**Node Exporter via Ansible with Monitoring**

**Github link -** [**https://github.com/aaryan-19/luganodes-devops-task2**](https://github.com/aaryan-19/luganodes-devops-task2)

**Introduction**

**1. Prometheus -**

Prometheus is an open-source monitoring and alerting system designed for collecting, storing, querying, and alerting on time-series data.

**2. Node Exporter -**

Popular open-source software Node Exporter was created by the Prometheus community. It is a compact and effective Prometheus exporter made to gather different system-level metrics from platforms that operate similarly to Unix, such as Linux, Windows, and macOS. By exposing essential metrics for analysis, visualization, and alerting, Node Exporter enables you to keep an eye on the health and performance of your servers and workstations.

**3. Grafana -**

Grafana is an open-source monitoring and observability software that focuses on creating dashboards and data visualization. It is an effective tool for tracking the performance and well-being of systems, applications, and services because it is made to assist users in seeing and analysing their metrics and logs in real time.

**4. Ansible –**

You can automate the configuration, deployment, and maintenance of IT infrastructure, applications, and services with the open-source automation tool Ansible. It is intended to decrease manual intervention, streamline difficult tasks, and boost the effectiveness of IT operations.

**Module Explanation**

**Ansible.cfg file –**

First we have to configure the ansible file –

[defaults]

inventory = ./inventory

remote\_user = ec2-user

host\_key\_checking = False

Let's break down the steps:

1. **[defaults] Section:** The **[defaults]** section is a section header in the **ansible.cfg** file. It indicates that the following settings are considered defaults for Ansible's behavior.
2. **inventory = ./inventory:** This setting specifies the path to the inventory file that Ansible should use. The **./inventory** path points to an inventory file named **inventory** located in the same directory as the **ansible.cfg** file. The inventory file contains a list of target hosts on which Ansible tasks and playbooks will be executed.
3. **remote\_user = ec2-user:** This setting specifies the default remote user that Ansible should use when connecting to target hosts via SSH. In this case, the **ec2-user** is specified as the default user. When running Ansible commands or playbooks, if you don't explicitly specify a remote user, Ansible will use **ec2-user** as the default user.
4. **host\_key\_checking = False:** This setting controls whether Ansible should check and validate host SSH keys when connecting to target hosts. Setting it to **False** disables host key checking. Host key checking is a security feature that helps prevent man-in-the-middle attacks, as Ansible verifies that the target host's public key matches the one stored in your known\_hosts file. Disabling host key checking is generally not recommended for security reasons, as it could expose you to potential security risks.

It is important to note that while these settings can be configured in the **ansible.cfg** file, they can also be overridden on the command line or in playbooks using variables and command-line options. The **ansible.cfg** file is a way to provide default settings for your Ansible environment, but you can customize these settings based on your needs.

**Inventory File –**

[prometheus]

127.0.0.1

[node\_exporter]

127.0.0.1

[grafana]

127.0.0.1

In Ansible, the inventory file is used to specify the target hosts or groups of hosts on which you want to perform tasks, such as deploying software or running playbooks. The inventory file helps Ansible know where to apply the automation.

In my inventory files-

[Prometheus], [node\_exporter], and [Grafana] are group names and `127.0.0.1` represents a target host that belongs to that group. In my case, I am targeting the local machine.

Note – You can also add multiple hosts for other servers.

**Now we need to create roles –**

1. **Prometheus\_Node\_exporter :**

Command - ansible-galaxy init roles/prometheus\_node\_exporter

A Node Exporter role in Ansible is typically used to automate the deployment and configuration of Node Exporter, an agent that collects system-level metrics from servers and other hosts. These metrics are then made available in a format that can be scraped and stored by Prometheus, a popular open-source monitoring and alerting system.

Creating a Node Exporter role allows you to encapsulate all the necessary tasks, configurations, and files needed to install, configure, and manage Node Exporter on multiple hosts. This makes it easier to deploy Node Exporter consistently across your infrastructure and ensures that the required setup is maintained.

Main.yml in roles/prometheus\_node\_exporter/tasks –

It is responsible for deploying the Prometheus Node Exporter, which is a component used to export system and hardware metrics from a host to Prometheus for monitoring. These tasks perform various actions to install, configure, and start the Node Exporter. Let's break down each task step by step:

- name: Creating node\_exporter user group

  group: name="{{groupId}}"

  become: true

This task creates a user group named **node\_exporter** on the target host. The **become: true** statement indicates that Ansible should escalate its privileges (typically using **sudo**) to execute this task.

- name: Creating node\_exporter user

  user:

    name: "{{userId}}"

    group: "{{groupId}}"

    system: yes

    shell: "/sbin/nologin"

    comment: "{{userId}} nologin User"

    createhome: "no"

    state: present

This task creates a system user named **node\_exporter**. The user is added to the **node\_exporter** group, given the specified shell and user comment. The **system: yes** attribute indicates that it's a system user, and **createhome: "no"** specifies not to create a home directory for the user.

- name: Install prometheus node exporter

  unarchive:

    src: "https://github.com/prometheus/node\_exporter/releases/download/v{{ version }}/node\_exporter-{{ version }}.linux-amd64.tar.gz"

    dest: /tmp/

    remote\_src: yes

This task downloads the Prometheus Node Exporter binary release from the specified URL and unpacks it into the **/tmp/** directory on the target host. The **remote\_src: yes** attribute tells Ansible to fetch the source file from a remote location.

- name: Copy prometheus node exporter file to bin

  copy:

    src: "/tmp/node\_exporter-{{ version }}.linux-amd64/node\_exporter"

    dest: "/usr/local/bin/node\_exporter"

    owner: "{{userId}}"

    group: "{{groupId}}"

    remote\_src: yes

    mode: 0755

This task copies the Node Exporter binary from the temporary directory to the **/usr/local/bin/** directory on the target host. It sets the owner and group to **node\_exporter**, ensures execute permissions (**mode: 0755**), and specifies that the source file is located remotely.

- name: Delete node exporter tmp folder

  file:

    path: '/tmp/node\_exporter-{{ version }}.linux-amd64'

    state: absent

This task deletes the temporary directory created during the unarchiving process.

- name: Copy systemd init file

  template:

    src: init.service.j2

    dest: /etc/systemd/system/node\_exporter.service

This task uses a Jinja2 template (assumed to be named **init.service.j2**) to generate a systemd unit file for the Node Exporter service and places it in the appropriate location.

- name: Start node\_exporter service

  service:

    name: node\_exporter

    state: started

    enabled: yes

This task starts the Node Exporter service and enables it to start on system boot.

- name: Check if node exporter emits metrices

  uri:

    url: http://127.0.0.1:9100/metrics

    method: GET

    status\_code: 200

This task sends an HTTP GET request to the Node Exporter metrics endpoint (**http://127.0.0.1:9100/metrics**) to verify if metrics are being emitted correctly. It checks that the response status code is 200 (OK).

These tasks collectively install, configure, and start the Prometheus Node Exporter on the target host as part of a larger Ansible playbook or role.

1. **Prometheus Role –**

Command - ansible-galaxy init roles/Prometheus

A Prometheus role in Ansible is a way to organize and encapsulate tasks, configurations, and files needed to install, configure, and manage Prometheus, an open-source monitoring and alerting system. Using a Prometheus role helps you create a modular and reusable approach to setting up Prometheus instances across different hosts or environments.

Main.yml in roles/prometheus/tasks –

It is responsible for deploying Prometheus, an open-source monitoring and alerting toolkit. These tasks perform various actions to install, configure, and start Prometheus. Let's break down each task step by step:

- name: Creating prometheus user group

  group: name="{{groupId}}"

  become: true

This task creates a user group named **prometheus** on the target host. The **become: true** statement indicates that Ansible should escalate its privileges (typically using **sudo**) to execute this task.

- name: Creating prometheus user

  user:

    name: "{{userId}}"

    group: "{{groupId}}"

    system: yes

    shell: "/sbin/nologin"

    comment: "{{userId}} nologin User"

    createhome: "no"

    state: present

This task creates a system user named **prometheus**. The user is added to the **prometheus** group, given the specified shell and user comment. The **system: yes** attribute indicates that it's a system user, and **createhome: "no"** specifies not to create a home directory for the user.

- name: Install prometheus

  unarchive:

    src: "https://github.com/prometheus/prometheus/releases/download/v{{ version }}/prometheus-{{ version }}.linux-amd64.tar.gz"

    dest: /tmp/

    remote\_src: yes

This task downloads the Prometheus binary release from the specified URL and unpacks it into the **/tmp/** directory on the target host. The **remote\_src: yes** attribute tells Ansible to fetch the source file from a remote location.

- name: Copy prometheus file to bin

  copy:

    src: "/tmp/prometheus-{{ version }}.linux-amd64/prometheus"

    dest: "/usr/local/bin/prometheus"

    owner: "{{userId}}"

    group: "{{groupId}}"

    remote\_src: yes

    mode: 0755

This task copies the Prometheus binary from the temporary directory to the **/usr/local/bin/** directory on the target host. It sets the owner and group to **prometheus**, ensures execute permissions (**mode: 0755**), and specifies that the source file is located remotely.

- name: Delete prometheus tmp folder

  file:

    path: '/tmp/prometheus-{{ version }}.linux-amd64'

    state: absent

This task deletes the temporary directory created during the unarchiving process.

- name: Creates directory

  file:

    path: "/data/prometheus/"

    state: directory

    owner: "{{userId}}"

    group: "{{groupId}}"

    mode: 0755

This task creates a directory named **/data/prometheus/** on the target host. It ensures the directory exists with the specified owner, group, and permissions.

- name: Creates directory

  file:

    path: "/etc/prometheus/"

    state: directory

    owner: "{{userId}}"

    group: "{{groupId}}"

    mode: 0755

This task creates a directory named **/etc/prometheus/** on the target host, ensuring it exists with the specified owner, group, and permissions.

- name: config file

  template:

    src: prometheus.conf.j2

    dest: /etc/prometheus/prometheus.conf

This task uses a Jinja2 template (assumed to be named **prometheus.conf.j2**) to generate a Prometheus configuration file and places it in the **/etc/prometheus/** directory.

- name: Copy systemd init file

  template:

    src: init.service.j2

    dest: /etc/systemd/system/prometheus.service

  notify: systemd\_reload

This task uses a Jinja2 template (assumed to be named **init.service.j2**) to generate a systemd unit file for the Prometheus service and places it in the appropriate location. The **notify: systemd\_reload** statement notifies Ansible to trigger a systemd reload after the file is copied.

- name: Start prometheus service

  service:

    name: prometheus

    state: started

    enabled: yes

This task starts the Prometheus service and enables it to start on system boot.

- name: Check if prometheus is accessible

  uri:

    url: http://localhost:9090

    method: GET

    status\_code: 200

This task sends an HTTP GET request to the Prometheus web UI (**http://localhost:9090**) to verify

1. **Grafana Role –**

Command - ansible-galaxy init roles/Grafana

A Grafana role in Ansible is a way to organize and encapsulate tasks, configurations, and files needed to install, configure, and manage Grafana, an open-source platform for data visualization and dashboard creation. Using a Grafana role helps you create a modular and reusable approach to setting up Grafana instances across different hosts or environments.

Main.yml in roles/grafana/tasks –

It is responsible for installing and configuring Grafana, an open-source platform for monitoring and observability. These tasks perform various actions to install Grafana, add the necessary repository, and start the Grafana service. Let's break down each task step by step:

- name: Install nessesary package

  apt:

      name: apt-transport-https

      state: present

      update\_cache: yes

This task installs the **apt-transport-https** package on the target host, which is required for securely handling HTTPS-based repositories. The **update\_cache: yes** option updates the local package cache before performing the installation.

- name: add grafana gpg key

  shell: curl https://packages.grafana.com/gpg.key | sudo apt-key add -

This task uses the **curl** command to fetch the GPG key from Grafana's package repository and then adds the key to the host's trusted keyring using the **apt-key add** command. This step is necessary to verify the authenticity of packages from the Grafana repository.

- name: add grafana repo

  apt\_repository:

    repo: deb https://packages.grafana.com/oss/deb stable main

    state: present

    filename: grafana

This task adds the Grafana repository to the list of APT repositories on the target host. It specifies the repository URL and indicates that the repository should be in a "present" state. The **filename** attribute defines the name of the repository file that will be created in the **/etc/apt/sources.list.d/** directory.

- name: Install grafana

  apt:

      name: grafana

      state: present

      update\_cache: yes

This task installs the **grafana** package on the target host using the APT package manager. The **update\_cache: yes** option updates the local package cache before performing the installation.

- name: Enable and start grafana service

  service:

    name: grafana-server

    enabled: yes

    state: started

This task enables and starts the Grafana service (**grafana-server**). It ensures that the service is set to start on system boot (**enabled: yes**) and then starts the service (**state: started**).

Here's an example of how you might structure a role directory:

grafana\_role/

├── defaults/

│ └── main.yml

├── handlers/

├── tasks/

│ └── main.yml ├

── templates/

├── vars/

│ └── main.yml

└── README.md

* **defaults**: Contains default variables for the role.
* **files**: Could include configuration files, scripts, or other files needed for that role setup.
* **handlers**: Defines handlers, such as service restarts, triggered by tasks.
* **tasks**: Defines the main tasks for installing and configuring that role, setting up data sources, dashboards, and plugins.
* **templates**: Stores template files for Grafana configuration.
* **vars**: Contains variable definitions for the role.
* **README.md**: Documentation explaining the purpose and usage of the role.

**Steps for playbook.yml -**

The provided Ansible playbook consists of three main sections, each targeting different groups of hosts and applying specific roles. Let us break down the playbook step by step

---

- hosts: node\_exporter

  connection: local

  become: yes

  become\_user: root

  become\_method: sudo

  roles:

    - prometheus\_node\_exporter

In this section:

* `hosts: node\_exporter` specifies that the playbook tasks will be executed on the hosts belonging to the `node\_exporter` group. These are are hosts where you want to deploy and configure the Node Exported for Prometheus monitoring.
* `connection: local` indicates playbook will be run locally, not remotely over SSH which means that playbook is meant to configure the local machine where Ansible is running.
* `become: yes` indicates that the playbook tasks will be executed with escalated privileges (root)
* `become\_user: root` specifes the user to become when escalating priviledges.
* `become\_method: sudo` speicifies the method to use for privilege escalation.
* `roles:` defines the roles that will be applied to the hosts in the `Prometheus\_node\_exporter` group.

The playbook applies the same structure to the next two sections for **prometheus** and **grafana**:

- hosts: prometheus

  connection: local

  become: yes

  become\_user: root

  become\_method: sudo

  roles:

    - prometheus

- hosts: grafana

  connection: local

  become: yes

  become\_user: root

  become\_method: sudo

  roles:

    - grafana

These sections target the **prometheus** and **grafana** groups, respectively. They execute roles named **prometheus** and **grafana**, which are expected to contain the necessary tasks and configurations for deploying and setting up Prometheus and Grafana on the local machine (or the machine where Ansible is being run).

Roles are reusable units of work in Ansible that encapsulate tasks, variables, and configurations related to a specific component or function. The idea behind the playbook is to use roles to modularize the configuration and setup of different components (Node Exporter, Prometheus, and Grafana)s and apply them to appropriate host groups.

Overall, this playbook suggests a setup where Node Exporter, Prometheus, and Grafana are all deployed and configured on the same local machine where Ansible is running.

**Steps for Running the project**

1. Clone the repository.
2. Modify the IP address in inventory file accordingly for use case
3. Run the command - `ansible-playbook` -i inventory playbook.yml`

The command **ansible-playbook -i inventory playbook.yml** will execute an Ansible playbook using the specified inventory file (**inventory**) and playbook file (**playbook.yml**). The inventory file defines the target hosts or groups of hosts on which the playbook tasks will be executed. The playbook file contains a series of tasks and instructions to be carried out on those target hosts.

Note – If you want multiple hosts then below is the example of inventory file for multiple hosts –

[monitorserver]

db\_server   ansible\_host=<YOUR-DB-SERVER-IP>   ansible\_user=ec2-user  ansible\_ssh\_private\_key\_file=~/<YOUR-PEM-FILE>

[nodeservers]

server1  ansible\_host=<YOUR-WEB-SERVER-IP>  ansible\_user=ec2-user  ansible\_ssh\_private\_key\_file=~/<YOUR-PEM-FILE>

server2  ansible\_host=<YOUR-WEB-SERVER-IP>  ansible\_user=ec2-user  ansible\_ssh\_private\_key\_file=~/<YOUR-PEM-FILE>

Here you can add path to your public key in <YOUR-PEM-FILE> and ansible will use specified SSH key for authentication

Expected Output –

1. Ansible's standard output: During the execution of the playbook, Ansible will display output for each task it is running. This output includes task names, status (whether tasks were changed or not), and any additional debug or verbose information if enabled.
2. Progress updates: You'll see progress updates for each task as Ansible processes them. It will show which task is being executed and on which host.
3. Task results: For each task, you'll see whether the task was executed ("changed") or skipped because the system was already in the desired state ("ok").
4. Summary: Once the playbook execution is complete, Ansible will provide a summary that includes the number of hosts affected, the number of tasks executed, the number of tasks that resulted in changes, and the total runtime.

**Steps for Grafana –**

1. Go to url – localhost:3000
2. Add new Connection prometheus and enter url as – localhost:9090
3. Enter default username and password as “admin” for both.
4. Go to dashboard and click on new -> import and enter the dashboard id mentioned in <https://grafana.com/grafana/dashboards/1860-node-exporter-full/>
5. This will create a Grafana dashboard which consists of visualizations , panels , widgets that display metrics and data collected by Prometheus from target hosts.

(Note – Output Screenshot are added at the end of the documentation)

**Troubleshooting steps**

1. Check for IP in inventory file
2. Go to prometheus.conf.j2 in roles/prometheus/templates and add a list in target for job\_name: ‘node\_exporter’ containing your host ip and port number 9090 for prometheus

Note – if you have multiple target host then add the ip and port to the target list only.

**\*\* For personal Testing, I have used Ubuntu 22.04 hosted using AWS EC2 instance and clone the repository there and configured the security group to allow my IP to access the server accordingly. I have attached the output screenshot below:**

**Results**

**Ansible Output -**



**Grafana Dashboard Output –**





