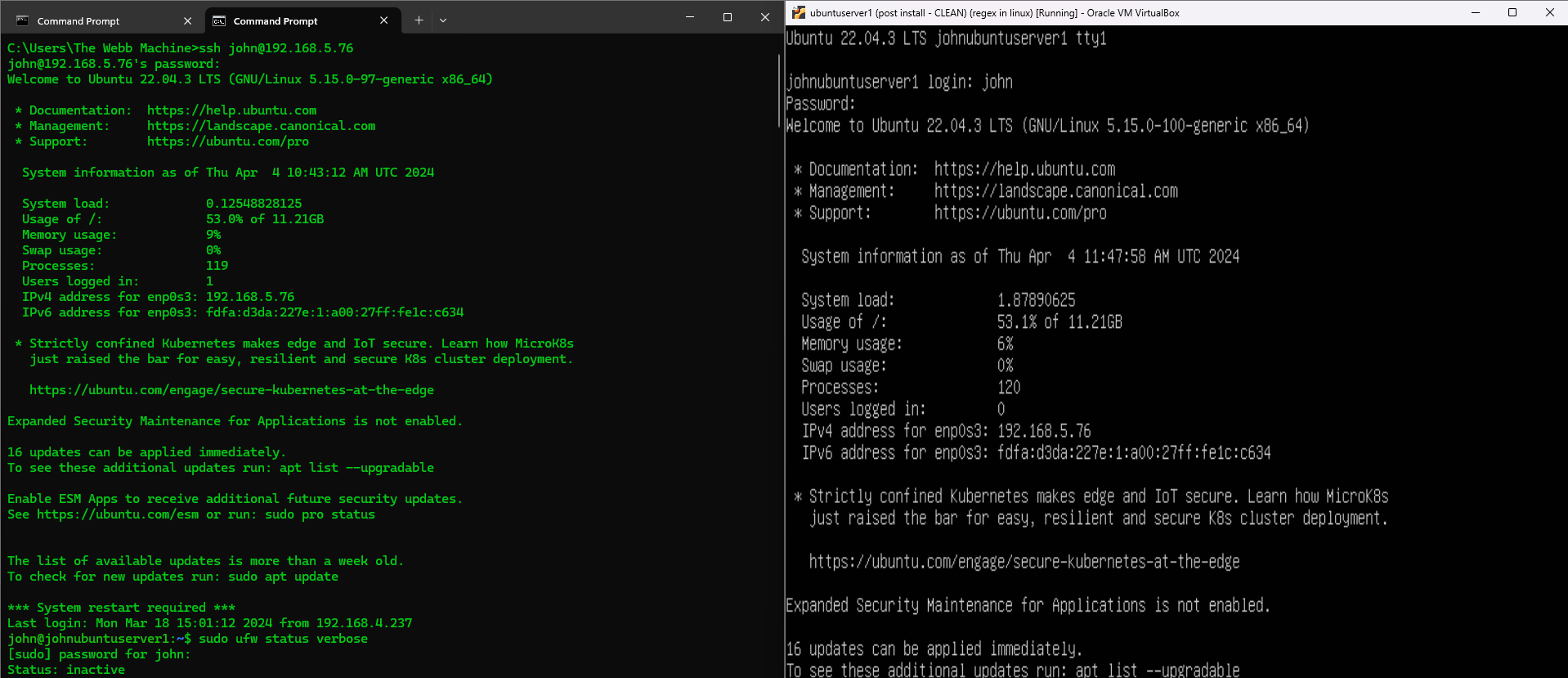
**UFW Configuration Testing on Ubuntu Server Lab**

**Task 1: Enabling UFW and Observing Results**

Booted up Ubuntu 22.04 Server and logged into the console.

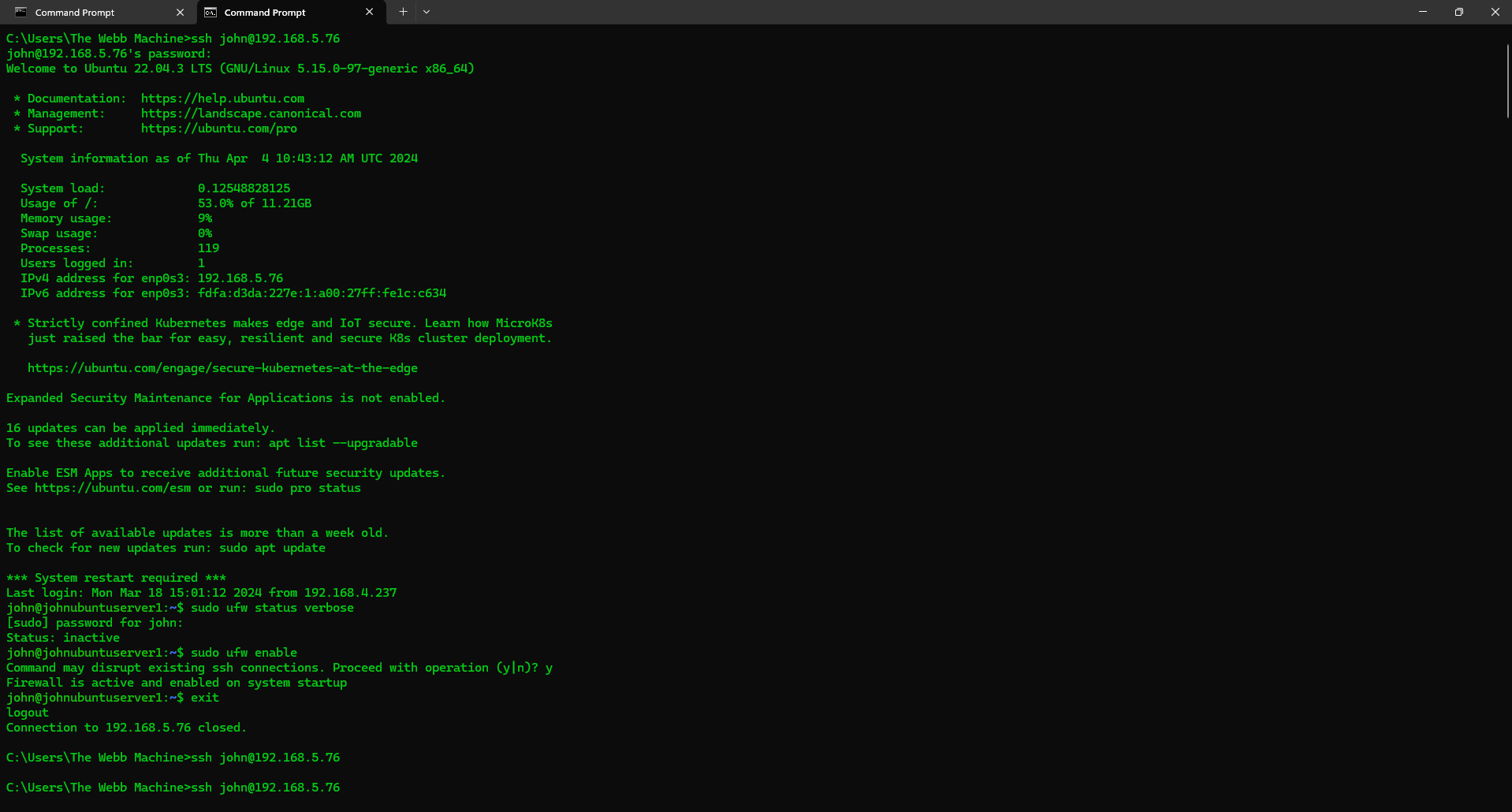
Obtained the IP address of the Linux server.

Opened Windows Command Prompt on the Host OS and initiated an SSH connection to the Linux server.

Within the Linux server via SSH, executed **sudo ufw status verbose** and observed that the status was inactive.

Enabled the UFW Firewall using the **sudo ufw enable** command: Prompt received: "Command may disrupt existing SSH connections. Proceed with operation (y|n)?"

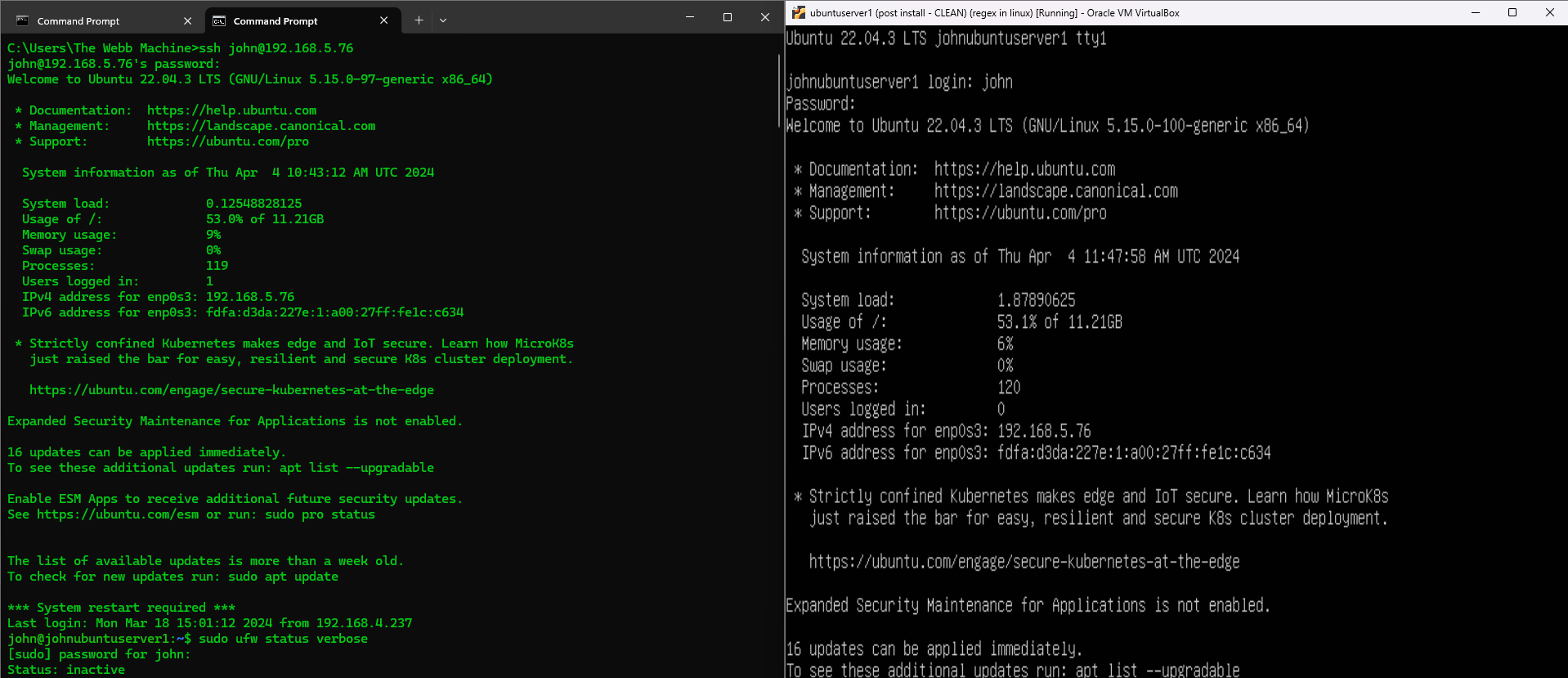
Exited from the SSH session using **exit** command.



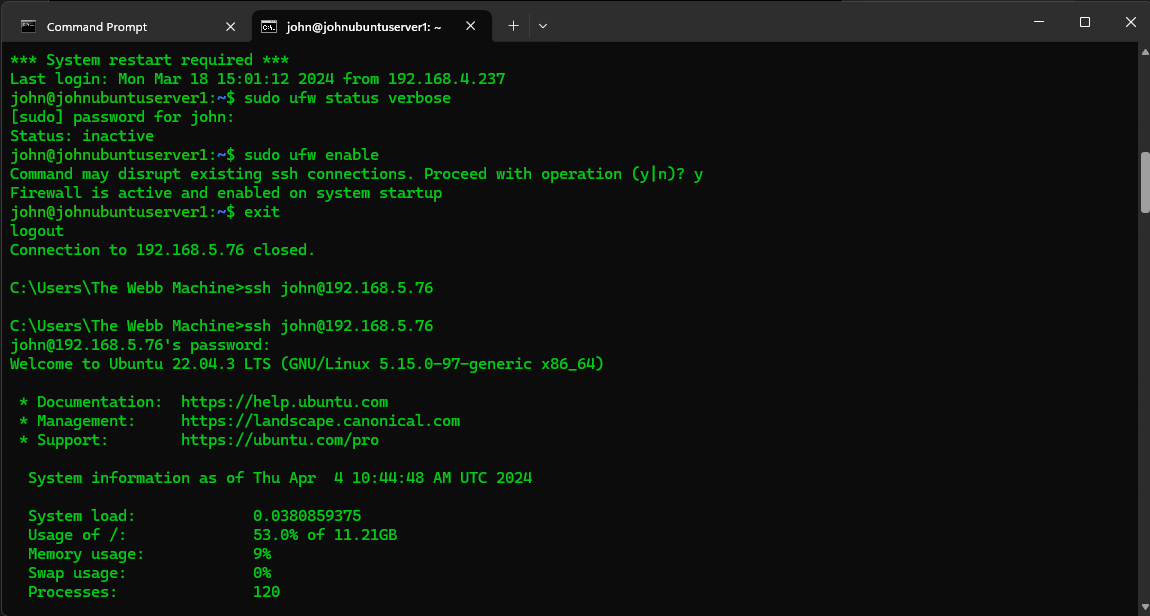
Attempted to reconnect via SSH, using the **ssh** [**john@192.168.5.76**](mailto:john@192.168.5.76) command but failed. The UFW Firewall was blocking access as no rules were set up allowing SSH traffic.

**Task 2: Disabling UFW and Setting Up SSH Rule**

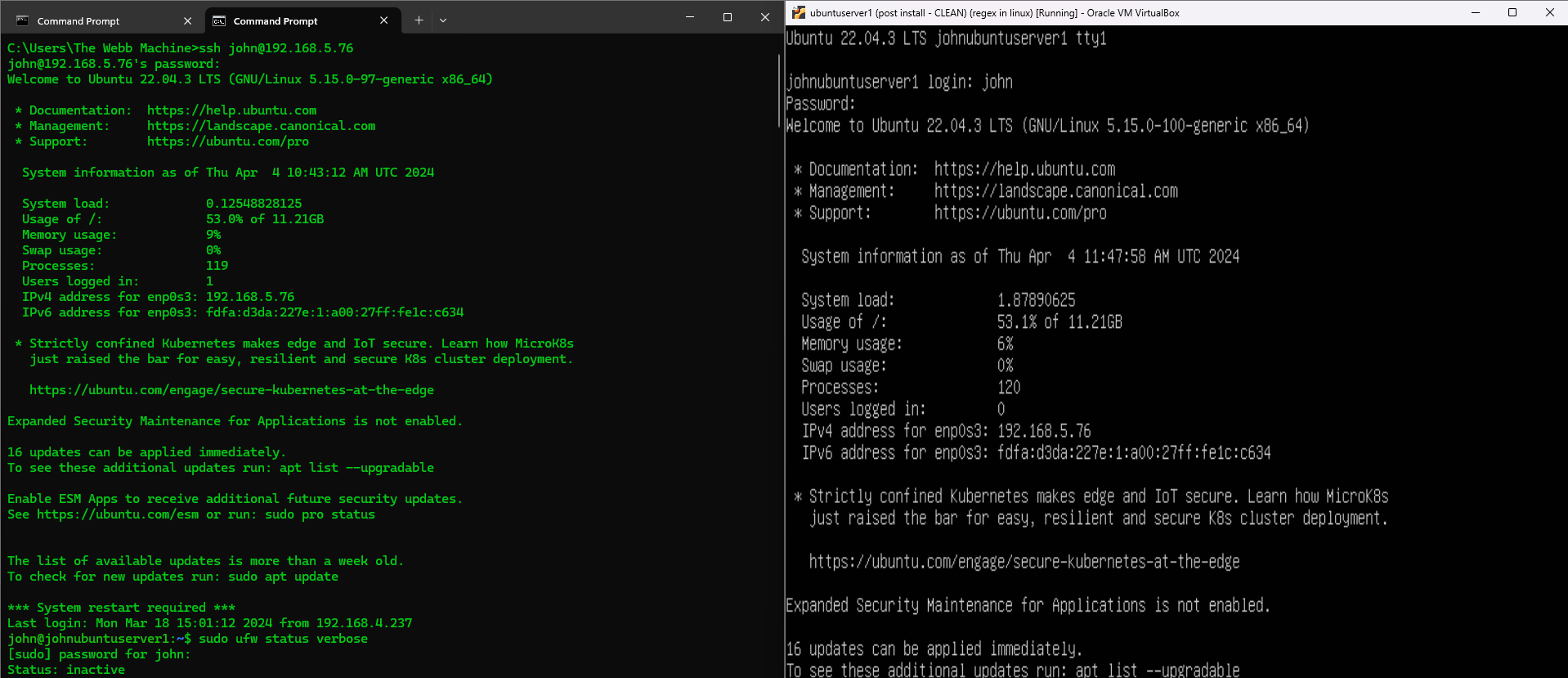
Modified UFW to allow SSH traffic using command **sudo ufw allow ssh**.



Enabled the UFW Firewall. using the **sudo ufw enable** command: Prompt received: "Command may disrupt existing SSH connections. Proceed with operation (y|n)?"

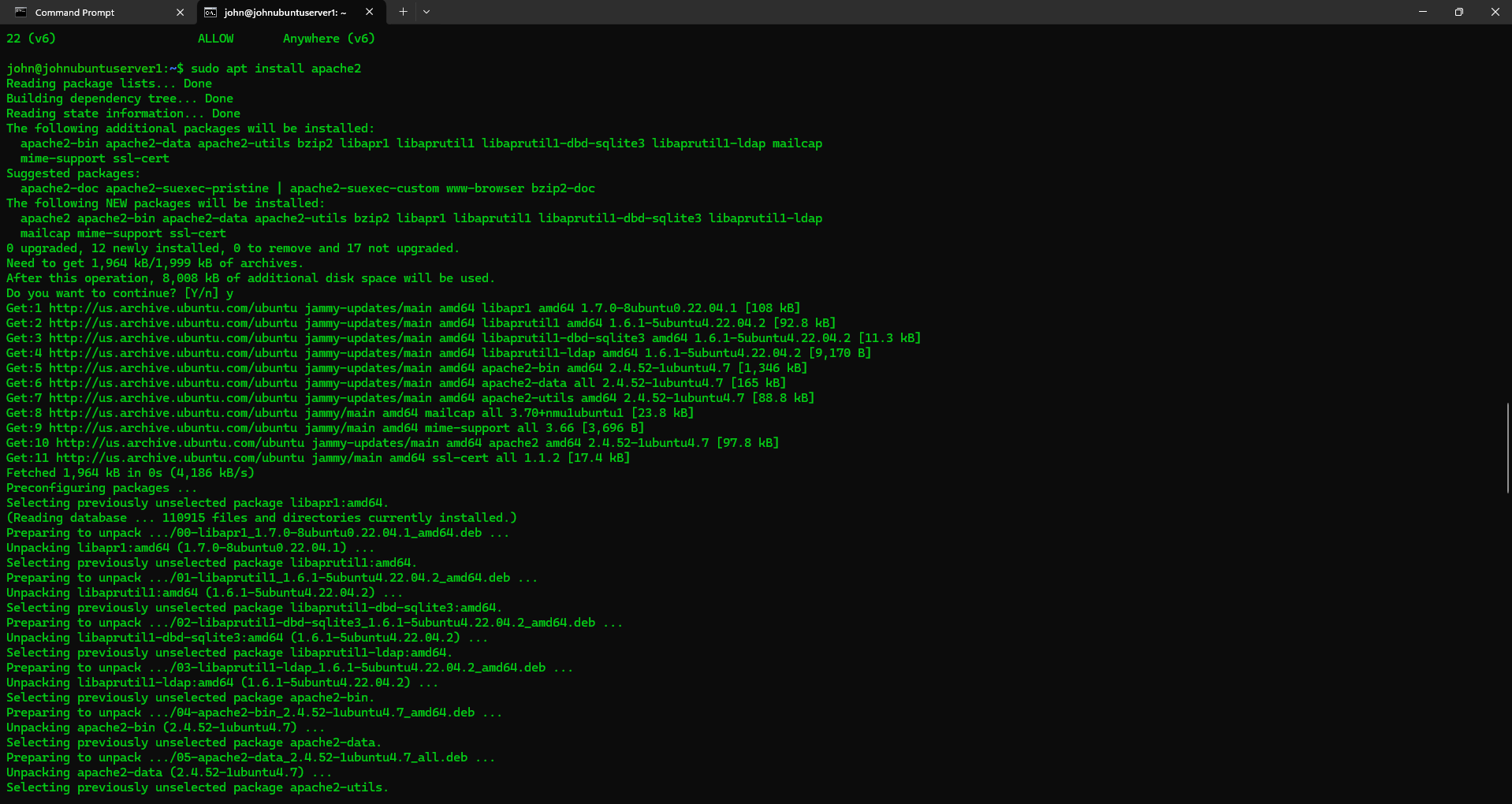
Exited from the SSH session.

Attempted to reconnect via **ssh** [**john@192.168.5.76**](mailto:john@192.168.5.76) command and success. The UFW Firewall allowed access based on the newly set up SSH rule.

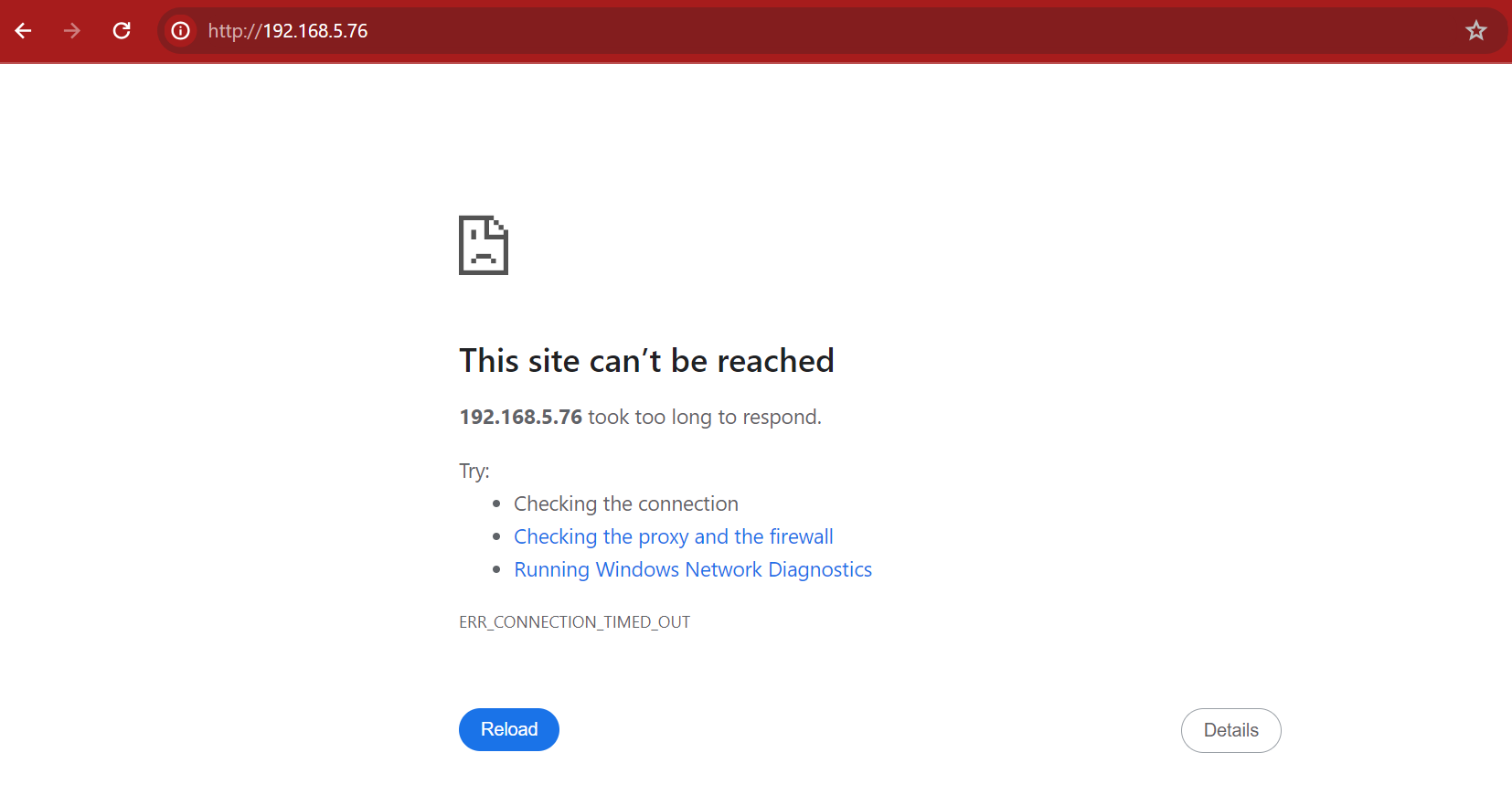


**Task 3: Testing Apache2 Installation and UFW Effectiveness**

Ensured that Apache2 was installed on Ubuntu 22.04 Server using the **sudo apt install apache2** command.

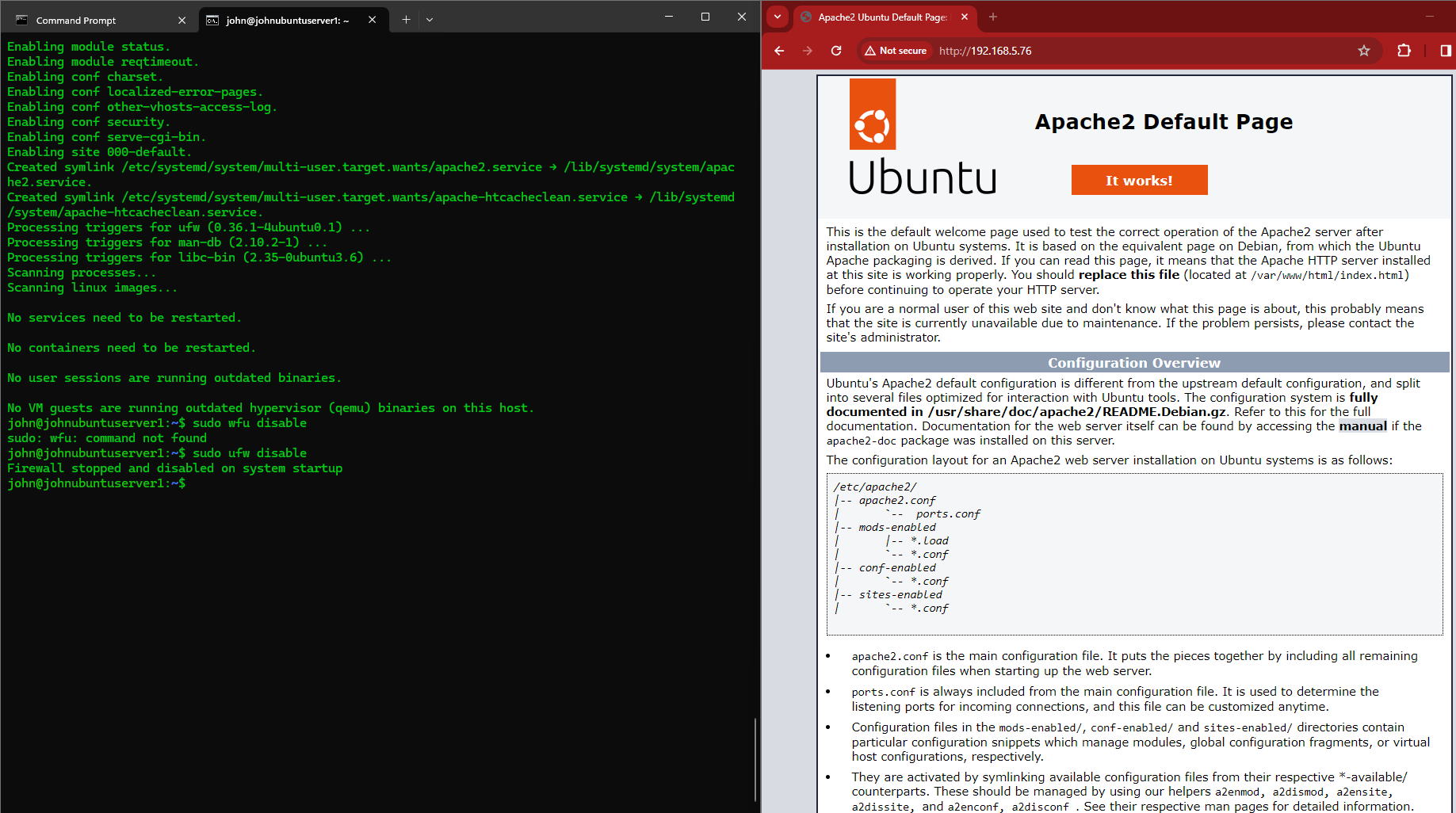


From the Windows Host, attempted to browse to the IP address of the Linux server. Failed to render the webpage due to UFW blocking access.

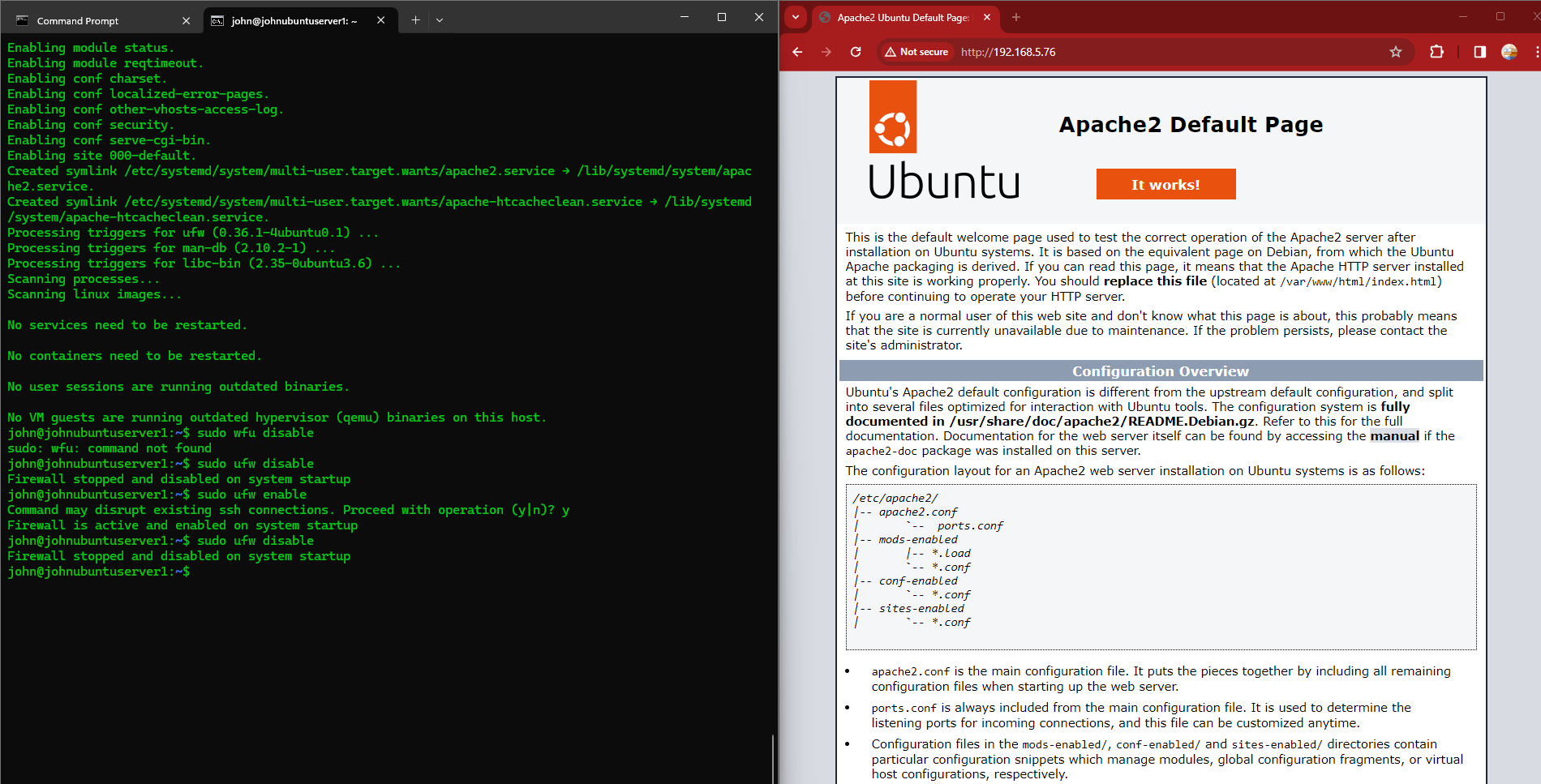


Disabled the UFW Firewall using the **sudo ufw disable** command.

Refreshed the browser and immediately rendered the default web page, indicating the UFW control was indeed disabled.

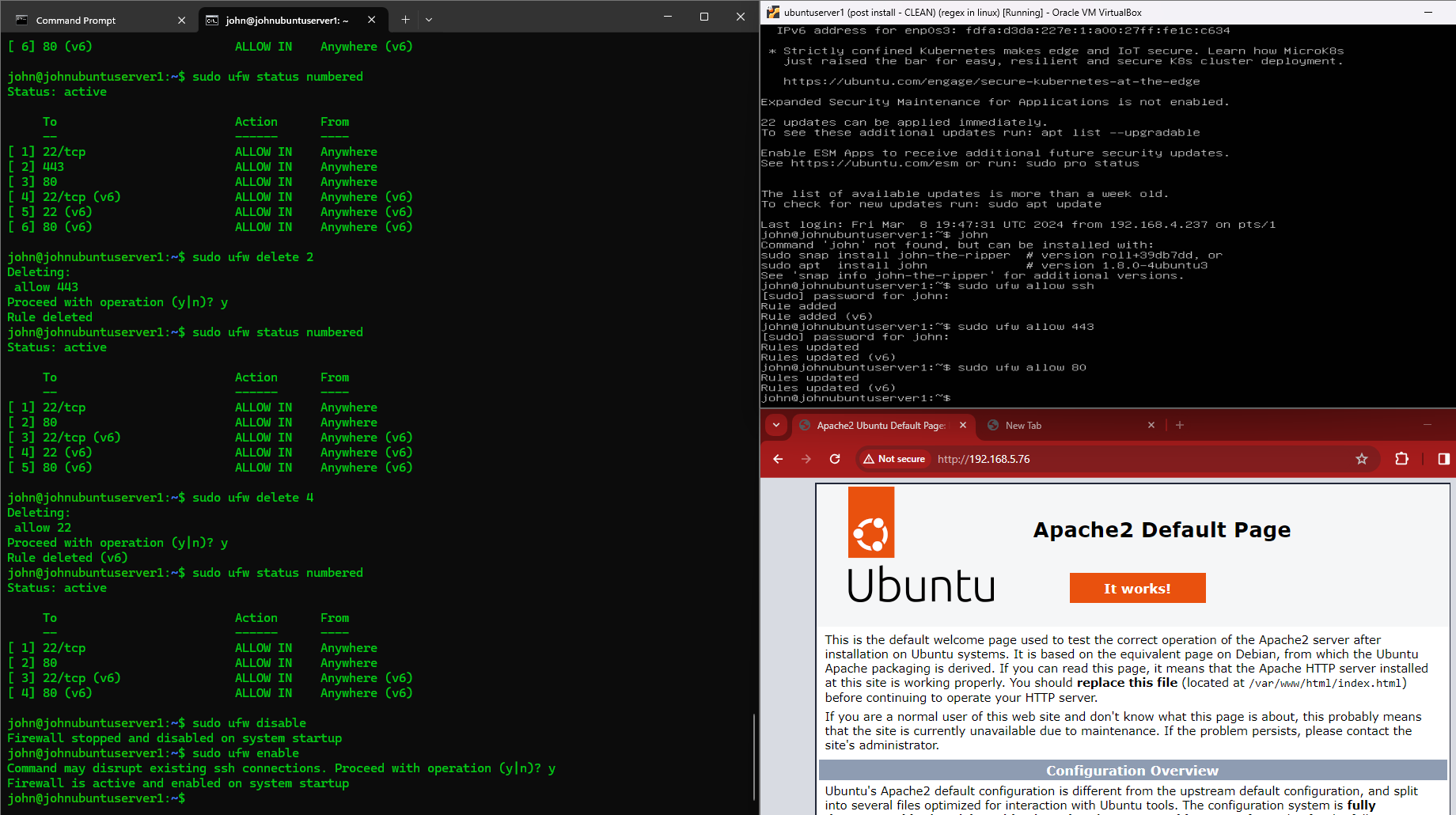


Toggled the UFW firewall on and off to confirm its control mechanism using both the **sudo ufw enable** and **sudo ufw disable** commands.



**Task 4: Creating Rule to Allow HTTP Traffic**

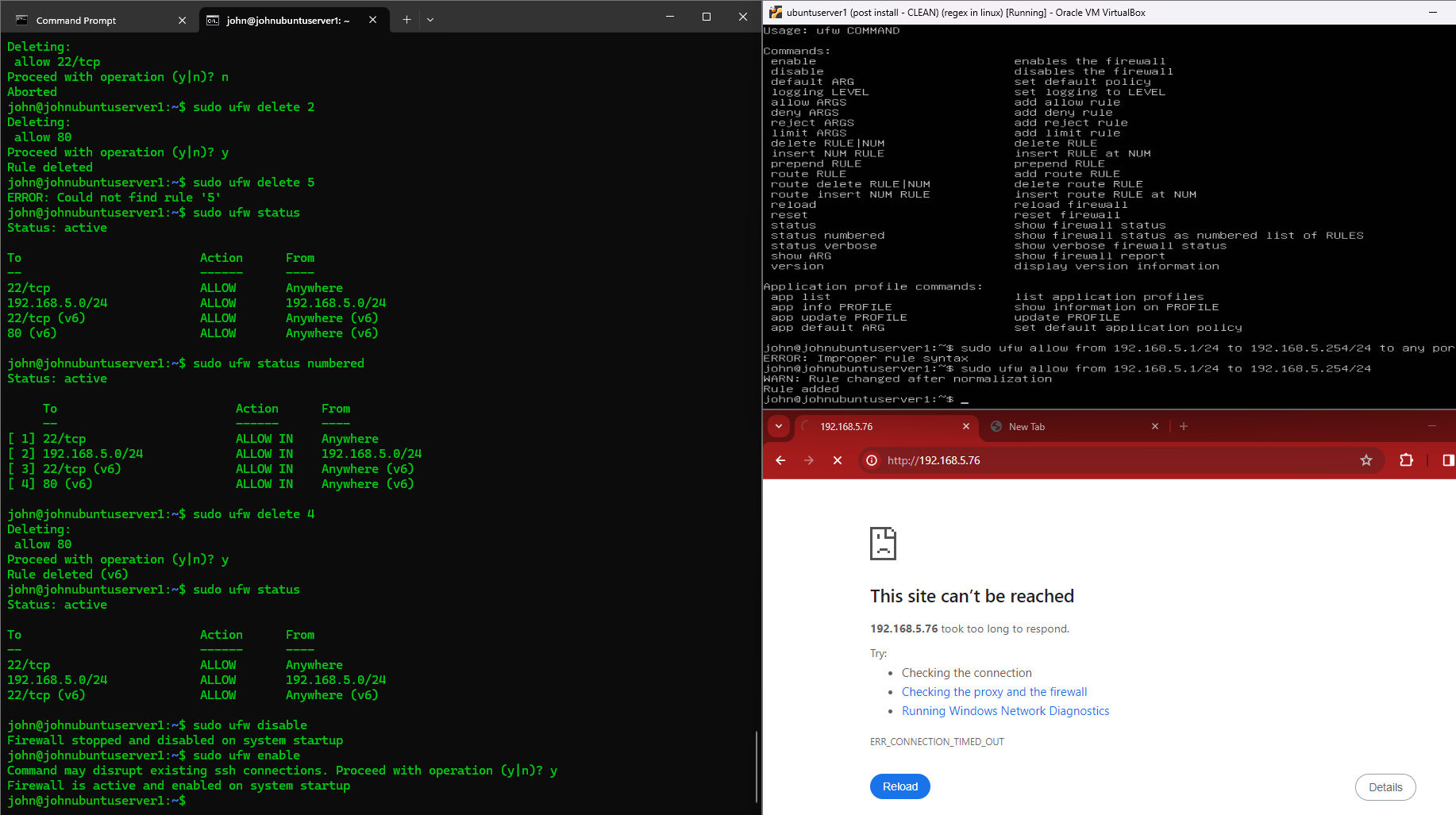
Created a specific rule on Ubuntu 22.04 Server to allow HTTP traffic using the **sudo ufw allow 80** command.



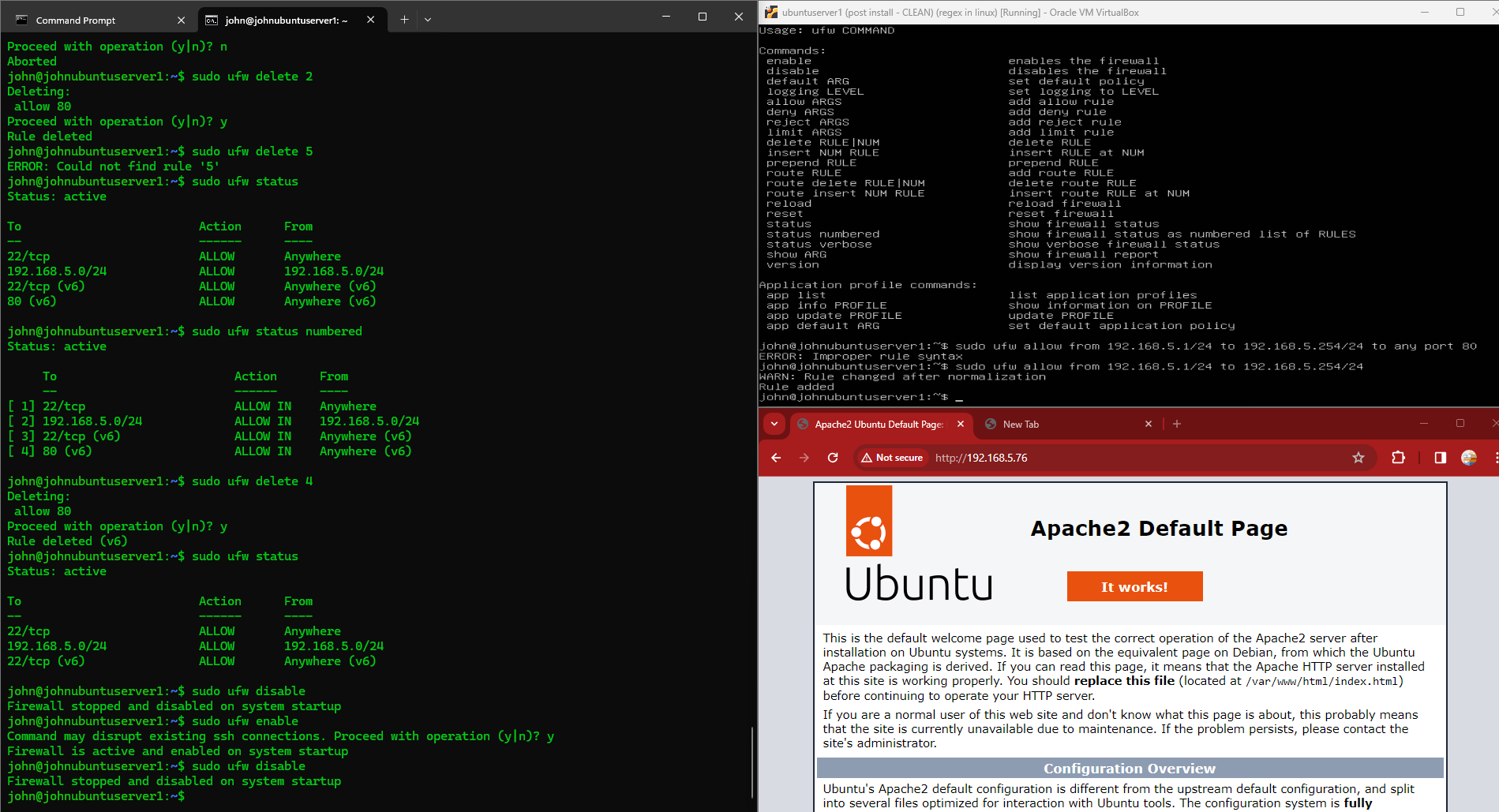
Validated the rule's effectiveness by accessing the default page of the Apache Web Server from the Windows Host.

**Task 5: Creating Rule to Allow HTTP Traffic from a Range of Addresses**

Created a specific rule on Ubuntu 22.04 Server to allow HTTP traffic only from a range of addresses using CIDR notation: **sudo ufw allow from 192.168.5.1/24 to 192.168.5.254/24**



Validated the rule's effectiveness by accessing the default page of the Apache Web Server from the Windows Host.



Displayed UFW configuration by using the **sudo ufw status** command.

**Conclusion**

In this lab, we focused on configuring the Uncomplicated Firewall (UFW) on an Ubuntu 22.04 Server to control access to services such as SSH and HTTP. Through a series of tasks, we gained hands-on experience with enabling and disabling UFW, creating specific rules to allow traffic, and observing the firewall's impact on network connectivity.

**Key Highlights:**

1. **Understanding UFW Status:** We began by understanding the initial status of UFW, which was inactive. This highlighted the importance of enabling UFW to enforce firewall rules.
2. **Enabling SSH Rule:** By configuring UFW to allow SSH traffic, we ensured continued access to the server. This task emphasized the necessity of setting up rules to permit essential services while maintaining security.
3. **Testing UFW Effectiveness:** The lab provided opportunities to test UFW's effectiveness in controlling access. We observed how enabling or disabling UFW impacted connectivity and service availability, showcasing the firewall's role in network security.
4. **Rule Creation and Validation:** Tasks involving the creation of specific rules for HTTP traffic demonstrated the flexibility of UFW in defining access controls. Validation of these rules through successful access to web services affirmed their correctness.
5. **CIDR Notation Usage:** The task of allowing HTTP traffic from a range of addresses using CIDR notation highlighted best practices in specifying IP address ranges for firewall rules, enhancing scalability and manageability.
6. **Documentation and Review:** Throughout the lab, documenting configuration changes and observed results was essential. It emphasized the importance of thorough documentation for future reference and review.

Overall, this lab provided valuable hands-on experience in configuring and managing UFW on Ubuntu 22.04 Server, reinforcing fundamental concepts in network security and access control. By understanding how to effectively use UFW, we can enhance the security posture of our systems while ensuring essential services remain accessible.