Configuration and Performance Management of Ansible to Deploy CMS Platform and Development of Odoo Server Automation Engine

Muharman Lubis, Adityas Widjajarto, Amalia Fiqhiyah, Raihanah I. Ramadhani, Arif Ridho Lubis

***Abstract*— In container technology, the deployment process becomes more complicated, time-consuming, and has extensive resources, hence, there is a necessity for a faster and more practical process. A very familiar one used among IT engineers and technicians is Kubernetes. Meanwhile, the configuration management tool for deployment that widely known is Ansible, as an innovation machine that enables automation for cloud availability, configuration management, application deployment, intra-service orchestration, and other IT needs. This study analyses the performance of Ansible configuration management tools to deploy a Content Management System (CMS) platform that is WordPress and MySQL using Ubuntu 16.04 LTS as a management node and target deployment. The parameters used in this test are measured in terms of the time interval, CPU usage, memory usage, and configuration management. It also analyzed the deployment of the Ansible automation engine to develop Odoo Server using the Google Cloud Platform (GCP). The measurements of the Ansible configuration showed that nodes do not affect the time interval. On CPU usage, the more the number of nodes, the smaller the CPU usage. The memory (RAM) usage on the number of different nodes indicates that the results are unstable. Meanwhile, this study also compares the Ansible automation engine between manual and automation development.**

Keywords—configuration, performance, platform, development, automation

I. INTRODUCTION

TECHNOLOGICAL advances have evolved along with times, which can provide ease and effectiveness in daily activities [1-3]. One of the technologies that are currently increasingly in demand providing the benefit in term of efficiency, ease of use and maintenance, which is widely known as cloud computing [4]. This is a natural progression for the widespread adoption of multiple technological developments in distributed computing, including virtualization, grid computing, involuntary computing, utility computing and software as a service [5]. On the other hand, virtualization represents a trend for IT companies to focus on management tasks while improving scalability and workloads [6]. Interestingly, it is a combination of involuntary computing [7] while IT environments manage themselves based on perceived activity through facility computing [8], with the utility provided excessively by the computer processing power based on the client payment as needed [9].

The development of cloud computing technology is now widely used because it does not require a lot of energy through the automation of configuration management [10]. Research [11] has been conducted with the Smart Cloud-based Optimizing Workload (SCOW) Model, which uses predictive cloud computing capacity to consider continuous factors to assign tasks to heterogeneous clouds and achieve optimization goals. Also, a combination of several Workload Resource Minimization Algorithm (WRM) algorithms, Smart Task Assignment (STA) Algorithm, and Task Mapping Algorithm (TMA) was carried out.

The automation configuration management breakthroughs in the IT world is the concept of changing the process that was initially manual to automatic, and the processes in the configuration step combined into one process [12]. This new concept is expected to decrease the number of human configuration errors [13]. Therefore, WordPress as a higher service in server needs also directly proportional to the service needs [14]. Increasingly high server requirements make server management must be fast in installation and maintenance [15]. Thus, Ansible is the solution to easier server maintenance and management [16], if compared to the manual method conducted with many processes; this automated method is more efficient because performed with a single process.

The problem that arises is the frequent occurrence of human errors and the time needed to configure tends to be longer than usual. In this study, it will analyze the performance of an application deployment process. The configuration implemented is monitored with the architecture on the server towards the time interval, CPU usage, memory usage, and

Muharman Lubis, School of Industrial Engineering Telkom University Bandung, Indonesia. (e-mail: muharmanlubis@telkomuniversity.ac.id).

Adityas Widjajarto, School of Industrial Engineering Telkom University Bandung, Indonesia. (e-mail: adtwjrt@telkomuniversity.ac.id).

Amalia Fiqhiyah, School of Industrial Engineering Telkom University Bandung, Indonesia. (e-mail: amelfqhyh@gmail.com).

Raihanah Ismah Ramadhani, School of Industrial Engineering, Telkom University Bandung, Indonesia (e-mail: raihanahismahr@student.telkomuniversity.ac.id).

Arif Ridho Lubis, Department of Computer Engineering and Informatics, Politeknik Negeri Medan, Medan, Indonesia (e-mail: arifridho@polmed.ac.id)

configuration management. It will also discuss how to convert infrastructure processes from manual to automatic.

II. RESEARCH METHOD

The system built is cases implementation in the form of testing. In designing the system for testing, several testing instruments are needed to support the testing scenarios, namely physical and program instruments. Physical instruments are hardware testing tools to support scenarios, which the one that are used can be seen in Table 1. Meanwhile, the following Table 2 is a list of information that explain the details of the hardware used in making the Ansible automation test flow design. On the other hand, it also describes the model or series and specifications of the physical instruments used. On the other hand, the devices used are adjusted to the simulation needs, which each specification can vary and adapt to the requirements. Meanwhile, the topology of testing is an initial description created to conduct test scenarios based on the system design to suit the needs of the following scenario. In the topology for testing, two things must be considered: physical topology and IP address allocation. The physical connection is used as a basis for testing, which is explained in the Fig. 1 while Fig. 2 depicts the architecture within the Kubernetes.

TABLE I
PHYSICAL INSTRUMENT OF CMS PLATFORM DEPLOYMENT

Device	Model/Series	Specification
Laptop	Apple Macbook Air 13" – A1932	1. Processor: Intel Core i5 dual-core 1,6 GHz, Turbo Boost up to 3,6 GHz, with cache L3 of 4 MB 2. Memory: 8 GB LPDDR3 2133 MHz 3. Graphics: Intel UHD Graphics 617; external graphics (eGPU) capable Thunderbolt 3 4. Storage: SSD 128GB

TABLE II
PHYSICAL INSTRUMENT OF CMS PLATFORM DEPLOYMENT

Component	Specification	Function
Hardware	1. Processor: Intel Core R i5-6200U 2.3Ghz up to 2.8GHz (3MB Cache) 2. Memory: 8GB RAM 3. Hard Disk: 1 TB 4. System Type: 64-bit Operating System 5. System Model: X456U 6. Operating System: Windows 10 Pro (10.0, Build 18362) 7. Dual Boot: Ubuntu Xenial Xerus	As Ansible Server
Virtual Machine	1. Processor: Intel Core R i5-6200U 2.3Ghz up to 2.8GHz (3MB Cache) 2. Memory: 8 GB RAM 3. Hard Disk: 20 GB 4. System Type: 64-bit Operating System 5. System Model: X456U 6. Operating System: Ubuntu Xenial Xerus 7. Version: 16.04.1	As Odoo Server

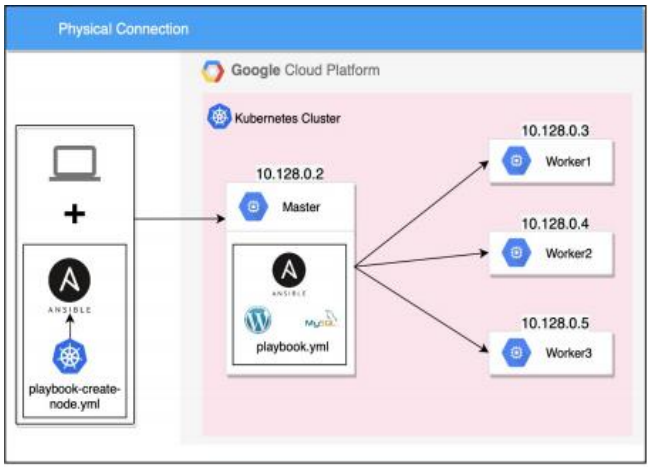

Fig. 1: Physical Connection

The test scenario is based on the physical connection in the Fig. 1 above to test the deployment process's time interval and measure resource usages, such as CPU usage and memory usage. The Fig. 2 explained two Ansible on the local computer and Ansible on the cloud server in physical topology. Ansible on the local computer is used to deploy the Kubernetes cluster to host instances on the cloud server. Meanwhile, Ansible on master-node is used for WordPress and MySQL deployment after the successful Kubernetes deployment on a local computer. Kubernetes cluster is divided into two categories, such as Kubernetes master and Kubernetes nodes or workers, in which the master manages the nodes.

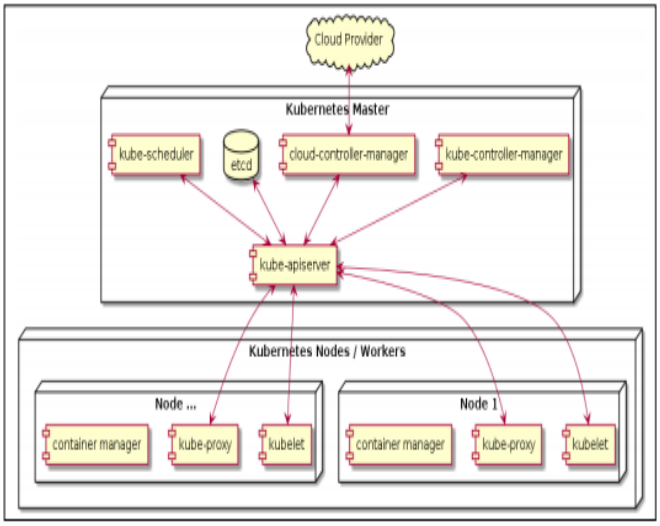

Fig. 2: Kubernetes Architecture

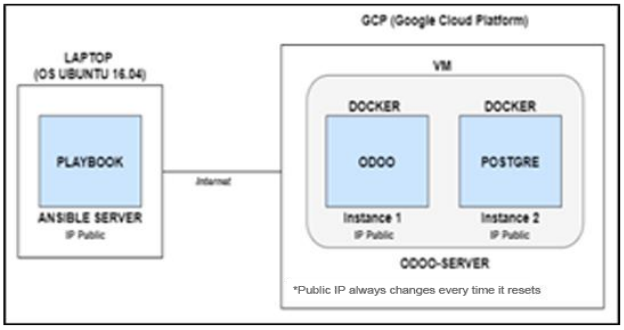

Fig. 3. Automation Logical Architecture

The Fig. 3 is the logical architecture of the automation processes, which is used in this case with two instances of Google Cloud Platform and one laptop with Ubuntu OS in the logical structure. For instance, one will use Odoo and Docker, while Instance 2 will use PostgreSQL and docker databases. Both instances are in one Odoo-Server, and on the laptop; it will become the Ansible server and creating a Playbook that can directly access the VM (Instance 1 and Instance 2) to perform automation while Docker serves as a container for Odoo and PostgreSQL. Ansible measurement flow starts from the installation process on the local computer (MacOS). After successful installation, it creates instances on the GCP cloud server, consisting of masters and workers.

Users can access and modifying various settings and files on the server using SSH. It is necessary to generate the existing SSH key on the cloud server instances and then put the RSA local computer's public key to each cloud server instance. Next, it will add a server that Ansible will target by writing it in the host's file. After putting the hosts on Ansible Local lists, the command will deploy the Kubernetes cluster on the cloud server. After successful deployment, it then installs Ansible on the master node cloud server to deploy MySQL and WordPress by adding kubeadm hosts on the Ansible master cloud server. Then, Ansible will be seen on the CPU and Memory usage and the comparison of time intervals on the cloud server. Ansible automation test flow is using the netdata as a tool to see CPU and Memory while Bandwidth measurements will be analyzed on Ansible cloud test and command time tools to monitor the running time required by the system. The flow starts with preparing the target for testing, then runs Ansible by running a playbook to perform Ansible automation.

III. RESULTS AND DISCUSSION

This chapter discusses the analysis of the test results using predefined scenarios and parameters. The test is carried out by implementing Ansible configuration management tools using a server control running on Ubuntu 16.04 LTS (Fig. 4). A GCP cloud server deployment simulation is run to check the performance of the Ansible Configuration Manager tool. Indeed, the platform is always online and needs to provide robust service to all customers while shutdown and restart operations for maintenance purposes must be properly organized to prevent data corruption and loss [17]. Having the proper response process can improve customer satisfaction as a result [18].

Automated testing gives software testers an easy way to automate the software testing process so it is most efficient when it comes to time, cost, and usability [19-20]. Each test is carried out five times and the average value of each parameter is calculated. Each cloud server will be reset if the Ansible deployment process has been completed to get optimal results. Meanwhile, each measurement's calculation results are included to ensure that the results obtained are statistically significant. The deployment time interval is the time that Ansible has taken to deploy commands to all nodes. This test is carried out five times to get time data from Ansible spread by using the command on the variation of the number of nodes as many as one node, two nodes and three nodes.

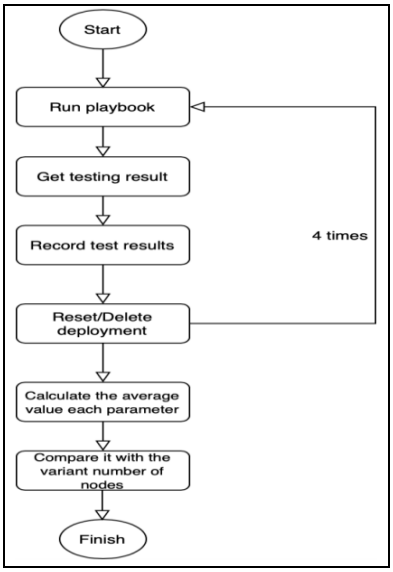

Fig. 4: Result of Analysis

The test results of deployment time interval on the cloud server can be seen in Fig. 5, which shows that the time interval of cloud deployment between one node, two nodes, and three nodes tends to fluctuate where unstable results occur to change. For one node, it needs the average value of time interval is 9.6s, two nodes need the average value of time interval is 11.6s, and three nodes need the average value of time interval is 10s. This inconsistency is caused by the internet network speed available on the local computer network connection. On the other hand, the Central Processing Unit (CPU) also called the processor is the computer hardware to carry out commands and process data from the software. When the program is executed or processed, the contents of the hard disk program are recovered and stored in the RAM.

The control unit distinguishes between instructions and data. Instructions are placed in Program Storage while data are placed in Work-Storage. In addition, the instructions and data obtained by the controller are stored in the records. When an instruction calculates logic or arithmetic, it is sent to ALU (Arithmetic Logic Unit) for processing. The result is stored in the complex. The console then retrieves it and puts it back into the RAM and displays it on an output device such as a screen. A node's CPU utilization is the number of CPU cores used in a node by all the pods running on that node. Measuring CPU usage using the highest commands, which aims to determine how much Ansible is using the CPU after running the command Ansible.

The results of testing using the top command can be seen in Fig. 6. If the CPU usage value is lower, the better the performance ratio with the difference in CPU usage can be seen when Ubuntu 16.04 LTS runs Ansible on the number of certain nodes. Higher CPU usage differences are caused by an application being deployed in the k8s cluster. Compared to the four nodes, pods can work more optimally, because there are more available resources, so the workload of nodes is lighter. When users want to run an application program, the data or files needed to run the program will be retrieved from secondary storage media (Hard Disk / SSD). Then, the system

transfers the data to RAM for further processing by the processor. After processing is completed, the processor will display the results to the output device or return them to the storage device. If the amount of data to be stored has exceeded the RAM capacity, the operating system will run the swap or temporary transfer procedure. The data will be moved temporarily to a secondary storage space called a swap file or virtual Memory, which the node memory usage is the total memory usage of all pods.

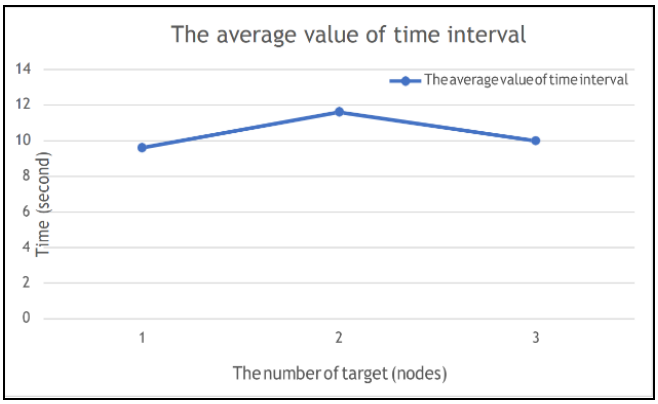

Fig. 5: Result of Deployment Time Interval

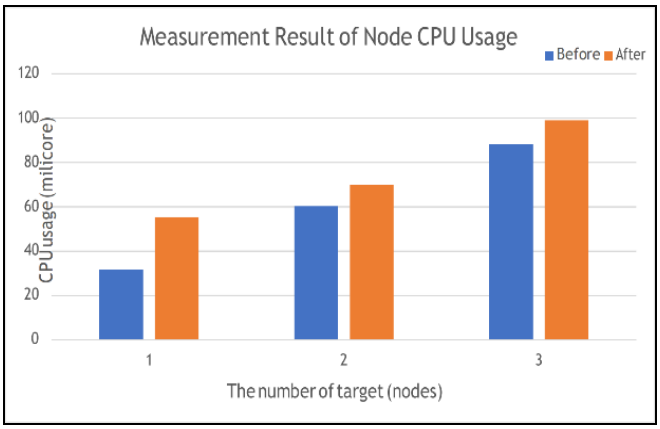

Fig. 6: Node of CPU

The test results of the memory usage on the cloud server resource can be seen in Fig. 7, which shows similar result with previous test in CPU usage. This result is affected because RAM is a temporary data storage area when programs on a computer run while it can be accessed randomly or not depending on its layout setting. As for the automation process analysis, the comparison of the implementation of manual and automation of Ansible systems have been provided in Fig. 8. Manual configuration is a configuration that is implemented manually on each Personal Computer, the commands used and configuration settings are done manually by the user. It can be seen that the flow of this manual configuration is carried out one by one for each PC. This process makes the manual system take a long time because it is conducted one by one per application. Ansible automation configuration is a configuration that uses an automatic system by simply running a playbook. The resource pool allows for material savings, which indirectly leads to a reduction in electrical energy consumption but leads to some internal risks that must be carefully considered to anticipate [21]. Therefore, it is useful to classify the information type to reduce the time spent searching and other automation activity [22]. Interestingly, it also promises simplified interactions between the devices used due to standard protocols and increased flexibility regarding process implementation and re-engineering [23]. Thus, at an extended level, certain types of intelligent process automation should cover five main technologies namely automated process automation, intelligent workflows, advanced machine learning or analytics, natural language creation (NLG) and cognitive factors to increase productivity, improve efficiency, reduce operational risk and strengthening customer experience [24].

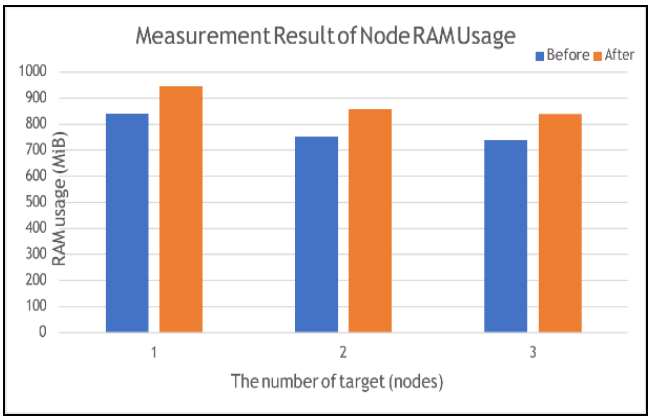

Fig.7: Node RAM Usage

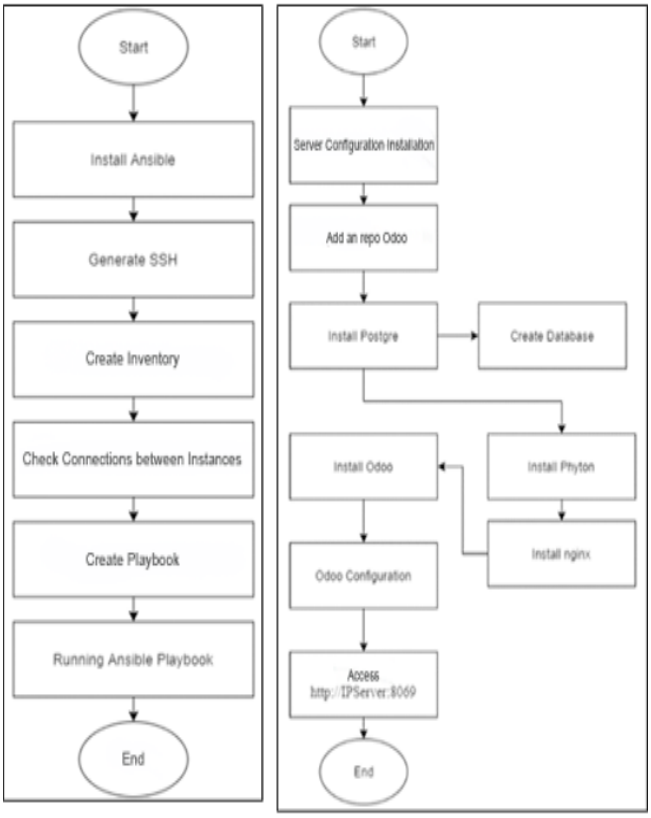

Fig. 8. Automatic and Manual Configuration

System testing carried out in manual and automation Ansible systems run by the predetermined design and test flow. Measurement of CPU, Memory, and Bandwidth usage is obtained from the output of the "netdata" command before, during, and after the backup and restore process occurred. The analysis is carried out on the test results on the two flows that have been carried out. The analysis results will be obtained to compare manual and automation Ansible systems based on processing time and data on CPU, Memory and Bandwidth. The following is an explanation of the analysis results of the testing process. In the manual system testing flow, users manually install and configure Odoo, then test it by the automation using Ansible. In the analysis, a comparison of the results in the two streams is carried out. The following is a comparison of the two paths by observing CPU usage and processing time.

The Fig. 9 shows the comparison graph of CPU usage obtained from the tests carried out on manual and automation Ansible systems. The test results found an increase in CPU usage in the initial changes when configuring. In the manual system, the user will carry out the installation and configuration so that Odoo can be used on the graph. The manual system has changed 24 times. It reaches the highest point at 9,6 %. In the Ansible automation process as can be seen the difference that the automation carries out the installation and configuration process with the changes as much as 20 times when running Ansible. The graph displayed is quite stable at 3.6%. The problem, however, it is often that most companies try to scale up, the results are likely to fail, which is a one-off initiative in a separate unit that does not have much impact across the company. Meanwhile, adopting remedial methods will always yield disappointing results and programs that provide temporary but unsustainable benefits [25].

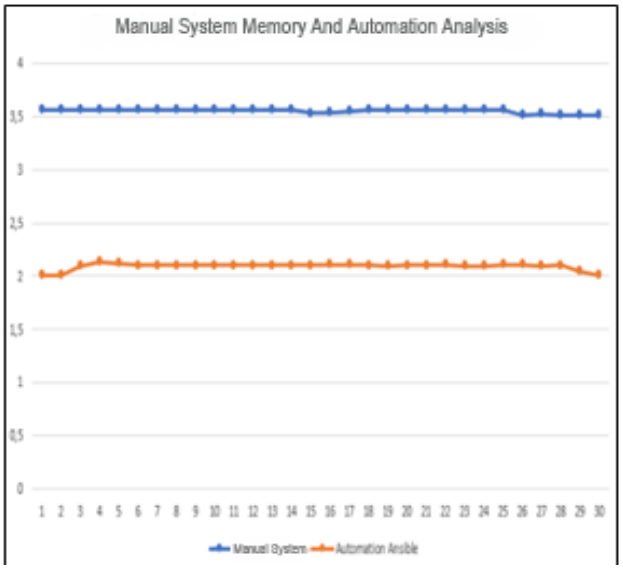

Fig. 9. Manual and Automation CPU Usage Testing

One way to provide IT infrastructure services for effective business operations within an organization is to plan accordingly before the implementation and deployment phases take place [26], which configuration and performance management become the critical planning to be executed [27]. The shortcomings of simplified initial modeling platforms are often twofold as they fail to accurately represent the complex interactions and communications between architectural design and building programs. It related to inform a fully integrated design approach used and the need to have design breaks at the project level as the clear details are beyond what is exactly in the program [28]. Thus, it is more efficient to have a concrete and comprehensive identification system in all silos than to have a vendor-specific identification method [29]. With real-time monitoring, testing, configuration, control, and evaluation based on network data, network administrators can obtain network system performance, evaluate quality of service (QoS) and searching where network trouble spots are located, which optimize the detection, integration and calculation, especially the level of thoroughness [30- 32].

The Fig. 10 shows a comparison graph of the memory usage obtained from tests carried out on manual and automation Ansible systems. From the test results, it shows that the manual system has changed 24 times and reached the highest point at 9.6%. The difference that automation can carry out the installation and configuration process with changes as much as 20 times when running Ansible. The graph displayed is quite stable at 3.6%. Therefore, the attempt to encompasses every configuration model across life cycle tend to have more seamless integration of the business unit and external stakeholders in terms of process continuity and data exchange by emphasizing the automation process as social process through hybrid approach for reliable control [33-37].

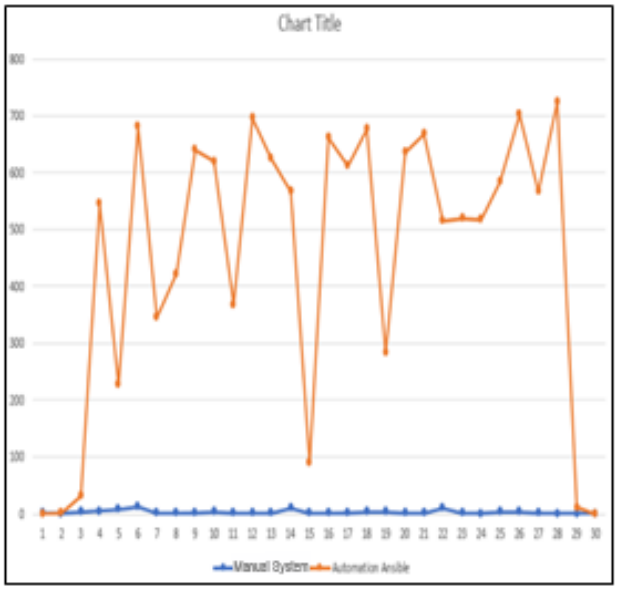

Fig.10. Manual and Automation Memory Usage Testing

Fig. 11 above shows the bandwidth usage comparison chart obtained from tests carried out on manual systems and Ansible automation. Importantly, poor decisions can have escalating negative consequences in the subsequent phases in performance and configuration management, lead to improper investment, which in this case requires good structure, communication and evaluation to be put together in providing the service [38]. Collaboration with other colleagues to transfer the knowledge acquired by group members to all other members will enhance and support the management process

[39-42]. The test results show an increase in bandwidth usage in the manual system, which is relatively low because the user only installs and configures the CLI. Therefore, the bandwidth used by the manual system is low. In the automation Ansible process, it can be seen that the difference that the bandwidth usage is high with a change of 29 times when running Ansible so that the graph displayed is quite high at 723.

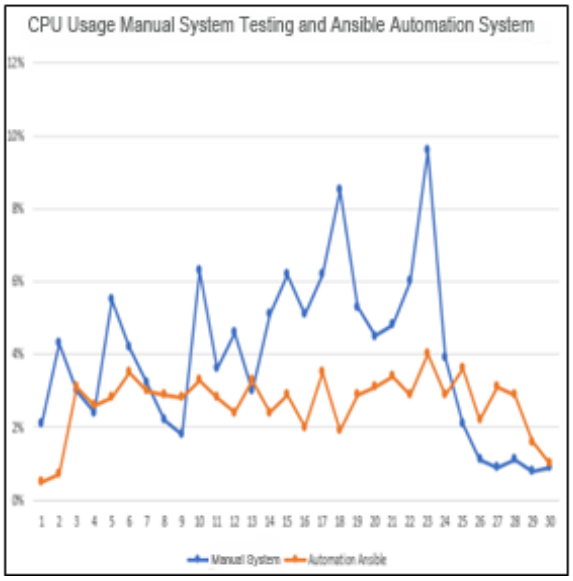

Fig.11. Manual and Automation Bandwidth Usage Testing

IV. CONCLUSION

Change configuration management could be managed by using CLI or AWX Ansible Tower from the configuration management perspective. Based on the analysis of CPU usage carried out in manual and automation Ansible system processes concluded that in manual systems, there is a lot of troubleshooting during installation and configuration. This process makes the time for determining how to work on the Odoo manual system cannot be detailed and precise. This system can cause the manual process possible and take longer than expected, depending on the existing specifications. There was an increase in CPU during Ansible automation on the Ansible server, but it was stable at 2.6%. Whereas the Memory gets an average value of 2.09kb, and the average bandwidth obtained is 4.82kb. In the comparison of Manual and Automation Ansible, it can be seen that there are significant differences, because the two processes are different, it can be concluded that automation Ansible is more effective in helping system administrators. It also can be said that using Ansible automation is more effective and easier than manual systems but requires a larger Internet source than using manual systems.

The results of testing and analysis of the implementation of Ansible to deploy on the local machine and cloud server, also concluded that based on a comparison of measurements in the deployment process time interval from local computer to cloud server, the difference between the numbers of nodes indicates that the results tend to increase in each. The more the number of nodes needed, the more time to do the deployment process. Meanwhile, based on the CPU usage measurements on the number of different nodes, the more the number of nodes, smaller the CPU usage. This condition might happen because more resources are available. Based on the results of memory usage measurements (RAM) on the number of different nodes, the results increase in each. In scaling, estimation of node CPU usage takes 28,3% for increasing 1 unit (node) after deployment, unless scaling estimation RAM usage takes 50,9% to increase 1 unit (node) after deployment. Scaling estimation RAM usage is greater than CPU usage because it is used for running the application deployed. Running and managing the Ansible configuration can be done with an open-source web application, namely AWX Ansible. This study, using it to deploy Kubernetes clusters from local to hosts instances on the cloud server.

REFERENCES

[1] S. K. Uzaman, A. U. R. Khan, J. Shuja, T. Maqsood, F. Rehman, and S. Mustafa, "A systems overview of commercial data centers: Initial energy and cost analysis," *Int. J. Inf. Technol. Web Eng.*, vol. 14, no. 1, pp. 42–65, 2019, DOI: 10.4018/IJITWE.2019010103.

[2] H. Hamidi and A. Chavoshi, "Analysis of the essential factors for the adoption of mobile learning in higher education: A case study of students of the University of Technology," *Telemat. Informatics*, vol. 35, no. 4, pp. 1053–1070, 2018, DOI: 10.1016/j.tele.2017.09.016.

[3] R. Shaw, Y. Kim, and J. Hua, "Governance, technology and citizen behavior in pandemic: Lessons from COVID-19 in East Asia," *Prog. Disaster Sci.*, vol. 6, p. 100090, 2020, DOI: 10.1016/j.pdisas.2020.100090.

[4] M. Attaran, "Cloud computing technology: leveraging the power of the internet to improve business performance," *J. Int. Technol. Inf. Manag.*, vol. 26, no. 1, pp. 112–137, 2017. DOI: 10.1080/08276331.2018.1466850.

[5] A. Sunyaev, "Cloud Computing," in *Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies*, Cham: Springer Int. Publishing, 2020, pp. 195–236. DOI: 10.1007/978-3-030-34957-8.

[6] J. Vora, S. Kaneriya, S. Tanwar, and S. Tyagi, "Performance Evaluation of SDN based Virtualization for Data Center Networks," in *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, Feb. 2018, pp. 1–5, DOI: 10.1109/IoT-SIU.2018.8519846.

[7] E. Mezghani, E. Exposito, and K. Drira, "A Model-Driven Methodology for the Design of Autonomic and Cognitive IoT-Based Systems: Application to Healthcare," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 1, no. 3, pp. 224–234, 2017, DOI: 10.1109/TETCI.2017.2699218.

[8] M. Giannakis, K. Spanaki, and R. Dubey, "A cloud-based supply chain management system: effects on supply chain responsiveness," *J. Enterp. Inf. Manag.*, vol. 32, no. 4, pp. 585–607, Jan. 2019, DOI: 10.1108/JEIM-05-2018-0106.

[9] L. Chen and J. Xu, "Socially Trusted Collaborative Edge Computing in Ultra Dense Networks," 2017, DOI: 10.1145/3132211.3134451.

[10] A. Yassine, S. Singh, M. S. Hossain, and G. Muhammad, "IoT big data analytics for smart homes with fog and cloud computing," *Futur. Gener. Comput. Syst.*, vol. 91, no. September, pp. 563–573, 2019, DOI: 10.1016/j.future.2018.08.040.

[11] K. Gai, M. Qiu, H. Zhao, and X. Sun, "Resource Management in Sustainable Cyber-Physical Systems Using Heterogeneous Cloud Computing," *IEEE Trans. Sustain. Comput.*, vol. 3, no. 2, pp. 60–72, Apr. 2018, DOI: 10.1109/TSUSC.2017.2723954.

[12] T. Stützle and M. López-Ibáñez, "Automated Design of Metaheuristic Algorithms," in *Handbook of Metaheuristics*, M. Gendreau and J.-Y. Potvin, Eds. Cham: Springer International Publishing, 2019, pp. 541–579. DOI:

10.1007/978-3-319-91086-4_17
[13] V. Anu, W. Hu, J. C. Carver, G. S. Walia, and G. Bradshaw, "Development of a human error taxonomy for software requirements: A systematic literature review," *Inf. Softw. Technol.*, vol. 103, pp. 112–124, 2018, DOI: 10.1016/j.infsof.2018.06.011.
[14] M. Kusuma, Widyawan, and R. Ferdiana, "Performance comparison of caching strategy on wordpress multisite," in *2017 3rd International Conference on Science and Technology - Computer (ICST)*, 2017, pp. 176–181, DOI: 10.1109/ICSTC.2017.8011874.
[15] M. Soltys and K. Soltys, "WordPress on AWS: a Communication Framework," *arXiv*, 2020.
[16] L. Hochstein and R. Moser, *Ansible: Up and Running: Automating configuration management and deployment the easy way*. " O'Reilly Media, Inc.," 2017. arXiv:2007.01823
[17] A. Widjajarto, D. W. Jacob, and M. Lubis, "Live migration using checkpoint and restore in userspace (CRIU): Usage analysis of network, memory and CPU," *Bull. Electr. Eng. Informatics*, vol. 10, no. 2, pp. 837–847, 2021, DOI: 10.11591/eei.v10i2.2742.
[18] M. Lubis, C. Wardana, and A. Widjajarto, "The Development of Information System Security Operation Centre (SOC): Case Study of Auto Repair Company," in *2020 6th International Conference on Interactive Digital Media (ICIDM)*, 2020, pp. 1–8, DOI: 10.1109/ICIDM51048.2020.9339678.
[19] H. V. Gamido and M. V. Gamido, "Comparative review of the features of automated software testing tools," *Int. J. Electr. Comput. Eng.*, vol. 9, no. 5, pp. 4473–4478, 2019, DOI: 10.11591/ijece.v9i5.pp4473-4478.
[20] M. Narayana, N. Raghu Ram Reddy, and N. Hyndavi Reddy, "High speed script execution for GUI Automation using Computer Vision," *Int. J. Electr. Comput. Eng.*, vol. 9, no. 1, pp. 231–236, 2019, DOI: 10.11591/ijece.v9i1.pp231-236.
[21] W. A. Jbara and K. A. Ali, "Cloud Computing of Network and Impact of Communication," vol. 55, no. 2, 2020. DOI: 10.35741/issn.0258-2724.55.2.8.
[22] M. Mahmood, W. J. Al-kubaisy, and B. Al-Khateeb, "Using Artificial Neural Network for Multimedia Information Retrieval," *J. Southwest jiaotong Univ.*, vol. 3, 2019. DOI: 10.35741/issn.0258-2724.54.3.19.
[23] M. Mathes, J. Gärtner, H. Dohmann, and B. Freisleben, "SOAP4IPC: A Real-Time SOAP Engine for Industrial Automation," in *2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, Feb. 2009, pp. 220–226, DOI: 10.1109/PDP.2009.21.
[24] F. Berruti, G. Nixon, G. Taglioni, and R. Whiteman, "Intelligent Process Automation: The Engine at the Core of the Next-Generation Operating Model," *McKinsey Artic.*, 2017.
[25] A. Bollard, E. Larrea, A. Singla, and R. Sood, "The Next-Generation Operating Model for Digital World," *McKinsey Artic.*, 2017.
[26] A. Widjajarto, M. Lubis, and U. Yunan, "Architecture Model of Information Technology Infrastructure based on Service Quality at Government Institution," *Procedia Comput. Sci.*, vol. 161, pp. 841–850, 2019, DOI: 10.1016/j.procs.2019.11.191.
[27] A. Q. Gill and E. Chew, "Configuration information system architecture: Insights from applied action design research," *Inf. Manag.*, vol. 56, no. 4, pp. 507–525, 2019, DOI: 10.1016/j.im.2018.09.011.
[28] E. Niemeyer and S. Rao, "A Modeling Framework for Engine-Neutral Automation of Building Analysis and Compliance Reporting," *Build. Perform. Anal. Conf. SimBuild*, 2020.
[29] N. Yousefnezhad, A. Malhi, and K. Främling, "Security in product lifecycle of IoT devices: A survey," *J. Netw. Comput. Appl.*, vol. 171, p. 102779, 2020, DOI: 10.1016/j.jnca.2020.102779.
[30] D. Zhou, Z. Yan, Y. Fu, and Z. Yao, "A survey on network data collection," *J. Netw. Comput. Appl.*, vol. 116, pp. 9–23, 2018, DOI: 10.1016/j.jnca.2018.05.004.
[31] R. Sigit, A. Wulandari, N. Rofiqah, and H. Yuniarti, "Automatic detection brain segmentation to detect brain tumor using MRI," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 9, no. 6, pp. 1913–1920, 2019, DOI: 10.18517/ijaseit.9.6.8536.
[32] R. Romli, S. Sarker, M. Omar, and M. Mahmod, "Automated test cases and test data generation for dynamic structural testing in automatic programming assessment using MC/DC," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 10, no. 1, pp. 120–127, 2020, DOI: 10.18517/ijaseit.10.1.10166.
[33] A. Myrodia, T. Randrup, and L. Hvam, "Configuration lifecycle management maturity model," *Comput. Ind.*, vol. 106, pp. 30–47, 2019, DOI: 10.1016/j.compind.2018.12.006.
[34] P. T. Makowski and Y. Kajikawa, "Automation-driven innovation management? Toward Innovation-Automation-Strategy cycle," *Technol. Forecast. Soc. Change*, vol. 168, pp. 1–24, 2021, DOI: 10.1016/j.techfore.2021.120723.
[35] S. Karthick *et al.*, "Realization of industrial automation using Bluetooth technologies," *Mater. Today Proc.*, 2021, DOI: 10.1016/j.matpr.2020.11.928.
[36] A. P. F. P. L. Barbosa, M. S. Salerno, P. T. de Souza Nascimento, A. Albala, F. P. Maranzato, and D. Tamoschus, "Configurations of project management practices to enhance the performance of open innovation R&D projects," *Int. J. Proj. Manag.*, vol. 39, no. 2, pp. 128–138, 2021, DOI: 10.1016/j.ijproman.2020.06.005.
[37] G. R. Molaeimanesh, S. M. Mirfallah Nasiry, and M. Dahmardeh, "Impact of configuration on the performance of a hybrid thermal management system including phase change material and water-cooling channels for Li-ion batteries," *Appl. Therm. Eng.*, vol. 181, p. 116028, 2020, DOI: 10.1016/j.applthermaleng.2020.116028.
[38] A. Haug, S. Shafiee, and L. Hvam, "The causes of product configuration project failure," *Comput. Ind.*, vol. 108, pp. 121–131, 2019, DOI: 10.1016/j.compind.2019.03.002.
[39] K. Deepika and N. Sathyanarayana, "Relief-F and Budget Tree Random Forest based Feature Selection for Student Academic Performance Prediction," *Int. J. of Intelligent Engineering & Systems*, vol. 12(1), pp. 30–39, 2019, DOI: 10.22266/ijies2019.0228.04.
[40] R. Jaganathan and V. Ramasamy, "Performance Modeling of Bio-Inspired Protocols in Cognitive Radio Ad Hoc Network to Reduce End-to-End Delay," *Int. J. of Intelligent Engineering & Systems*, vol. 12(1), pp. 30–39, 2019, DOI: 10.22266/ijies2019.0228.22.
[41] M. Lubis, M. Kartiwi and S. Zulhuda, "Current State of Personal Data Protection in Electronic Voting: Criteria and Indicator for Effective Implementation," *Telkomnika*, vol. 16(1), pp. 290–301, 2018. DOI: 10.12928/TELKOMNIKA.v16i1.7718.
[42] M. Lubis, A.R Lubis, B. Lubis and A. Lubis, "Incremental Innovation towards Business Performance: Data Management Challenges in Healthcare Industry in Indonesia," *MATEC Web of Conferences*, 218, 04015, 2018. DOI: 10.1051/matecconf/201821804015.