Accredited Ranking SINTA 2
Decree of the Director General of Higher Education, Research and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online on: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 7 No. 5 (2023) 1174 - 1181 ISSN Media Electronic: 2580-0760

Service Automation Implementation for Delivering CaaS at the Ministry of Finance of Indonesia

Achmad Farid Rusdi[1], Teguh Raharjo[2], Bob Hardian[3], Tiarma Simanungkalit[4]
[1,2,3,4]Faculty of Computer Science, Universitas Indonesia, Jakarta, Indonesia
[1]achmad.farid11@ui.ac.id, [2]teguhr2000@gmail.com, [3]hardian@cs.ui.ac.id, [4]tiarma.simanungkalit@ui.ac.id

Abstract

Evaluation is an essential aspect of service improvement. Within the Ministry of Finance, there is an organization responsible for providing IT services to various units and employees. One of the services offered by this organization is cloud computing, which supports the development of information systems. However, there are several challenges related to service fulfillment. For instance, the time required to fulfill a service is relatively long, taking two days, in contrast to public cloud providers who can deliver their services within minutes. Additionally, there is a potential for human errors in the manual process carried out by the Request Fulfillment Team (RFT) during service delivery. This study aims to explore the design and implementation of automated service fulfillment for containers, transforming them into self-service products. The author employs the Finite State Automata (FSA) model to test the input and output of the automation system using seven states and inputs. The results indicate that the service cycle for containers when compiled and tested with FSA using predetermined inputs, can generate containers according to user-selected specifications. As a result, the implementation of an automated and self-service model is proposed to reduce delivery time and mitigate potential errors in the Container-as-a-Service (CaaS) offering.

Keywords: cloud service automation; cloud service implementation; container as a service; CaaS; finite state automata

1. Introduction

Cloud computing is a computerized process for the needs of applications, and infrastructure, including storage, as a service to customers with Internet access [1]. Cloud services are generally provided in three models: (1) Infrastructure as a Service (IaaS), namely infrastructure through web portals and APIs; (2) Platform as a Service (PaaS), which is a platform for application development services, such as database services and scripting environments. (3) Software as a Service (SaaS), namely the provision of applications as a service, (example: Google Service) [2]. In its development, there is Container, a virtualization technology that does not require a VM (virtual machine). Container virtual environments are isolated at the operating system level, so a particular operating system will be executed by a different container [3]. Containers use fewer resources for random-access memory (RAM) and the central processing unit (CPU) than conventional VMs and are located between IaaS and PaaS [4]. A container can be used as a cloud service, and it's called "container as a service (CaaS). CaaS is a technology where resources are stored in containers and can be accessed by applications according to the needs of the application [5].

Technological developments require organizations to innovate and utilize information technology to continuously improve their services. ITSM (Information Technology Service Management) is a framework used by companies to manage IT services and achieve business goals. The ITIL (Information Technology Infrastructure Library) provides a general structure for ITSM and guides the IT governance structure and continuous improvement of IT service performance provided by the organization [6]. Within ITIL, there is a lifecycle to implement ITSM efficiently, which does not define a definite start and endpoint. The lifecycle consists of service strategy, service design, service transition, service operation, and continual service improvement [7].

There is an organization within the Ministry of Finance that offers cloud computing services, in terms of PaaS server-tier development, including container services. These services are managed using the ITSM framework. This organization manages its cloud data center, it can be categorized as a private cloud [8]. In fulfilling PaaS server-tier development service, there are several obstacles, service fulfillment which takes two working days [9], relatively more time compared to cloud computing services in the current public cloud,

Accepted: 06-04-2023 | Received in revised: 16-09-2023 | Published: 22-10-2023

and the potential for errors in service fulfillment by the RFT (Request Fulfillment Team) caused by human error. After conducting service evaluation activities and gathering feedback through various forums, meetings, and user satisfaction surveys, gaps were identified in the cloud services that require improvement. Specifically, the areas that need attention are service fulfillment time and personnel availability. These improvements will be part of the continuous service improvement process.

Several studies have shown an automated iterative process to shorten the processing time, for example, automated teller machines (ATMs), where services previously performed by humans are replaced by automated machines [10]. In the cloud area, there is a cloud computing architecture of the Government of India that is moving towards cloud service automation scenarios, which includes the use of containers using the VMware Technology Stack [1]. The development of Cloud CAMP as a tool that automates and orchestrates the cloud that complies with OASIS standards with GUI-based. This tool aims to reduce the manual effort involved in developing and orchestration cloud services [11]. As technology continues to evolve, there is potential for automating increasingly complex processes [12]. In developing automation, predefined inputs, and intended outputs are required. There is a mathematical model that can be used to automate the processing of inputs and outputs, known as finite state automata (FSA). An FSA is used to represent a system that has a finite number of states and can transition between them based on the input it receives [14]. FSA can be used to test automation processes in which there are specified inputs and outputs. FSA is expressed in 5 tuples [16]: Q (state set), $\sum$ (input set), δ (transition function), S (initial state), and F (final state).

In realizing the design of PaaS server-tier development services for containers that are more efficient and effective, organizations need to consider automating these services. This can be done through container orchestration, which enables service providers to define how to coordinate and continuously manage containers in the cloud [13], and by using low-level scripting in the container service implementation process as per established procedures [11].

The conditions described previously indicate that service automation for the PaaS server-tier development for containers can reduce service fulfillment times and minimize coordination and potential errors. Thus, this research was conducted to answer the questions “What kind of service design can be implemented for self-service containers?” and “How can container services be automated?”.

Several studies have been conducted on CaaS. Research [4] describes the pricing scheme for CaaS services, research [14] explains data center resource savings that can be achieved by implementing CaaS, and research [15] discusses container implementation. In addition, there are studies related to FSA, research [8], [9] using FSA as a conceptual model of e-services, research [17]–[21] discussing the use of FSA in service automation tests through vending machines, research [22]–[24] discussing the use of FSA in system menu selection, and research [25] using FSA for virtual server service automation test. In our study, we will use non-deterministic finite automata (NFA) as there is more than one transition direction. In comparison with previous research, this research contributes to the form of service cycle design and its application to the provision of CaaS through automated processes in the public sector.

2. Research Methods

This part will outline the research methodology to utilize container service automation design and implementation. We are starting with the research phase, data collection, and design test method.

2.1 Research Phase

This research consists of four stages, the first is understanding the current business process flow. The second is the service cycle design that the author will try to construct a service cycle. This cycle consists of activities that will be automated using tools. The third is the service cycle validation. In this case, the researcher uses the JFLAP application in making FSA diagrams and FSA testing by describing the transition table. The last stage is the process of designing a solution design and implementing a container service as can be seen in Figure 1.

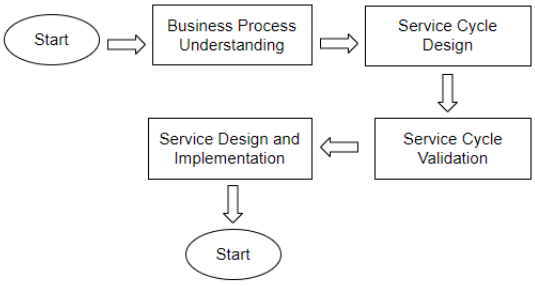

Figure 1. Research Process

Figure 1 shows that this research was carried out in 5 stages. First is business process understanding, at this stage the current business process flow in the organization will be explained. The second is service cycle design. At this stage, the service cycle will be designed when the service is automated using tools and becomes a self-service service. Third is state diagram design, at this stage, the states contained in the service cycle will be designed later. Fourth is the state test. At this stage, the researcher uses the input transition function and the output transition function to test the

DOI: https://doi.org/10.29207/resti.v7i5.5032

stages of each state of the FSA so that it can be ensured that there are no errors in the state design of the FSA. The final step is service design and implementation. Solution design for CaaS and interface of the CaaS service form is carried out on the development tier on the Service Portal where users can make service requests according to what the user needs.

2.2 Data Collection

Data collection in this study used a qualitative approach with an exploratory literature review and semi-structured interview [26]. A literature study was used to obtain secondary data and interviews were to collect primary data. Secondary data was collected from the Service Catalog, IT Regulations, and presentation documents related to the CaaS at the Ministry of Finance. For primary data, we conducted semi-structured interviews.

For the interview, the informants were selected by the purposive sampling method because they can provide in-depth and detailed information. The source people in this study are the Server Manager (13 years of working experience), the Quality Assurance Manager (13 years of working experience), and two technical staff (12- and 9 years of working experience). In addition, information was also gathered from representatives from some public providers that already offer products of CaaS as self-service in the private sector.

2.3 Design test method

All repetitive processes that are typically performed by individuals will be carried out by machines or tools. Therefore, to ensure the smooth operation of the process with these tools, this research will utilize the FSA model to test the input and output of the service cycle design for container service automation.

Once the automation process is set up, the next step will be testing with FSA. The objective is to validate the flow and make any necessary adjustments based on the results obtained. A flow diagram will be created for testing purposes using FSA.

3. Results and Discussions

3.1 Result

The study discusses the transformation of services that were previously manually executed by multiple individuals into automated processes using tools. Providing PaaS development server services previously took two days because it not only involved many people fulfilling the request manually but also encompassed many steps in the process itself. Therefore, this study will concentrate on automating repetitive activities to enhance service quality.

Some processes will be conducted to automate repetitive activities using tools. The discussion begins with comprehending the pre-existing business processes, followed by designing the service cycle to fulfill the service, validating the service cycle, and concluding with the design and implementation of the service solution.

During business process understanding, requests for container services currently running in the organization are still running manually. Users make requests via email or the service desk service portal. After the request is received, the Service Desk team will check the request. If the information and files for the request are considered complete by the Service Desk team, then the ticket goes to the Server Management team which is the RFT (Request Fulfillment Team) for the service and will then be fulfilled by them according to specifications that have been determined by the user and approved by Change Manager if the request specification exceeds the standard specifications specified in the Service Catalog, as shown in the Figure 2.

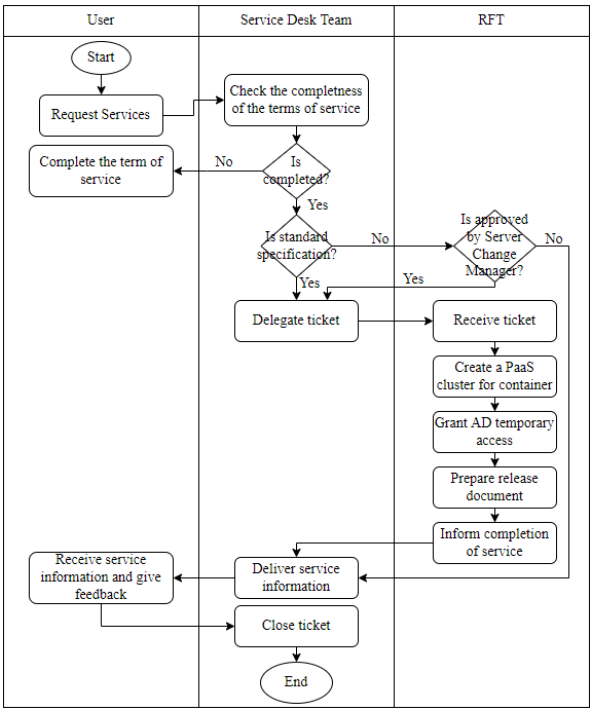

Figure 2. Existing Service Fulfillment Flowchart for Container Services

Based on the picture above, we can understand that the process of fulfilling these services is routine and repetitive. Meeting the demand for services requires the availability of human resources and time from the service desk team and the RFT team. In addition, this RFT consists of two different teams (server team and network team) so it means that to deliver the service, the organization needs resources from three teams.

DOI: https://doi.org/10.29207/resti.v7i5.5032

As stated in the Service Catalog, the norm for this service time is two working days. Therefore, it is necessary to design a system that can help simplify the tasks of the Server Manager team to shorten the time for service fulfillment and as the Server Manager said, "have a service model that can be fully self-service like services in public clouds". The system in question can enable users to make service requests on the Service Portal according to the required specifications independently (self-service). In the Service Portal, the service cycle flow can run immediately, and the service fulfillment process is automated using tools so that requests are possible to be fulfilled in minutes if the request is immediately approved by the user's Direct Manager.

There are nine stages of the PaaS development server service process for the current container. The process starts with a user who makes a service request via the Service Desk Service Portal. After that, the Service Desk team will receive and check the completeness of the service requirements. If the requirements are not complete, then the request is returned to the user to be completed and resubmitted the request. If the requirements are complete, it will be checked whether the standard specifications are as stated in the Service Catalog or not. If it meets the standard specifications, then the process will continue to the ticket delegation process to the RFT which is the Server Team. If the specifications are non-standard, then approval by the Server Change Manager is required. If the Server Change Manager approves, the process will continue to the ticket delegation process to the RFT service. If rejected, the Service Desk team will convey the status of service information to the user. Thirdly, RFT will create a PaaS cluster for the container which includes DNS registration and port opening. For the DNS registration process, RFT will coordinate with the Network Team the DNS, and AD (Active Directory) management team. For the opening of the port, RFT will coordinate with the network team. Next, RFT will grant AD account temporary access to users. Then, RFT will prepare the release document. After this document is completed, RFT will inform the Service Desk about the completion of services. Having this information, the next step is to convey the service status information to Users by the Service Desk team. Then, users can receive information and provide feedback on services. Finally, the process ends with the Service Desk Team closing the service ticket.

During service cycle design, based on the existence of the service fulfillment flowchart for container services, the process needed to fulfill these services requires a long step, the availability of time and human resources as well as good synergy between teams. Based on an interview with Server Manager and Quality Assurance Manager, the required service design is fully self-service, and faster than current services, such as public cloud services.

After the data collection process is carried out, the author constructs a service cycle in which the process of activities that are still carried out manually by RFT and related teams will be automated using tools. However, for good governance and validating that the request is indeed needed by the organization, each request still requires approval by the Direct Manager of the User.

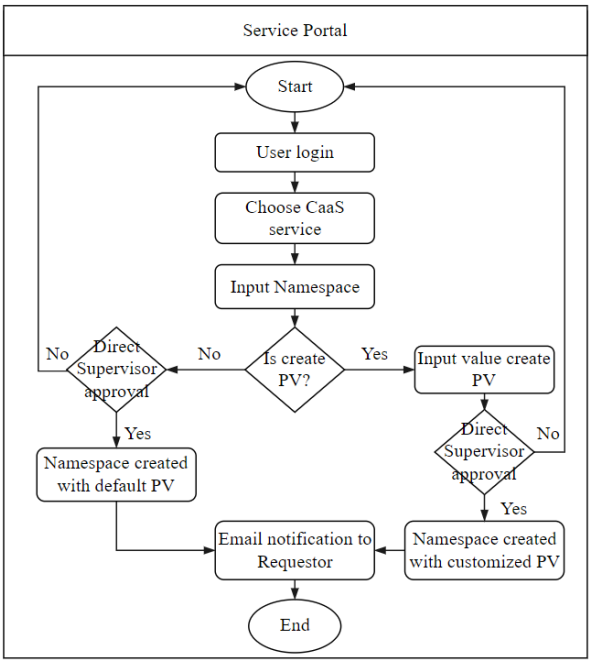

Figure 3. The Service Fulfillment Flowchart

Figure 3 illustrates the service fulfillment process that has been automated and the request can be made self-service by the user. With this service cycle, it is hoped that the fulfillment of PaaS server-tier development for container services will meet the requirement of the Server Manager, which is to provide faster and more dynamic development infrastructure for users.

The process in the implemented service cycle starts with the user logging into the Service Portal. Then, the user can select CaaS service. After that, the user is demanded to input a form request such as Namespace Name. In this form, there is an option to customize the size of PV by checkbox Created PV. When this is selected, the user will input the size for customized PV and then PVC will be updated to claim PV according to the input size. The next process will be to the Direct Supervisor for approval. If the User didn't select created PV, the request will continue to the Direct Supervisor for approval. After getting approval, Namespace will be created with default or customized size PV based on User selection. Finally, after successful creation, email notifications will be sent to the user with details of

DOI: https://doi.org/10.29207/resti.v7i5.5032

Namespace Name and Cluster URL to be accessible with their AD.

In the process of the Service Cycle Validation, there are two states which are State Diagram Design and State Testing. The State Diagram Design is used to describe in detail the flow of the system that will be designed so that it can describe a complete and comprehensive system.

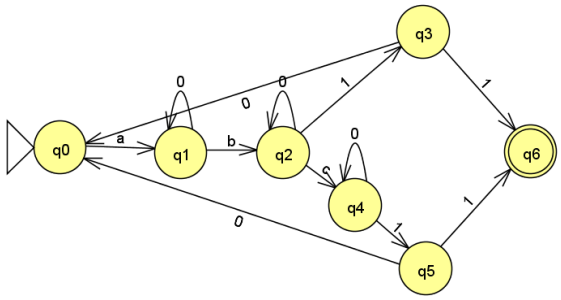

Figure 4. Diagram State Design

The FSA diagram illustrates a machine model that can obtain input and provide output with a finite number of states and can move from one state to the next according to the input obtained from the transition functions [25]. An item in Figure 4 can be explained in Table 1.

Table 1. FSA Item Description

Tuple	State and Input	Description
Q	q0	Initial State - service portal login
	q1	Show container service options
	q2	Show a field to input Namespace Name
	q3	Show a direct supervisor approval for Namespace with a default size of PV
	q4	Show a display to input the customized size of the PV
	q5	Show a direct supervisor approval for Namespace with customize the size of PV
	q6	Final State – Namespace created, and Email Notification sent to Requestor
∑	a	Chose container service
	b	Input Namespace Name
	c	Input customizes the size of the PV
	0	follow-up
	1	execute
S	q0	Initial state
F	q6	Final state

In the State Diagram Design in Figure 4, state q0 describes that the user is logged in to the service portal. State q1 displays the choice of container services. State q2 displays a field to input the Namespace Name that will be built into the project and bind to the AD account. State q3 is a user selection that Namespace is processed with a default size of PV, this state will go for Direct Supervisor approval. State q4 shows a field to input the customized size of PV for the Namespace that will be built. After input size for customized PV, the process will then be executed in state q5 for Direct Supervisor approval. After getting approval from the Direct Supervisor in states q3 and q5. Last transition in q6, a Namespace will be created based on the user selection size of PV, and an email notification will be sent to the requestor with a detailed Namespace and Cluster URL for access with the AD account. The transition function between states can be seen in the following table 2.

Table 22. Transition Function

δ	q0	q1	q2	q3	q4	q5	q6
a	q1	-	-	-	-	-	-
b	-	q2	-	-	-	-	-
c	-	-	q4	-	-	-	-
d	-	-	-	-	-	-	-
e	-	-	-	-	-	-	-
0	-	q1	q2	q0	q4	q0	-
1	-	-	q3	q6	q5	q6	-

In the State Testing, FSA testing on the PaaS development server service for this container uses the JFLAP application as a tool or tool in making FSA and testing [5]. The system will follow the flow from starting the login process into the Service Portal. FSA is used to read the given input symbol from the initial state to the end of the process so that it can be recognized by the Service Portal. The following is an overview of testing whether the input entered is accepted or rejected based on input from the user:

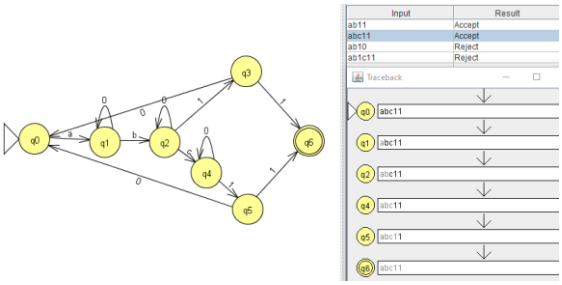

Figure 5. FSA Testing State

Figure 5 shows the design of the FSA input system with the JFLAP application trial. All the test input results in the transition function table show consistent results according to the transition function table above. An example of FSA testing with the input "abc11" is accepted. The input “ab11” was accepted but not complete to the final state, another input “ab10” was rejected because there is no further transition from q2 to q3 or q4.

In service solution design and implementation phase, the solution design used for Caas automation as shown in Figure 6. The service will be developed using a service portal, orchestrator, container solution, virtualization management solution, and AD Server that will be integrated to make the process of delivering container service. So, when the users choose the CaaS directly on the catalog, the tools that have already been configured before will process the request automatically.

DOI: https://doi.org/10.29207/resti.v7i5.5032

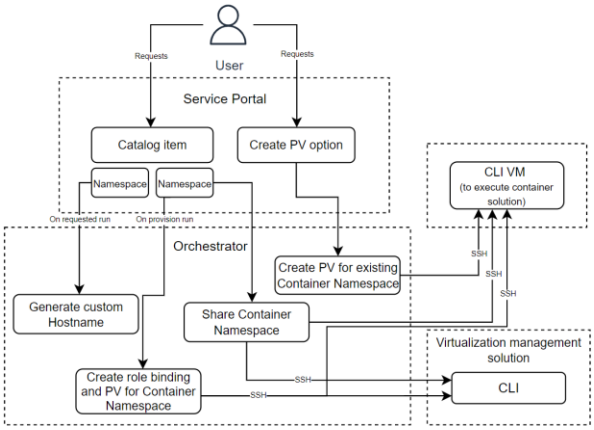

Figure 6. Solution Design CaaS

A service portal is needed that can allow users to select container services, input Namespace Name, an option to create PV, and the approval request from their Direct Supervisor. The form and display for the service can be shown in Figure 7 and 8.

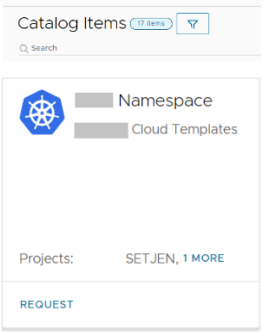

Figure 7. Container Service Options Display

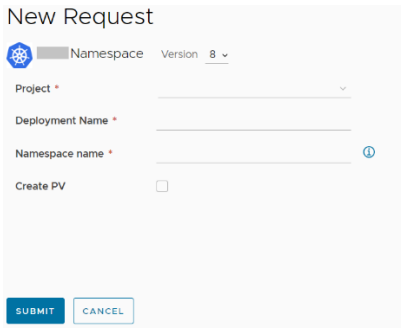

Figure 8. Display Container Service Request Form

Figures 7 and 8 illustrate the input form that the user must fill out when requesting a service. The user will input the container name in Deployment and Namespace, and an option to customize the size of PV by filling the checkbox Create PV. After the user completes the input on this form, it will trigger the workflow to execute pre- and post-provision actions. Cluster Role Binding is created for the requested user for the given container cluster to allow access to storage classes and PV. Role binding was created for the requested AD user for the new namespace in the CLI VM server for the administrative role. The size of PV can be increased when the request is submitted.

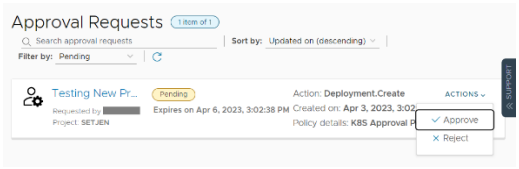

Figure 9. Approval Form Display

Figure 9 is a display of the approval form when the direct supervisor wants to approve or reject an incoming request from a requestor.

3.2 Discussions

According to the results obtained in this research, the automation process uses the service design that has been explained in service design and implementation. The service design describes the service process flow starting from service requests by users to automatic fulfillment of services through the system. Based on interviews that have been conducted, the backend of the system that runs the service flow is supported by two products, namely the cloud management platform and the container platform which are also integrated with AD and DNS servers. With this flow, it is hoped that the fulfillment of services can be carried out by users independently as Server Manager hoped, so that users can immediately deploy microservice applications without waiting for admin availability. PV selection is intended to fulfill storage needs if the developer requires PV outside the predetermined storage classes and PVs that are already included when requesting a namespace.

The supervisor's approval is needed as a form of check and control so that the requested content is to the needs of the organization. Container control by the admin is carried out by using monitoring tools and scanning for containers that will enter the production area. If high and very high-security gaps are found, then the deployment process to the production area will not be able to be carried out by the user. Security gap findings need to be closed by the user.

Other things that can be added to this service, first is an option to add PV size according to system developer requirements. Along with the use of containers, there is a potential need for more storage, so options are needed to increase the size of the PV. However, according to the initial PV submission form, the addition of the PV size can only be done on the Namespace that is bound to the AD's user account and the initial Namespace submission does not choose to change the PV size. The second is in system development, the common thing is collaboration between developers. So, another thing that can be added is a collaboration between developers. Initially, each Namespace will only be bound with one

DOI: https://doi.org/10.29207/resti.v7i5.5032

AD account, therefore an additional form is needed to add other developers to that Namespace within the same project.

In providing services that are automated with the system, according to the Server Manager, some things need attention, such as standardization. It is required in several aspects, such as standardization of containers used and security. Capacity planning is also important because it is difficult to predict, especially during the mid-year period when many system changes and developments occur. Then, for the smooth use of the service, users need to be provided with training and guidelines for using the service.

Change management is also a challenge for organizations, such as how to rearrange existing human resources after the automation of these services. Some recommendations to minimize the impact of automation [7], [27]. The success of automation in organizations requires a change in organizational culture to reduce resistance to automation projects, training of automation tools and techniques to its users in the organization identifying new roles that arise with the automation, and mapping these new roles to existing human resources.

4. Conclusion

Routine and repetitive processes in fulfilling the services of containers are implemented to be automated using tools. Service lifecycle proposals of automated container services can be drawn up and tested using an FSA. The implementation of FSA is described by providing input that is given to the system as a language that can be recognized by the system. Then the system will issue an output in the form of a container with a default or customized size of PV.

The design for container service in section 3 can answer research question 1. Based on the design, the Service Desk Team and Request Fulfillment Team no longer have a role in fulfilling the container services. Their role is replaced by a system flow that is designed using a cloud management platform and container platform tools that can do automatic tasks. Users can get the container by self-service in Portal. The system generates Namespace and Cluster URLs which can be accessed by AD's User account.

To answer research question 2, the first thing that we did was identify routine and repetitive activities in the provision of container services and then configure the activities to be automated by the system. As a form of control, there is a direct supervisor as the party that validates the container request, and the server admin who monitors the containers that have been created.

This automation will have a good impact on the organization, routine and repetitive processes are automated. Existing human resources can be directed to other processes that require more skills and knowledge to complete the process. Implementing the automated and self-service model aims to reduce the time and potential for errors in the fulfillment of container services.

This research's limitations are that it is restricted to the provision of container service at the Ministry of Finance in Indonesia. As a result, the findings do not fully describe the issues the public sector faces in Indonesia related to the provision of container services. Furthermore, there are few references to container services design in the public sector. Further research can be conducted by using more research objects, especially in the public sector, to provide a more comprehensive view.

Acknowledgment

This research was funded by the Ministry of Communication and Information of the Republic of Indonesia based on the announcement letter number B-2048/BLSDM.1/LT.02.03/10/2022.

References

[1] N. V. Choudhari and A. B. Sasankar, "Architectural vision of cloud computing in the Indian government," *2021 Int. Conf. Innov. Trends Inf. Technol. ICITIIT 2021*, 2021, doi: 10.1109/ICITIIT51526.2021.9399598.

[2] W. Faiq, "IMPLEMENTASI CLOUD COMPUTING DI BEBERAPA INSTANSI PEMERINTAHAN Cloud Computing Implementation in Several Government Institutions Faiq Wildana," *J. Masy. Telemat. dan Inf.*, pp. 97–108, 2017.

[3] M. K. Hussein, M. H. Mousa, and M. A. Alqarni, "A placement architecture for a container as a service (CaaS) in a cloud environment," *J. Cloud Comput.*, vol. 8, no. 1, pp. 1–15, 2019, doi: 10.1186/s13677-019-0131-1.

[4] V. Liagkou, G. Fragiadakis, E. Filiopoulou, C. Michalakelis, T. Kamalakis, and M. Nikolaidou, "A pricing model for Container-as-a-Service, based on hedonic indices," *Simul. Model. Pract. Theory*, vol. 115, no. November 2021, p. 102441, 2022, doi: 10.1016/j.simpat.2021.102441.

[5] K. Kaur, T. Dhand, N. Kumar, and S. Zeadally, "Container-as-a-Service at the Edge: Trade-off between Energy Efficiency and Service Availability at Fog Nano Data Centers," *IEEE Wirel. Commun.*, vol. 24, no. 3, pp. 48–56, 2017, doi: 10.1109/MWC.2017.1600427.

[6] R. Yandri, Suharjito, D. N. Utama, and A. Zahra, "Evaluation model for the implementation of information technology service management using fuzzy ITIL," *Procedia Comput. Sci.*, vol. 157, pp. 290–297, 2019, doi: 10.1016/j.procs.2019.08.169.

[7] G. Krishnan and V. Ravindran, "IT service management automation and its impact on IT industry," *ICCIDS 2017 - Int. Conf. Comput. Intell. Data Sci. Proc.*, vol. 2018-Janua, pp. 5–8, 2018, doi: 10.1109/ICCIDS.2017.8272633.

[8] F. Mohammed, A. M. Ali, A. S. A. M. Al-Ghamdi, F. Alsolami, S. M. Shamsuddin, and F. E. Eassa, "Cloud computing services: Taxonomy of discovery approaches and extraction solutions," *Symmetry (Basel).*, vol. 12, no. 8, pp. 1–15, 2020, doi: 10.3390/sym12081354.

[9] Pusintek, "IT Service Catalog Pusat Sistem Informasi dan Teknologi Keuangan." Kementerian Keuangan, Jakarta, 2021.

[10] S. H. Ivanov, C. Webster, and K. Berezina, "Adoption of robots

and service automation by tourism and hospitality companies," *Rev. Tur. …*, vol. 1, no. 27/28, pp. 1501–1517, 2017, [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2964308.

[11] A. Bhattacharjee, Y. Barve, A. Gokhale, and T. Kuroda, "(WIP) CloudCAMP: Automating the deployment and management of cloud services," *Proc. - 2018 IEEE Int. Conf. Serv. Comput. SCC 2018 - Part 2018 IEEE World Congr. Serv.*, pp. 237–240, 2018, doi: 10.1109/SCC.2018.00038.

[12] S. E. Sampson, "A Strategic Framework for Task Automation in Professional Services," *J. Serv. Res.*, vol. 24, no. 1, pp. 122–140, 2021, doi: 10.1177/1094670520940407.

[13] C. Pahl, A. Brogi, J. Soldani, and P. Jamshidi, "Cloud container technologies: A state-of-the-art review," *IEEE Trans. Cloud Comput.*, vol. 7, no. 3, pp. 677–692, 2019, doi: 10.1109/TCC.2017.2702586.

[14] N. Kumar, G. S. Aujla, S. Garg, K. Kaur, R. Ranjan, and S. K. Garg, "Renewable Energy-Based Multi-Indexed Job Classification and Container Management Scheme for Sustainability of Cloud Data Centers," *IEEE Trans. Ind. Informatics*, vol. 15, no. 5, pp. 2947–2957, 2019, doi: 10.1109/TII.2018.2800693.

[15] E. Kristiani, C. T. Yang, C. Y. Huang, Y. T. Wang, and P. C. Ko, "The Implementation of a Cloud-Edge Computing Architecture Using OpenStack and Kubernetes for Air Quality Monitoring Application," *Mob. Networks Appl.*, vol. 26, no. 3, pp. 1070–1092, 2021, doi: 10.1007/s11036-020-01620-5.

[16] Y. Jiang, "A formal model of semantic computing," *Soft Comput.*, vol. 23, no. 14, pp. 5411–5429, 2019, doi: 10.1007/s00500-018-3502-5.

[17] D. Berardi, F. De Rosa, L. De Santis, and M. Mecella, "Finite state automata as conceptual model for e-Services," *J. Integr. Des. Process Sci.*, vol. 8, no. 2, pp. 105–121, 2004.

[18] R. Muhammad, W. Gata, H. B. Novitasari, L. Kurniawati, and S. Rahayu, "Penerapan Finite State Automata Pada Desain Vending Machine Masker Dan Hand Sanitizer," *J. Inf. dan Komput.*, vol. 10, no. 1, pp. 21–28, 2022, doi: 10.35959/jik.v10i1.275.

[19] E. Supriyanto, A. Ardiansyah, F. Frieyadie, S. Rahayu, and W. Gata, "Penerapan Finite State Automata Pada Vending Machine Penjual Obat Non Resep Dokter Dan Keperluan Medis," *J. Inf. dan Komput.*, vol. 9, no. 2, pp. 08–14, 2021, doi: 10.35959/jik.v9i2.206.

[20] D. Erwanto, "Penerapan Konsep Finite State Automata Pada Desain Vending Machine Angkringan," *J. Inform.*, vol. 21, no. 2, pp. 161–173, 2022, doi: 10.30873/ji.v21i2.3063.

[21] A. Faisal, G. V. Saragih, and W. Gata, "Desain Vending Machine Rokok Dengan Mengimplementasikan Finite State Automata Terintegrasi Dengan E-KTP," *Matics*, vol. 12, no. 1, p. 55, 2020, doi: 10.18860/mat.v12i1.8693.

[22] F. Said, D. Andriyanto, R. Sari, and W. Gata, "Perancangan Validasi Permohonan Narasumber Pada Sistem Informasi Permohonan Narasumber Menggunakan Finite State Automata," *Paradig. - J. Komput. dan Inform.*, vol. 22, no. 2, pp. 189–196, 2020, doi: 10.31294/p.v22i2.8157.

[23] T. Rivanie, "Implementasi Finite State Automata dalam Proses Registrasi Workout Plan pada Pusat Kebugaran," *Matics*, vol. 12, no. 1, p. 94, 2020, doi: 10.18860/mat.v12i1.8573.

[24] F. Aziz, "Penerapan Konsep Finite State Automata Dalam Proses Pendaftaran Kelas Kursus Bahasa Inggris Pada Tempat Kursus," *Matics*, vol. 12, no. 2, pp. 93–98, 2021, doi: 10.18860/mat.v12i2.9330.

[25] M. B. Shidiq, W. Gata, H. B. Novitasari, A. Bayhaqy, and H. Setiawan, "Penerapan Layanan Cloud Server Secara Self-Service Menggunakan Model Finite State Automata," *INTECOMS J. Inf. Technol. Comput. Sci.*, vol. 5, no. 1, pp. 73–82, 2022, doi: 10.31539/intecoms.v5i1.3216.

[26] J. Recker, *Scientific Research in Information Systems*. Springer, 2013.

[27] R. Dunlap and M. Lacity, "Resolving tussles in service automation deployments: Service automation at Blue Cross Blue Shield North Carolina (BCBSNC)," *J. Inf. Technol. Teach. Cases*, vol. 7, no. 1, pp. 29–34, 2017, doi: 10.1057/s41266-016-0008-9.

DOI: https://doi.org/10.29207/resti.v7i5.5032