

Ansible: A Reliable Tool for Automation

Mohammed Daffalla Elradi

Communication Systems Engineering Department, University of Science and Technology, Khartoum, Sudan

Email: mohd_daf_elradi@hotmail.com

Abstract. Technologies have advanced drastically throughout the last decade, where IT professionals are challenged with the task to install, configure, manage and troubleshoot a considerable amount of IT infrastructure. The challenge is to keep your IT environment consistent and up to date. Here, automation hops into the scene, as it plays a pivotal role in daily activities of modern IT. Automation provides an approach to streamline multiple processes across different platforms to guarantee an efficient, scalable, collaborative and cost-effective infrastructure management. There are many automation tools that are used nowadays, each has its pros and cons. In this paper Ansible was used to provide a complete web app due to the fact that it is agentless, which is more flexible in terms of compatibility and requires no installation on managed nodes and other distinct features. It was found to be a reliable tool in tasks automation, as it reduced the time required to obtain the final outcome by 70%, which is approximately 5.6 times less than when no automation was used.

Keywords. Automation, agentless, ansible, yaml, playbook.

1. Introduction

As technology advances rapidly and technologies are being deployed here and there in various domains i.e., Cloud Computing, Cybersecurity, Artificial Intelligence (AI), Internet of Things (IOT), etc. As a result, devices are increasing infectious and the need for orchestration and automation will be a necessity and not an option [1].

Orchestration is the capability to manage multiple systems and allow them to seamlessly operate, which involves workflow automation, monitoring performance, optimizing how repeatable tasks should be implemented to save time and focus more on increasing efficiency [2].

In the other hand, automation, can be defined as the act of using an approach to perform repetitive tasks to save time and boost productivity as well as avoid errors as possible [3]. Looking at the two definitions, it might feel a little bit similar and makes it hard to distinguish between the two terms but as a matter of fact, the two terms are different yet complement each other.

Automation is finding a way to run tasks without human intervention while orchestration is having multiple tasks from which (some might not be automated) to conduct a job in a cohesive manner. It takes action relying on the outcome of a task and determine the following to be implemented [4].

Automation and orchestration can be applied to miscellaneous fields like but not limited to:

- *Manufacturing*

It can help optimizing production lines and diminish machinery downtime via PLC (Programmable Logic Controller).

- *Healthcare*

It can be used to track patients' data, schedule appointments, and process claims [5].

- *Business*

Helps in essential areas like customer service represented in automated response chatbots, in financial auditing and reports generation [6] and human resources operations.

- *Information Technology Operations*

It can greatly support broad domains of Information Technology like Networks, DevOps (An approach that emphasizes collaboration between Development and Operations teams), Cloud Computing and Security.

There are also many other fields where automation and orchestration might be applied [7]. The context of this paper will mainly focus on the application of automation and orchestration in the domain of Information Technology.

Technology has expanded drastically throughout the last decade which was a matter of concern for professionals as they turned out to adopt automation as a business-enabler and not an optional mean to smoothly run day-to-day activities.

Automation is used in installation, configuration, management, troubleshooting, update, migration and even decommissioning of IT infrastructure. These processes will be discussed briefly in the following sections.

1.1. Installation

Automation can be used for provisioning of software packages of any OS (Operating Systems), which can be a time-consuming process. Thus, allowing professionals to focus more in productive tasks. The process also involves running updates and patching [8].

1.2. Configuration

Configuration management is of pivotal importance, especially in environments with a considerable amount of

assets. As there is a concern that the same configuration be applicable to every single node to guarantee a unified infrastructure and avoid compatibility issues [9].

1.3. Management

Management of nodes involves checking their status for the purpose of indicating performance metrics, automation of such a process will bring an insightful overview about the infrastructure been managed [10].

1.4. Troubleshooting

Automation can be merely utilized in troubleshooting issues through log inspection for anomalies, checking for service status and then do something accordingly and many other use cases.

All the previously mentioned operations can utilize automation to achieve an impressive outcome in a timely-manner.

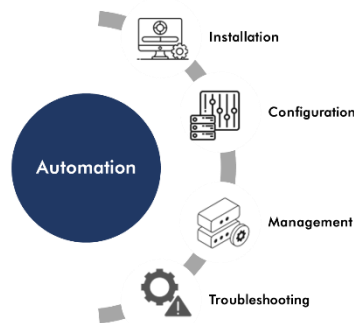


Figure 1. Some Automation processes scope

There are many tools that are used for automating these processes. They share some commonalities like packages and software provisioning, configuration management but we will discuss their architecture model, which indicates the way the automation tools are designed and integrated including processes and workflows involved and articulate the overall strategy been followed. The benefits of having architectural model for automation tools are:

- *Consistency*

To ensure that automation process is consistent across miscellaneous systems and been applied to obtain the desired outcome.

- *Efficiency*

By reducing the time and effort it takes to manually complete a task and ensuring that no error is expected due to manual interaction, hence increasing the productivity.

- *Collaboration*

It defines how automation allows for seamlessly integrating with different systems and how teams manage to participate in defining the streamline for optimum results.

- *Scalability*

Ability to accommodate to the changes and expanding on-demand.

- *Risk Management*

The architecture model plays a pivotal role in assessing the risks that might be associated with the automation processes in regard to lack of proper planning and incompatibility.

- *Cost Reduction*

It adopts the optimum utilization of resources and time to be allocated for more productive actions.

Hereby, we will take a look at some popular tools used in automation They are going to be highlighted in brief in the following section.

- *Puppet*

Puppet allows the usage of agent-master architecture or stand-alone architecture. Puppet master delivers catalogs (which are the instructions for a desired state) requested by agents running on managed nodes. Puppet agents periodically send facts (System information collected from the node) to the master and request catalogs to be applied and reports back to the master [11].

- *Chef*

Adopts client-server architecture and uses Chef client to retrieve the instructions on what to be configured in a pull model. The Chef workstation (Where the administrator interacts with the Chef server) is used to create and test Cookbooks, which are collection of related recipes (Units of code that define a certain state) as well as other resources that are used to manage configuration [12].

- *SaltStack*

Uses client-server architecture, at its core is the Salt Master that manages the configuration files named Salt States for all Salt Minions (Nodes controlled by Salt Master). In addition, there is Salt Formulas, which are pre-built configuration templates that can be used to automate tasks. Sensitive data such as passwords can be securely stored in some-

thing called Salt Pillar [13].

- *Terraform*

It is basically focused on cloud provisioning and infrastructure management. It enables the creation and management of virtual machines, containers, databases, load balancers and security groups from a unified platform.

Terraform store the state of the infrastructure in a state file and Terraform configuration files are used to write the required task. Once pushed, simply via API calls to the cloud provider; The state file is changed accordingly [14].

- *Ansible*

It adopts a client-server architecture and uses SSH (Secure Shell) to communicate with the managed nodes. The control machine is used to create playbooks (Set of commands to be executed sequentially). Also, ad-hoc commands can be utilized to run individual commands on nodes.

Modules are the building blocks of Ansible playbooks which are used to define the state on the managed nodes [15].

Table 1 shows the listing of each automation tools in terms of agent-based and agentless architecture model.

TABLE 1. AGENT-BASED AND AGENTLESS AUTOMATION TOOLS LISTING

<i>Automation Tool</i>	<i>Agent-based</i>	<i>Agentless</i>
Puppet	✓	-
Chef	✓	-
SaltStack	✓	-
Terraform	-	✓
Ansible	-	✓

As you have seen that most of the previously-mentioned tools use a client-server architecture, which can encounter some issues represented in the complexity to manage software components on both client and server, which might lead to inconsistency. Also, another concern is the connectivity speed between client and server might cause delay or failure in running automation tasks.

In this paper, we will adopt Ansible due to the fact that it uses agentless architecture model, which overcomes the limitations caused by the client-server architecture.

Agentless architecture offers simplicity, flexibility and reduced overhead, as there is no need for the installation of agents (Which might require specific agent for each platform) in endpoints. Table 2 illustrates the comparison between agent-based and agentless automation tools.

TABLE 2. AGENT-BASED AND AGENTLESS AUTOMATION TOOLS COMPARISON

<i>Feature</i>	<i>Agent-based</i>	<i>Agentless</i>
Installation	Requires installation of target agents on target devices	No installation required
Compatibility	Can be limited by some compatibility issues with different systems	More flexible in terms of compatibility
Configuration	More complex configuration process due to the need to configure	No agents to configure, which makes it easier
Security	Might pose some security risks within the agent software	Considered more secure as there are no agents
Scalability	Limited by the number of agents on each system	Can scale without additional software
Resource Usage	Consumes significant system resources	Consumes fewer resources

From Table 2, it can be observed that agentless automation tools provide better performance in every feature need to achieve a smooth automation process.

The following sections will discuss how Ansible (Agentless) can be utilized to achieve a smooth automation process for multiple use cases.

2. Literature review

Automation tools have been recently used excessively for automating some routine tasks like installation, configuration and management of IT infrastructure either on-premise or on-cloud. These tools provide a broad set of capabilities that proof to be efficient in complex environments, making them an indispensable part of the modern IT orientation.

In [16], the authors highlighted the increasing demand for cloud technology and the popularity it has gained and how DevOps teams are under constant pressure to automate and deliver deployments to the cloud. Also, the criticality for server services to be installed, configured and been run as quick and consistent as possible. By using automation tools, tasks that take hours to complete can be done within minutes.

Kodali, Sailesh [17] developed an automated system that streamlines server maintenance processes and reduce the risk of errors using Puppet. The tool had significantly helped in identifying potential losses and increased the company's profits as well as secure access to the data in servers.

Chef automation was used in [18] to automate the installation and management of packages in production environments achieving a distributed system architecture and its impact on security reliability and repeatability.

Michael Howard [19] focused in code control management, which is a feature provided by Terraform, as well as automating infrastructure provisioning, mainly on cloud.

S, Likitha Implemented Ansible to maintain consistency and reduce manual intervention to install and configure Apache Tomcat on endpoints [20]. Ansible is gaining popularity over other configuration management tools due to its ease of use and being an open-source.

3. Method

This paper will highlight the features that make Ansible a reliable tool for automating a broad range of tasks ranging from packages installation, configuration, management and troubleshooting. Definitely, that would have a considerable contribution to resources utilization, increasing efficiency, reducing compatibility and scalability issues.

To get started with ansible, it was installed in a virtualized server referred to as (Controller) with the specifications detailed in Table 3 below.

TABLE 3. CONTROLLER SPECIFICATIONS

<i>Specification</i>	<i>Details</i>
System	VMware® Workstation 17 Pro
Processor Cores	4
Hard Disk	100 GB
RAM	8 GB
Operating System	Alma Linux 8.7

Ansible was installed following the official guide on the server mentioned in Table 3. Regarding the client side, referred to as (Managed node) where modifications are to be implemented, it was implemented in an endpoint described in Table 4.

TABLE 4. MANAGED NODE SPECIFICATIONS

<i>Specification</i>	<i>Details</i>
System	VMware® Workstation 17 Pro
Processor Cores	4
Hard Disk	100 GB
RAM	8 GB
Operating System	Alma Linux 8.7

3.1. Ansible Architecture

Since Ansible is agentless, it is simple to deploy and requires no additional custom security infrastructure. It is also easy to set the required task that needs to be automated in a straight-forward and easy human-readable language with the utilization of YAML (Yet Another Markup Language), which uses indentation to define relationships between data elements [15]. Ansible architecture is fairly simple and includes the use of modules, playbooks, inventories and can also be extended with plugins. They will be discussed in the following section. Figure 2 depicts the architecture of Ansible.

As it can be observed from Figure 2, the architecture of Ansible is simple and will be highlighted as follows:

- The administrator logs into the Ansible controller, with the necessary permissions and privileges.
- Then, SSH keys must be generated and copied to the managed nodes, so as to obtain password-less access.
- The administrator should create an inventory file which involves the IP addresses or hostnames of the endpoints where a task is to be implemented.
- Then creates the task. Here there are multiple options ranging from Modules (Pre-built automation tasks), Ad-hoc commands (Commands that are directly run from the controller), can also involve Plugins (Extensible functionalities) and Playbooks (Series of tasks to be implemented).
- Ansible connects to the managed nodes via SSH and implement the desired task.
- If necessary, notifications are sent to the administrator about the status of the task or any errors that might have

occurred.

Ansible disconnects from the managed nodes once the tasks are completed.

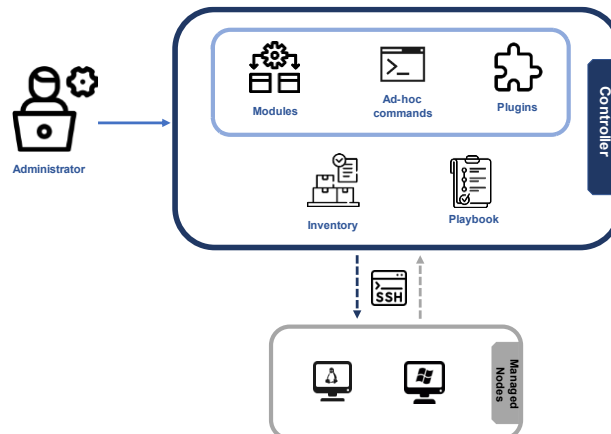


Figure 2. Ansible Architecture

3.2. Modules

Are pre-built reusable automation tasks that provide many capabilities like modules for installing packages, modules for creating, modifying or deleting files and directories as well as another for users and permissions. In addition to modules to start, stop, enabling or disabling services.

3.3. Ad-hoc Commands

Are designed to perform one-time tasks in a quick way, without the need for creating a playbook. For example, you want to check the status of a service in multiple managed nodes.

3.4. Plugins

Plugins are intended to extend the functionality of Ansible, allowing for more complex tasks. They fall into categories as action, filter, connection, call-back and lookup.

3.5. Inventory

Is a simple file that contains information like IP addresses and hostnames of the nodes to be managed by Ansible. Nodes can be grouped by function, role, location or other factors as desired. This makes maintaining your infrastructure more manageable.

3.6. Playbook

Is a sequence of instructions to achieve a desired outcome on managed nodes, written in YAML. It is easy to write and read. It is idempotent, which means when run multiple times, only the required changes are conducted.

3.7. Roles

Are a way to break down the automation code into smaller, manageable units that make it easier to test, debug and troubleshoot. Once created, it can be reused across multiple playbooks.

The rest of this paper will go throughout the process of creating a playbook for automating time-consuming tasks. It is important to note that the use case might differ according to your environment.

4. Result

This section describes the results obtained after implementing Ansible. As been discussed earlier that automation is of great importance as it maintains consistency across your environment, utilizes time to let administrators focus in more productive tasks and avoid errors that might arise from manual interaction.

Ansible automation tool adopted in this paper aims at providing a highly reliable and scalable task automation approaches to achieve a reliable automation process by deploying and configuring a web application composed of Nginx, MySQL and Tomcat.

The first step is creating the inventory file, where the IP of the managed node will be added to “webservers” group as follows:

[webservers]

10.10.20.249

We will utilize Ansible Roles to segment our automation task to smaller chunks. So, we firstly create a directory called “webapp” inside “roles” directory, inside there are sub-directories “templates” where jinja2 files (A powerful templating language used to generate dynamic content based on variables and expressions) will be used to provide the configuration files for each web app component, “tasks” will contain the tasks required to install the webapp components and “files” where a static ‘index.html’ file will be used to demonstrate the success of the process.

Each of the files mentioned will be highlighted below.

In the “templates” directory, three jinja2 files ‘my.cnf.j2’, ‘nginx.conf.j2’ and ‘server.xml.j2’ were created for mysql, nginx and tomcat respectively. The files are meant for configuring each requirement. The defaults for these files are found online.

Inside the “tasks” directory, a file called “mysql.yml” has been added to install and configure MySQL with the following content:

```
---

- yum:

  name: mysql-server

  state: present

- template:

  src: templates/my.cnf.j2

  dest: /etc/my.cnf

- service:

  name: mysqld

  state: started

  enabled: yes

- firewalld:

  service: mysql

  permanent: yes

  state: enabled

  immediate: yes

- mysql_user:

  name: root

  password: "{{ mysql_root_password }}"

  update_password: always

  login_unix_socket: /var/run/mysqld/mysqld.sock
```

Noting that the `{{ mysql_root_password }}` has been encrypted using Ansible Vault.

ansible-vault encrypt secret

The file named “secret” involved `mysql_root_password` and a password was added to be used later to run the playbook

The other component for the webapp is the nginx, where it consists of the following:

```
---
- yum:
  name: nginx
  state: latest

- template:
  src: templates/nginx.conf.j2
  dest: /etc/nginx/nginx.conf

- systemd:
  name: nginx
  state: started
  enabled: true
  daemon_reload: true
```

Follows are the tasks required to install and configure tomcat.

```
---
- group:
  name: tomcat

- user:
  name: tomcat
  group: tomcat

- file:
  path: /opt/tomcat
  state: directory
  mode: 0755

- unarchive:
  src: https://dlcdn.apache.org/tomcat/tomcat-11/v11.0.0-M4/bin/apache-tomcat-11.0.0-M4.tar.gz
```

```
dest: /opt/tomcat

remote_src: yes

extra_opts: [--strip-components=1]

- file:

  path: /opt/tomcat

  owner: tomcat

  group: tomcat

  mode: "u+rwx,g+rx,o=rx"

  recurse: yes

  state: directory

- copy:

  src: tomcat.service

  dest: /etc/systemd/system/

  mode: 0755

- systemd:

  name: tomcat1

  state: started

  enabled: true

  daemon_reload: true
```

The playbook named “main.yml”, which involves all the roles been created in one place to achieve the overall desired outcome i.e., Installing a complete webapp as follows:

```
---

- name: Install and configure MySQL

  import_tasks: mysql.yml

- name: Install and configure nginx

  import_tasks: nginx.yml

- name: Install and configure tomcat

  import_tasks: tomcat.yml
```

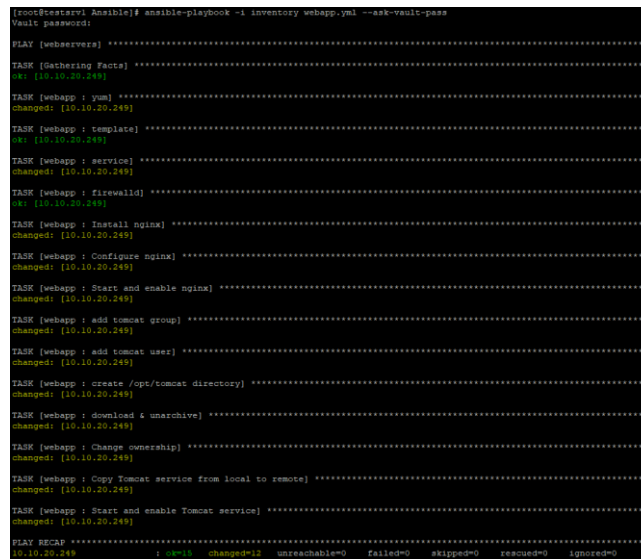

In addition, the “webapp.yml” playbook is used to achieve the tasks required by including the roles as follows:

```
---
- hosts: webservers
  become: true
  roles:
    - webapp
```

The playbook is then run using the following command:

```
ansible-playbook -i inventory main.yml --ask-vault-pass
```

The parameter ask-vault-pass is the password used to encrypt the secret file. Figure 3 depicts the success of running the playbook.



```
[root@ostara:~]# ansible-playbook -i inventory webapp.yml --ask-vault-pass
Vault password:
PLAY [webservers] *****
TASK [Gathering Facts] *****
ok: [10.10.20.249]
TASK [webapp : yum] *****
changed: [10.10.20.249]
TASK [webapp : template] *****
ok: [10.10.20.249]
TASK [webapp : service] *****
changed: [10.10.20.249]
TASK [webapp : firewall] *****
ok: [10.10.20.249]
TASK [webapp : Install nginx] *****
changed: [10.10.20.249]
TASK [webapp : Configure nginx] *****
changed: [10.10.20.249]
TASK [webapp : Start and enable nginx] *****
changed: [10.10.20.249]
TASK [webapp : add tomcat group] *****
changed: [10.10.20.249]
TASK [webapp : add tomcat user] *****
changed: [10.10.20.249]
TASK [webapp : create /opt/tomcat directory] *****
changed: [10.10.20.249]
TASK [webapp : download & unarchive] *****
changed: [10.10.20.249]
TASK [webapp : Change ownership] *****
changed: [10.10.20.249]
TASK [webapp : Copy Tomcat service from local to remote] *****
changed: [10.10.20.249]
TASK [webapp : Start and enable Tomcat service] *****
changed: [10.10.20.249]
PLAY RECAP *****
10.10.20.249 : ok=15 changed=12 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
```

Figure 3. Playbook running success

5. Discussion

The process of repetitive tasks automation plays a pivotal role in facilitating work pace and guarantee consistency across your IT environment. In addition to the time, it saves to let IT professionals focus in other tasks. Kaushik, S., & Gupta, S. [21] addressed the suffering of doing manual deployment, especially for more than one server that requires running commands one by one and the possibility of getting errors which will lead to inefficiency and not meeting deadlines. Two test cases were performed to assess the time difference between adopting manual deployment or while using Ansible. Case 1 adopted configuring a single node of OpenStack Cloud on a dedicated hardware and Case 2 was configuring multiple node setup of OpenStack Cloud. Ansible reduced the time required from about 4 hours (using manual deployment) to about 2 hours (for the deployment using Ansible) for Case 1, while reduced the time from about 7 hours (using manual deployment) to about 2.7 hours (for the deployment using Ansible).

In this paper, Ansible was used to automate the installation and configuration of a complete web application composed of MySQL, Nginx and Tomcat.

Normally, the process is time consuming and needs focus as not to forget a step. But the most catastrophic part is when you have multiple servers that you should install and configure the web app in it as well. For IT professionals, that’s a nightmare.

Ansible proved to be an efficient automation tool, as it relies on YAML playbooks, which are simple to write. Also, the utilization of the modules made it easy to achieve tasks required. But the most impressive benefit was the time it took to run the playbook and get all services running.

The average of time required to get the web server running was estimated to be 45 minutes per server. But when using Ansible, it diminished to an average of 8 minutes. Taking into consideration the advantage of getting things running on multiple servers simultaneously.

The time utilization illustrated in Figure 4 demonstrates the high capabilities provided by Ansible.

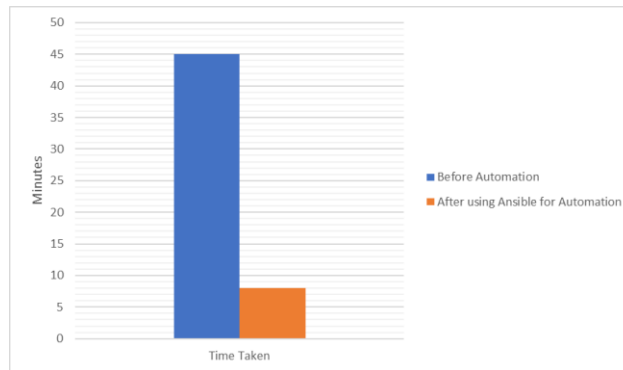


Figure 4. Time utilization before and after automation

The percentage of time taken showed in Figure 5, clearly proves that Ansible reduced the time required to run the web app by 70%, which is a massive in terms of time utilization.

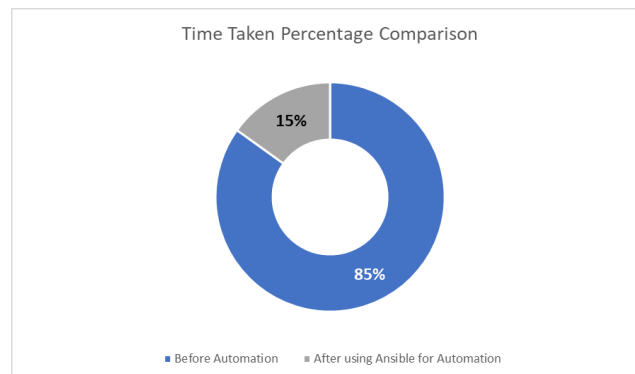


Figure 5. Time taken Percentage

6. Conclusion

In this paper, Ansible was adopted to provide a smooth and fast way to install and configure a web application. The ease been provided by Ansible modules and the simplicity of YAML made it easy to write the playbooks. Also, roles were utilized to break down the code and allow for simple tracking and troubleshooting processes. Not forgetting to mention that although YAML is simple, it was so important to watch out for indentation. As if you miss a single space, the playbook will throw an error and won't be able to run. The good news is that Ansible will notify you of the exact place of the error.

The results show that the Ansible is very powerful and full of features that makes it a reliable tool to be used for automating miscellaneous tasks in a timely-manner which is crucial factor in managing IT infrastructure.

It has been found that Ansible decreased the amount of time taken to get all the required services running by an approximately 5.6 times than when no automation was involved.

It is recommended to conduct further studies to highlight some other use cases where ansible is applicable and can facilitate the workflow for IT professionals.

References

- [1] Ahmad N. et al. (2020) Cloud Computing Trends and Cloud Migration Tuple. In: Saini H.S., Singh R.K., Tariq Beg M., Sahambi J.S. (eds) Innovations in Electronics and Communication Engineering (2020): 737-745. Lecture Notes in Networks and Systems, vol 107. Springer, Singapore
- [2] Knight, Simon; Rienties, Bart; Littleton, Karen; Tempelaar, Dirk; Mitsui, Matthew and Shirag, Chirag (2017). The Orchestration of a Collaborative Information Seeking Learning Task. Information Retrieval Journal, 20(5) pp.480–505.
- [3] Automation vs. Orchestration: Which One to Choose in 2023? <https://research.aimultiple.com/orchestration-vs-automation/>. Accessed 6 Apr. 2023.
- [4] McKinsey Global Institute. (2017a). "A Future That Works: Automation, Employment and Productivity".
- [5] Praveen Kumar Kollu, Kailash Kumar, Pravin R. Kshirsagar, Saiful Islam, Quadri Noorulhasan Naveed, Mohammad Rashid Hussain, Venkatesa Prabhu Sundramurthy, "Development of Advanced Artificial Intelligence and IoT Automation in the Crisis of COVID-19 Detection", Journal of Healthcare Engineering, vol. 2022, Article ID 1987917, 12 pages, 2022. <https://doi.org/10.1155/2022/1987917>.
- [6] Weidong Gai, Yue Gu, Jiaming Qin, "Financial Automation Audit Method Based on Blockchain Technology", Computational Intelligence and Neuroscience, vol. 2022, Article ID 9941585, 12 pages, 2022. <https://doi.org/10.1155/2022/9941585>.
- [7] Pillai, U. (2022). Automation, productivity, and innovation in information technology. Macroeconomic Dynamics, 1-27. doi:10.1017/S1365100521000699.

- [8] Wurster, M., Breitenbücher, U., Brogi, A., Diez, F., Leymann, F., Soldani, J. and Wild, K., 2021. Automating the Deployment of Distributed Applications by Combining Multiple Deployment Technologies. Proceedings of the 11th International Conference on Cloud Computing and Services Science.
- [9] Hintsch, J., Göring, C., & Turowski, K. (2016). A Review of the Literature on Configuration Management Tools. International Conference on Information Resources Management.
- [10] Zeng, S., Adam, C., Wu, F., Guo, S., Ruan, Y., Venugopal, C., & Puri, R. (2014, May). Managing risk in multi-node automation of endpoint management. 2014 IEEE Network Operations and Management Symposium (NOMS).
- [11] Overview of Puppet’s Architecture. <https://www.puppet.com/docs/puppet/5.5/architecture.html>. Accessed 4 Apr. 2023.
- [12] Chef Infra Overview. https://docs.chef.io/chef_overview/. Accessed 4 Apr. 2023.
- [13] Salt System Architecture. https://docs.saltproject.io/en/3004/topics/salt_system_architecture.html. Accessed 4 Apr. 2023.
- [14] Howard, Michael. Terraform -- Automating Infrastructure as a Service. 2022. DOI.org (Datacite), <https://doi.org/10.48550/ARXIV.2205.10676>.
- [15] Ansible Architecture — Ansible Documentation. https://docs.ansible.com/ansible/latest/dev_guide/overview_architecture.html. Accessed 4 Apr. 2023.
- [16] Ramandeep Singh, Dr. Ravindra Kumar Purwar, 2019, Cloud Automation with Configuration Management using CHEF Tool, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 08, Issue 04 (April – 2019).
- [17] Kodali, Sailesh, "Automation of Production Servers using DevOps (Puppet) Tools" (2016). Culminating Projects in Mechanical and Manufacturing Engineering. 64.
- [18] Kumaran, N., et al. "Chef Automation on Google Cloud." International Journal for Research in Applied Science and Engineering Technology, vol. 10, no. 3, Mar. 2022, pp. 1491–98. DOI.org (Crossref), <https://doi.org/10.22214/ijraset.2022.40909>.
- [19] Howard, Michael. Terraform -- Automating Infrastructure as a Service. 2022. DOI.org (Datacite), <https://doi.org/10.48550/ARXIV.2205.10676>.
- [20] S, Likitha. "Automation of Server Configuration Using Ansible." International Journal for Research in Applied Science and Engineering Technology, vol. 10, no. 6, June 2022, pp. 4109–13. DOI.org (Crossref), <https://doi.org/10.22214/ijraset.2022.44840>.
- [21] Kaushik, S., & Gupta, S. (2017). Implementation of Open Stack Through Ansible. International Journal of Engineering and Advanced Technology (IJEAT).