

# Linux VM Provisioning, Identity Integration, and Automation Readiness

8:06 PM

This project documents the complete lifecycle of building and validating a Linux virtual machine in an enterprise-style environment, from infrastructure provisioning to automation readiness.

The virtual machine is deployed on vSphere and configured with proper networking, routing, and hostname alignment to meet organizational standards. Network interfaces, IP configuration, gateways, and DNS resolution are verified to ensure stable connectivity and interoperability with centralized services.

The system is enrolled into a FreeIPA domain to enable centralized identity and access management. Core services including SSSD, Kerberos, LDAP, and SSH are configured and validated, allowing domain users to authenticate successfully and receive appropriate group memberships and sudo privileges. Local user management is also implemented where required, following least-privilege and security best practices.

Secure access is established using SSH key-based authentication, leveraging modern cryptographic keys. SSH daemon settings are reviewed, stale host keys are cleaned up, trust relationships are rebuilt, and passwordless access between hosts is verified. Static hostname mappings are applied when necessary to support controlled environments and troubleshooting scenarios.

An Ansible inventory is created and refined to represent multiple environments and roles. Initial connectivity issues are identified and resolved, and successful communication is confirmed using the ansible -m ping module. This validates that the environment is fully prepared for repeatable, reliable automation workflows.

Finally, the system is documented in an asset management platform, capturing ownership, configuration details, and lifecycle status to ensure traceability, accountability, and operational visibility.

## Security & Sanitization Notice

All IP addresses, hostnames, usernames, domain names, serial numbers, MAC addresses, timestamps, and environment-specific identifiers shown in screenshots, command output, and configuration files have been sanitized or obfuscated. No production credentials, sensitive infrastructure data, or real customer information are exposed in this repository.

This screenshot demonstrates SSH key distribution and bidirectional trust validation across multiple hosts in the environment. Public SSH keys are successfully copied between the Ansible control node and application, performance, and stage web servers using `ssh-copy-id`. Each key installation is followed by successful SSH logins, confirming passwordless authentication and trusted connectivity in both directions. This validates that all systems are correctly prepared for secure remote administration and Ansible-driven automation. All IP addresses, hostnames, usernames, timestamps, and environment-specific details shown are sanitized for security purposes.

```
egarrido@dev-ansible:~$ ssh-copy-id egarrido@dev-app-eg3.procure.prod1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/egarrido/.ssh/id_ed25519.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'egarrido@dev-app-eg3.procure.prod1'"
and check to make sure that only the key(s) you wanted were added.

[egarrido@dev-ansible ~]$ ssh egarrido@dev-app-eg3.procure.prod1
Last login: Thu Sep 25 20:16:49 2025 from 10.1.30.41
[egarrido@dev-app-eg3 ~]$ ssh egarrido@dev-ansible.procure.prod1
(egarrido@dev-ansible.procure.prod1) Password:

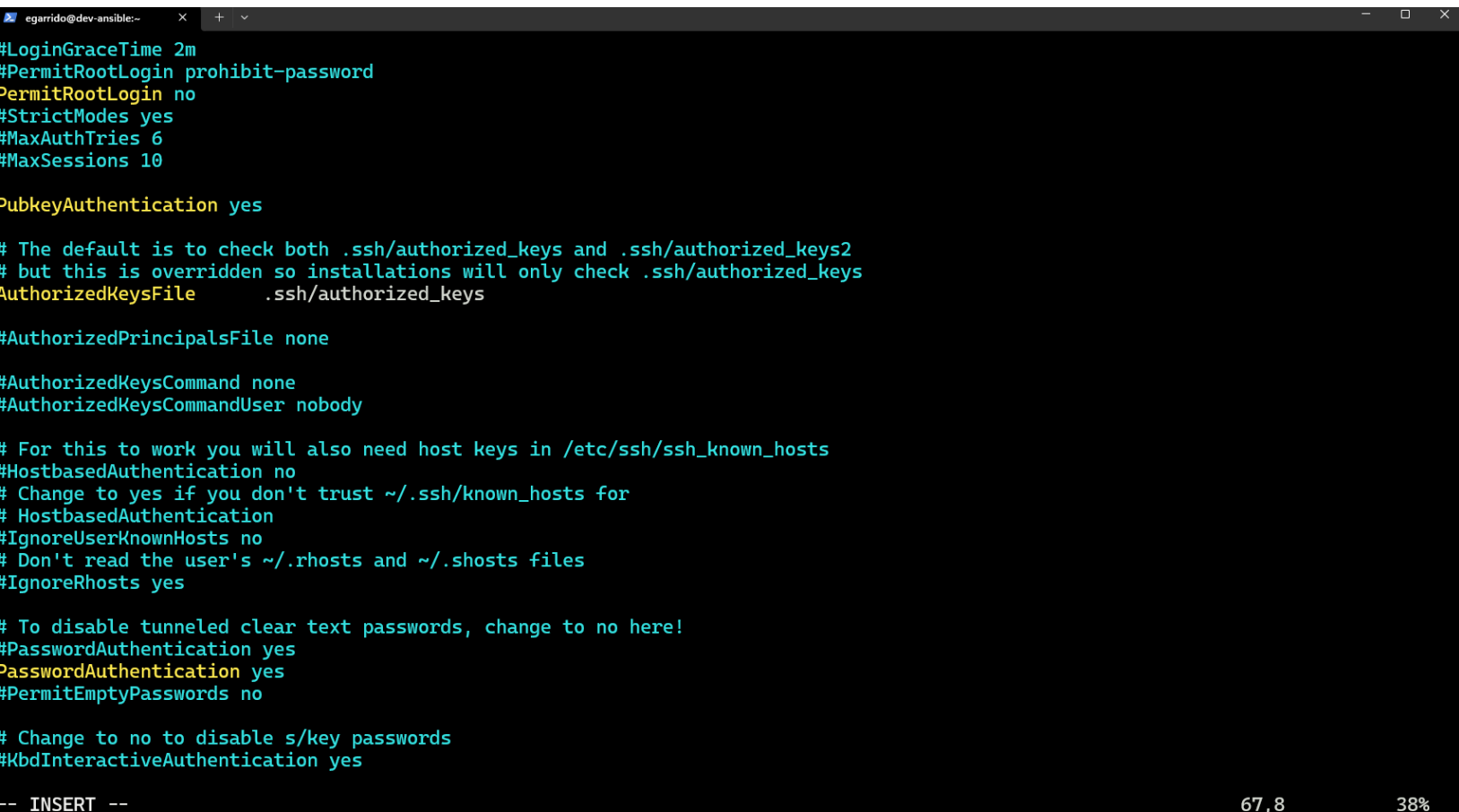
[egarrido@dev-app-eg3 ~]$ ssh-copy-id egarrido@dev-ansible.procure.prod1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/egarrido/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
(egarrido@dev-ansible.procure.prod1) Password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'egarrido@dev-ansible.procure.prod1'"
and check to make sure that only the key(s) you wanted were added.

[egarrido@dev-app-eg3 ~]$ ssh egarrido@dev-ansible.procure.prod1
Last login: Thu Sep 25 20:17:49 2025 from 10.1.31.124
[egarrido@dev-ansible ~]$ ssh egarrido@dev-performance-eg3.procure.prod1
Last login: Thu Sep 25 20:15:32 2025 from 10.1.10.112
[egarrido@dev-performance-eg3 ~]$ ssh egarrido@dev-ansible.procure.prod1
Last login: Thu Sep 25 20:22:20 2025 from 10.1.31.124
[egarrido@dev-ansible ~]$ ssh egarrido@stage-web-eg3.procure.prod1
Last login: Thu Sep 25 16:14:47 2025 from 10.1.30.41
[egarrido@stage-web-eg3 ~]$ ssh egarrido@dev-ansible.procure.prod1
Last login: Thu Sep 25 20:23:38 2025 from 10.1.31.135
[egarrido@dev-ansible ~]$
```

This screenshot demonstrates SSH key distribution and bidirectional trust validation across multiple hosts in the environment. Public SSH keys are successfully copied between the Ansible control node and application, performance, and stage web servers using ssh-copy-id. Each key installation is followed by successful SSH logins, confirming passwordless authentication and trusted connectivity in both directions. This validates that all systems are correctly prepared for secure remote administration and Ansible-driven automation. All IP addresses, hostnames, usernames, timestamps, and environment-specific details shown are sanitized for security purposes.

A terminal window with a dark background and light-colored text. The window title is 'egarrido@dev-ansible:~'. The output shows the contents of the /etc/ssh/sshd\_config file, with some lines highlighted in yellow. The configuration includes settings for LoginGraceTime, PermitRootLogin, StrictModes, MaxAuthTries, MaxSessions, PubkeyAuthentication, AuthorizedKeysFile, AuthorizedPrincipalsFile, AuthorizedKeysCommand, AuthorizedKeysCommandUser, HostbasedAuthentication, IgnoreUserKnownHosts, IgnoreRhosts, PasswordAuthentication, PermitEmptyPasswords, and KbdInteractiveAuthentication. The output ends with '-- INSERT --'.

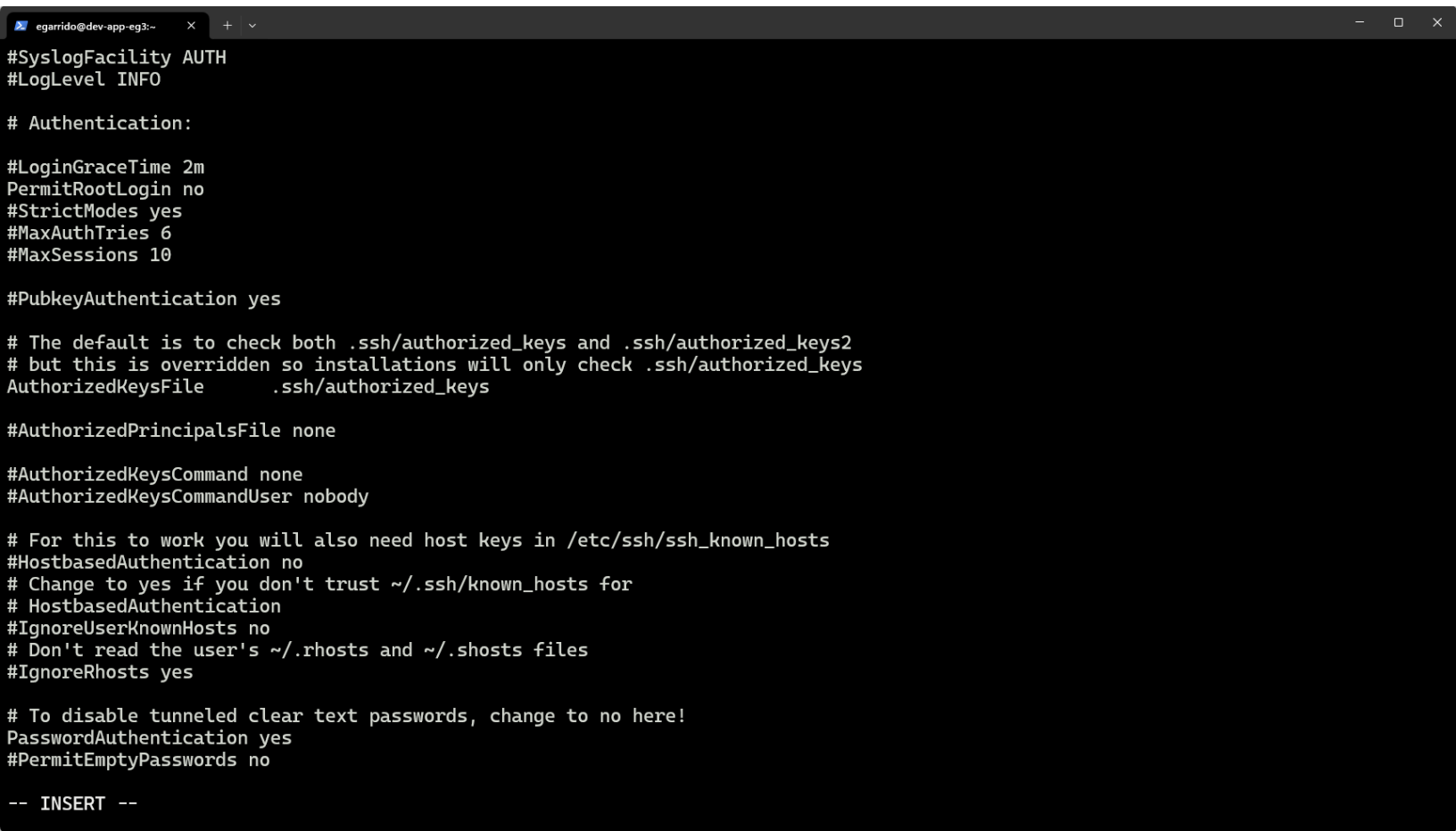
```
egarrido@dev-ansible:~  
#LoginGraceTime 2m  
#PermitRootLogin prohibit-password  
PermitRootLogin no  
#StrictModes yes  
#MaxAuthTries 6  
#MaxSessions 10  
  
PubkeyAuthentication yes  
  
# The default is to check both .ssh/authorized_keys and .ssh/authorized_keys2  
# but this is overridden so installations will only check .ssh/authorized_keys  
AuthorizedKeysFile .ssh/authorized_keys  
  
#AuthorizedPrincipalsFile none  
  
#AuthorizedKeysCommand none  
#AuthorizedKeysCommandUser nobody  
  
# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts  
#HostbasedAuthentication no  
# Change to yes if you don't trust ~/.ssh/known_hosts for  
# HostbasedAuthentication  
#IgnoreUserKnownHosts no  
# Don't read the user's ~/.rhosts and ~/.shosts files  
#IgnoreRhosts yes  
  
# To disable tunneled clear text passwords, change to no here!  
#PasswordAuthentication yes  
PasswordAuthentication yes  
#PermitEmptyPasswords no  
  
# Change to no to disable s/key passwords  
#KbdInteractiveAuthentication yes  
  
-- INSERT --
```

This screenshot shows SSH daemon configuration validation on the Ansible control host. After reviewing or modifying `/etc/ssh/sshd_config`, the SSH service is restarted and its status is verified using `systemctl`. The output confirms that the OpenSSH server is enabled at boot, actively running, and listening on the expected port, indicating that SSH access is stable and properly configured for remote administration and automation tasks. All hostnames, usernames, timestamps, and environment-specific details shown are sanitized for security purposes.

```
egarrido@dev-ansible:~$ sudo vi /etc/ssh/sshd_config
egarrido@dev-ansible:~$ sudo systemctl restart sshd
[sudo] password for egarrido:
egarrido@dev-ansible:~$ sudo systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; preset: enabled)
   Active: active (running) since Thu 2025-09-25 21:04:11 EDT; 12s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Main PID: 521336 (sshd)
    Tasks: 1 (limit: 48871)
  Memory: 1.5M (peak: 1.8M)
     CPU: 16ms
   CGroup: /system.slice/sshd.service
           └─521336 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Sep 25 21:04:11 dev-ansible systemd[1]: Starting OpenSSH server daemon ...
Sep 25 21:04:11 dev-ansible sshd[521336]: Server listening on 0.0.0.0 port 22.
Sep 25 21:04:11 dev-ansible sshd[521336]: Server listening on :: port 22.
Sep 25 21:04:11 dev-ansible systemd[1]: Started OpenSSH server daemon.
egarrido@dev-ansible:~$
```

This screenshot shows a hardened OpenSSH server authentication configuration being edited in `sshd_config`. Root login is explicitly disabled, public key authentication is enabled, and secure defaults are enforced for session handling and authorization files. Password authentication settings are clearly defined, and legacy or insecure mechanisms such as host-based authentication and `rhosts` files are disabled. The file is open in INSERT mode, indicating active configuration to align SSH access with enterprise security best practices. All hostnames, usernames, and environment-specific details shown are sanitized for security purposes.

A screenshot of a terminal window with a dark background. The window title bar shows 'egarrido@dev-app-eg3:~' and standard window controls. The terminal displays the configuration of the `sshd_config` file. The text is as follows:

```
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# The default is to check both .ssh/authorized_keys and .ssh/authorized_keys2
# but this is overridden so installations will only check .ssh/authorized_keys
AuthorizedKeysFile      .ssh/authorized_keys

#AuthorizedPrincipalsFile none

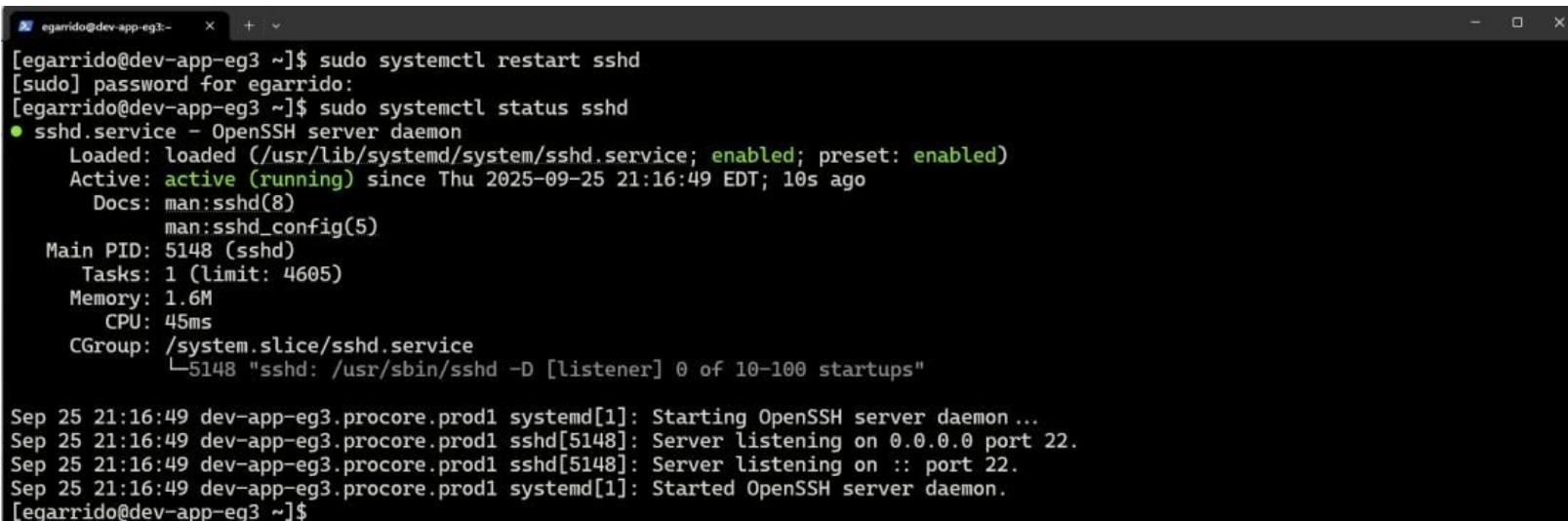
#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no

-- INSERT --
```

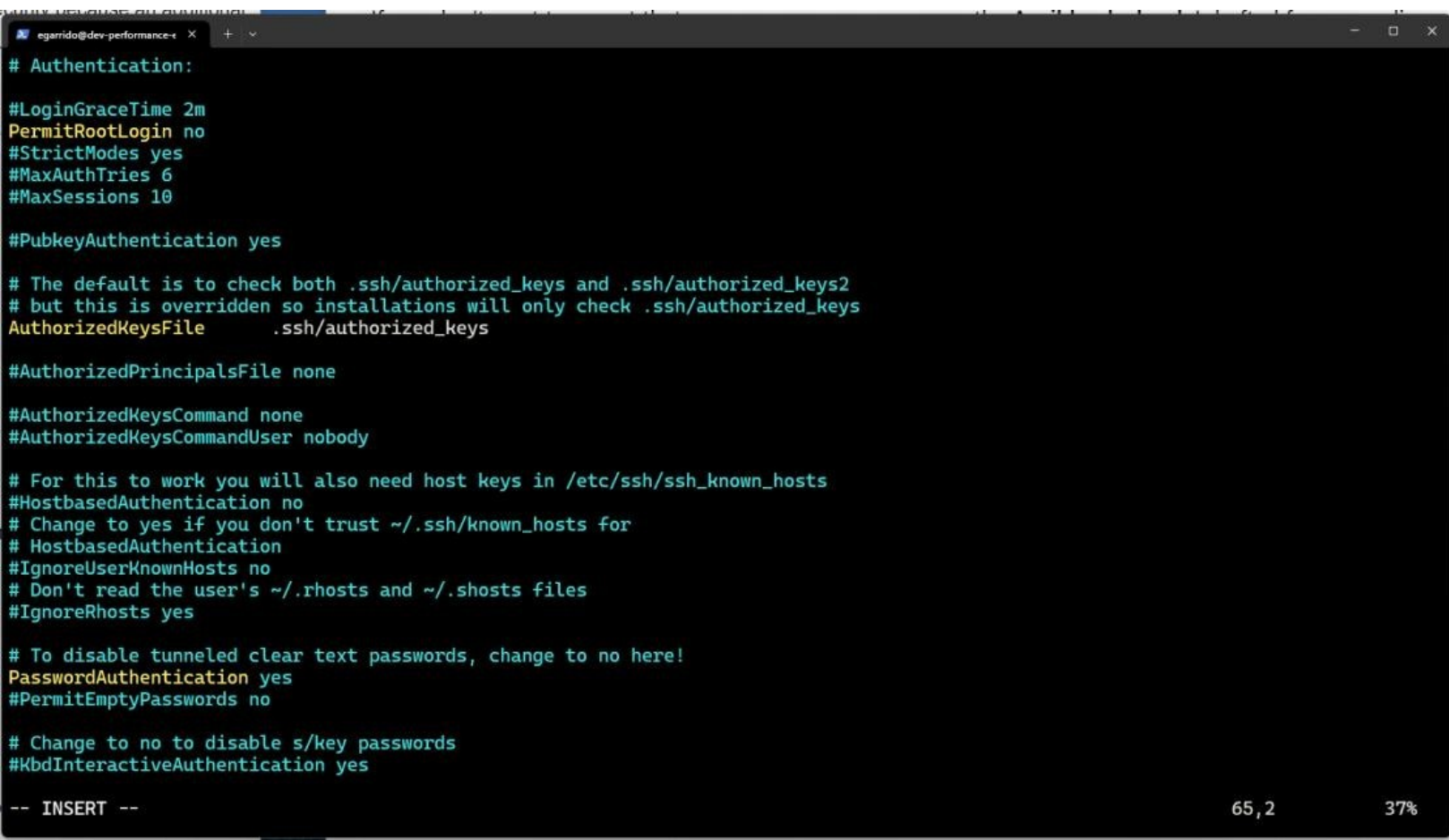
This screenshot confirms successful SSH daemon restart and validation on the application server. After restarting the sshd service, its status is checked with systemctl, showing the service is enabled, actively running, and listening on the expected port. The log output confirms a clean startup with no errors, verifying that recent SSH configuration changes were applied successfully and secure remote access remains available. All hostnames, usernames, timestamps, and environment-specific details shown are sanitized for security purposes.

A terminal window with a dark background and light text. The window title is 'egarrido@dev-app-eg3-'. The user enters 'sudo systemctl restart sshd', followed by a password prompt. Then they enter 'sudo systemctl status sshd'. The output shows the service is 'active (running)' and 'enabled'. Below this, detailed service information is shown, including the main PID (5148), tasks (1), memory (1.6M), CPU (45ms), and CGroup. At the bottom, system logs show the service starting and listening on port 22.

```
[egarrido@dev-app-eg3 ~]$ sudo systemctl restart sshd
[sudo] password for egarrido:
[egarrido@dev-app-eg3 ~]$ sudo systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; preset: enabled)
   Active: active (running) since Thu 2025-09-25 21:16:49 EDT; 10s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
   Main PID: 5148 (sshd)
    Tasks: 1 (limit: 4605)
   Memory: 1.6M
      CPU: 45ms
   CGroup: /system.slice/sshd.service
           └─5148 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Sep 25 21:16:49 dev-app-eg3.procore.prod1 systemd[1]: Starting OpenSSH server daemon ...
Sep 25 21:16:49 dev-app-eg3.procore.prod1 sshd[5148]: Server listening on 0.0.0.0 port 22.
Sep 25 21:16:49 dev-app-eg3.procore.prod1 sshd[5148]: Server listening on :: port 22.
Sep 25 21:16:49 dev-app-eg3.procore.prod1 systemd[1]: Started OpenSSH server daemon.
[egarrido@dev-app-eg3 ~]$
```

This screenshot shows the OpenSSH authentication configuration on the performance server being reviewed and edited in `sshd_config`. Key hardening measures are visible, including disabling root login, enforcing strict modes, limiting authentication attempts and sessions, and defining authorized key handling. Password authentication settings are explicitly controlled, and legacy or insecure mechanisms are disabled to align with security best practices. The file is open in INSERT mode, indicating active configuration. All hostnames, usernames, and environment-specific details shown are sanitized for security purposes.



```
egarrido@dev-performance:~$ vi /etc/ssh/sshd_config
# Authentication:
#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# The default is to check both .ssh/authorized_keys and .ssh/authorized_keys2
# but this is overridden so installations will only check .ssh/authorized_keys
AuthorizedKeysFile .ssh/authorized_keys

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no

# Change to no to disable s/key passwords
#KbdInteractiveAuthentication yes

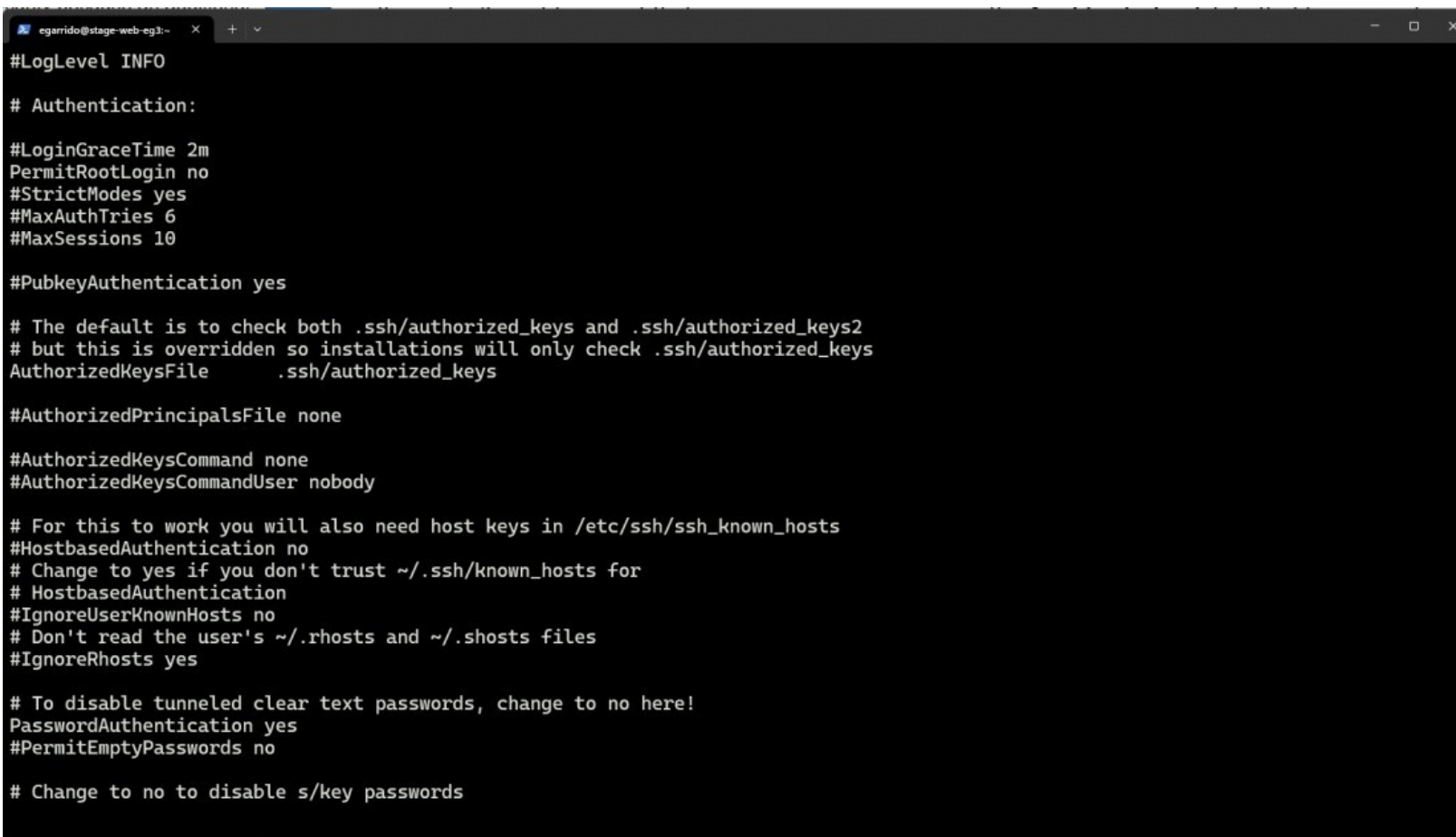
-- INSERT --
```

This screenshot confirms successful SSH configuration and service validation on the performance server. After editing the OpenSSH configuration file, the sshd service is restarted and its status is checked using systemctl. The output shows the service is enabled, actively running, and listening on the expected port with no errors, confirming that the updated SSH hardening settings were applied correctly and secure remote access remains available. All hostnames, usernames, timestamps, and environment-specific details shown are sanitized for security purposes.

```
egarrido@dev-performance-eg3 ~$ sudo vi /etc/ssh/sshd_config
[sudo] password for egarrido:
egarrido@dev-performance-eg3 ~$ sudo systemctl restart sshd
egarrido@dev-performance-eg3 ~$ sudo systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; preset: enabled)
   Active: active (running) since Thu 2025-09-25 21:12:58 EDT; 13s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Main PID: 7968 (sshd)
    Tasks: 1 (limit: 4605)
   Memory: 1.6M
      CPU: 46ms
   CGroup: /system.slice/sshd.service
           └─7968 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Sep 25 21:12:58 dev-performance-eg3.procore.prod1 systemd[1]: Starting OpenSSH server daemon ...
Sep 25 21:12:58 dev-performance-eg3.procore.prod1 sshd[7968]: Server listening on 0.0.0.0 port 22.
Sep 25 21:12:58 dev-performance-eg3.procore.prod1 sshd[7968]: Server listening on :: port 22.
Sep 25 21:12:58 dev-performance-eg3.procore.prod1 systemd[1]: Started OpenSSH server daemon.
egarrido@dev-performance-eg3 ~$
```

This screenshot shows the OpenSSH authentication configuration on the stage web server being reviewed or edited in `sshd_config`. The configuration reflects standard SSH hardening practices, including disabling root login, enforcing strict permission checks, limiting authentication attempts and sessions, and defining the authorized keys file for public key authentication. Legacy and insecure mechanisms such as host-based authentication and `rhosts` files are disabled, and password authentication behavior is explicitly controlled. These settings help ensure secure, controlled SSH access across the environment. All hostnames, usernames, and environment-specific details shown are sanitized for security purposes.



```
# egarrido@stage-web-eg3:~
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# The default is to check both .ssh/authorized_keys and .ssh/authorized_keys2
# but this is overridden so installations will only check .ssh/authorized_keys
AuthorizedKeysFile      .ssh/authorized_keys

#AuthorizedPrincipalsFile none

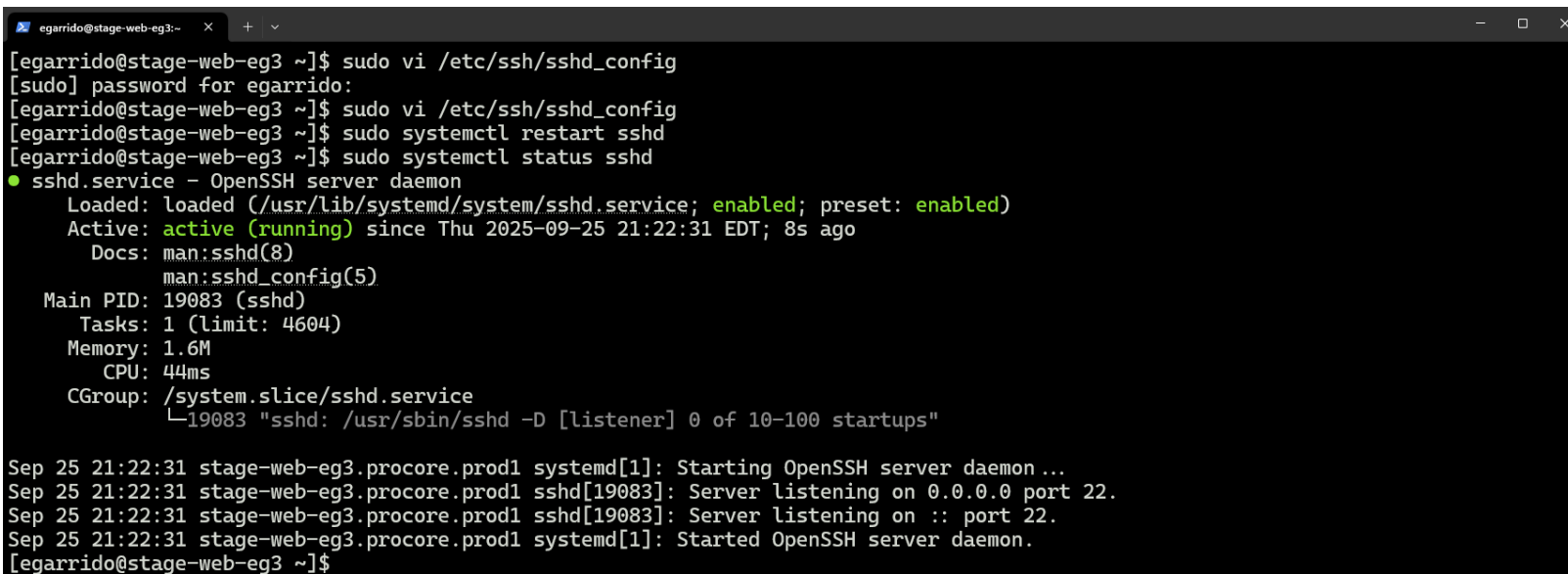
#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no

# Change to no to disable s/key passwords
```

This screenshot confirms successful SSH configuration and service validation on the stage web server. After reviewing or updating the OpenSSH configuration file, the sshd service is restarted and its status is verified using systemctl. The output shows the service is enabled, actively running, and listening on the expected port with no errors, confirming that the SSH hardening settings were applied correctly and secure remote access is functioning as intended. All hostnames, usernames, timestamps, and environment-specific details shown are sanitized for security purposes.

A terminal window titled 'egarrido@stage-web-eg3:~' showing a series of commands and their outputs. The user runs 'sudo vi /etc/ssh/sshd\_config', followed by 'sudo systemctl restart sshd', and finally 'sudo systemctl status sshd'. The status output shows that the 'sshd.service' is loaded, active (running), and enabled. It also displays resource usage like memory (1.6M) and CPU (44ms). At the bottom, there are four log entries from 'systemd' and 'sshd' confirming the daemon's start and listening on port 22.

```
egarrido@stage-web-eg3:~$ sudo vi /etc/ssh/sshd_config
[sudo] password for egarrido:
egarrido@stage-web-eg3:~$ sudo vi /etc/ssh/sshd_config
egarrido@stage-web-eg3:~$ sudo systemctl restart sshd
egarrido@stage-web-eg3:~$ sudo systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; preset: enabled)
   Active: active (running) since Thu 2025-09-25 21:22:31 EDT; 8s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Main PID: 19083 (sshd)
    Tasks: 1 (limit: 4604)
   Memory: 1.6M
      CPU: 44ms
   CGroup: /system.slice/sshd.service
           └─19083 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Sep 25 21:22:31 stage-web-eg3.procore.prod1 systemd[1]: Starting OpenSSH server daemon ...
Sep 25 21:22:31 stage-web-eg3.procore.prod1 sshd[19083]: Server listening on 0.0.0.0 port 22.
Sep 25 21:22:31 stage-web-eg3.procore.prod1 sshd[19083]: Server listening on :: port 22.
Sep 25 21:22:31 stage-web-eg3.procore.prod1 systemd[1]: Started OpenSSH server daemon.
egarrido@stage-web-eg3:~$
```

## Summary

This project showcases the end-to-end provisioning and validation of a Linux virtual machine in an enterprise environment. It covers network and hostname configuration, FreeIPA-based centralized identity management, secure SSH key-based access, and Ansible connectivity testing to ensure automation readiness. The system is deployed on vSphere, documented in an asset management platform, and verified for reliable operation. All IP addresses, hostnames, usernames, serial numbers, and environment-specific details shown are sanitized for security purposes.