Journal of Engineering and Applied Sciences 12 (5): 1261-1264, 2017
ISSN: 1816-949X

Containerized Architecture for Software as a Service Applications Development

Vladimir N. Solovyev, Andrey V. Prokofyev and Roman G. Chesov
Flexbby Solution's LLC Dolgoprudny, Moscow Region, Russia

Abstract: This study describes the containerization of a multi-tier client-server architecture based on LXC isolation technology. A multi-tier client-server architecture was used as a single node to create the orchestration application to manage saas cloud. Several types of applications like as CRM, contract management have been developed and placed in application container.

Key words: LXC, docker, SaaS, virtualization, containerization

INTRODUCTION

Virtualization and cloud computing have great impact on how enterprise customers use software. Organizations are adopting virtualization technology, using public and private clouds, SaaS, integrating storage systems, network infrastructure, computing capacity to increase resource utilization and improve fault tolerance of mission-critical business applications (Columbus, 2015).

Meanwhile, from the perspective of business management and saas service maintenance, the main goal of the enterprise or service provider computing infrastructure is the ability to quickly and cheaply create, deploy and manage applications (CRM, ERP, Contract Management, BI, etc.).

This is the reason for the increased interest to application containerization technologies such as LXC by Docker (2017). Containerization can be effectively used to build application and services hosting cloud infrastructure as well as to unify and simplify the process of applications development and maintenance. According to some studies, containerization reduces deployment time by 54%, labor costs by 40% and the total cost of application deployment by 30% (Rastogi and Sushil, 2015).

MATERIALS AND METHODS

The idea of applications containerization is not new. It was used in unix systems such as free bsd jail and solaris containers but did not gain widespread adoption until the appearance of LXC and Docker. Development and use of containerization is currently one of the drivers for public a nd private clouds infrastructure (Linux Containers, 2014). In contrast to hypervisor-based virtualization (e.g., Microsoft Hyper-V, XEN, ESXi Hypervisor) that implements hardware abstraction layer,

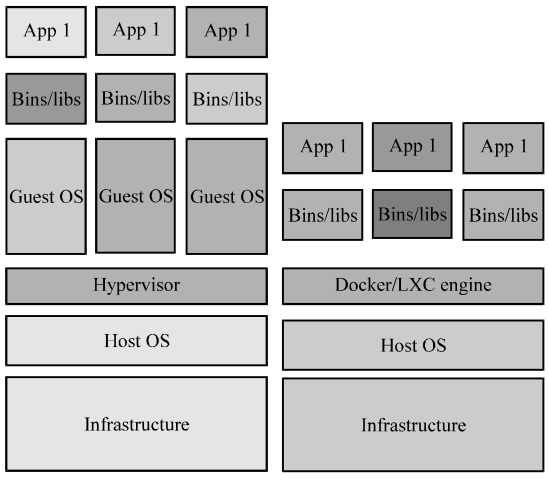

Fig. 1: Virtualization and containerization

containers allow to isolate operation system calls and run multiple isolated application containers at the same operation system kernel without hardware emulation (Fig. 1). This approach allows to save system resources (Dua *et al*., 2014; Morabita *et al*., 2015). Application containerization simplifies applications delivery to end users, change management, migration and backup. As a result, it is gaining popularity among software developers and consumers.

Architecture of the containerized cloud

Multi-tier client-server architecture: The multitier client-server architecture has been divided into several functional layers to simplify administration, scalability and change management (Fig. 2). To implement each functional layer, one or more techniques can be used. Presentation layer includes everything related to user interaction with the system. It can be as simple as a command-line interface but now a days users

Corresponding Author: Vladimir N. Solovyev, Flexbby Solution's LLC, Dolgoprudny, Moscow Region, Russia

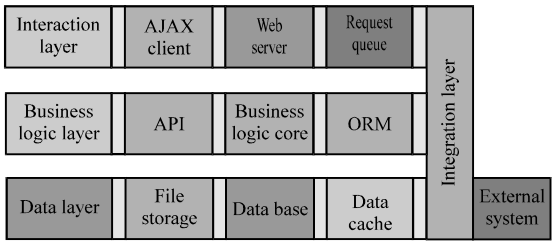

Fig. 2: Multitier client-server architecture layers

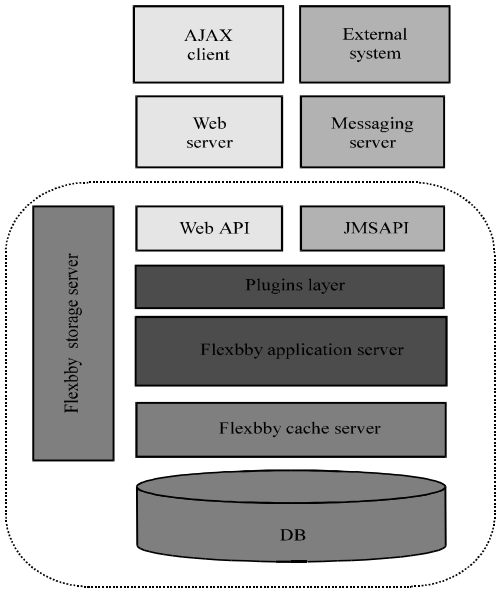

Fig. 3: Multi-tier client-server architecture of the application

and customers have a number of requirements such as compatibility of graphical user interface with a variety of web-browsers, mobile accessibility, convenient operation with different screen resolutions.

Data layer is a part of the system that provides data storage and provisioning (for most business and analytical systems the data source is the database management system). Additional data sources can be external systems or in terms of the system being developed, integration interfaces and non-structured data storage.

Business logic layer (domain logic) describes the main functionality of the application that constitutes the business domain of analytical or business applications. Integration layer provides the integration between the components of the multi-tier architecture and external systems. This multi-layer approach has been used to create a multi-tier client-server architecture which later was containerized using LXC and Docker technologies. This type of architecture is a multi-tier client-server application (Fig. 3).

Key components of this system are a database (Postgre SQL, MSSQL, etc.) an application server, a messaging server (Apache ActiveMQ, other standard components can also be used as a messaging server). In addition, there are 3 components providing access to the application logic from the web-browser. They are a web server (Apache, nginx or others), web API adapter (Fast CGI Service) and AJAX web-client. Another component, a tomcat servlet container is used for generating document print forms and saving user documents. The components of the system can be installed in the same container node or be separated into different application containers for load balancing and scalability (Slideshare, 2017).

RESULTS AND DISCUSSION

Containerization of multi-tier client-server architecture Management hierarchy for an isolated container: A high-level API has been developed to manage containerization which allows access to the host machine Linux kernel and LXC environment. At the level of an isolated application container, "initsys" component has been designed which provides access for external components to the container core through a high-level API and allows to control start up and shutdown of the required services. "SSN-manager" has been developed at the level of the host machine which uses a high-level API to provide access for the external administration server to Linux kernel functions, visualization tier (LXC), virtual container and its services through "initsys" component. In addition, "SSN-manager" manager has access interfaces to file storage server and web-servers to provide control from a single administrator console. "SSN-manager" has an API to developing business logic for subscription management (at the cloud administrator level) which can also be deployed in one of the containers. It provides a homogeneous environment in which the control system uses the same infrastructure which it controls. Management hierarchy for an isolated container is shown in Fig. 4.

File hierarchy of a container: A "base container" component has been developed to provide virtual environment updates. Base container is used to initialize the base application container (Linux Foundation, 2003). A "subscription project" may be created for each type of containerized applications which includes a "base container" +project business logic +files+database. In turn, the virtual application container contains only user data, settings and files. If the container cloud is updated, it is enough to simply upgrade the "base container" once and restart the subscription through the administrator console. The subscription will start with the new update.

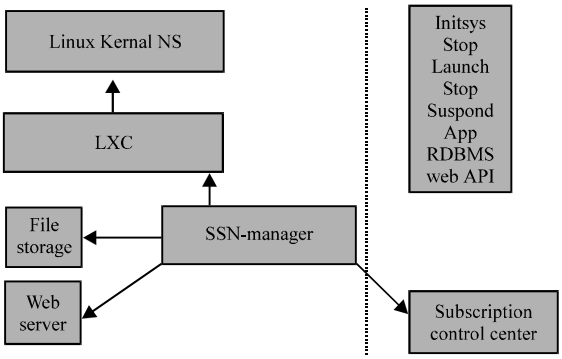

Fig. 4: Management hierarchy for an isolated container

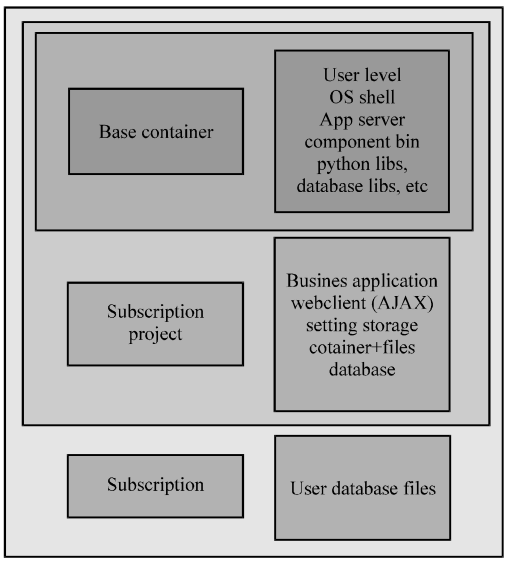

Fig. 5: File hierarchy of a container

In case of updating the business logic of subscription, client, etc., it is enough to update only the subscription project. All the subscriptions created based on this project will also be updated (Fig. 5). As a result, subscription update time does not depend on the number of subscriptions and takes only a few seconds (Albert, 2015).

Network hierarchy of container management: All components of the cloud infrastructure interact via IP protocol which allows to locate the cloud components in a physically and geographically distributed infrastructure (Fig. 6).

Usage and testing: The developed containerization layer was used to containerize analytical and business applications and deploy them in the cloud for saas applications. Ubuntu OS was used as a host OS. Examples of user interfaces of the saas applications are shown in Fig. 6.

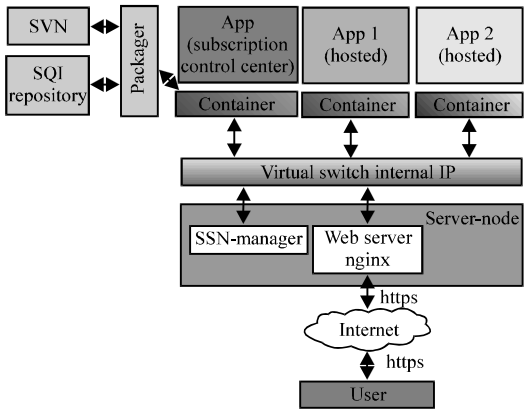

Fig. 6: Network hierarchy of container management

CONCLUSION

This study describes the containerization of a multi-tier client-server architecture based on LXC isolation technology for saas application developments. A multi-tier client-server architecture was used as a single node which includes the following elements: database server and cache subsystem, application server, messaging server, providing the integration of architecture components, application server, web-server and AJAX web client. The multi-tier client-server architecture has an open API based on Python and XML for creating application business logic and modifications of the user interface. In order to provide the isolation layer of a single node in the multi-tier client-server architecture, implementation classes and containerization management classes have been designed and implemented in C++.

The approach proposed in this study allowed to create a homogeneous environment in which the control nodes are built using the same technology as the controlled nodes. This technology allowed to create a prototype of the cloud infrastructure for business and analytical SaaS applications hosing. The first test results show that containerization technology has a potential to create scalable clouds for business and analytical SaaS applications.

ACKNOWLEDGEMENT

The research was performed with support of the Ministry of Education and Science of the Russian Federation (research No. RFMEFI 57914X0069).

REFERENCES

Albert, P., 2015. Worldwide cloud applications market forecast 2014-2018. Apps Run the World, Dublin, California. https://www.appsruntheworld.com/the-hand-rail-is-going-a-little-faster-than-the-moving-sidewalk/.

Columbus, L., 2015. Roundup of cloud computing forecasts and market estimates, 2015. Forbes, New York, USA. https://www.forbes.com/sites/louiscolumbus/%202015/01/24/roundup-of-cloud-computing-forecasts-and-market-estimates-2015/.

Docker, 2017. Docker documentation. Docker Inc., San Francisco, California. https://docs.docker.com/.

Dua, R., A.R. Raja and D. Kakadia, 2014. Virtualization vs containerization to support paas. Proceedings of the 2014 IEEE International Conference on Cloud Engineering (IC2E), March 11-14, 2014, IEEE, Boston, Massachusetts, USA., ISBN:978-1-4799- 3768-4, pp: 610-614.

Linux Containers, 2014. LinuxContainers.org Infrastructure for container projects. Linux Containers Organization, USA. https:// linuxcontainers.org/.

Linux Foundation, 2003. Virtualization the open source standard for hardware virtualization. San Francisco, California, USA. http://www.xenproject.org/users/virtualization.html.

Morabito, R., J. Kjallman and M. Komu, 2015. Hypervisors vs. lightweight virtualization: A performance comparison. Proceedings of the 2015 IEEE International Conference on Cloud Engineering (IC2E), March 9-13, 2015, IEEE, Tempe, Arizona, USA., ISBN:978-1-4799-8219-6, pp: 386-393.

Rastogi, G. and R. Sushil, 2015. Cloud computing implementation: Key issues and solutions. Proceeings of the 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), March 11-13, 2015, IEEE, Dehradun, India, ISBN:978-9-3805-4415-1, pp: 320-324.

Slideshare, 2017. Discover, share, present, share what you know and love through presentations, infographics, documents and more. LinkedIn Corporation, Mountain View, California. http://www.slideshare.net/TBR_Market_ Insight/the-developers-coup-2015-applications-development-demands-and-vendor-opportunities.