e-ISSN. 2476-9843
p-ISSN. 1858-4144

Jurnal MATRIK
Vol.19 No.2 (Mei) 2020, Hal 347 -357
DOI https://doi.org/10.30812/matrik.v19i2.724

Implementation Of Configuration Management Virtual Private Server Using Ansible

I Putu Hariyadi[1], Khairan Marzuki[2]
[1,2]Computer Science,Universitas Bumigora
E-Mail: [1]putu.hariyadi@stmikbumigora.ac.id, [2]khairan.marzuki@universitasbumigora.ac.id

ABSTRACT

Virtualization technology has been applied to universities with computer study programs to support practicum in Network Management courses. Each user gets a Virtual Private Server (VPS) with container technology. VPS system that is prepared manually requires a long time, especially when the number of users is increasing. The activity repeats every semester so it becomes ineffective and inefficient. Application of automation using Ansible can help to manage VPS objects in the Promox Virtual Environment (PVE) Cluster dynamically. Network Development Life Cycle (NDLC) was used as a method in this study. The design of the VPS management automation system created supports the grouping of container, user and permission resource management for users. The design is implemented in Ansible Playbook. The test results show the average time of making VPS objects per student with an automation system 2 (two) times faster, that is 26.25 seconds compared to the old system which takes 2 minutes 15 seconds. Besides that, the Playbook that was created succeeded in automating the start and stop containers per group of students based on the practicum schedule dynamically so as to maintain the availability of services from the PVE Cluster.

Keywords: Automation, Virtual Private Server, Proxmox, Ansible, Playbook

*Author Korespondensi (**I Putu Hariyadi**)*
Email : putu.hariyadi@stmikbumigora.ac.id

I. INTRODUCTION

Virtualization technology has been implemented by universities with computer study program to support practical learning. Universitas Bumigora is one of universities which has been implemented a virtualization-based practical learning media innovation built using PVE [1]. Each users acquires a Virtual Private Server (VPS) with a container technology to support practicum activities of network Management (NM). VPS is a virtual server that can run various operating and accepted by the user as a dedicated or private server making it possible to control hardware and soft on instances of the operating system [2]. The PVE-based virtual laboratory has also been used by STMIK to implement the network and Server management practicum [3].

Simply, there are 3 (three) steps that the administrator must take before the user can use the VPS to create containers, account for authentication and set permissions. If these three stages are done manually via the Web GUI of PVE then it takes an average time of 2 minutes 15 seconds per users. Manually setting up VPS and systems becomes ineffective and inefficient when the number of VPS users get more and the activity is repeated in every semester.

Ansible is an information technology automation tool used for configuration management Application and orchestration as well as agentless [4]. The automation feature developed are creation and deletion of VPS related objects dynamically based on user data that classified into 3 (three) students, lecturers and assistants. In addition, there is a grouping container feature based on user practicum group and scheduling to run (start) and stop the container per student practicum group.

The benefits of VPS management automation with Ansible administrators can manage good container services, account authentication and permissions more effectively and efficiently. Container grouping in accordance with the practicum group. This is as the impact of the automation related to the container that is started and is stopped

e-ISSN. 2476-9843
p-ISSN. 1858-4144

Jurnal MATRIK
Vol.19 No.2 (Mei) 2020, Hal 347 -357
DOI https://doi.org/10.30812/matrik.v19i2.724

dynamically based on the practicum schedule of each user group per week.

II. METHODOLOGY

This study uses Network Development Life Cycle (NDLC) method. NDLC consists of 6 (six) stages including analysis, design, simulation prototyping, implementation, monitoring and management, as shown in Figure 1 [5].

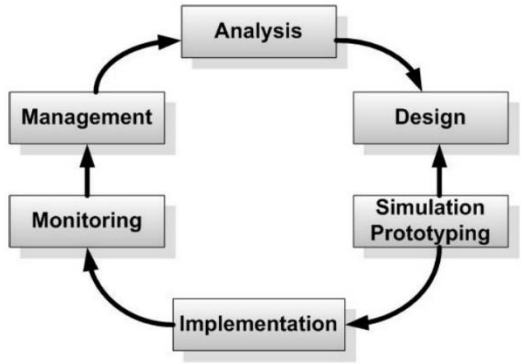

Figure 1. Network Development Life Cycle [5]

Researcher only uses three stages in NLDC, namely analysis, design and simulation prototyping.

2.1. Step of Analysis

At this step, identifying of user passion is based on the results of needs and problems analysis obtained from data collection [5]. Observation, interview and documentation are 3 (three) data collection techniques used. Based on the analysis of data that has been collected the result obtained that the administrator prepared the VPS of each user manually using the Web GUI from PVE. Problems emerge when administrators had to prepare VPS for many users at one time. These activities did repeatedly in each semester so that it becomes ineffective and inefficient. This prompted researchers to implement automation of VPS configuration management using Ansible.

2.2. Step of Design

This stage consists of 4 (four) parts, namely the design of the test network, the design of the VPS management automation system and the design of IP addressing as well as the hardware and software requirements. Figure 2 shows the trial network design used in this study. In trial network design, it shows there are 2 (two) subnets, namely the subnet server and the subnet of the laboratory (lab) computer network. The two subnets are connected using a gateway router so that between subnets are able to communicate and connect to the Internet. On the subnet server there are 3 (three) servers, namely Ansible Control Machine, PVE1 and PVE2 that are connected to the sw_server switch. PVE1 and PVE2 servers are installed with the PVE hypervisor and in clusters with the name "bumigora". Whereas in Communication networking laboratory there are 26 (twenty six) Personal Computers (PCs) that function as clients connected to the switch sw communication networking laboratory. Students and lecturers use the computer client to access the VPS generated by the Ansible-based automation system so that the practicum activities of the Management networking course can run.

The design of the VPS management automation system that was built, as shown in Figure 3. The administrator created an Playbook. Playbook is a file that is written using the YAML format and contains a sequence of tasks related to the module to be executed [6]. AAE consists of inventory, modules and plugins [7]. The inventory accommodate the Internet Protocol (IP) addresses of 2 (two) PVE servers, namely 172.16.0.11 for pve1 and 172.16.0.12 for pve2 incorporated in the Bumigora Cluster PVE.

This automation system uses the Comma Separated Values (CSV) file format in storing data related to the division of practicum groups for students, lecturers and assistants. There are 93 students data that are automated by making their VPS and divided into 4 (four) practicum groups namely A, B, C and D Preview of Student data with information on Student Identification Number (NIM), Student Name, Group stored in the group-lab-student.csv file, as shown in Figure 4.

e-ISSN. 2476-9843
p-ISSN. 1858-4144

Jurnal MATRIK
Vol.19 No.2 (Mei) 2020, Hal 347 -357
DOI https://doi.org/10.30812/matrik.v19i2.724

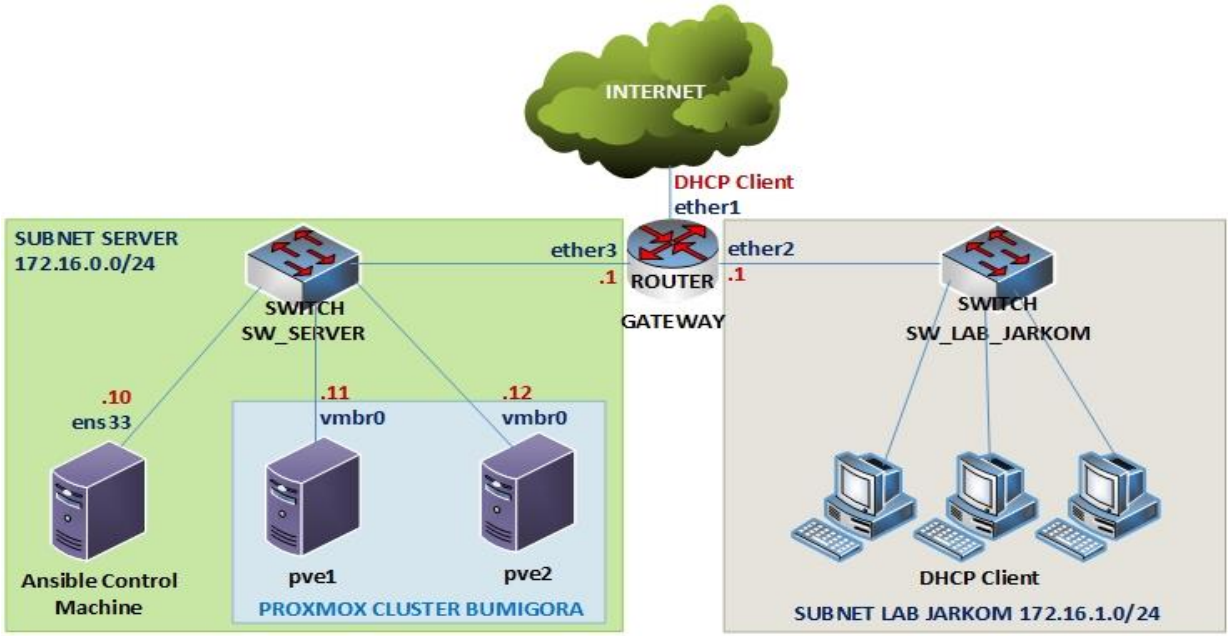

Figure 2. Network Design

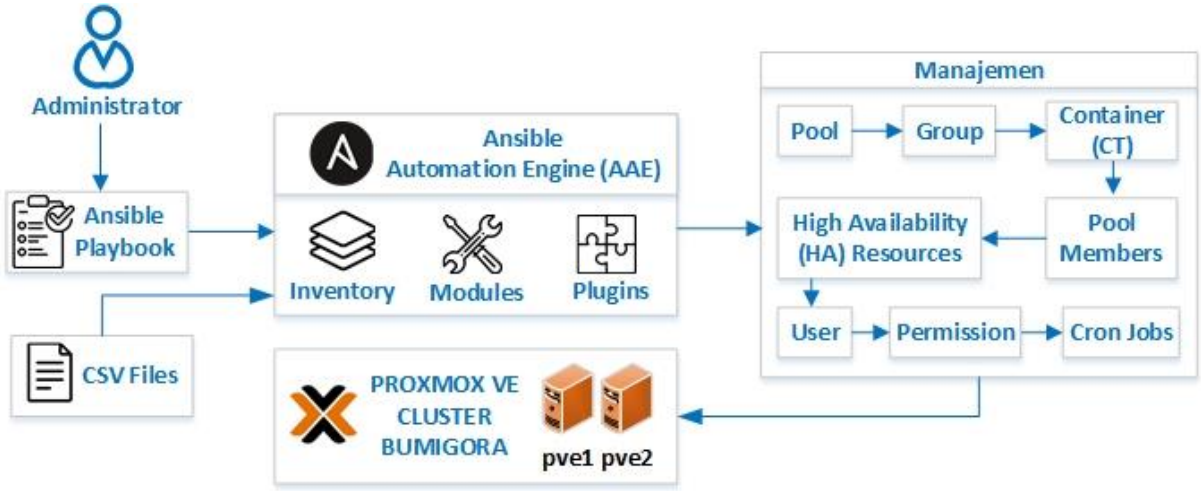

Figure 3. Automatic VPS Management System Design

```
1  nim,nama_mahasiswa,kelompok
2  1710510157,Rudi Kurniawan,A
3  1710520179,Siti Nislamuna,A
4  1710520180,Talitha Ellyyani,A
5  1810550276,Gladys Alisa Suciani,B
6  1710520089,Nanang Samudra,B
7  1710510064,Alga Doni Saputra,B
8  1710510145,Hendra Sukianto,C
9  1710510136,Abdul Gaffar MB,C
10 1710530124,Nuridin Agus Ismail,C
11 1510520066,Ardi Hidayat,D
12 1710530132,Ahmad Irfan Sazali,D
13 1710510047,Lalu Dwi Iwan Saputra,D
```

Figure 4. Preview Content of CSV File Data on Student Practicum Group

The first line in the file is a title of column of student data separated by commas. The lecturer data is stored in the lecturer-group-practicum.csv file which contains information related to the National Lecturer Identification Number (NIDN), the names and practicum groups that are taught, as shown in Figure 5.

```
1  nidn,nama_dosen,kelompok
2  0827068001,I Putu Hariyadi,A-B-C-D
```

Figure 5. Contents of CSV File for Practicum Group Lecturers

The first line in the file is also a title of column . While the score in the group column are

e-ISSN. 2476-9843
p-ISSN. 1858-4144

Jurnal MATRIK
Vol.19 No.2 (Mei) 2020, Hal 347 -357
DOI https://doi.org/10.30812/matrik.v19i2.724

separated using hyphen marks (-) per practicum group. Whereas practicum assistant data is stored in the assistant-group-lab.csv file containing information related to Employee Identification Number (NIK), names and practicum groups of each assistant, as shown in Figure 6.

```
1  nik,nama_asisten,kelompok
2  19.4.01,Naufal Hanif,A-B-C-D
3  1610520067,Rukmin Ulfa,A-B-C-D
```

Figure 6. Contents of the CSV File Practicum Assistant Group

The first line in the file is also a title of column. Score in the group column are also separated using hyphen marks (-) for each practicum group.

The design of resource management in the PVE Cluster of Bumigora consists of eight stages. The first and second stages are dynamically creating a pool and group in accordance with the name of the student practicum group contained in the group-practicum-student file. Pool is used to group containers allocated to each student according to the practicum group. Likewise for each lecturer using the DOSEN-NIDN naming convention based on the data in the lecturer-group-practicum.csv file and assistant with the ASISTEN-NIDN naming convention based on the data in the assistant-group-practicum.csv file. While groups are used to classify student data by name in accordance with student practicum groups. The third step is to make containers on each server. The fourth step is to add Container Identification (CT ID) as a member of the practicum pool. The fifth step is to add CT ID to High Availability (HA) Resources to support live migration so that containers can be moved between servers when running [8]. The sixth step is to create a user to authenticate login to PVE with the realm type of PVE Authentication. The seventh step is to create permissions so that each user can manage containers independently. Besides that, lecturers and assistants also do permission settings so they can manage containers and help reset student passwords. The eighth stage is to make scheduling related to start and stop containers of each practicum group automatically based on the practicum schedule data per group stored in the schedule-practicum.csv file. The file contains information related to the practicum group name, day and time practicum takes place, as shown in Figure 7.

```
1  kelompok,hari,jam
2  A,selasa,8:50-10:30
3  B,selasa,10:30-12:10
4  C,senin,16:50-18:30
5  D,kamis,11:20-13:00
```

Figure 7. Contents of CSV File Practicum Schedule

The first line is also the title of title column. Time of Practicum consists of opening time and ending time separated by hyphen (-). The eight stages are defined in the tasks of Ansible Playbook to be applied in the PVE Cluster.

The IP address design uses a class B network address, 172.16.0.0/16, which is subnetting to produce the 172.16.0.0/24 subnet address allocated for the subnet server and 172.16.1.0/24 allocated for the subnet of Communication networking laboratory. The details of IP address allocation per network device involved in the trial network design, as shown in table 1.

Table 1. IP Address Design Per Network Device

No.	Device Name	Interface	IP Address	Gateway
1.	*Router Gateway*	*Ether 1*	*DHCP Client*	
		Ether 2	172.16.0.1/24	-
		Ether 3	172.16.1.0/24	-
2.	*Server Ansible Control Machine*	*ens33*	172.16.0.10/24	172.16.0.1
3.	*Server PVE1*	*vmbr0*	172.16.0.11/24	
4.	*Server*	*vmbr*	172.16.0.	

e-ISSN. 2476-9843
p-ISSN. 1858-4144

Jurnal MATRIK
Vol.19 No.2 (Mei) 2020, Hal 347 -357
DOI https://doi.org/10.30812/matrik.v19i2.724

	PVE2	*0*	12/24	
5.	*Client Lab Jarkom*	*Ether net0*	*DHCP Client*	

There are five hardware requirements in this study including (a) 3 PCs functioned as Ansible Control Machine and PVE servers, (b) 26 PCs as clients in the communication networking laboratory, (c) 1 unit of 24 port switches for network attachments from servers in the subnet server, (d) 1 unit of 24 port switch and 1 unit of 8 port switch for network attachment from PC in sub-network of communication networking laboratory, (e) 1 unit of router as a gateway to connect Internet connection. While the main software needed in this study include (a) PVE version 6.1-3 as a hypervisor of the PVE1 and PVE2 servers, (b) CentOS 7 (1804) as the operating system of the Ansible Control Machine server, (c) Images Container Template CentOS 7 for making containers in PVE, (d) Ansible as a configuration management tool, (e) Putty as a tool to remote access Secure Shell (SSH), (f) Chrome browser for accessing the Web Graphical User Interface (GUI) of PVE.

2.3 Simulation of Prototyping Step

The prototyping simulation stage is divided into 4 (four) parts, i.e. installation and configuration, making the Ansible Playbook and testing. The automated trial scenario consists of 6 (six) parts including (a) Creating a pool, group, container, pool members, HA resources, users and permissions for students, lecturers & assistants, (b) Running / stopping containers per practicum group, (c) Changing user passwords, (d) Disabling / activating user accounts, (e) Scheduling management for starting & stopping containers per student group, (f) Removing all pools, groups, containers, pool members, HA Resources, users and permissions from students, lecturers & assistants.

III. RESULTS AND DISCUSSION

3.1. Results of Server Installation and Configuration

A preview of the result of the installation of Ansible version 2.9.1 on the server that functions as Ansible Control Machine, as shown in Figure 8.

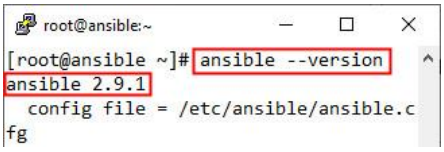

Figure 8. Results of Ansible Installation

While the footage of the results of the making of PVE Cluster with the name of the natives, as shown in figure 9.

Figure 9. Joining Clusters of natives

The cluster has two nodes as members, namely pve1 & pve2 servers.

3.2. Results of the VPS Management Automation Playbook

Snippet results from the program code in the main.yml file which is the main playbook of the VPS management automation system, as shown in Figure 10.

```
---
- name: Otomasi Manajemen Virtual Private Server (VPS)
  hosts: pve1
  gather_facts: no
  vars_prompt:
  - name: menu
    prompt: " Otomasi Manajemen Virtual Private Server
             Menggunakan Ansible Pada Media Pembelajar
             Praktikum Manajemen Jaringan Berbasis\n
             Proxmox Virtual Environment (PVE) Cluster
             \n
             1. Menampilkan Data Dari File CSV.\n
             2. Membuat Pool, Group, Container, Pool M
             3. Menjalankan/menghentikan Container Per
             4. Mengubah Sandi User.\n
             5. Menonaktifkan/mengaktifkan Akun User.\
             6. Menampilkan status container dari maha
             7. Menghapus Pool, Group, Container, Pool
             8. Manajemen penjadwalan start & stop con
             Keluar (Tekan Enter atau CTRL+C).\n
             \n
             Pilih menu (1-8)"
    private: no
  tasks:
  - name: Menampilkan Data Dari File CSV
    block:
    - name: Menampilkan prompt untuk input pilihan mah
      pause:
        prompt: "Data yang ingin ditampilkan? Ketik 1
      register: applicable
```

Figure 10. Main Playbook Program Code main.yml

e-ISSN. 2476-9843
p-ISSN. 1858-4144

Jurnal MATRIK
Vol.19 No.2 (Mei) 2020, Hal 347 -357
DOI https://doi.org/10.30812/matrik.v19i2.724

The first line, sign --- is the beginning of the YAML document. The second line is the name used to determine the name of the ansible playbook in the form of "Virtual Private Server Management Automation (VPS)". The third line, namely hosts, is used to determine the execution location of the task in the form of pve1. The fourth line, gather_facts, is used to disable the collection of information related to the remote pve1 system. The fifth to twenty-third lines are used to display prompts so that the administrator can enter the automation operation options that you want to perform. Lines twenty-fourth to thirty represent a list of actions or actions to be executed in accordance with the automation operation selected by the administrator. The twenty-fifth row is used to determine the name of the block. While the twenty-sixth line, namely block, is used to group logical lists of actions for 8 (eight) types of automation operation options.

The results of the execution use ansible-playbook on the main.yaml file, as shown in Figure 11.

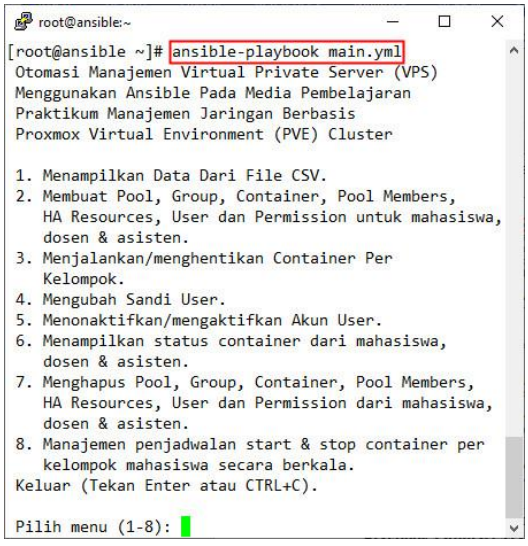

Figure 11. Main Menu of Ansible Playbook VPS Management Automation

There are 8 (eight) VPS automation management operation selection menus available. In addition there is an option to exit by pressing Enter or pressing CTRL + C.

3.3. Results of the Test

There are 6 (six) scenarios carried out to test the VPS automation system that which have been built. The first scenario is to provide a pool, group, container, pool members, HA resources, users and permissions for students, lecturers & assistants by typing 2 in the options menu of the VPS automation playbook. To measure the creation and deletion time of VPS-related objects, the playbook file is executed with an additional time utility so that it becomes "time ansible-playbook main.yml". Figure 12 shows the final results of the first attempt to execute the order to create 93 student VPS objects.

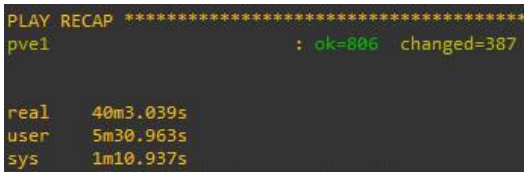

Figure 12. Time of Making VPS for Students

As shown in figure 12, the time needed to complete the process is 40 minutes 3039 seconds. The experiment was carried out 5 (five) times both to make and delete.

A sample of the results of making the pool object for each practicum group of students, lecturers and assistants, as shown in Figure 13.

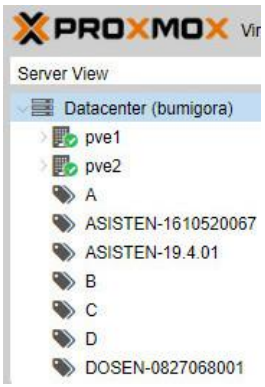

Figure 13. Pool Making Results

Figure 14 shows the program code snippet in the playbook file used to create the pool.

```
- name: Membuat pool
  command: pvesh create /pools -poolid
  "{{ item }}"
  with_items: "{{ sf_pools_diff }}"
  when: sf_pools_diff | length > 0
```

Figure 14. Program Code Creating a Pool

e-ISSN. 2476-9843
p-ISSN. 1858-4144

Jurnal MATRIK
Vol.19 No.2 (Mei) 2020, Hal 347 -357
DOI https://doi.org/10.30812/matrik.v19i2.724

The first line of program code is the name of the task that is "Creating a pool". Whilst the second to fourth rows are used to do a loop related to creating a pool if there is a value in the sf_pools_diff variable. The command module is used to execute the pvesh utility with an argument in the form of an API Path which contains information about the name of the pool to be created. pvesh is a shell interface for interacting with the PVE API [9].

The preview of the pool members' arrangement result for container grouping for lecturers with NIDN 0827068001 which has 4 lxc type containers, as shown in Figure 15.

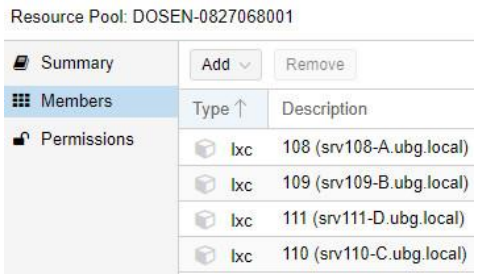

Figure 15. Results of Pool Members Lecturer Arrangements

It appears that the lecturer has access permission to use containers with IDs 108, 109, 110 and 111. While Figure 16 shows the program code snippet in the playbook file used to make the container.

```
- name: Proses pembuatan container
  command: pvesh create /nodes/{{
  ct_item.node }}/lxc -vmid "{{
  next_vmid }}" -hostname "srv{{
  next_vmid }}.{{ ct_searchdomain }}"
  -password "{{ ct_password }}"
  -ostemplate 'local:vztmpl/{{
  ct_ostemplate }}' -storage local-lvm
  -rootfs 5 -memory 512 -swap 512
  -net0 name=eth0,bridge=vmbr0
  -searchdomain ubg.local
  when:
  - ct_item.node == inventory_hostname
```

Figure 16. Program Code Making a Container

The second to fourth line is used to make containers on certain nodes. The command module is used to execute the pvesh utility with arguments in the form of an API Path that contains information related to the vmid parameter, hostname, password from the root, the container template used, storage, memory & swap size, network interface, and the search domain of the container to be created.

A screenshot of permission settings for the use of containers with certain access permissions for students, as shown in Figure 17.

Path ↓	User/Group	Role
/vms/107	1710530132@pve	PVEVMUser
/vms/106	1510520066@pve	PVEVMUser

Figure 17. Results of User Permission Settings

As shown on the figure, the two students had a PVEVM role User on CT ID 106 & 107. Furthermore, the footage of the results of adding two containers with ID 100 and 101 into HA.

Resources thus support migration between servers, as shown in Figure 18.

Figure 18. Results of Adding Container to HA Resources

The second scenario is to run or stop the containers per practicum group by typing 3 in the options menu of VPS automation. The code snippet in the playbook file for the operation, as shown in Figure 19.

```
- name: Menjalankan Container
  command: pvesh create /nodes/"{{ node
  }}"/lxc/"{{ item.vmid }}"/status/start
  with_items: "{{ poolresult.members }}"
  when:
  - item.type == 'lxc'
  - status == 'l'
  - item.node == inventory_hostname
```

Figure 19. Program Code Running a Container

The second to seventh line is used to do a loop associated with running a container that is a member of a particular pool. The command module is used to execute the pvesh utility with an argument in the form of an API Path which contains information about the PVE node as the location of the container and the vmid of the container to be run.

e-ISSN. 2476-9843
p-ISSN. 1858-4144

Jurnal MATRIK
Vol.19 No.2 (Mei) 2020, Hal 347 -357
DOI https://doi.org/10.30812/matrik.v19i2.724

The third scenario is to change the user's password by typing 4 in the options menu of the VPS automation playbook. The program code snippet in the playbook file for the operation, as shown in Figure 20.

```
- name: Mengubah sandi login mahasiswa
  command: pvesh set /access/password
  --userid "{{ userid }}" --password
  "{{ new_user_password }}"
```

Figure 20. Program Code Changing User Password

The second line in the program code is used to execute the module command with the argument from the pvesh utility in the form of an API Path which contains information about the userid and the new password of the user.

The fourth scenario is to deactivate or activate a user account with type 5 in the options menu of the VPS automation playbook. Tzhe program code snippet in the playbook file for the operation, as shown in Figure 21.

```
- name: Menonaktifkan/Mengaktifkan User
  command: pvesh set /access/users/"{{
  item.userid }}" -enable "{{ status }}"
  with_items: "{{ users.stdout }}"
  when:
  - item.userid != "root@pam"
  - item.comment == "{{ groupid }}"
```

Figure 21. Program Code Enabling / Disabling Users

The second to sixth lines in the program code are used to execute the module command with an argument from the pvesh utility in the form of an API Path which contains information about the userid of the user and a status value of 0 to deactivate or 1 to activate.

The fifth scenario is the management of scheduling start & stop containers per student group by typing 8 in the options menu from the VPS automation playbook. Figure 22 shows the program code snippet in the playbook file for the operation. The second to nine lines in the program code are used to ensure jobs run using the root user on hours, minutes, days that are executed using the cron module on localhost (Ansible Control Machine).

A screenshot of the scheduling results in the crontab file can be verified using the "crontab -l" command, as shown in Figure 23.

```
- name: Start container secara terjadwal
  cron:
     name: "Start Container Kelompok {{
     item.kelompok }}"
     user: "root"
     hour: "{{ item.start_hour }}"
     minute: "{{ item.start_minute }}"
     weekday: "{{ item.weekday }}"
     job: '/usr/bin/ansible-playbook
     /root/cron/start-stop-container.yml
     -e "poolid={{ item.kelompok }}"
     --skip-tags "stop"'
  delegate_to: localhost
```

Figure 22. Program Code Scheduling Start Container Per Group

```
root@ansible:~
[root@ansible ~]# crontab -l
#Ansible: Start Container Kelompok A
50 8 * * 2 /usr/bin/ansible-playbook /root/cron/start-
stop-container.yml -e "poolid=A" --skip-tags "stop"
#Ansible: Stop Container Kelompok A
30 10 * * 2 /usr/bin/ansible-playbook /root/cron/start
-stop-container.yml -e "poolid=A" --skip-tags "start"
```

Figure 23. Jobs Snippet on Crontab

Seen scheduling for group A practicum where containers of all students in the group will start automatically every Tuesday at 8:50 and stop every Tuesday at 10:30.

The sixth scenario is to delete all pools, groups, containers, pool members, HA Resources, users and permissions from students, lecturers & assistants by typing 7 in the options menu of the VPS automation playbook. Snippet of the results of verification of removal of all VPS-related objects for all users through the PVE GUI Web, as shown in Figure 24.

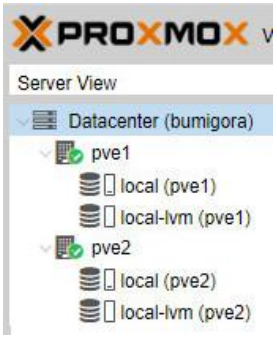

Figure 24. Deletion of All VPS Related Objects

Seen all objects in both the PVE1 and PVE2 servers have been successfully deleted. While Figure 25 shows the program code snippet contained in the playbook file to delete the container on a particular PVE node.

e-ISSN. 2476-9843
p-ISSN. 1858-4144

Jurnal MATRIK
Vol.19 No.2 (Mei) 2020, Hal 347 -357
DOI https://doi.org/10.30812/matrik.v19i2.724

```
1  - name: Menghapus container pada node PVE
2    command: pvesh delete /nodes/"{{
     node_item.node }}"/lxc/"{{ item.vmid
     }}"
3    with_items: "{{ containers.stdout }}"
4    when:
5    - node_item.node == inventory_hostname
```

Figure 25. Removing the Container Program Code

The second to fifth lines of the program code are used to execute module commands with arguments from the pvesh utility which is in the form of an API Path to delete containers.

3.4. Analysis of Test Results

Based on the trials that have been conducted, it can be obtained that the results of the analysis which include:

1. The command module on Ansible is used to execute the pvesh utility with an argument in the form of API Path from the PVE object related to VPS to be managed.
2. The cron module of Ansible can be used to automate scheduling related to start and stop containers periodically based on the practicum schedule per group.
3. The average time needed to automate the creation of VPS objects for 93 students is 40.7 minutes based on 5 (five) attempts, as shown in table 2. Based on the average time, it can be calculated the time to make one student VPS object is 40.7 minutes = 2442 seconds / 93 = 26.25 seconds.

Table 2. Experiment Time of Making Student VPS Objects

Experiment	Time of Making 93 Student VPS Objects
1	40 minutes 3.039 second
2	41 minutes 46.985 second
3	40 minutes 36.792 second
4	40 minutes 21.799 second
5	40 minutes 41.810 second
Average Time	40.7 minutes

4. The average time needed to automate the creation of VPS objects for one lecturer who is in charge of 4 practicum groups is 1.9 minutes based on 5 (five) experiments, as shown in table 3.

Table 3. Experiment Time of Making lecturer VPS Objects

Experiment	Time of Making 93 Lecturer VPS Objects
1	1 minutes 52.960 second
2	1 minutes 57.467 second
3	1 minutes 59.139 second
4	1 minutes 52.040 second
5	2 minutes 0.957 second
Average Time	1.9 menit

5. The average time needed to automate the creation of VPS objects for 2 assistants who provide assistance to 4 practicum groups is 3 (three) minutes based on 5 (five) attempts, as shown in table 4.

Table 4. Experiment Time of Making Asisstant VPS Objects

Experiment	Time of Making 93 Asisstant VPS Objects
1	3 minutes 41.845 second
2	3 minutes 51.993 second
3	3 minutes 47.690 second
4	3 minutes 47.437 second
5	3 minutes 56.271 second
Average Time	3 minutes

6. The average time needed to automate the elimination of all VPS objects for students, lecturers and students is 6.2 minutes based on 5 (five) attempts, as shown in table 5.

Jurnal MATRIK
Vol.19 No.2 (Mei) 2020, Hal 347 -357
DOI https://doi.org/10.30812/matrik.v19i2.724

Table 5. Timing of Deletion of All VPS User Objects

Experiment	Time of Deletion of all user VPS Objects
1	7 minutes 28.047 second
2	8 minutes 14.508 second
3	8 minutes 3.344 second
4	7 minutes 28.619 second
5	7 minutes 31.412 second
Average Time	6.2 minutes

IV. CONCLUSIONS AND SUGGESTIONS

4.1. CONCLUSION

Based on the results of the installation, configuration and testing as well as an analysis of the results of the trials that have been carried out, the conclusions can be drawn as follows:

1. An impossible playbook that was created successfully accommodates the overall design of the VPS management automation system.
2. The overall VPS management automation feature on Ansible Playbook works well and is successfully applied to the PVE Cluster.
3. The Playbook created successfully automates the start and stop containers per group of students based on the practicum schedule dynamically so as to maintain the availability of services from the PVE Cluster.
4. The average time to create a VPS object per student with an automation system is 26.25 seconds so that it is 2 (two) times faster than a manual system that takes 2 minutes 15 minutes.

4.2. SUGGESTIONS

The suggestions for further development of this research are as follows:

1. Developing the network automation feature of PVE that supports the application of VLANs in containers and integrates with automation of VLAN configuration management on switches and routing on routers.
2. Developing automated assessment of answers to each student's MJ practical exam questions so as to help students correct answers and speed up examination of exam results.

V. ACKNOWLEDGEMENT

Thank you to Universitas Bumigora for funding and providing supporting facilities and infrastructure for this research.

REFERENCES

[1] I. P. Hariyadi and A. Juliansyah, "Analisa Penerapan Private Cloud Computing Berbasis Proxmox Virtual Environment Sebagai Media Pembelajaran Praktikum Manajemen Jaringan," *MATRIK J. Manajemen, Tek. Inform. dan Rekayasa Komput.*, vol. 18, no. 1, pp. 1–12, 2018.

[2] J. Balen, D. Vajak, and K. Salah, "Comparative Performance Evaluation of Popular Virtual Private Servers," *J. Internet Technol.*, vol. 21, no. 2, pp. 343–356, 2020.

[3] I. N. B. Hartawan and I. K. S. Satwika, "Rancang Bangun Laboratorium Virtual Berbasis Cloud Computing Di Stmik Stikom Indonesia," *S@Cies*, vol. 7, no. 1. pp. 54–60, 2016.

[4] Red Hat Ansible, "Ansible in Depth," p. 5, 2017.

[5] D. Stiawan, "Fundamental Internetworking Development & Design Life Cycle," 2009.

[6] P. Masek, M. Stusek, J. Krejci, K. Zeman, J. Pokorny, and M. Kudlacek, "Unleashing full potential of ansible framework: University labs administration," *Conf. Open Innov. Assoc. Fruct*, vol. 2018-May, pp. 144–150, 2018.

Jurnal MATRIK
Vol.19 No.2 (Mei) 2020, Hal 347 -357
DOI https://doi.org/10.30812/matrik.v19i2.724

[7] Edureka, "Ansible Cheat Sheet." Brain4ce Education Solutions Pvt. Ltd, p. 1, 2019.

[8] N. Muthiah, A. B. Osmond, and R. Latuconsina, "Live Migration Pada Cloud Computing Berbasis Proxmox Dengan Metode Pre-Copy Live Migration in Cloud Computing Using Pre-Copy Method on Proxmox," *e-Proceeding Eng.*, vol. 6, no. 1, pp. 1432–1441, 2019.

[9] P. S. S. Gmbh, "Proxmox VE Administration Guide Release 6.1," *Documentation*. Proxmox Server Solutions Gmbh, 2020.