Ansible Directory Management Automation

This repository demonstrates the use of Ansible automation to standardize directory creation and permissions across multiple Linux hosts in a development and staging environment. The playbook ensures that a user-specific directory is consistently created under /opt/scripts on all targeted systems.

The automation leverages Ansible variables to dynamically create directories based on the executing user, while enforcing correct ownership, group assignment, and permissions. Syntax validation was performed prior to execution to ensure proper YAML structure and reliable deployment.

The playbook was executed with privilege escalation and verified across multiple hosts, confirming successful and repeatable results. Post-execution validation shows that the desired directory structure exists uniformly across environments, demonstrating idempotent behavior and configuration consistency.

This project highlights practical skills in:

Writing and validating Ansible playbooks

Using privilege escalation securely (become)

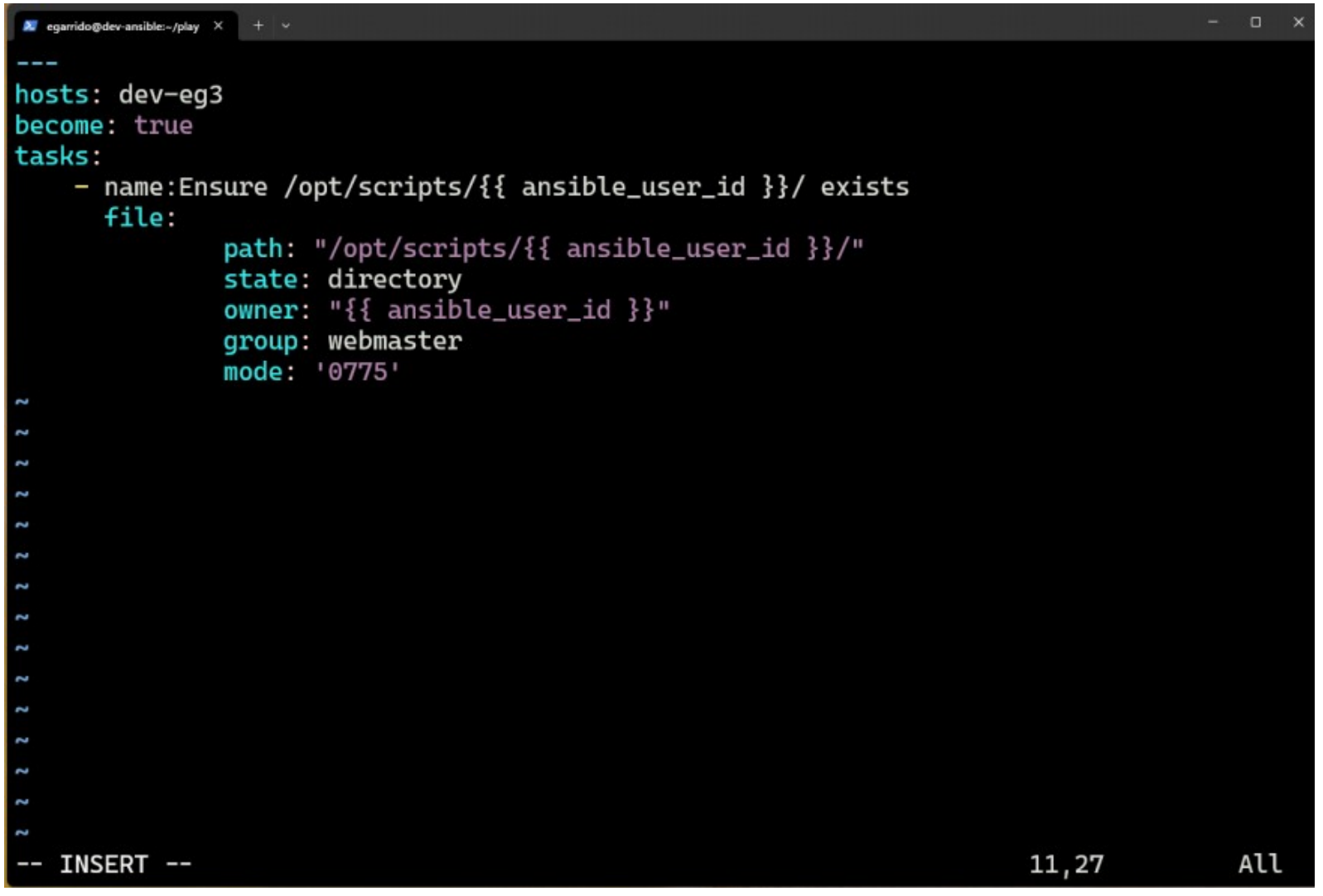
Managing file system state across multiple hosts

Enforcing permissions, ownership, and group policies

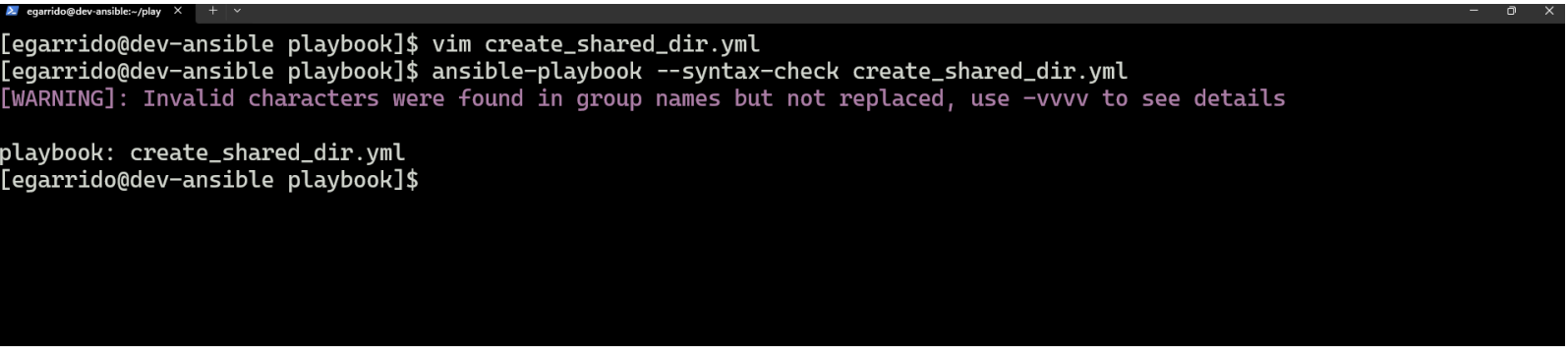
Verifying results through manual and automated checks

Overall, this repository reflects real-world infrastructure automation practices commonly used in enterprise Linux and DevOps environments.

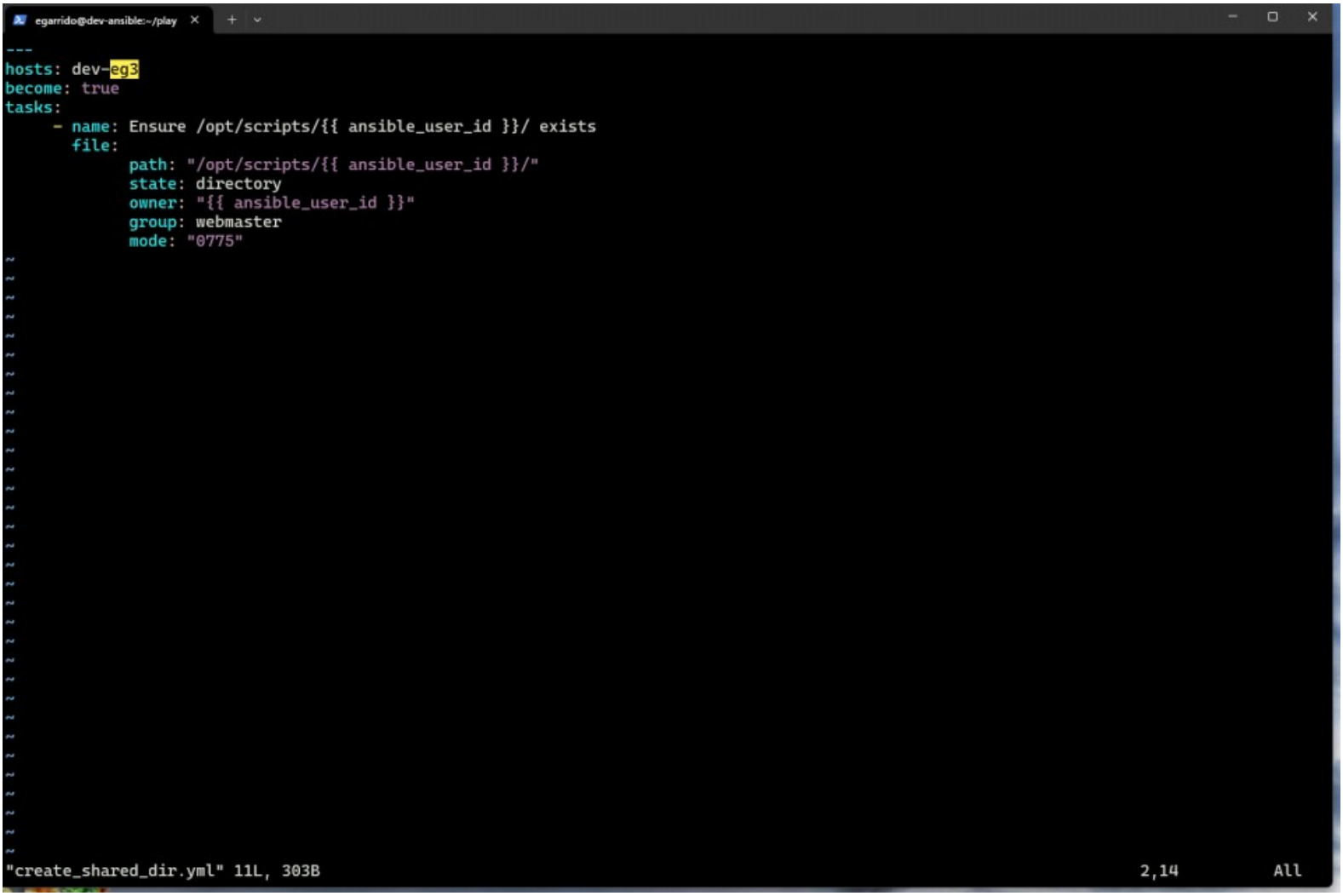
An Ansible playbook was created to ensure a standardized directory structure under /opt/scripts for the executing user across the dev-eg3 host group. The task enforces directory existence, correct ownership, group assignment, and permissions using Ansible variables, demonstrating idempotent configuration management and secure privilege escalation.



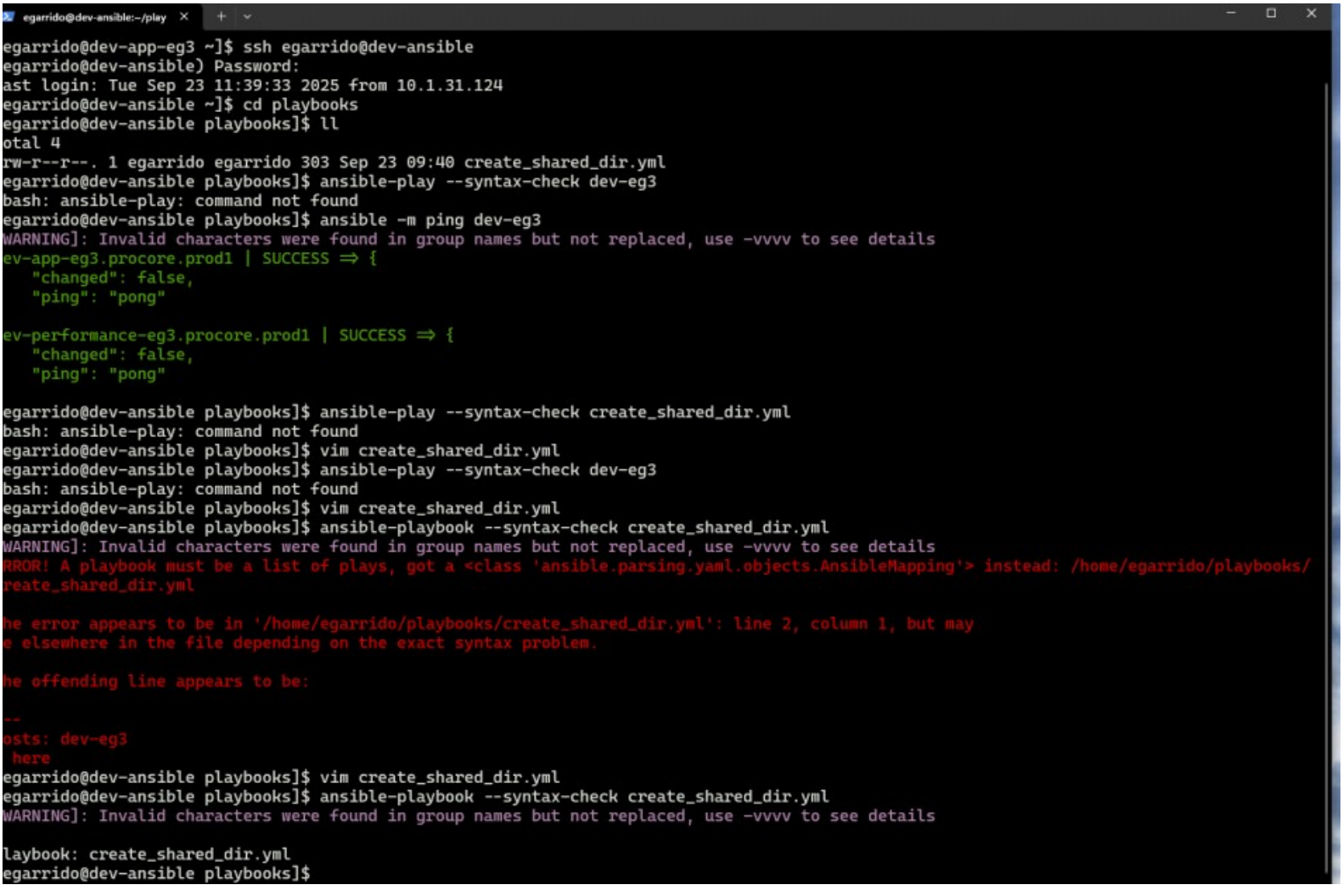
The Ansible playbook was validated using ansible-playbook --syntax-check, confirming correct YAML structure and task definitions. The syntax check completed successfully, indicating the playbook does not have a syntax error



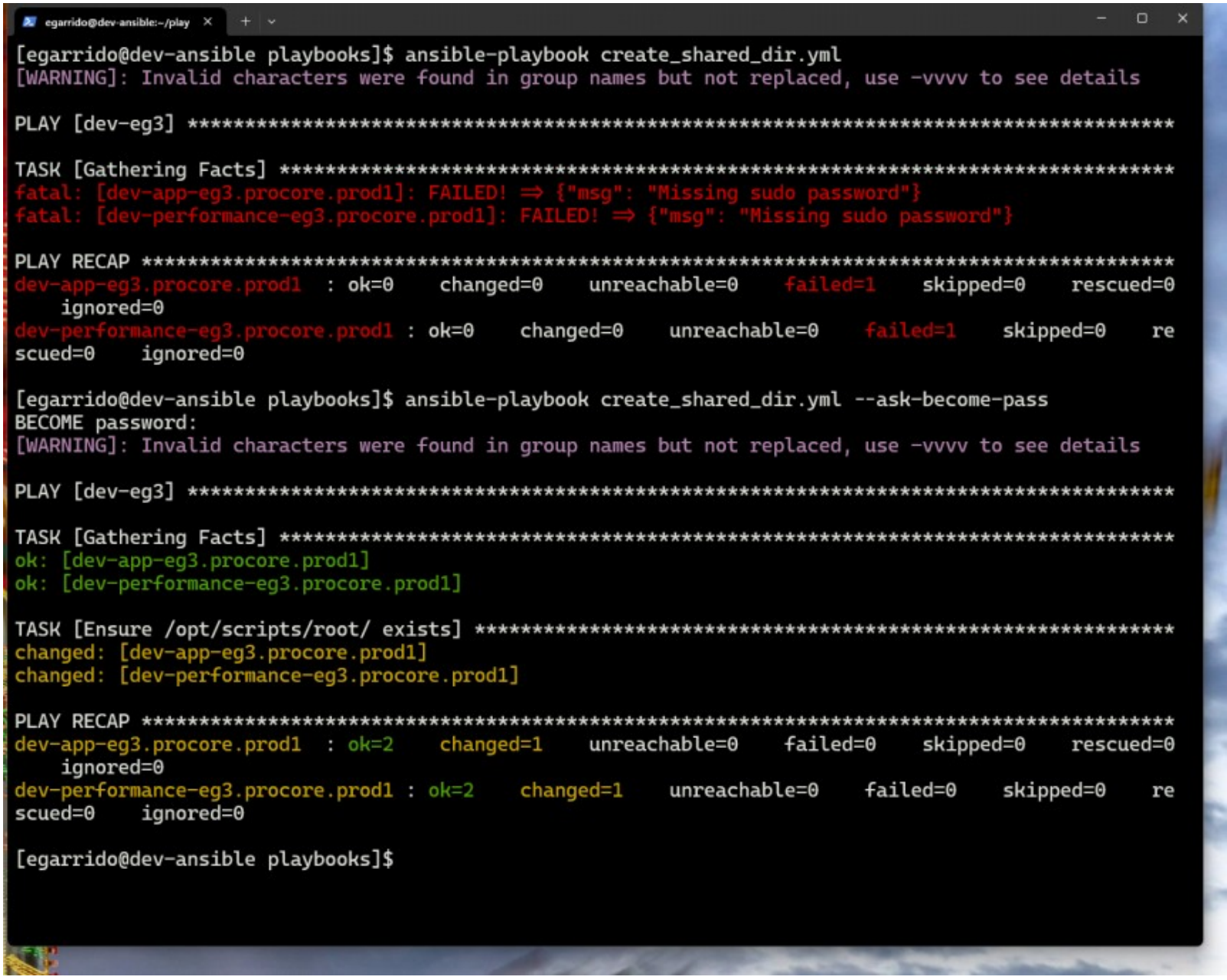
An Ansible playbook was defined to create a user-specific directory under /opt/scripts with enforced ownership, group assignment, and permissions using templated variables.



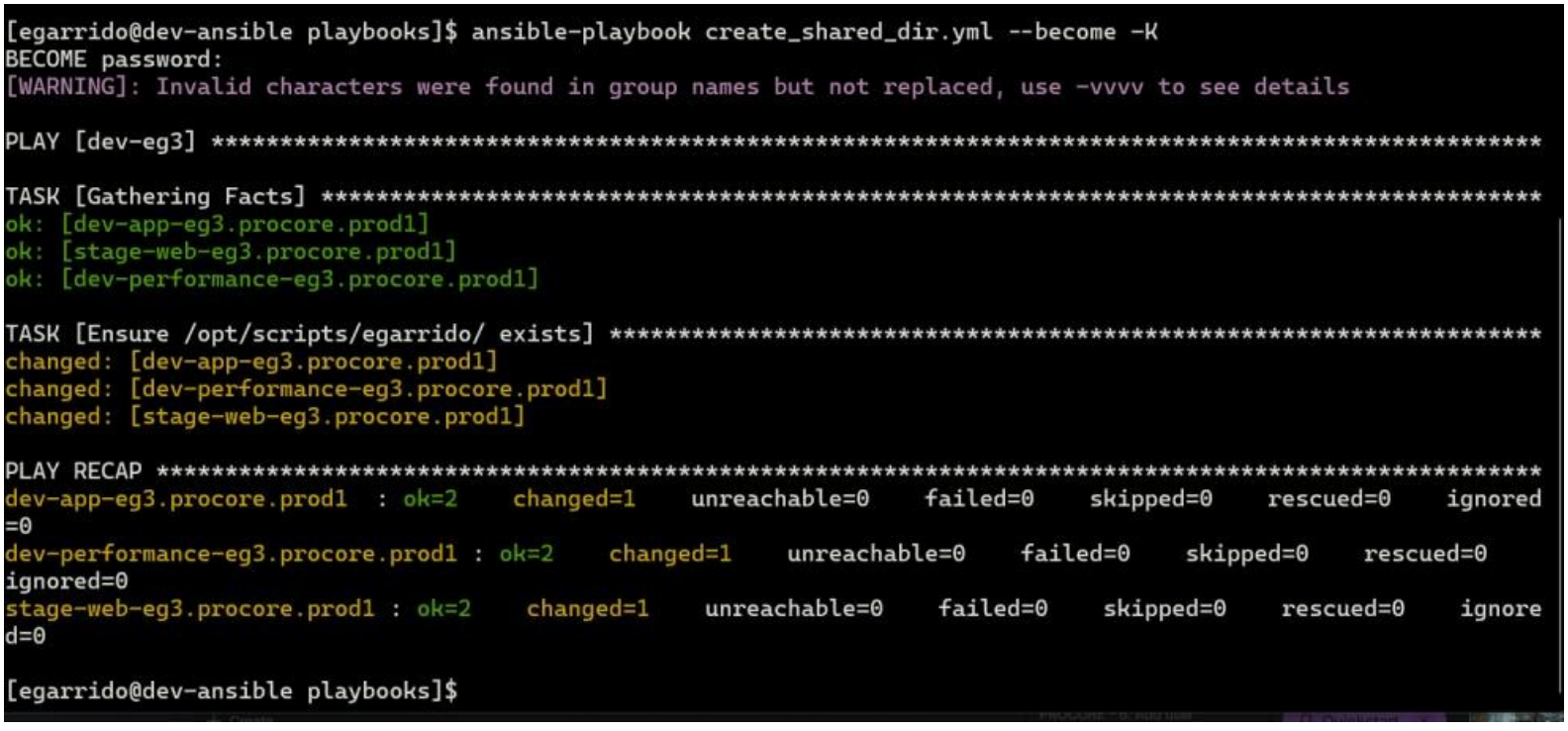
An Ansible playbook syntax issue was identified and resolved by validating the file with ansible-playbook --syntax-check. After correcting the playbook structure to use a proper list of plays, the syntax check completed successfully, confirming the playbook is valid and ready to run.



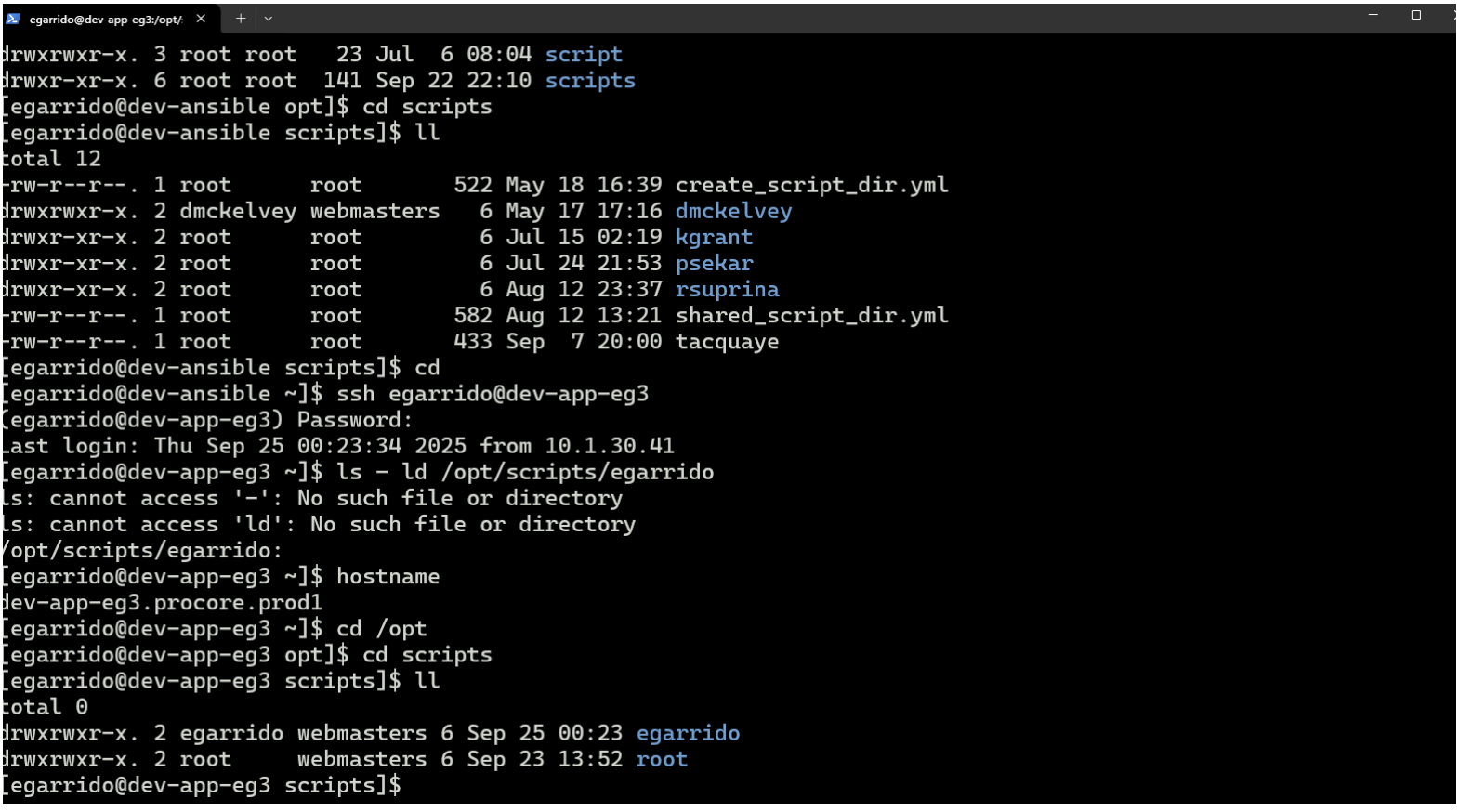
The Ansible playbook initially failed due to missing privilege escalation credentials. After rerunning the playbook with --ask-become-pass, it executed successfully and created the required /opt/scripts directory on all target hosts, confirming proper use of become and successful task execution.



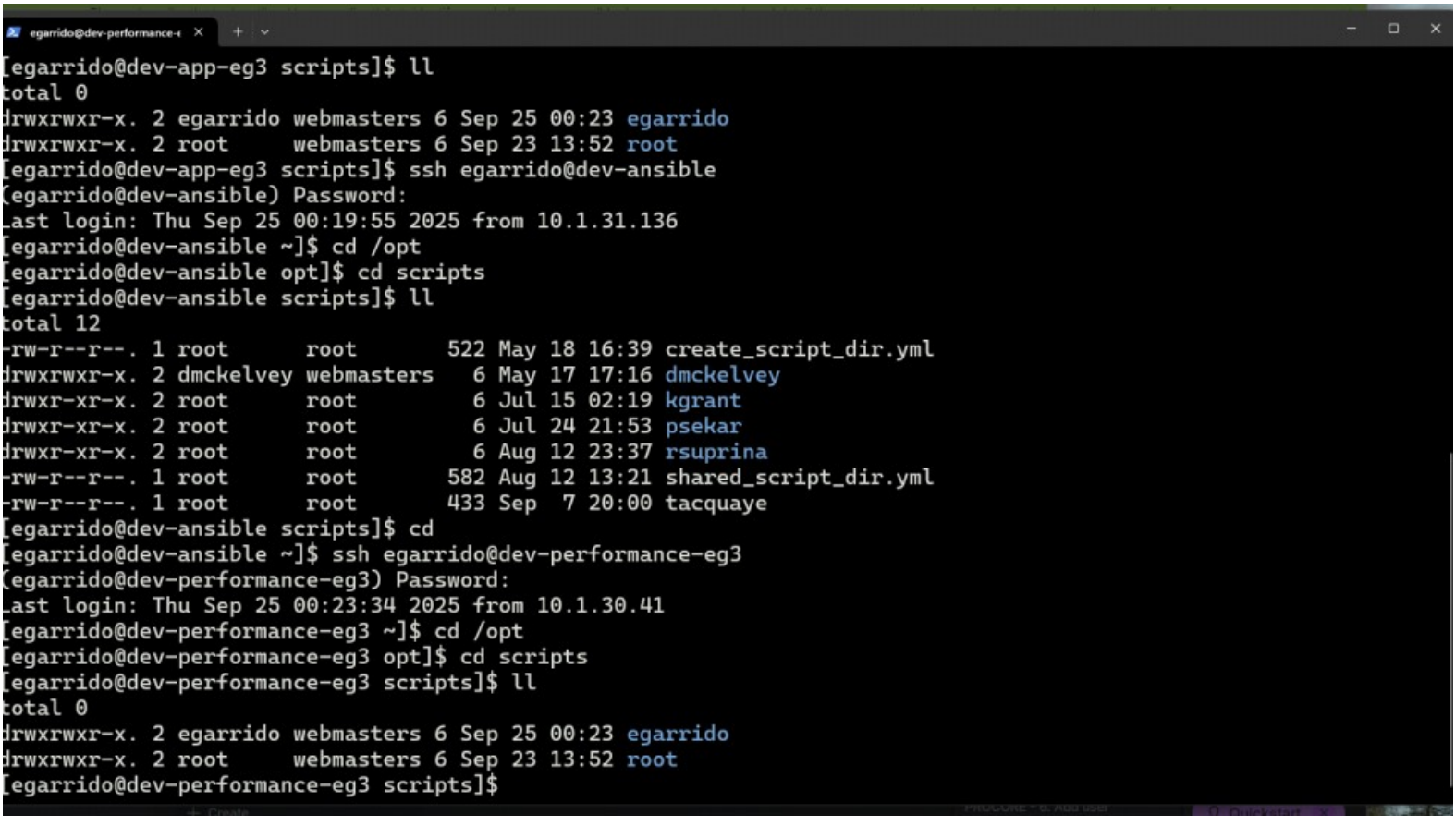
The Ansible playbook was executed with privilege escalation using --become -K and completed successfully across all target hosts. The required user-specific directory under /opt/scripts was created with correct permissions, confirming consistent execution across the environment.



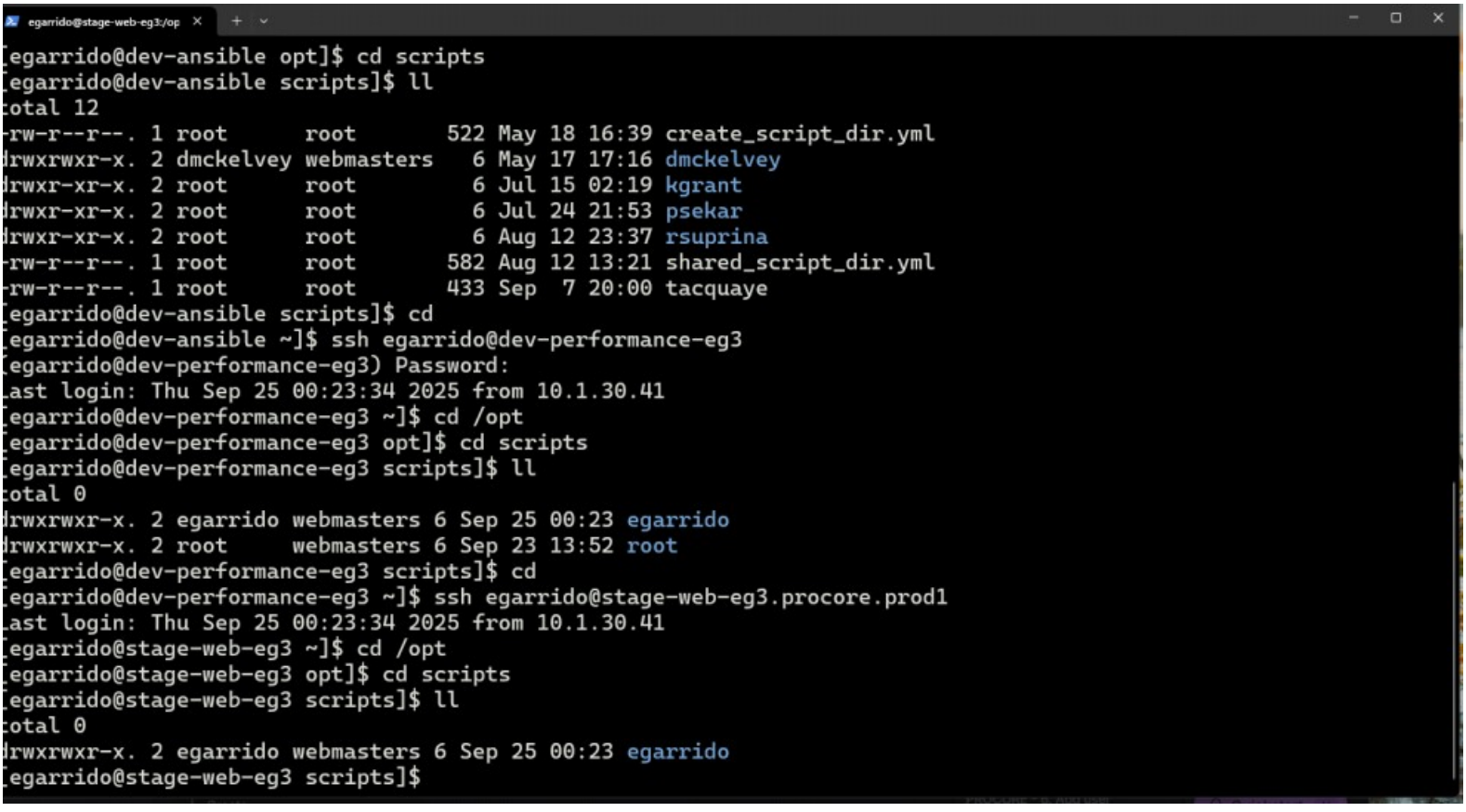
Post-execution validation confirmed that the /opt/scripts/egarrido directory was successfully created on the target host with the expected ownership, group assignment, and permissions, verifying correct playbook execution and desired state enforcement.



Directory validation was performed across multiple hosts, confirming that /opt/scripts/egarrido exists consistently with the correct ownership, group assignment, and permissions. This verification demonstrates successful, repeatable Ansible execution and consistent state enforcement across the environment.



Final validation across all environments confirmed that /opt/scripts/egarrido exists on each host with consistent ownership, group membership, and permissions, verifying reliable and repeatable Ansible automation.



Summary

An Ansible playbook was developed to create a user-specific directory under /opt/scripts with enforced ownership, group membership, and permissions. The playbook was validated using ansible-playbook --syntax-check, during which a structural syntax issue was identified and corrected. Initial execution failed due to missing privilege escalation credentials, which was resolved by rerunning the playbook with --become -K.
After successful execution, the directory was created consistently across all targeted development and staging hosts. Manual verification on each system confirmed the presence of /opt/scripts/egarrido with the correct permissions and ownership, demonstrating reliable, repeatable, and idempotent Ansible automation across multiple environments.