Container-Based Virtualization for Real-Time Data Streaming Processing on the Edge Computing Architecture

Endah Kristiani[1,2], Chao-Tung Yang[1(✉)], Yuan-Ting Wang[3], Chin-Yin Huang[2], and Po-Cheng Ko[1]

[1] Department of Computer Science, Tunghai University, Taichung 40704, Taiwan, ROC
endahkristi@gmail.com, ctyang@thu.edu.tw, pocheng0605@gmail.com

[2] Department of Industrial Engineering and Enterprise Information, Tunghai University, Taichung 40704, Taiwan, ROC
huangcy@thu.edu.tw

[3] Cloud Computing Laboratory, Chunghwa Telecom Laboratories, No. 99, Dianyan Rd. Yangmei, Taoyuan 326, Taiwan, ROC
yttom@cht.com.tw

Abstract. Container-based virtualization is one of the prominent technologies in the cloud computing. Containers virtualize at the operating system level which provides a lightweight operation than traditional virtualization on a hypervisor. The combination of the Internet of Things (IoT), edge computing and container-based virtualization is going to make system rapid, inexpensive, and more reliable. In this paper, we intend to implement a complete set of edge computing architectures based on containerization on an IoT environment. In this case, we implemented container-based virtualization which constructs Kubernetes Minion (Nodes) in the Docker container service independently for each service on the Edge side. We used humidity and temperature sensory data as our case study. We set up the Raspberry Pi on the Edge Gateway and Kubernetes minion on the Raspberry Pi to provide the service application, which contains Grafana, the open platform for analytics and monitoring. For short-term data storage, we use InfluxDB as a data store for large amounts of time-series data.

Keywords: Edge computing · Container-based virtualization · Kubernetes · Docker · Internet of Things (IoT)

1 Introduction

Internet of Things (IoT) industry has grown substantially. So that, it triggers an exponentially increasing amount of data which need to improve the data analysis process. Cloud computing is essential to the success of IoT, however pure cloud

J.-L. Chen et al. (Eds.): WiCON 2018, LNICST 264, pp. 203–211, 2019.
https://doi.org/10.1007/978-3-030-06158-6_21

computing itself not quite adequate for faster data analysis. The long distance of the network logic between the cloud and the end device might easily cause Network delay, thereby affecting the system can not respond promptly. In this case, the concept of edge computing [1,2] meets the need for faster data analysis. Edge computing is a form of cloud computing, which pushes data processing out to the edge device to handle. This mechanism transports only the results of the data processing over networks. Therefore, it provides accurate results and consumes more lightweight network bandwidth than independent cloud computing. Additionally, we implemented Container-based virtualization which provides a lightweight operation than traditional virtualization.

In this paper, we integrated open source software to implement a complete set of edge computing architectures. Edge computing between IoT devices and the cloud establishes a relay station to process the data collected by the sensors and provide the most immediate preprocessing and response. We set up the Raspberry Pi on the Edge Gateway and Kubernetes minion on the Raspberry Pi to provide the service application, which contains Grafana [3], the open platform for analytics and monitoring. For short-term data storage, we use InfluxDB [4] as a data store for large amounts of time-series data.

In particular, our specific purpose is to deploy a complete set of cloud-edge computing architectures and IoT using container-based virtualization which constructs Kubernetes Minion (Nodes) in the Docker container service on the Edge side. In this case, we integrate Raspberry Pi 3+, DHT11 Humidity and Temperature Sensor, Grafana and InfluxDB in the Kubernetes and Docker environment.

The rest of the paper is organized as follows. Section 2 describes the background and related work. Section 3 presents the system architecture. Section 4 shows the experimental results. The last one, Sect. 5 provides a conclusion and the future work of this paper.

2 Background and Related Work

In this section, we provide several components that are approaching in this paper: Edge Computing, Docker, and Kubernetes.

2.1 Edge Computing

Edge computing is a method of optimizing cloud computing systems by performing data processing at the edge of the network, near the source of the data. IoT applications are ideally suited for edge computing architectures. Notably, in emerging IoT applications such as self-driving, drone, augmented reality (AR)/virtual reality (VR), and robotics. All of which are new applications that emphasize immediate of the image analysis and identification processing capabilities, low latency and high bandwidth requirements that need tens of microseconds or even milliseconds response time. In edge computing, the transfer of data to and from the cloud over the Internet can reach hundreds of milliseconds to respond time. Therefore these type of applications is very suitable to use the edge computing architecture.

2.2 Docker

Docker [5–8] is an open-source project that automates the deployment of applications inside software containers, by providing an additional layer of abstraction and automation of operating-system-level virtualization on Linux. Docker uses the resource isolation features of the Linux kernel such as cgroups and kernel namespaces, and a union-capable filesystem such as aufs and others to allow independent "containersto" run within a single Linux instance, avoiding the overhead of starting and maintaining virtual machines. The Linux kernel‘s support for namespaces mostly isolates an application‘s view of the operating environment, including process trees, network, user IDs and mounted file systems, while the kernel‘s cgroups provide resource limiting, including the CPU, memory, block I/O and network. Since version 0.9, Docker includes the libcontainer library as its way to directly use virtualization facilities provided by the Linux kernel, in addition to using abstracted virtualization interfaces via libvirt, LXC [9–11] (Linux Containers) and systemd-nspawn.

By using containers, resources can be isolated, services restricted, and processes provisioned to have an almost entirely private view of the operating system with their own process ID space, file system structure, and network interfaces. Multiple containers share the same kernel, but each container can be constrained to only use a defined amount of resources such as CPU, memory and I/O. Using Docker to create and manage containers may simplify the creation of highly distributed systems by allowing multiple applications, worker tasks, and other processes to run autonomously on a single physical machine or across multiple virtual machines. Docker enables the deployment of nodes to be performed as the resources become available or when more nodes are needed, allowing a platform as a service (PaaS)-style of deployment and scaling for systems like Apache Cassandra, MongoDB or Riak. Docker also simplifies the creation and operation of task or workload queues and other distributed systems. Docker architecture is different from virtual architecture.

2.3 Kubernetes

Kubernetes is particularly well-suited for microservices such architectures. Combining several containers into a single service, Kubernetes also provides an excellent service discovery mechanism for each service to communicate with one another. Most importantly K8S great programming can automatically expand services, and even for large-scale containers for rolling updates (Rolling update) and rollback (Rolling back/Undo), but also can integrate CI/CD and other DevOps tools, Absolutely allow users to manage an extensive system with the least effort [12].

2.4 Related Works

The Cisco introduced Fog Computing Systems as a new model to ease wireless data transfer to distributed devices in the Internet of Things (IoT) network paradigm. Cisco defines Fog Computing as a paradigm that extends Cloud computing and services to the edge of the network. Similar to Cloud, Fog provides data, compute, storage, and application services to end-users. The distinguishing Fog characteristics are its proximity to end-users, its dense geographical distribution, and its support for mobility. Services are hosted at the network edge or even end devices such as set-top-boxes or access points. By doing so, Fog reduces service latency, and improves QoS, resulting in superior user-experience. Fog Computing supports emerging Internet of Everything (IoE) applications that demand real-time/predictable latency (industrial automation, transportation, networks of sensors and actuators). Thanks to its wide geographical distribution the Fog paradigm is well positioned for a real-time big data and real-time analytics. Fog supports densely distributed data collection points, hence adding a fourth axis to the often mentioned Big Data dimensions (volume, variety, and velocity) [13].

China Venkanna Varma et al. [14] in 2016, studied the working of Docker networks, various factors of CPU context switch latency and how network IO throughput will be impacted with the number of live Docker containers. A Hadoop cluster environment built and executed benchmarks such as TestDFSIO-write and TestDFSIO-read against varying amount of the live containers. They observed that Hadoop throughput is not linear with increasing number of live container nodes sharing the same system CPU.

3 System Architecture

To develop the proposed solution by this paper, we needed the use of different types of software and hardware components. The used hardware was a Raspberry Pi 3 Model B+ with DHT11 temperature and humidity sensor. The details of hardware and software specification describe as following in Table 1: To get a general overview, Fig. 1 shows the scheme of the node architecture with all of the systems components. In the first step, we deploy the raspberry using kubernetes and docker with grafana installed and connected to the DHT11 sensor. Data captured by sensor then send to the InfluxDB using MQTT. Grafana uses a plugin to connect with InfluxDB in order to visualize the data in the graph and give the alert as our rule.

The overall device architecture is shown in Fig. 2, mainly using Arduino LoRa Shield module and DHT11 sensor as the overall sensing module and LoRa sensing

Table 1. Hardware and software specification

Category	Item	Specification	Description
Hardware	*Raspberry*	*OS*	Raspbian Jessie
		CPU	4 Cores, Broadcom BCM2837 64 bits
			ARMv8 Processor, 1.2 GHz
		Power	5V@2.4A(with, MicroUSB)
		Storage	*microSD*
		Network	Ethernet (RJ45)
		WiFi	BCM43143 WiFi
		Bluetooth	BCM43438 wireless LAN and
			Bluetooth Low Energy (BLE) on board
	Sensor	*DHT11*	Temperature and Humidity
Software	*Application*	*Grafana*	An open platform for analytics and monitoring
	Database	*InfluxDB*	A data store for large amounts of time series data

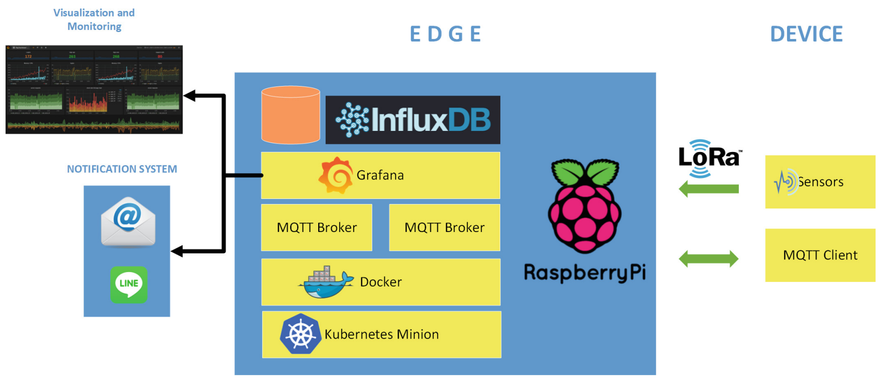


Fig. 1. The edge architecture

data collected by LoRa Nodes will be transmitted to LoRa Gateway through LoRa. In this process, LoRa offers the Low Power Wide Area Network (LPWAN) and the star-topology, and the LoRa Gateway is also connected to the miniature ramps made by the Raspberry Pi 3 for ultimate unified delivery to the data center(cloud).

4 Experimental Results

In this work, we build the system on campus with four LoRa nodes and a LoRa Gateway as shown in Fig. 3.

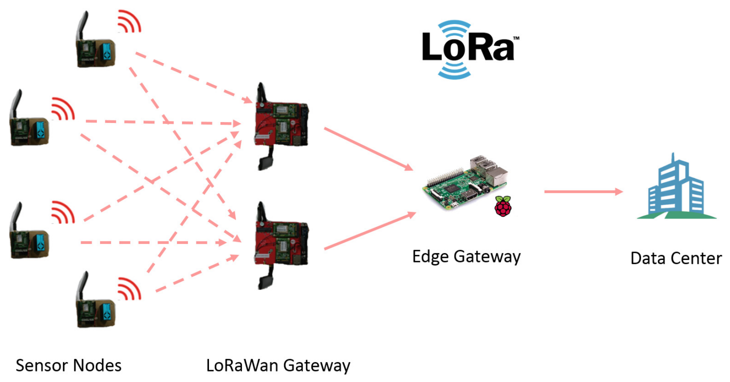


Fig. 2. The device architecture

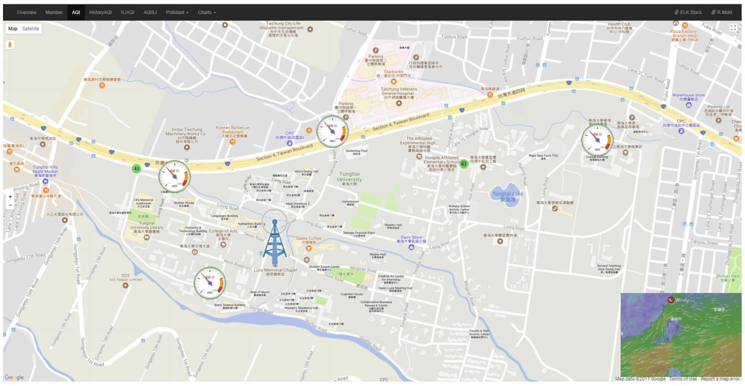


Fig. 3. LoRa Deployment

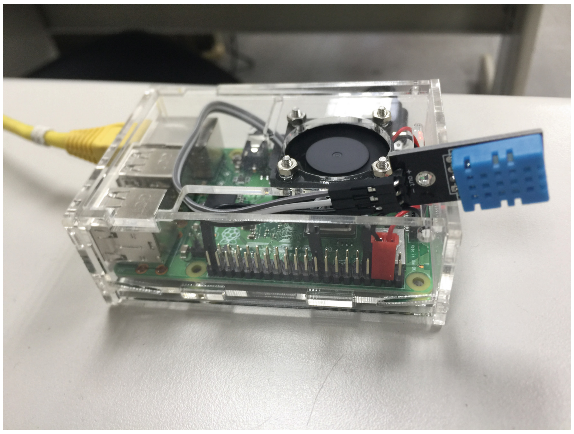
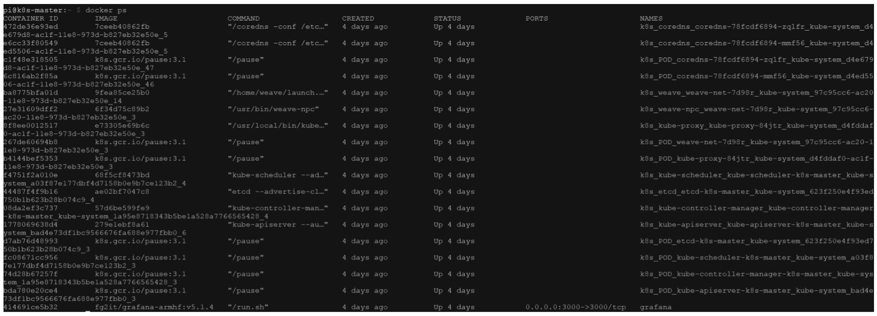


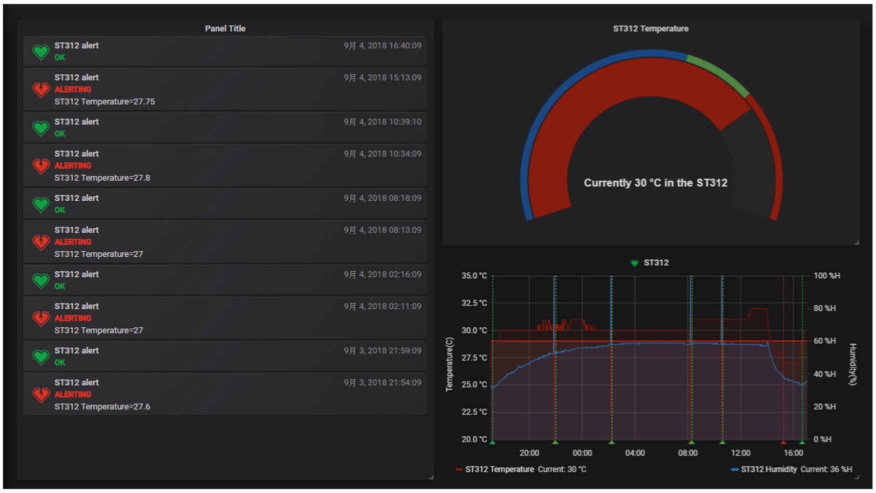
Fig. 4. Raspberry Pi integrated with DHT11

To implement an Edge Computing, first, we prepared Raspberry Pi integrated with DHT11 humidity and temperature sensor as shown in Fig. 4. Second, we set up docker container with Grafana and InfluxDB installed as shown in Figs. 5, 6 and 7.



```
pi@k8s-master:~ $ docker ps
CONTAINER ID        IMAGE                      COMMAND                  CREATED             STATUS              PORTS                    NAMES
472de36e93ed        7ceeb40862fb               "/coredns -conf /etc…"   4 days ago          Up 4 days                                    k8s_coredns_coredns-78fcdf6894-zqlfr_kube-system_d4
e679d8-ac1f-11e8-973d-b827eb32e50e_5
e6cc33f80549        7ceeb40862fb               "/coredns -conf /etc…"   4 days ago          Up 4 days                                    k8s_coredns_coredns-78fcdf6894-mmf56_kube-system_d4
ed5506-ac1f-11e8-973d-b827eb32e50e_5
c1f48e318505        k8s.gcr.io/pause:3.1       "/pause"                 4 days ago          Up 4 days                                    k8s_POD_coredns-78fcdf6894-zqlfr_kube-system_d4e679
d8-ac1f-11e8-973d-b827eb32e50e_47
6c816ab2f85a        k8s.gcr.io/pause:3.1       "/pause"                 4 days ago          Up 4 days                                    k8s_POD_coredns-78fcdf6894-mmf56_kube-system_d4ed55
06-ac1f-11e8-973d-b827eb32e50e_46
ba8775bfa01d        9fea85ce25b0               "/home/weave/launch.…"   4 days ago          Up 4 days                                    k8s_weave_weave-net-7d98r_kube-system_97c95cc6-ac20
-11e8-973d-b827eb32e50e_14
27e31609dff2        6f34d75c89b2               "/usr/bin/weave-npc"     4 days ago          Up 4 days                                    k8s_weave-npc_weave-net-7d98r_kube-system_97c95cc6-
ac20-11e8-973d-b827eb32e50e_3
8f8ee0012517        e73305e69b6c               "/usr/local/bin/kube…"   4 days ago          Up 4 days                                    k8s_kube-proxy_kube-proxy-84jtr_kube-system_d4fddaf
0-ac1f-11e8-973d-b827eb32e50e_3
267de60694b8        k8s.gcr.io/pause:3.1       "/pause"                 4 days ago          Up 4 days                                    k8s_POD_weave-net-7d98r_kube-system_97c95cc6-ac20-1
1e8-973d-b827eb32e50e_3
b4144bef5353        k8s.gcr.io/pause:3.1       "/pause"                 4 days ago          Up 4 days                                    k8s_POD_kube-proxy-84jtr_kube-system_d4fddaf0-ac1f-
11e8-973d-b827eb32e50e_3
f4751f2a010e        68f5cf8473bd               "kube-scheduler --ad…"   4 days ago          Up 4 days                                    k8s_kube-scheduler_kube-scheduler-k8s-master_kube-s
ystem_a03f87e177dbf4d7158b0e9b7ce123b2_4
44487f4f9b16        ae02bf7047c8               "etcd --advertise-cl…"   4 days ago          Up 4 days                                    k8s_etcd_etcd-k8s-master_kube-system_623f250e4f93ed
750b1b623b28b074c9_4
08da2ef3c737        57d6be599fe9               "kube-controller-man…"   4 days ago          Up 4 days                                    k8s_kube-controller-manager_kube-controller-manager
-k8s-master_kube-system_1a95e8718343b5be1a528a7766565428_4
1778069638d4        279e1ebf8a61               "kube-apiserver --au…"   4 days ago          Up 4 days                                    k8s_kube-apiserver_kube-apiserver-k8s-master_kube-s
ystem_bad4e73df1bc9566676fa688e977fbb0_6
d7ab76d48993        k8s.gcr.io/pause:3.1       "/pause"                 4 days ago          Up 4 days                                    k8s_POD_etcd-k8s-master_kube-system_623f250e4f93ed7
50b1b623b28b074c9_3
fc08671cc956        k8s.gcr.io/pause:3.1       "/pause"                 4 days ago          Up 4 days                                    k8s_POD_kube-scheduler-k8s-master_kube-system_a03f8
7e177dbf4d7158b0e9b7ce123b2_3
74d28b67257f        k8s.gcr.io/pause:3.1       "/pause"                 4 days ago          Up 4 days                                    k8s_POD_kube-controller-manager-k8s-master_kube-sys
tem_1a95e8718343b5be1a528a7766565428_3
bda780e20ce4        k8s.gcr.io/pause:3.1       "/pause"                 4 days ago          Up 4 days                                    k8s_POD_kube-apiserver-k8s-master_kube-system_bad4e
73df1bc9566676fa688e977fbb0_3
414691ce5b32        fg2it/grafana-armhf:v5.1.4 "/run.sh"                4 days ago          Up 4 days           0.0.0.0:3000->3000/tcp   grafana
```

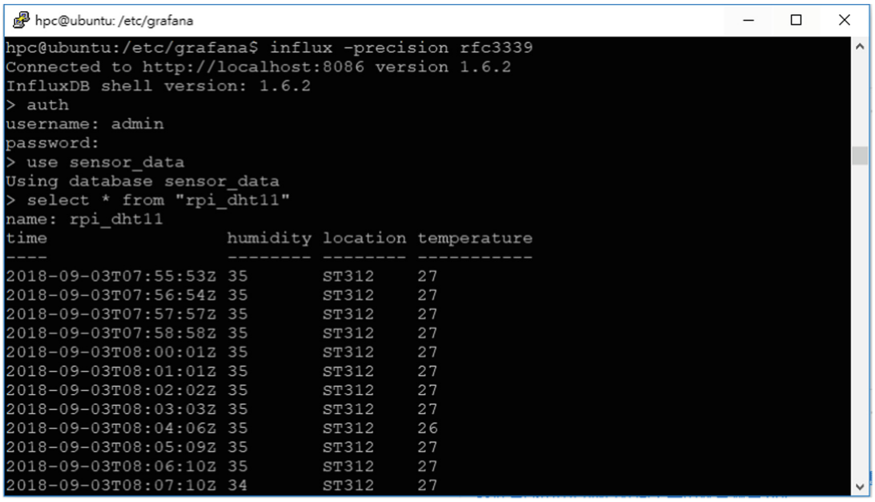
Fig. 5. Docker container



Panel Title

ST312 alert OK	9月 4, 2018 16:40:09
ST312 alert ALERTING ST312 Temperature=27.75	9月 4, 2018 15:13:09
ST312 alert OK	9月 4, 2018 10:39:10
ST312 alert ALERTING ST312 Temperature=27.8	9月 4, 2018 10:34:09
ST312 alert OK	9月 4, 2018 08:18:09
ST312 alert ALERTING ST312 Temperature=27	9月 4, 2018 08:13:09
ST312 alert OK	9月 4, 2018 02:16:09
ST312 alert ALERTING ST312 Temperature=27	9月 4, 2018 02:11:09
ST312 alert OK	9月 3, 2018 21:59:09
ST312 alert ALERTING ST312 Temperature=27.6	9月 3, 2018 21:54:09

Fig. 6. Grafana dashboard


hpc@ubuntu: /etc/grafana

```
hpc@ubuntu:/etc/grafana$ influx -precision rfc3339
Connected to http://localhost:8086 version 1.6.2
InfluxDB shell version: 1.6.2
> auth
username: admin
password:
> use sensor_data
Using database sensor_data
> select * from "rpi_dht11"
name: rpi_dht11
time                 humidity location temperature
----                 -------- -------- -----------
2018-09-03T07:55:53Z 35       ST312    27
2018-09-03T07:56:54Z 35       ST312    27
2018-09-03T07:57:57Z 35       ST312    27
2018-09-03T07:58:58Z 35       ST312    27
2018-09-03T08:00:01Z 35       ST312    27
2018-09-03T08:01:01Z 35       ST312    27
2018-09-03T08:02:02Z 35       ST312    27
2018-09-03T08:03:03Z 35       ST312    27
2018-09-03T08:04:06Z 35       ST312    26
2018-09-03T08:05:09Z 35       ST312    27
2018-09-03T08:06:10Z 35       ST312    27
2018-09-03T08:07:10Z 34       ST312    27
```

Fig. 7. InfluxDB

5 Conclusion

This paper integrates the technologies of Kubernetes, Docker, Grafana, InfluxDB, IoT, and LoRa to implement a complete set of edge computing architectures. From the experiment, we can prove that Edge Computing improves application performance such as analysis and monitoring system. By using Grafana and influxDB, we can monitor the movement of time-series data and send the alert based our rule automatically. In summary, the implementation of Edge Computing Architecture using container-based virtualization for real-time data streaming Processing provides significant improvement of application performance. By using this architecture, we can achieve lower latency levels on the edge, as opposed to a faraway cloud or data center.

Acknowledgment. This work was supported in part by the Ministry of Science and Technology, Taiwan R.O.C., under grants number 107-2221-E-029-008-.

References

1. Varghese, B., Buyya, R.: Next generation cloud computing: new trends and research directions. Futur. Gener. Comput. Syst. **79**, 849–861 (2018)
2. Kristiani, E., Yang, C.-T., Wang, Y.T., Huang, C.-Y.: Implementation of an edge computing architecture using openstack and kubernetes. In: Kim, K.J., Baek, N. (eds.) ICISA 2018. LNEE, vol. 514, pp. 675–685. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-1056-0_66
3. Grafana (2018). https://grafana.com/
4. Influxdb (2018). https://www.influxdata.com/

5. Špaček, F., Sohlich, R., Dulk, T.: Docker as platform for assignments evaluation. Energy Procedia, 1665–1671 (2015)
6. Build, ship and run any app, anywhere (2015). https://www.docker.com/
7. Docker (software) (2015). http://en.wikipedia.org/wiki/Docker%28software%29
8. Liu, D., Zhao, L.: The research and implementation of cloud computing platform based on docker. In: 2014 11th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), pp. 475–478 (2014)
9. Felter, W., Ferreira, A., Rajamony, R., Rubio, J.: An updated performance comparison of virtual machines and linux containers. In: 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp. 171–172 (2015)
10. Nakagawa, G., Oikawa, S.: Behavior-based memory resource management for container-based virtualization. In: Proceedings of 4th International Conference on Applied Computing and Information Technology, 3rd International Conference on Computational Science/Intelligence and Applied Informatics, 1st International Conference on Big Data, Cloud Computing, Data Science and Engineering, ACIT-CSII-BCD 2016, pp. 213–217 (2016)
11. Soltesz, S., Pötzl, H., Fiuczynski, M.E., Bavier, A., Peterson, L.: Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. In: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, pp. 275–287 (2007)
12. Kubernetes (2017). https://kubernetes.io/
13. Ahmed, E., Rehmani, M.H.: Mobile edge computing: opportunities, solutions, and challenges (2017)
14. China Venkanna Varma, P., Kalyan Chakravarthy, K.V., Valli Kumari, V., Viswanadha Raju, S.: Analysis of network IO performance in hadoop cluster environments based on docker containers. In: Pant, M., Deep, K., Bansal, J.C., Nagar, A., Das, K.N. (eds.) Proceedings of Fifth International Conference on Soft Computing for Problem Solving. AISC, vol. 437, pp. 227–237. Springer, Singapore (2016). https://doi.org/10.1007/978-981-10-0451-3_22