ISBN 978-93-86878-04-5
6th International Conference on Science, Engineering, Technology and Healthcare (SETH-2017)
Kuala Lumpur (Malaysia), Nov. 20-21, 2017

Design and Implementation of Log Collection and System Statistics Model for Android Containers in OpenStack

Nam Pham Nguyen Xuan[1], Jinung Ahn[2] and Souhwan Jung[3]

[1]School of Software Convergence, Soongsil University, Seoul, South Korea
[2]School of Information and Telecommunication Engineering, Soongsil University, Seoul, South Korea
[3]School of Electronic Engineering, Soongsil University, Seoul, South Korea

Abstract: *In the cloud computing, OpenStack is an enormous popular open-source cloud platform for deploying Infrastructure-as-a-service (IaaS) and in the mobile ecosystem, Android Container – an Android environment stimulated on HW device is becoming a strong technology for Android Malware analysis solution. In this paper, we propose a design and implementation of log collection and system statics model for Android Container in OpenStack Horizon. Through our proposal, logs from android container and HW device such as: system log, application log, kernel log are collected and optimized by Elastic Search before display on OpenStack Horizon. System statistics for Android Container and HW devices are also implemented by us to provide the overall information of whole system for administrator.*

Keywords: *Android Container, OpenStack, Container Runtime, HW device, Elastic Search*

1. Introduction

In virtualization technology, Container is an operating-system-level virtualization technology has been successfully applied to run multiple isolated systems called containers on a single host using existing features in Linux kernel such as: Namespaces [1], Cgroups [2], Linux capabilities [3], Seccomp filter [4], Linux security modules (SELinux, AppArmor) [5]. Containers are deployed and managed by Container Runtimes, some of which are: Linux Container (LXC) [6], Docker [7] or by Container Orchestration Engine (COE) including: Docker Swarm, Kubernetes [8], Mesos [9]. In the traditional virtualization technology, a hypervisor layer must be install on the host kernel; after that guest kernels are deployed on that hypervisor layer for running isolated systems. When making a comparison with hypervisor, Container runs directly on the host kernel for deploying multiple isolated systems, therefore, containers have higher utilization levels of their underlying hardware. In addition, the speed of containers is higher than normal virtual machines running on hypervisor and container reduces the cost of management because only host OS needs to be maintained including paths, securities, bugs fixed.

Container technology is also used for creating Android container in order to provide Bring Your Own Device (BYOD) solution which separate working environment and user environment. In malware analysis, Android container is used for simulating an Android device like real device. Android container can be deployed on Android operating system [10] or on Linux operating system. We already deployed Android container on Linux OS successfully. Furthermore, we also designed the management solution for android container integrated in OpenStack platform. In addition, log collection is really important to monitor, audit and resolve incident occurred in android containers. System statistics is a standard to evaluate the performance of the system resource. Therefore, in this paper, we will propose a new design and implementation of log collection and system statistic for Android Containers in OpenStack.

The rest of the paper is arranged as follows. Section 2 describes related works. The proposed architecture will be introduced in Section 3. Section 4 provides a detail implementation of proposed solution. Finally, conclusion will be mentioned in Section 5.

2. Related Works

Zun [11] is a Container Management service for OpenStack which aims to provide an OpenStack API for launching and managing containers backed by different container technologies. The current Container Runtime is being used by Zun is Docker. Container operations provided by Zun can easily integrated with other OpenStack resources like: networking service, image service. Zun also provide resource Also, Magnum [12] works on OpenStack to provide APIs for managing multi-tenant container infrastructure on a virtualization layer. Magnum does not use single Container Runtime, it works with Container Orchestration Engine such as: Kubernetes, Mesos to deploy containers.

Nevertheless, Zun and Magnum only manage and calculate the resource for containers, they do not provide log monitoring function. In the other hand, Zun and Magnum only support for Linux containers such as: Ubuntu Container, CentOS container, Debian Container and do not support for Android Container.

3. Proposed Architecture

This section illustrates in detail the proposed architecture of log collector and system statistics for android container in OpenStack. As depicted in Figure 1, our architecture consists of the following components: OpenStack Horizon, HTTP REST APIs, Android Container, Log Collector agent, System Statistics agent, Database and Elastic Search.

3.1. OpenStack Horizon

OpenStack Horizon is an OpenStack dashboard project for managing different type of instances such as: virtual machine (KVM, Xen) and physical machine (Ironic). OpenStack Horizon is as a popular project that many organizations are integrating their system into it. We develop an Android container management module on OpenStack Horizon to provide a graphic user interface for administrator carries out orchestration tasks such as: manage all of Android containers and HW device in our platform, Log monitoring and system statistics. Our module on OpenStack Horizon communicates directly with HRA to send the requests, receive the result and display on Horizon.

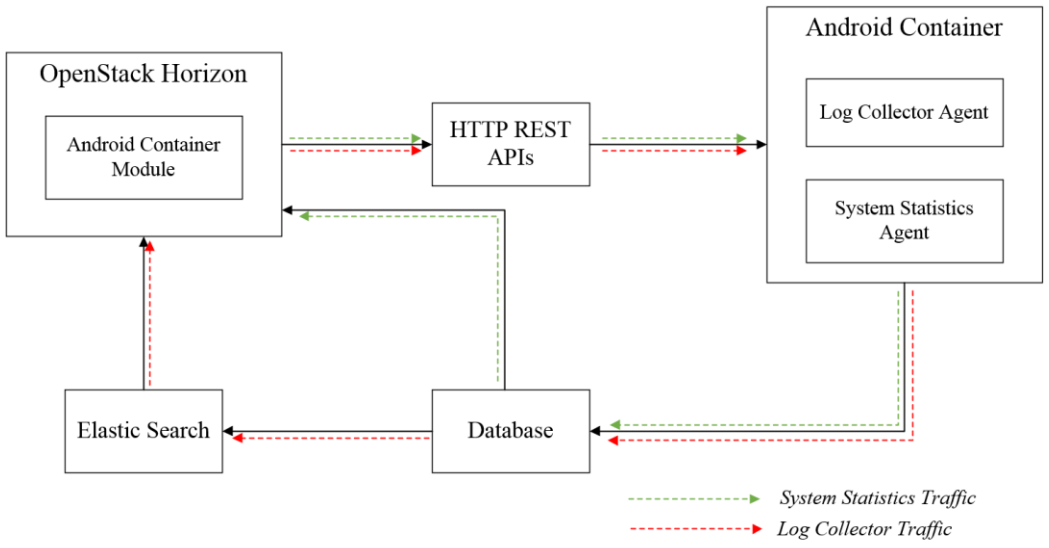
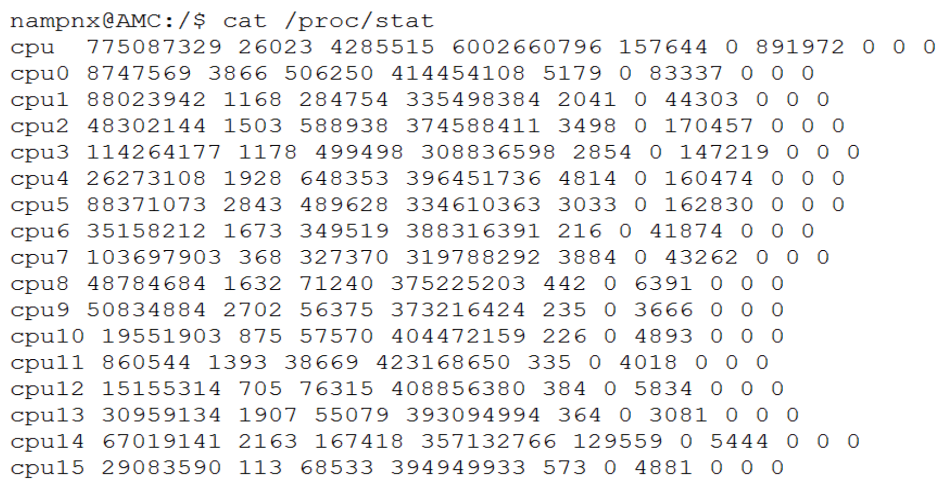


Fig.1. The architecture of Log Collector and System Statistics for Android Container in OpenStack Horizon.



```
nampnx@AMC:/$ cat /proc/stat
cpu  775087329 26023 4285515 6002660796 157644 0 891972 0 0 0
cpu0 8747569 3866 506250 414454108 5179 0 83337 0 0 0
cpu1 88023942 1168 284754 335498384 2041 0 44303 0 0 0
cpu2 48302144 1503 588938 374588411 3498 0 170457 0 0 0
cpu3 114264177 1178 499498 308836598 2854 0 147219 0 0 0
cpu4 26273108 1928 648353 396451736 4814 0 160474 0 0 0
cpu5 88371073 2843 489628 334610363 3033 0 162830 0 0 0
cpu6 35158212 1673 349519 388316391 216 0 41874 0 0 0
cpu7 103697903 368 327370 319788292 3884 0 43262 0 0 0
cpu8 48784684 1632 71240 375225203 442 0 6391 0 0 0
cpu9 50834884 2702 56375 373216424 235 0 3666 0 0 0
cpu10 19551903 875 57570 404472159 226 0 4893 0 0 0
cpu11 860544 1393 38669 423168650 335 0 4018 0 0 0
cpu12 15155314 705 76315 408856380 384 0 5834 0 0 0
cpu13 30959134 1907 55079 393094994 364 0 3081 0 0 0
cpu14 67019141 2163 167418 357132766 129559 0 5444 0 0 0
cpu15 29083590 113 68533 394949933 573 0 4881 0 0 0
```

Fig.2. CPU information gotten for /proc/stat.

3.2. Log Collector Agent

In originally, log collector agent is set of Shell scripts. These scripts eventually collect logs from Android operating system and HW devices such as: kernel log, application log, system log. After collecting, this agent parses logs according to predefined parameters. Finally, log collector agent pushing collection of logs into database server.

3.3. System Statistics Agent

In order to calculate the statistics for system, we develop an agent to collect and calculate the CPU utilization, Memory usage and number of running processes. This agent consists of Python scripts and works similar to log collector agent is collect and push information in to database server. For example, in order to calculate CPU utilization, firstly the agent executes reading */proc/stat* to get the CPU information including columns: *user cpu, system cpu, user nice cpu, idle cpu, io wait cpu, hardware interrupt, software interrupt, steal time* that described in Figure 2. Then, it sums all off the values found on that fist line to get total CPU values. After that, it divides the fourth column ("*idle*") by total value to get fraction of value spent being idle and subtracts the previous fraction from 1.0 too get the time spent being not idle. Finally, multiplying by 100 to get a percentage of CPU utilization.

3.4. Database - Elastic Search

Time goes by, the amount off Log from system will be increased so that searching to find the appropriate result being more difficult. To solve this problem, we using Elastic Search [13] to perform and combine many types of searches such as: structured, unstructured, geo, metric before display searching result onto OpenStack.

Information of Android container such as: container name, ip address, granted resource, created time will be stored in a Database server. Database server not only stores Android container information, but also stores user authentication, Android template version, HW device information and monitoring logs.

4. Implementation

In this section, we provide readers with a detailed implementation of Log Collector and System Statistic for Android container. Our implementation topology showed in Figure 3 includes an OpenStack Horizon server, HTTP REST APIs server, HW devices which hosting Android containers.

4.1. System Preparation

We perform our implementation on the Linux operating system for our platform. Newton version is used for OpenStack Horizon. MariaDB is database server and android container are deployed on HW devices using LXC runtime. Kibana is used for implementing Elastic Search and programming language is PHP and Python.

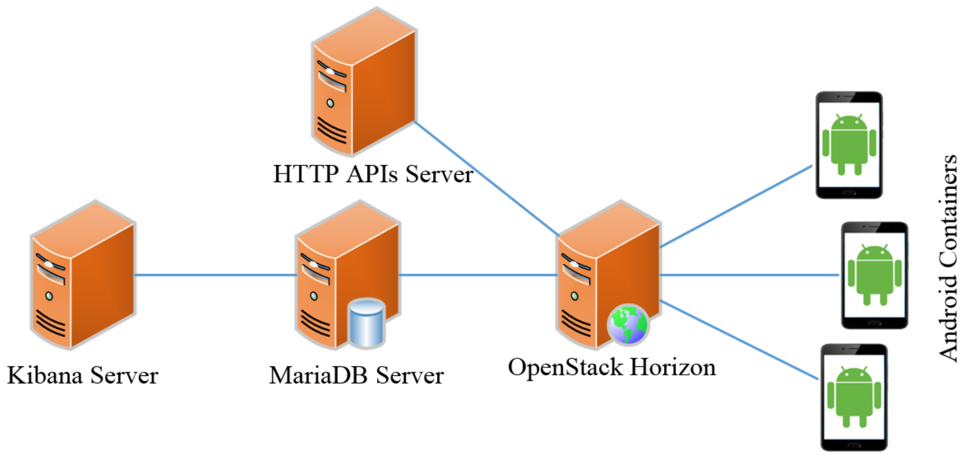
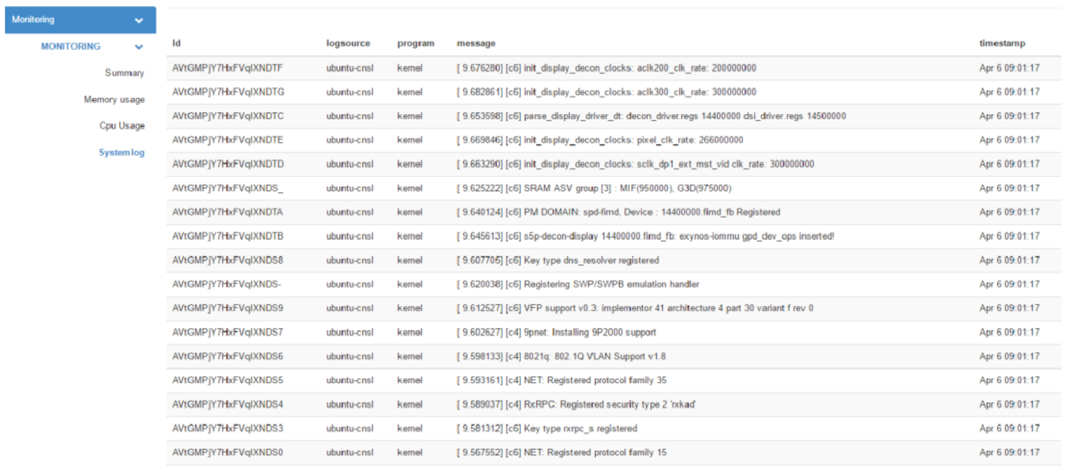


Fig. 3. The implementation of Log Collector and System Statistics for Android Containers.

4.2. Implementation

On OpenStack Horizon, we already developed a new Android container module that including Log monitoring and system information. As shown in Figure 4, the results of Logs monitoring functions on OpenStack Horizon that collected from Android Containers and HW devices. And Figure 5 illustrates the summary of system information such as CPU, memory and running processes.



Id	logsource	program	message	timestamp
AVtGMPjY7HxFVqIXNDTF	ubuntu-cnsl	kernel	[9.676280] [c6] init_display_decon_clocks: aclk200_clk_rate: 200000000	Apr 6 09:01:17
AVtGMPjY7HxFVqIXNDTG	ubuntu-cnsl	kernel	[9.682861] [c6] init_display_decon_clocks: aclk300_clk_rate: 300000000	Apr 6 09:01:17
AVtGMPjY7HxFVqIXNDTC	ubuntu-cnsl	kernel	[9.653598] [c6] parse_display_driver_dt: decon_driver.regs 14400000 dsi_driver.regs 14500000	Apr 6 09:01:17
AVtGMPjY7HxFVqIXNDTE	ubuntu-cnsl	kernel	[9.669846] [c6] init_display_decon_clocks: pixel_clk_rate: 266000000	Apr 6 09:01:17
AVtGMPjY7HxFVqIXNDTD	ubuntu-cnsl	kernel	[9.663290] [c6] init_display_decon_clocks: sclk_dp1_ext_mst_vid clk_rate: 300000000	Apr 6 09:01:17
AVtGMPjY7HxFVqIXNDS_	ubuntu-cnsl	kernel	[9.625222] [c6] SRAM ASV group [3] : MIF(950000), G3D(975000)	Apr 6 09:01:17
AVtGMPjY7HxFVqIXNDTA	ubuntu-cnsl	kernel	[9.640124] [c6] PM DOMAIN: spd-fimd, Device : 14400000.fimd_fb Registered	Apr 6 09:01:17
AVtGMPjY7HxFVqIXNDTB	ubuntu-cnsl	kernel	[9.645613] [c6] s5p-decon-display 14400000.fimd_fb: exynos-iommu gpd_dev_ops inserted!	Apr 6 09:01:17
AVtGMPjY7HxFVqIXNDS8	ubuntu-cnsl	kernel	[9.607705] [c6] Key type dns_resolver registered	Apr 6 09:01:17
AVtGMPjY7HxFVqIXNDS-	ubuntu-cnsl	kernel	[9.620038] [c6] Registering SWP/SWPB emulation handler	Apr 6 09:01:17
AVtGMPjY7HxFVqIXNDS9	ubuntu-cnsl	kernel	[9.612527] [c6] VFP support v0.3: implementor 41 architecture 4 part 30 variant f rev 0	Apr 6 09:01:17
AVtGMPjY7HxFVqIXNDS7	ubuntu-cnsl	kernel	[9.602627] [c4] 9pnet: Installing 9P2000 support	Apr 6 09:01:17
AVtGMPjY7HxFVqIXNDS6	ubuntu-cnsl	kernel	[9.598133] [c4] 8021q: 802.1Q VLAN Support v1.8	Apr 6 09:01:17
AVtGMPjY7HxFVqIXNDS5	ubuntu-cnsl	kernel	[9.593161] [c4] NET: Registered protocol family 35	Apr 6 09:01:17
AVtGMPjY7HxFVqIXNDS4	ubuntu-cnsl	kernel	[9.589037] [c4] RxRPC: Registered security type 2 'rxkad'	Apr 6 09:01:17
AVtGMPjY7HxFVqIXNDS3	ubuntu-cnsl	kernel	[9.581312] [c6] Key type rxrpc_s registered	Apr 6 09:01:17
AVtGMPjY7HxFVqIXNDS0	ubuntu-cnsl	kernel	[9.567552] [c6] NET: Registered protocol family 15	Apr 6 09:01:17

Fig. 4. Summary of system information.

Fig. 5. Monitoring Logs of system

5. Conclusion

In this paper, we already proposed a design and implementation of Log Collector and System Statistics for Android containers in OpenStack. Our proposal can efficiently support system administrators to monitor

Android containers from many HW devices on OpenStack Horizon. On the other hand, by integrating in OpenStack Horizon, OpenStack Horizon becomes a portal where the administrator not only manage Android containers but also manage other types of OpenStack instances such as: Nova virtual machine, physical machine. As for further work, we plan to extend more features for our platform like: collect more types of log from system, develop alert functions for administrator.

6. Acknowledgement

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2017-2012-0-00646) supervised by the IITP(Institute for Information & communications Technology Promotion).

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2016-0-00078, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning).

7. References

[1] Namespaces, “Resource Isolation”, Accessed 2017 [Online]. Available: https://lwn.net/Articles/531114/

[2] Cgroups, “Resource Management”, Accessed 2017 [Online]. Available: http://man7.org/linux/man-pages/man7/cgroups.7.html

[3] Linux Capabilites, “Permission Mechanism”, Accessed 2017 [Online]. Available: http://man7.org/linux/man-pages/man7/capabilities.7.html

[4] Seccomp Filter, “Permission Filter Mechanism”, Accessed 2017 [Online]. Available: http://man7.org/linux/man-pages/man2/seccomp.2.html

[5] Morris, James, Stephen Smalley, and Greg Kroah-Hartman. "Linux security modules: General security support for the linux kernel." USENIX Security Symposium. 2002.

[6] LXC, “Linux Container Runtime”, Accessed 2017 [Online]. Available: https://linuxcontainers.org/

[7] Docker, “Application Containers”, Accessed 2017 [Online]. Available: https://www.docker.com/

[8] Kubernetes, “Google COE”, Accessed 2016 [Online]. Available: https://kubernetes.io/

[9] Mesos, “A distributed systems kernel”, Accessed 2017 [Online]. Available: http://mesos.apache.org/

[10] Uri Kanonov, Avishai Wool, “Secure Containers in Android: The Samsung KNOX Case Study,” in Computer and Communications Security (CSS), 2016, Pages 3-12.

[11] Zun, “Container Management service for OpenStack”, Accessed 2017 [Online]. Available: https://wiki.openstack.org/wiki/Zun

[12] Magnum, “A management solution of application containers orchestration engines”, Accessed 2017 [Online]. Available: https://wiki.openstack.org/wiki/Magnum

[13] Elastic Search, “Perform and combine many types of searches”, Accessed 2017 [Online]. Available: https://www.elastic.co/products/elasticsearch