Ansible Firewall Hardening and Validation

This project demonstrates the use of Ansible to enforce firewall hardening on a Linux system and validate the resulting security posture using firewalld. The automation focuses on reducin attack surface by disabling unnecessary network exposure while maintaining controlled administrative access.

The playbook applies firewall changes in a repeatable and idempotent manner, ensuring that commonly exposed web ports are closed and that all changes are enforced immediately and permanently. After execution, the system's firewall state is verified directly on the host to confirm that the desired configuration is active.

Automated Firewall Enforcement

The automation performs the following actions:

Disables inbound traffic on TCP port 80 (HTTP)

Disables inbound traffic on TCP port 443 (HTTPS)

Reloads the firewalld service to apply changes without requiring a reboot

All tasks are executed with privilege escalation and are safe to re-run, reporting changes only when the system state requires modification.

Post-Deployment Verification

Firewall validation is performed using firewall-cmd --list-all, confirming that:

The public zone is active and bound to the correct network interface

HTTP and HTTPS services are no longer exposed

Only essential services remain permitted

SSH access is tightly controlled using rich rules, including explicit source-based allow and deny rules

This verification step ensures that automation results align with security expectations and operational requirements.

Purpose and Use Case

This project reflects real-world infrastructure security practices where firewall policies must be centrally managed, auditable, and consistently enforced across environments. It highlights the use of Ansible not only for configuration changes, but also as part of a broader workflow that includes validation and compliance checks.

An Ansible playbook is open in vim, written in YAML and designed to manage firewall rules on a development environment. The play is titled “Close ports 80 and 443 on dev-app”, targets the dev-app host group, and runs with elevated privileges using become: true.

The playbook defines a sequence of firewall management tasks using the ansible.posix.firewalld module:

Remove firewall rule for port 80 (HTTP)

Disables inbound traffic on TCP port 80, applies the change permanently, and enforces it immediately without requiring a reboot.

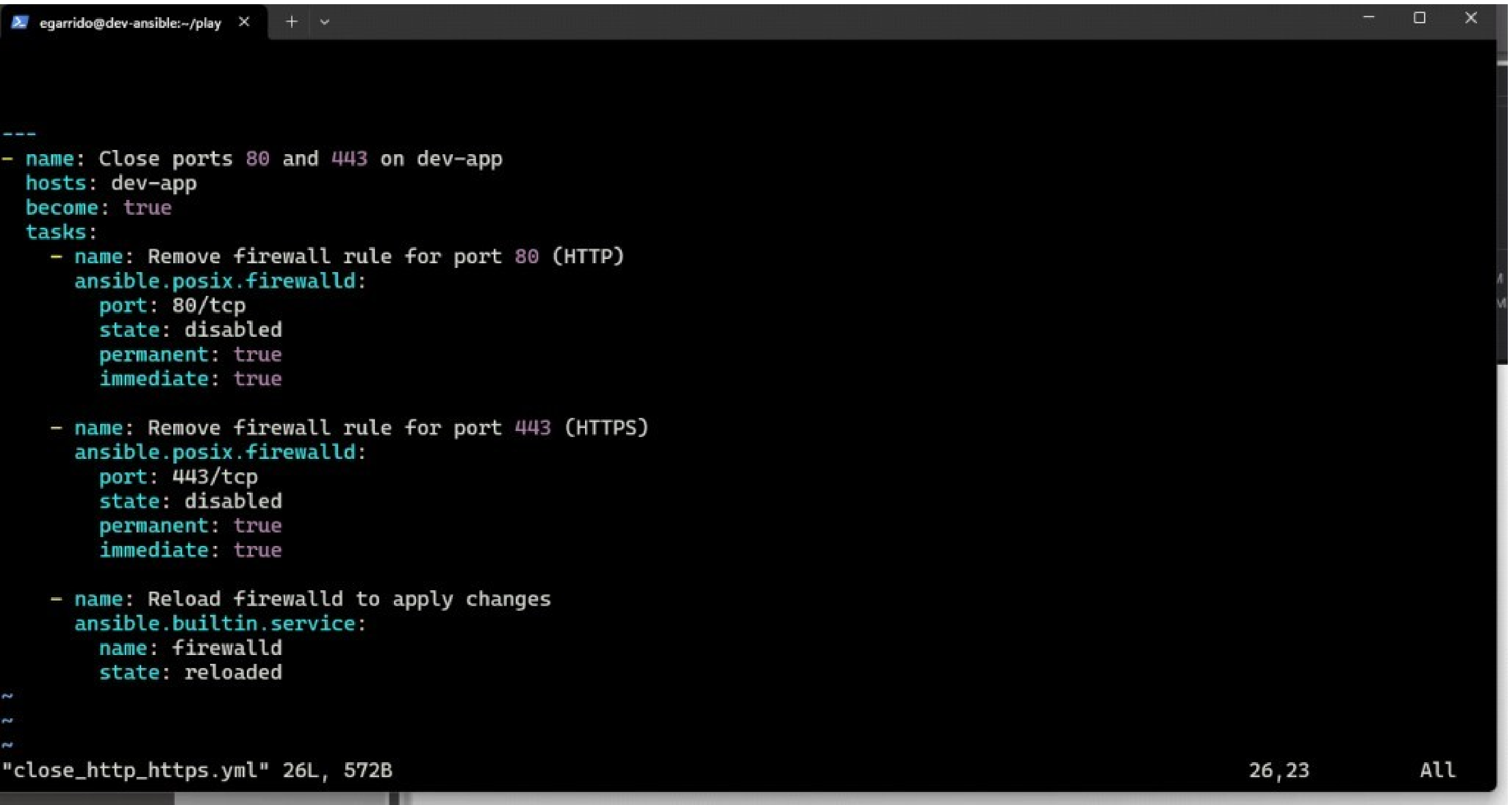
Remove firewall rule for port 443 (HTTPS)

Disables inbound traffic on TCP port 443 with the same permanent and immediate enforcement settings.

Reload firewalld to apply changes

Uses the ansible.builtin.service module to reload the firewalld service, ensuring all firewall rule changes take effect.

The file name shown at the bottom of the editor is close_http_https.yml, and the editor is in insert mode, indicating the playbook is actively being edited or finalized.



```
---
- name: Close ports 80 and 443 on dev-app
  hosts: dev-app
  become: true
  tasks:
    - name: Remove firewall rule for port 80 (HTTP)
      ansible.posix.firewalld:
        port: 80/tcp
        state: disabled
        permanent: true
        immediate: true

    - name: Remove firewall rule for port 443 (HTTPS)
      ansible.posix.firewalld:
        port: 443/tcp
        state: disabled
        permanent: true
        immediate: true

    - name: Reload firewalld to apply changes
      ansible.builtin.service:
        name: firewalld
        state: reloaded
~
~
~
"close_http_https.yml" 26L, 572B                                          26,23         All
```

Ansible Firewall Hardening – Closing HTTP and HTTPS Ports

This project demonstrates the use of Ansible to centrally manage and harden firewall configurations on Linux systems using firewalld. The playbook is designed to close commonly exposed web ports (HTTP and HTTPS) on a development environment in a controlled, repeatable, and auditable manner.

Before execution, the playbook is validated using Ansible's built-in syntax check to ensure correctness and prevent configuration errors. The playbook is then executed with privilege escalation to apply firewall changes at the system level.

Automated Firewall Tasks

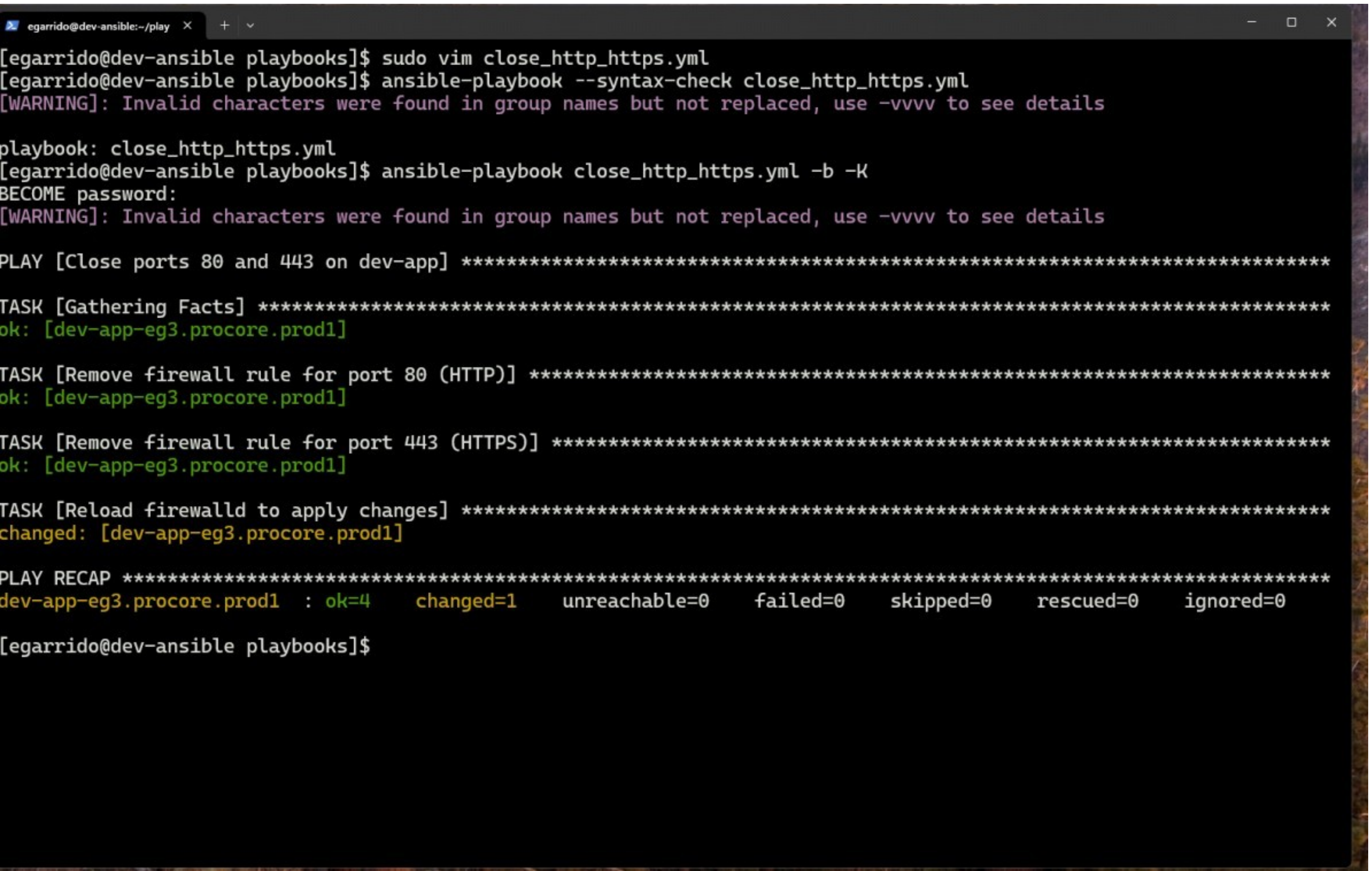
The playbook performs the following actions:

Disables inbound traffic on TCP port 80 (HTTP) using the ansible.posix.firewalld module

Disables inbound traffic on TCP port 443 (HTTPS) with permanent and immediate enforcement

Reloads the firewalld service to ensure all rule changes take effect without requiring a reboot

Each task is idempotent, meaning the playbook can be safely re-run without introducing unintended changes.



```
[egarrido@dev-ansible playbooks]$ sudo vim close_http_https.yml
[egarrido@dev-ansible playbooks]$ ansible-playbook --syntax-check close_http_https.yml
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details

playbook: close_http_https.yml
[egarrido@dev-ansible playbooks]$ ansible-playbook close_http_https.yml -b -K
BECOME password:
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details

PLAY [Close ports 80 and 443 on dev-app] ************************************************************

TASK [Gathering Facts] ******************************************************************************
ok: [dev-app-eg3.procore.prod1]

TASK [Remove firewall rule for port 80 (HTTP)] ******************************************************
ok: [dev-app-eg3.procore.prod1]

TASK [Remove firewall rule for port 443 (HTTPS)] ****************************************************
ok: [dev-app-eg3.procore.prod1]

TASK [Reload firewalld to apply changes] ************************************************************
changed: [dev-app-eg3.procore.prod1]

PLAY RECAP ******************************************************************************************
dev-app-eg3.procore.prod1  : ok=4    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[egarrido@dev-ansible playbooks]$
```

The terminal output shows the result of running firewall-cmd --list-all on a Linux system to verify the active firewalld configuration after applying firewall changes.

The public zone is active and bound to the ens192 network interface. Default firewall behavior is enabled, with ICMP block inversion disabled and packet forwarding allowed. Network address translation (masquerading) is turned off.

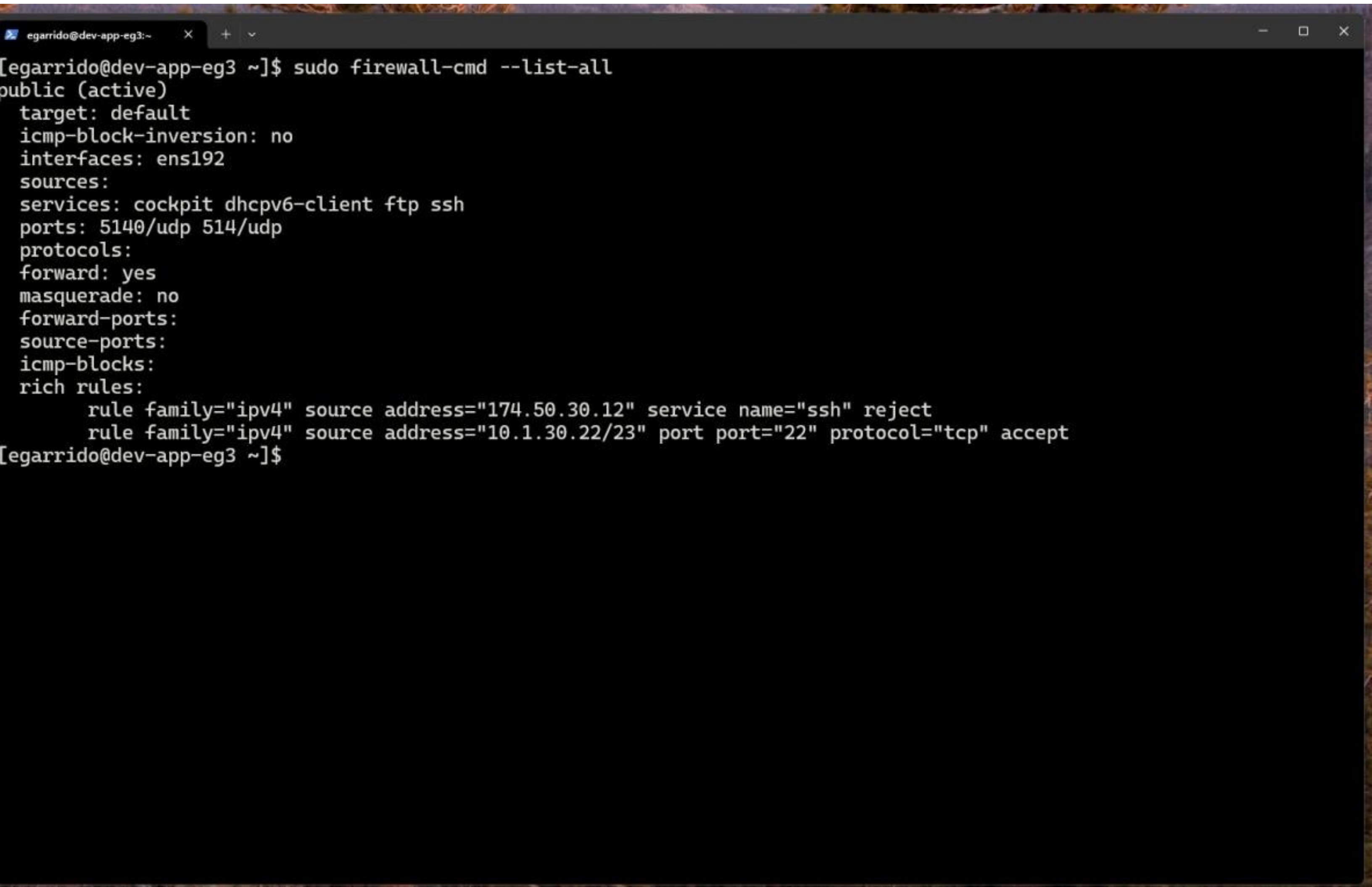
Only a limited set of services are permitted through the firewall, including cockpit, dhcpv6-client, ftp, and ssh. No HTTP or HTTPS services are listed, confirming that ports 80 and 443 are no longer open. The only explicitly allowed ports shown are 514/udp, indicating restricted network exposure.

Two rich rules are configured for SSH access control:

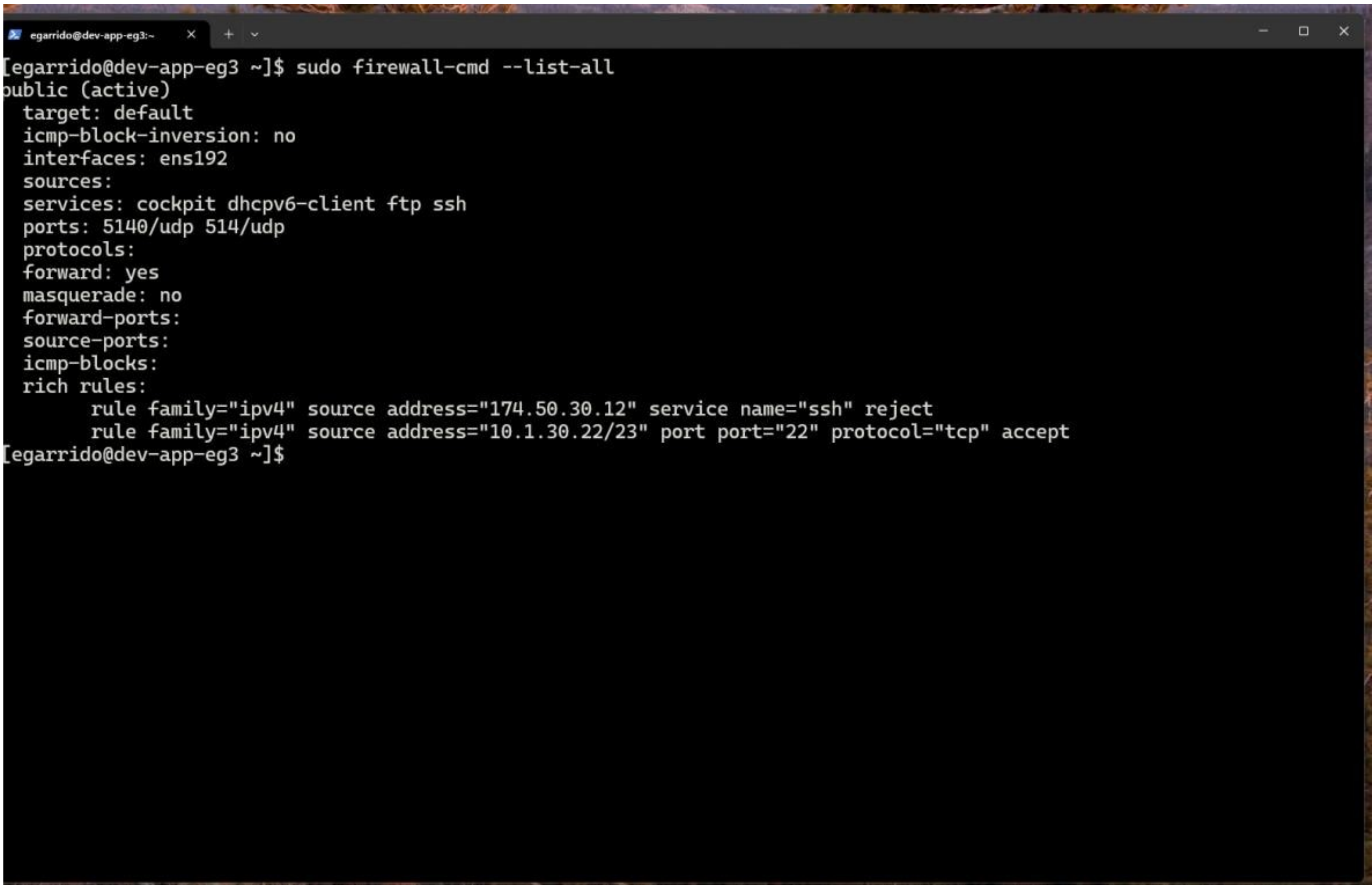
One rule explicitly rejects SSH traffic from a specific external IPv4 source address.

A second rule allows SSH access on TCP port 22 only from a defined internal subnet.

This output confirms that the firewall is actively enforcing a hardened configuration, limiting access to essential services and tightly controlling SSH connectivity based on source address.



```
[egarrido@dev-app-eg3 ~]$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens192
  sources:
  services: cockpit dhcpv6-client ftp ssh
  ports: 5140/udp 514/udp
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
        rule family="ipv4" source address="174.50.30.12" service name="ssh" reject
        rule family="ipv4" source address="10.1.30.22/23" port port="22" protocol="tcp" accept
[egarrido@dev-app-eg3 ~]$
```



```
[egarrido@dev-app-eg3 ~]$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens192
  sources:
  services: cockpit dhcpv6-client ftp ssh
  ports: 5140/udp 514/udp
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
        rule family="ipv4" source address="174.50.30.12" service name="ssh" reject
        rule family="ipv4" source address="10.1.30.22/23" port port="22" protocol="tcp" accept
[egarrido@dev-app-eg3 ~]$
```

Summary

This project uses Ansible to harden a Linux firewall by disabling unnecessary HTTP and HTTPS access and enforcing the changes through firewalld. The configuration is validated directly on the host to confirm that only essential services remain accessible and that SSH access is tightly controlled using source-based rules. The workflow demonstrates secure, repeatable firewall management with built-in verification suitable for enterprise environments.