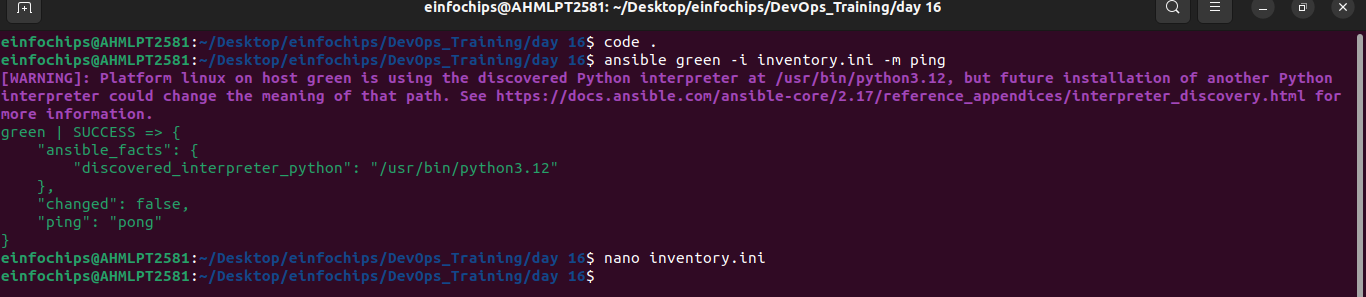
### **Project 1: Deploying Ansible**

### **Problem Statement:** You are tasked with deploying Ansible in a multi-node environment consisting of multiple Linux servers. The goal is to set up Ansible on a control node and configure it to manage several managed nodes. This setup will be used for automating system administration tasks across the network.

**Deliverables:**

1. **Control Node Setup:**
   * Install Ansible on the control node.
   * Configure SSH key-based authentication between the control node and managed nodes.
2. **Managed Nodes Configuration:**
   * Ensure all managed nodes are properly configured to be controlled by Ansible.
   * Verify connectivity and proper setup between the control node and managed nodes.
3. **Documentation:**
   * Detailed installation and configuration steps.
   * Troubleshooting guide for common issues encountered during deployment.



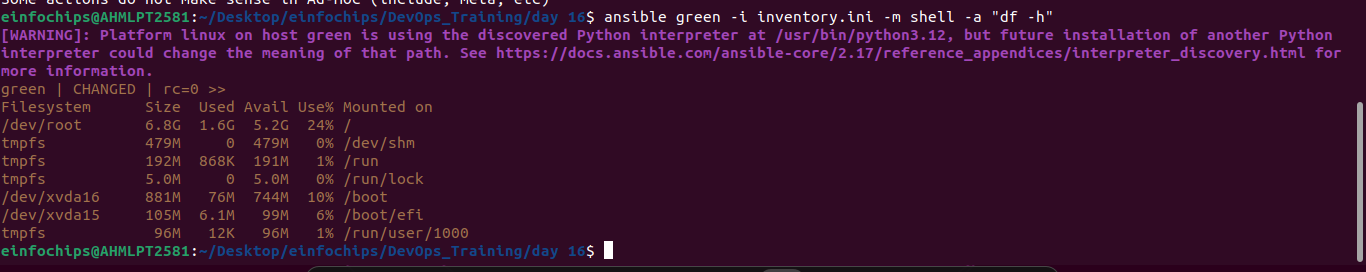
### **Project 2: Ad-Hoc Ansible Commands**

**Problem Statement:** Your organization needs to perform frequent, one-off administrative tasks across a fleet of servers. These tasks include checking disk usage, restarting services, and updating packages. You are required to use Ansible ad-hoc commands to accomplish these tasks efficiently.

**Deliverables:**

1. **Task Execution:**
   * Execute commands to check disk usage across all managed nodes.

ansible green -i inventory.ini -m shell -a “df -h”



* + Restart a specific service on all managed nodes.

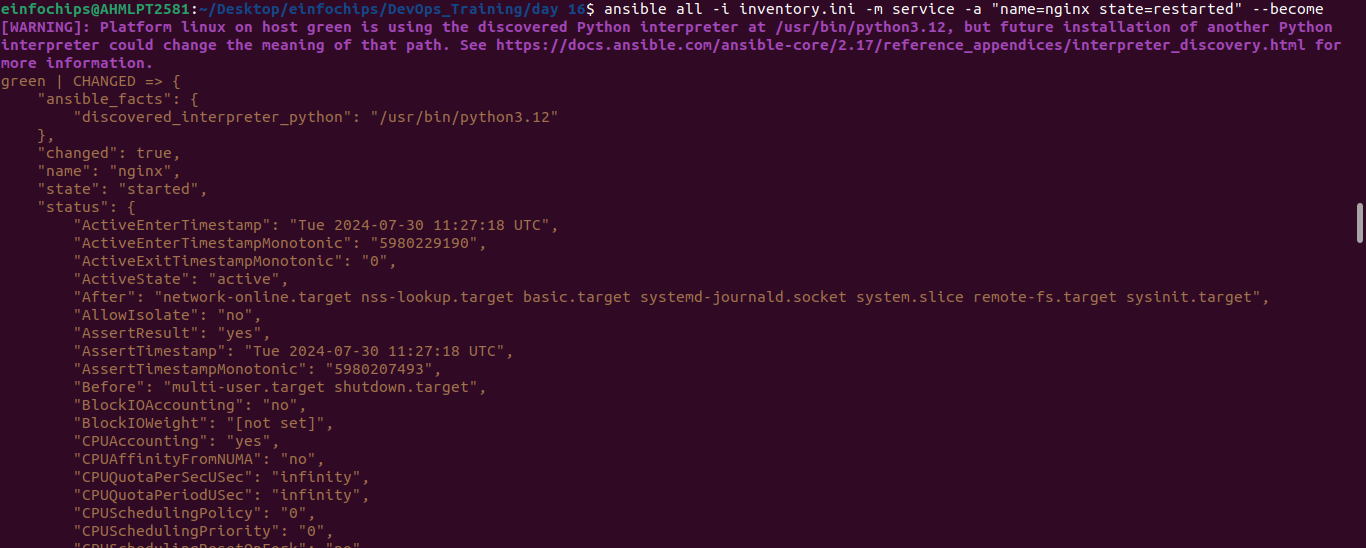
ansible green -i inventory.ini -m apt -a “name=nginx state=present” --become



.

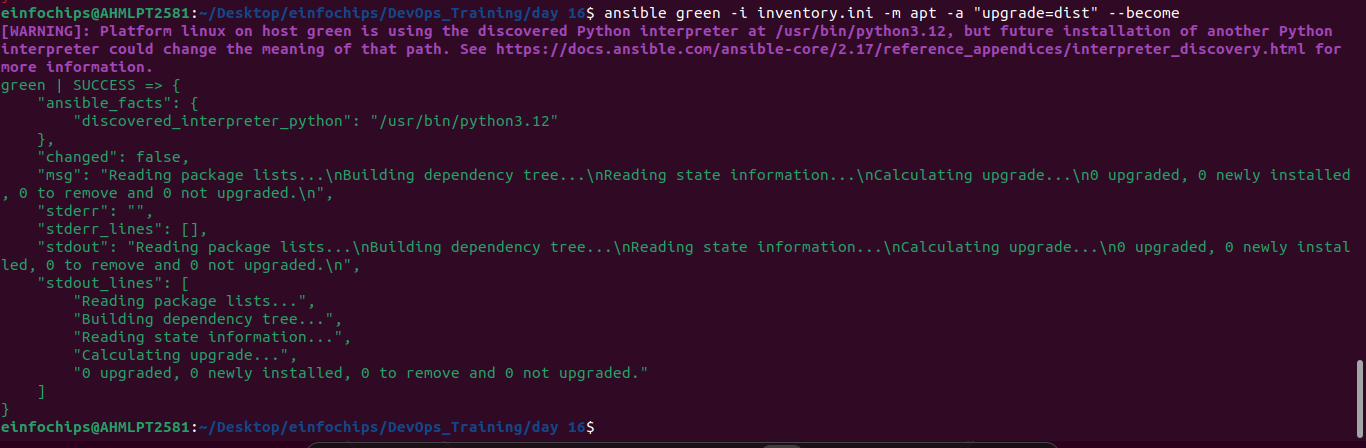
to restart service

ansible all -i inventory.ini -m service -a "name=nginx state=restarted" --become

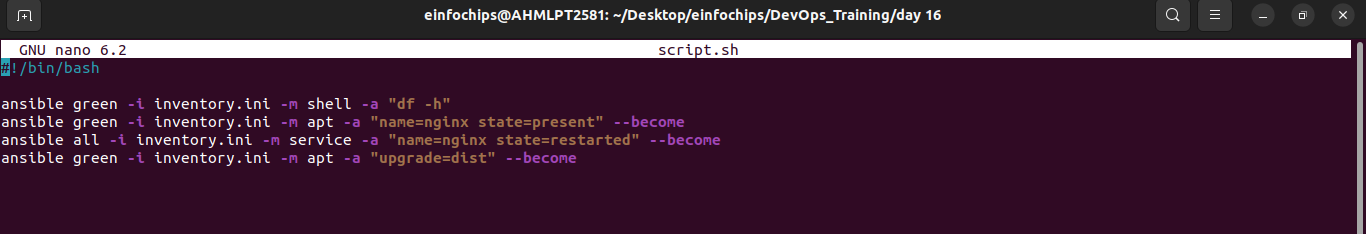


* + Update all packages on a subset of managed nodes.

ansible green -i inventory.ini -m apt -a "upgrade=dist" --become



.

1. **Command Scripts:**
   * Create a script or documentation for each task, detailing the ad-hoc command used and its output.
   * 

.

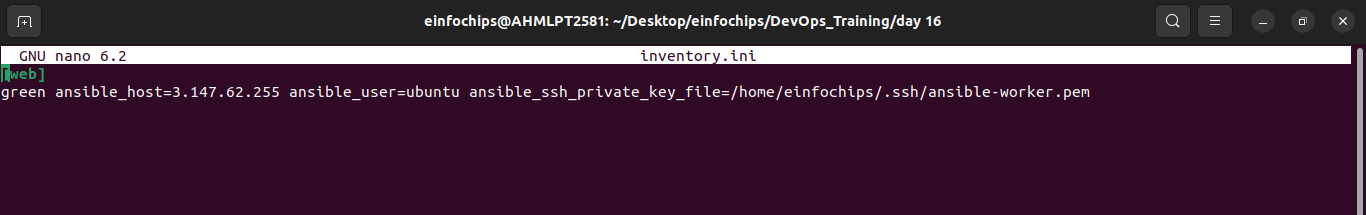
1. **Documentation:**
   * Provide a comprehensive guide on using Ansible ad-hoc commands.
   * Include examples of common administrative tasks that can be performed with ad-hoc commands.

### **Project 3: Working with Ansible Inventories**

**Problem Statement:** You need to manage a dynamic and diverse set of servers, which requires an organized and flexible inventory system. The project involves creating static and dynamic inventories in Ansible to categorize servers based on different attributes such as environment (development, staging, production) and roles (web servers, database servers).

**Deliverables:**

1. **Static Inventory:**
   * Create a static inventory file with different groups for various environments and roles.
   * Verify that the inventory is correctly structured and accessible by Ansible.

.

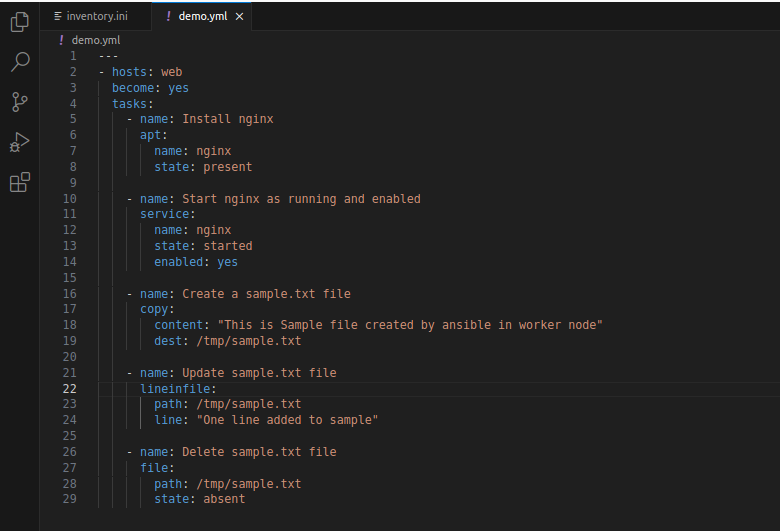
1. **.Dynamic Inventory:**
   * Implement a dynamic inventory script or use a dynamic inventory plugin.
   * Configure the dynamic inventory to categorize servers automatically based on predefined criteria.
2. **Documentation:**
   * Instructions for setting up and using static and dynamic inventories.
   * Examples of playbooks utilizing both types of inventories.

### **Project 4: Ansible Playbooks: The Basics**

**Problem Statement:** Your team needs to automate repetitive tasks such as installing packages, configuring services, and managing files on multiple servers. The project involves writing basic Ansible playbooks to automate these tasks, ensuring consistency and efficiency in the operations.

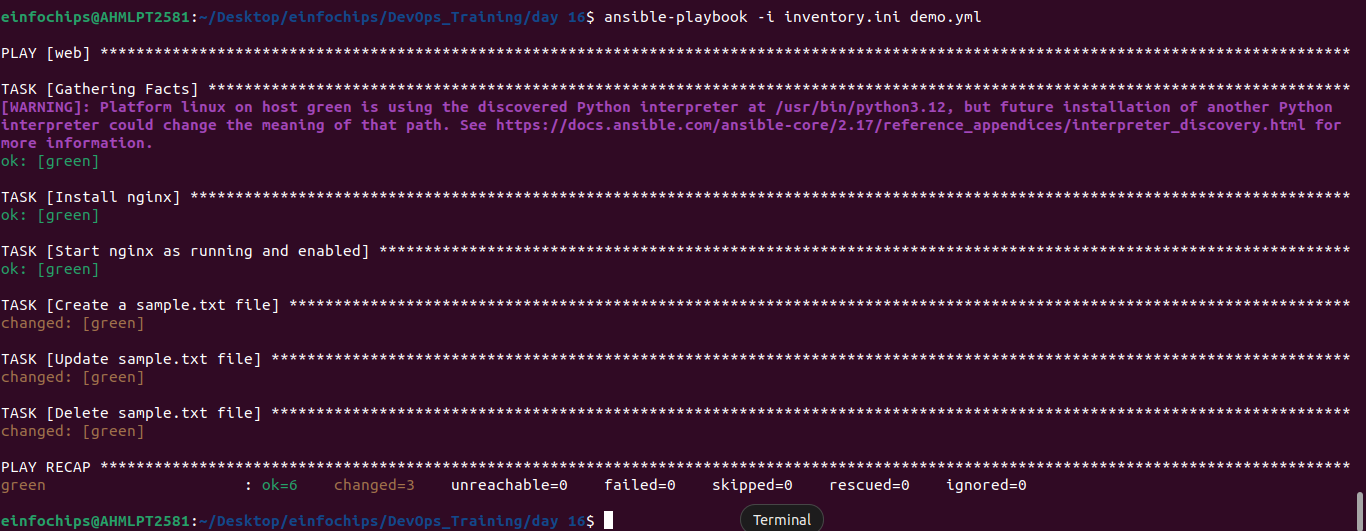
**Deliverables:**

1. **Playbook Creation:**
   * Write a playbook to install a specific package on all managed nodes.
   * Create a playbook to configure a service with specific parameters.
   * Develop a playbook to manage files, such as creating, deleting, and modifying files on managed nodes.



.

1. **Testing and Verification:**
   * Test the playbooks to ensure they run successfully and perform the intended tasks.
   * Validate the changes made by the playbooks on the managed nodes.

.

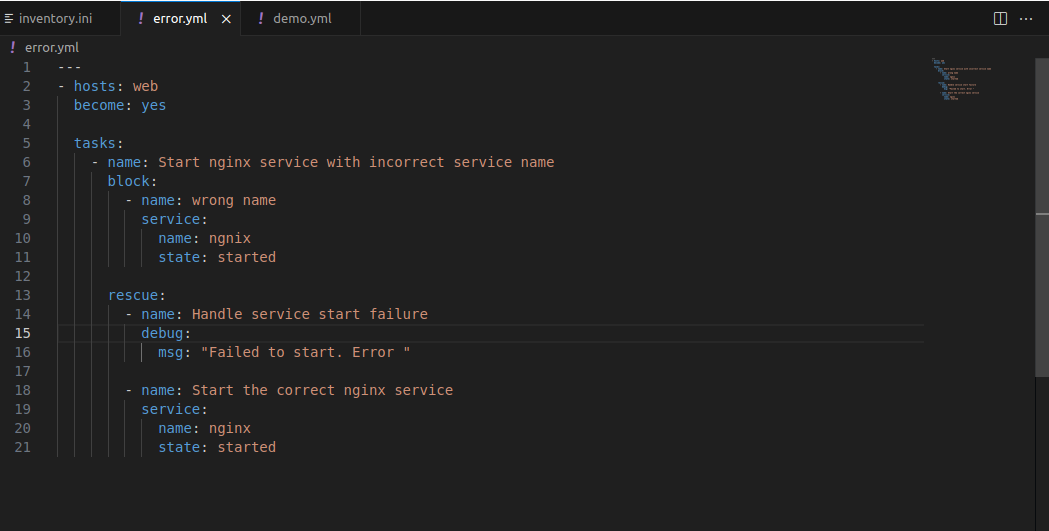
1. **Documentation:**
   * Detailed explanation of each playbook, including the tasks and modules used.
   * Guide on how to run the playbooks and interpret the results.

### **Project 5: Ansible Playbooks - Error Handling**

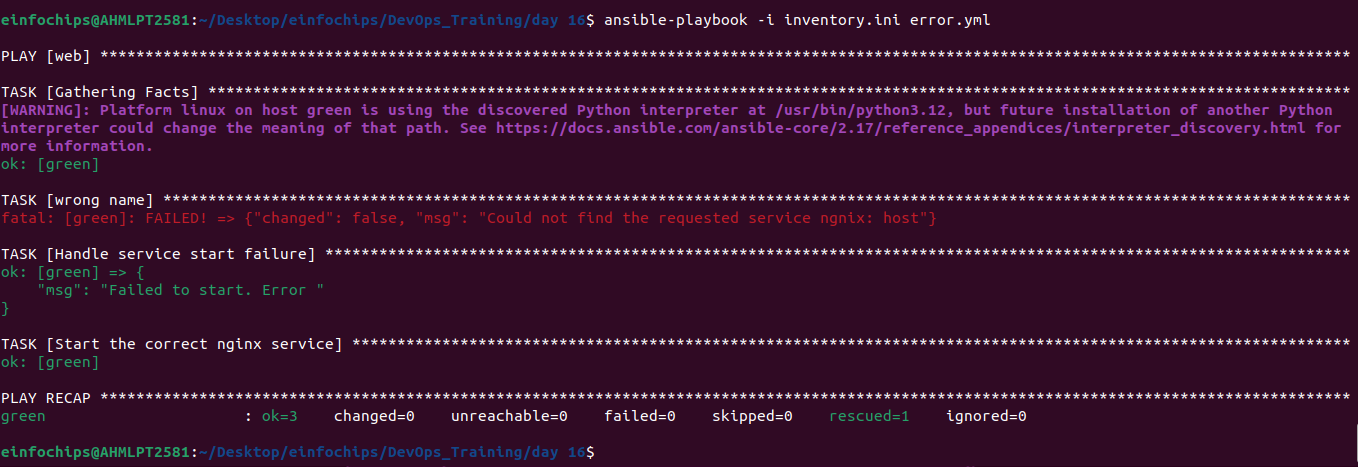
**Problem Statement:** In a complex IT environment, tasks automated by Ansible playbooks may encounter errors due to various reasons such as incorrect configurations, unavailable resources, or network issues. The project focuses on implementing error handling in Ansible playbooks to ensure resilience and proper reporting of issues.

**Deliverables:**

1. **Playbook with Error Handling:**
   * Write a playbook that includes tasks likely to fail, such as starting a non-existent service or accessing a non-existent file.
   * Implement error handling strategies using modules like block, rescue, and always.

.

Output:



1. **Logging and Notifications:**
   * Configure the playbook to log errors and notify administrators of any issues encountered during execution.
   * Use Ansible modules to send notifications via email or other communication channels.
2. **Documentation:**
   * Comprehensive guide on error handling in Ansible playbooks, including examples of common scenarios and solutions.
   * Detailed instructions on setting up logging and notifications for error handling.