UNIVERSITY
of
INFORMATION TECHNOLOGY

File and Service Management with Ansible

Presented by Group (7)

Team Members

Shin Min Khant	(4SE-1081)
Thiri Minn Thu	(4SE-1265)
Lin Thawtar Tin	(4SE-1924)
Hsu Wai Htet	(4BIS-909)
Htet Htet Oo	(4SE-924)
Swann Htet Moe Lwin	(4BIS-929)

Table of Contents

I. Introduction to Configuration Management

Configuration management is a system engineering process for establishing and maintaining consistency of a product's performance, functional, and physical attributes with its requirements, design, and operational information throughout its life. It is the process of maintaining systems, such as computer hardware and software, in a desired state. Configuration Management (CM) is also a method of ensuring that systems perform in a manner consistent with expectations over time.

Advantages of Configuration Management System

1. Consistency
2. Efficiency & Risk Reduction
3. Scalability
4. Disaster Recovery

II. Prerequisites

Install Ansible on a control node, which then uses SSH (by default) to communicate with your managed nodes (those end devices you want to automate).

Control node requirements

Currently Ansible can be run from any machine with Python 2 (version 2.7), or Python 3 (versions 3.5 and higher) installed. This includes Red Hat, Debian, CentOS, macOS, any of the BSDs, and so on. Windows is not supported for the control node.

When choosing a control node, bear in mind that any management system benefits from being run near the machines being managed. If you are running Ansible in a cloud, consider running it from a machine inside that cloud. In most cases this will work better than on the open Internet. The macOS by default is configured for a small number of file handles, so if you want to use 15 or more forks, you'll need to raise the limits with sudo launchctl limit maxfiles unlimited. This command can also fix any "Too many open files" error. Note that some modules and plugins have additional requirements. For modules these need to be satisfied on the 'target' machine (the managed node) and should be listed in the module specific docs.

Managed Node Requirements

On the managed nodes, you need a way to communicate, which is normally SSH. By default, this uses SFTP. If that's not available, you can switch to SCP in ansible.cfg. You also need Python 2 (version 2.6 or later) or Python 3 (version 3.5 or later).

If you have SELinux enabled on remote nodes, you will also want to install libselinux-python on them before using any copy/file/template related functions in Ansible. You can use the yum module or dnf module in Ansible to install this package on remote systems that do not have it.

By default, Ansible uses the Python interpreter located at /usr/bin/python to run its modules. However, some Linux distributions may only have a Python 3 interpreter installed to /usr/bin/python3 by default. On those systems, you may see an error like:

"module_stdout": "/bin/sh: /usr/bin/python: No such file or directory\r\n"

Can either set the ansible_python_interpreter inventory variable (see How to build your inventory) to point at your interpreter or you can install a Python 2 interpreter for modules to use. You will still need to set ansible_python_interpreter if the Python 2 interpreter is not installed to /usr/bin/python.

Ansible's raw module, and the script module, do not depend on a client-side installation of Python to run. Technically, you can use Ansible to install a compatible version of Python using the raw module, which then allows you to use everything else. For example, if you need to bootstrap Python 2 onto a RHEL-based system, you can install it as follows:

```
$ ansible myhost --become -m raw -a "yum install -y python2"
```

III. Introduction to Ansible

Ansible is simple open-source IT engine which automates application deployment, intra service orchestration, cloud provisioning and many other IT tools.

Ansible is easy to deploy because it does not use any agents or custom security infrastructure.

Ansible uses playbook to describe automation jobs, and playbook uses very simple language i.e., YAML (It's a human-readable data serialization language & is commonly used for configuration files but could be used in many applications where data is being stored) which is very easy for humans to understand, read and write. Hence the advantage is that even the IT infrastructure support guys can read and understand the playbook and debug if needed (YAML – It is in human readable form).

Ansible is designed for multi-tier deployment. Ansible does not manage one system at time, it models IT infrastructure by describing all your systems are interrelated. Ansible is completely agentless which means Ansible works by connecting your nodes through ssh(by default). But if you want other method for connection like Kerberos, Ansible gives that option to you.

After connecting to your nodes, Ansible pushes small programs called as "Ansible Modules". Ansible runs those modules on your nodes and removes them when finished. Ansible manages your inventory in simple text files (These are the hosts file). Ansible uses the hosts file where one can group the hosts and can control the actions on a specific group in the playbooks.

IV. What is Ansible?

Ansible is an open-source IT automation tool that provides a simple and powerful way to automate tasks across a wide range of IT environments, including servers, networks, cloud infrastructure, and more.
It was created by Michael DeHaan and released in 2012.
Ansible is designed to streamline complex tasks, simplify infrastructure management, and enhance collaboration among IT teams.

1. Advantages of Ansible

Simple to Learn

The foremost mention among advantages of Ansible refers to its simplicity. Simplicity is not only meant for professionals but also for beginners. It is easy to learn, and so, users could learn to use Ansible quickly along with better productivity. Ansible receives the support of comprehensive and easily interpretable documentation.

Therefore, you can learn the logic of Ansible operations and the workflow in a limited period. The lack of a dependency system could imply that Ansible tasks execute sequentially and stop when identifying an error. As a result, troubleshooting becomes a lot easier, even in the initial stages of learning about ansible.

Easily Understandable Python Language

One of the prominent advantages of Ansible also refers to the language in which it is written. Python is a human-readable language and serves as the basis for Ansible. It provides better facilities for getting up Ansible and running it due to the presence of Python libraries on most Linux distributions by default.

Python is a highly ideal alternative for administration and scripting tasks implying higher popularity among engineers and system administrators. Another interesting aspect of Ansible is the facility of Ansible modules that can improve its functionality. The Ansible modules can be written in any language. However, the important concern, in this case, is that the module should return data in JSON format.

No Dependency on Agents

The next important addition among the benefits of Ansible refers to its agentless nature. Ansible manages all the master-agent communications through Standard SSH or Paramiko module. The Paramiko module is a Python implementation of SSH2 and is crucial for managing nodes. Therefore, Ansible does not require any form of agents installed on remote systems for ensuring management. As a result, maintenance overheads and performance degradations are reduced considerably by huge margins with Ansible.

Playbooks are written in YAML

The use of Playbooks in Ansible is also another reason for the major advantages of Ansible. Playbooks are Ansible configuration files, and the language for writing them is YAML. The interesting factor, in this case, is that YAML is a better alternative for configuration management and automation.

The superiority of YAML over other formats like JSON makes Ansible better configuration management and automation tool. Ansible makes it easy to read and supports comments. Most important of all, it also includes the use of anchors to reference other items.

Ansible Galaxy

Another notable entry in the Ansible best practices refers to the Ansible Galaxy. Ansible Galaxy is a portal that acts as the central repository for locating, reusing, and sharing Ansible-related content. The best advantage of Ansible Galaxy is in the example of downloading reusable Roles for installing application or server configuration. The downloads are ideal for use in a particular user's playbooks and can contribute substantially to an increase in deployment speed.

2. Disadvantages of Ansible

Insufficient User Interface

The first entry in the disadvantages of Ansible is the crude user interface. Ansible was initially a command-line only tool. The first effort of Ansible at making a user interface was with AWX graphical user interface. The other component in the UI was the REST endpoint that is meant for easier infrastructure management.

Subsequently, the AWX turned into the Ansible Tower, which is a web management UI. Ansible Tower offers visual management features and a team-based workflow instrument. However, the Ansible Tower requires considerable improvements. For example, almost 85% of tasks that could be completed through the command line can be achieved through the UI.

You could also come across another mention of Ansible disadvantages arising from its UI. The failure of synchronization between the GUI and the command line can lead to conflicting query results. On a general basis, Ansible Tower is still in the development stages and could not do everything like a command-line interface.

Lack of any Notion of State

Another prominent mention among the disadvantages of Ansible is the lack of any notion of state. Ansible does not have any notion of state like other automation tools such as Puppet. Ansible does not track dependencies and simply executes sequential tasks and stops when tasks finish, fail, or any error comes.

These traits are not ideal for users who want the automation tool to maintain a detailed catalog for ordering. The catalog can help in reaching a specific state without any influence of changes in environmental conditions. However, Ansible lacks it and presents a formidable disadvantage.

Limited Windows Support

The next prominent mention among Ansible disadvantages is the half-built Windows support. Ansible version 1.7 supports Windows as well as Linux/Unix nodes. In the case of Windows, Ansible employs a native PowerShell remote rather than SSH. As a result, a Linux control machine is mandatory for the management of Windows hosts. The limited support for Windows in Ansible presents one of the formidable setbacks with the configuration management and automation tools.

Ansible does not have Experience

The lack of enterprise support experience also draws down the appeal of Ansible. Ansible does not have full-fledged working experience with large enterprises like its competitors, such as Puppet and Chef (Read: Chef vs Puppet). Even though Ansible claims the facility of enterprise-grade extended support options, limited practical experience reduces the accountability of Ansible.

Ansible is New to the Market

Finally, you can note one of the most common entries in Ansible advantages and disadvantages as a prominent setback of Ansible. Ansible is new to the market, unlike its renowned competitors. As a result, it does not have a large developer or user community. Furthermore, the new presence of Ansible on the market implies the possibilities of undiscovered bugs, software issues, and edge scenarios.

3. Key Features of Ansible

- Agentless
- Idempotent
- Scalability
- Declarative
- Extensibility
- Community
- Playbooks
- Integration
- Eco-system

4. Ansible Applied Areas

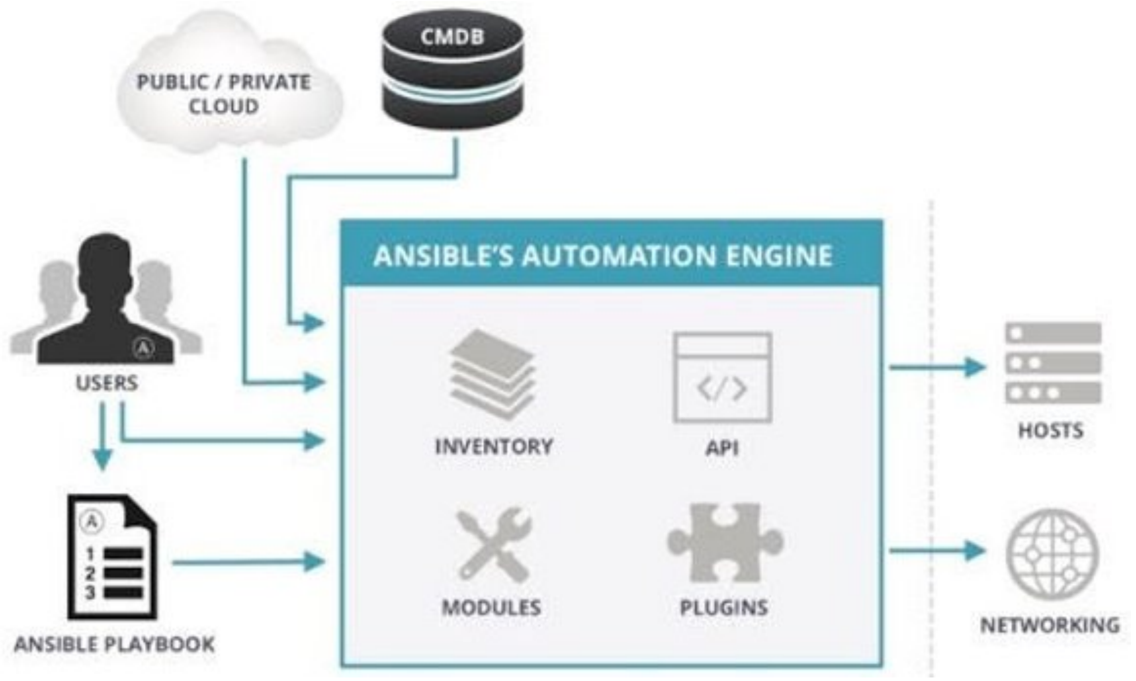
- Configuration Management
- Provisioning
- Continuous Integration/Continuous Deployment (CI/CD)
- Infrastructure as Code (IaC)
- Security Automation
- Network Automation
- Data Management

5. Steps of How Ansible Works

1. Installation
2. Inventory Setup
3. SSH Key Steup
4. Playbooks
5. Tasks and Modules
6. Running Playbooks
7. Gathering Facts

8. SSH Connections
9. Idempotence
10. Reporting and Output
11. Declarative Configuration Management

V. Ansible Architecture



Inventory: Inventory is lists of nodes or hosts having their IP addresses, databases, servers, etc. which need to be managed.

API's: The Ansible API's works as the transport for the public or private cloud services.

Modules: Ansible connected the nodes and spread out the Ansible modules programs. Ansible executes the modules and removed after finished. These modules can reside on any machine; no database or servers are required here. You can work with the chose text editor or a terminal or version control system to keep track of the changes in the content.

Plugins: Plugins is a piece of code that expends the core functionality of Ansible. There are many useful plugins, and you also can write your own.

Playbooks: Playbooks consist of your written code, and they are written in YAML format, which describes the tasks and executes through the Ansible. Also, you can launch the tasks synchronously and asynchronously with playbooks.

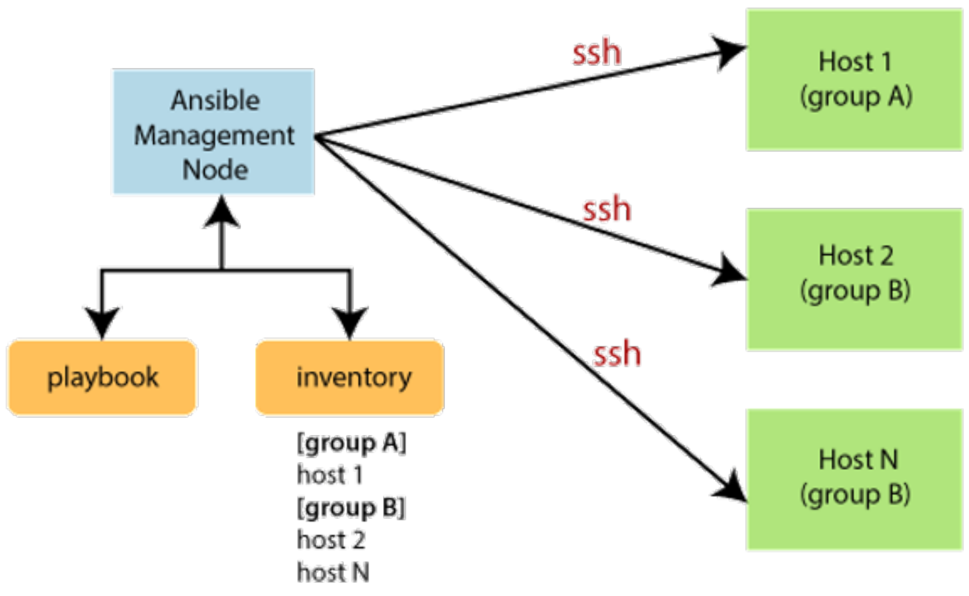
Hosts: In the Ansible architecture, hosts are the node systems, which are automated by Ansible, and any machine such as RedHat, Linux, Windows, etc.

Networking: Ansible is used to automate different networks, and it uses the simple, secure, and powerful agentless automation framework for IT operations and development. It uses a type of data model which separated from the Ansible automation engine that spans the different hardware quite easily.

Cloud: A cloud is a network of remote servers on which you can store, manage, and process the data. These servers are hosted on the internet and storing the data remotely rather than the local server. It just launches the resources and instances on the cloud, connect them to the servers, and you have good knowledge of operating your tasks remotely.

CMDB: CMDB is a type of repository which acts as a data warehouse for the IT installations.

1. Ansible workflow



2. Ansible Playbooks

Playbooks are the files where the Ansible code is written. Playbooks are written in YAML format. YAML means "Yet Another Markup Language," so there is not much syntax needed.

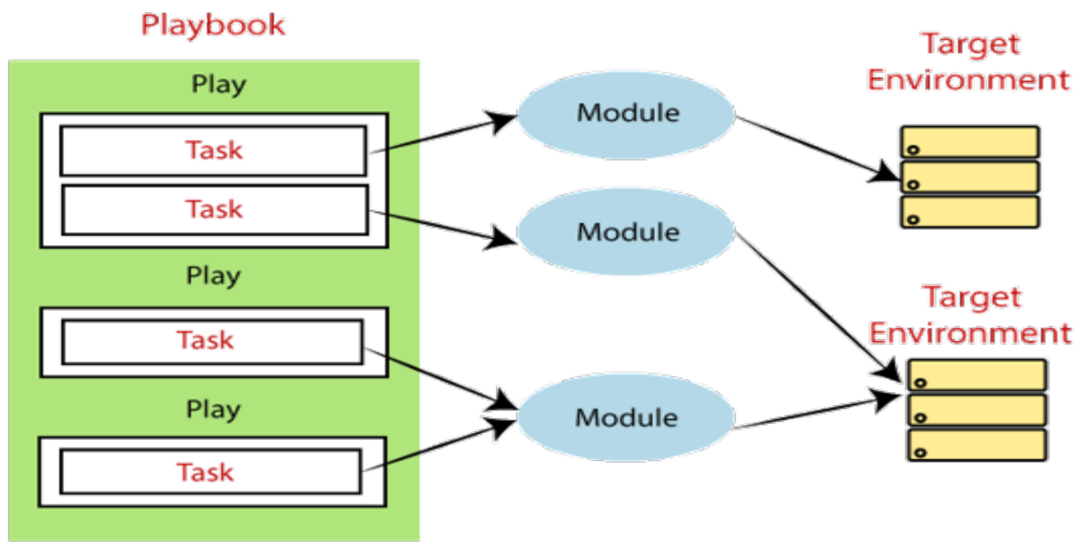
Playbooks are one of the core features of Ansible and tell Ansible what to execute, and it is used in complex scenarios. They offer increased flexibility.

Playbooks contain the steps which the user wants to execute on a particular machine. And playbooks are run sequentially. Playbooks are the building blocks for all the use cases of Ansible.

Ansible playbooks tend to be more configuration language than a programming language.

Through a playbook, you can designate specific roles to some of the hosts and other roles to other hosts. By doing this, you can orchestrate multiple servers in very different scenarios, all in one playbook.

3. Playbook Structure



4. Playbook Example

```
▪ ---
▪   - name: install and configure DB
▪     hosts: testServer
▪     become: yes
▪
```

```
▪       vars:
▪          oracle_db_port_value : 1521
▪
▪     - tasks:
▪     - name: Install the Oracle DB
▪       yum: <code to install the DB>
▪
▪     - name: Ensure the installed service is enabled and running
▪       service:
▪          name: <your service name>
```

VI. Ansible Installation

We can only support the operating systems indicated above. Our instructions have been tested with MacOS, Ubuntu, and official flavours of Ubuntu. We do not recommend installing an OS that is only based on Ubuntu (like Mint, Pop!_OS, ElementaryOS, etc).

Virtual Machine (Recommended)

Installing a Virtual Machine (VM) is the easiest and most reliable way to get started creating an environment for web development. A VM is an entire computer emulation that runs inside your current Operating System (OS), like Windows. The main drawback of a VM is that it can be slow because you're essentially running two computers at the same time. We'll do a few things to improve its performance.

Step 1: Download VirtualBox and Xubuntu

Installing a VM is a simple process. This guide uses Oracle's VirtualBox program to create and run the VM. This program is open-source, free, and simple. What more can you ask for? Now, let's make sure we have everything downloaded and ready for installation.

Once you have completed these instructions, you are expected to work entirely in the VM. Maximize the window, add more virtual monitors if you have them, fire up the Internet Browser in the Whisker Menu on the top left of the desktop. You should not be using anything outside of the VM while working on The Odin Project. If you feel like you have a good understanding after using the VM for a while, and or want to improve your experience, we recommend dual-booting Ubuntu, which there are instructions for below.

Step 1.1: Download VirtualBox

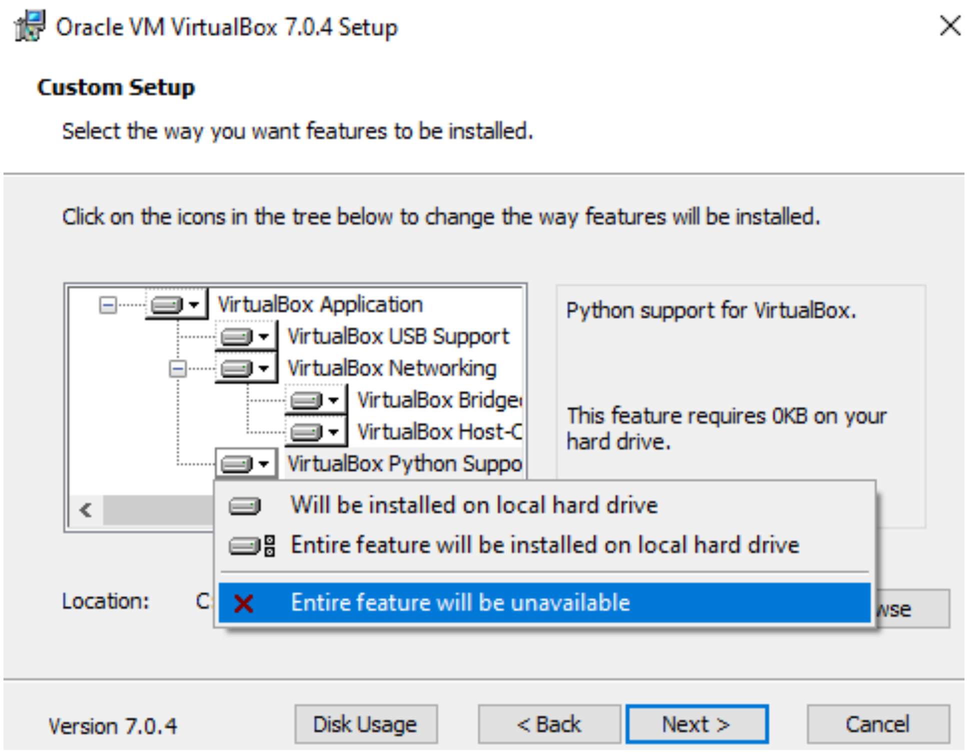
Step 1.2: Download Xubuntu There are thousands of distributions of Linux out there, but Xubuntu is undoubtedly one of the most popular and user friendly. When installing Linux on a VM, we recommend downloading Xubuntu 22.04. There are a few files listed here, download the one ending in .iso. Xubuntu uses the same base software as Ubuntu but has a desktop environment that requires fewer computer resources and is therefore ideal for virtual machines. If you find the download speed slow, consider using a different mirror as the one linked before is a US one. If you reach the download page and are unsure about what version to choose, it is recommended that you pick the latest Long-Term Support (LTS) version (22.04 at the time of writing). You may be tempted to choose a more recent non-LTS release, but LTS releases have the advantage of guaranteed support for up to 5 years, making them more secure, stable and hence reliable.

Step 2: Install VirtualBox and set up Xubuntu

Step 2.1: Install VirtualBox

Installing VirtualBox is very straightforward. It doesn't require much technical knowledge and is the same process as installing any other program on your Windows computer. Double clicking the downloaded VirtualBox file will start the installation process. If you receive an error about needing Microsoft Visual C++ 2019 Redistributable Package, you can find it on official Microsoft Learn page. You most likely want the version with X64 Architecture (that means 64-bit) - download and install it then try installing VirtualBox again.

During the installation, you'll be presented with various options. We suggest dropping the Python Support as you don't need it by clicking on the drive icon with an arrow and choosing Entire feature will be unavailable:


Oracle VM VirtualBox 7.0.4 Setup

Custom Setup

Select the way you want features to be installed.

Click on the icons in the tree below to change the way features will be installed.

- VirtualBox Application
  - VirtualBox USB Support
  - VirtualBox Networking
    - VirtualBox Bridge
    - VirtualBox Host-C
  - VirtualBox Python Suppo

Python support for VirtualBox.

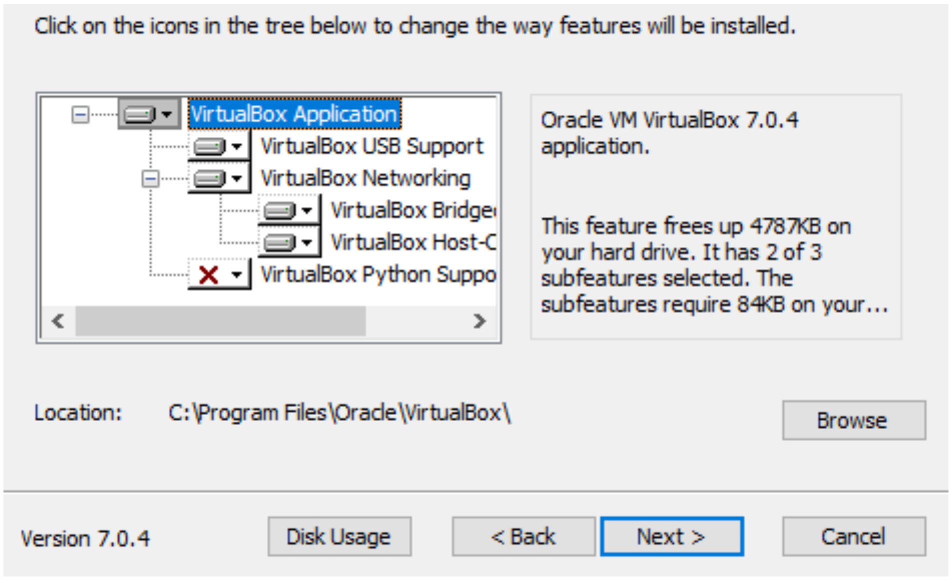
This feature requires 0KB on your hard drive.

- Will be installed on local hard drive
- Entire feature will be installed on local hard drive
- Entire feature will be unavailable

Location: C

Version 7.0.4

This is how your installation window should look like after turning it off:


Click on the icons in the tree below to change the way features will be installed.

- VirtualBox Application
  - VirtualBox USB Support
  - VirtualBox Networking
    - VirtualBox Bridge
    - VirtualBox Host-C
  - VirtualBox Python Suppo

Oracle VM VirtualBox 7.0.4 application.

This feature frees up 4787KB on your hard drive. It has 2 of 3 subfeatures selected. The subfeatures require 84KB on your…

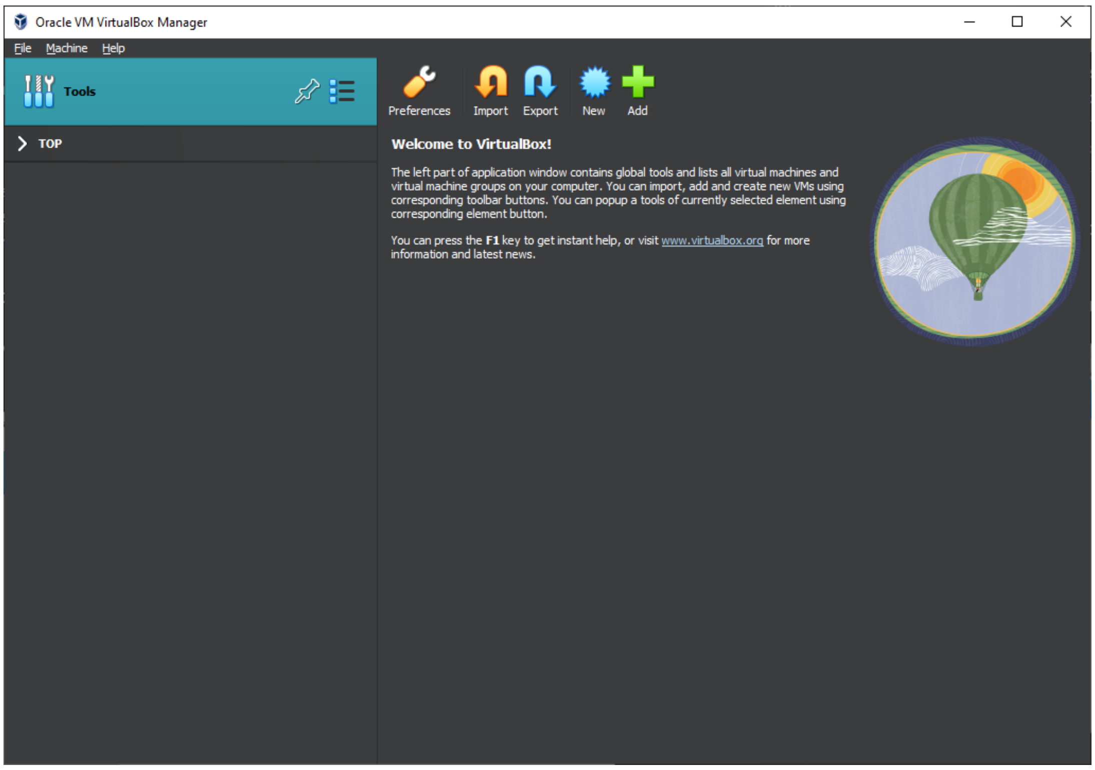
Location: C:\Program Files\Oracle\VirtualBox\

Version 7.0.4

Make sure you install the application on C: drive, as it has tendency to error out otherwise. The virtual machine itself can be installed anywhere but we'll get to that soon. As the software installs, the progress bar might appear to be stuck; just wait for it to finish.

Step 2.2: Prepare VirtualBox for Xubuntu

Now that you have VirtualBox installed, launch the program. Once open, you should see the start screen.

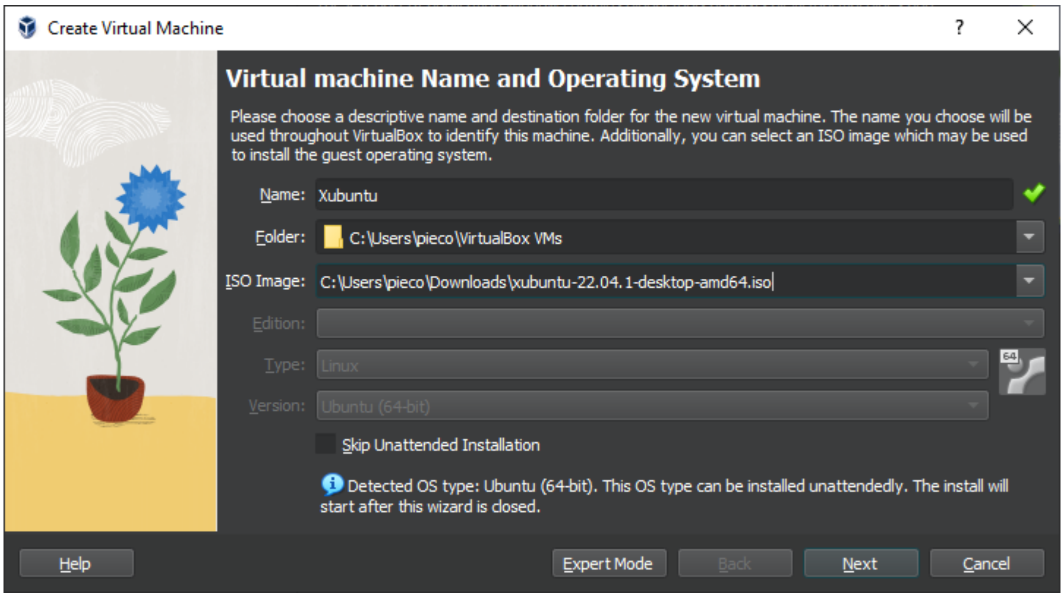


Welcome to VirtualBox!

The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can import, add and create new VMs using corresponding toolbar buttons. You can popup a tools of currently selected element using corresponding element button.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for more information and latest news.

Click on the New button to create a virtual operating system. Give it a name of Xubuntu, if you want the VM installed somewhere else than default C: location, change that accordingly in the Folder option. This is the place where your virtual disk will reside, so make sure that you've got at least 30GB for that. In ISO Image choose Other - you'll see a window open for you to find the .iso file on your PC. It most likely is in the Downloads folder. Leave Skip Unattended Installation as it is.


Create Virtual Machine

Virtual machine Name and Operating System

Please choose a descriptive name and destination folder for the new virtual machine. The name you choose will be used throughout VirtualBox to identify this machine. Additionally, you can select an ISO image which may be used to install the guest operating system.

Name: Xubuntu

Folder: C:\Users\pieco\VirtualBox VMs

ISO Image: C:\Users\pieco\Downloads\xubuntu-22.04.1-desktop-amd64.iso

Edition:

Type: Linux

Version: Ubuntu (64-bit)

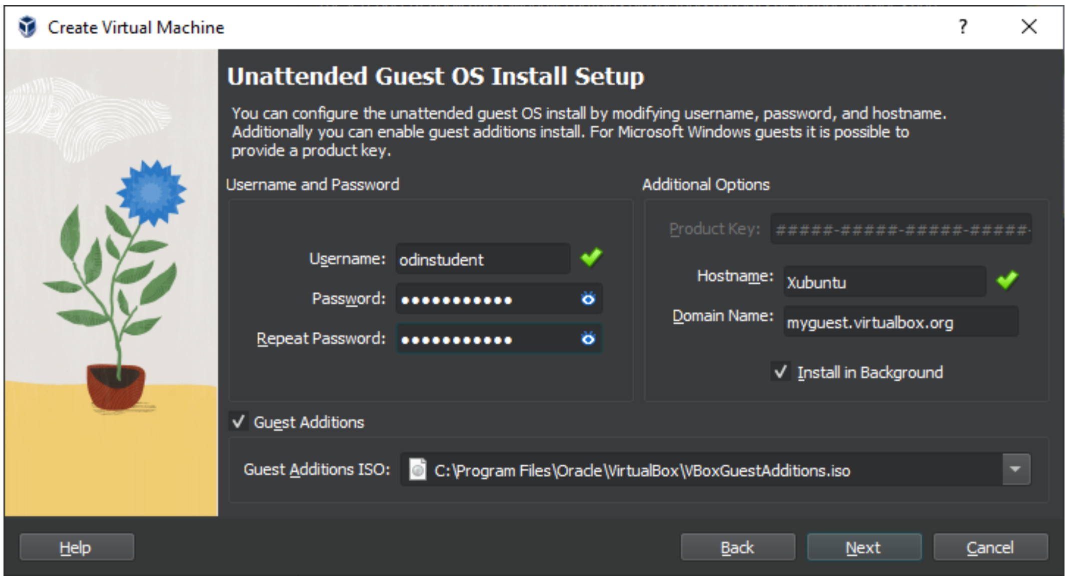
Skip Unattended Installation

Detected OS type: Ubuntu (64-bit). This OS type can be installed unattendedly. The install will start after this wizard is closed.

Continue by pressing Next and follow the next steps:

Step 2.2.1: Unattended guest OS install setup

You should see a window like this one now:


Create Virtual Machine

Unattended Guest OS Install Setup

You can configure the unattended guest OS install by modifying username, password, and hostname. Additionally you can enable guest additions install. For Microsoft Windows guests it is possible to provide a product key.

Username and Password

Username: odinstudent

Password: ●●●●●●●●●●●

Repeat Password: ●●●●●●●●●●●

Additional Options

Product Key: #####-#####-#####-#####-

Hostname: Xubuntu

Domain Name: myguest.virtualbox.org

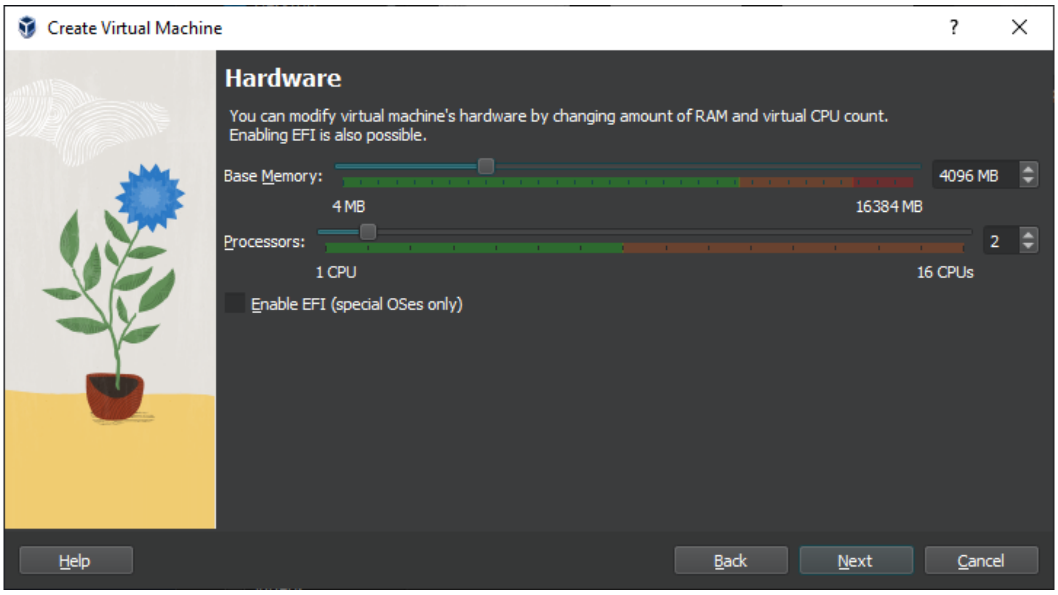
✓ Install in Background

✓ Guest Additions

Guest Additions ISO: C:\Program Files\Oracle\VirtualBox\VBoxGuestAdditions.iso

You want to tick the Guest Additions and Install in Background options and also change your Username and Password fields to your liking. Note that your username must be all lower-case and no more than 32 characters. If you forget to change the default password, it will be changeme. Leave the Guest Additions ISO, Hostname and Domain Name as they are. Continue by pressing Next.

Step 2.2.2: Hardware



You can modify virtual machine's hardware by changing amount of RAM and virtual CPU count. Enabling EFI is also possible.

Base Memory: 4096 MB (4 MB – 16384 MB)

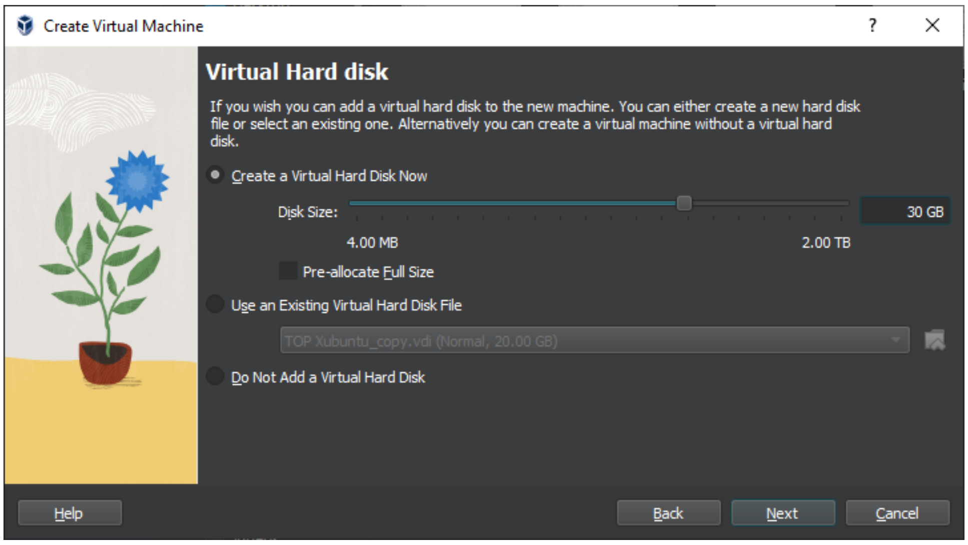
Processors: 2 (1 CPU – 16 CPUs)

Enable EFI (special OSes only)

In the Hardware section of the installation, you want to set your Base Memory to at least 2048 MB or more if possible - the upper limit is half of your total RAM but 4096 MB with the settings we recommend should give you a smooth experience.

For example, if you have 8 GB (8192 MB respectively) of RAM, you could allocate up to 4096 MB (1024 MB to 1 GB) to your VM's operating system.

Step 2.2.3: Virtual hard disk



If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select an existing one. Alternatively you can create a virtual machine without a virtual hard disk.

● Create a Virtual Hard Disk Now

Disk Size: 30 GB (4.00 MB – 2.00 TB)

Pre-allocate Full Size

Use an Existing Virtual Hard Disk File

TOP Xubuntu_copy.vdi (Normal, 20.00 GB)

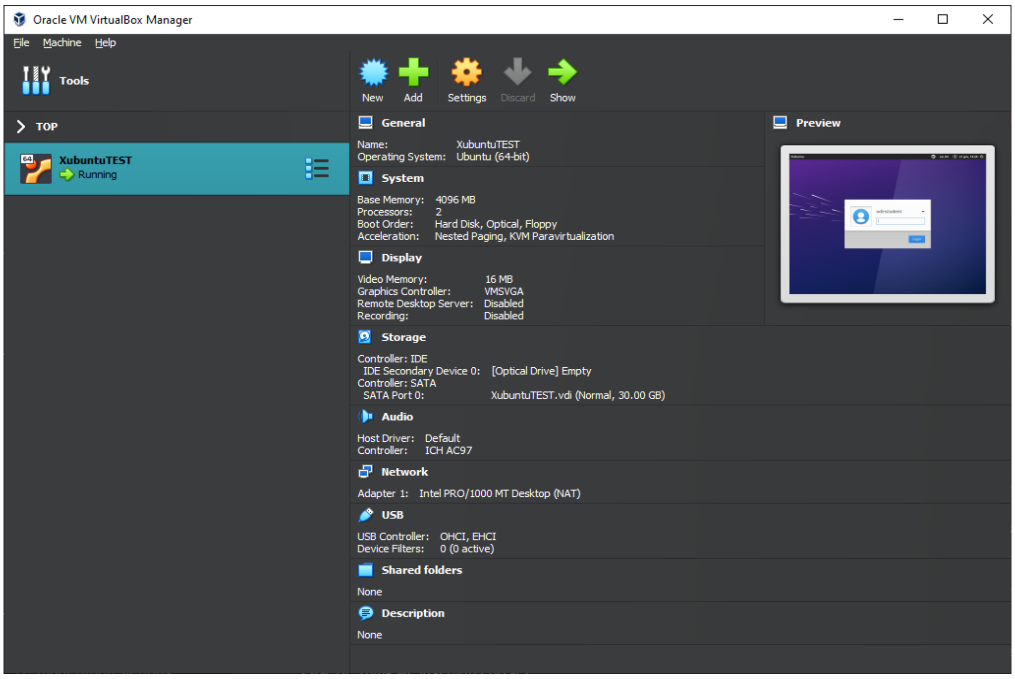
Do Not Add a Virtual Hard Disk

Now, you want to leave all the settings as they are besides the Disk Size, we recommend giving the VM at least 30GB of space. Reminder that this disk will be created in the folder that you've specified on the very first step of the VM creation process but nonetheless, the disk can be moved and resized in the future if needed.

Step 2.2.4: Begin the unattended installation

Click Next to be taken to a Summary page, on which you can simply click Finish to begin the process of unattended installation. The neat thing about it? It installs the OS and GuestAdditions on its own, without your input!

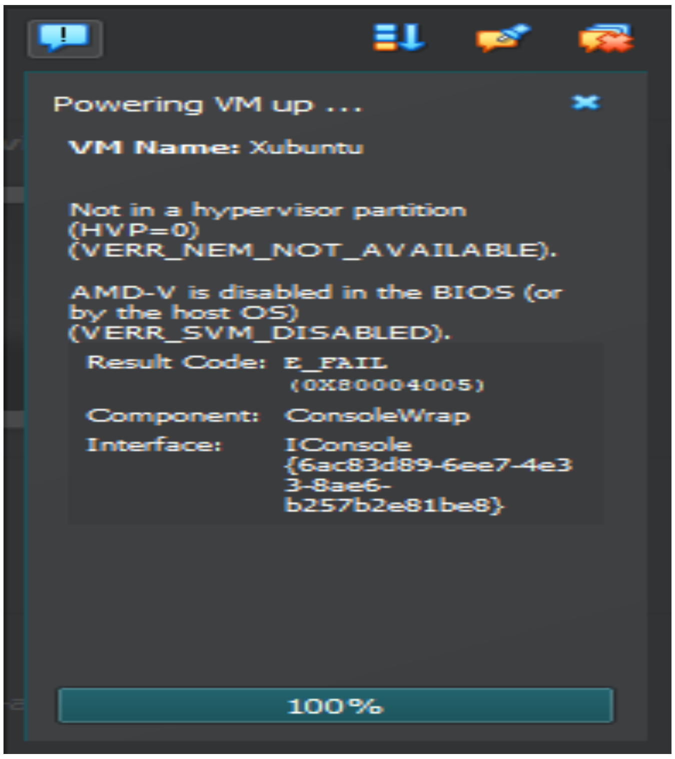
Just let it do its own thing, you will know it is finished when you will see a login screen like this one in the Preview section:



```
General
Name:              XubuntuTEST
Operating System:  Ubuntu (64-bit)
System
Base Memory:   4096 MB
Processors:    2
Boot Order:    Hard Disk, Optical, Floppy
Acceleration:  Nested Paging, KVM Paravirtualization
Display
Video Memory:           16 MB
Graphics Controller:    VMSVGA
Remote Desktop Server:  Disabled
Recording:              Disabled
Storage
Controller: IDE
 IDE Secondary Device 0:  [Optical Drive] Empty
Controller: SATA
 SATA Port 0:             XubuntuTEST.vdi (Normal, 30.00 GB)
Audio
Host Driver:  Default
Controller:   ICH AC97
Network
Adapter 1:  Intel PRO/1000 MT Desktop (NAT)
USB
USB Controller:  OHCI, EHCI
Device Filters:  0 (0 active)
Shared folders
None
Description
None
```

Just click the green arrow called Show and you'll be presented with a VM window and the login screen. Log in with the password you've set up during the installation process and we'll have one bit of configuration left to do.

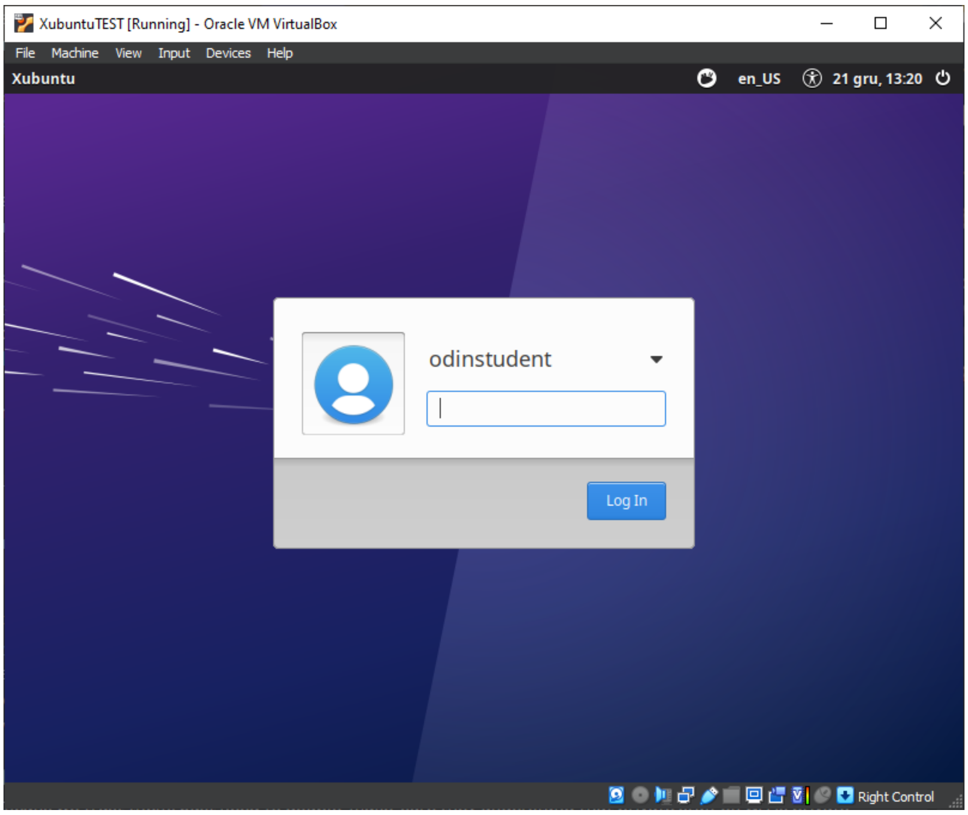
It is possible that you'll receive an error like this one after clicking Finish:



```
Powering VM up ...
VM Name: Xubuntu

Not in a hypervisor partition
(HVP=0)
(VERR_NEM_NOT_AVAILABLE).

AMD-V is disabled in the BIOS (or
by the host OS)
(VERR_SVM_DISABLED).
Result Code:  E_FAIL
              (0X80004005)
Component:    ConsoleWrap
Interface:    IConsole
              {6ac83d89-6ee7-4e3
              3-8ae6-
              b257b2e81be8}

100%
```

It means you must enable virtualization in your computer's BIOS/UEFI settings. Alternative set of instructions. If you have an AMD CPU, you're probably looking for something called SVM to enable, for Intel CPUs, Intel Virtualization Technology. The error should tell you what it is looking for.

After you deal with it, just Start the machine and let things happen, you'll know that the process has finished when you see a login screen:


XubuntuTEST [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Xubuntu
en_US
21 gru, 13:20
odinstudent
Log In
Right Control

VII. SSH Key in Ansible

An SSH key is a combination of two keys which are public and private. The private key is kept locally, and the public key is shared with remote hosts to which we want to connect. This combination is used for achieving asymmetric encryption, means is something is encrypted with one key of this combination, then another key of combination is used to decrypt that. In Ansible, we are using OpenSSH to make SSH connections to remote target nodes.

Generating a new SSH key and adding it to the ssh-agent:

After you've checked for existing SSH keys, you can generate a new SSH key to use for authentication, then add it to the ssh-agent.

Generating a new SSH key:

You can generate a new SSH key on your local machine. After you generate the key, you can add the public key to your account on GitHub.com to enable authentication for Git operations over SSH.

1. **Open Git Bash.**
2. **Paste the text below, substituting in your GitHub email address.**

ssh-keygen -t ed25519 -C "your_email@example.com"

Note: If you are using a legacy system that doesn't support the Ed25519 algorithm, use:

ssh-keygen -t rsa -b 4096 -C "your_email@example.com"

This creates a new SSH key, using the provided email as a label.

> Generating public/private ALGORITHM key pair.

When you're prompted to "Enter a file in which to save the key", you can press Enter to accept the default file location. Please note that if you created SSH keys previously, ssh-keygen may ask you to rewrite another key, in which case we recommend creating a custom-named SSH key. To do so, type the default file location and replace id_ssh_keyname with your custom key name.

> Enter a file in which to save the key (/c/Users/YOU/.ssh/id_ALGORITHM): [Press enter]

At the prompt, type a secure passphrase. For more information, see "Working with SSH key passphrases."

>Enter passphrase (empty for no passphrase): [Type a passphrase]
>Enter same passphrase again: [Type passphrase again]

Adding your SSH key to the ssh-agent:

Before adding a new SSH key to the ssh-agent to manage your keys, you should have checked for existing SSH keys and generated a new SSH key.

If you have GitHub Desktop installed, you can use it to clone repositories and not deal with SSH keys.

Ensure the ssh-agent is running. You can use the "Auto-launching the ssh-agent" instructions in "Working with SSH key passphrases", or start it manually:

```
# start the ssh-agent in the background
$ eval "$(ssh-agent -s)"
> Agent pid 59566
```

Add your SSH private key to the ssh-agent. If you created your key with a different name, or if you are adding an existing key that has a different name, replace id_ed25519 in the command with the name of your private key file.

ssh-add ~/.ssh/id_ed25519

Add the SSH public key to your account on GitHub.

Generating a new SSH key for a hardware security key:

If you are using macOS or Linux, you may need to update your SSH client or install a new SSH client prior to generating a new SSH key. For more information, see "Error: Unknown key type."

1. **Insert your hardware security key into your computer.**
2. **Open Git Bash.**
3. **Paste the text below, substituting in the email address for your account on GitHub.**

ssh-keygen -t ed25519-sk -C "YOUR_EMAIL"

Note: If the command fails and you receive the error invalid format or feature not supported, you may be using a hardware security key that does not support the Ed25519 algorithm. Enter the following command instead.

ssh-keygen -t ecdsa-sk -C your_email@example.com

When you are prompted, touch the button on your hardware security key.

When you are prompted to "Enter a file in which to save the key," press Enter to accept the default file location.

> Enter a file in which to save the key (/c/Users/YOU/.ssh/id_ed25519_sk): [Press enter]

When you are prompted to type a passphrase, press Enter.

> Enter passphrase (empty for no passphrase): [Type a passphrase]
> Enter same passphrase again: [Type passphrase again]

Add the SSH public key to your account on GitHub.

VIII. File Management with Ansible

File management is a critical aspect of system administration and configuration management. It involves tasks such as creating, modifying, copying, moving, and deleting files and directories on remote servers or machines. Ansible, a powerful automation tool, provides a streamlined and efficient way to manage files across multiple systems, ensuring consistency and reliability in your infrastructure.

Ansible employs a declarative approach, allowing you to define the desired state of your systems rather than scripting specific actions. This approach makes file management more intuitive and less error-prone. Here's how Ansible handles file management:

Playbooks: Ansible organizes tasks into playbooks. Each playbook is a collection of tasks that define what should be done on a set of target machines. For file management, tasks are used to specify file-related operations.

Modules: Ansible provides various modules tailored for file management tasks. Some commonly used file-related modules include:

copy: Copies files from the control machine to remote hosts.

template: Copies files while processing Jinja2 templates for dynamic content.

file: Manages file attributes (permissions, ownership, etc.) and existence.

lineinfile: Manages lines in text files, such as appending, modifying, or deleting lines.

assemble: Assembles files from fragments on remote hosts.

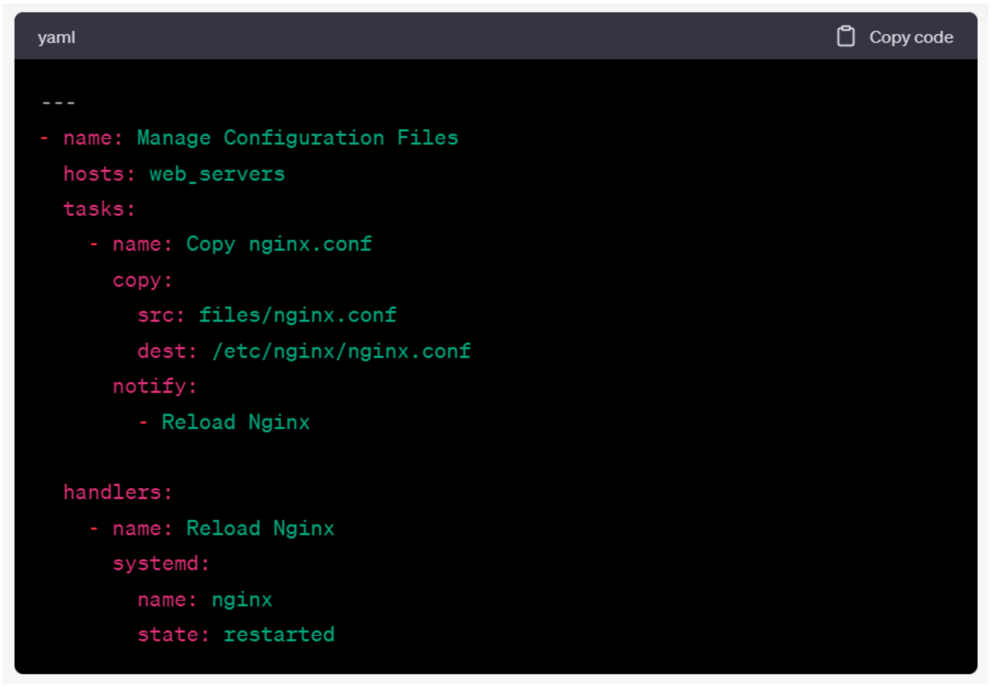
Inventory: Ansible uses an inventory file to define the hosts on which tasks will be executed. This file can specify groups of hosts and their connection information.

Variables: Ansible allows you to define variables that can be used in playbooks. This is especially useful for configuring file paths, permissions, and other settings that may vary across different hosts.

Handlers: Handlers are tasks that are triggered when notified by other tasks. They are often used in conjunction with file management tasks to ensure changes take effect only when necessary.

Idempotence: Ansible tasks are idempotent, meaning they can be run multiple times without causing unintended side effects. If a file task is defined to ensure certain attributes, Ansible will make sure those attributes are correct, even if the task is run multiple times.

Here's a simple example of using Ansible to manage files:


yaml Copy code

```yaml
---
- name: Manage Configuration Files
  hosts: web_servers
  tasks:
    - name: Copy nginx.conf
      copy:
        src: files/nginx.conf
        dest: /etc/nginx/nginx.conf
      notify:
        - Reload Nginx

  handlers:
    - name: Reload Nginx
      systemd:
        name: nginx
        state: restarted
```

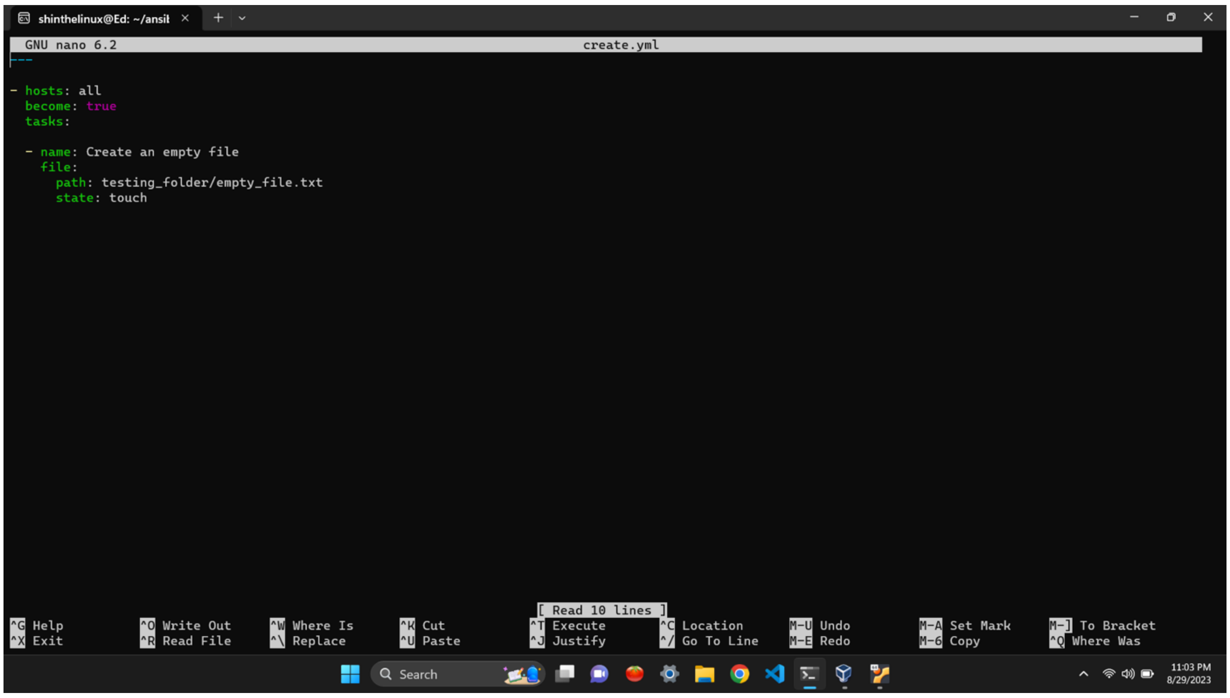
In this example, Ansible will copy the **nginx.conf** file from the control machine to the remote hosts in the "web_servers" group. After copying, it will trigger the "**Reload Nginx**" handler to restart the Nginx service.

Ansible simplifies and standardizes file management across your infrastructure, reducing manual intervention and ensuring that your systems are always in the desired state.

Testing of File Management

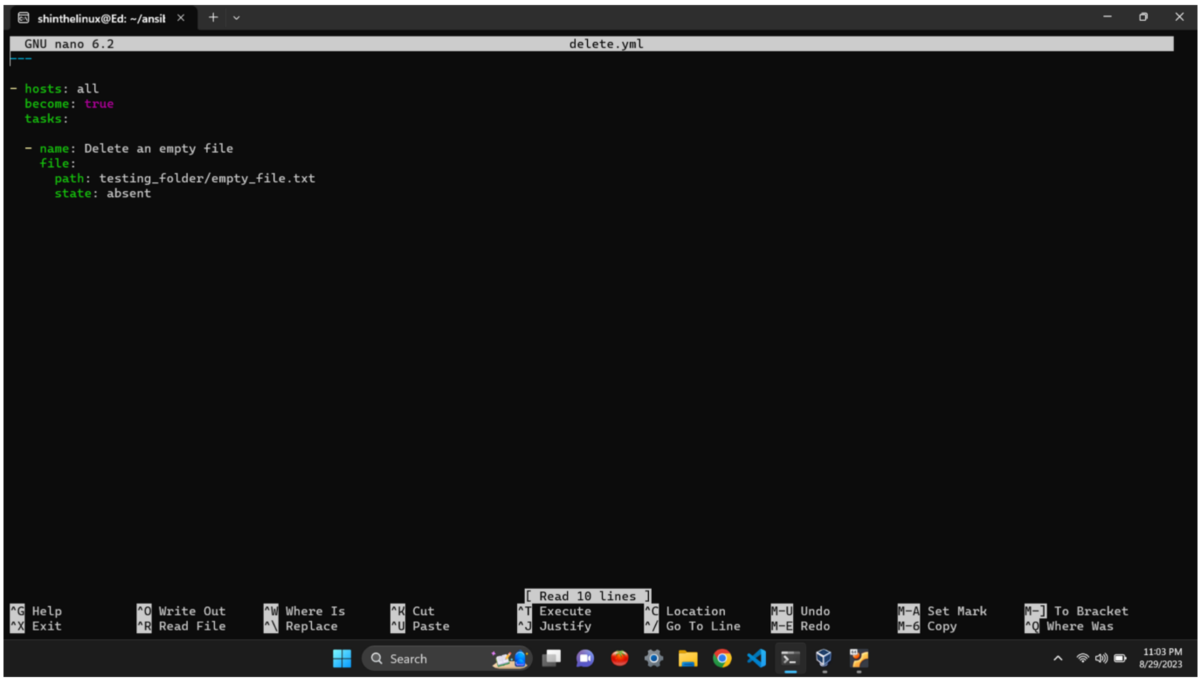
Creating an Empty File:

The picture is shown in the next page.



```
GNU nano 6.2                                    create.yml
---

- hosts: all
  become: true
  tasks:

  - name: Create an empty file
    file:
      path: testing_folder/empty_file.txt
      state: touch
```

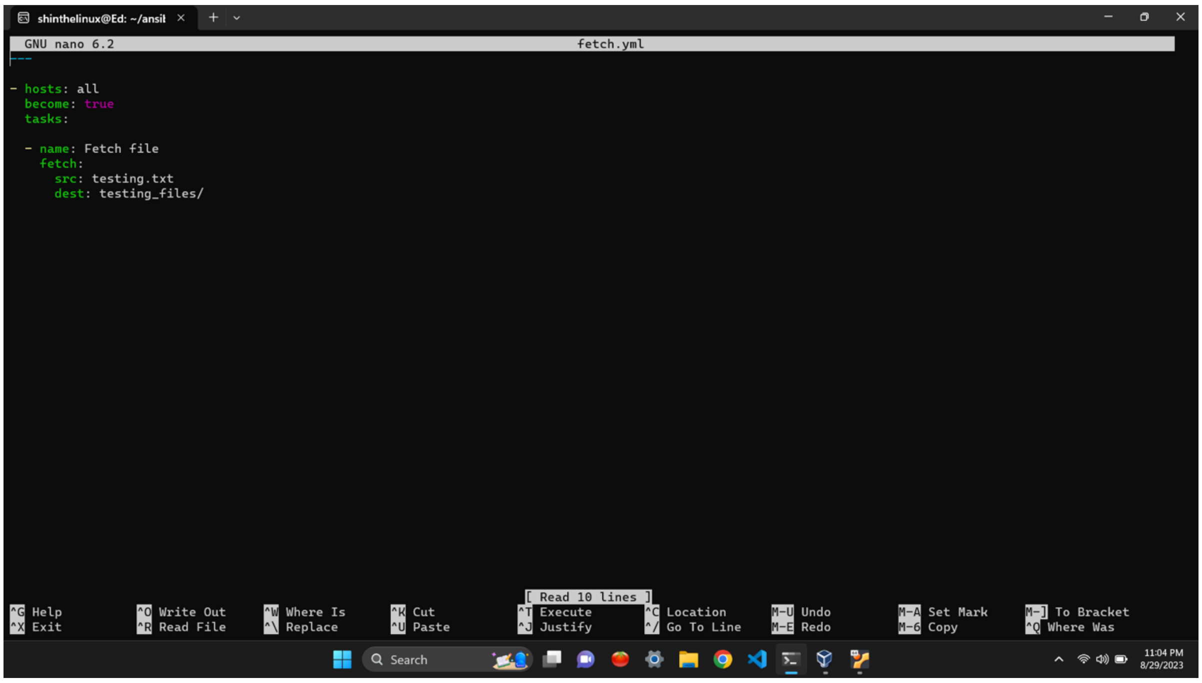
Deleting an Empty File:



```
GNU nano 6.2                                    delete.yml
---

- hosts: all
  become: true
  tasks:

  - name: Delete an empty file
    file:
      path: testing_folder/empty_file.txt
      state: absent
```

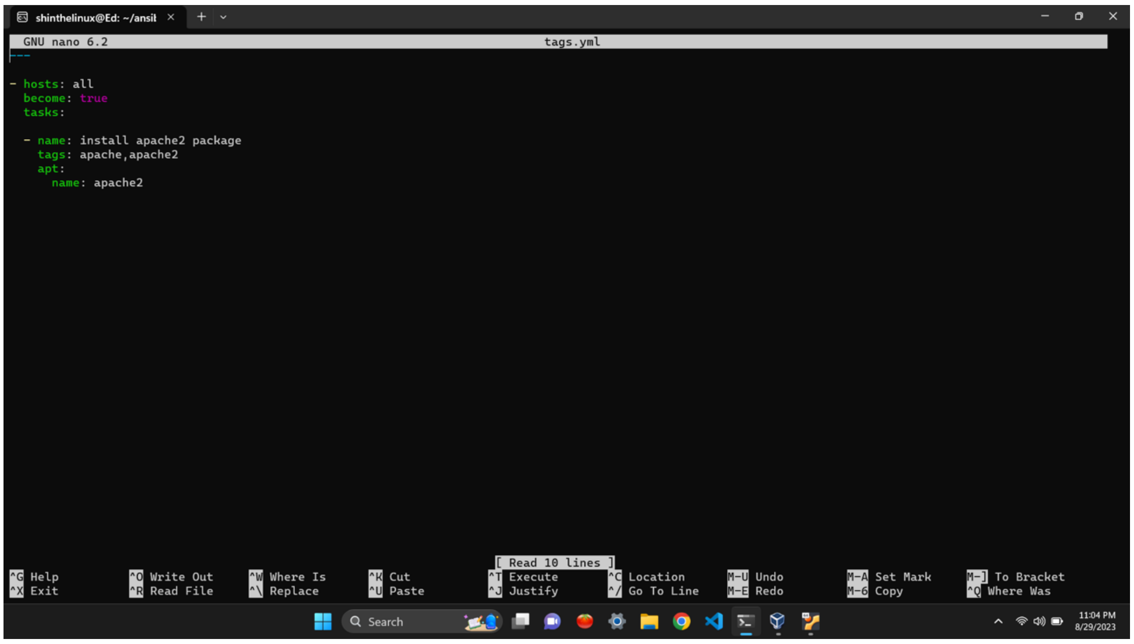
Fetching a File:

The picture is shown in the next page.



```
---

- hosts: all
  become: true
  tasks:

  - name: Fetch file
    fetch:
      src: testing.txt
      dest: testing_files/
```

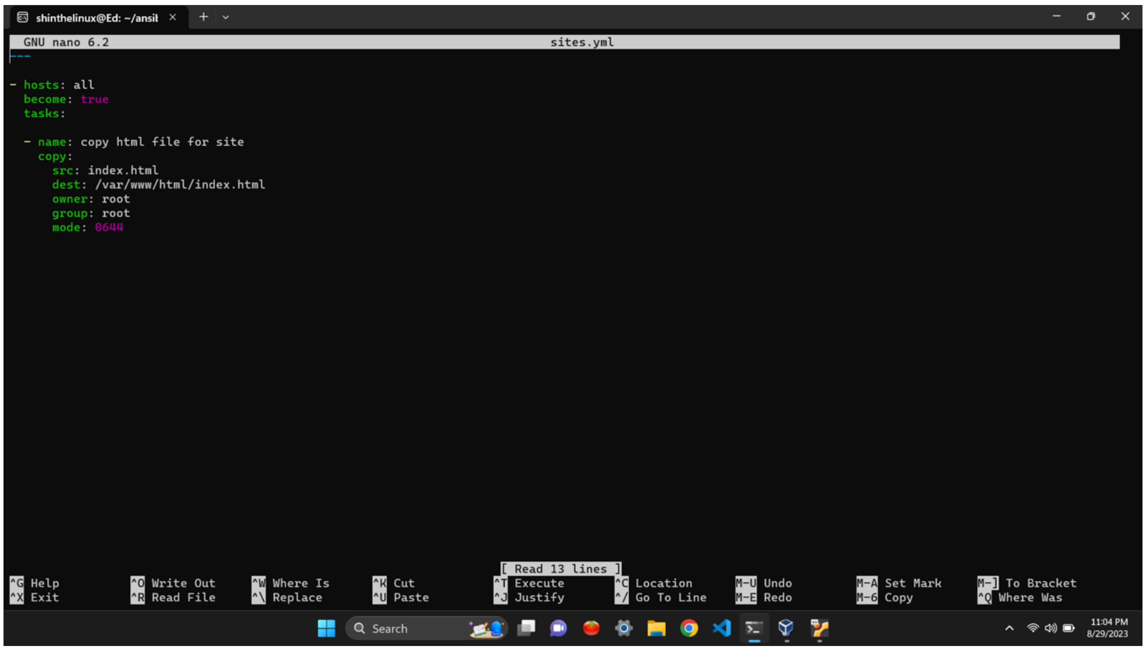
Installing apache2 Package:



```
---

- hosts: all
  become: true
  tasks:

  - name: install apache2 package
    tags: apache,apache2
    apt:
      name: apache2
```

Copying html File for Site:

The picture is shown in the next page.



```
  GNU nano 6.2                                    sites.yml
---

- hosts: all
  become: true
  tasks:

  - name: copy html file for site
    copy:
      src: index.html
      dest: /var/www/html/index.html
      owner: root
      group: root
      mode: 0644

                                              [ Read 13 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark   M-] To Bracket
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy       ^Q Where Was
```

IX. Service Management with Ansible

Service management is a crucial aspect of maintaining and controlling the state of services (applications, daemons, processes) running on remote servers or machines. Ansible, an automation tool, provides a convenient and efficient way to manage services across multiple systems, ensuring consistency and reliability in your infrastructure.

Using Ansible for service management involves defining the desired state of services on target machines and then orchestrating actions to achieve and maintain that state. Here's how Ansible handles service management:

Playbooks: Ansible uses playbooks to organize tasks. Each playbook is a collection of tasks that outline what actions should be taken on a set of target machines. For service management, tasks are used to control the state of services.

Modules: Ansible offers various modules for service management tasks. Some commonly used modules include:

systemd: Interacts with the systemd init system to manage services (start, stop, restart, enable, disable).

service: Works with traditional init scripts to manage services.

win_service: Manages services on Windows systems.

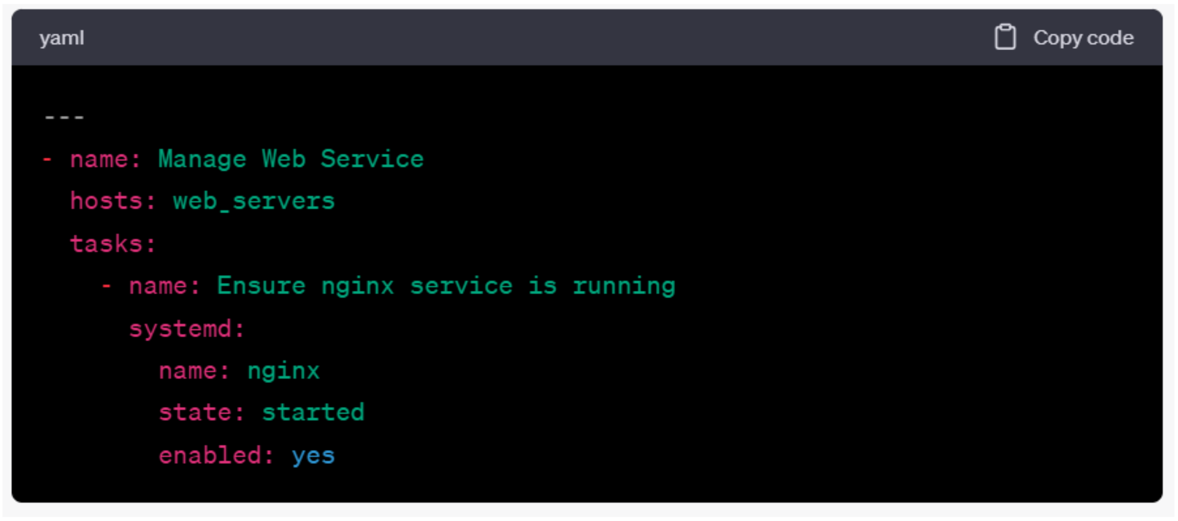
Inventory: Ansible uses an inventory file to specify the hosts on which tasks will be executed. This file can organize hosts into groups and provide connection details.

Variables: Ansible lets you define variables to customize playbook behavior. For service management, variables might include service names, paths, or specific configuration options.

Handlers: Handlers are tasks that are triggered when notified by other tasks. They're commonly used in service management to ensure that changes take effect when needed.

Idempotence: Ansible tasks are idempotent, meaning they can be executed multiple times without causing unintended side effects. If a service task is defined to ensure a certain state, Ansible will make sure the service is in that state, even if the task is run repeatedly.

Here's a simple example of using Ansible to manage services:

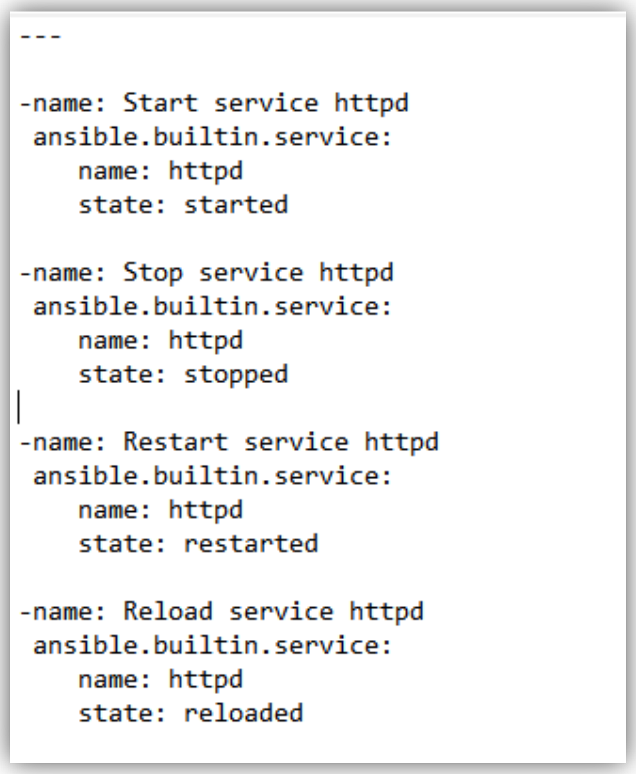


```yaml
---
- name: Manage Web Service
  hosts: web_servers
  tasks:
    - name: Ensure nginx service is running
      systemd:
        name: nginx
        state: started
        enabled: yes
```

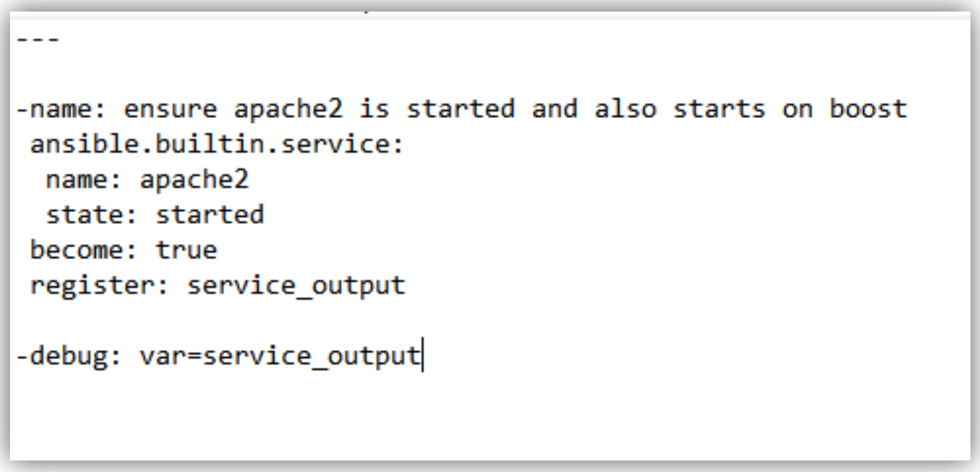
In this example, Ansible will ensure that the Nginx service is running and enabled (starts on boot) on the remote hosts in the "web_servers" group.

Using Ansible for service management helps you automate routine tasks and ensure services are consistently maintained across your infrastructure. It simplifies configuration changes, reduces manual intervention, and enhances the overall stability of your systems.

Testing of Service Management

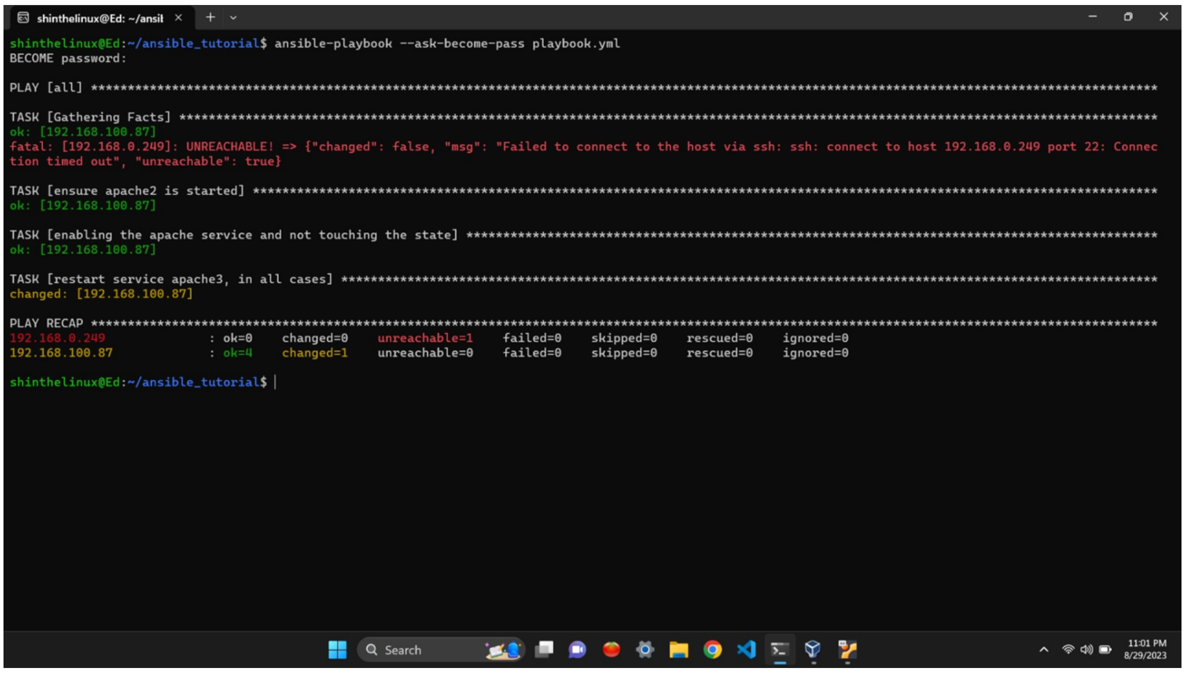


```
---

-name: Start service httpd
 ansible.builtin.service:
    name: httpd
    state: started

-name: Stop service httpd
 ansible.builtin.service:
    name: httpd
    state: stopped

-name: Restart service httpd
 ansible.builtin.service:
    name: httpd
    state: restarted

-name: Reload service httpd
 ansible.builtin.service:
    name: httpd
    state: reloaded
```

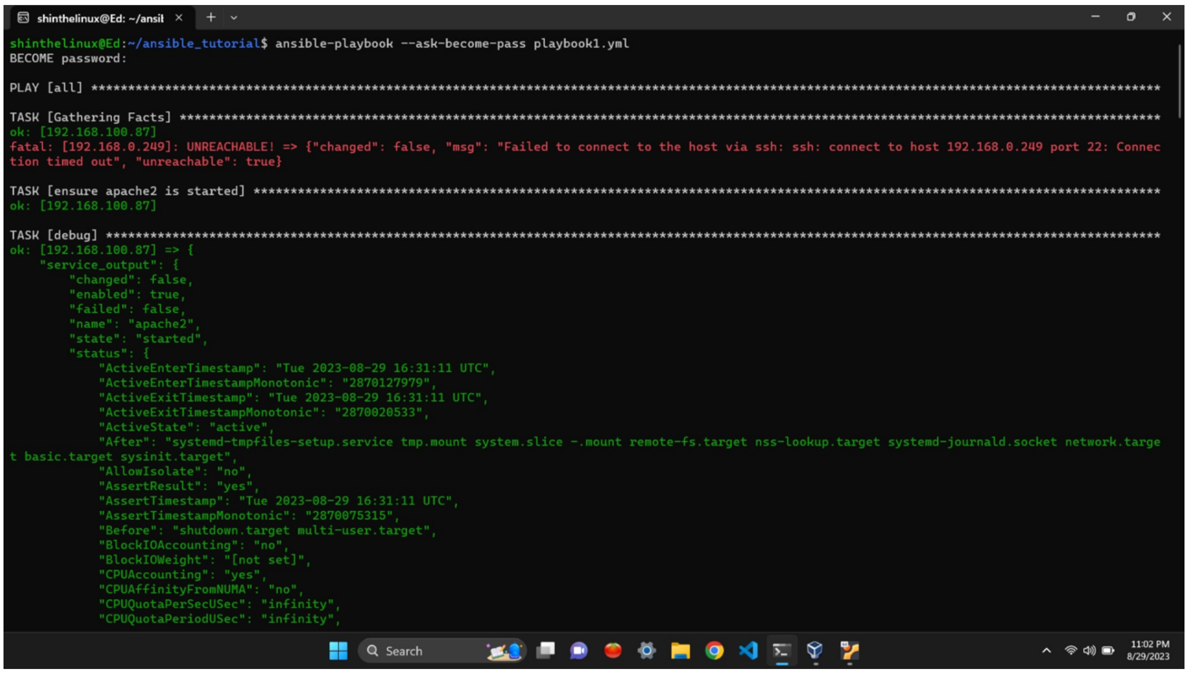


```
---

-name: ensure apache2 is started and also starts on boost
 ansible.builtin.service:
  name: apache2
  state: started
 become: true
 register: service_output

-debug: var=service_output
```

Results:



```
shinthelinux@Ed:~/ansible_tutorial$ ansible-playbook --ask-become-pass playbook.yml
BECOME password:

PLAY [all] *****************************************************************************************************************

TASK [Gathering Facts] *****************************************************************************************************
ok: [192.168.100.87]
fatal: [192.168.0.249]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect to the host via ssh: ssh: connect to host 192.168.0.249 port 22: Connec
tion timed out", "unreachable": true}

TASK [ensure apache2 is started] *******************************************************************************************
ok: [192.168.100.87]

TASK [enabling the apache service and not touching the state] **************************************************************
ok: [192.168.100.87]

TASK [restart service apache3, in all cases] *******************************************************************************
changed: [192.168.100.87]

PLAY RECAP *****************************************************************************************************************
192.168.0.249              : ok=0    changed=0    unreachable=1    failed=0    skipped=0    rescued=0    ignored=0
192.168.100.87             : ok=4    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

shinthelinux@Ed:~/ansible_tutorial$ |
```



```
shinthelinux@Ed:~/ansible_tutorial$ ansible-playbook --ask-become-pass playbook1.yml
BECOME password:

PLAY [all] *******************************************************************************************************************************

TASK [Gathering Facts] *******************************************************************************************************************
ok: [192.168.100.87]
fatal: [192.168.0.249]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect to the host via ssh: ssh: connect to host 192.168.0.249 port 22: Connec
tion timed out", "unreachable": true}

TASK [ensure apache2 is started] *********************************************************************************************************
ok: [192.168.100.87]

TASK [debug] *****************************************************************************************************************************
ok: [192.168.100.87] => {
    "service_output": {
        "changed": false,
        "enabled": true,
        "failed": false,
        "name": "apache2",
        "state": "started",
        "status": {
            "ActiveEnterTimestamp": "Tue 2023-08-29 16:31:11 UTC",
            "ActiveEnterTimestampMonotonic": "2870127979",
            "ActiveExitTimestamp": "Tue 2023-08-29 16:31:11 UTC",
            "ActiveExitTimestampMonotonic": "2870020533",
            "ActiveState": "active",
            "After": "systemd-tmpfiles-setup.service tmp.mount system.slice -.mount remote-fs.target nss-lookup.target systemd-journald.socket network.targe
t basic.target sysinit.target",
            "AllowIsolate": "no",
            "AssertResult": "yes",
            "AssertTimestamp": "Tue 2023-08-29 16:31:11 UTC",
            "AssertTimestampMonotonic": "2870075315",
            "Before": "shutdown.target multi-user.target",
            "BlockIOAccounting": "no",
            "BlockIOWeight": "[not set]",
            "CPUAccounting": "yes",
            "CPUAffinityFromNUMA": "no",
            "CPUQuotaPerSecUSec": "infinity",
            "CPUQuotaPeriodUSec": "infinity",
```

X. Conclusion

As a last thought, we conclude that Ansible exists to offer a straightforward and effective package for configuration management and automation. Being new to the software application market, Ansible faces tough competition from renowned sources. The availability of a limited amount of Ansible-related documentation creates setbacks for learning Ansible.

On a positive note, a growing interest can be seen in Ansible due to its adoption by NASA. The potentiality of Ansible can be described by its functionalities such as provisioning, application, orchestration, deployment, and security and compliance.

Ansible's benefits could be strengthened while the disadvantages are being addressed; by achieving this, a promising future for Ansible can be foreseen.

In this article, we have tried our best to cover as maximum as possible important aspects of Ansible to make you aware of this technology in the best possible way.

XI. References

- https://www.javatpoint.com/ansible
- www.google.com

- https://www.tutorialspoint.com/ansible/ansible_introduction.htm
- https://docs.ansible.com/ansible/2.9/installation_guide/intro_installation.html
- www.stackoverflow.com
- https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent
- https://www.whizlabs.com/blog/