Name: John Era Roldan	**Date Performed: 10/10/2025**
Course/Section: CPE212	**Date Submitted: 10/10/2025**
Instructor: Engr. Robin Valenzuela	**Semester and SY: 1st Sem 2025-2026**

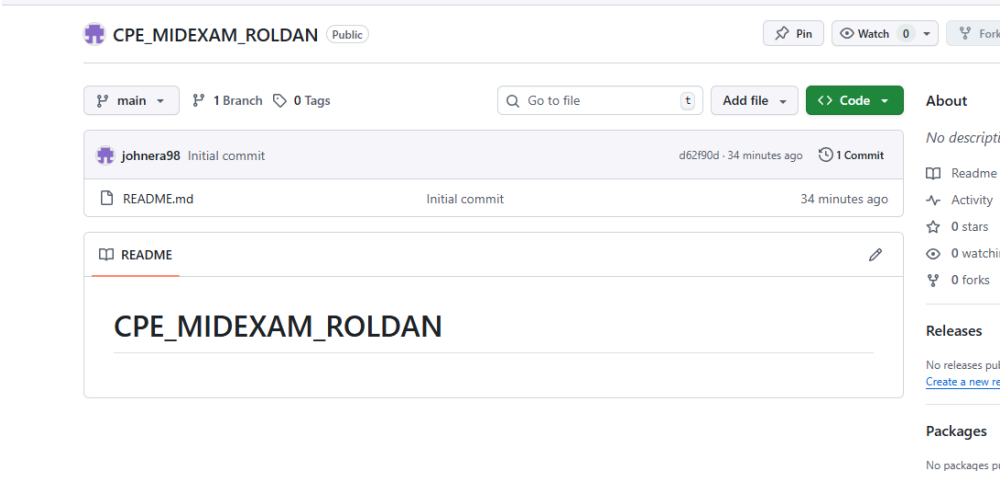
Midterm Skills Exam: Install, Configure, and Manage Log Monitoring tools

1. Objectives

Create and design a workflow that installs, configure and manage enterprise availability, performance and log monitoring tools using Ansible as an Infrastructure as Code (IaC) tool.

2. Instructions

1. Create a repository in your GitHub account and label it CPE_MIDEXAM_SURNAME.



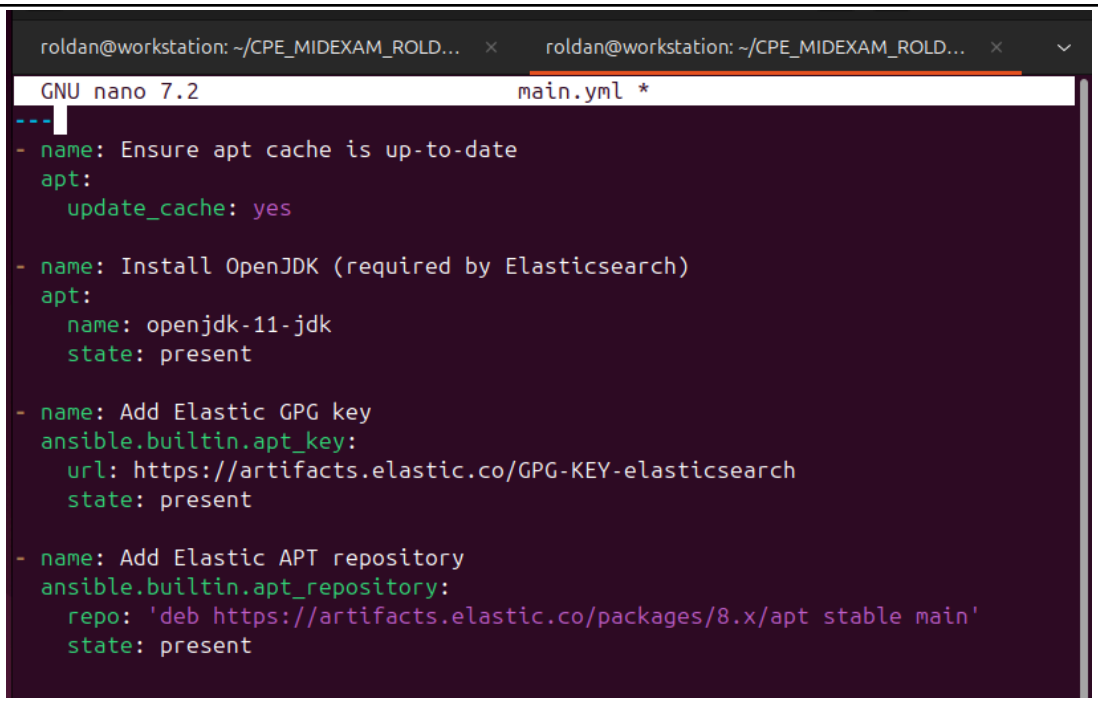
2. Clone the repository and do the following:
   2.1. Create an Ansible playbook that does the following with an input of a config.yaml file and arranged Inventory file:

```
roldan@workstation:~/CPE_MIDEXAM_ROLDAN$ ls
ansible.cfg  config.yaml  group_vars  inventory.ini  README.md  roles  site.yml
```

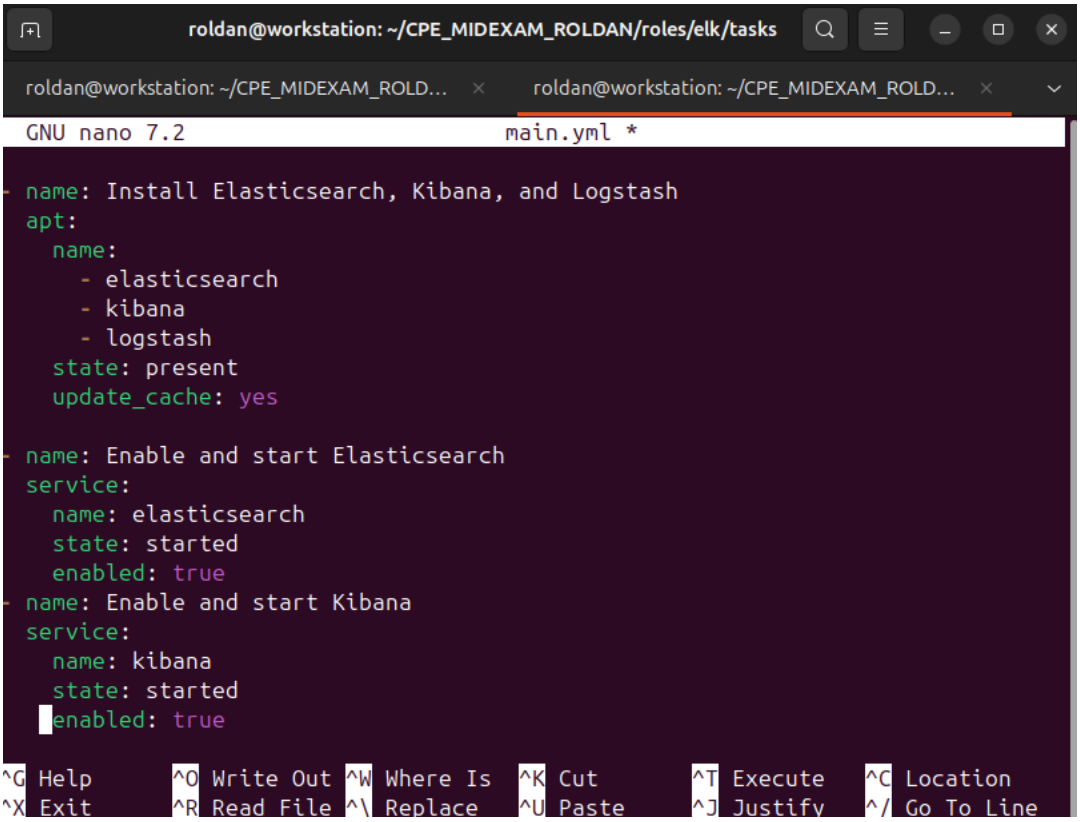
   2.2. Install and configure Elastic Stack in separate hosts (Elastic Search, Kibana, Logstash) • Install Nagios in one host

ELASTIC

```
roldan@workstation:~/CPE_MIDEXAM_ROLDAN/roles/elk/tasks$ sudo nano main.yml
```


roldan@workstation: ~/CPE_MIDEXAM_ROLD...
roldan@workstation: ~/CPE_MIDEXAM_ROLD...
GNU nano 7.2 main.yml *

- name: Ensure apt cache is up-to-date
apt:
update_cache: yes
- name: Install OpenJDK (required by Elasticsearch)
apt:
name: openjdk-11-jdk
state: present
- name: Add Elastic GPG key
ansible.builtin.apt_key:
url: https://artifacts.elastic.co/GPG-KEY-elasticsearch
state: present
- name: Add Elastic APT repository
ansible.builtin.apt_repository:
repo: 'deb https://artifacts.elastic.co/packages/8.x/apt stable main'
state: present


roldan@workstation: ~/CPE_MIDEXAM_ROLDAN/roles/elk/tasks
roldan@workstation: ~/CPE_MIDEXAM_ROLD...
roldan@workstation: ~/CPE_MIDEXAM_ROLD...
GNU nano 7.2 main.yml *
- name: Install Elasticsearch, Kibana, and Logstash
apt:
name:
- elasticsearch
- kibana
- logstash
state: present
update_cache: yes
- name: Enable and start Elasticsearch
service:
name: elasticsearch
state: started
enabled: true
- name: Enable and start Kibana
service:
name: kibana
state: started
enabled: true
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line

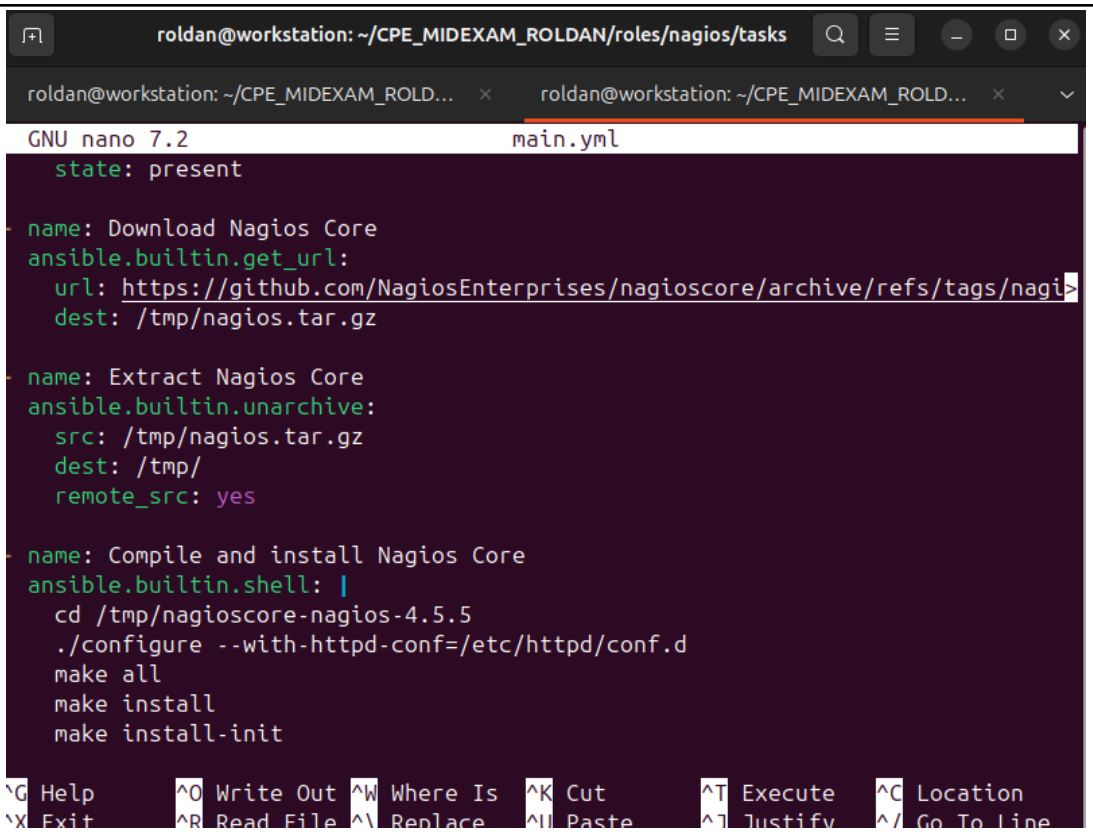
```
- name: Enable and start Logstash
  service:
    name: logstash
    state: started
    enabled: true
```

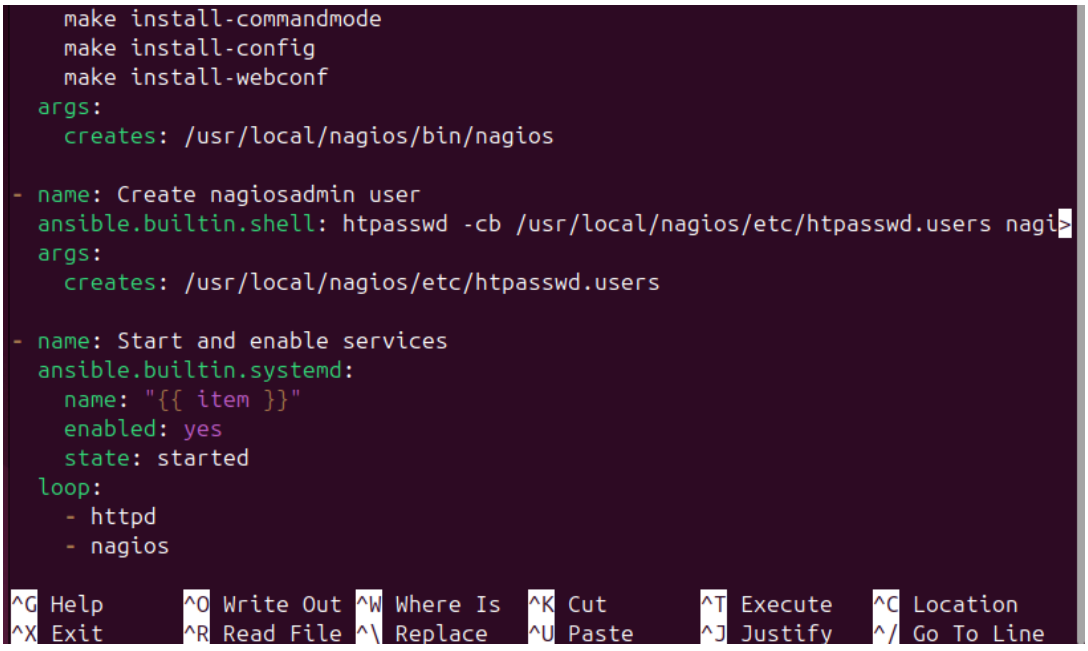
NAGIOS

```
roldan@workstation:~/CPE_MIDEXAM_ROLDAN/roles/nagios/tasks$ sudo nano main.yml
```

roldan@workstation: ~/CPE_MIDEXAM_ROLD... × roldan@workstation: ~/CPE_MIDEXAM_ROLD... ×

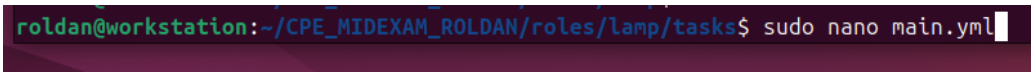
```
 GNU nano 7.2                                main.yml
---
- name: Install required dependencies
  ansible.builtin.yum:
    name:
      - gcc
      - glibc
      - glibc-common
      - make
      - wget
      - unzip
      - httpd
      - php
      - openssl
      - openssl-devel
      - gd
      - gd-devel
      - perl
      - postfix
      - autoconf
      - automake
```





2.3. Install Grafana,Prometheus and Influxdb in seperate hosts (Influxdb,Grafana,Prometheus)

2.4. Install Lamp Stack in separate hosts (Httpd + Php,Mariadb)




roldan@workstation: ~/CPE_MIDEXAM_ROLDAN/roles/lamp/tasks
roldan@workstation: ~/CPE_MIDEXAM_ROLD...
roldan@workstation: ~/CPE_MIDEXAM_ROLD...
GNU nano 7.2 main.yml

- name: Install Apache and PHP packages
apt:
name: "{{ item }}"
state: present
update_cache: yes
loop: "{{ lamp.php_packages | default(['apache2','php','libapache2-mod-php','>
when: "'lamp_web' in group_names and ansible_os_family == 'Debian'"
- name: Ensure Apache is started and enabled
service:
name: apache2
state: started
enabled: true
when: "'lamp_web' in group_names and ansible_os_family == 'Debian'"
- name: Install MariaDB server
apt:
name: mariadb-server
state: present
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line
roldan@workstation: ~/CPE_MIDEXAM_ROLDAN/roles/lamp/tasks
roldan@workstation: ~/CPE_MIDEXAM_ROLD...
roldan@workstation: ~/CPE_MIDEXAM_ROLD...
GNU nano 7.2 main.yml
update_cache: yes
when: "'lamp_db' in group_names and ansible_os_family == 'Debian'"
- name: Ensure MariaDB is started and enabled
service:
name: mysql
state: started
enabled: true
when: "'lamp_db' in group_names and ansible_os_family == 'Debian'"
- name: Secure MariaDB - Set root password
mysql_user:
login_user: root
login_password: ""
name: root
host_all: yes
password: "{{ lamp.mariadb_root_password | default('rootpass') }}"
check_implicit_admin: yes
when: "'lamp_db' in group_names and ansible_os_family == 'Debian'"
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line

3. Document all your tasks using this document. Provide proofs of all the ansible playbooks codes and successful installations.

```
roldan@workstation:~/CPE_MIDEXAM_ROLDAN$ cat ansible.cfg
[defaults]
inventory = inventory.ini
remote_user = jroldan
host_key_checking = False
private_key_file = ~/.ssh/id_rsa
```

- This file tells Ansible how to connect to my servers and where to find my roles. It sets the SSH user, points to my private key for passwordless login, disables annoying host key checks, and tells Ansible where to look for roles.

```
roldan@workstation:~/CPE_MIDEXAM_ROLDAN$ cat config.yaml
elk:
   elastic_version: "8.x"
   heap_size: "1g"

nagios:
    nagiosadmin_password: "johnerapogi"


lamp:
  php_packages: ["php", "libapachee2-mod-php", "php-mysql"]
  mariadb_root_password: "johnerapogi"
```

- This file is my control panel for software versions and settings. Instead of hardcoding package versions in my playbooks

```
roldan@workstation:~/CPE_MIDEXAM_ROLDAN$ cat inventory.ini
[ubuntu]
ubuntu1 ansible_host=192.168.56.118 ansible_user=roldan

[centos]
centos1 ansible_host=192.168.56.122 ansible_user=jroldan

[elk]
elk1 ansible_host=192.168.56.118 ansible_user=roldan

[nagios]
nagios1 ansible_host=192.168.56.122 ansible_user=jroldan

[lamp]
lamp_web ansible_host=192.168.56.118 ansible_user=roldan
lamp_db ansible_host=192.168.56.122 ansible_user=jroldan
[all:vars]
ansible_become=true
```

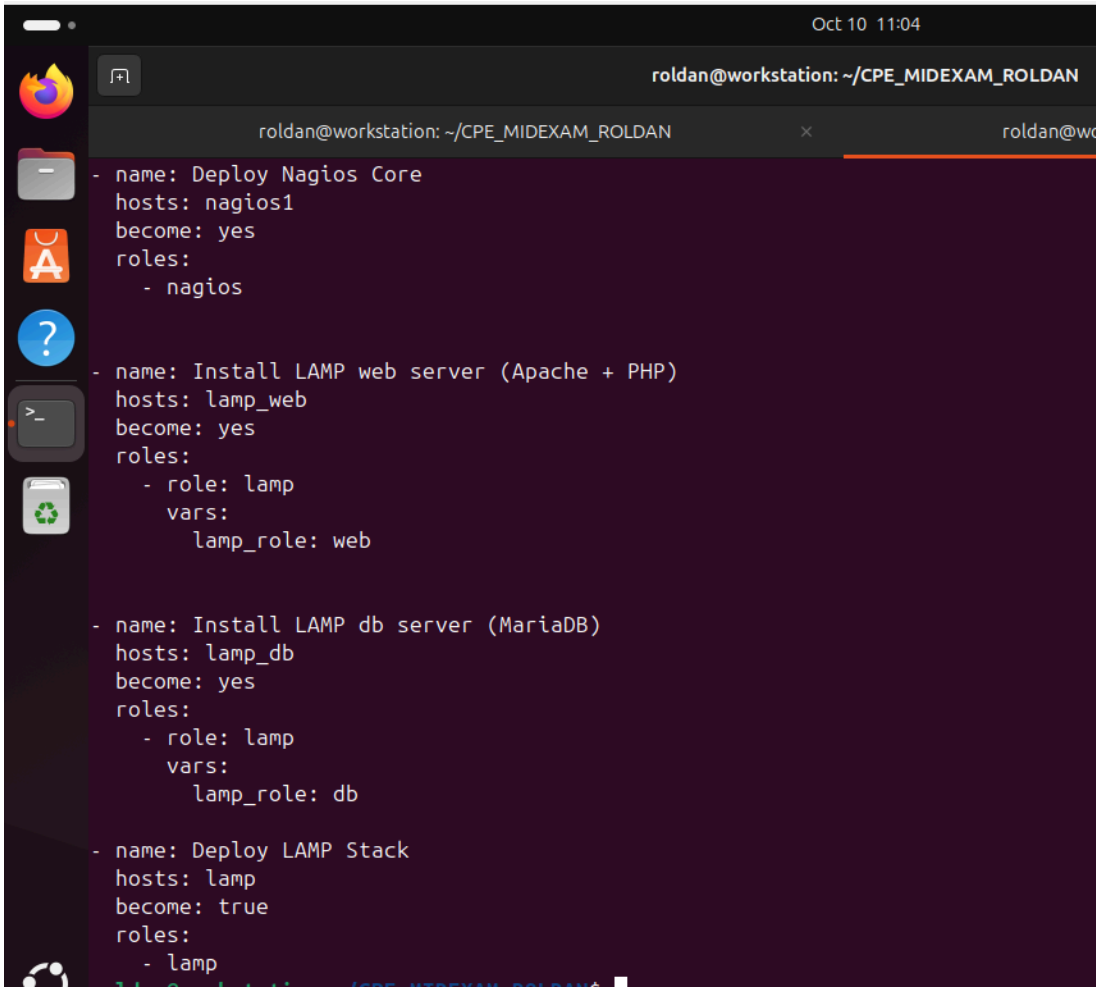
- The inventory is like an address book for Ansible. It lists all my servers and groups them by purpose to ELK servers, Nagios servers, and LAMP servers. Each server entry includes its IP and login info so Ansible knows where to send commands.

```
roldan@workstation:~/CPE_MIDEXAM_ROLDAN$ cat site.yml
---
- name: Load config
  hosts: localhost
  gather_facts: no
  vars_files:
    - ./config.yaml
  tasks:
    - name: include nothing (just load vars)
      debug:
        msg: "Loaded config"


- name: Install ELK stack (Elasticsearch, Logstash, Kibana) on elk hosts
  hosts: elk
  become: yes
  roles:
    - role: elk

- name: Install Nagios
  hosts: nagios
  become: yes
  roles:
    - role: nagios
```

- This is the “director” of my automation. It tells Ansible which roles to run and in what order. It calls ELK, Nagios, Monitoring, and LAMP roles for the appropriate servers, making sure each group gets the right setup. Basically, it schedules all my automation tasks.



4. Document the push and commit from the local repository to GitHub.
5. Finally, paste also the link of your GitHub repository in the documentation.

3. Output

ELASTIC

```
roldan@workstation:~/CPE_MIDEXAM_ROLDAN$ ansible-playbook -i inventory.ini site.yml --limit elk --ask-become-pass

BECOME password:

PLAY [Load config] *********************************************************
skipping: no hosts matched

PLAY [Install ELK stack (Elasticsearch, Logstash, Kibana) on elk hosts] ********

TASK [Gathering Facts] *****************************************************
ok: [elk1]

TASK [elk : Ensure apt cache is up-to-date] ********************************
changed: [elk1]

TASK [elk : Install OpenJDK (required by Elasticsearch)] ***********************
ok: [elk1]

TASK [elk : Check Java version] ********************************************
changed: [elk1]

TASK [elk : Fail if Java not installed properly] ***************************
skipping: [elk1]

TASK [elk : Add Elastic GPG key] *******************************************
ok: [elk1]

TASK [elk : Add Elastic APT repository] ************************************
ok: [elk1]
```

```
TASK [elk : Install Elasticsearch, Kibana, and Logstash] ***********************
ok: [elk1]

TASK [elk : Set vm.max_map_count for Elasticsearch] ****************************
ok: [elk1]

TASK [elk : Ensure Elasticsearch directories owned by elasticsearch user] ******
ok: [elk1] => (item=/etc/elasticsearch)
ok: [elk1] => (item=/var/lib/elasticsearch)
ok: [elk1] => (item=/var/log/elasticsearch)

TASK [elk : Enable and start Elasticsearch] ************************************
```

This role installs the Elastic Stack (Elasticsearch, Logstash, Kibana). It first installs dependencies, adds the Elastic GPG key, registers the APT repository correctly, installs the packages, and starts the services. Think of it as setting up a mini logging and visualization center on my Ubuntu VM.

NAGIOS

```
roldan@workstation:~/CPE_MIDEXAM_ROLDAN$ ansible-playbook -i inventory.ini site.yml --limit nagios --ask-become-pass
BECOME password:

PLAY [Load config] *************************************************************************************************
skipping: no hosts matched

PLAY [Install ELK stack (Elasticsearch, Logstash, Kibana) on elk hosts] ********************************************
skipping: no hosts matched

PLAY [Install Nagios] **********************************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [nagios1]

TASK [nagios : Install required dependencies] **********************************************************************
ok: [nagios1]

TASK [nagios : Ensure Nagios etc directory exists] *****************************************************************
ok: [nagios1]

TASK [nagios : Ensure Nagios var directory exists] *****************************************************************
ok: [nagios1]

TASK [nagios : Download Nagios Core] *******************************************************************************
ok: [nagios1]

TASK [nagios : Extract Nagios Core] ********************************************************************************
ok: [nagios1]
```

```
TASK [nagios : Compile and install Nagios Core] ********************************************************************
changed: [nagios1]

TASK [nagios : Create nagiosadmin user] ****************************************************************************
ok: [nagios1]

TASK [nagios : Create systemd service for Nagios] ******************************************************************
ok: [nagios1]

TASK [nagios : Reload systemd to recognize new Nagios service] *****************************************************
ok: [nagios1]

TASK [nagios : Start and enable services] **************************************************************************
ok: [nagios1] => (item=httpd)
failed: [nagios1] (item=nagios) => {"ansible_loop_var": "item", "changed": false, "item": "nagios", "msg": "Unable to st
art service nagios: Job for nagios.service failed because the control process exited with error code.\nSee \"systemctl s
tatus nagios.service\" and \"journalctl -xeu nagios.service\" for details.\n"}

PLAY RECAP *********************************************************************************************************
nagios1                    : ok=10   changed=1    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0

roldan@workstation:~/CPE_MIDEXAM_ROLDAN$
```

This role installs Nagios from source on CentOS. It installs dependencies, downloads Nagios, compiles it, installs configuration and commands, and sets up the web interface. But when I start and enable it there's an error. I don't know why I already reload the systemd on nagios service but still there's an error.

GitHub link:

https://github.com/johnera98/CPE_MIDEXAM_ROLDAN

Conclusions: (link your conclusion from the objective)

In this midterm exam, I learned how to use Ansible to automate the setup of multiple servers for different purposes, like monitoring, logging, and web hosting. By organizing tasks into roles in ELK for logs, Nagios, and LAMP for web services. We could install, configure, and start services automatically without manually typing commands on each server. This approach saves time, reduces errors, and makes it easy to reproduce the environment on other machines. Overall, it showed how automation can turn a complex multi-server setup into a smooth, repeatable process, making managing servers much simpler and more efficient.