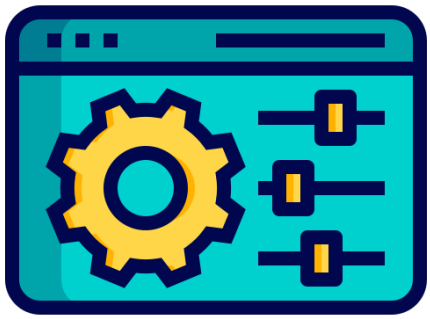

Configuration Management with Ansible

Prepared By:
Mahmoud Alsalloum

Table of contents

Contents

Table of contents 2
Technology Stack 3
Project Overview 3
Implementation Steps 4

Technology Stack

Ubuntu Server – My personal home server where the entire project was executed

Ansible – Used locally to automate the configuration

NTP (Network Time Protocol) – Ensures accurate system time

Systemd – Manages and monitors the NTP service

Localhost – Since the automation runs on the same server

Project Overview

Welcome to my project!

Thank you for visiting my portfolio website and taking the time to explore my work. Time synchronization is a fundamental part of system administration. Without accurate system time, logs can become unreliable, scheduled jobs may fail, and services that depend on time (like TLS certificates or authentication tokens) may not function correctly. In this project, I used Ansible to automate the setup of NTP on my Ubuntu server. Since I only have one server at home, I installed Ansible directly on that server and used localhost as the managed host. The goal was to demonstrate my ability to automate system configurations using modern tools, even in a simple environment. This setup mimics how automation works in larger infrastructures, but on a smaller and more practical scale for my learning and personal use.

Implementation Steps

I started by installing Ansible on the same Ubuntu server that I planned to configure. See Figure 1.

Figure 1: Installing Ansible

This gave me everything I needed to start building and running playbooks locally.

To keep things organized, I created a dedicated directory for the project. All files related to this automation - including the inventory file, playbook, and NTP configuration – were stored here. Even though I'm working on a single server, Ansible still requires an inventory. I created a file named inventory and added the following content in Figure 2.

Figure 2: Inventory file

This tells Ansible to apply the playbook to the local machine using a direct connection.

Next, I created the main playbook, which installs NTP, uploads the configuration file, and ensures the service is always running. And then I saved and closed the file. See Figure 3.

```
  GNU nano 7.2                                  ntp-setup.yml *
- name: Configure NTP on my Ubuntu Server
  hosts: local
  become: true

  tasks:
    - name: Install NTP package
      apt:
        name: ntp
        state: present
        update_cache: yes

    - name: Upload custom NTP configuration
      copy:
        src: ntp.conf
        dest: /etc/ntp.conf
        owner: root
        group: root
        mode: '0644'
        backup: yes

    - name: Enable and start the NTP service
      systemd:
        name: ntp
        enabled: yes
        state: started
```

Figure 3: Ansible-Playbook

Now I wrote a basic, but secure NTP configuration that defines which time servers to use. This configuration provides a balanced setup with security restrictions. See Figure 4.

```
  GNU nano 7.2                                        ntp.conf *
driftfile /var/lib/ntp/ntp.drift

pool 0.ubuntu.pool.ntp.org iburst
pool 1.ubuntu.pool.ntp.org iburst
pool 2.ubuntu.pool.ntp.org iburst
pool 3.ubuntu.pool.ntp.org iburst

restrict -4 default kod notrap nomodify nopeer noquery
restrict -6 default kod notrap nomodify nopeer noquery
```

Figure 4: NTP configuration

With everything ready, I ran the playbook directly on my server. Ansible executed each task installing the NTP package, uploading the configuration, and enabling the service. See Figure 5.

```
mahmoud@ubuntu-server:~/ansible-ntp-setup$ ansible-playbook -i inventory ntp-setup.yml

PLAY [Configure NTP on my Ubuntu Server] ***************************************************

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

TASK [Install NTP package] *****************************************************************
changed: [localhost]

TASK [Upload custom NTP configuration] *****************************************************
changed: [localhost]

TASK [Enable and start the NTP service] ****************************************************
ok: [localhost]

PLAY RECAP *********************************************************************************
localhost                  : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0
   ignored=0
```

Figure 5: Run Ansible-Playbook

After the playbook ran successfully, I checked if the NTP service was working and synchronized. I ran the command ntpq -p. This displayed a list of remote time servers and their status, confirming that my Ubuntu server was correctly syncing with external source. See Figure 6.

```
mahmoud@ubuntu-server:~/ansible-ntp-setup$ ntpq -p
     remote                                refid      st t when poll reach   delay   offset   jitter
==============================================================================================
 0.ubuntu.pool.ntp.org                .POOL.          16 p    -  256    0   0.0000   0.0000   0.0002
 1.ubuntu.pool.ntp.org                .POOL.          16 p    -  256    0   0.0000   0.0000   0.0002
 2.ubuntu.pool.ntp.org                .POOL.          16 p    -  256    0   0.0000   0.0000   0.0002
 3.ubuntu.pool.ntp.org                .POOL.          16 p    -  256    0   0.0000   0.0000   0.0002
 prod-ntp-5.ntp4.ps5.canonical.com    183.160.133.132  2 u   37   64    1  22.2555  -1.4201   0.0000
+a.ntp.madduck.net                    131.188.3.222    2 u   31   64    1  17.2847   0.2531   0.6878
+mail.masters-of-cloud.de             205.46.178.169   2 u   31   64    1  14.0961   0.1678   1.0363
+srv01.spectre-net.de                 130.149.17.21    2 u   31   64    1  10.1117  -0.3882   0.8795
+vps-nue1.orleans.ddnss.de            68.126.43.53     2 u   31   64    1  14.3726   0.3737   1.2113
+time2.sebhosting.de                  189.97.54.122    2 u   30   64    1  10.9235   0.8990   0.9039
#217.160.19.219                      10.50.0.2        2 u   30   64    1  10.9675   1.2078   0.8704
+ntp1.kashra-server.com               237.17.204.95    2 u   30   64    1  12.6030  -0.1075   0.8355
+stage3.opensuse.org                  127.51.226.51    3 u   30   64    1  15.9459  -0.3757   0.8871
+2003:a:42b:e400::3                   237.17.204.95    2 u   29   64    1  14.9923  -0.0346   0.7544
+2003:a:47f:abe4:48ba:cd42:dbcc:1000  237.17.204.95    2 u   29   64    1  14.8036   0.0750   0.7195
+ntp0-de-fks.inps-jung.net            191.45.67.67     3 u   29   64    1  15.2857   0.2594   0.6663
*2a03:4000:5:e51:123:123:123:123      68.126.43.53     2 u   29   64    1  13.6645   0.0128   1.0457
+netcup01.theravenhub.com             131.188.3.222    2 u   29   64    1  13.8105   0.2249   0.7368
+t1.ipfu.de                           195.201.20.16    3 u   29   64    1  14.3808   0.4552   0.8378
#ctb01.martinmoerch.dk                80.209.87.103    2 u   29   64    1  20.2541  -3.2581   0.6376
+vps-fra2.orleans.ddnss.de            169.254.169.254  3 u   29   64    1  10.2982   0.0692   1.8056
#vps-fra1.orleans.ddnss.de            68.126.43.53     2 u   28   64    1  24.3567   6.9335   3.5237
+x1.ncomputers.org                    82.64.42.185     2 u   28   64    1  13.7820   0.0806   0.5786
#ntp2.kashra-server.com               237.17.204.95    2 u   28   64    1  19.3103  -0.0850   1.6092
+130.162.222.153                     169.254.169.254  3 u   28   64    1  11.1402   0.0623   0.5768
```

Figure 6: The status of NTP service