Nagios NRPE Monitoring Configuration – Overview

This work documents the configuration and validation of Nagios Remote Plugin Executor (NRPE) monitoring for a Linux staging web server environment. The objective of this effort was to securely integrate a new host into the existing Nagios monitoring infrastructure, ensuring reliable service checks, controlled access, and continuous availability monitoring.

The process begins with configuring the NRPE daemon to run under a dedicated service account and restricting access to explicitly authorized monitoring hosts. Network-level controls and daemon settings are validated to ensure NRPE listens on the correct port and only accepts approved connections. Service management is handled through systemctl, confirming that NRPE is enabled at boot and actively running.

On the Nagios server side, a structured host definition is created using standardized templates to maintain consistency across environments. Monitoring parameters such as check intervals, retry limits, and notification periods are defined to support 24x7 monitoring and timely alerting. Host object files are organized and included modularly within the Nagios configuration to support scalability and maintainability across development and staging systems.

The configuration concludes with verification steps to confirm successful service startup, host registration, and communication between the Nagios server and the monitored host. Together, these steps demonstrate a complete, production-style monitoring setup aligned with best practices for security, clarity, and operational reliability.

The terminal output shows the installation of NRPE (Nagios Remote Plugin Executor) on an Enterprise Linux 9 (x86_64) system using yum.

The command executed is sudo yum install nrpe -y.

A warning appears indicating that /etc/yum.repos.d/MariaDB.repo could not be loaded, but the operation continues and is not blocked.

Package metadata is retrieved from the EPEL 9 repositories, including EPEL, EPEL Next, and EPEL OpenH264.

Dependencies are successfully resolved, and two packages are selected for installation:

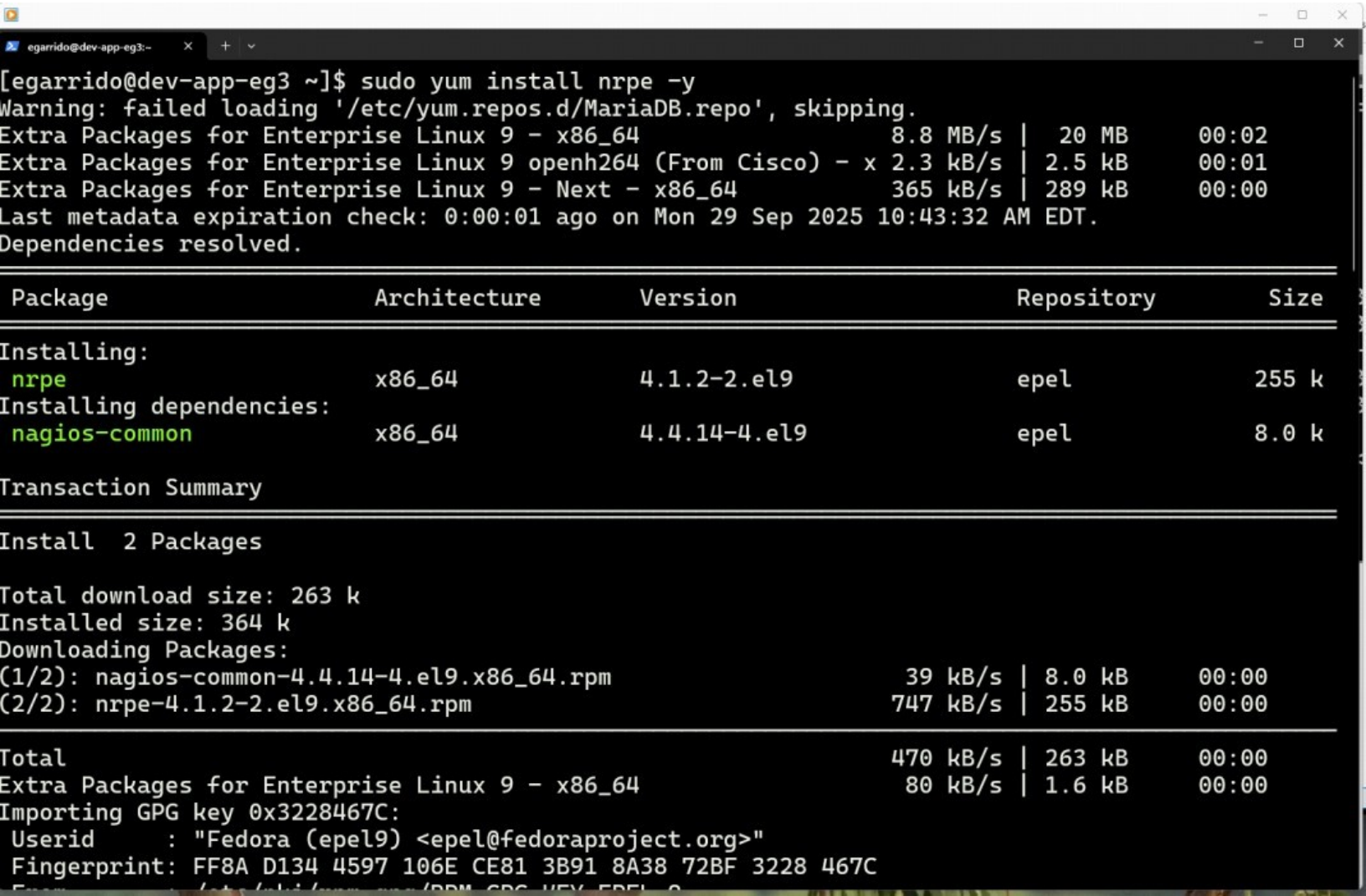
nrpe version 4.1.2-2.el9

nagios-common version 4.4.14-4.el9

Both packages are downloaded and installed successfully with a total download size of 263 KB.

The system imports and trusts the EPEL GPG key, confirming package authenticity.

Overall, the screenshot confirms a successful NRPE installation from the EPEL repository, with only a non-fatal repository warning unrelated to NRPE.



This screenshot displays a portion of the NRPE configuration file located at /etc/nagios/nrpe.cfg opened in a text editor on an Enterprise Linux 9 system.

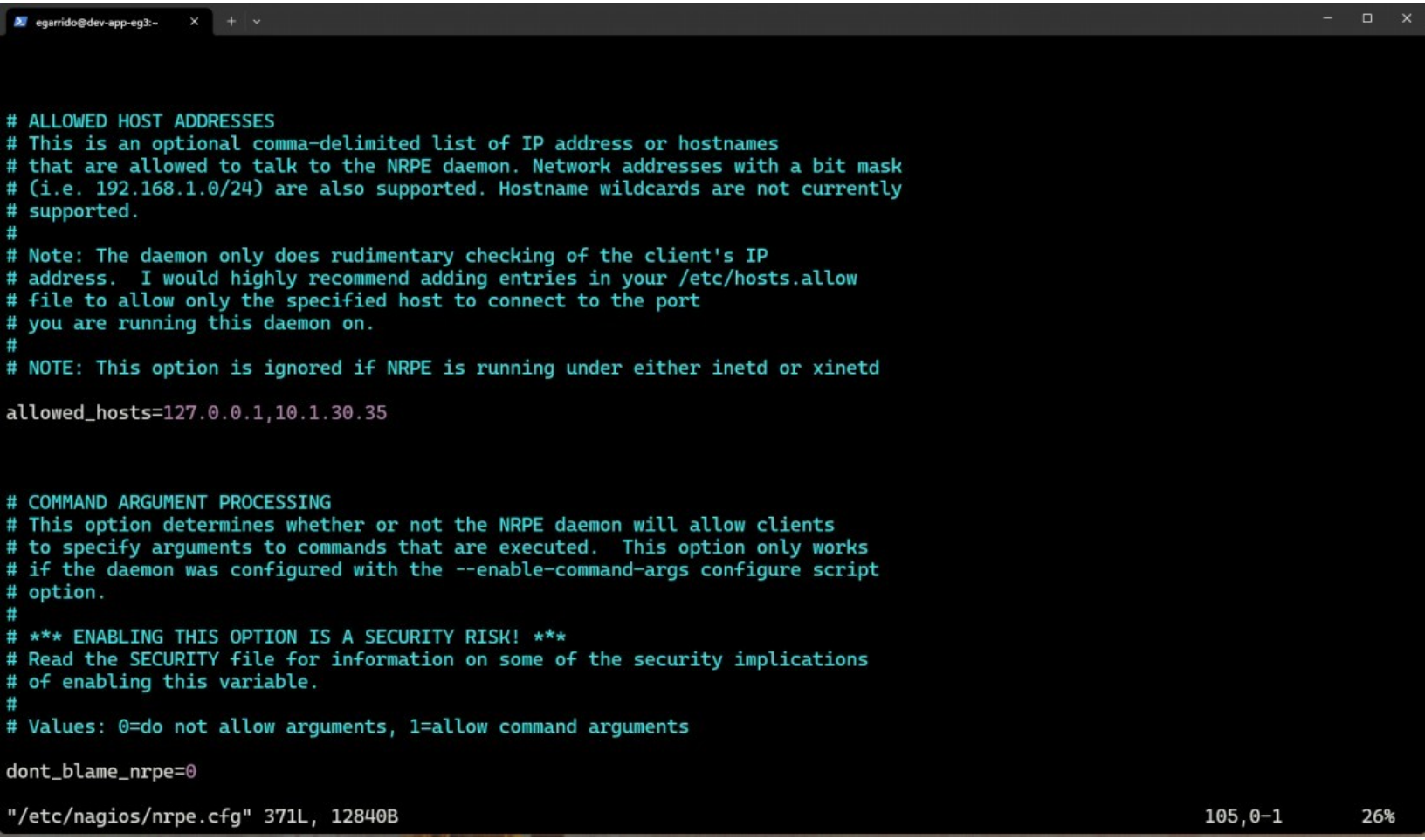
The allowed_hosts directive is configured to restrict which systems are permitted to communicate with the NRPE daemon.

Only the local host (127.0.0.1) and a specific monitoring server IP (10.1.30.35) are allowed to connect, enforcing basic network-level access control.

Inline comments explain that NRPE performs only minimal client IP validation and recommend additional protection via /etc/hosts.allow.

The configuration notes that this setting is ignored if NRPE is run under inetd or xinetd.

The command argument processing section shows dont_blame_nrpe=0, meaning NRPE is configured not to accept command arguments from remote clients, which helps reduce security risk.

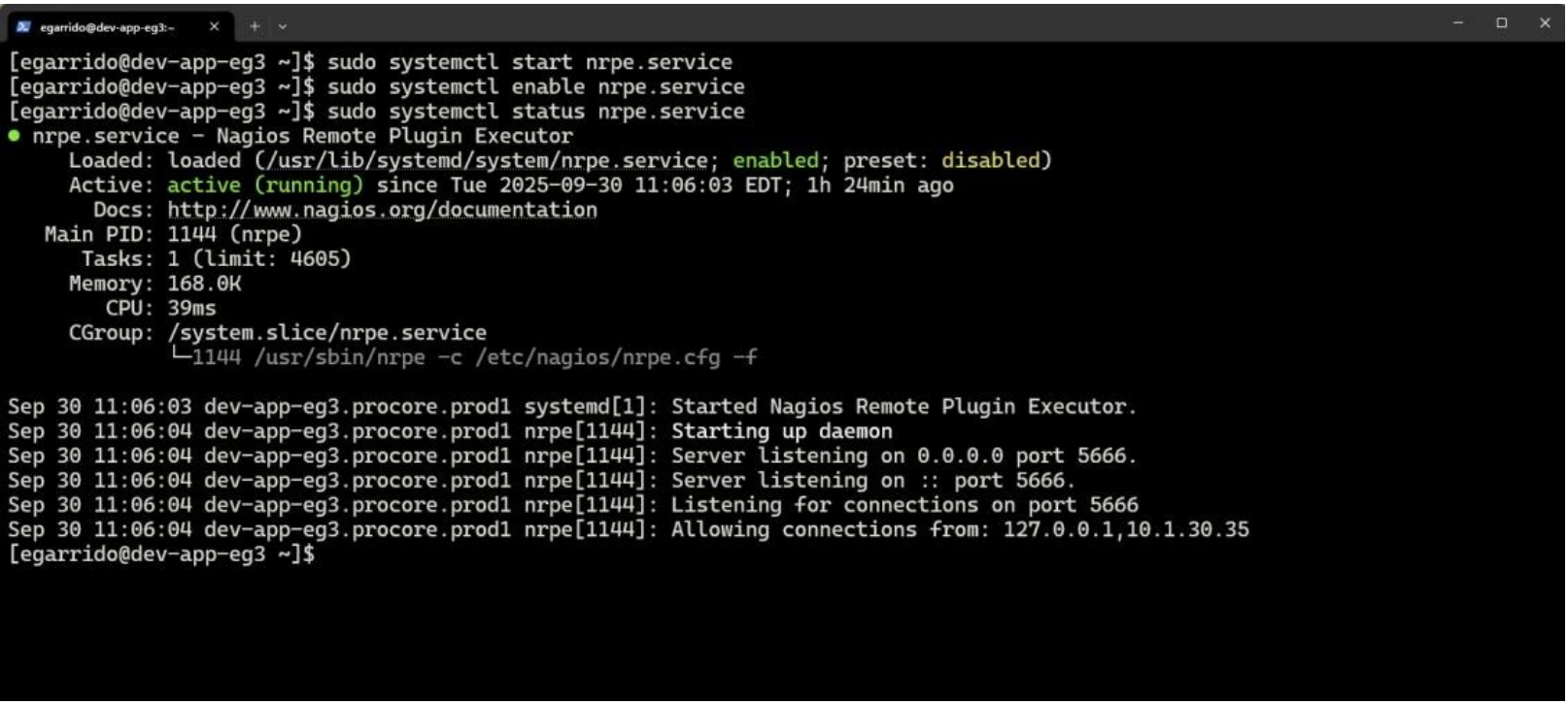


The NRPE service has been successfully started, enabled, and verified on an Enterprise Linux 9 system using systemd. The service is configured to launch automatically at boot and is currently running without errors.

Systemd confirms that nrpe.service is loaded from /usr/lib/systemd/system/nrpe.service and is in an active (running) state. The NRPE daemon is operating under its own process and is using the configuration file located at /etc/nagios/nrpe.cfg.

Runtime logs show that NRPE is listening on the default monitoring port 5666 for both IPv4 and IPv6 connections. Incoming connections are restricted to the local host and the authorized Nagios monitoring server, as defined in the allowed_hosts configuration.

This verification confirms that NRPE is properly configured, securely restricted, and ready to accept remote health checks from approved monitoring system

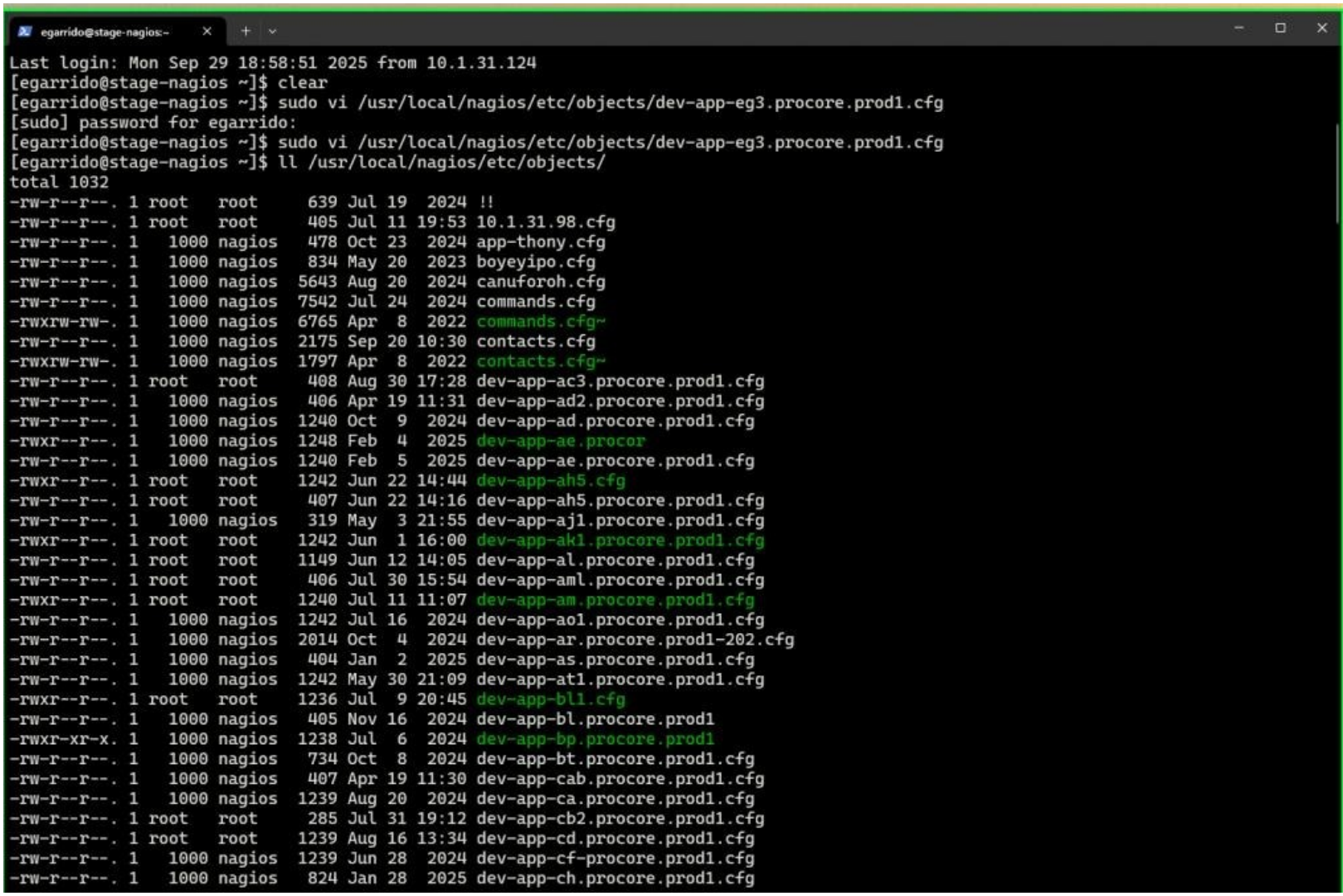


Access to the Nagios configuration directory on the stage monitoring server is shown, where host and service definitions are centrally managed. The administrator navigates to /usr/local/nagios/etc/objects/ and edits a host-specific configuration file for the dev-app-eg3 system within the production environment.

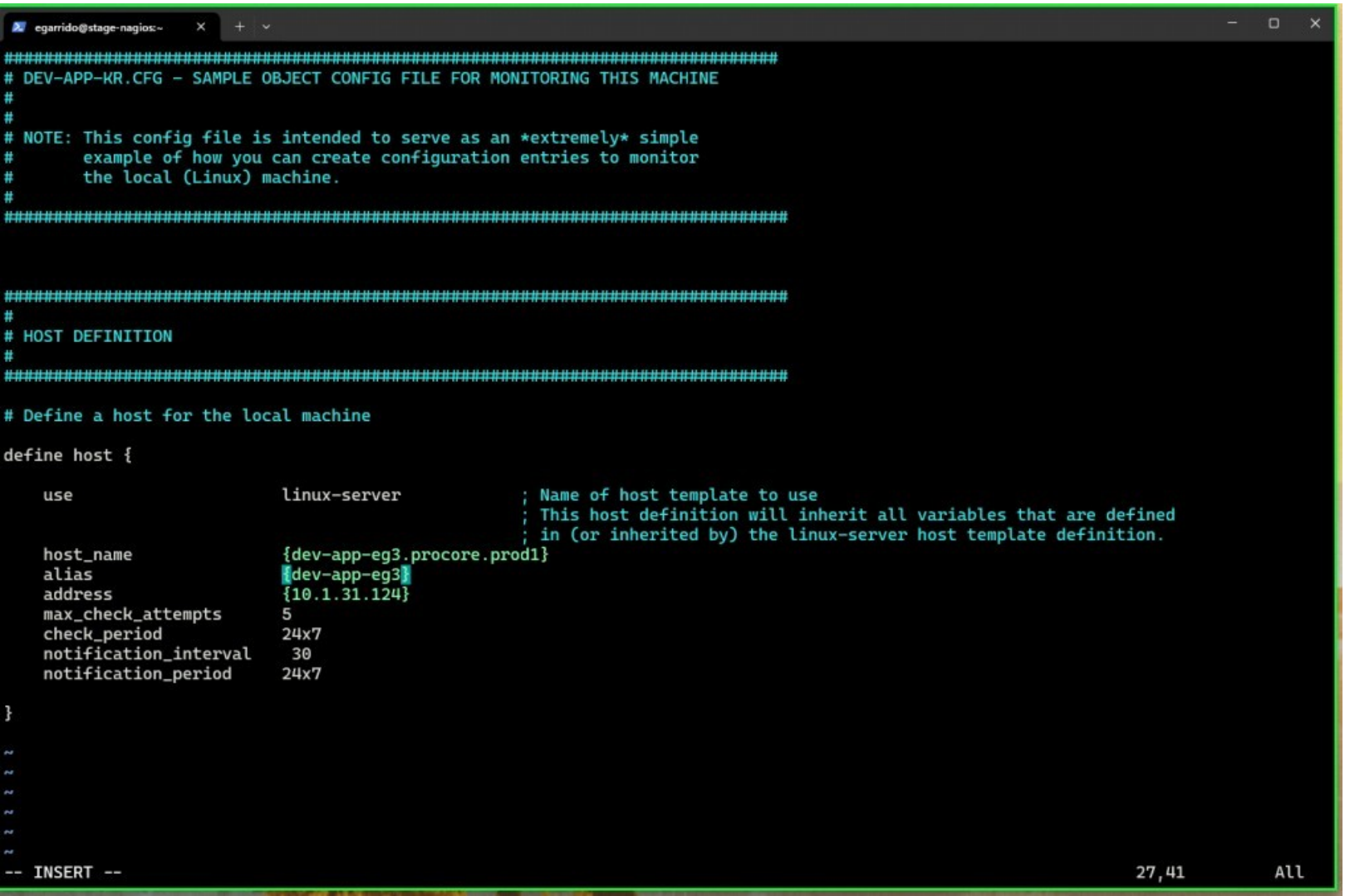
A directory listing confirms the presence of multiple Nagios object configuration files, including host definitions, service checks, contact definitions, and command configurations. Files follow a consistent naming convention that reflects environment, application role, and domain structure, supporting clarity and maintainability at scale.

The permissions and ownership displayed indicate that Nagios configuration files are properly secured while remaining editable by authorized users. This step demonstrates how new or updated application hosts are integrated into the Nagios monitoring framework by creating or modifying dedicated object configuration files.

Overall, this validates the host onboarding process within Nagios, ensuring monitored systems are accurately defined and aligned with existing monitoring standards and conventions.



A new Nagios host object definition is being created to onboard an application server into centralized monitoring. The configuration file defines a Linux-based host using the standard linux-server template, allowing the host to inherit common monitoring settings and thresholds.
The host is identified by the name dev-app-eg3.procore.prod1, with an alias for readability and an explicitly defined IP address. Monitoring parameters such as maximum check attempts, continuous check coverage (24x7), and notification intervals are configured to ensure consistent availability tracking and alerting behavior.
By using a dedicated host object file and inheriting from an existing template, this approach maintains consistency across environments while simplifying ongoing monitoring management. This step completes the host definition phase required before attaching service checks and NRPE-based monitoring to the system

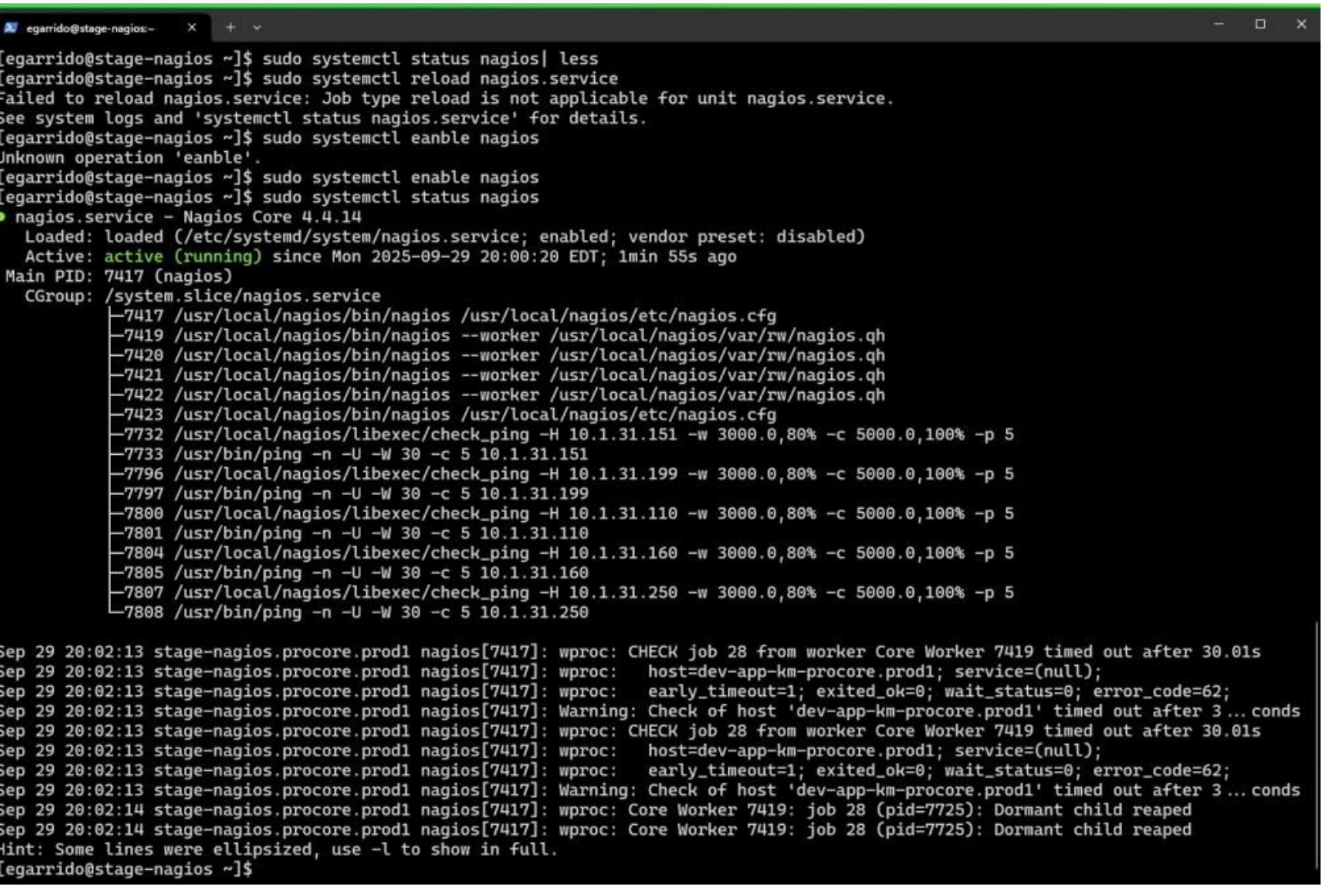


The Nagios monitoring service is validated and brought into a running state using systemd on the stage monitoring server. An initial attempt to reload the service indicates that the Nagios unit does not support a reload operation, requiring a full service enablement and start sequence instead.

Nagios is successfully enabled to start at boot and confirmed to be active (running). The service is loaded from /etc/systemd/system/nagios.service and is running version Nagios Core 4.4.14 under the main Nagios process, with multiple worker processes handling active checks concurrently.

The process tree shows active plugin executions, including ICMP reachability checks using check_ping, confirming that Nagios is actively monitoring configured hosts. Log output at the bottom reflects real-time worker activity, including completed and timed-out checks, which demonstrates that the monitoring engine is actively processing host and service checks.

This step confirms that Nagios is operational, enabled, and actively executing monitoring jobs, validating the successful integration of newly added host definitions into the monitoring system.

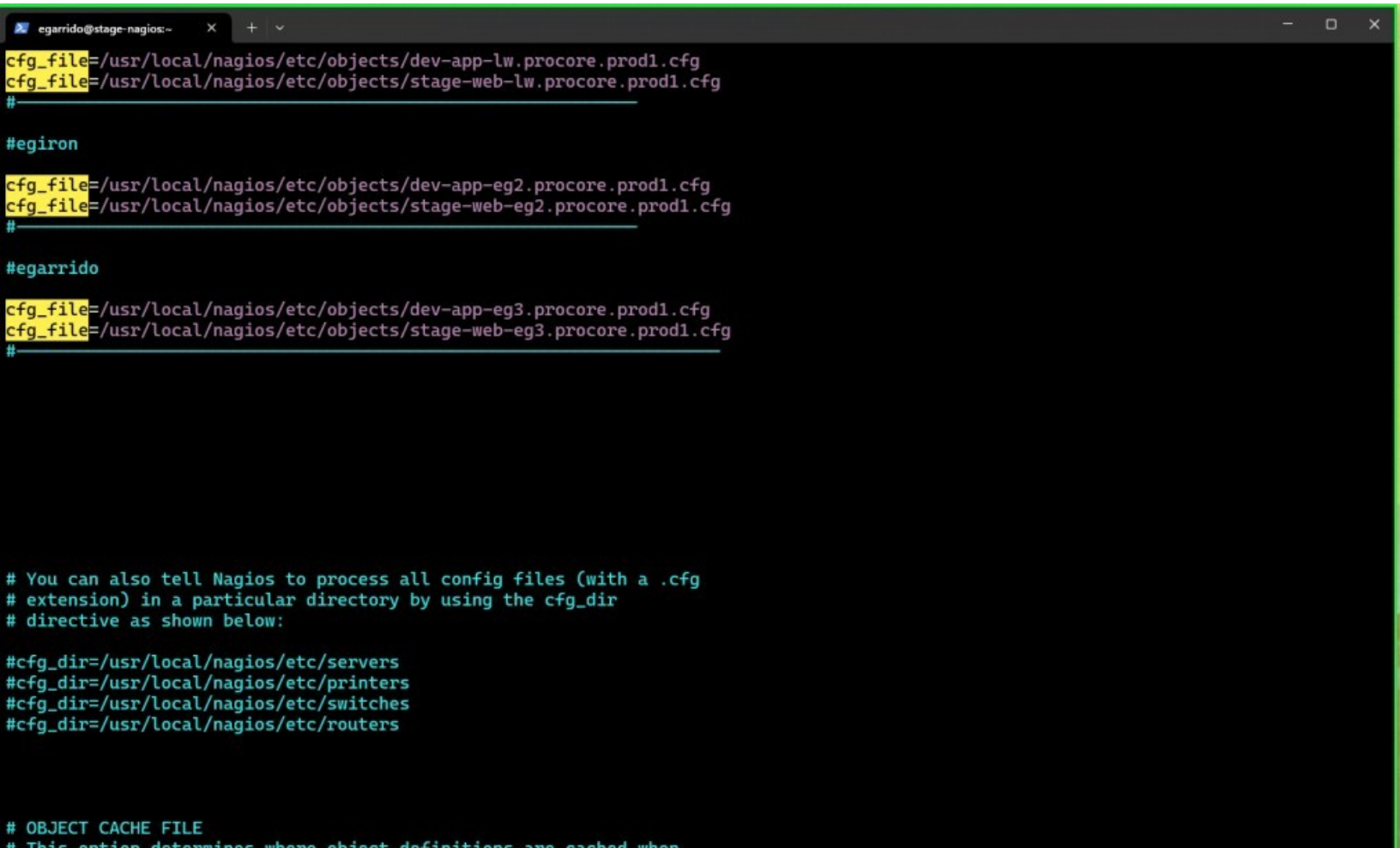


Nagios is configured to explicitly load host and service object definitions by referencing individual configuration files using the cfg_file directive. This approach provides precise control over which monitoring objects are included in the active configuration.

The configuration is logically organized by administrator or responsibility group, with comments separating related host definitions. New application and web hosts are added by appending their corresponding object files to the main Nagios configuration, ensuring they are recognized during startup or service restarts.

Each referenced file points to a dedicated object definition under /usr/local/nagios/etc/objects/, allowing hosts to be managed independently while maintaining a consistent naming convention across environments.

This step finalizes the Nagios onboarding process, ensuring newly defined hosts are included in the monitoring engine and actively evaluated once the service is running.

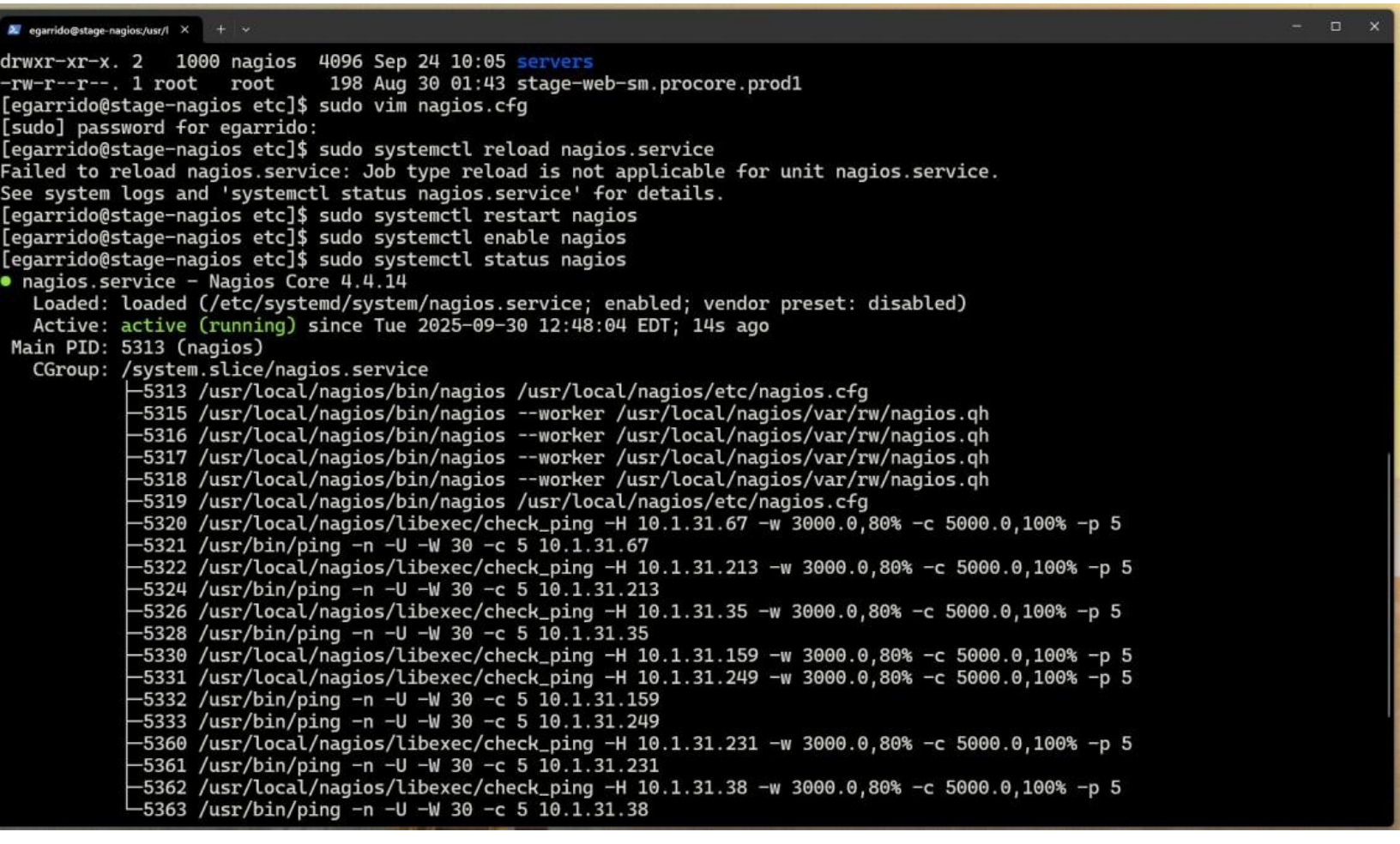


After updating the Nagios configuration, the monitoring service is restarted to apply the new object definitions. An attempted reload confirms that the Nagios systemd unit does not support reload operations, requiring a full service restart instead.

Nagios is successfully restarted and enabled to start automatically at boot. The service status output confirms Nagios Core 4.4.14 is active and running, using the primary configuration file located at /usr/local/nagios/etc/nagios.cfg.

The process tree shows multiple active worker processes and live plugin executions, including ICMP reachability checks via check_ping, indicating that host checks are being executed in real time across multiple monitored systems.

This validation confirms that the updated configuration has been successfully applied and that Nagios is fully operational, actively monitoring hosts, and processing checks as expected.

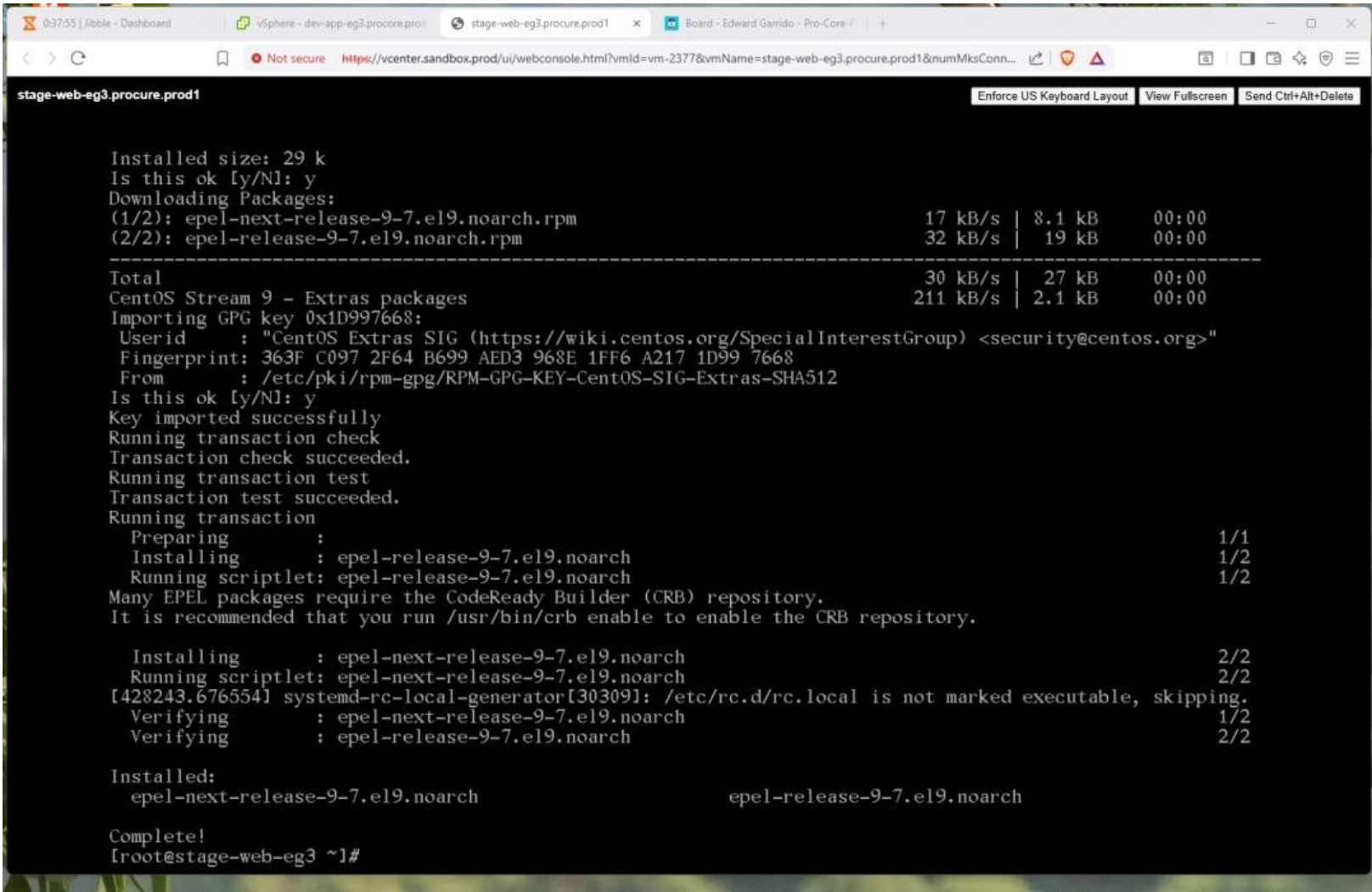


The EPEL (Extra Packages for Enterprise Linux) repositories are installed on a CentOS Stream 9 system to enable access to additional community-supported packages required for monitoring and tooling.

The system downloads and installs both epel-release and epel-next-release packages, importing the CentOS Extras SIG GPG key to ensure package authenticity. Transaction checks and tests complete successfully, confirming repository integrity before installation proceeds.

During installation, the system notes that some EPEL packages may depend on the CRB (CodeReady Builder) repository and recommends enabling it if required by future packages. The installation completes without errors, and the repositories are verified as installed.

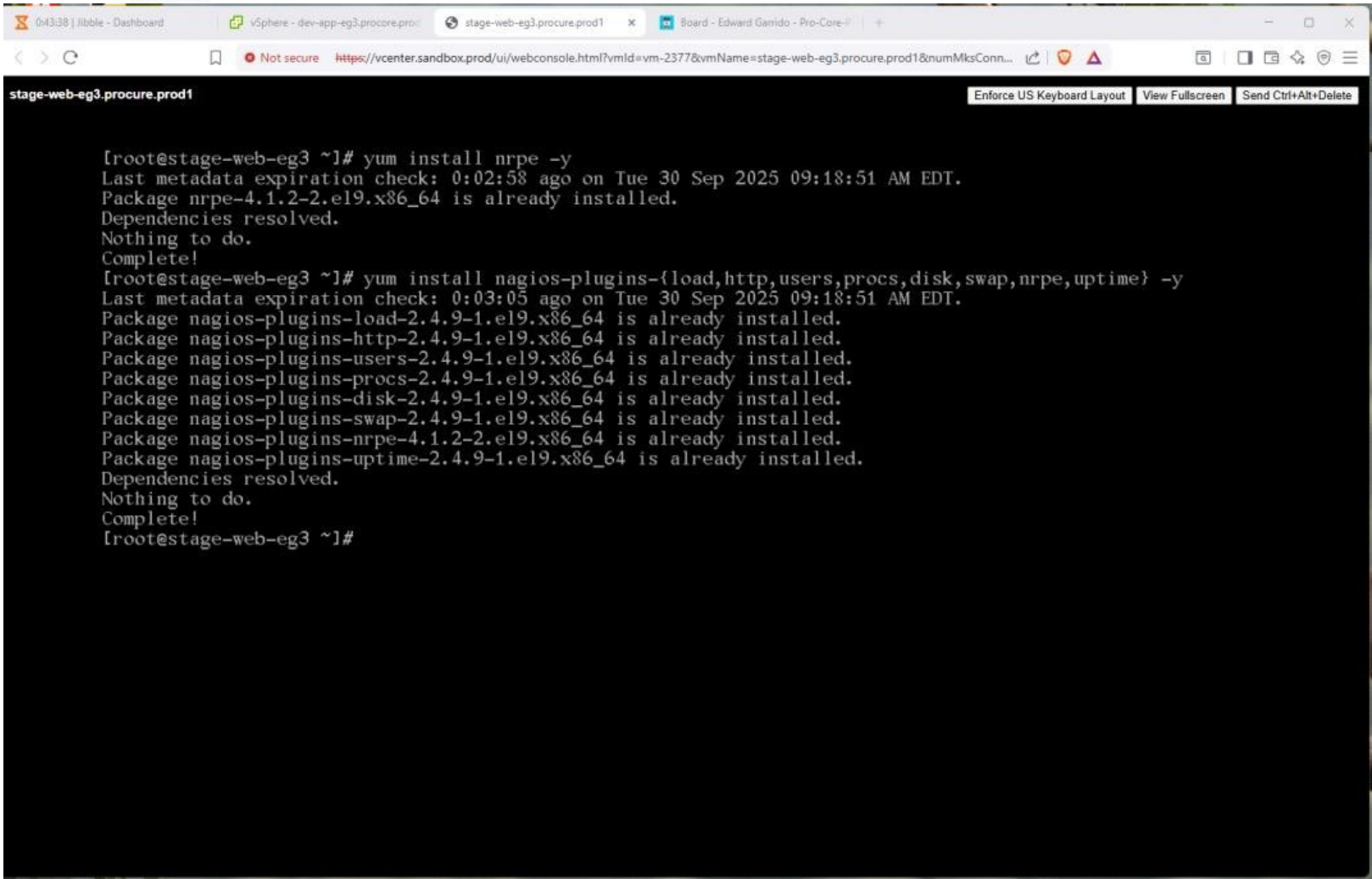
This step prepares the system for installing NRPE, Nagios plugins, and other EPEL-hosted dependencies, ensuring compatibility with enterprise monitoring workflows.



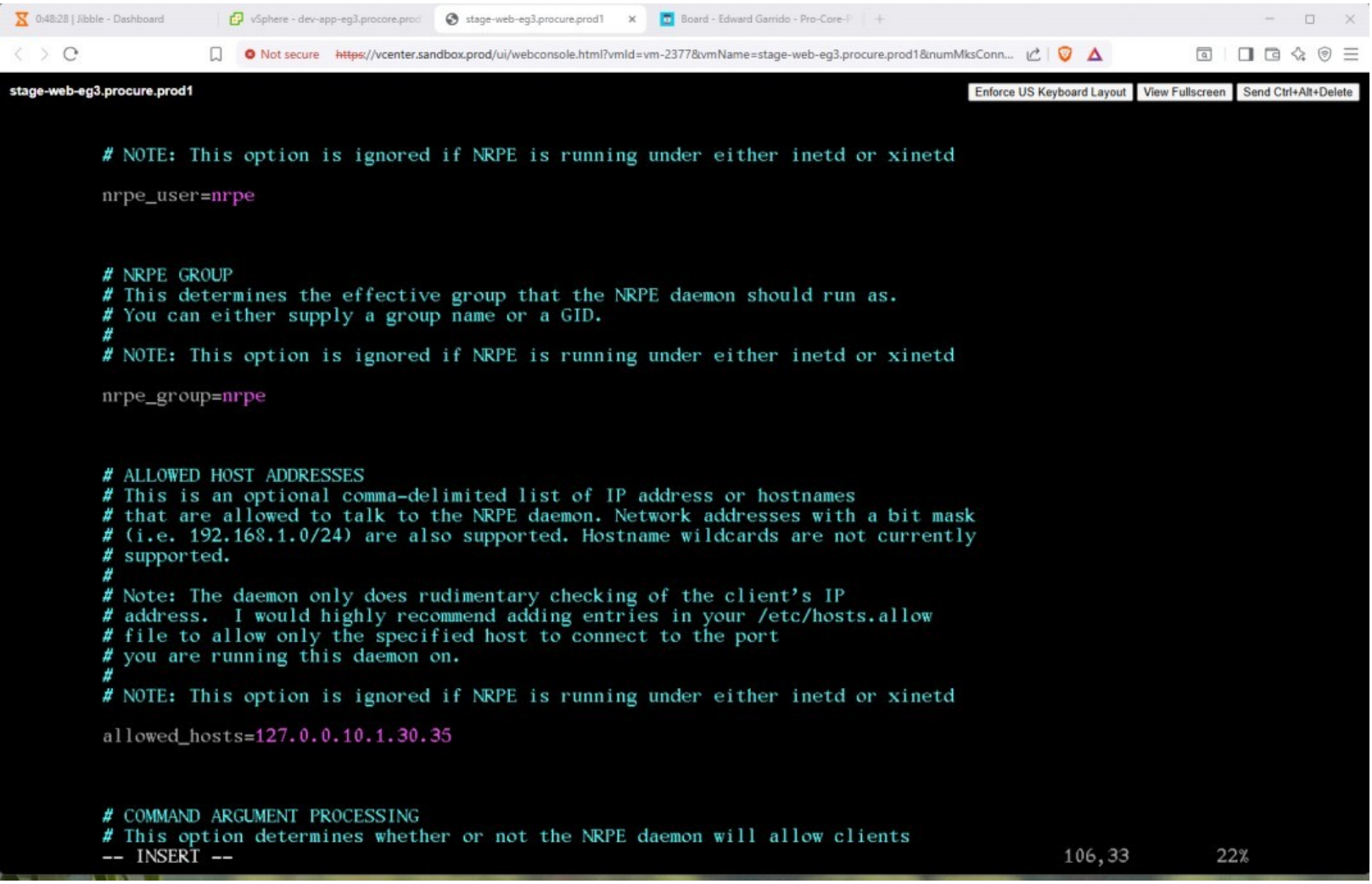
NRPE and the required Nagios monitoring plugins are verified on the application host to ensure readiness for remote monitoring. The system confirms that the NRPE package is already installed and up to date, requiring no further action.

A targeted set of Nagios plugins is then checked for availability, including load, HTTP, user sessions, process counts, disk usage, swap usage, NRPE checks, and system uptime. Each plugin is reported as already installed, confirming that the host has all necessary monitoring components in place.

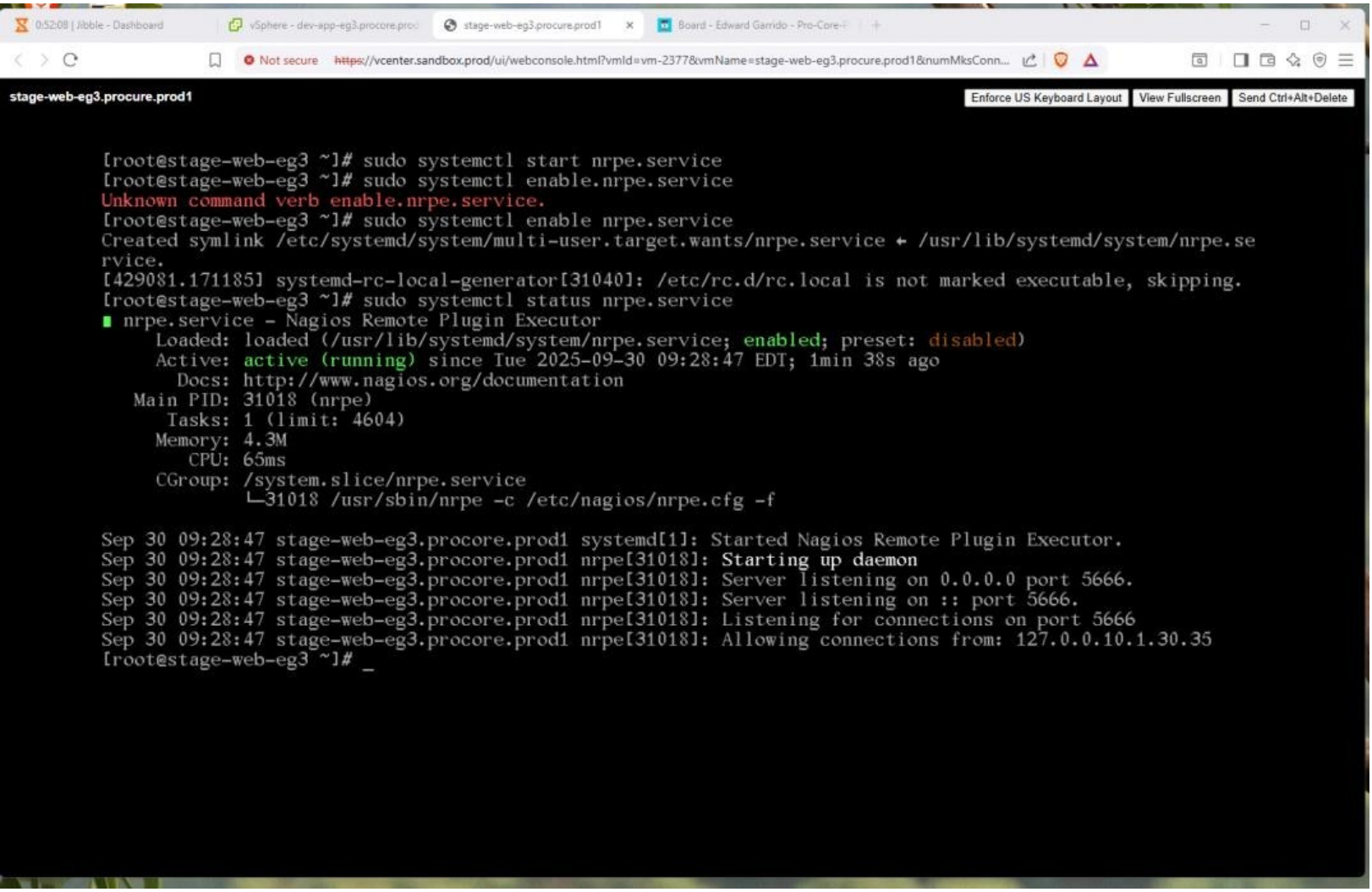
This verification step ensures the server is fully prepared to respond to NRPE-based health checks from the Nagios monitoring server, completing the agent-side monitoring setup without introducing redundant package changes.



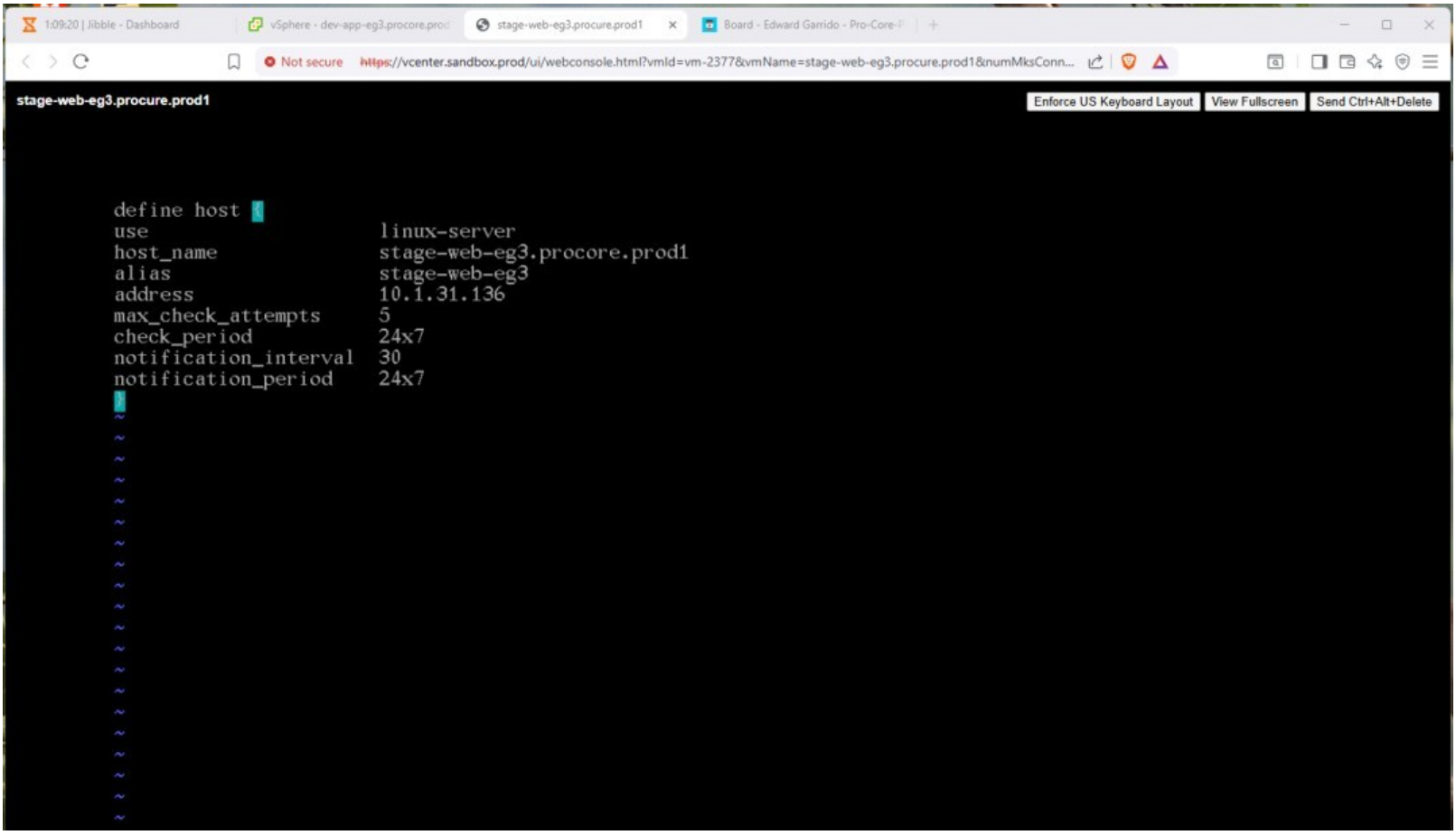
This screenshot shows the NRPE configuration file being edited on the host stage-web-eg3.procore.prod1. The configuration defines the NRPE runtime user and group (nrpe) and specifies the allowed host addresses permitted to communicate with the NRPE daemon. The allowed_hosts directive includes localhost and a designated monitoring server IP, ensuring that only authorized systems can execute remote Nagios checks. This step confirms proper access control and secure NRPE communication as part of the monitoring setup.



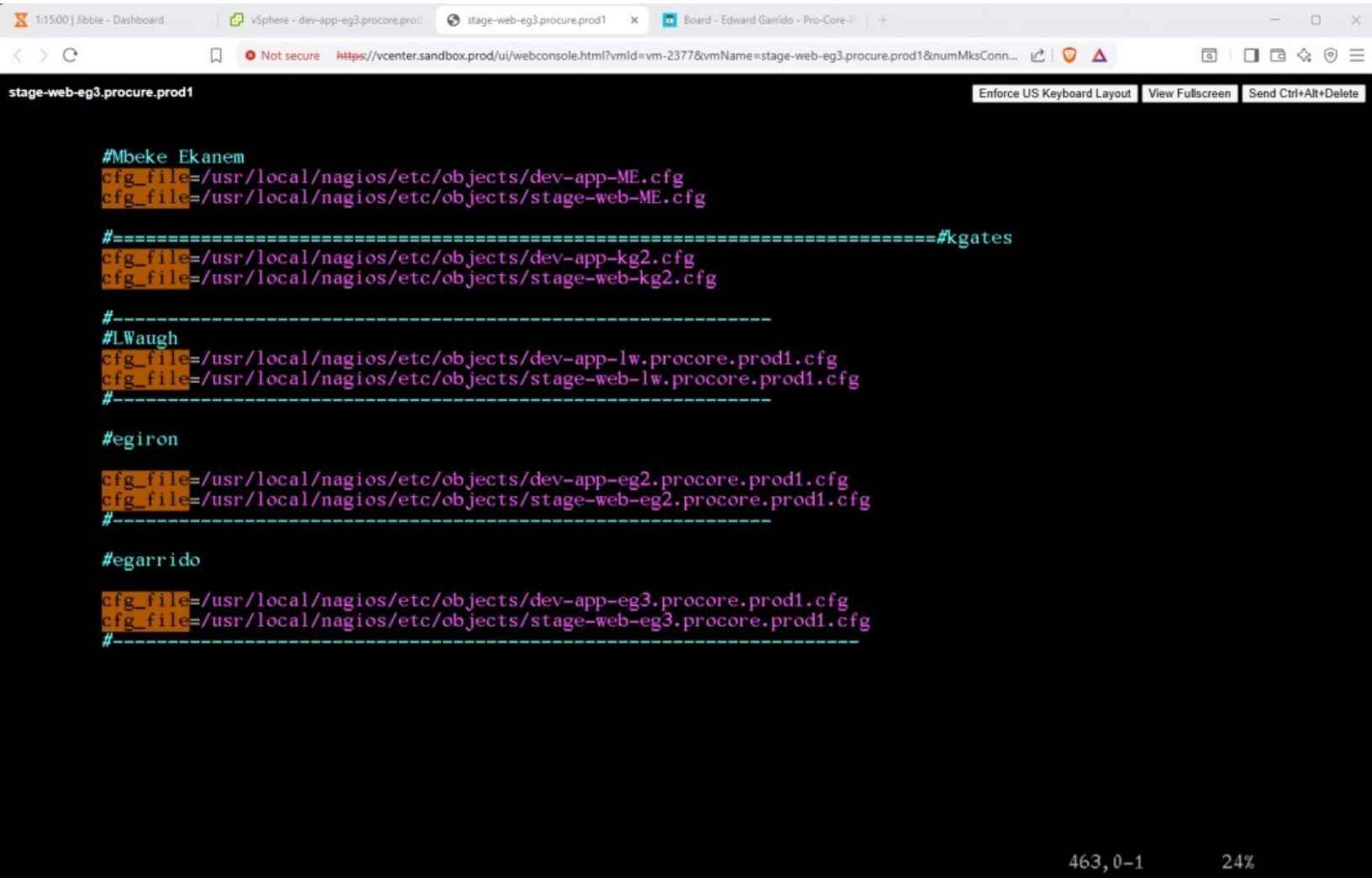
The NRPE service is started and enabled using systemctl, ensuring it runs automatically at boot. Service status output confirms that nrpe.service is active and running on stage-web-eg3.procore.prod1. The daemon is listening on the default NRPE port 5666 and explicitly accepting connections from the authorized monitoring hosts defined in the configuration file. This confirms a successful NRPE startup and validates readiness for remote Nagios health checks.



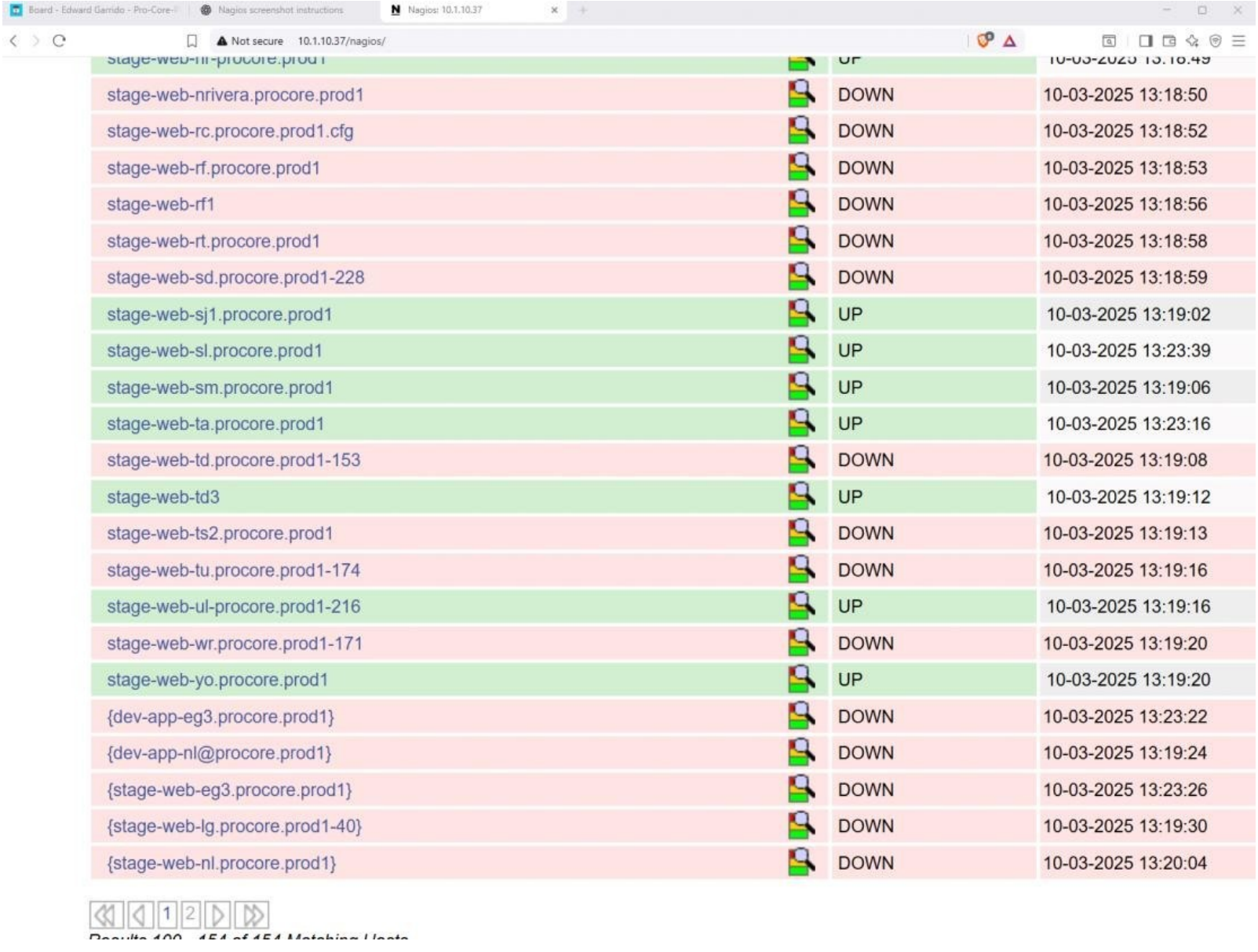
A new Nagios host definition is created for stage-web-eg3.procore.prod1, associating the system with the linux-server template. The configuration specifies the hostname, alias, and IP address, along with monitoring parameters such as check frequency, maximum retry attempts, and notification intervals. This ensures the host is properly registered in Nagios and continuously monitored with 24x7 availability and alerting enabled.



Nagios is configured to load multiple host object definition files by explicitly including them in the main configuration. Entries are grouped and commented for clarity, with separate object files defined for development and staging systems. The configuration includes the newly added stage-web-eg3.procore.prod1 host file, ensuring that the monitoring server recognizes the host and applies the appropriate checks and alerting policies after a configuration reload.



Nagios is configured to load multiple host object definition files by explicitly including them in the main configuration. Entries are grouped and commented for clarity, with separate object files defined for development and staging systems. The configuration includes the newly added stage-web-eg3.procore.prod1 host file, ensuring that the monitoring server recognizes the host and applies the appropriate checks and alerting policies after a configuration reload.



Summar

This work documents the successful integration of a staging web server into an existing Nagios monitoring environment using NRPE. The configuration includes installing required monitoring components, securing NRPE communication by restricting allowed hosts, enabling and validating the NRPE service, and defining Nagios host objects with appropriate monitoring and notification parameters. Final verification confirms that the monitoring server recognizes the host, executes remote checks successfully, and applies configuration changes as expected, demonstrating a complete and secure monitoring setup suitable for enterprise environments.