Scale Cloud Services using Orchestration

Submitted for the course:

NETV 379 - Cloud Computing

November 2024

Ryufath Alief Adhyaksa Putera Soepeno (23792097)

Information Systems

Faculty of Engineering & Technology

Sampoerna University

&

Applied Computing

College of Applied Science & Technology

University of Arizona

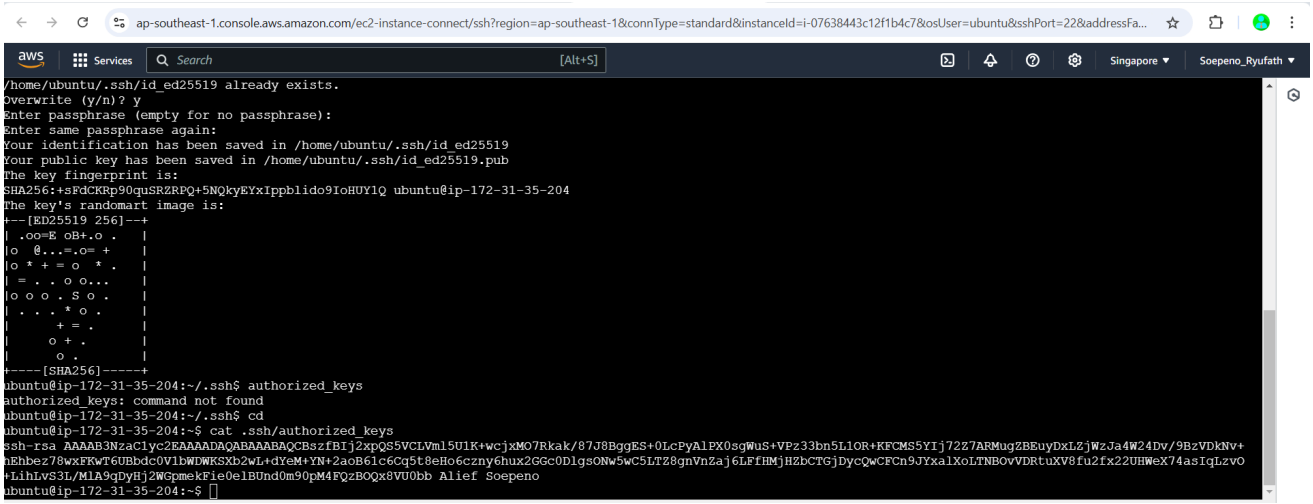
1. Introduction

This project aims to utilize automation and orchestration tools to enhance the scalability and efficiency of web services in a cloud environment. Specifically, it involves configuring a pre-existing AWS instance to support Ansible-based configuration management, enabling streamlined and consistent setup across multiple cloud resources. Through this approach, the project aims to demonstrate how automation can simplify large-scale web service management, reduce manual tasks, and promote high-availability deployments.

2. Discussion

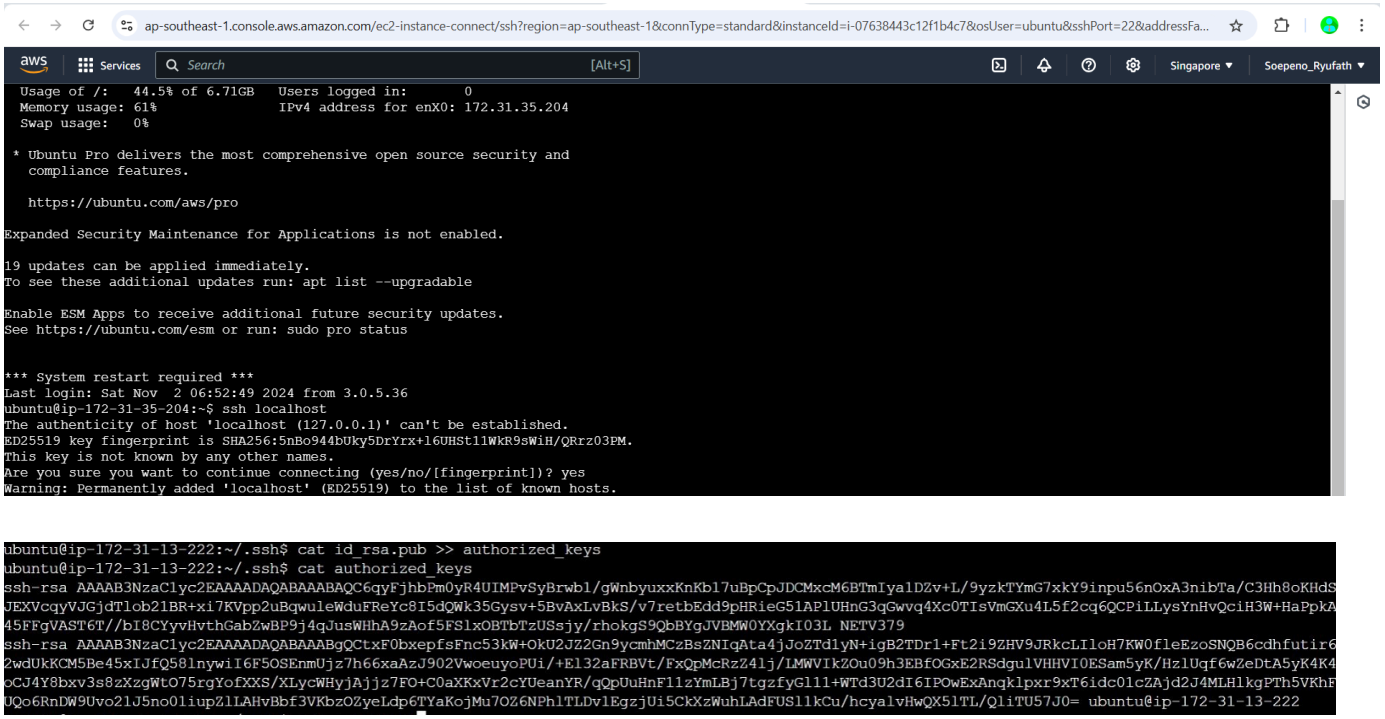
Part 1: Initial Actions

To begin, connect to the AWS instance created in Lab 1, running on Ubuntu 24.04 LTS. The writer generated a key pair named "*alief soepeno*" and add the required details, then select the instance to connect and run it. After connecting to the instance's Linux terminal, update the system and check if the Linux version is up to date. The instance's output will display after these initial steps.



Next, generate an RSA key pair and append the public key to `*authorized_keys*` without adding arguments for simplicity. Return to the home directory and verify that the public key has

been correctly added to `*authorized_keys*`. Finally, connect to localhost via SSH to complete the setup.



Part II: Ansible

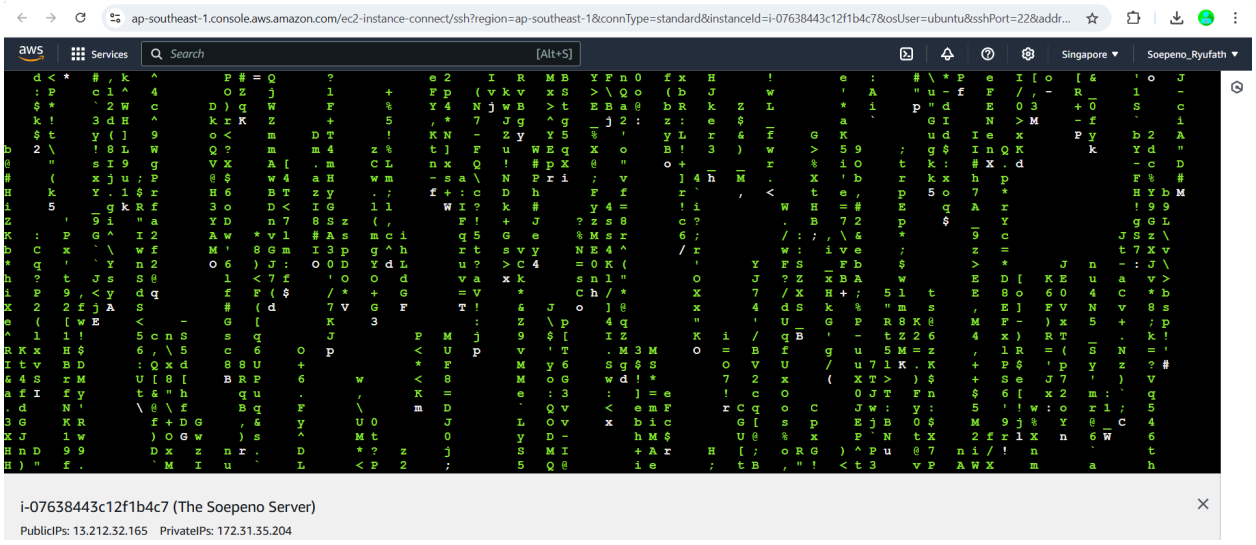
First, start by updating the instance and its properties using the `*sudo apt-get update*` command. Then, add the Ansible PPA to allow for easy installation of Ansible and its components. Next, create a custom Ansible inventory file by opening the SSH window and using `*sudo nano /etc/ansible/hosts*`. Add the server name in a structured format, replacing any spaces with underscores, and write it in the format "Server_name Localhost" for clarity.

To check if Ansible is set up correctly, try running a basic command, such as `*ansible all -m ping*`, which will verify connectivity. Also, install the `*cowsay*` package to test Ansible's functionality further.

```
ubuntu@ip-172-31-13-222:~$ cowsay 'Hello World'
 _____________
< Hello World >
 -------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
ubuntu@ip-172-31-13-222:~$
```

```
ubuntu@ip-172-31-13-222:~$ fortune | cowsay
 ________________________________________
/ Q: What do you call a WASP who doesn't \
| work for his father, isn't a           |
|                                        |
| lawyer, and believes in social causes? |
\ A: A failure.                          /
 ----------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Then, configure the `*cmatrix.yml*` file within Ansible by setting up the configurations needed for later steps. Once configurations are complete, execute the `*.yml*` file using `*sudo*` to apply the settings.



After configuring and running `*cmatrix.yml*`, execute `*cmatrix*` to see it in action. Now, add the public IP of the instance to the Ansible configuration, allowing for direct connection. Run the IP with a shell `*sudo*` command, and when prompted, ensure the connection continues without requiring additional fingerprint verification.

```
ubuntu@ip-172-31-35-204:~$ ansible The_Soepeno_Server -m shell -a 'free | grep Mem:'
localhost | CHANGED | rc=0 >>
Mem:           989392      644264       78604        3652      266524      182928
54.200.43.172 | CHANGED | rc=0 >>
Mem:           989392      644896       80384        3652      264112      182436
```

Finally, to monitor memory and disk usage, run an Ansible command to view a summary of memory output, enable a query in Ansible to check free and used disk space specifically within the home folder.

```
ubuntu@ip-172-31-35-204:~$ df -hT /home
Filesystem     Type  Size  Used Avail Use% Mounted on
/dev/root      ext4  6.8G  3.5G  3.3G  52% /
ubuntu@ip-172-31-35-204:~$
```

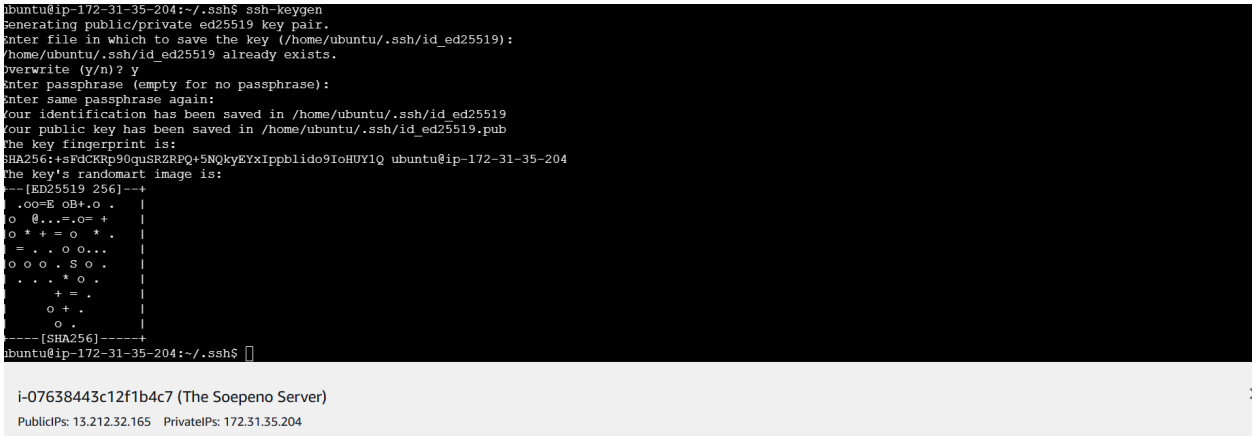
i-07638443c12f1b4c7 (The Soepeno Server)
PublicIPs: 13.212.32.165 PrivateIPs: 172.31.35.204

3. Observation

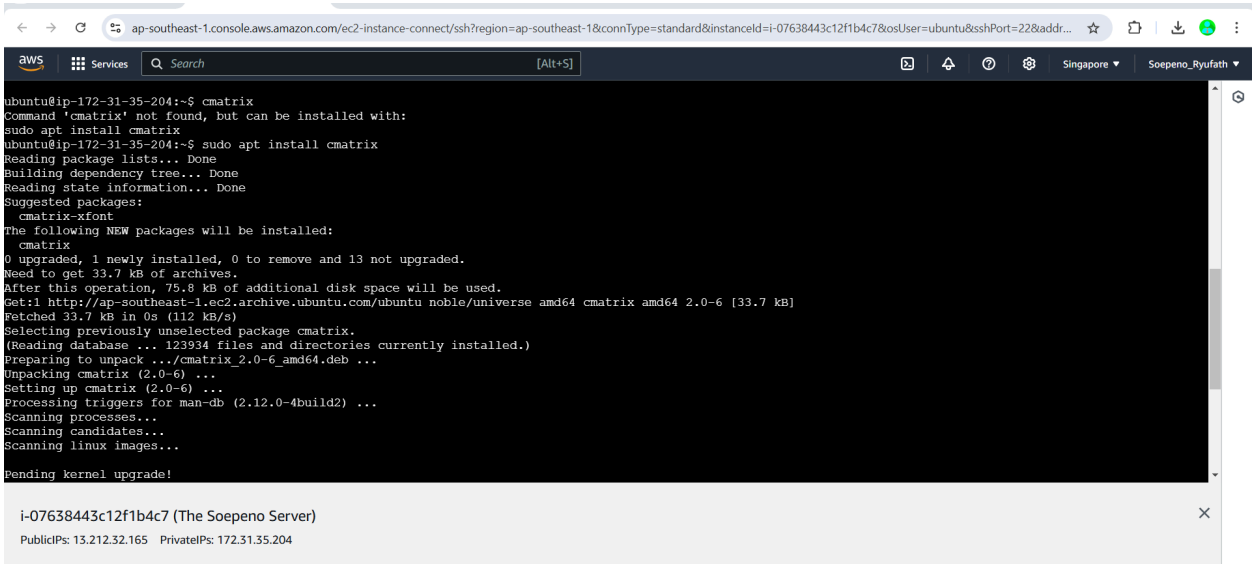
This lab involves deploying and managing an AWS instance with a focus on configuration automation via Ansible, highlighting several structured key aspects of cloud resource management, security, and automation practices. This demonstrates the benefits of automating cloud infrastructure, from updating instances to verifying configurations. Using commands like `*sudo apt-get update*` and defining configurations in YAML enables CI/CD pipelines, crucial for agile and DevOps practices. In Ansible and AWS contexts, these practices reduce downtime and ensure system consistency across environments.

During the lab study, there were also a few errors that I faced and noticed on my behalf and I figured out solutions to overcome those errors. At first, I spent hours on the ansible inventory file, finding out what was wrong. Then I remembered that since server files are also named in file form–those used in networking as well–and all spaced names and variables should be written with an underscore, so I did.

Then, there was an error on my behalf, so I overwritten the key pair again, this is the result



Another one was in my AWS instance; it seemed that I was not able to run cmatrix immediately, therefore, `*cmatrix*` had to be installed to the instance beforehand.



4. Conclusion

This lab provided foundational knowledge in automating and scaling cloud environments by configuring connectivity, inventory, and YAML-based settings for streamlined management. Through hands-on practice, I learned how automating updates, verifying configurations, and managing resources can improve scalability, consistency, and reduce downtime.