5. We install VNCserver in Docker container and access the Docker container through IP: Port.

We deploy the docker registry as our private registry. And we can make Docker images using Dockerfile according users' needs. It is convenient for users to deploy their applications or environments.

System Implementation

Docker cluster management uses Kubernetes management system [5]. It is built on Docker technology, providing the container resource scheduling of the application, the deployment operation, service discovery, expansion of volume reduction and other functions. Its architecture is shown in Figure 1.

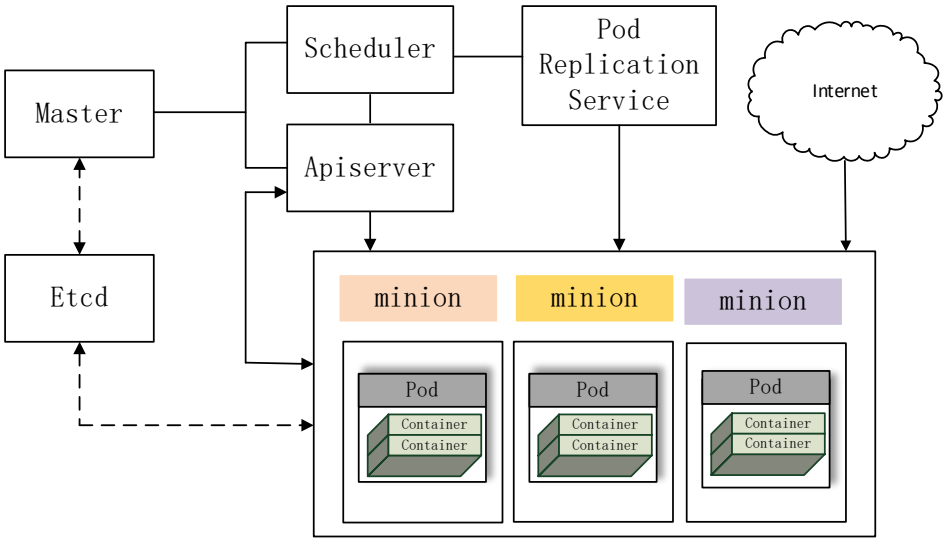

Fig. 1. Architecture

We choose three virtual machines to deploy the PaaS platform in the OpenStack platform. We install the kubernetes with the source complier on the ubuntu 14.04 operating system. It is master/slave architecture in this cluster. There are two minion nodes and one master node. The master node is also the minion node.

We use flannel network as the way of communication with pods in the different nodes. Flannel runs an agent, flanneld, on each host and is responsible for allocating a subnet lease out of a preconfigured address space. Flannel uses etcd to store the network configuration, allocated subnets, and auxiliary data (such as host's IP). The forwarding of packets is achieved using one of several strategies that are known as backend. The simplest backend is udp and uses a TUN device to encapsulate every IP fragment in a UDP packet, forming an overlay network.

Etcd is a highly-available key value store which we use for persistent storage of all of its REST API objects. It serves as the backbone of distributed systems by providing a canonical hub for cluster coordination and state management – the systems source of truth.

We use Docker as the basic components and deploy it in every node. Now comparison traditional virtualization with Docker technology architecture is shown in Figure 2. Traditional virtualization technology is at the hardware level virtualization and needs to have additional virtual machine management application and virtual machine operating system layer [10]. Docker container is a virtualized, direct reuse local host operating system on the operating system level, and therefore is more lightweight.

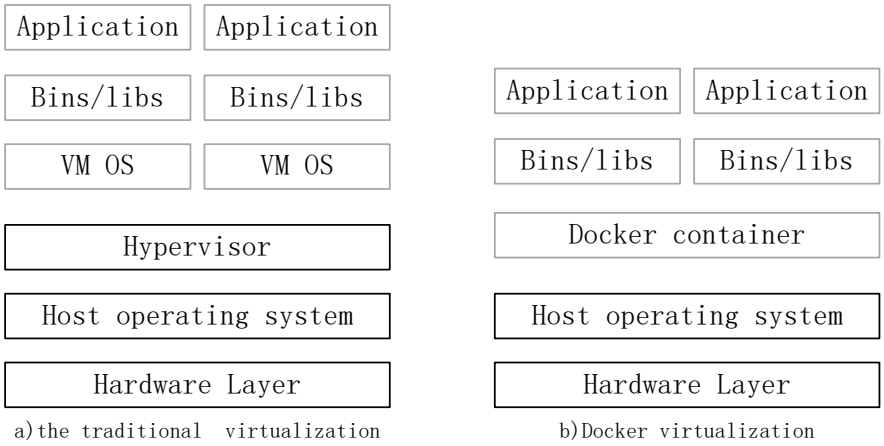

Fig. 2. Comparison traditional virtualization with Docker

We develop a dashboard using Spring MVC framework. The processing flow shown in Figure 3. In this project, the general style of the API is RESTful - clients create, update, delete, or retrieve a description of an object via the standard HTTP verbs (POST, PUT, DELETE, and GET) - and those APIs preferentially accept and return JSON. Then we parse the JSON strings and show in the web dashboard.

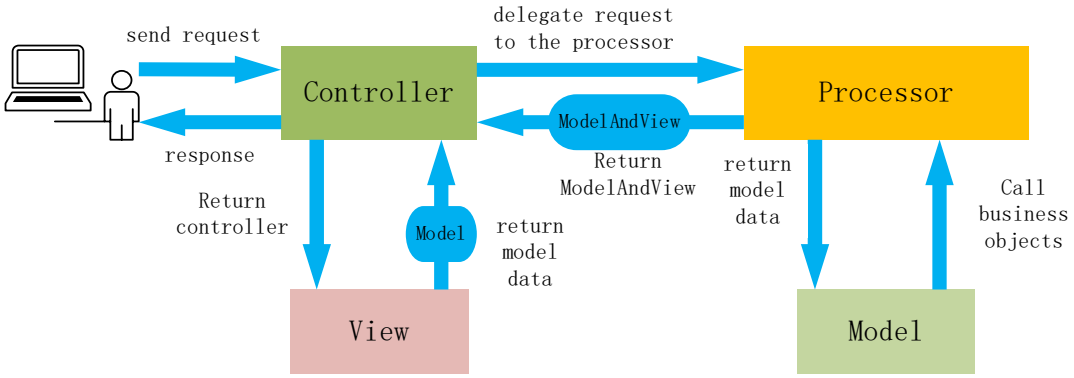

Fig. 3. Spring MVC processing flow

VNCserver is to meet the distributed users to share server resources and is opened on the server. NoVNC is the VNC client based on HTML5. It is widely used in the major cloud computing and virtual machine control panel, such as OpenStack Dashboard and OpenNebula Sunstone. NoVNC is achieved by WebSockets, but current many VNC servers do not support WebSockets. NoVNC does not directly connect to VNC server and needs a proxy to turn convert between WebSockets and TCP sockets.

Docker uses the Docker hub to store images. Users create containers by pulling images from Docker hub. The problem is that the network delay is so long. So we deploy a private registry to

store our images. And we can make images by Dockerfile to meet users' demands. The registry is deployed in the container in the node.

In this cluster, we deploy a DNS cluster addon. The running DNS pod holds three containers – skydns, etcd and kube2sky. The etcd is a private instance which skydns uses and the kube2sky process watches the master for changes in Services and then writes the information to etcd which skydns reads.

Experience

By the Docker cluster management, we log on to the master node and execute kubectl command as follows. As shown in Figure 4, the view of the current status of the nodes in the cluster, ip information and pod cluster name, status, and location information such as node ip. First, we input the URL and choose the GET method. Then the request of get post is sent to master node and return JSON format pod Information. It is shown in Figure 5.

```
root@master1:~# kubectl get nodes
NAME              LABELS                                                  STATU
S
192.168.100.224   kubernetes.io/hostname=192.168.100.224,nodename=node2    Ready
192.168.100.225   kubernetes.io/hostname=192.168.100.225,nodename=node1    Ready
192.168.100.232   kubernetes.io/hostname=192.168.100.232,nodename=master   Ready
root@master1:~# kubectl get pods
NAME      READY     STATUS    RESTARTS   AGE
busybox   1/1       Running   2          2d
nginx     1/1       Running   0          2h
root@master1:~#
```

Fig. 4. Nodes information

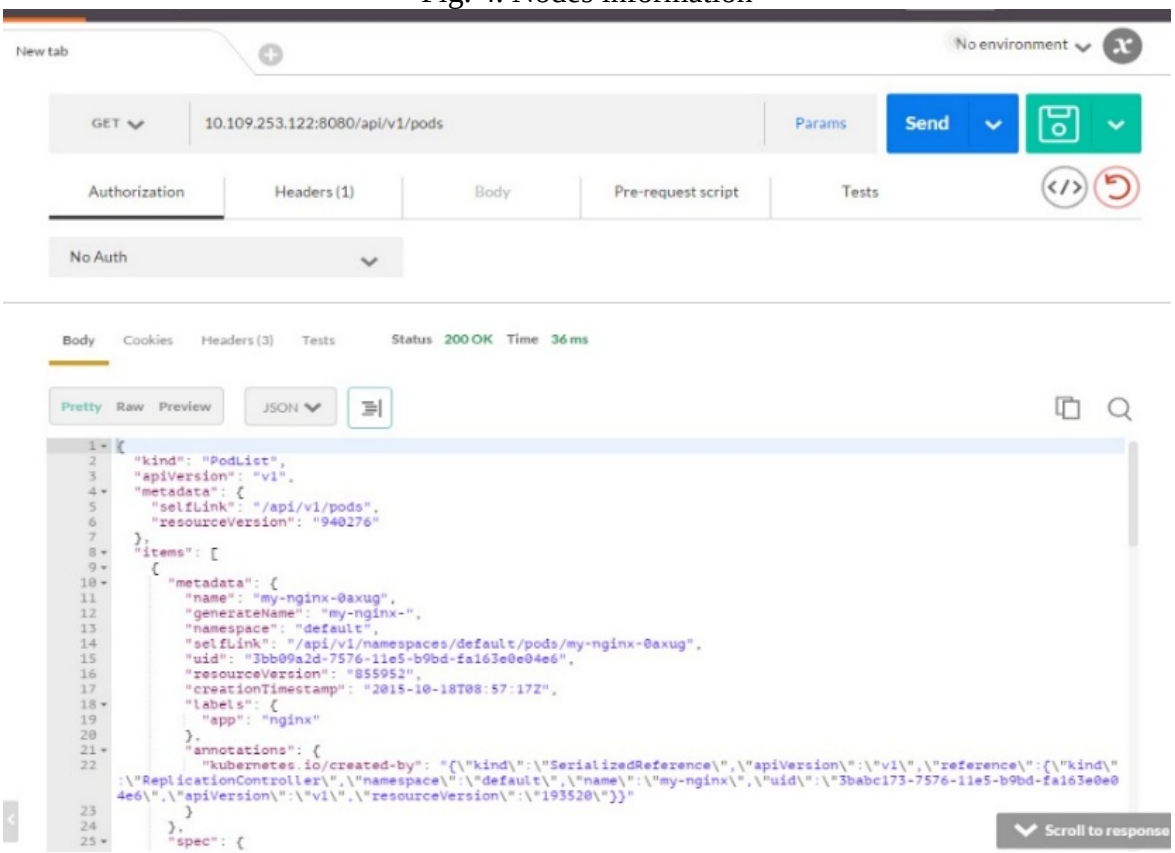

Fig. 5. Pod information from GET function

We send the kind, apiserver, metadata and other information in JSON to the master node. And we will receive the success response of creation. These steps is shown in Figure 6.

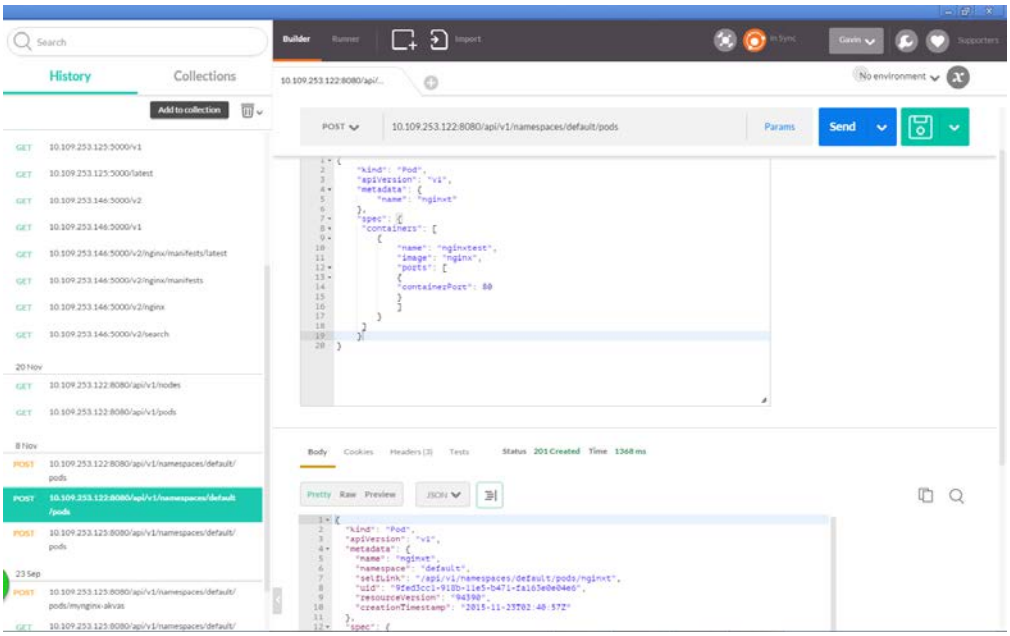

Fig. 6. Creating a pod

Web interface is shown in Figure 7. In the web interface, we can find out information about node, pod and image. And in the pod management, we can create and delete a pod.

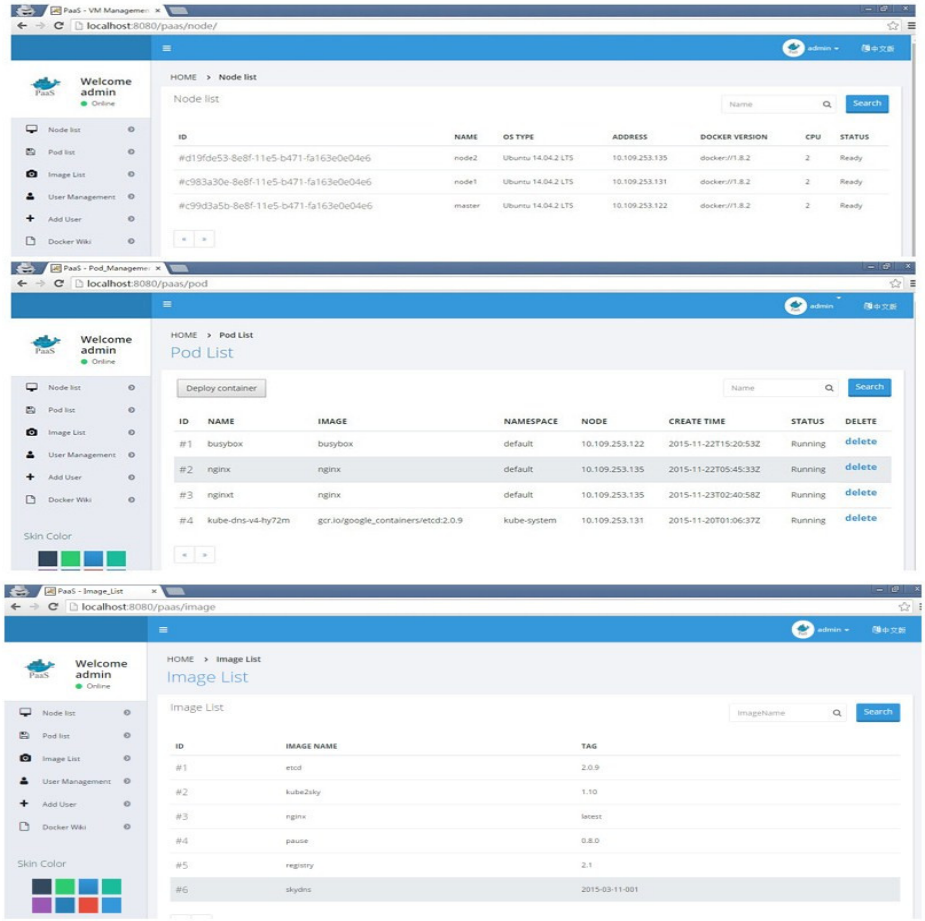

Fig. 7. Web interface

Conclusion

The development of PaaS platform meets the demands of development environment for developers. However virtual machine resource consumption and hosting environment simplification is still the burden of developers. In this paper, the PaaS platform provides developers with a

lightweight development environment and the runtime environment of various applications. Our system is easy to user, it can be adopted in many other platforms (cloud platform or bare metal). For users, this platform is simple to use with a web interface. Users can be better to build private clouds and use virtualization based on flexibility and maintainability of Docker. It can be taken advantage of the IT assets and solve the waste problem.

Acknowledgement

This research is supported by the National Key Technology Research and Development Program of China (Grant No. 2012BAH94F02); National Natural Science Foundation of China under Grant No. 61132001); National High-tech R&D Program of China (863 Program) under Grant No. 2013AA102301; Project of New Generation Broad band Wireless Network under Grant No. 2012ZX03005008-001.

References

[1] M. Armbrust, A. Fox, R. Griffith, et al., “A view of cloud computing,” Communications of the ACM, 2010, 53(4): 50-58.

[2]KIBEL S,WATANABE S,KUNISHIMA K,et al.PaaS on IaaS[C].2013IEEE 27th International Conference on Advanced Information Networking and Applications,2013.

[3] Xavier M G, Neves M V, Rossi F D, et al. Performance evaluation of container- based virtualization for high performance computing environ‐ments[C]//Parallel, Distributed and Network- Based Processing (PDP),2013 21st Euromicro International Conference on. IEEE, 2013: 233-240.

[4] Walraven,Stefan,Truyen,et al.Comparing PaaS offerings in light of SaaS development:A comparison of PaaS platforms based on a practical case study[j] .Computing,2014,96(8):669-724.

[5] D. Bernstein, “Containers and Cloud: From LXC to Docker to Kubernetes,” IEEE Cloud Computing, vol. 1, no. 3, 2014, pp. 81‐84.

[6] L. Richardson and S. Ruby, RESTful web services, O'Reilly Media,Inc., 2008.

[7] O. Gass, H. Meth, and A. Maedche, “PaaS Characteristics for Productive Software Development: An Evaluation Framework,” IEEE Internet Computing, vol. 18, no. 1, 2014, pp. 56‐64.

[8] J. Turnbull, The Docker Book, 2014; www.dockerbook.com.

[9] R. Ranjan, “The Cloud Interoperability Challenge,” IEEE Cloud Computing, vol. 1, no. 2,2014, pp. 20‐24

[10] Pahl, C. Containerization and the PaaS Cloud, IEEE Cloud Computing, vol.2,no. 3, 2015, pp.24-31.