

## IMPROVEMENT OF SETUP TIME ON SERVER INFRASTRUCTURE AUTOMATION USING ANSIBLE FRAMEWORK

BANU SANTOSO<sup>1</sup>, MARTI WIDYA SARI<sup>2\*</sup>

<sup>1</sup>Department of Computer Engineering, Universitas AMIKOM Yogyakarta  
North Ring Road Condongcatur, Depok, Sleman, Yogyakarta, 55281, Indonesia

<sup>2</sup>Department of Informatics, Universitas PGRI Yogyakarta Jalan PGRI I No. 117  
Sonosewu, Yogyakarta, 55182, Indonesia

\*Corresponding Author: marti@upy.ac.id

### Abstract

System administrators must be prepared to quickly and accurately design the server infrastructure due to technology development in the server field this day. A way to manage servers is to use the Ansible framework, which can install and configure multiple servers at once. Ansible can make installing and configuring numerous servers easier than conventional (shell scripts), and future designs can be easily adjusted centrally. This study aims to manage servers automatically and measure setup time using the Ansible framework. The method used is to access Alibaba Cloud, then the sysadmin creates access\_key & secret\_key on the profile page. Once started, the Ansible script can be run using a terminal or cmd and receive server information such as Hostname, IP Address, and Web Address that can be accessed. This study indicates that the automation method using Ansible can shorten the time of the entire installation and configuration. The measurement results show that if using the conventional way, the average time needed for each server is 27 minutes 28 seconds. Using Ansible, the average time it takes is 3 minutes 31 seconds.

Keywords: Ansible, Automation, Infrastructure, Server, Shell Script.

## **1. Introduction**

Technology in the server field is experiencing very rapid growth [1]. It allows an administrator to handle more than one server [2-4], causing an administrator to experience difficulties in server management [2, 5]. Therefore, it is necessary to have a DevOps approach as a bridge between the developer (Dev) and the Ops (System) to make it easier to handle quickly and in control of the system to be developed [6-8]. The in-house server infrastructure for website creation carried out by DevOps still uses the traditional way, from buying a server to configuring servers for production, testing, and development done repeatedly [9, 10]. The processing time for this server infrastructure to be faster and more efficient is what the company hopes for [11]. So that we need a method so that the creation and configuration of servers can be done quickly, precisely, and efficiently [12].

Based on the problems above, it automatically takes a method to generate infrastructure using an Ansible application [13]. In the future, this method is expected to be used, with the aim of the person in charge of the server infrastructure to provide the need for developers to start a project [14-16]. DevOps can create one script through the Ansible framework, and then it can be run directly to many servers [17]. In addition, the centralized configuration makes Ansible easy to change according to the needs of any website application quickly [18, 19].

The Ansible framework used for the automation process can significantly speed up the server configuration process and be on target and simplify the multi-server configuration simultaneously [20]. The objective of this research is to improve the setup time of server infrastructure automation using Ansible Framework.

## **2. Literature Review**

### **2.1. State of the art**

The contribution of research results related to reducing setup time in cloud services, such as deploying server infrastructure, building server infrastructure, and building applications using automation that several previous studies have done, is presented in Table 1. The table contains studies - research in terms of researchers, location, research contribution, and methods.

Based on the literature study, different methods can be proposed as a positive contribution to this research. This is in the form of designing and creating automation scripts using Ansible and Alibaba Cloud API to improve the processing time of the server infrastructure and facilitate the rapid deployment of applications that are being developed to the server [24].

Katsuno and Takahashi [21] explained that the deployment time can be decreased by up to 40% with the parallel approach method on virtual machines using the Chef framework. The advantage of using Chef is that it can significantly reduce the spreading time of fish. The weakness of this research is that the implementation is not very User Friendly, which makes the time to learn this method very long.

Singh et al. [22] discussed Automated Provisioning of Applications in the IAAS Cloud using Ansible Configuration Management said that tools can now help DevOps build servers on AWS EC2 less than a minute compared to using the method. It can take a week to configure Hardware on AWS EC2. Automation can reduce infrastructure provisioning time from days to hours. The advantage of this research is

that it can reduce the time to build servers on AWS EC2 and help DevOps build server infrastructure. The drawback of this research is that DevOps must optimize the automation process before it can run because running automation in an un-optimized state can interfere with productivity.

Moreover, Juopperi [23] researched the deployment of applications to servers using Chatbots, namely ChatOps and Ansible. The method used in this study can reduce time and efficiency in the deployment of applications. The advantage of this study that uses ChatOps is visibility. Everyone on the team sees every action taken. Action history is also saved in chats. It provides a clear record of what has been deployed for production or testing across the team.

In addition, Garg and Garg [24] researched to reduce the time for continuous Integration and Sustainable Shipping, commonly known as CICD, using Docker with Ro-bust Container Security. This research also provides a strategy for the deployment and maintenance of automated resources to be managed effectively. The advantage of this research is that virtualization can reduce the time required to use these computing resources from days to minutes. The drawback of this research is that Docker requires extensive server resources, increasing the cost of renting a server with significant resources.

Furthermore, Poornalinga and Rajkumar [25] discussed an effective automation system that can help save time and costs in software quality and productivity. This research describes automation in application builders from source code using Maven and Jen-kins. This concept can reduce waiting time and find errors from the source code because Jenkins will check the source code and notify the developer of an error in the source code that Jenkins built. The advantage of this research is that the application building process can be completed in a short time. The drawback of this research is the lack of knowledge about sustainable integration tools and their environment.

**Table 1. Research overview.**

Researcher/Year	Location	Research Contribution	Methods
Katsuno and Takahashi [21]	Virtual Machine, IaaS	Reduces time on application deployment across IaaS services and Instances	Chef
Singh et al. [22]	Amazon AWS	Time savings in provisioning servers on AWS EC2	Ansible, AWS API
Juopperi [23]	Amazon AWS	Time efficiency and monitoring in CICD	ChatOps with Hubot, Ansible
Garg and Garg [24]	Google Cloud	Reduces time on Cloud Infra-structure, and CICD	Docker with Robust Container Security
Poornalinga and Rajkumar [25]	Amazon AWS	Reduces time on the CIDD infrastructure	Jenkins, Maven
Current research	Alibaba Cloud	Reduced time in server infrastructure deployment and configuration on Alibaba Cloud	Ansible, Alibaba Cloud API

## 2.2. Ansible

Ansible is an open-source automation tool for managing and configuring computers. Red Hat and the open-source community-developed Ansible. Ansible is designed to handle complex infrastructure rather than a single case. Dynamic inventory of instances, which is required when models come and go, is supported with additional packages for standard cloud providers.

The method used in this study is the Ansible method, as presented in Table 2. The reason for choosing to use the Ansible method compared to other methods is that it has advantages, including that it is easier in the setup process, Ansible management is also easy. It is cheap to use up to 100 nodes using Ansible Tower or free of charge using the AWX Cloud. In making the Ansible script, it is also easy to understand by System Administrators and DevOps because it uses the YAML Configuration File that is easy to understand and learn and administrator-oriented. Ansible distribution uses SSH so that the server setup does not require additional setup, and the server setup process is faster.

**Table 2. Comparison of automation software [26, 27].**

Classification	Chef	Puppet	Ansible	Saltstack
Availability	✓	✓	✓	✓
Setup Easiness	Difficult	Difficult	Easy	Difficult
Management	Difficult	Difficult	Easy	Easy
Scalability	Very measurable	Very measurable	Very measurable	Very measurable
Configuration Language	DSL (Ruby)	DSL (PuppetDSL)	YAML (Python)	YAML (Python)
Interoperability	High	High	High	High
Price (up to 100 node)	\$13700	\$11200-\$19900	\$10,000	\$15,000 (Estimation)

## 3. Research Methods

### 3.1. Materials

There are several tools and materials in the form of hardware and software used in this research. The materials and tools used will be described in the following subsections.

#### 3.1.1. Hardware

The hardware used in this study is as follows.

- CPU 1.6 GHz Intel Core i5-8250U
- Memory DDR 16 GB
- NVIDIA GeForce 940MX
- Linux Ubuntu Operating System version 16.04 LTS and Windows 10
- SSD storage with 250 GB capacity

#### 3.1.2. Software

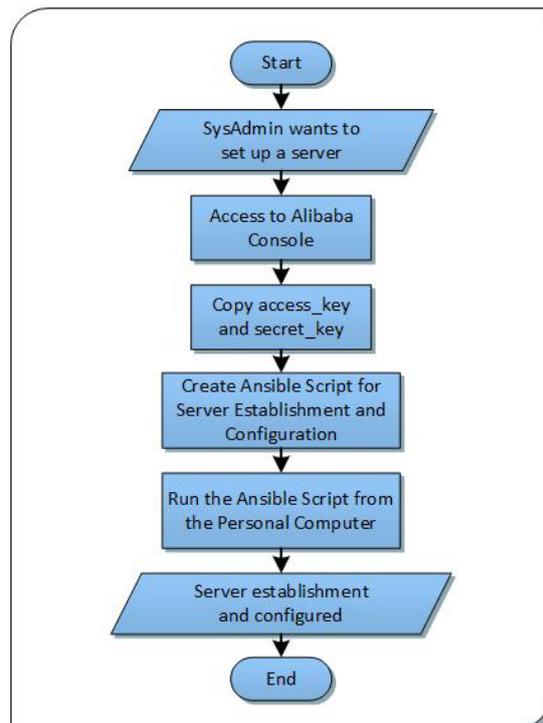
The software used in this study is as follows.

- Alibaba account to get access\_key and secret\_key and server instances.

- Nginx web server handles requests from the internet, which are processed by the backend and respond to the internet.
- PHP programming language to process requests requested by the Nginx web server and forwarded to the PostgreSQL database. After that, it will be forwarded back to the Nginx web server.
- Composer is a PHP-specific dependency manager with functionality like Gem (Ruby) or Maven (Java). The composer library will automatically install the other libraries needed without the need to download one by one.
- The PostgreSQL database is a robust open-source object-relational database system. The PostgreSQL database has over 15 years of active development and a proven architecture that has earned a strong reputation for reliability, data integrity, and correctness.

### 3.2. Research flow

Figure 1 is a flowchart performed by the system administrator if using Ansible. In Fig. 1, it can be explained, before setting up server automation, you must have an account from the selected cloud, here using Alibaba Cloud. If you already have an Alibaba Cloud account, the sysadmin creates access\_key & secret\_key on the profile page. This key is used to access, create, delete services from Alibaba Cloud. After the sysadmin gets the access\_key and secret\_key, the sysadmin can generate an Ansible script for automated server creation. Once created, the Ansible script can be run using a terminal or cmd and receive server information such as Hostname, IP Address, and Web Address that can be accessed.



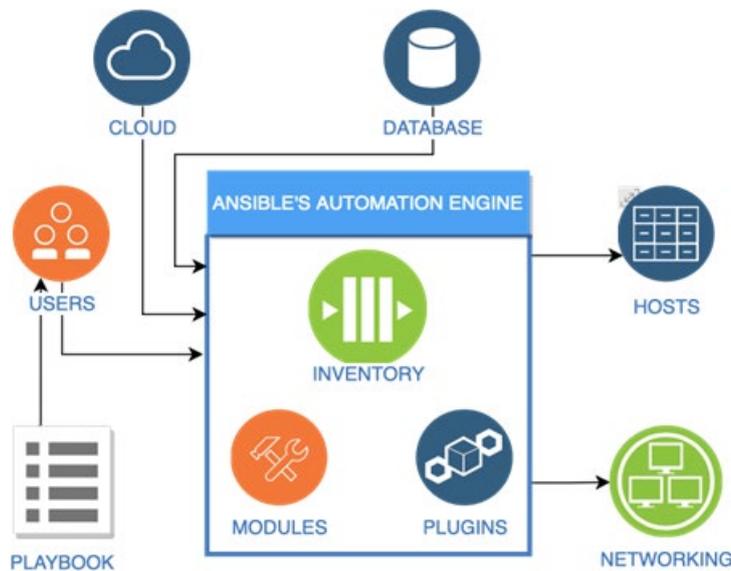
**Fig. 1. Research flowchart.**

The IP address obtained from creating the server is used to automate server configuration, such as software installation, design, and fetching project data on GitHub, GitLab, etc.

Alibaba account is used to get access\_key and secret\_key and also create Server Instance. Then internet connection is used to connect User laptop devices (Administration System/DevOps) to Alibaba website and also Server configuration. The terminal is used to run Ansible scripts, and the browser is used to visit the Alibaba website and enter the console to get the access\_key and secret\_key. The Code Editor is used to create and modify Ansible configurations.

### 3.3. System design

This system design block diagram can be seen in Fig. 2. The system for improving setup and configuration times in automating server infrastructure using the Ansible method.



**Fig. 2. System design.**

Things that need to be prepared when automating the Cloud server include Server Accounts, namely Access\_Key and Secret\_Key. After obtaining Access\_Key and Secret\_Key, the System Administrator can develop Script Ansible to automate the Server Infrastructure. The Ansible script can be run using a terminal command prompt to get server information in the form of an IPv4 or IPv6 Address, Hostname, and Web Address (Domain Name Server). The IP address information on the server is used to automate server configuration, including the installation of required software, design and data retrieval on Gitlab, Github, and others. The Playbook consists of several collections of roles with specific tasks for working on modules installed and configured. The Module is the default application that is used to establish or prepare several applications. Plugins are an extension from Ansible so that Ansible can run as desired. At the same time, the Inventory is a collection of lists of addresses or IP address servers directed to be configured automatically on the server.

## 4. Results and Discussion

### 4.1. Comparison of processing time and hardware specifications

Testing is done by testing the script made on the server and testing using the traditional method. The test results on the script and traditional include measuring the time of Setup to servers that are made with different specifications. For testing, the script starts when DevOps runs the script until the resulting output is that the server is installed, and configuration can display the web application. In contrast, for traditional, it starts when DevOps enters the server using SSH until the resulting output, namely the configured server, can display web applications. The results of the server setup time measurement on Ansible are presented in Table 3 and Fig. 3. Experiments on Ansible were carried out ten times, with the different numbers of CPU cores and RAM sizes. The CPU cores used are 1, 2, 4, 8, 16 and 32, while the RAM sizes used are 1 GB, 2 GB, 4 GB, 8 GB, 16 GB and 32 GB.

**Table 3. Results of the experiment on ansible.**

Num ber of CPU Core	RA M (GB)	Experiment									
		1	2	3	4	5	6	7	8	9	10
1	1	3:48 .92	3:42 .58	4:34 .22	4:34 .22	4:33 .45	4:06 .54	4:15 .09	4:11 .17	4:18 .13	4:32 .17
1	2	3:45 .40	5:20 .41	4:28 .91	4:44 .09	4:43 .97	4:56 .15	3:28 .01	3:54 .13	4:07 .16	3:51 .72
2	4	3:08 .66	3:46 .64	3:41 .40	4:49 .31	4:36 .48	4:52 .15	4:24 .59	4:17 .15	4:16 .94	3:58 .38
4	4	3:10 .93	3:34 .47	3:27 .44	4:00 .62	4:20 .89	5:07 .88	4:25 .87	4:47 .74	4:10 .10	3:50 .23
4	8	3:35 .73	3:28 .43	3:32 .61	3:22 .60	3:34 .68	3:52 .52	3:28 .94	3:36 .01	3:48 .34	3:51 .44
4	16	3:05 .53	3:02 .11	2:58 .46	3:12 .79	3:10 .55	3:01 .27	3:21 .24	3:01 .93	3:30 .23	3:02 .47
8	8	2:58 .43	2:55 .23	3:10 .45	3:04 .92	2:59 .47	3:15 .72	2:55 .29	3:08 .63	3:22 .98	3:01 .52
8	16	2:49 .40	3:28 .94	3:27 .44	2:50 .43	2:45 .22	2:48 .26	2:53 .83	3:01 .69	2:47 .48	2:49 .22
16	16	2:45 .88	2:40 .59	2:50 .63	2:55 .91	2:42 .19	2:47 .28	3:01 .57	2:59 .42	2:41 .85	2:58 .97
32	32	2:39 .67	2:44 .13	2:35 .05	2:48 .70	2:39 .71	2:45 .97	2:52 .73	2:35 .83	2:32 .68	2:39 .32

The results of the server setup time measurement in the traditional way are presented in Table 4 and Fig. 4. Experiments on the traditional way were carried out ten times, with different CPU cores and RAM sizes. The CPU cores used are 1, 2, 4, 8, 16 and 32, while the RAM sizes used are 1 GB, 2 GB, 4 GB, 8 GB, 16 GB and 32 GB.

The Traditional method's average setup time is 25 minutes 34 seconds to 30 minutes 29 seconds, while using Ansible is 2 minutes 41 seconds to 4 minutes 15 seconds. From ten experiments, the average, the longest time (average), fastest time (average), and standard deviation can be taken, which are presented in Table 5.

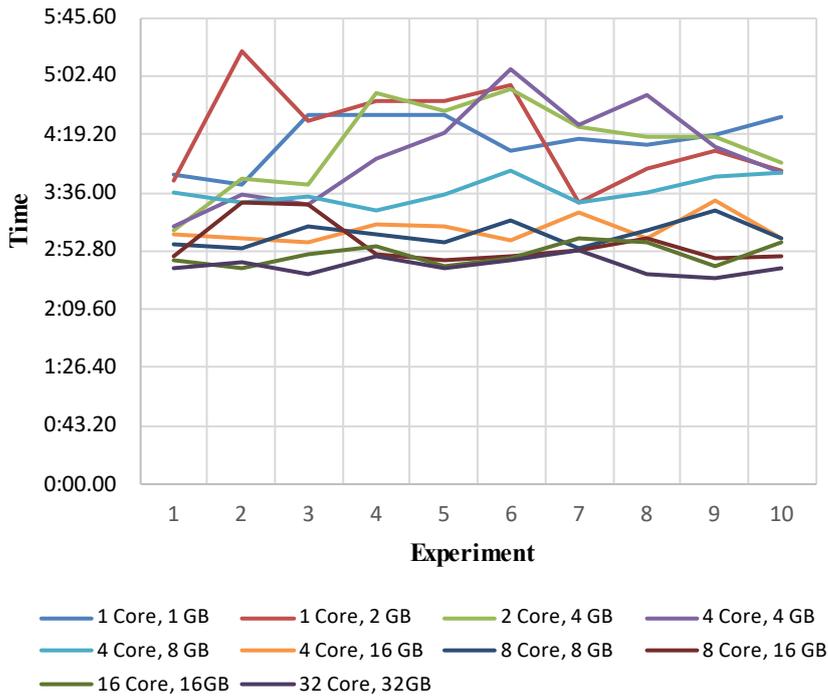


Fig. 3. Comparison of experiment time on ansible.

Table 4. Results of the experiment in the traditional way.

Num ber of CPU Core	RAM (GB)	Experiment									
		1	2	3	4	5	6	7	8	9	10
1	1	31:3	30:4	29:3	28:5	29:0	31:5	31:5	31:5	29:5	29:2
		8.89	3.65	8.29	3.23	1.68	8.83	4.22	1.13	6.49	1.02
1	2	26:0	26:2	29:3	27:1	29:2	29:4	27:4	26:3	29:0	26:2
		8.96	6.35	6.94	9.75	5.33	0.99	5.78	1.82	5.88	3.28
2	4	25:4	29:5	26:4	28:0	28:4	28:2	25:3	29:3	25:1	27:5
		7.45	0.12	4.27	2.26	3.13	8.68	6.28	1.12	7.90	7.83
4	4	27:3	28:4	28:5	27:1	28:0	28:4	28:1	27:2	29:0	27:2
		7.24	3.23	7.53	9.65	9.82	8.77	8.20	3.07	1.80	9.13
4	8	26:1	29:1	28:4	26:5	29:4	28:5	27:3	29:0	26:2	26:5
		2.75	9.40	6.65	6.84	0.73	8.70	3.29	6.71	3.21	6.19
4	16	27:3	28:2	27:1	26:5	28:3	25:0	25:0	25:5	26:5	27:5
		7.66	6.67	7.66	3.53	3.22	2.86	9.18	2.32	9.33	0.84
8	8	27:3	26:4	27:4	28:4	27:4	25:5	25:0	25:5	28:0	25:1
		1.50	2.78	5.02	6.05	5.13	3.59	2.69	2.50	9.24	1.82
8	16	27:3	24:1	26:1	26:3	23:1	28:1	23:2	28:5	28:3	25:3
		3.77	6.76	1.19	0.98	3.62	0.87	9.45	9.82	3.48	7.66
16	16	24:4	28:4	27:3	26:3	29:5	23:4	29:2	22:1	28:4	28:0
		8.79	8.46	9.18	3.67	7.77	2.13	8.53	5.62	9.38	8.07
32	32	27:3	29:0	23:2	26:4	25:3	26:5	22:4	21:1	28:4	23:3
		7.53	7.76	8.88	1.97	9.21	3.76	8.40	0.76	2.61	3.78

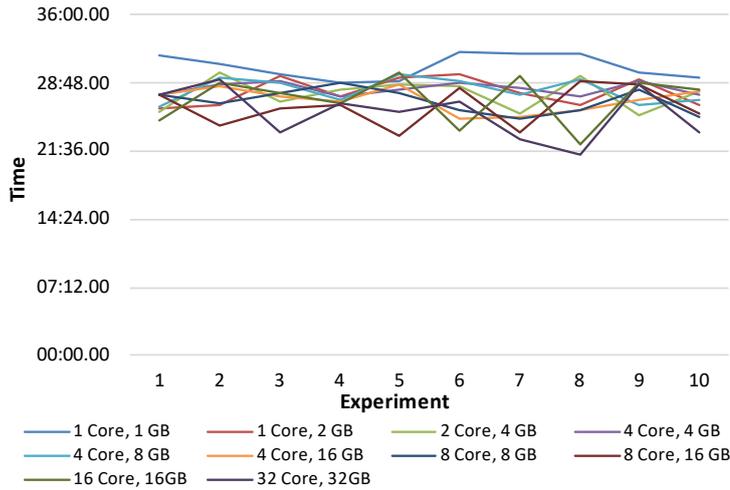


Fig. 4. Comparison of experiment time in the traditional way.

Table 5. Comparison of the setup time of ten experiments.

Information	Ansible	Traditional
Average	03:31,35	27:28,82
Fastest Time (avg.)	03:04,50	25:03,80
Longest Time (avg.)	04:00,36	29:33,71
Deviation Standard	00:18,46	01:33,03

#### 4.2. Server setup time

The comparison chart of server setup time between Ansible and traditional is presented in Fig. 5. In this graphic image, the traditional setup time is above 21 minutes and under 40 minutes, while the Setup using Ansible is above 2 minutes and under 6 minutes.

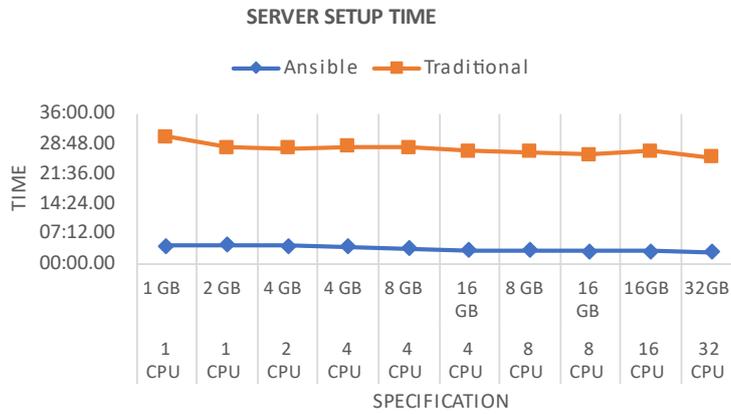


Fig. 5. Server setup time.

In testing the setup time using the traditional method, there are problems when configuring, so there is no significant reduction in time even though using high server hardware. Ansible, meanwhile, shows the decreasing time on different server hardware.

## 5. Conclusion

Based on the formulation of the problem about the length of time to create a VPS server setup on Alibaba Cloud, it can be concluded that making a server on Alibaba through the console from the Alibaba website takes a long time, an average of 27 minutes. In this study, an automation method is offered through Ansible to make servers on Alibaba can be made in one script with a relatively faster time with an average time of 3 minutes 30 seconds. Automation methods using Ansible can significantly increase installation and configuration times on multiple servers. In this study, trials were carried out and experienced a reduction in more than 5 hours in the traditional way. By way of Ansible, it took less than 1 hour. The test results show that the higher the server specifications on Alibaba are used, the faster the Ansible method's setup process. Future work can discuss the use of Ansible to automate the configuration process on the server and can be used to configure the server for load balancing. In addition, the Ansible Script to automate server creation on Alibaba can be made to be adapted to other Cloud Providers, for example, Amazon Web Service or Microsoft Azure.

## References

1. Kheirabadi, C.; and Groulx, D. (2016). Cooling of server electronics: A design review of existing technology. *Applied Thermal Engineering*, 105, 622-638.
2. Ul haq, I.; Wang, J.; Zhu, Y.; and Maqbool, S. (2021). An efficient hash-based authenticated key agreement scheme for multi-server architecture resilient to key compromise impersonation. *Digital Communications and Networks*, 7(1), 140-150.
3. Liu, H.; Bao, C.; Xie, T.; Gao, S.; Song, X.; and Wang, W. (2019). Research on the intelligent diagnosis method of the server based on thermal image technology. *Infrared Physics & Technology*, 96, 390-396.
4. Padhy, N. (2021). An automation API to optimize the rate of transmission using relone from local system to cloud storage environment. *Materials Today: Proceeding*, 37(Part 2), 2462-2466.
5. Ali, E.; Susandri; and Rahmaddeni. (2015). Optimizing server resource by using virtualization technology. *Procedia Computer Science*, 59, 320-325.
6. Smeds, J.; Nybom, K.; and Porres, I. (2015). DevOps: A definition and perceived adoption impediments. In: Lassenius, C., Dingsøyr, T., Paasivaara, M. (eds) *Agile Processes in Software Engineering and Extreme Programming. XP 2015. Lecture Notes in Business Information Processing*, vol 212. Springer, Cham. 212, 166-177.
7. Babar, Z.; Lapouchnian, A.; and Yu, E. (2015). Modeling DevOps deployment choices using process. In: Ralyté, J., España, S., Pastor, Ó. (eds) *The Practice of Enterprise Modeling. PoEM 2015. Lecture Notes in Business Information Processing*, vol 235. Springer, Cham, 322-337.
8. Lwakatare, L.E.; Kilamo, T.; Karvonen, T.; Sauvola, T.; Heikkilä, V.; Itkonen, J.; Kuvaja, P.; Mikkon, T.; Oivo, M.; and Lassenius, C. (2019).

- DevOps in practice: A multiple case study of five companies. *Information and Software Technology*, 114, 217-230.
9. Forti, S.; Ferrari, G.-L.; and Brogi, A. (2020). Secure cloud-edge deployments with trust. *Future Generation Computer Systems*, 102(2020), 775-788.
  10. Arcangeli, J.-P.; Boujbel, R.; and Leriche, S. (2015). Automatic deployment of distributed software systems: Definitions and state of the art. *The Journal of Systems and Software*, 103, 198-218.
  11. Lloyd, W.; Pallickara, S.; David, O.; Lyon, J.; Arabi, M.; and Rojas, K. (2013). Performance implications of multi-tier application deployments on Infrastructure-as-a-Service clouds: Towards performance modeling. *Future Generation Computer Systems*, 29, 1254-1264.
  12. Tso, F.P.; Jouet, S.; and Pezaros, D.P. (2016). Network and server resource management strategies for data centre infrastructures: A survey. *Computer Networks*, 106, 209-225.
  13. Masek, P.; Stusek, M.; Krejci, J.; Zeman, K.; Pokorný, J.; and Kudlacek, M. (2018). Unleashing full potential of ansible framework: University labs administration. *Proceedings of the 2018 22nd Conference of Open Innovations Association (FRUCT)*, Jyväskylä, Finland, 144-150.
  14. Dalla, S.D.; Di Nucci; and Tamburri, D.A. (2020). SoftwareX AnsibleMetrics : A Python library for measuring Infrastructure-as-Code blueprints in Ansible. *SoftwareX*, 12, 100633.
  15. Pribiš, R.; Beňo, L.; and Drahoš, P. (2019). Implementation of micro embedded OPC unified architecture server-client. *IFAC-PapersOnLine*, 52(27), 114-120.
  16. Martinez, J.; Dasari, D.; Hamann, A.; Sañudo, I.; and Bertogna, M. (2020). Exact response time analysis of fixed priority systems based on sporadic servers. *Journal of System Architecture*, 110, 101836.
  17. Mazumdar, S.; and Pranzo, M. (2017). Power efficient server consolidation for Cloud data center. *Future Generation Computer Systems*, 70, 4-16.
  18. Kumar, R.; and Goyal, R. (2020). Modeling continuous security: A conceptual model for automated DevSecOps using open-source software over cloud (ADOC). *Computers & Security*, 97, 101967.
  19. Keupondjo, G.A.S.; Anoh, N.G.; Adepo, J.C.; and Oumtanaga, S. (2019). Hybrid routing with latency optimization in SDN networks. *Journal of Engineering Science and Technology (JESTEC)*, 14(5), 3062-3072.
  20. Yiran, W.; Tongyang, Z.; and Yidong, G. (2018). Design and implementation of continuous integration scheme based on Jenkins and Ansible. *Proceedings of the 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*, Chengdu, China, 245-249.
  21. Katsuno, Y.; and Takahashi, H. (2015). An automated parallel approach for rapid deployment of composite application servers. *Proceeding of 2015 IEEE International Conference on Cloud Engineering*, Tempe, AZ, USA, 126-134.
  22. Singh, N.K.; Thakur, S.; Chaurasiya, H.; and Nagdev, H. (2016). Automated provisioning of application in IAAS cloud using Ansible configuration management. *Proceeding of 2015 1st International Conference on Next Generation Computing Technologies (NGCT)*, Dehradun, India, 81-85.

23. Juopperi, M. (2017). *Deployment automation with ChatOps and Ansible*. Bachelor of Engineering Information Technology. Helsinki Metropolia University of Applied Sciences.
24. Garg, S.; and Garg, S. (2019). Automated cloud infrastructure, continuous integration and continuous delivery using docker with robust container security. *Proceeding of the 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, San Jose, CA, USA, 467-470.
25. Poornalinga, K.S.; and Rajkumar, P. (2016). Continuous integration, deployment and delivery automation in AWS cloud infrastructure. *International Research Journal of Engineering and Technology*, 3(5), 426-431.
26. Zhang, G.; and Ravishankar, M.N. (2019). Exploring vendor capabilities in the cloud environment: A case study of Alibaba cloud computing. *Information and Management*, 56(3), 343-355.
27. Hochgeschwender, N.; Biggs, G.; and Voos, H. (2018). A reference architecture for deploying component-based robot software and comparison with existing tools. *Proceeding of the 2018 Second IEEE International Conference on Robotic Computing IRC 2018*, Laguna Hills, CA, USA, 121-128.