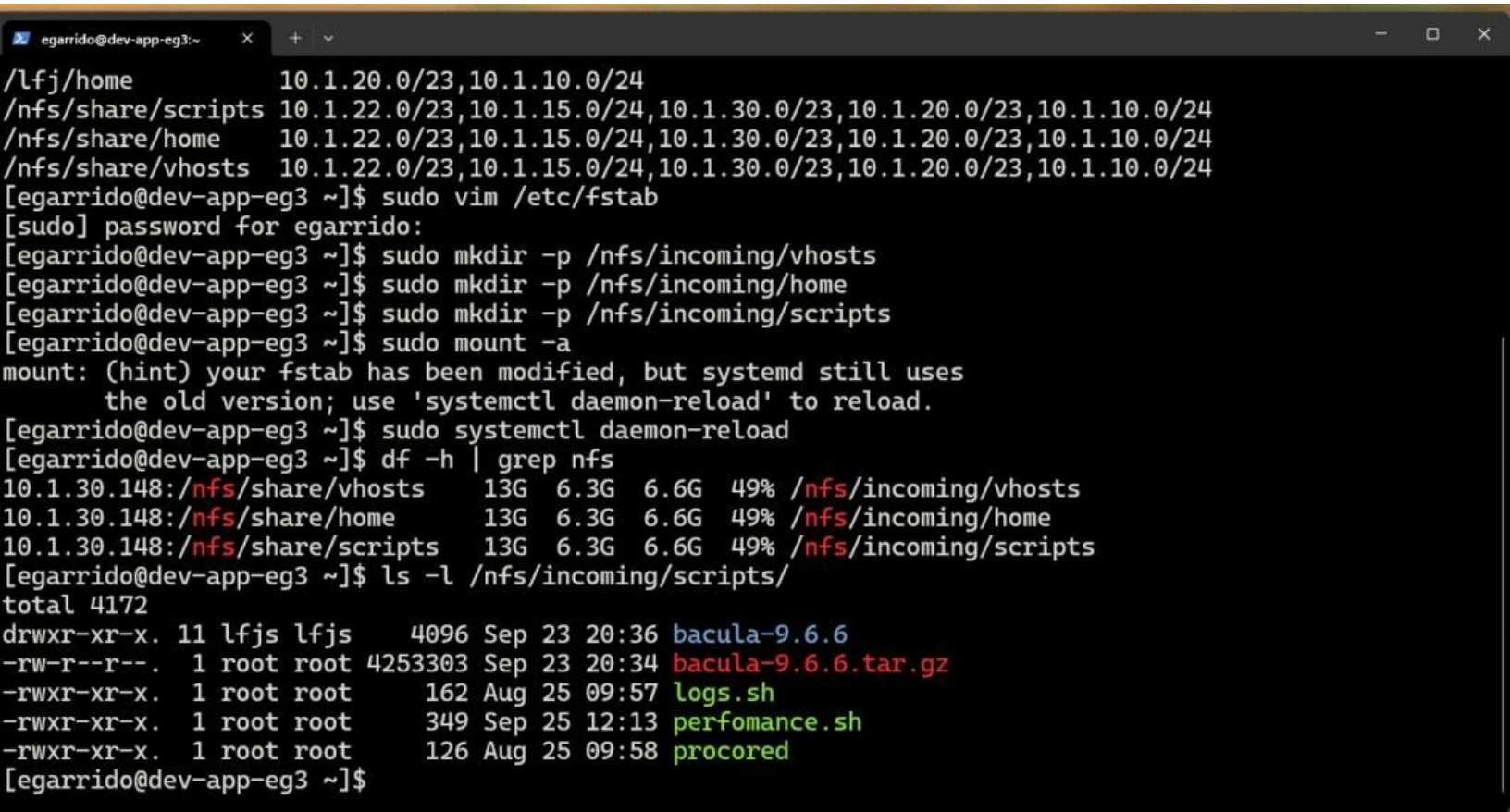
Systemd Daemon Deployment and Validation Across Environments

This effort focused on delivering a consistent, resilient background service across development, performance, and staging systems. Shared storage was standardized by defining persistent mounts, creating required directory paths, and validating that all network file system resources were available after reloads. Each host was checked to ensure uniform access to shared scripts and supporting assets.

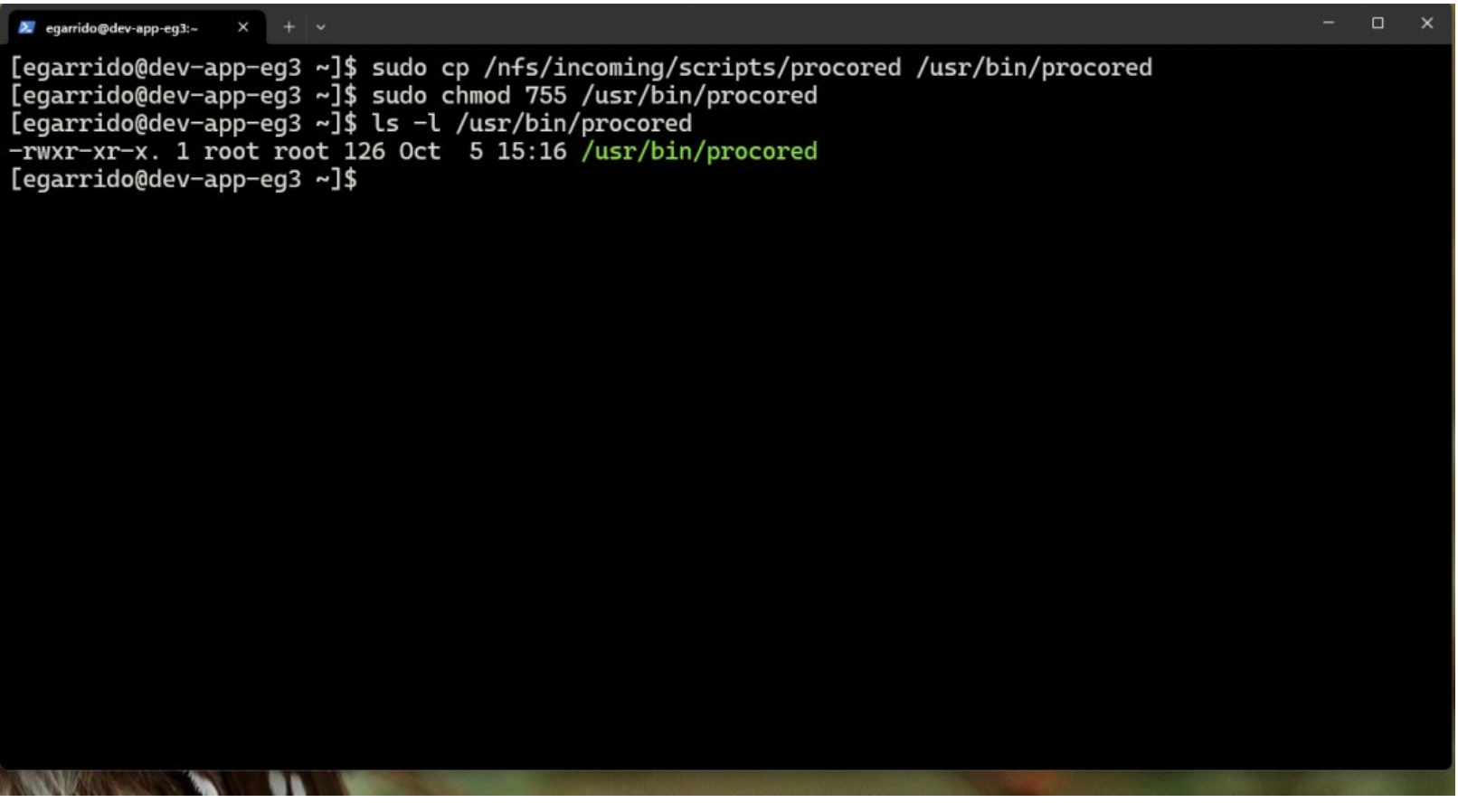
The executable was deployed locally on every server by placing it in a system-wide binary location and applying the correct ownership and execution permissions. A dedicated system service definition was then created to manage the process lifecycle, including proper startup ordering, automatic restarts, and boot-time activation to guarantee reliability.

Final validation steps included reloading the service manager, enabling the service to start automatically, and confirming active runtime status on each host. Process details, resource usage, and log output were reviewed to ensure correct operation within an isolated control group. The result is a repeatable, environment-wide implementation that ensures the daemon runs persistently and consistently across all targeted systems.

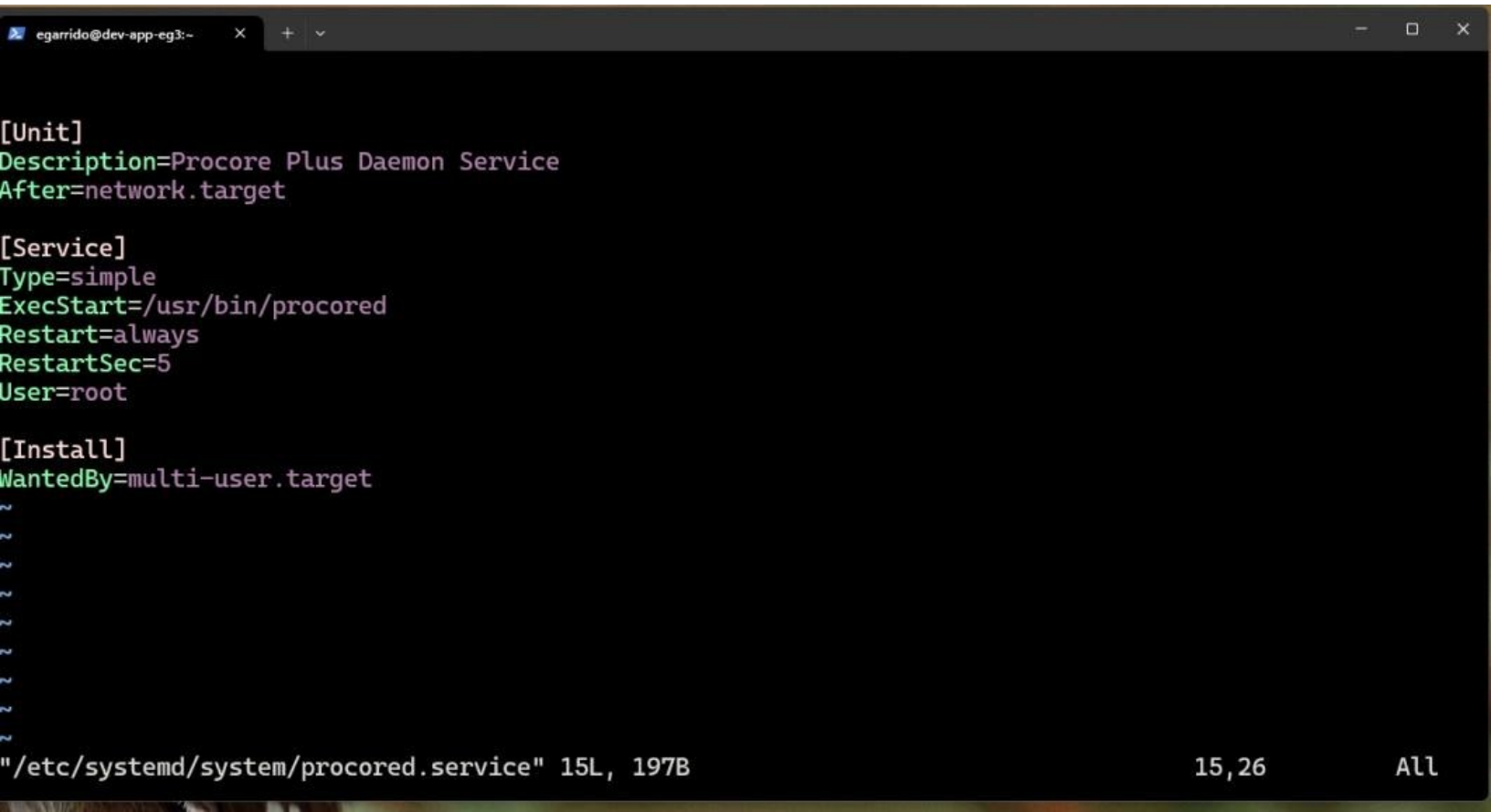
The output shows NFS mounts being configured and validated on the dev-app-eg3 server. Multiple NFS entries are present in /etc/fstab, directories under /nfs/incoming/ are created, and all mounts are applied successfully after reloading the systemd daemon. Disk usage confirms the NFS shares are mounted from the remote server, and listing /nfs/incoming/scripts/ verifies the presence of deployment scripts, including procorded, along with related utilities and archives.



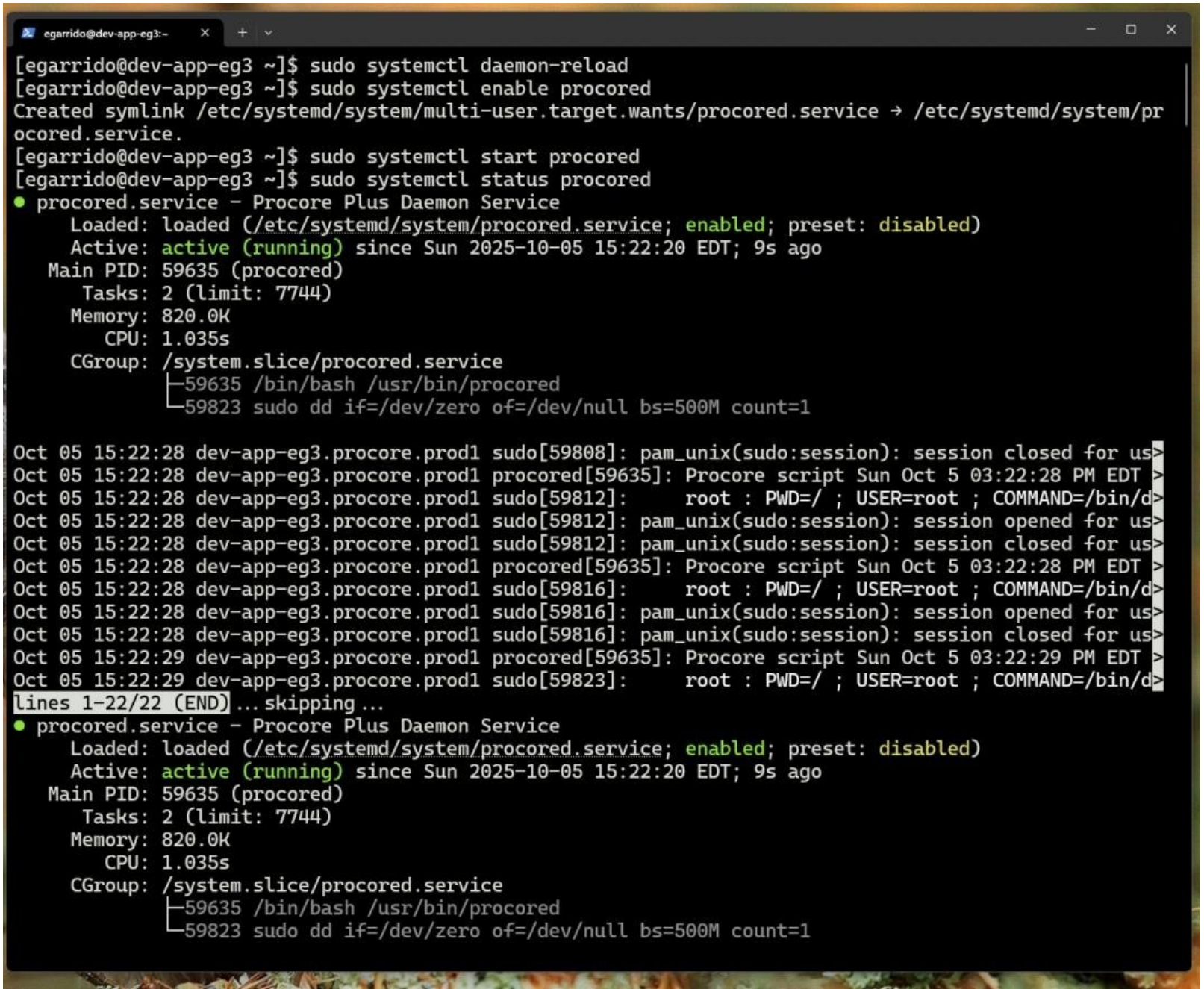
The script procorded is copied from the shared NFS path into /usr/bin, making it available locally on the system. Execute permissions are then applied, and a long listing confirms the file exists in /usr/bin/procorded, is owned by root, and is marked as executable.



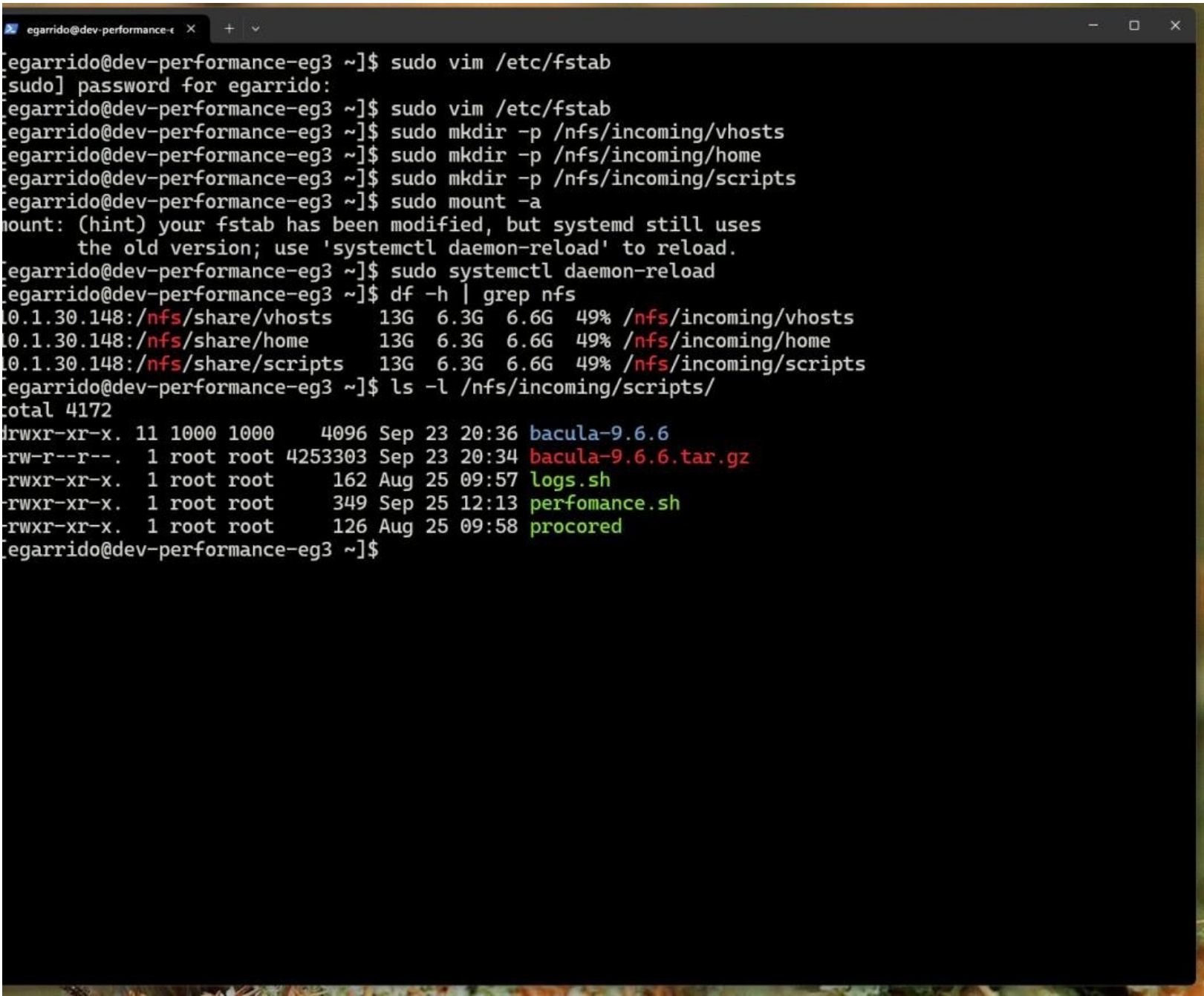
A systemd unit file is defined to manage the daemon as a persistent service. The unit specifies startup after the network is available, runs the executable from /usr/bin/procorded using a simple service type, enforces automatic restarts with a five-second delay, and operates with root privileges. The service is enabled for the multi-user target to ensure it starts automatically during system boot.



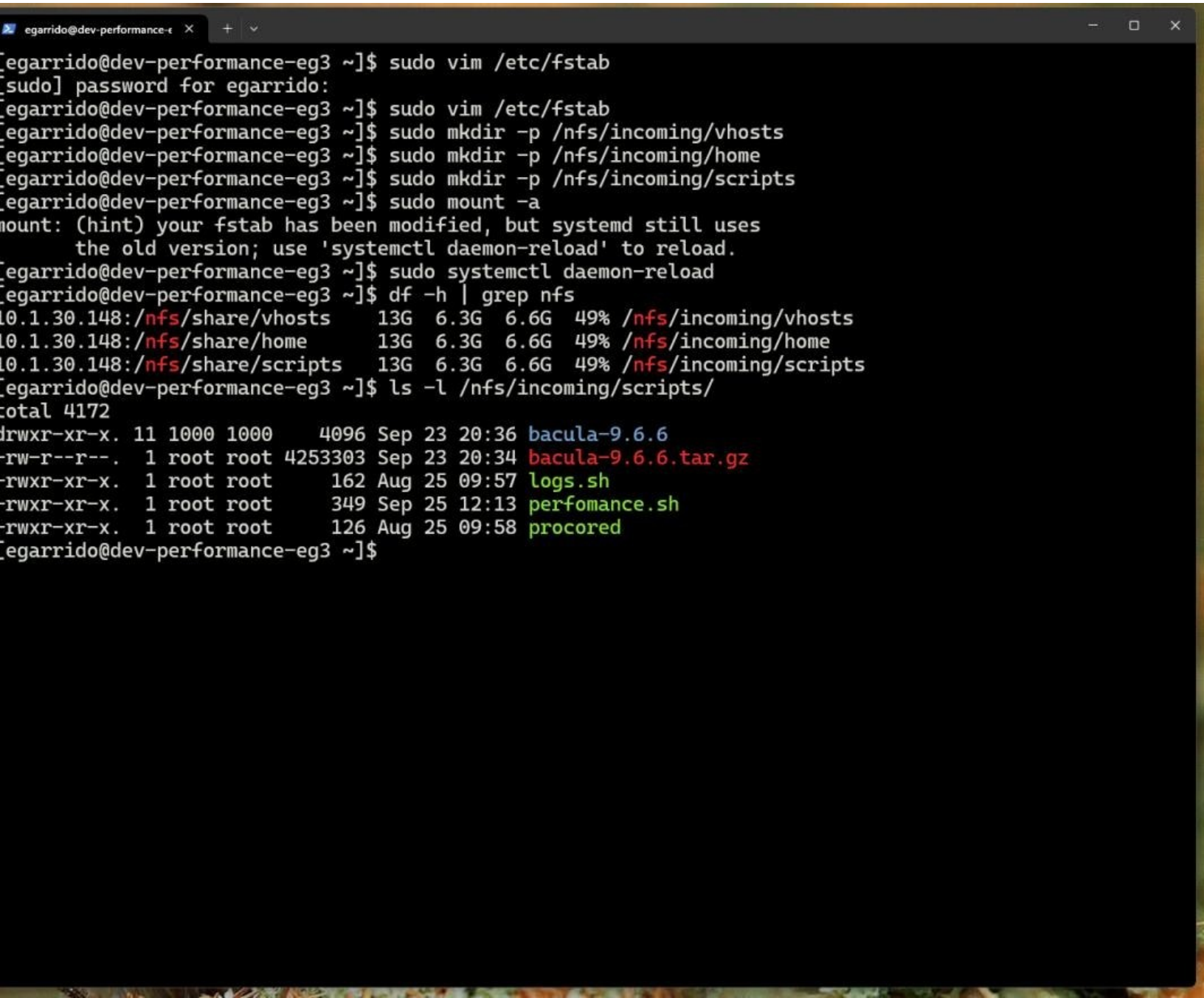
The systemd configuration is reloaded to register the new unit file, and the service is enabled to start automatically at boot. The daemon is then started successfully, and its status confirms it is active and running under the expected service name. The output shows the service loaded from /etc/systemd/system, operating within its own cgroup, and executing the procorded script as intended, verifying a successful deployment and persistent runtime state.



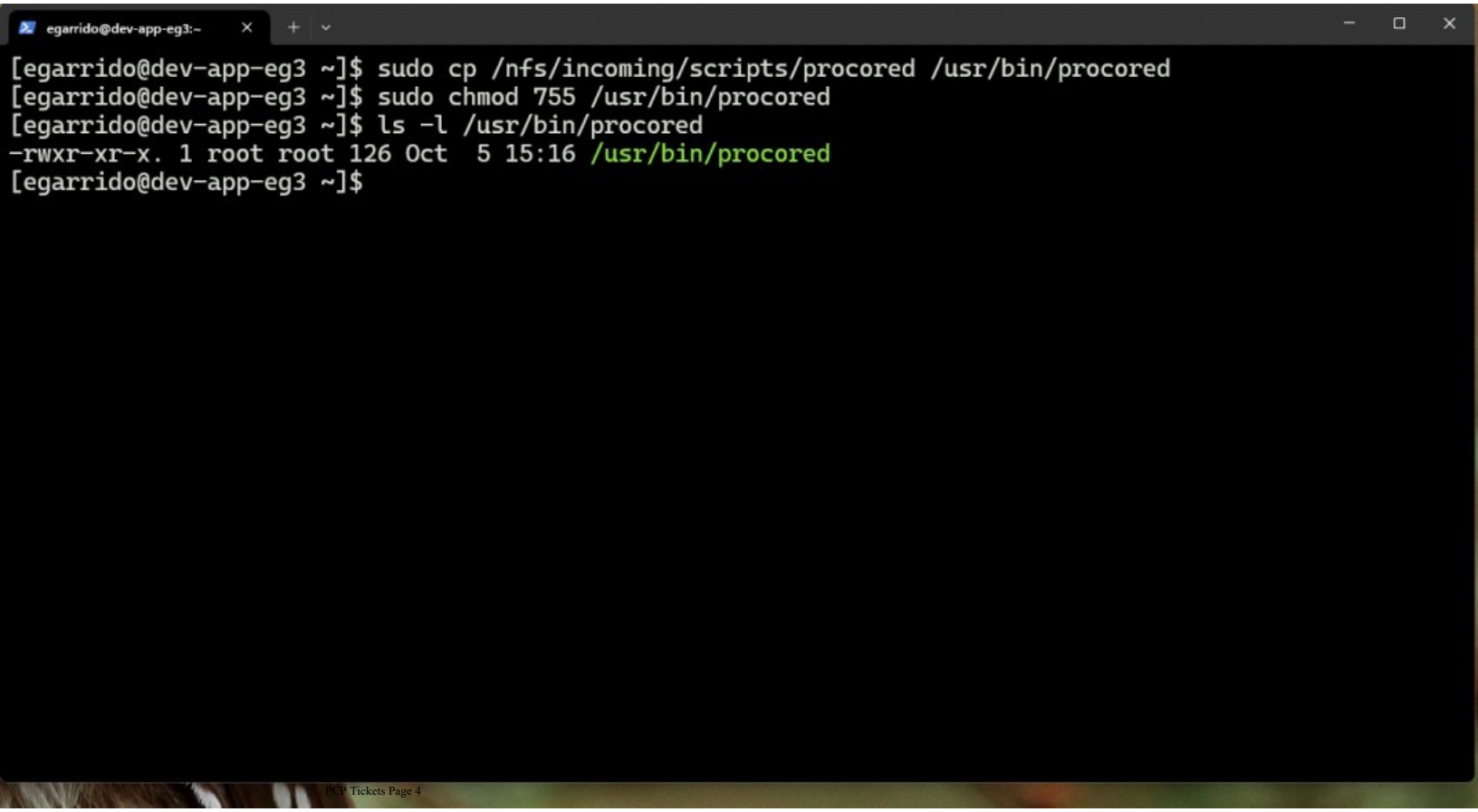
The NFS configuration is repeated on the dev-performance-eg3 server by updating /etc/fstab, creating the required mount directories under /nfs/incoming, and applying the mounts. After reloading the systemd daemon, disk usage confirms that all NFS shares are mounted successfully from the remote server. Listing the scripts directory verifies that the shared files, including the executable used for the daemon, are present and accessible on this host.



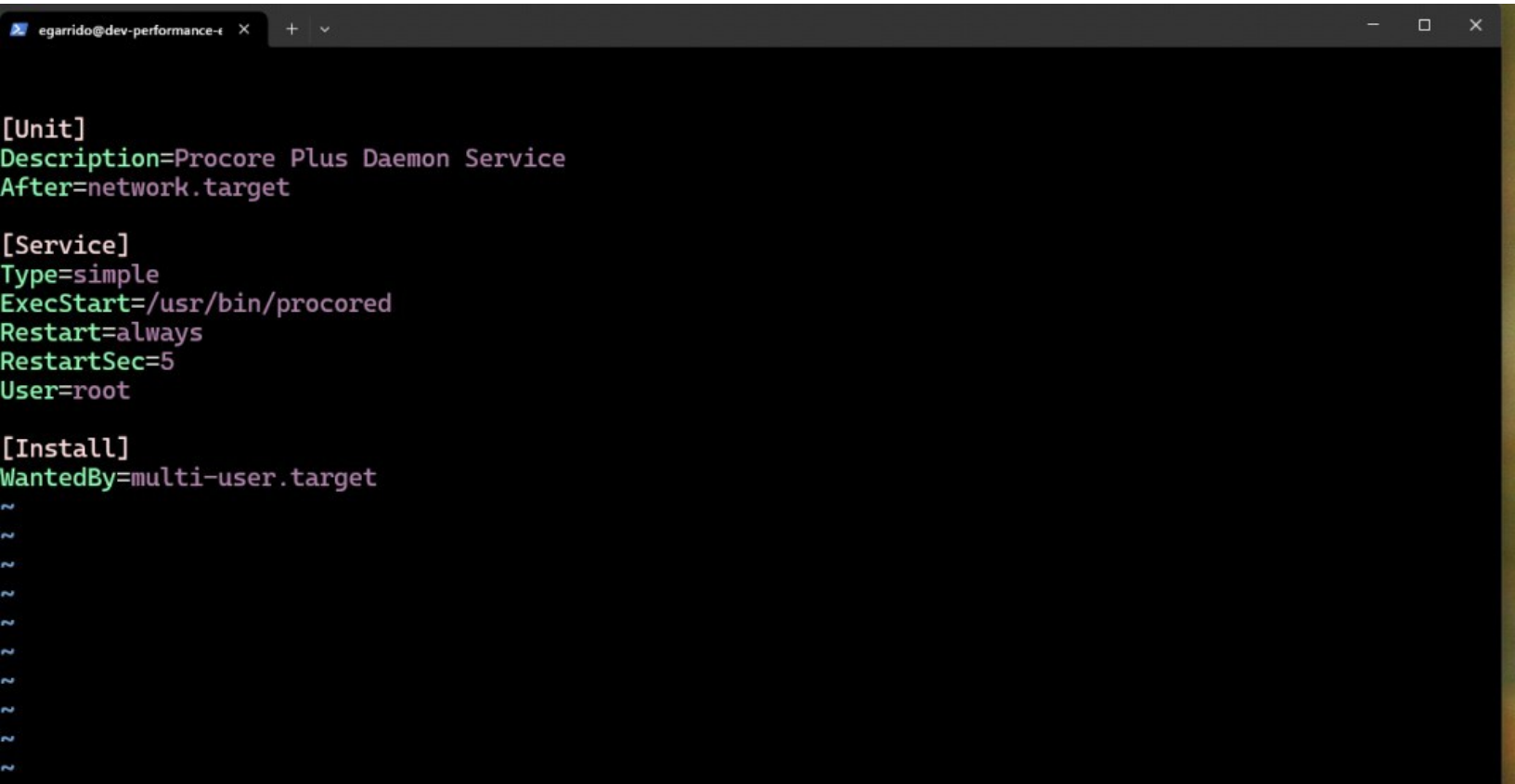
The shared storage configuration is applied on the dev-performance-eg3 host by editing /etc/fstab, creating the required directories under /nfs/incoming, and mounting all entries. After reloading the systemd configuration, the NFS shares are confirmed as mounted successfully. A directory listing of /nfs/incoming/scripts verifies that all shared files are available on the system, including the executable required for the daemon.



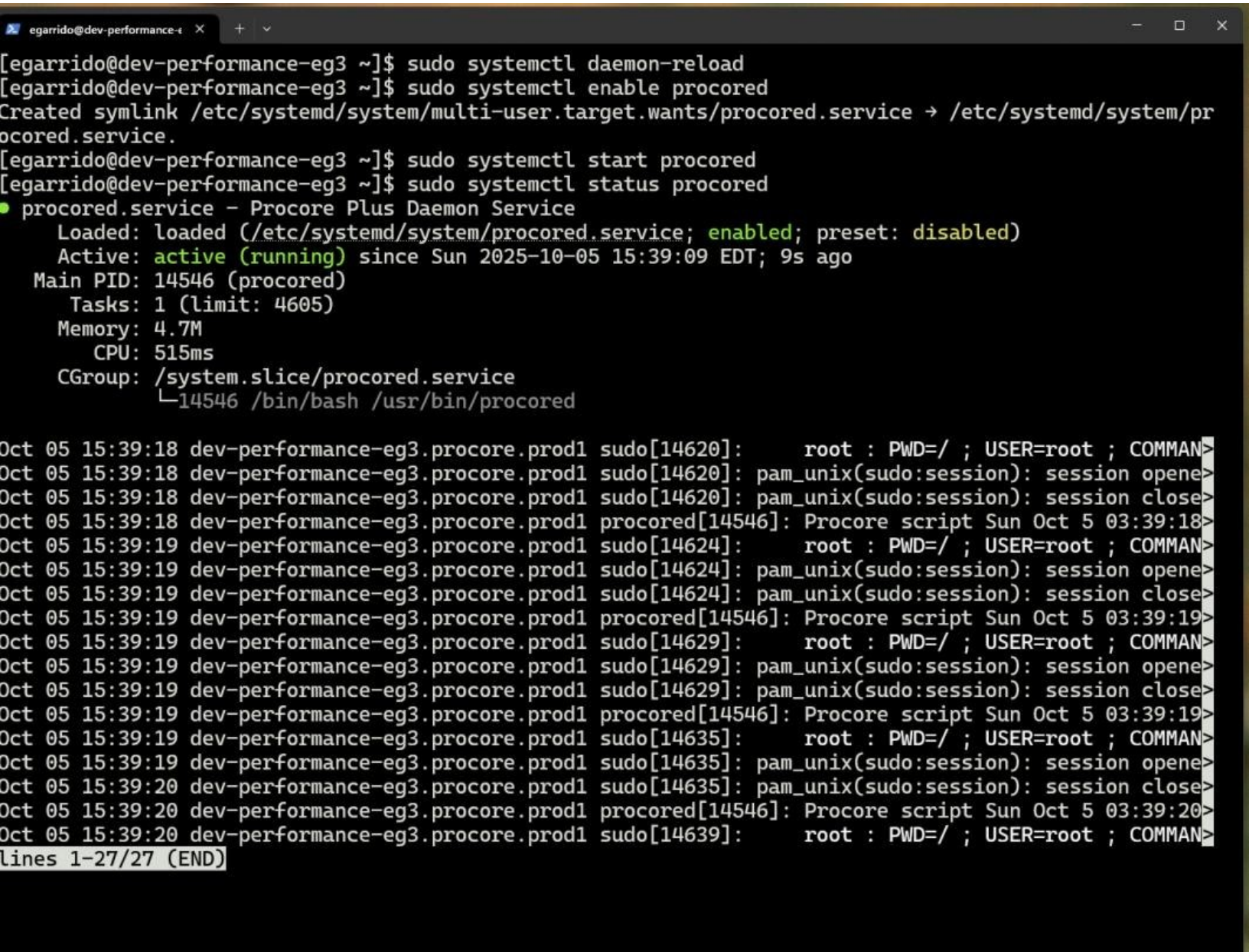
The executable is copied from the shared NFS scripts directory into /usr/bin on the system. File permissions are updated to allow execution, and a long listing confirms the binary exists in /usr/bin, is owned by root, and has the correct executable permissions applied.



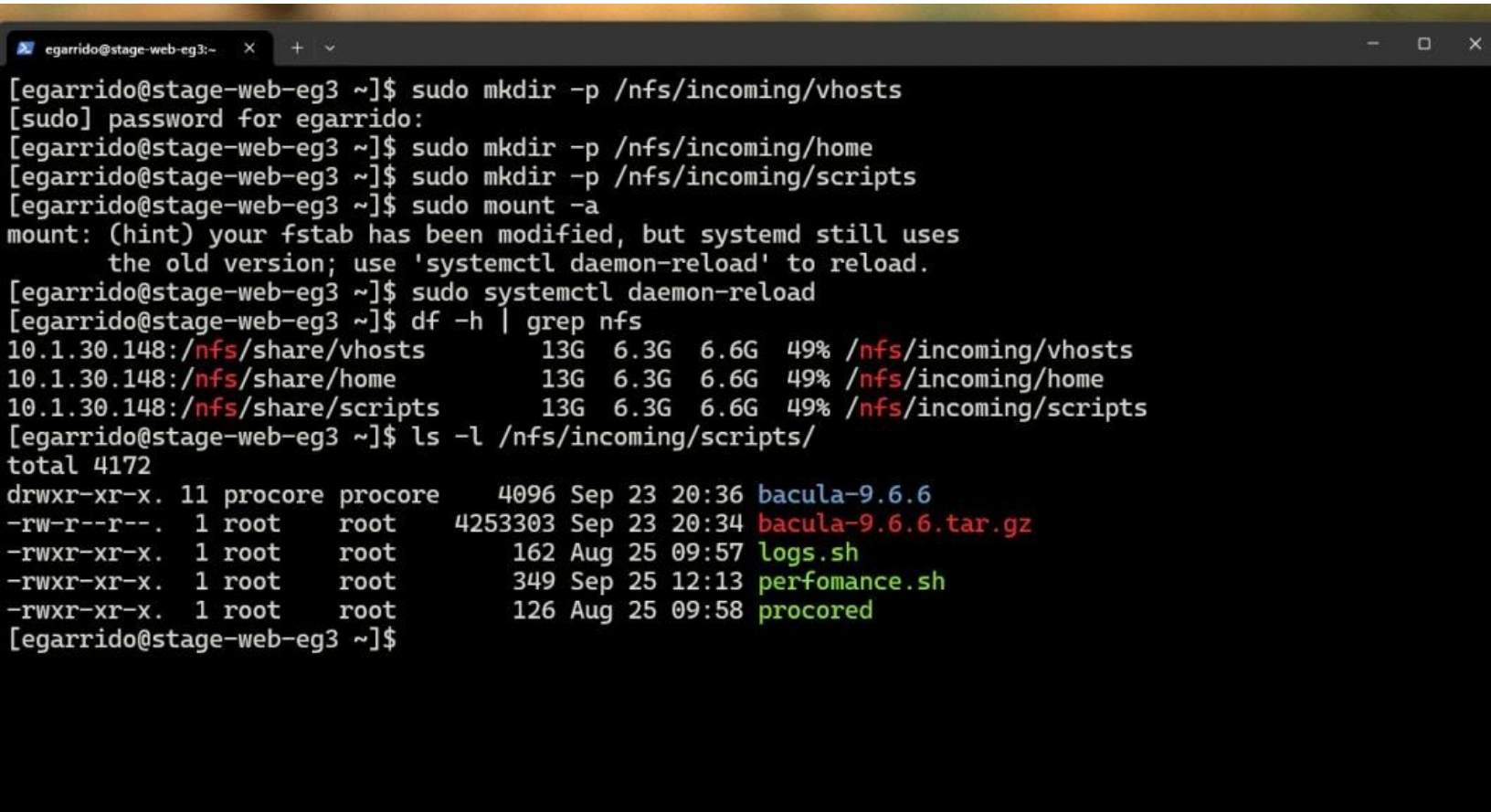
A systemd unit file is created on the dev-performance-eg3 host to manage the daemon as a persistent service. The unit is configured to start after network initialization, execute the binary from /usr/bin, automatically restart on failure with a defined delay, and run with root privileges. The service is tied to the multi-user target to ensure it starts automatically during system boot.



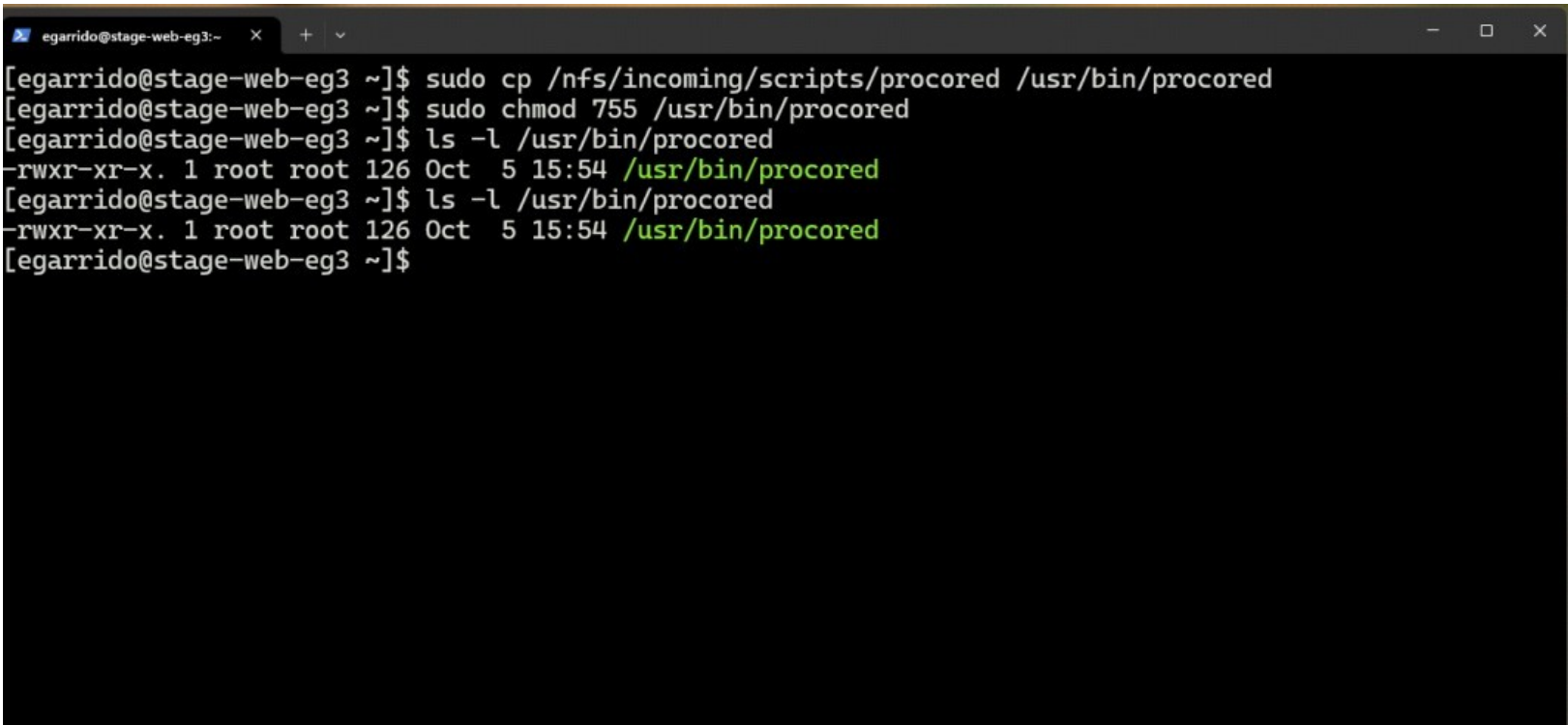
The systemd daemon is reloaded to register the new unit file, and the service is enabled to start automatically at boot on the dev-performance-eg3 host. The service is then started successfully, and the status output confirms it is active and running. The process is executing the binary from /usr/bin, operating within its dedicated cgroup, and generating expected log entries, verifying that the daemon is functioning correctly and persists across reboots.



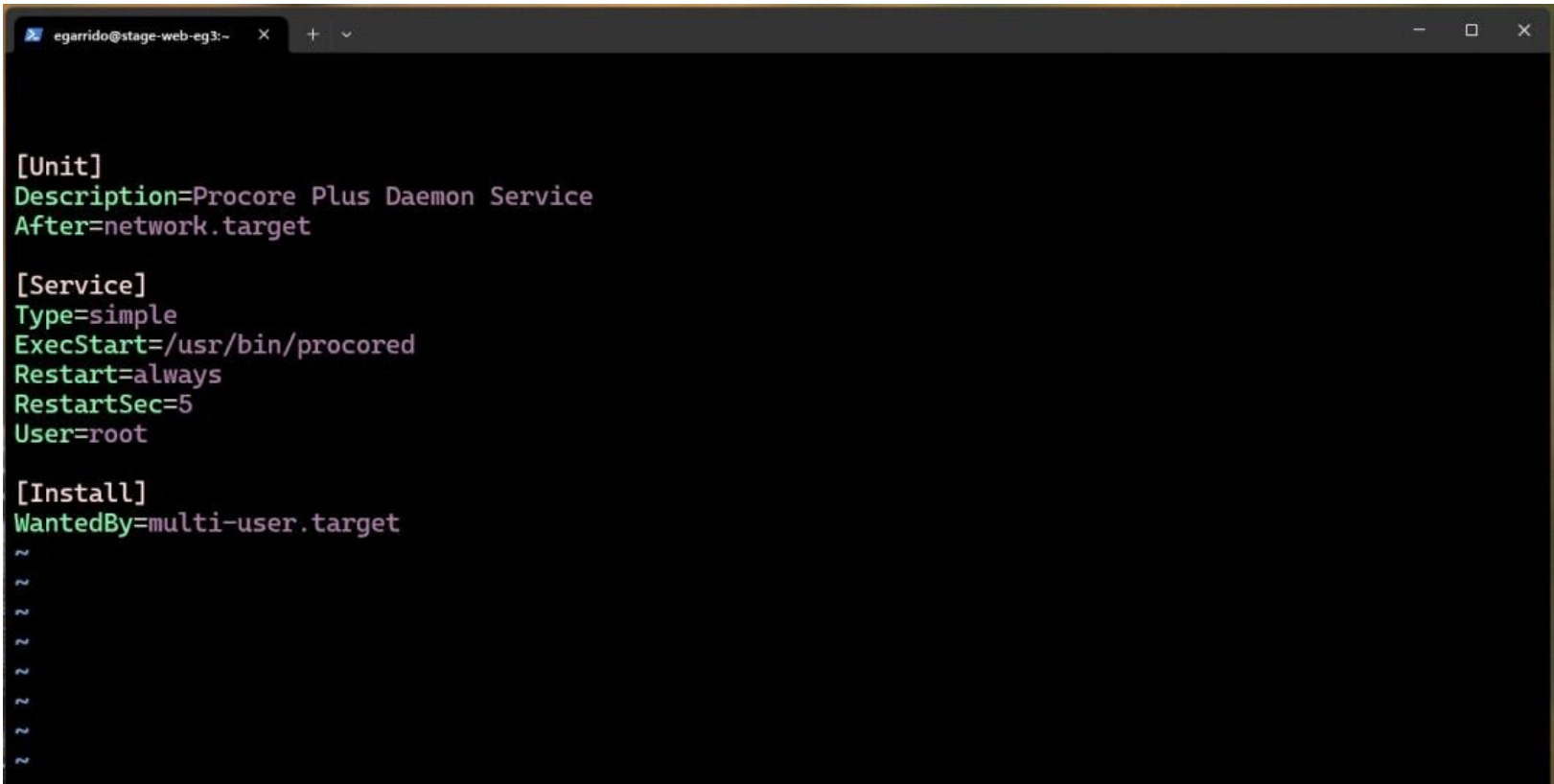
The shared storage setup is completed on the stage-web-eg3 host by creating the required directories under /nfs/incoming, mounting the NFS entries, and reloading the systemd daemon to apply the updated configuration. Disk usage output confirms that all NFS shares are mounted successfully from the remote server. A directory listing of /nfs/incoming/scripts verifies that the shared scripts are accessible on the system, including the executable required for the daemon.



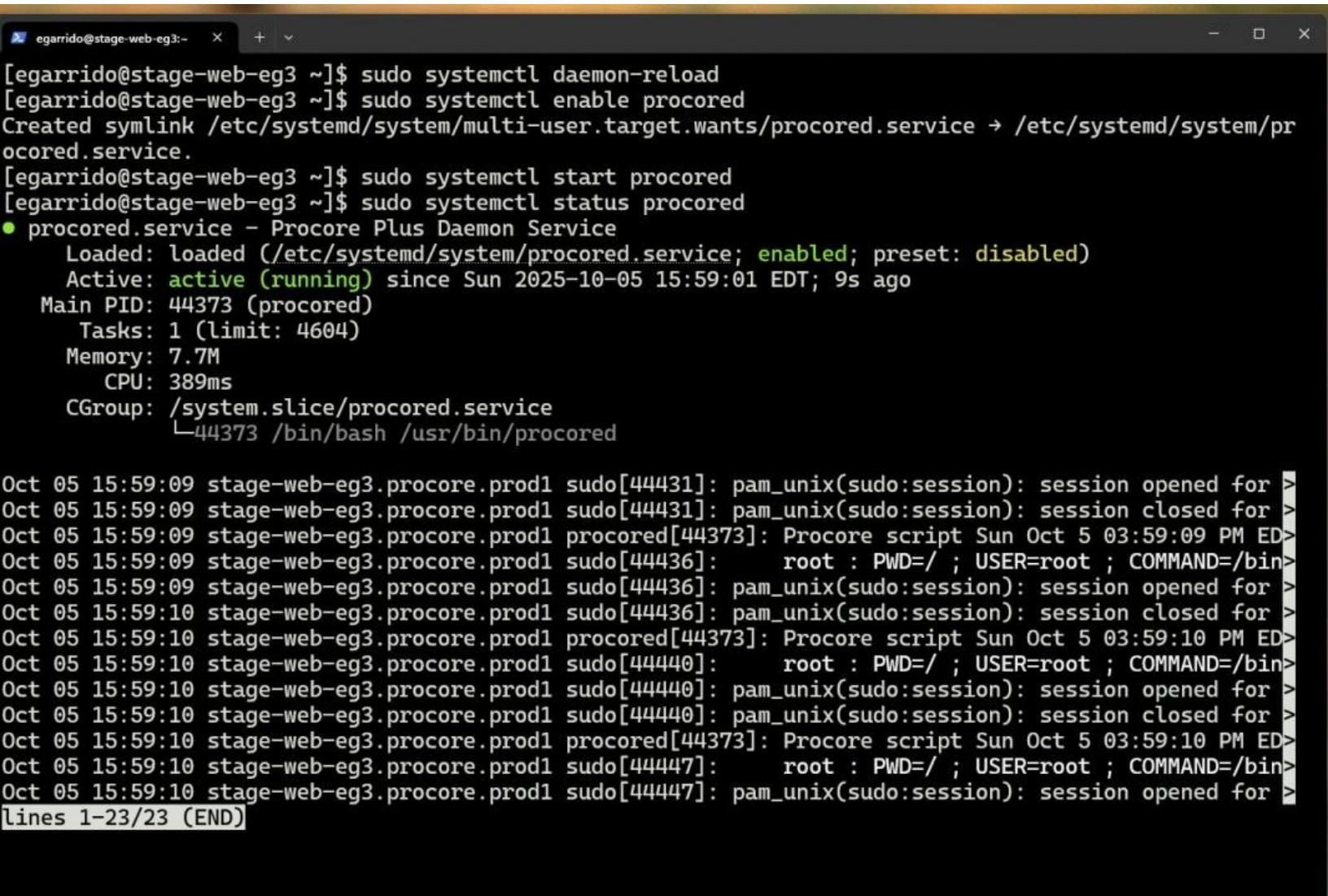
The executable is deployed on the stage-web-eg3 host by copying it from the shared NFS scripts directory into /usr/bin. Execute permissions are applied, and a long listing confirms the file exists in the correct location, is owned by root, and has the appropriate executable permissions set.



A systemd unit file is created on the stage-web-eg3 host to manage the daemon as a persistent background service. The unit is configured to start after the network is available, execute the binary from /usr/bin, automatically restart on failure with a defined delay, and run with root privileges. The service is enabled under the multi-user target to ensure it starts automatically during system boot.



The systemd daemon is reloaded on the stage-web-eg3 host to register the new unit file, and the service is enabled to start automatically at boot. The daemon is then started successfully, and the status output confirms it is active and running. The service is loaded from /etc/systemd/system, executes the binary from /usr/bin, operates within its own cgroup, and generates expected log entries, confirming correct operation and persistence across reboots.



Summary

The most recent work focused on standardizing and deploying a persistent background service across multiple environments, including development, performance, and staging systems. Shared storage was configured consistently by defining persistent NFS mounts, creating required directory structures, and validating mount availability after system reloads. Each host was verified to have reliable access to shared scripts and supporting resources.

The executable was deployed locally on each system by copying it from shared storage into a system-wide binary location and applying the correct ownership and execution permissions. A dedicated systemd service was created to manage the process lifecycle, ensuring proper startup sequencing, automatic restarts on failure, and activation at boot.

Final steps included reloading the systemd manager, enabling and starting the service, and validating its active runtime state on every host. Service status, process details, and log output were reviewed to confirm stable operation. This work established a repeatable, environment-wide approach for running and maintaining a critical daemon reliably across all targeted systems.