Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Issue 1, January 2023
DOI : https://doi.org/10.33395/sinkron.v8i1.11856
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Automation of two Ubuntu servers with Ansible and Telegram as notifications

Gading Haryono Wibowo[1)*], Indrastanti Ratna Widiasari[2)]
[1,2)]Universitas Kristen Satya Wacana, Indonesia
[1)]672016049@student.uksw.edu, [2)]indrastanti.widiasari@uksw.edu

Submitted : Oct 16, 2022 | **Accepted** : Oct 20, 2022 | **Published** : Jan 1, 2023

Abstract: Along with the development of increasingly advanced technology, the demand for servers to support operations increases. Meanwhile, most of the servers are controlled manuallly, one of them is through remote secure shell. So it is necessary to impement a server configuration automation system using Ansible tools on Ubuntu server greatly speeding up the service installation process on both Ubuntu servers in one go. All commands to execute Ansible reside in the Ansible playbook written using the YAML format. The Ansible state executes the requested command sent to the system administrator via Telegram message. The purpose of this study was to determine the application of the server configuration automation system using Ansible tools on Ubuntu Server. This study describes a descriptive qualitative research that examines a problem based on what is observed through certain media. The stages of research carried out in designing Automation of Two Ubuntu Servers with Ansible and Telegram as Notifications. The results of this study display two execution results, namely the status of Success and Ended. Ended state is caused by unreacable or unconnected host. The suggestion for the further development of this research is that the application of the automation system can also be carried out on the Windows Server operating system, not only on the Ubuntu Server.

Keywords: Ansible; Telegram; Nginx.

INTRODUCTION

The development of information technology in a company or agency is very rapid. Along with this, the demand for servers to support operations increases. Meanwhile, most of the servers are controlled manually, one of them is through remote Secure Shell (SSH). Remote SSH can only be done one by one in providing the services needed. Configuration that is done manually can lead to various problems such as the occurrence of human errors and tend to be less effective and efficient. Including when performing large-scale updates on the server, it takes a lot of time to manually configure one by one.

Based on these problems, research is carried out related to an automation system that provides server configuration using Ansible tools on Ubuntu Server which aims to automate the configuration of services on several interconnected servers (Putu et al., 2022). The research is applied to Ubuntu Server to provide services on the server as well as the Nginx service. This study uses the variable files method with YAML format to accommodate server configuration policies that are applied to service installation. Ansible Playbook calls variable files to execute based on parameters provided by Ansible to automate static Ubuntu server configuration. Ansible Playbook is a file that contains a series of tasks or tasks that trigger a particular Ansible module and is written using the YAML format (Chandrawaty et al., 2021).

In addition, to increase the convenience of a System Administrator in monitoring the results of tasks carried out by Ansible, the Telegram notification feature is added. Telegram notifications display information about currently running tasks and the results of those tasks. Telegram can be used on smartphones, tablets, and computers, making it very easy to be accessed anywhere and anytime by the System Administrator (Rifki Afandi et al., 2020).

LITERATURE REVIEW

In previous research, research on Linux container management automation and supervision (LCX) on Proxmox VE using Ansible has resulted in a system that can automate creation, run, stop and delete CT and also automate creation, reset and delete users and permissions. Management of CT, users and permissions with automation using the Ansible tool also makes the CT, user and permission management process more efficient

*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

than done manually. Ansible can be used to automate CT management for creating, executing, stopping and deleting CTs on PVE and also Ansible can be used to automate user & permission management for creating, resetting and deleting users & permissions on PVE. On the subject of network system administration at SMKN 6 Mataram (Hariyadi & Juliansyah, 2018; Pratama & Hariyadi, 2021).

In addition, there is research that discusses the automation carried out to configure and manage servers repeatedly on several servers. Implementation starts from testing connections, partitioning, sharing server IP groups, checking logins. The application of the DevOps method makes the process of delivering and managing information much more practical and agile. Effective and efficient server management requires the right method (Khumaidi, 2021).

DevOps is a combination of two words, namely development and operation, it becomes an explanation that what is meant is tasks related to the programming world, namely the development and operation of computer systems (Tohirin et al., 2020). While Ansible is a software or software that supports the running of DevOps or system administrators who work like applications, development and others (Khumaidi, 2021).

Previous research has been successful in implementing Ansible on LCX monitoring and on DevOps methods. Meanwhile, in this study, Ansible was implemented on the Ubuntu server. This research is needed when automation especially for large-scale migration and very fast error handling by System Administrators. So that with Ansible in the configuration, it is very easy for System Administrators to configure the system, deploy software, and others. Ansible is intended for System Administrators and Linux system developers who need to automate provisioning, configuration, application deployment, and orchestration. In addition, the Telegram notification feature was added to make it easier for System Administrators to monitor the configuration anytime and anywhere more quickly.

Ansible is an automation tool in the field of information technology that makes it easier for users to configure systems, deploy software, and others. Ansible is intended for System Administrators and Linux system developers who need to automate provisioning, configuration, application deployment, and orchestration. Ansible in its implementation to accommodate the server configuration using the variable files method with YAML format. Ansible Playbooks calls variable files to be executed based on the service to be installed. Ansible Playbook is a file that contains a series of tasks that trigger a specific Ansible module and is written using the YAML format (Chandrawaty et al., 2021).

Telegram is a cloud-based instant messaging application, where the focus is on speed and user security. Features on Telegram include text messages, audio, photos, videos and stickers (F. A. Fitriansyah, 2020). The Telegram application was chosen because it is free, lightweight and multi-platform. Telegram can send text messages, photos, videos and documents of any type (doc, zip, mp3, etc.), and can create a group of up to 20,000 members or channels to send unlimited broadcast messages. Apart from that, Telegram also supports end-to-end encrypted voice calls for added security. In addition to these advantages, one of the other advantages of Telegram is the Telegram Bot facility. Telegram bot is a special account that does not require an additional phone number to be registered to the Telegram Server.

This account serves as an interface between the program code and the Telegram server. Telegram is one application that supports this bot. With this bot, it can make it easier for users to create a kind of chat application. Bot is a program that runs on the server side and to get information by using the Telegram Client that has been installed on the server admin mobile device. The use of Telegram Client serves as an interface that displays certain information. In order for the bot to work optimally, good internet access is needed to connect all components to the Telegram server (F. Fitriansyah, 2019).

METHOD

This study describes a descriptive qualitative research that examines a problem based on what is observed through certain media (Fazil & Hendrawaty, 2020). The stages of research carried out in designing Automation of Two Ubuntu Servers with Ansible and Telegram as Notifications can be seen in Figure 1.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Issue 1, January 2023
DOI : https://doi.org/10.33395/sinkron.v8i1.11856
e-ISSN : 2541-2019
p-ISSN : 2541-044X

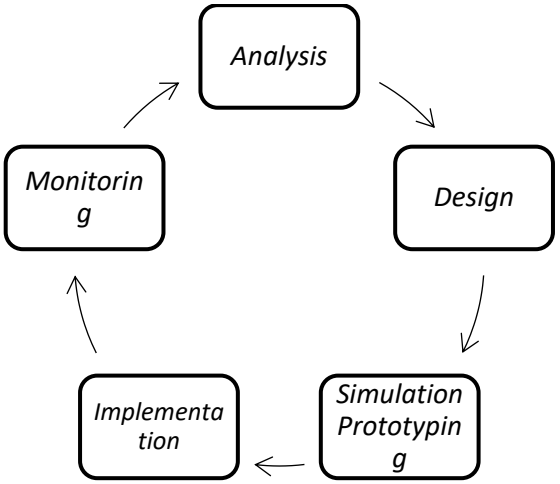

Figure 1 Research Stages (Siswanto et al., 2021).

In Figure 1 is an explanation of the stages of research carried out to design the automation of two Ubuntu servers with Ansible and Telegram as notifications. The stages of this research consist of five stages, namely Analysis, Design, Simulation Prototyping, Implementation, and Monitoring. In the Analysis stage, data collection is carried out in the form of scientific articles related to Ansible Server Linux Ubuntu. The second stage is the Design stage, which is the design of the system built in this research. The third stage is Simulation Prototyping, which is a more detailed design of the tools used and testing the configuration and scenario verification. The fourth stage is Implementation, which is the configuration execution stage to build a system that has been designed previously. The last stage is the Monitoring stage, in this study the Monitoring stage is carried out by the System Administrator using Telegram as a notification.

Needs Analysis

Based on the results of the analysis of the data that has been collected, information is obtained that previous studies focused on the time comparison between manual configuration and automation, but nothing added additional features to make it easier for System Administrators to monitor the results of Ansible. In the design of automation of two Ubuntu servers with Ansible and Telegram as notifications. At this stage of analyzing needs, observations are needed to get the software used and the service to be installed. In this study using Ubuntu 20.0.4 Linux server.

System Design

The design of the server network used in this study, as shown in Figure 2. In the design, it can be seen that there are 2 servers that are connected to each other. The first server is ControlNode-1-Ivory with IP 192.168.1.6/24. ControlNode-1-Ivory as Ansible server acts as automation controller node. The second server is ManagedNode-2-Ivory with IP 192.168.1.7/24.

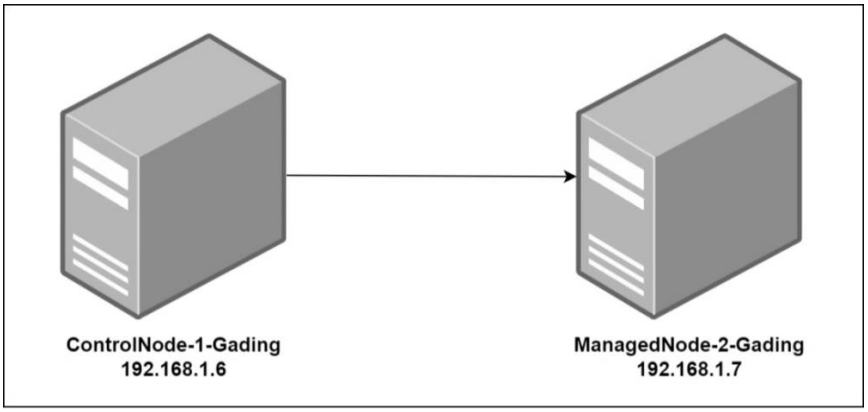

Figure 2 Server Network Design

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Issue 1, January 2023
DOI : https://doi.org/10.33395/sinkron.v8i1.11856
e-ISSN : 2541-2019
p-ISSN : 2541-044X

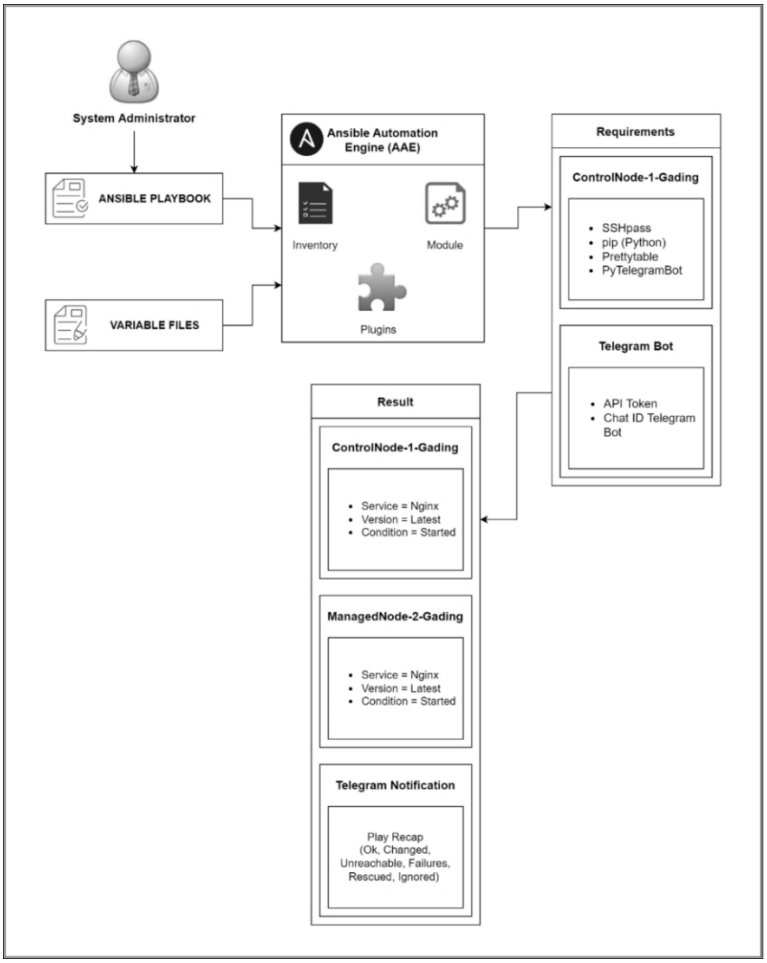

Figure 3 Research Stages

In Figure 3 is the design of the automation system used in this study. The System Administrator creates an Ansible Playbook which contains a series of automation tasks written using the YAML format. In addition, the System Administrator also creates variable files written in the Python programming language. The variable file contains variables related to the Telegram notification message that appears when Ansible starts running and the result of Ansible's execution. The Ansible Playbook when executed interacts with the Ansible Automation Engine (AAE) which contains inventory, modules, templates and plugins. Requirements needed on ControlNode-1-Gading are SSHPass, pip (Python), Prettytable, PyTelegramBot and what is needed from Telegram Bot is API Token and Chat ID Telegram Bot. If it has been successfully executed, the Nginx service with the latest version has been successfully installed and running on ControlNode-1-Gading and ManagedNode-2-Gading. In addition, the Telegram notification containing the play recap has also been successfully received by the System Administrator.

Prototype Simulation

In designing the automation of two Ubuntu servers with Ansible and Telegram as notifications, two Linux Ubuntu 20.04.3 servers are used which are called Control Node and Managed Node. The Control Node has installed Ansible, SSHPass, Python Interpreter, Python Telegram bot API, Prettytable. SSHPass is used to manage SSH connections automatically. Python Interpreter is used to execute instructions written in Python language to display required data on Telegram. Python Telegram Bot API is used to connect between Python and Telegram Bots. Prettytable is used to call table functions in Python code displayed in Telegram messages.

Then in the Ansible Playbook manufacturing section, the program code on the Control Node is compiled according to the design of the service installation automation system for the Managed Node. The trial is divided into two, namely configuration verification and scenarios. Configuration verification is a basic configuration check whether the two servers are ready to use and connected. Meanwhile, the scenario is automation using Anisible Playbook which contains service installation commands on Control Node and Managed Node.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Issue 1, January 2023
DOI : https://doi.org/10.33395/sinkron.v8i1.11856
e-ISSN : 2541-2019
p-ISSN : 2541-044X

RESULT

Installation and Configuration Results

In this study, 2 Ubuntu Linux Servers are used, namely Control Node and Managed Node. Both use Linux Ubutu version 20.04.3 as shown in Figures 4 and 5. On the Control Node there are installations and configurations such as Ansible installation, SSHPass, Python Interpreter, Python Telegram bot API, Prettytable. Figure 6 shows the versions of Anisible and Python already installed on the Control Node. Anisible version used is version 2.9.6 and Python used is version 3.8.10. In Figure 7 shows the version of SSH Pass used is version 1.06 "

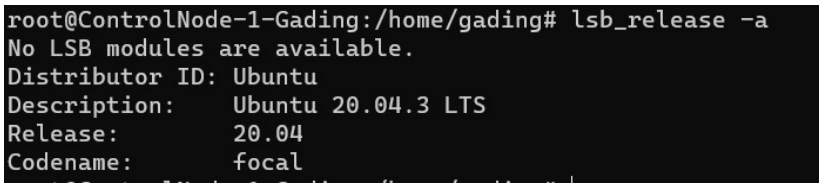

Figure 4. Control Node Installation Results

Based on Figure 4, it is an Ubuntu Linux server using control nodes.

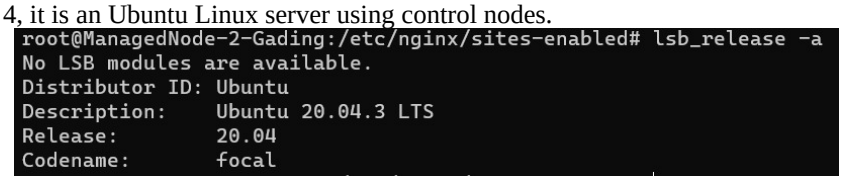

Figure 5. Managed Node Installation Results

Based on Figure 5, it is an Ubuntu Linux server using managed nodes.

Figure 6. Installation Results of Ansible and Python

Based on Figure 6, the versions of Anisible and Python are already installed on the Control Node. Anisible version used is version 2.9.6 and Python used is version 3.8.10.

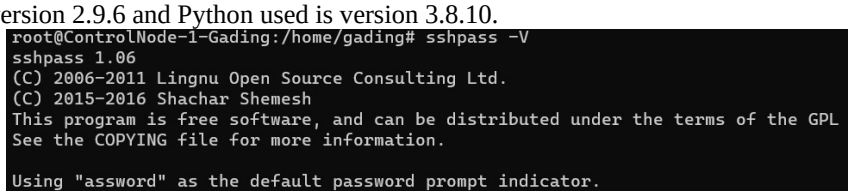

Figure 7. SSH Pass Installation Results

Based on figure 7 shows the version of SSH Pass used to search for version 1.06.

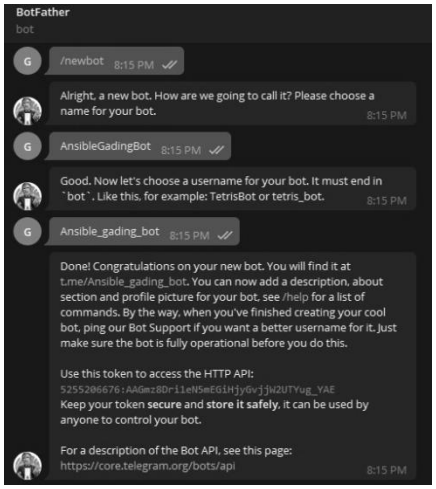

Figure 8 Telegram Token

Based on figure 8, it is used to search for BotFather accounts and create new bots using the new bot command provided by BotFather. If it is successful then get an access code in the form of Application Programming Interface (API).

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Issue 1, January 2023
DOI : https://doi.org/10.33395/sinkron.v8i1.11856
e-ISSN : 2541-2019
p-ISSN : 2541-044X

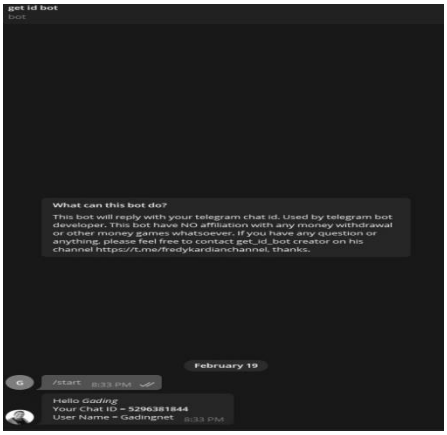

Figure 9 Telegram Chat ID

In Figure 9 it is used to search for Get ID Bot accounts to get Chat ID and Username. The access code and access token will connect the Telegram Bot with Ansible Server.

The Program Code 1 contains the Ansible Default Config which is set on the Control Node. The first part of Defaults line 1 is setting the interpreter to run automatically without any warning. The 2nd line is False so there is no need to check SSHKey. The 3rd line of the command used to prioritize the callback file is named Telegram.

Program Code 1 Ansible Config Default

```
[defaults]
1. interpreter_python  = auto_silent
2. host_key_checking   = False
3. callback_whitelist  = telegram
```

The Program Code 2 contains the Ansible Config Callback which is set on the Control Node. The Ansible Config Callback can be saved and can be opened in the Control Node by using the command "/etc/ansible/plugins". The file contains the Telegram Token and Chat ID obtained as shown in Figure 8 and Figure 9.

Program Code 2 Ansible Config Callback

```
[callback_telegram]
1.  tg_token      =  5213394966:AAFZf2AOeZidn03Hwoocxxxxxxxxxxxxxxx
2.  Tg_chat_id    =  998xxxxxx
```

The Program Code 3 contains Ansible Config Previlege Escalation which is set on the Control Node. Line 1 declares Control Node status to be super user or not. Line 2 declares that the user type is super user (sudo). Line 3 declares the Control Node user name is sudo.

Program Code 3 Ansible Config Privilege Escalation

```
[privilege_escalation]
1. become              = True
2. become_method       = sudo
3. become_user         = root
```

Ansible Inventory Results, Variable Files, and Playbook

Ansible Inventory can be saved and can be opened on the Control Node by using the command "/etc/ansible/hosts". Ansible Inventory as shown in Figure 10 contains a grouping of managed variables, such as ansible_sudo, ansible_sudo_password, ansible_user, ansible_password. Ansible_sudo and ansible_sudo_password are variables of the Ubuntu linux super user and password used. Ansible_user and ansible_password are variables to determine the user used to connect to network devices, namely System Administrator. In addition, the Ansible Inventory declared that the IP used as the Control Node is 192.168.1.6 and the IP used as the Managed Node is 192.168.1.7.

```
[linux]
192.168.1.6 ansible_sudo_user=root ansible_sudo_password=gading ansible_user=gading ansible_password=gading
192.168.1.7 ansible_sudo_user=root ansible_sudo_password=gading ansible_user=gading ansible_password=gading
```

Figure 10 Ansible Inventory

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Issue 1, January 2023
DOI : https://doi.org/10.33395/sinkron.v8i1.11856
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Figure 11. shows the results of the ping check connection between Control Node with IP 192.168.1.6 and Managed Node with IP 192.168.1.7 which are already connected to each other. Ping is done in the Control Node with the command "ansible linux –m ping".

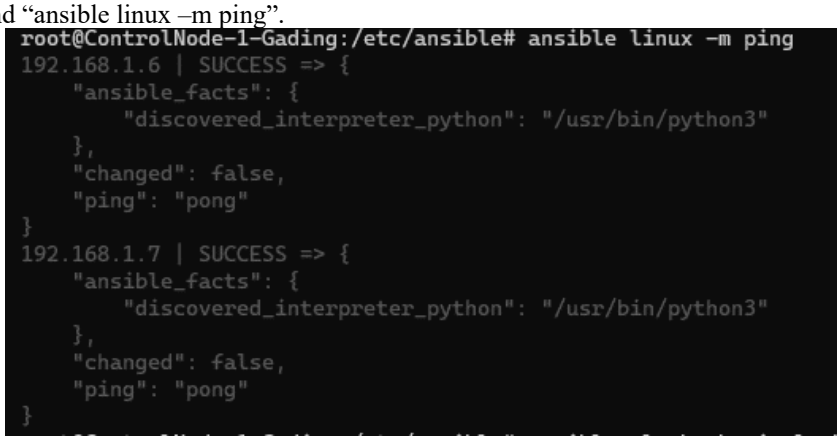

Figure 11. Ping Check Connection

Ansible variables as in Program Code 4 to Figure 14 are written using the Python programming language in a file called telegram.py. In Program Code 4 lines 4 to line 33 are background documentation of the command executed in Python, consisting of the name of the function being called, the type of function being called, the need for the function to be executed, a brief description, the version used, a description. Line 5 calls the callback function of Telegram. Line 6 declares the required callback type which is notification. Lines 7 to 11 describe the requirements that need to be prepared, such as whitelists in the configuration, telebot, prettytable, and latest requests. Line 13 declares a short description of the command being executed. Lines 18 to 25 declare the Telegram Bot Token with the name TG_TOKEN, which is used during the callback_telegram section. Lines 26 to 33 declare the Telegram Bot's Chat ID with the name TG_CHAT_ID, which is used during the callback_telegram section.

Program Code 4 Ansible Variables (1)

```
1.  from __future__ import (absolute_import, division, print_function)
2.  __metaclass__ = type
3.
4.  DOCUMENTATION = '''
5.      callback: telegram
6.      callback_type: notification
7.      requirements:
8.          - whitelist in configuration
9.          - telebot (pip install pyTelegramBotApi)
10.         - prettytable (pip install prettytable)
11.         - latest requests (pip install requests --upgrade)
12.     short_description: Sends play events to a telegram channel
13.     version_added: "2.1"
14.     description:
15.         - This is an ansible callback plugin that sends status updates to a telegram
            channel during playbook execution.
16.         - Before 2.4 only environment variables were available for configuring this
            plugin
17.     options:
18.         tg_token:
19.            required: True
20.            description: telegram bot token
21.            env:
22.               - name: TG_TOKEN
23.            ini:
24.               - section: callback_telegram
25.                 key: tg_token
26.         tg_chat_id:
27.            required: True
28.            description: telegram chat id to post in.
29.            env:
30.               - name: TG_CHAT_ID
31.            ini:
32.               - section: callback_telegram
33.                 key: tg_chat_id
```

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Issue 1, January 2023
DOI : https://doi.org/10.33395/sinkron.v8i1.11856
e-ISSN : 2541-2019
p-ISSN : 2541-044X

In Program Code 5 lines 36 through 42 declare the import function required. Lines 44 to 49 declare if the telebot import from the telebot import API helper is successful, then the value of the HAS_Telebot function day is true, but if it fails, the HAS_Telebot value is false. Lines 51 to 55 declare if import prettytable is successful, then the value of the HAS_prettytable function day is true, but if it fails, the HAS_prettytable value is false. Lines 57 to 84 are ansible callback plugins that send status updates to Telegram during execution.

Program Code 5 Ansible Variables (2)

```
36. import os
37. from datetime import datetime
38.
39. from ansible import context
40. from ansible.module_utils._text import to_text
41. from ansible.module_utils.urls import open_url
42. from ansible.plugins.callback import CallbackBase
43.
44. try:
45.     import telebot
46.     from telebot import apihelper
47.     HAS_TELEBOT = True
48. except ImportError:
49.     HAS_TELEBOT = False
50.
51. try:
52.     import prettytable
53.     HAS_PRETTYTABLE = True
54. except ImportError:
55.     HAS_PRETTYTABLE = False
56.
57. class CallbackModule(CallbackBase):
58.     """This is an ansible callback plugin that sends status
59.     updates to a telegram channel during playbook execution.
60.     """
61.     CALLBACK_VERSION = 2.0
62.     CALLBACK_TYPE = 'notification'
63.     CALLBACK_NAME = 'telegram'
64.     CALLBACK_NEEDS_WHITELIST = True
65.
66.     def __init__(self, display=None):
67.
68.         super(CallbackModule, self).__init__(display=display)
69.
70.         if not HAS_TELEBOT:
71.             self.disabled = True
72.             self._display.warning('The `telebot` python module is not '
73.                                   'installed. Disabling the Slack callback '
74.                                   'plugin.')
75.
76.         if not HAS_PRETTYTABLE:
77.             self.disabled = True
78.             self._display.warning('The `prettytable` python module is not '
79.                                   'installed. Disabling the Slack callback '
80.                                   'plugin.')
81.
82.         self.playbook_name = None
83.         self.play = None
84.         self.now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
```

In Program Code 6 lines 86 to 105 are functions used to declare callback modules. This condition can only be executed if the telegram token and telegram chat ID are available. If it is not available, a warning will appear as in line 96 to line 98 and line 102 to line 105. Line 107 to line 109 is a function used to run the ansible playbook function. Lines 111 to 128 are functions used to display a notification message when ansible starts executing. Lines 114 to 118 are used to set the message title. Line 119 calls the time command according to the format that has been set according to the Program Code 6 line 84. Line 120 is the command used to display the storage area of the playbook file. Lines 121 to line 123 are used to display running hosts. Lines 125 to line 127 to display the running tags.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Issue 1, January 2023
DOI : https://doi.org/10.33395/sinkron.v8i1.11856
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Program Code 6 Ansible Variables (3)

```
86.     def set_options(self, task_keys=None, var_options=None, direct=None):
87.
88.         super(CallbackModule, self).set_options(task_keys=task_keys,
    var_options=var_options, direct=direct)
89.
90.         self.tg_token = self.get_option('tg_token')
91.         self.tg_chat_id = self.get_option('tg_chat_id')
92.
93.         if self.tg_token is None:
94.             self.disabled = True
95.             self._display.warning('tg_token was not provided. The '
96.                                   'tg_token can be provided using '
97.                                   'the `TG_TOKEN` environment '
98.                                   'variable.')
99.
100.           if self.tg_chat_id is None:
101.               self.disabled = True
102.               self._display.warning('tg_chat_id was not provided. The '
103.                                     'tg_chat_id can be provided using '
104.                                     'the `TG_CHAT_ID` environment '
105.                                     'variable.')
106.
107.       def v2_playbook_on_start(self, playbook):
108.
109.           self.playbook_name = os.path.abspath(playbook._file_name)
110.
111.       def v2_playbook_on_play_start(self, play):
112.           self.play = play
113.
114.           title = [
115.               '<u><b>Ansible:</b></u> <b>STARTED</b> ⚙'
116.           ]
117.
118.           msg_items = [' '.join(title)]
119.           msg_items.append('\n         time: ' + '<code>' + str(self.now) + '</code>')
120.           msg_items.append('playbook: ' + '<code>' + self.playbook_name + '</code>')
121.           msg_items.append('       hosts:')
122.           for host in play.hosts:
123.               msg_items.append('<code>     - ' + host + '</code>')
124.           msg_items.append('       tags:')
125.           for tag in play.only_tags:
126.               msg_items.append('<code>     - ' + tag + '</code>')
127.           msg = '\n'.join(msg_items)
128.           self.send_msg(msg=msg)
```

In the 7 line program code 130 to 144 it is used to set the message to be sent to Telegram if Ansible fails to execute. Lines 133 to 136 are used to set the message title. Line 137 calls the time command according to the format that has been set according to Figure 15 line 84. Line 138 is the command used to display the storage area of the playbook file. Line 139 is used to display running hosts. Line 140 to display stderr (Standard Error).

Program Code 7 Ansible Variables (4)

```
130.     def v2_runner_on_failed(self, result, ignore_errors=False):
131.
132.         msg = []
133.         title = [
134.             '<u><b>Ansible:</b></u> <b>FAILED ✖</b>'
135.         ]
136.         msg_items = [' '.join(title)]
137.         msg_items.append('\n         time: ' + '<code>' + str(self.now) + '</code>')
138.         msg_items.append('playbook: ' + '<code>' + self.playbook_name + '</code>')
139.         msg_items.append('        host: ' + '<code>' + result._host.get_name() +
     '</code>')
140.         msg_items.append('      stderr: ' + '<code>' + result._result['stderr'] +
     '</code>')
141.
142.         msg = '\n'.join(msg_items)
143.
144.         self.send_msg(msg=msg)
```

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Issue 1, January 2023
DOI : https://doi.org/10.33395/sinkron.v8i1.11856
e-ISSN : 2541-2019
p-ISSN : 2541-044X

In the Program Code 8 lines 146 to 166 are used to display information about the execution of the playbook. Lines 151 to 152 are used to set the displayed table, namely Host, Ok, Changed, Unreachable, Failures, Rescued, Ignored. Line 154 declares that Failures is false. Line 155 declares that Unreacable is false. Lines 157 to line 163 declare if Failures and Unreacable are more than 0 then it is True. Rows 165 to 166 display Host, Ok, Changed, Unreachable, Failures, Rescued, Ignored data according to the column.
Lines 168 to 184 are functions that are used to display notification messages if the conditions of lines 154 to 163 are met or not. Lines 169 to 177 are used to set the message title. However, there is something different from the previous function, where if Failures and Unreacable have a value of more than 0 (zero) then a "✗" (cross) sign appears if it is less than or equal to 0 (zero) then a "☑" (tick) sign will appear. as in line 170 to line 177. Line 178 calls the time command according to the format that has been set according to Program Code 6 line 84. Line 179 is the command used to display the storage area of the playbook file.

Program Code 8 Ansible Variable (5)

```
146.       def v2_playbook_on_stats(self, stats):
147.           """Display info about playbook statistics"""
148.
149.           hosts = sorted(stats.processed.keys())
150.
151.           t = prettytable.PrettyTable(['Host', 'Ok', 'Changed', 'Unreachable',
152.                                        'Failures', 'Rescued', 'Ignored'])
153.
154.           failures = False
155.           unreachable = False
156.
157.           for h in hosts:
158.               s = stats.summarize(h)
159.
160.               if s['failures'] > 0:
161.                   failures = True
162.               if s['unreachable'] > 0:
163.                   unreachable = True
164.
165.               t.add_row([h] + [s[k] for k in ['ok', 'changed', 'unreachable',
166.                                               'failures', 'rescued', 'ignored']])
167.
168.           msg = []
169.           title = '<u><b>Ansible:</b></u> <b>ENDED</b>'
170.           if failures or unreachable:
171.               msg_items = [
172.                   title + ' ✗'
173.               ]
174.           else:
175.               msg_items = [
176.                   title + ' ✓'
177.               ]
178.           msg_items.append('\n         time: ' + '<code>' + str(self.now) + '</code>')
179.           msg_items.append('playbook: ' + '<code>' + self.playbook_name + '</code>')
180.           msg_items.append('<code>\n%s\n</code>' % t)
181.
182.           msg = '\n'.join(msg_items)
183.
184.           self.send_msg(msg=msg)
```

In Program Code 9 is playbook.yml which is used to execute service installation automation for both Ubuntu servers. Line 1 is the beginning of a YAML document marked with three hypens. Line 2 is used to specify the name of Ansible which is “TASK INSTALLING AND RUNNING NGINX”. Line 3 is used to specify the host list or host group as the location for the execution of the task, namely linux. Line 4 to enable privilege escalation. Line 6 is used to declare tasks. Line 7 to declare a task with the name " NGINX INSTALLING ...". Lines 8 to 10 are commands to install the latest version of the Nginx service on an Ubuntu Linux server. Line 12 declares a second task with the name “STARTING NGINX…”. Lines 13 to sixteen are commands to run the Nginx service that is already installed on the Ubuntu Linux server. Line 18 declares a handle that is executed only when there is a change to the machine.

Program Code 9 Ansible Playbook

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Issue 1, January 2023
DOI : https://doi.org/10.33395/sinkron.v8i1.11856
e-ISSN : 2541-2019
p-ISSN : 2541-044X

```
---
- name: TASK MENGINSTAL DAN MENJALANKAN NGINX
  hosts: linux
  become: true

  tasks:
     - name: SEDANG MENGINSTALL NGINX . . . .
       apt:
          name: ['nginx']
          state: latest

     - name: MEMULAI NGINX . . . .
       service:
          name: nginx
          state: started
          enabled: yes
```

Service Execution Results

Figure 12 shows the results of the successful execution of Ansible Playbook. It can be seen that there are 3 (three) tasks being executed, namely Gathering Facts, installing Nginx, and starting Nginx. The first task, Gathering Facts, is a task to collect data from the Managed Node and return it to the Control Node. The second task, "Installing Nginx" is a task to install the Nginx service on the Control Node and Managed Node. The third task, “Starting Nginx” is a task that is used to run services on the Control Node and Managed Node after installation is complete. In the Play Recap section, there are 7 (seven) Ansible execution results statuses. Status Ok declares ansible execution has been executed successfully. Status Changed declares a change to the server when all required conditions are met. Unreachable state declares if Ansible cannot connect to the host. The Failed state declares if Ansible has a connection failure, execution issue, or syntax error. The Skipped state declares when Ansible is in a state that does not match the hostname. The Rescued state declares Ansible to actively handle the consequences of the error. The Ignored state declares if Ansible ignores the failed task and continues execution and it does not affect the connection error.

Based on the results of executing playbook.yml in Control Node and Managed Node all tasks were successful. The play recap shows that each server has a value of 3 (three) for Ok status because it has successfully executed 3 (three) tasks.

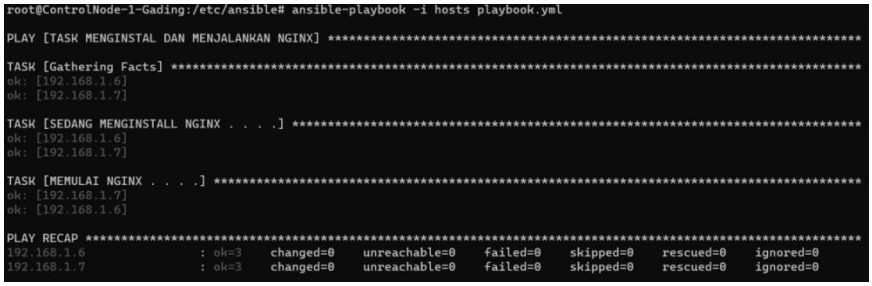

Figure 12 Ansible Playbook Run Task (Success)

In Figure 13 is a Telegram notification sent to the System Administrator which contains when Ansible starts executing and the results of Ansible's execution. The format displayed in the Telegram notification corresponds to the format set in Ansible Variable using the Python programming language. The execution results displayed on the Telegram notification are taken based on the play recap in the Ansible Playbook Run Task. From the Telegram message it can be seen that the execution process only takes a fraction of a second for both servers.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 7, Issue 1, January 2023
DOI : https://doi.org/10.33395/sinkron.v8i1.11856
e-ISSN : 2541-2019
p-ISSN : 2541-044X

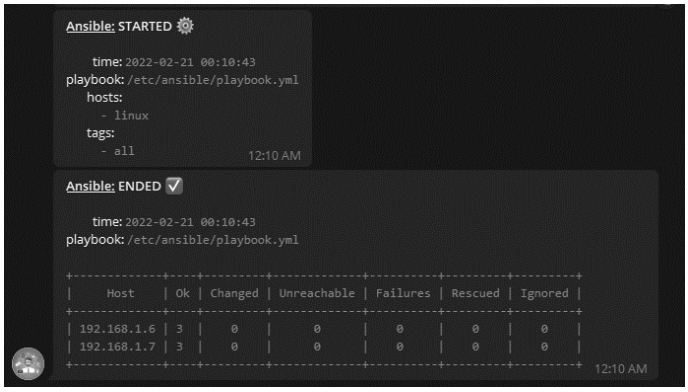

Figure 13 Telegram Notifications (Success)

In Figure 14 is the result of the failed execution of Ansible Playbook. It can be seen that there are 3 (three) tasks being executed, namely Gathering Facts, installing Nginx, and starting Nginx. The first task, Gathering Facts, failed to be executed by the Control Node because it was not connected to the host, but was successfully executed by the Managed Node with the status "Ok". The second task, “Installing Nginx” was only successfully executed by the Managed Node. The third task, “Starting Nginx” was only successfully executed by the Managed Node.

Figure 14 Ansible Playbook Run Task (Failed)

In the Play Recap section, there are 7 (seven) Ansible execution results statuses. Based on the results of the execution of playbook.yml in the Control Node it failed to execute in the first task because it failed to connect to the host so that the unreacable part was worth 1 (one). Then on the Managed Node all tasks were successfully executed so that the play recap has a value of 3 (three) for Ok status because it has successfully executed 3 (three) tasks. In Figure 15 is a Telegram notification sent to the System Administrator containing Ansible starting to be executed and the results of the execution. The execution results displayed on the Telegram notification are taken based on the play recap as shown in Figure 14.

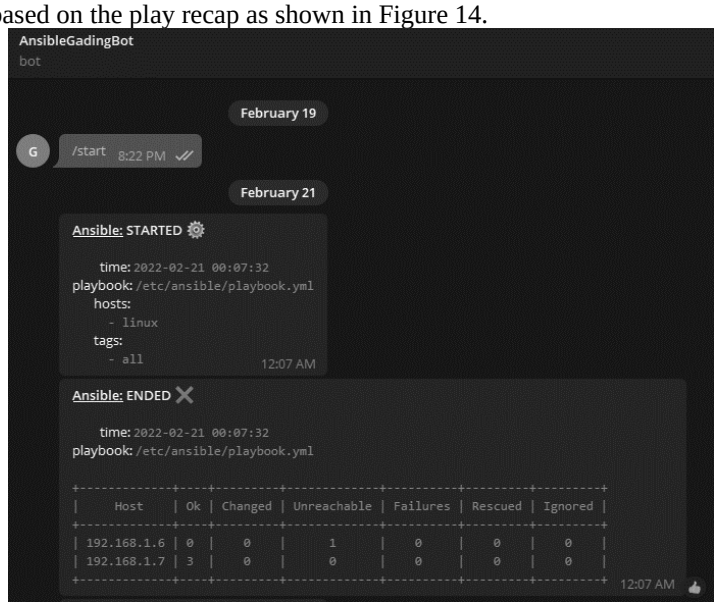

Figure 15 Telegram Notifications (Failed)

*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

DISCUSSIONS

Along with the development of increasingly advanced technology, the demand for servers to support operations is increasing. Meanwhile, most servers are controlled manually, one of which is via Remote Secure Shell (SSH). So it is necessary to implement a server configuration automation system using Ansible tools on Ubuntu Server, greatly speeding up the service installation process on both Ubuntu Servers at once. All commands to execute Ansible directly in the Ansible Playbook are written using the YAML format. Ansible Status executes the requested command which is sent to the System Administrator via Telegram message. Telegram messages contain Play Recap Ansible information, such as time, host, execution status. The displayed execution states are Ok, Changed, Unreacable, Failures, Rescued, and Ignored. This study displays two execution results, namely the status of Success and End. The ended state is caused by an inaccessible or unconnected host. Suggestions for further development of this research is the application of the automation system can also be done on the Windows Server operating system, not only on the Ubuntu Server.

CONCLUSION

Based on the results of implementing a server configuration automation system using the Ansible tools on Ubuntu Server, it greatly speeds up the service installation process on both Ubuntu Servers in one go. All commands to execute Ansible reside in the Ansible Playbook written using the YAML format. The Ansible state executes the requested command sent to the System Administrator via a Telegram message. The Telegram message contains Play Recap Ansible information, such as time, host, execution status. The execution status displayed is Ok, Changed, Unreacable, Failures, Rescued, and Ignored. This study displays two execution results, namely the status of Success and Ended. Ended state is caused by unreacable or unconnected host. The suggestion for the further development of this research is that the application of the automation system can also be carried out on the Windows Server operating system, not only on the Ubuntu Server.

REFERENCES

Chandrawaty, Y., Anggrena, N. M., & Hariyadi, I. P. (2021). Implementasi Ansible Playbook Untuk Mengotomatisasi Manajemen Konfigurasi VLAN Berbasis VTP Dan Layanan DHCP. *Jurnal Bumigora Information Technology (BITe)*, *3*(2), 107–122. https://doi.org/10.30812/bite.v3i2.1577

Fazil, F., & Hendrawaty, H. (2020). Rancang Bangun Sistem Inventaris Barang Berbasis Web Dengan Pemanfaatan Bot Telegram (Studi Kasus PT. PLN (Persero) Unit Pelaksana Pembangkitan Nagan …. *Prosiding Seminar Nasional Politeknik …*, *3*(1), 152–159. http://e-jurnal.pnl.ac.id/index.php/semnaspnl/article/view/1678

Fitriansyah, F. (2019). Analisis Kebutuhan Pengembangan Pembelajaran Penulisan Naskah PR I. *Cakrawala*, *19*(1), 79–86. doi: https://doi.org/10.31294/jc.v19i1

Fitriansyah, F. A. (2020). Penggunaan Telegram Sebagai Media Komunikasi Dalam Pembelajaran Online. *Jurnal Humaniora Bina Sarana Informatika*, *20*(Cakrawala-Jurnal Humaniora), 113. http://ejournal.bsi.ac.id/ejurnal/index.php/cakrawala

Hariyadi, I. P., & Juliansyah, A. (2018). Analisa Penerapan Private Cloud Computing Berbasis Proxmox Virtual Environment Sebagai Media Pembelajaran Praktikum Manajemen Jaringan. *MATRIK : Jurnal Manajemen, Teknik Informatika Dan Rekayasa Komputer*, *18*(1), 1–12. https://doi.org/10.30812/matrik.v18i1.329

Khumaidi, A. (2021). Implementation of Devops Method for Automation of Server Management Using Ansible. *Jurnal Transformatika*, *18*(2), 199. https://doi.org/10.26623/transformatika.v18i2.2447

Pratama, M. A. A., & Hariyadi, I. P. (2021). Otomasi Manajemen dan Pengawasan Linux Container (LCX) Pada Proxmox VE Menggunakan Ansible. *Jurnal Bumigora Information Technology (BITe)*, *3*(1), 82–95. https://doi.org/10.30812/bite.v3i1.807

Putu, I., Pratama, A. E., Suarnata, P. B., & Putra, W. (2022). *Pengujian IaC Berbasis DevOps dan Ansible Menggunakan Metode Black Box Testing*. *15*(2), 1979–276. https://doi.org/10.30998/faktorexacta.vx3ix.xxxx

Rifki Afandi, M., Hatta, P., & Efendi, A. (2020). Otomatisasi Perangkat Jaringan Komputer Menggunakan Ansible Pada Laboratorium Komputer. *SMARTICS Journal*, *6*(2), 48–53. https://doi.org/10.21067/smartics.v6i2.4599

Siswanto, D., Priyandoko, G., Tjahjono, N., Putri, R. S., Sabela, N. B., & Muzakki, M. I. (2021). Development of Information and Communication Technology Infrastructure in School using an Approach of the Network Development Life Cycle Method. *Journal of Physics: Conference Series*, *1908*(1). https://doi.org/10.1088/1742-6596/1908/1/012026

Tohirin, T., Utami, S. F., Widianto, S. R., & Mauludyansah, W. Al. (2020). Implementasi DevOps Pada Pengembangan Aplikasi e-Skrining Covid-19. *Multinetics*, *6*(1), 15–20. https://doi.org/10.32722/multinetics.v6i1.2764

*name of corresponding author