Introduction to Ansible on Linux

The Fundamentals

Thomas Cameron, RHCE
Senior Principal Solution Architect
thomas@redhat.com

What we'll discuss today

- Introductions
- About Ansible
- Ansible Core (command line) vs. AAP
- Requirements for Ansible
- Inventory files
- Ansible config
- Playbooks
- Ansible Ping
- Manual Configuration Of Services
- Automating Services
- Tying It All Together
- Questions
- Thanks!

Source:
Insert source data here
Insert source data here

Introductions

Who Am I? Who Are You?

Howdy, I'm Thomas!

A little about me:

- Been in the industry since 1993
  - Certs: Novell CNE, MCSE/MCT, AWS Solution Architect Professional, AWS certified instructor, RHCA level V, Red Hat Certified Instructor

Source:
https://www.linkedin.com/in/thomascameron/
https://rhtapps.redhat.com/verify?certId=100-000-702

Howdy, I'm Thomas!

A little about me:

- Been in the industry since 1993
  - Certs: Novell CNE, MCSE/MCT, AWS Solution Architect Professional, AWS certified instructor, RHCA level V, Red Hat Certified Instructor
- Started at Red Hat in 2005
  - Left Red Hat for AWS for a few years, saw the error of my ways, and I am back at Red Hat

Source:
https://www.linkedin.com/in/thomascameron/
https://rhtapps.redhat.com/verify?certId=100-000-702

How About You?

Tell Me If You've:

How About You?

Tell Me If You've:

- Used Ansible extensively

How About You?

Tell Me If You've:

- Used Ansible extensively
  - Why are you here, then???

Red Hat TXLF

How About You?

Tell Me If You've:

- Used Ansible extensively
  - Why are you here, then???
- Used Ansible a little bit

How About You?

Tell Me If You've:

- Used Ansible extensively
  - Why are you here, then???
- Used Ansible a little bit

Red Hat TXLF

How About You?

Tell Me If You've:

- Used Ansible extensively
  - Why are you here, then???
- Used Ansible a little bit
- Never used Ansible at all

How About You?

Tell Me If You've:

- Used Ansible extensively
  - Why are you here, then???
- Used Ansible a little bit
- Never used Ansible at all
  - I'm glad you're here!

Red Hat TXLF

About Ansible

What is it, what does it do, and why you should use it

About Ansible

What is it?

- Ansible is a suite of software tools that enables infrastructure as code. It is open-source and the suite includes software provisioning, configuration management, and application deployment functionality.

Source:
https://en.wikipedia.org/wiki/Ansible_(software)

About Ansible

What does it do?

- Originally written by Michael DeHaan in 2012, and acquired by Red Hat in 2015, Ansible is designed to configure both Unix-like systems and Microsoft Windows. Ansible is agentless, relying on temporary remote connections via SSH or Windows Remote Management which allows PowerShell execution. The Ansible control node runs on most Unix-like systems that are able to run Python, including Windows with Windows Subsystem for Linux installed. System configuration is defined in part by using its own declarative language.

Source:
https://en.wikipedia.org/wiki/Ansible_(software)

About Ansible

Why you should use it?

- It’s agentless. All you need is ssh or WinRM

Source:
https://en.wikipedia.org/wiki/Ansible_(software)

About Ansible

Why you should use it?

- It's agentless. All you need is ssh or WinRM
- It uses a pretty human-readable syntax - YAML

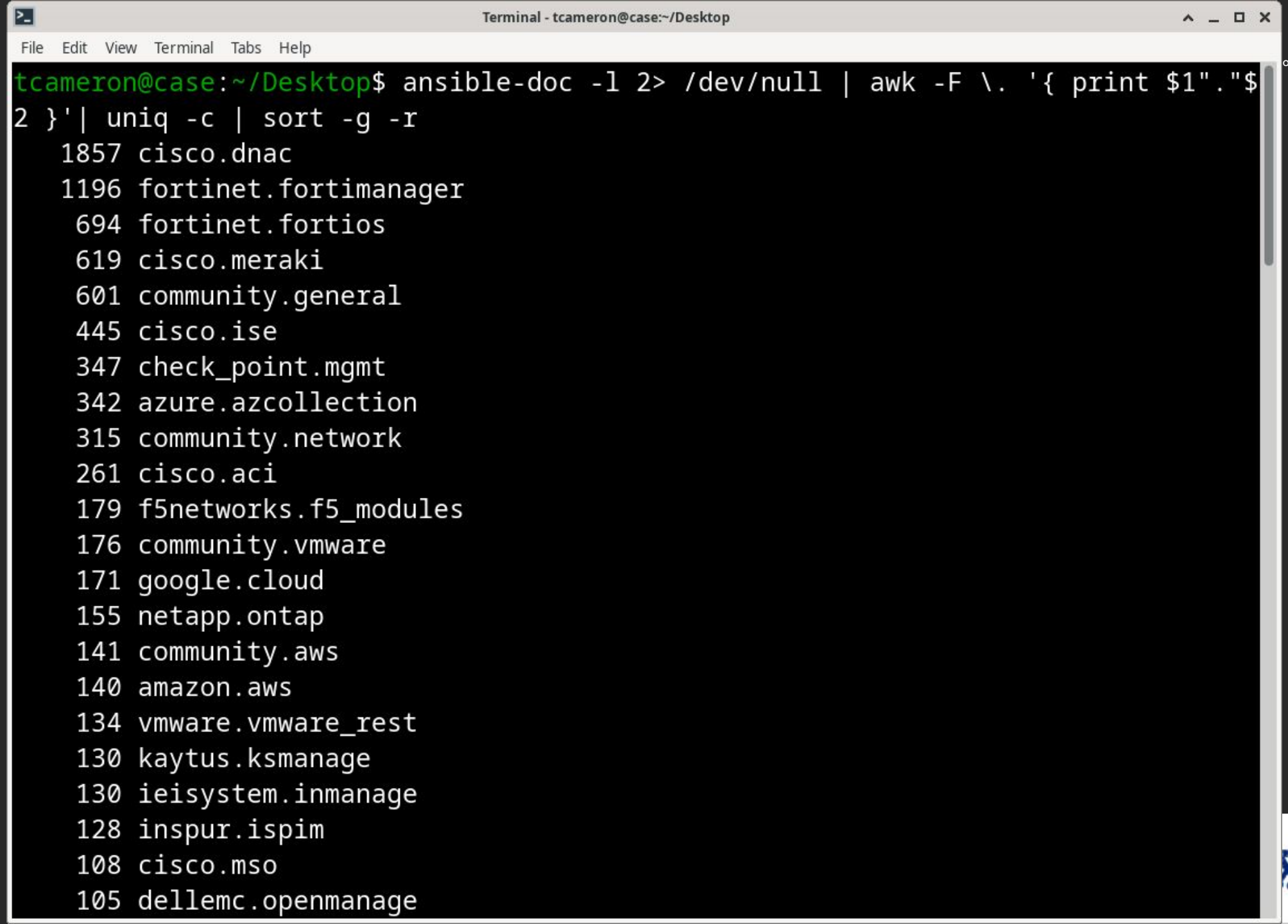
Source:
https://en.wikipedia.org/wiki/Ansible_(software)

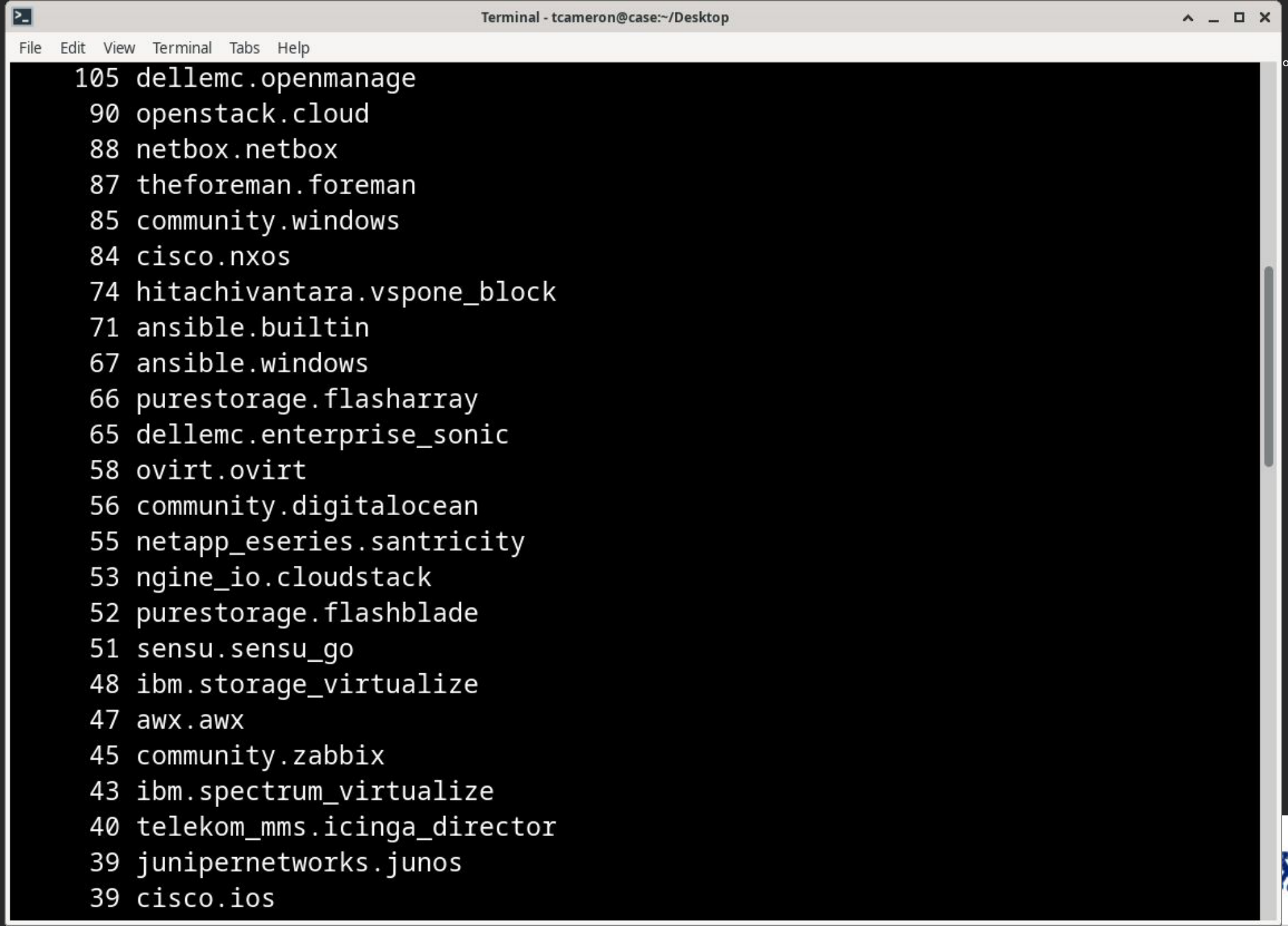
About Ansible

Why you should use it?

- It's agentless. All you need is ssh or WinRM
- It uses a pretty human-readable syntax - YAML
- It's available to manage Linux, Windows, network equipment from major vendors, cloud providers like AWS, Azure, Google, Oracle, IBM, virtualization, storage, firewalls, application gateways, and so on.

Source:
https://en.wikipedia.org/wiki/Ansible_(software)





Why Ansible

Why you should use it?

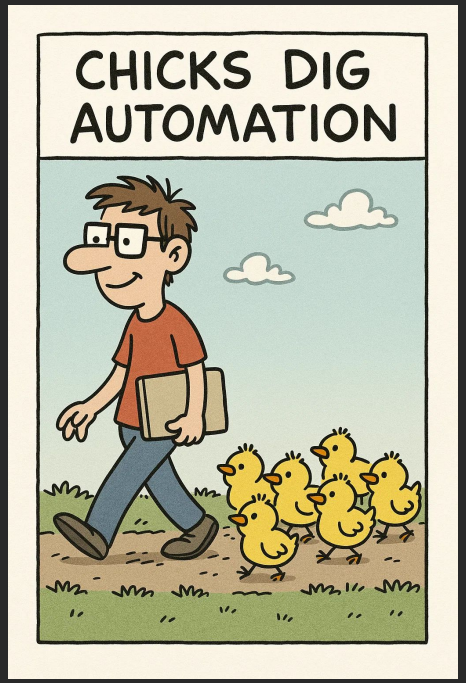
- It’s COOL

Source:

Why Ansible

Why you should use it?

- It's COOL



Source:
ChatGPT Sora

Ansible Core (command line) vs. AAP

Community vs. Enterprise

Ansible Core (command line) vs. AAP

Ansible core (command line) works well for small to medium environments.

- Sharing content is doable, but core is not really designed for it

Source:

Ansible Core (command line) vs. AAP

Ansible Automation Platform is designed for enterprise use

- Sharing content is easy
  - Shared projects
  - Shared execution environments

Source:

Ansible Core (command line) vs. AAP

Ansible Automation Platform is designed for enterprise use

- Sharing content is easy
  - Shared projects
  - Shared execution environments
- Includes Role Based Access Control
  - Organizations, teams, and users
  - Including authentication off of enterprise directory services like LDAP and Active Directory

Source:

Ansible Automation Platform - Google Chrome

Ansible Automation Platfor

aap.aus.redhat.com/overview

Red Hat | Personal | Rover Apps | Red Hat Support | I.T. New Hire Hub | I.T. Toolbox

Red Hat
Ansible Automation Platform

admin

- Overview
- Automation Execution
  Automation Controller
- Automation Decisions
  Event-Driven Ansible
- Automation Analytics
- Automation Content
  Automation Hub
- Ansible Lightspeed
- Access Management
- Settings
- QuickStarts

Welcome to the Ansible Automation Platform

Empower, automate, connect: Unleash possibilities with the Ansible Automation Platform.

Manage view

Resource Counts

55 Hosts — 55 Ready

2 Projects — 2 Ready

2 Inventories — 2 Synced

Job Activity

Past month | All job types | View all Jobs

Jobs

Recently finished jobs

View all Jobs

Projects

Recently updated projects

View all Projects

Requirements for Ansible

On your system and on target systems

Requirements for Ansible

There are a few requirements for running Ansible on your system

Source:
Insert source data here
Insert source data here

Requirements for Ansible

There are a few requirements for running Ansible on your system

- Install the ansible package using your system's package manager
  - I recommend you install the ansible package, as it drags in ansible-core and a ton of Ansible collections.

Source:
Insert source data here
Insert source data here

Requirements for Ansible

There are a few requirements for running Ansible on your system

- Install the ansible package using your system's package manager
  - I recommend you install the ansible package, as it drags in ansible-core and a ton of Ansible collections.
- Create a management account like "ansible," "sysadmin," or maybe a departmental account like "netops" or "sysops" or "storeageops" or similar. Grant them access via sudoers.

Source:
Insert source data here
Insert source data here

Requirements for Ansible

There are a few requirements for running Ansible on your system

- Install the ansible package using your system's package manager
  - I recommend you install the ansible package, as it drags in ansible-core and a ton of Ansible collections.
- Create a management account like "ansible," "sysadmin," or maybe a departmental account like "netops" or "sysops" or "storeageops" or similar. Grant them access via sudoers.
- Create ssh keys using ssh-keygen

Source:
Insert source data here
Insert source data here

Requirements for Ansible

Examples

- For the rest of today, txlf1.aus.redhat.com is the machine I will be running Ansible **from**.
- txlf2.aus.redhat.com, txlf3.aus.redhat.com, and txlf4.aus.redhat.com are the machines I will be managing using Ansible

Source:
Insert source data here
Insert source data here

Requirements for Ansible

On txlf1.aus.redhat.com:

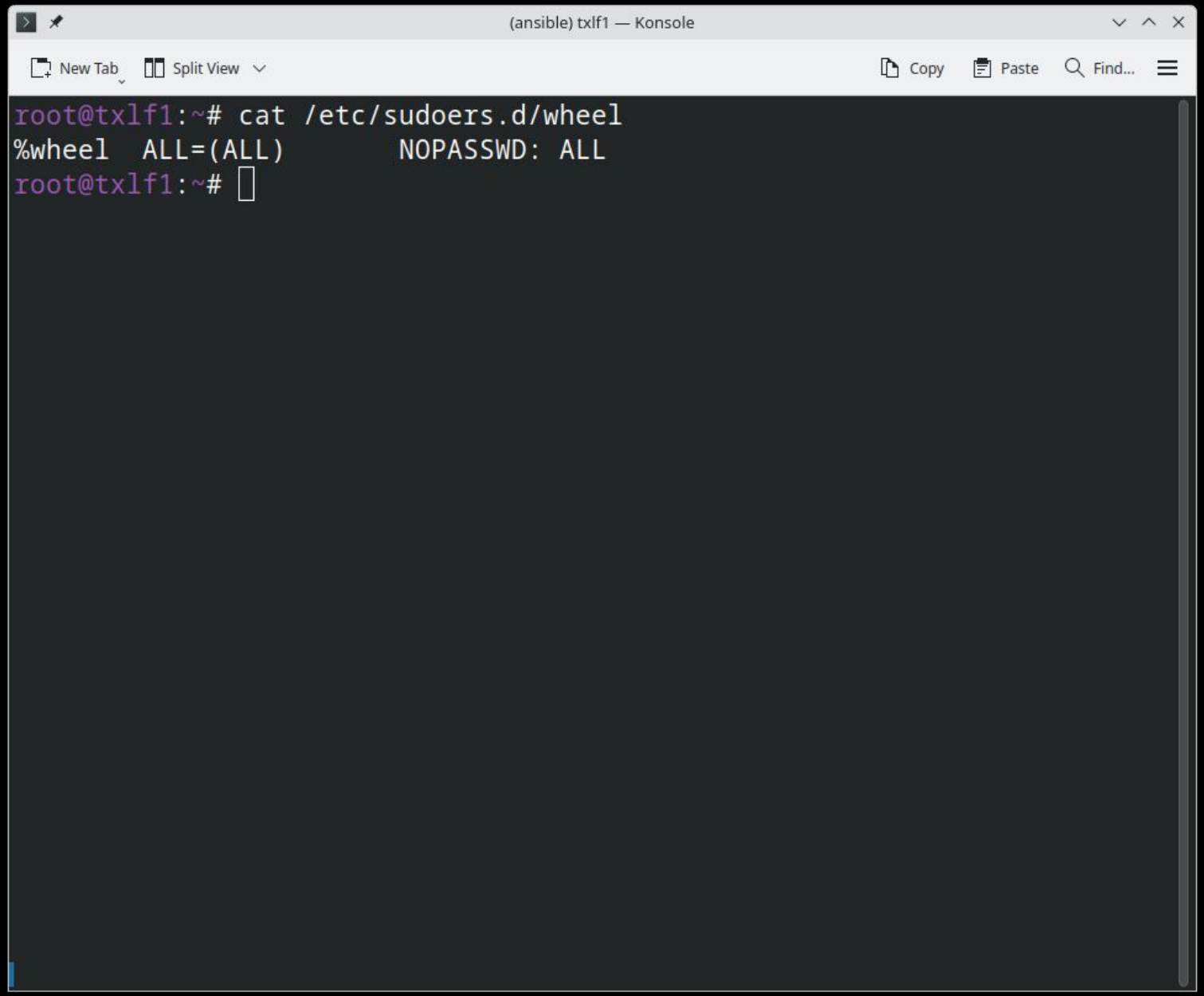
- Set up sudo access.
- Enable the wheel group to have all access

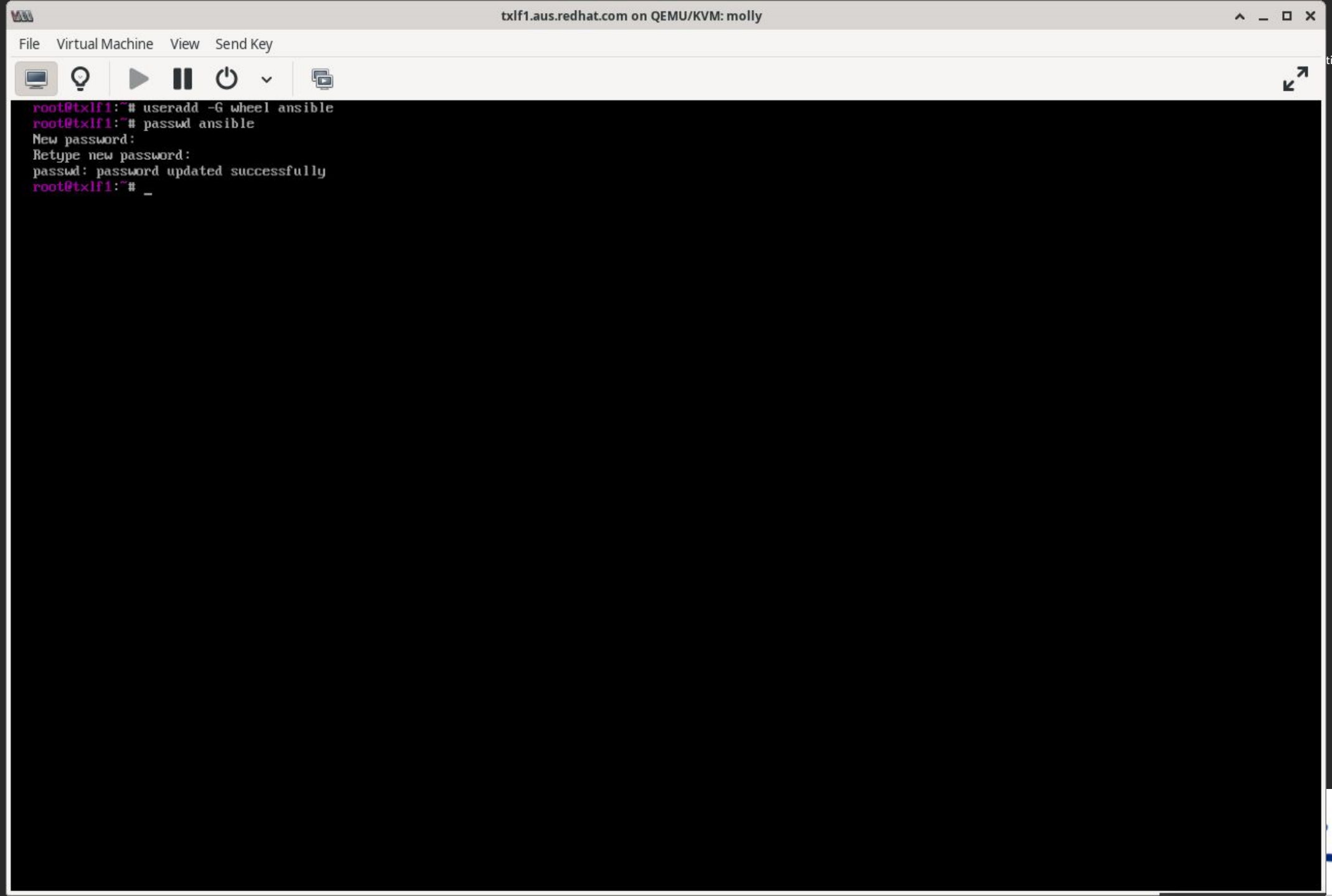
```
ansible ALL=(ALL) NOPASSWD: ALL
```

- NOTE: THE ABOVE IS DANGEROUS. You should probably limit it with something like:

```
ansible ALL=(ALL) NOPASSWD:/usr/bin/dnf,/usr/bin/systemctl,/usr/bin/firewall-cmd
```


Source:
Insert source data here
Insert source data here


(ansible) txlf1 — Konsole
New Tab
Split View
Copy
Paste
Find...
root@txlf1:~# cat /etc/sudoers.d/wheel
%wheel ALL=(ALL)   NOPASSWD: ALL
root@txlf1:~#

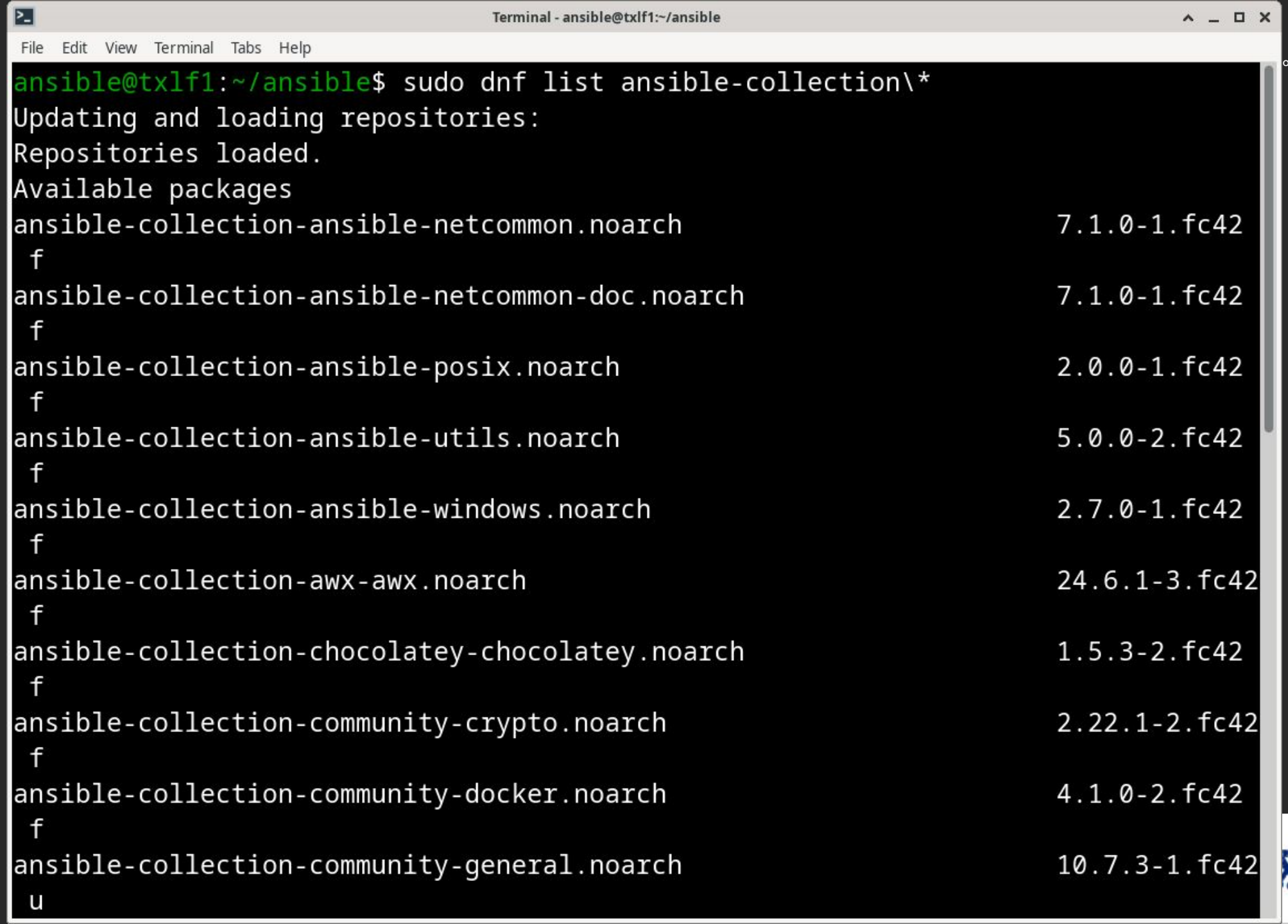


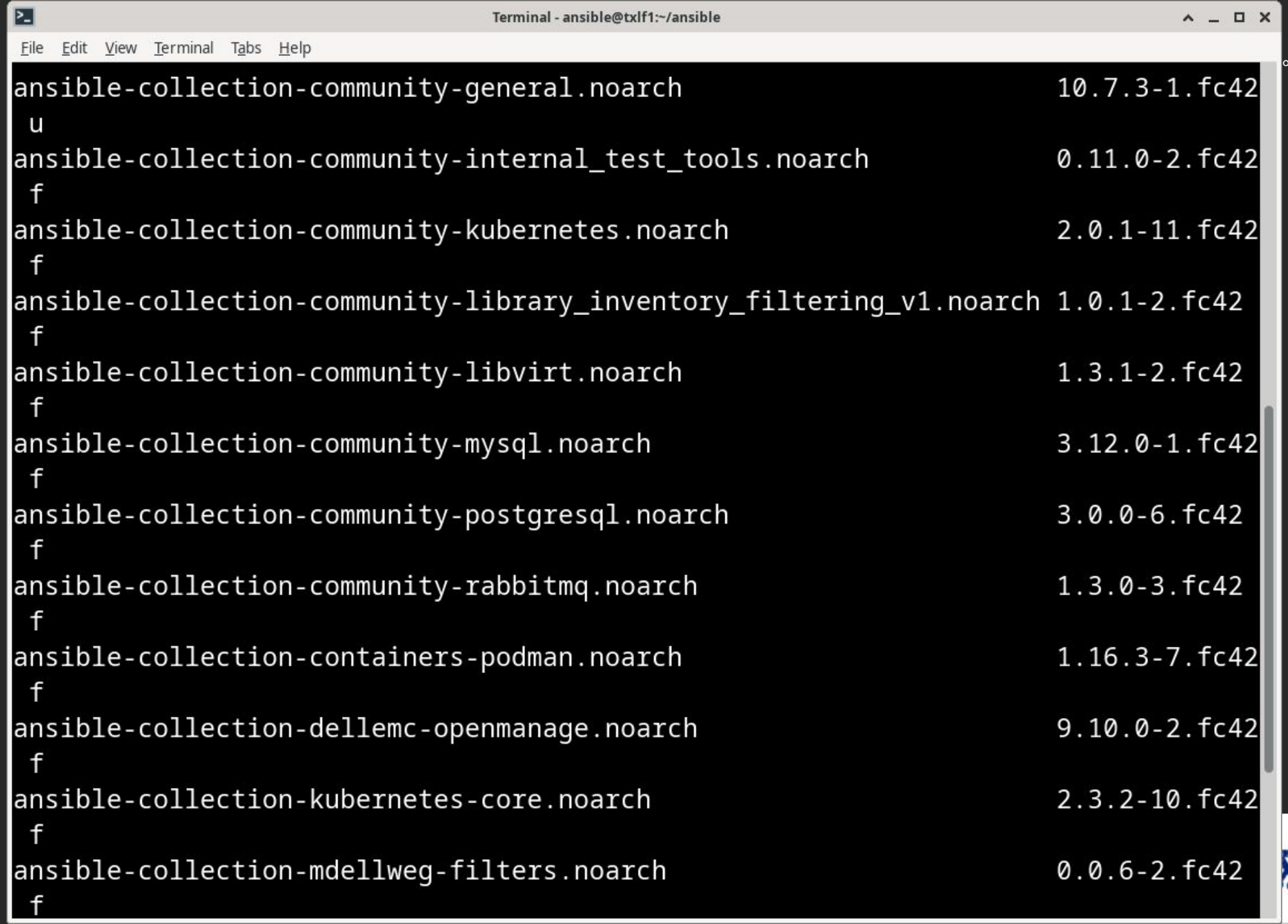
Requirements for Ansible

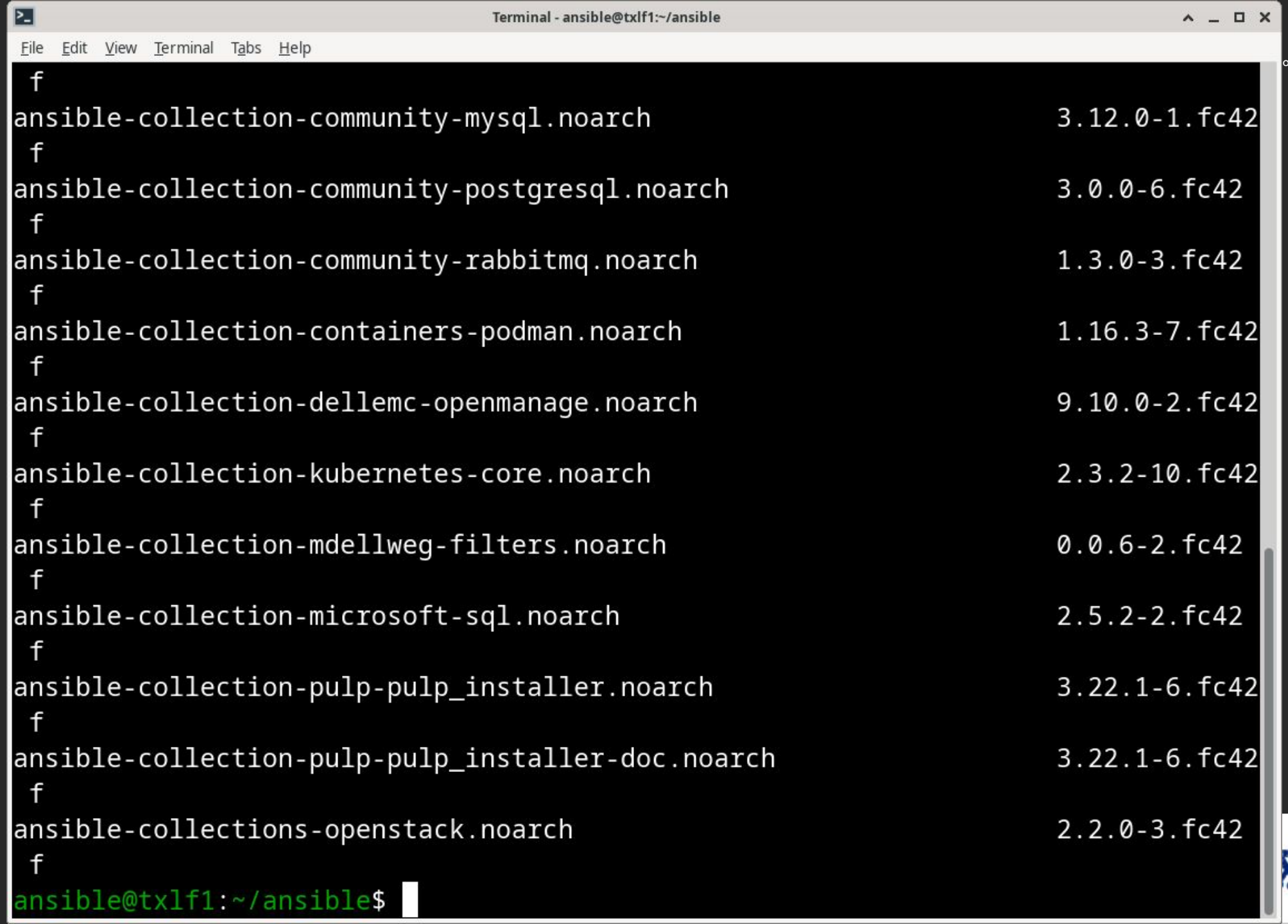
Install at least ansible-core (ansible metapackage preferred) and the **collections** you need

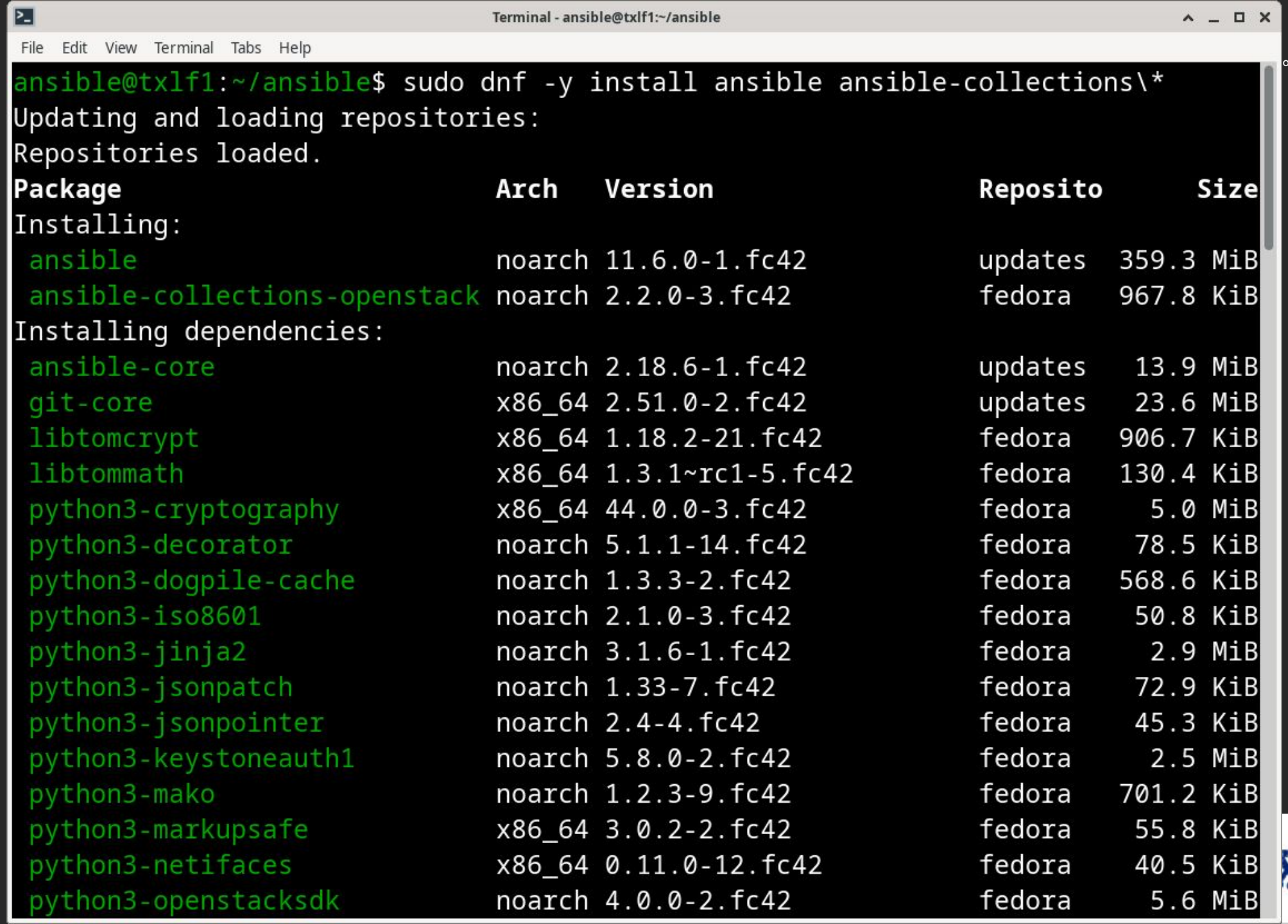
- Learning collections may take some time to learn, and that's cool!
- Collections are just bundles of python modules grouped together based on the technology being managed.
- Depending on your distro, they may be available to install via package management
    - Note that Red Hat certified collections are available with an AAP subscription

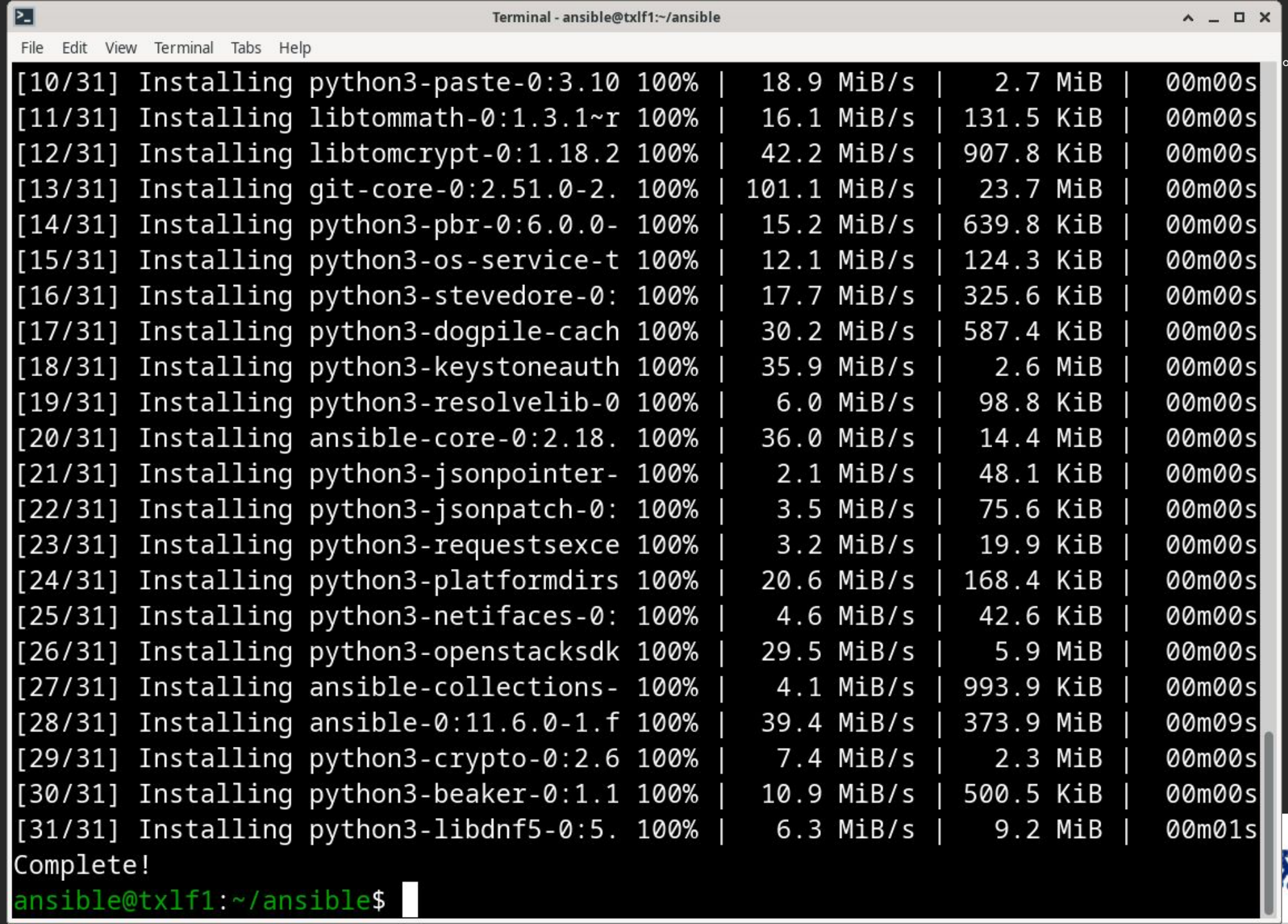
Source:
Insert source data here
Insert source data here











Check to see what collections you have installed

A quick way to make sure you have all the collections installed is to see how many collections you have *documentation* installed for.

- I use ansible-doc -l and pass it to wc -l
- I redirect errors to /dev/null because you will get a few warnings

Source:
Insert source data here
Insert source data here

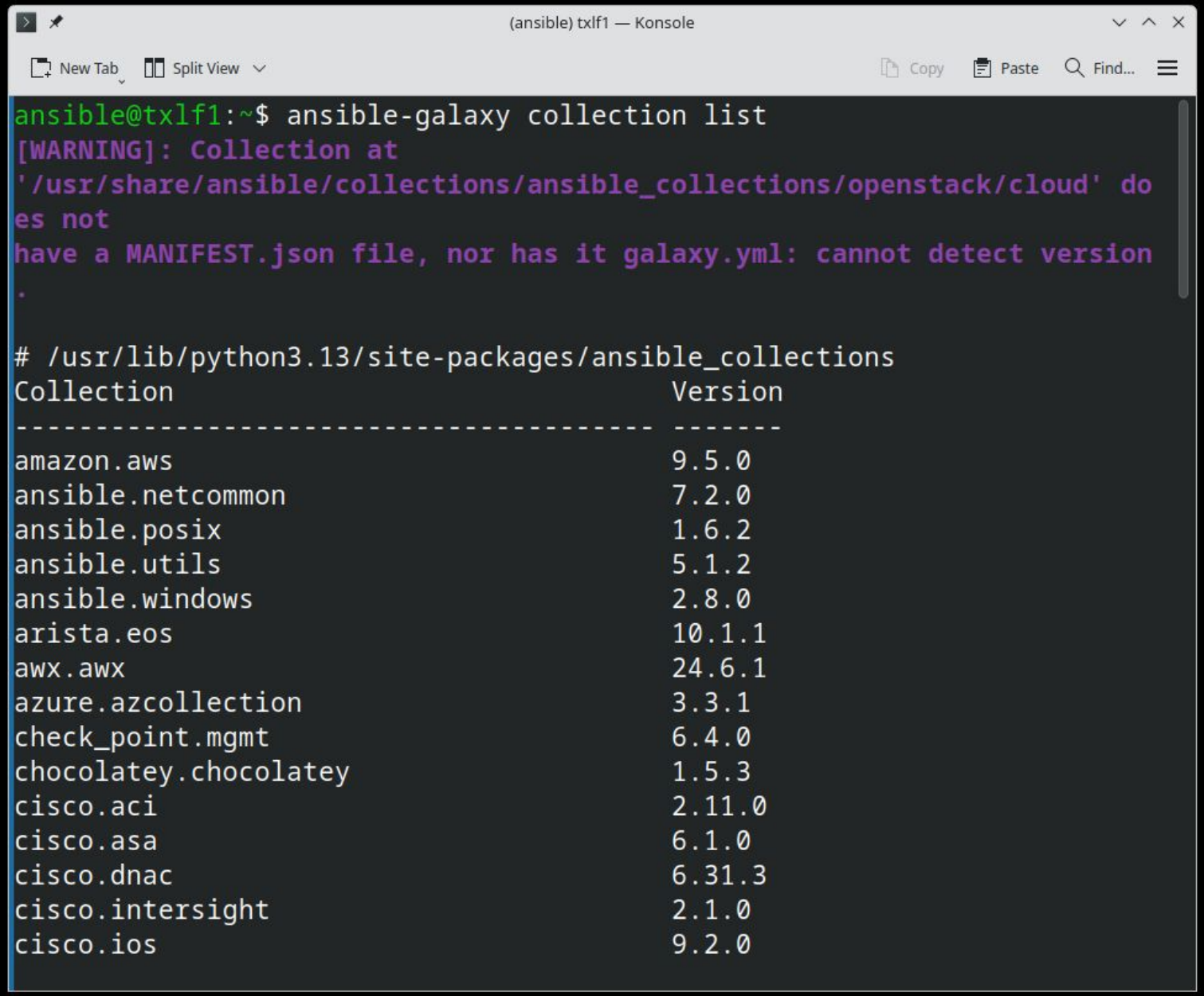
Red Hat TXLF

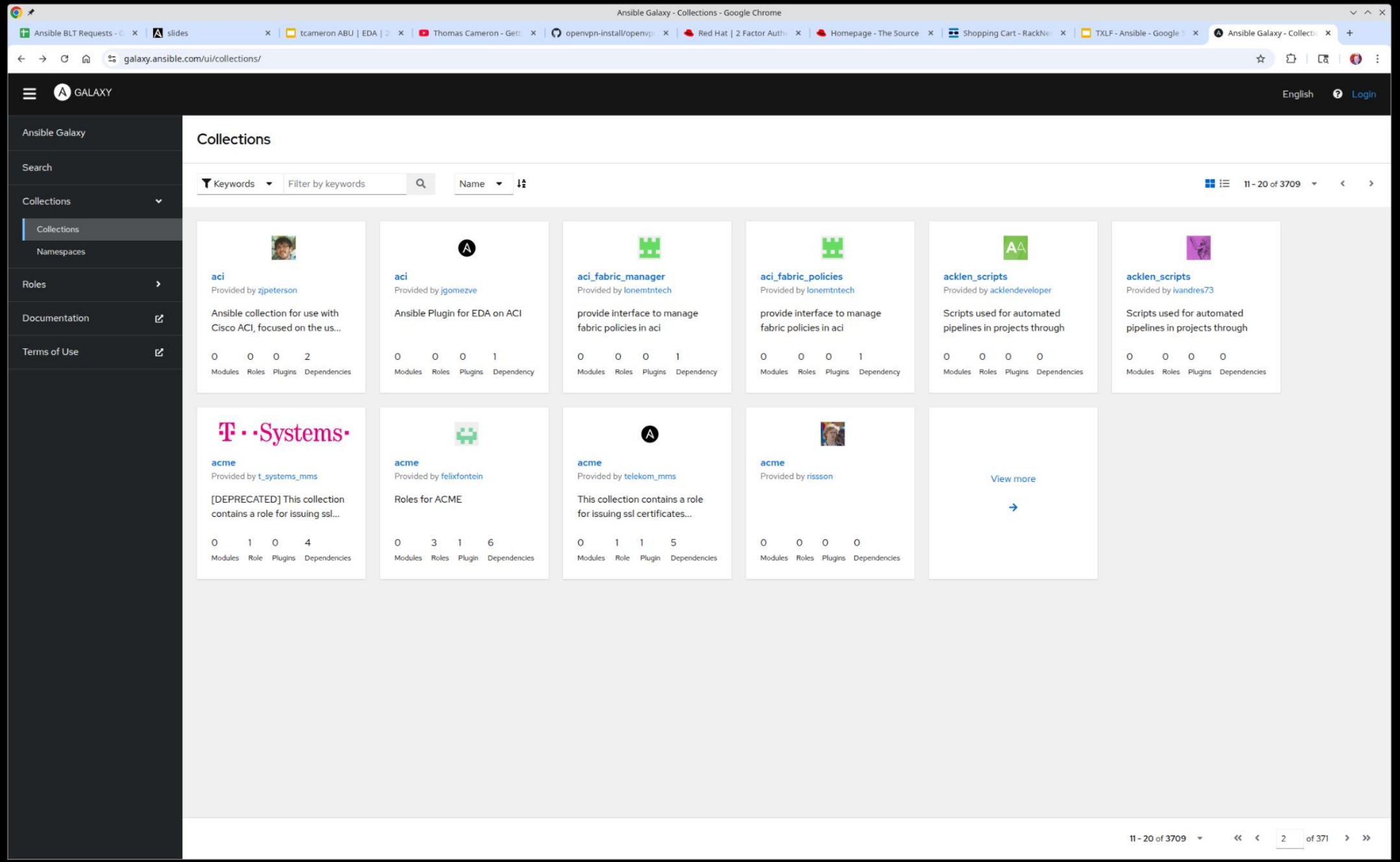
You can also install collections from Ansible Galaxy

From the command line, use ansible-galaxy:

- `ansible-galaxy collection list`
- `ansible galaxy collection install`

Source:
Insert source data here
Insert source data here


(ansible) txlf1 — Konsole
New Tab
Split View
Copy
Paste
Find...
ansible@txlf1:~$ ansible-galaxy collection list
[WARNING]: Collection at
'/usr/share/ansible/collections/ansible_collections/openstack/cloud' do
es not
have a MANIFEST.json file, nor has it galaxy.yml: cannot detect version
.
/usr/lib/python3.13/site-packages/ansible_collections
Collection Version
amazon.aws 9.5.0
ansible.netcommon 7.2.0
ansible.posix 1.6.2
ansible.utils 5.1.2
ansible.windows 2.8.0
arista.eos 10.1.1
awx.awx 24.6.1
azure.azcollection 3.3.1
check_point.mgmt 6.4.0
chocolatey.chocolatey 1.5.3
cisco.aci 2.11.0
cisco.asa 6.1.0
cisco.dnac 6.31.3
cisco.intersight 2.1.0
cisco.ios 9.2.0


Ansible Galaxy - Collections - Google Chrome
galaxy.ansible.com/ui/collections/
GALAXY
English
Login
Ansible Galaxy
Search
Collections
Collections
Namespaces
Roles
Documentation
Terms of Use
Collections
Keywords
Filter by keywords
Name
11 - 20 of 3709
aci
Provided by zjpeterson
Ansible collection for use with Cisco ACI, focused on the us...
0 Modules 0 Roles 0 Plugins 2 Dependencies
aci
Provided by jgomezve
Ansible Plugin for EDA on ACI
0 Modules 0 Roles 0 Plugins 1 Dependency
aci_fabric_manager
Provided by lonemtntech
provide interface to manage fabric policies in aci
0 Modules 0 Roles 0 Plugins 1 Dependency
aci_fabric_policies
Provided by lonemtntech
provide interface to manage fabric policies in aci
0 Modules 0 Roles 0 Plugins 1 Dependency
acklen_scripts
Provided by acklendeveloper
Scripts used for automated pipelines in projects through
0 Modules 0 Roles 0 Plugins 0 Dependencies
acklen_scripts
Provided by ivandres73
Scripts used for automated pipelines in projects through
0 Modules 0 Roles 0 Plugins 0 Dependencies
acme
Provided by t_systems_mms
[DEPRECATED] This collection contains a role for issuing ssl...
0 Modules 1 Role 0 Plugins 4 Dependencies
acme
Provided by felixfontein
Roles for ACME
0 Modules 3 Roles 1 Plugin 6 Dependencies
acme
Provided by telekom_mms
This collection contains a role for issuing ssl certificates...
0 Modules 1 Role 1 Plugin 5 Dependencies
acme
Provided by rissson
0 Modules 0 Roles 0 Plugins 0 Dependencies
View more
11 - 20 of 3709
2
of 371

Requirements for Ansible

Set up ssh keys for the ansible user on the management machine

- Use ssh-keygen

Source:
Insert source data here
Insert source data here

Terminal - ansible@txlf1:~

File Edit View Terminal Tabs Help

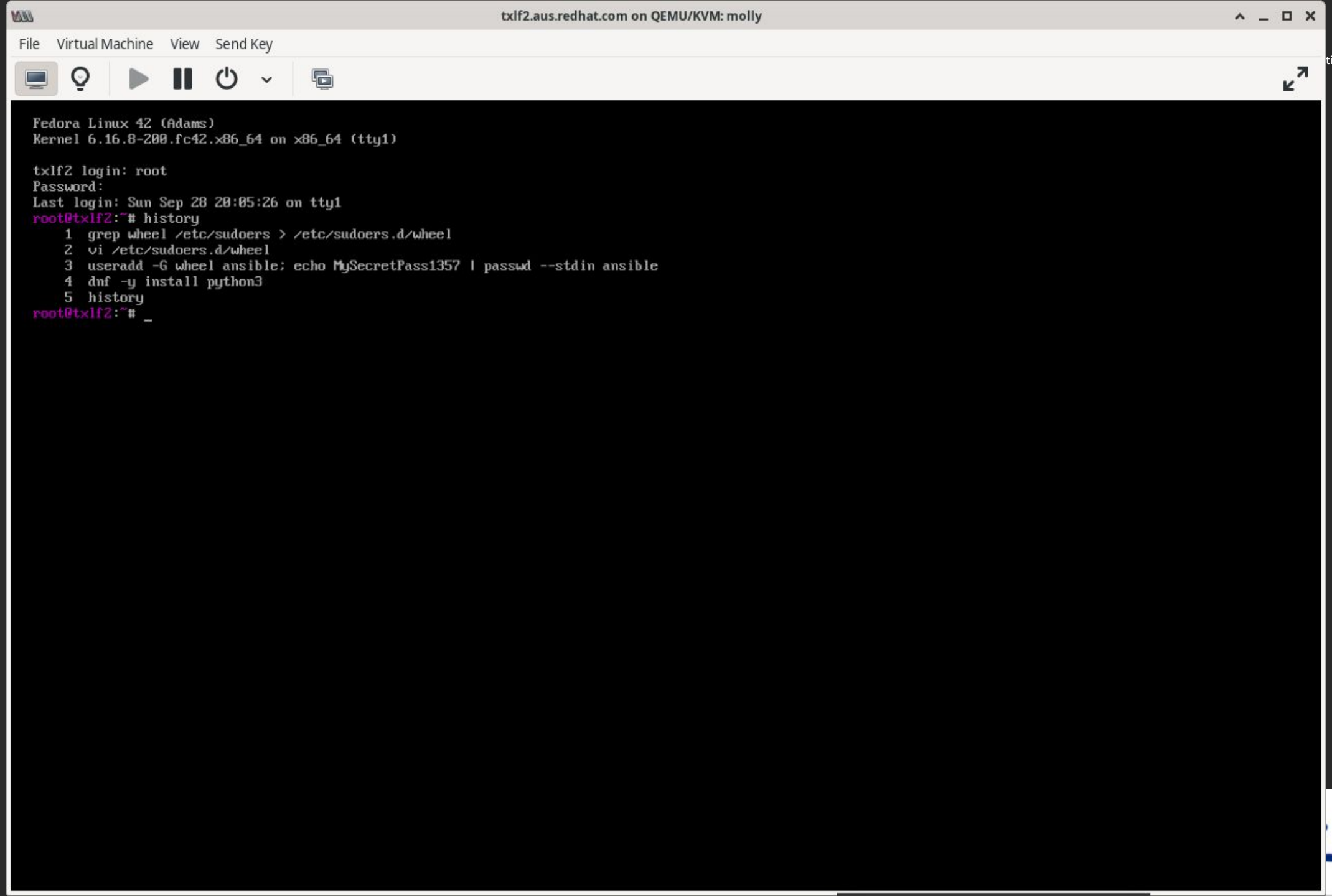
```
ansible@txlf1:~$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ansible/.ssh/id_ed25519):
Created directory '/home/ansible/.ssh'.
Enter passphrase for "/home/ansible/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ansible/.ssh/id_ed25519
Your public key has been saved in /home/ansible/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:JmmnBrp/Lbdl3khD/zDVCr5vw2vjQUFNWhvF+JY/shA ansible@txlf1
The key's randomart image is:
+--[ED25519 256]--+
|              .*=|
|             ..o=|
|              oo.|
|       .   E   +o|
|    . + S . o o.o|
|   . o = . + = o.|
|   .   o.  = *.= .|
|    . .o o= + **. |
|   .... o..o o==+ |
+----[SHA256]-----+
ansible@txlf1:~$
```

Requirements for Ansible

On the managed servers:

- Set up the ansible user with password
- Set up wheel access
- Ensure python3 is installed

Source:
Insert source data here
Insert source data here

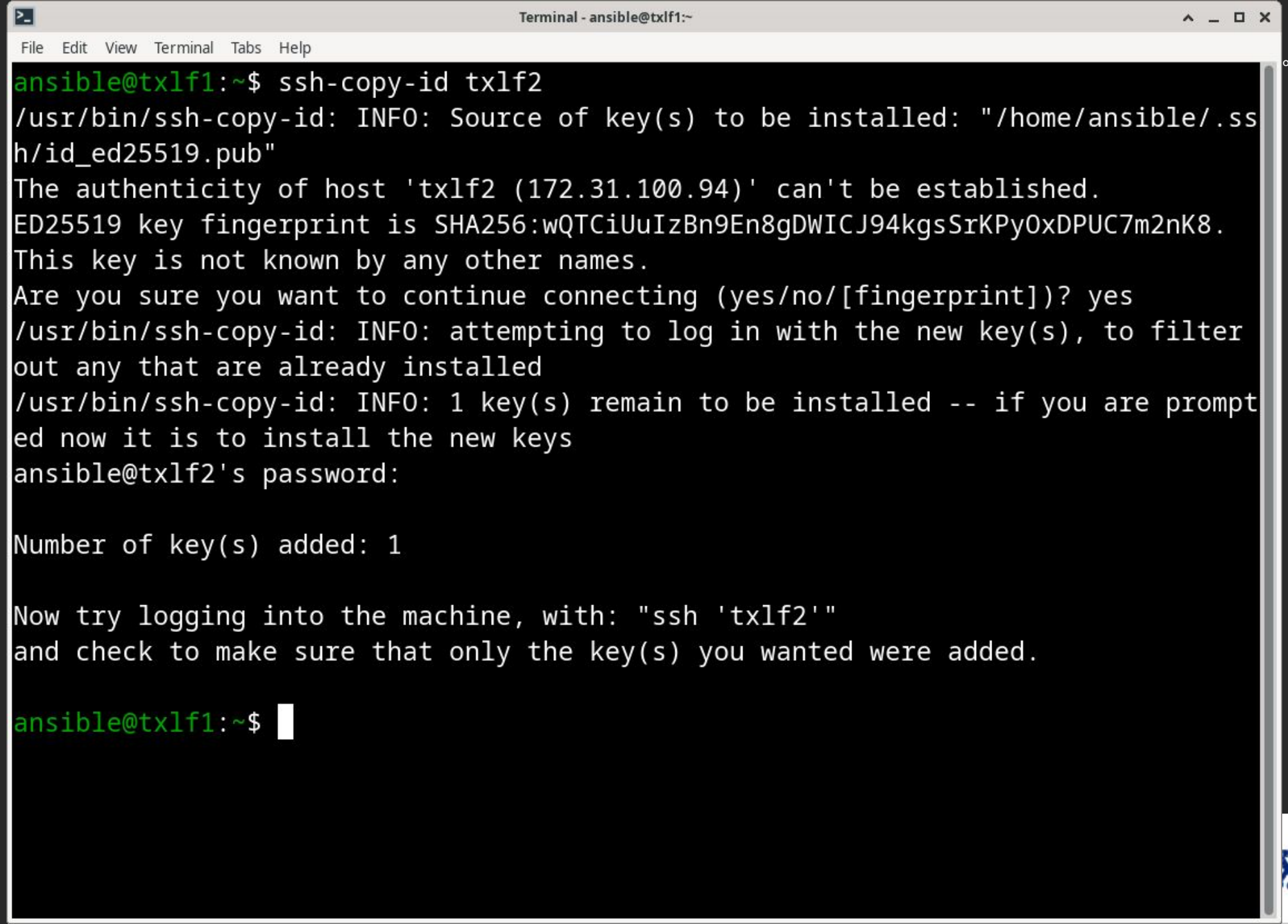


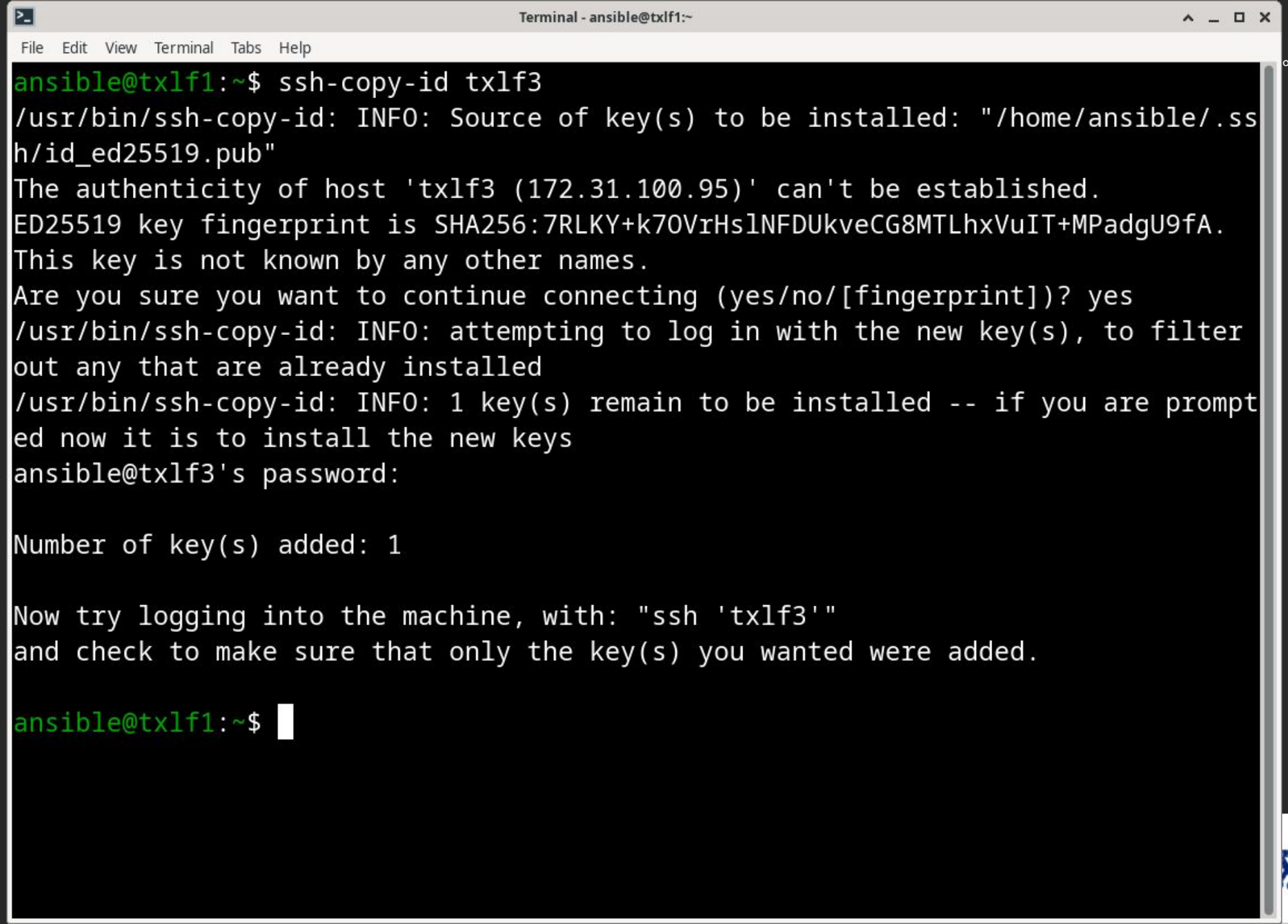
Requirements for Ansible

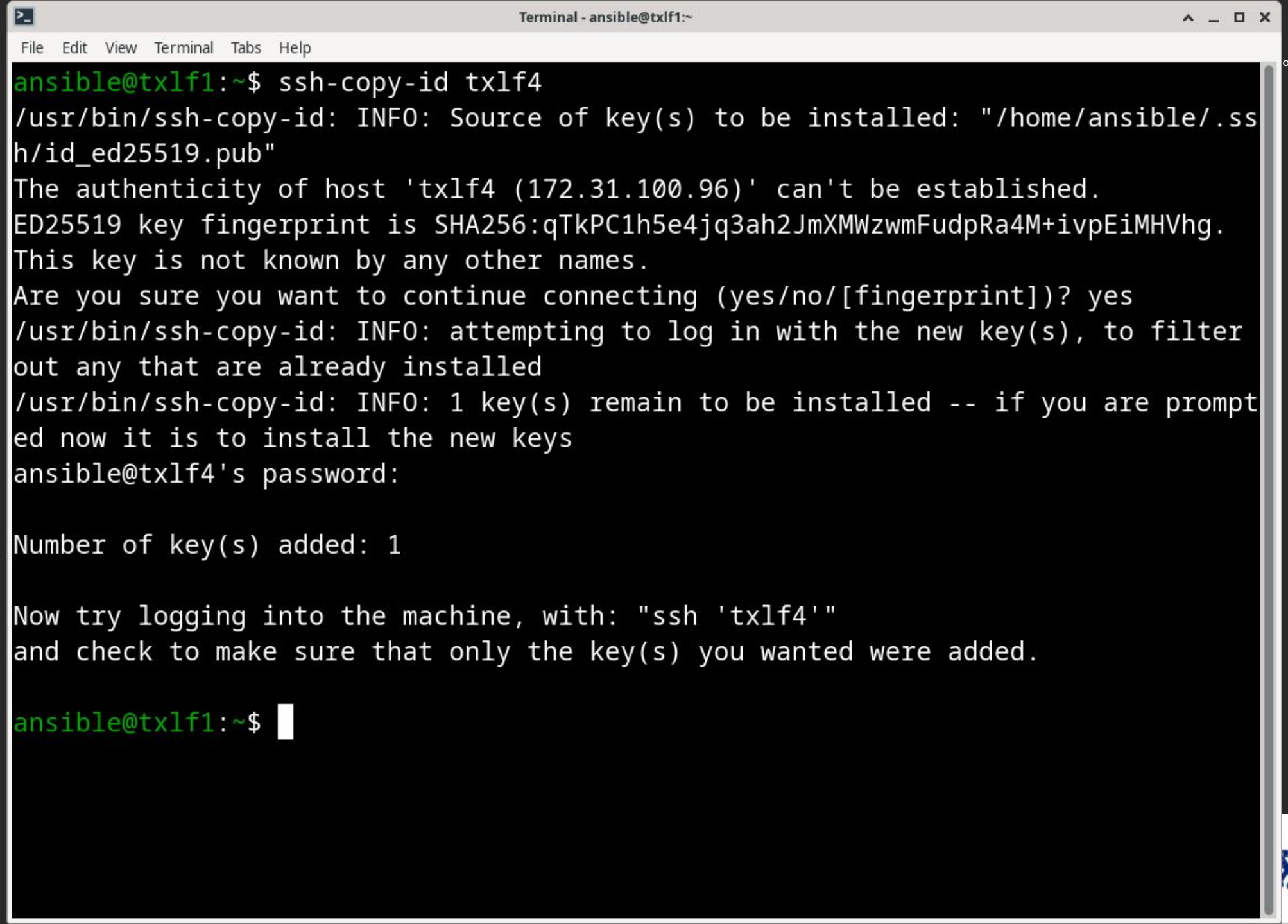
On the management server:

- Use ssh-copy-id to copy the public ssh key to each managed server

Source:
Insert source data here
Insert source data here







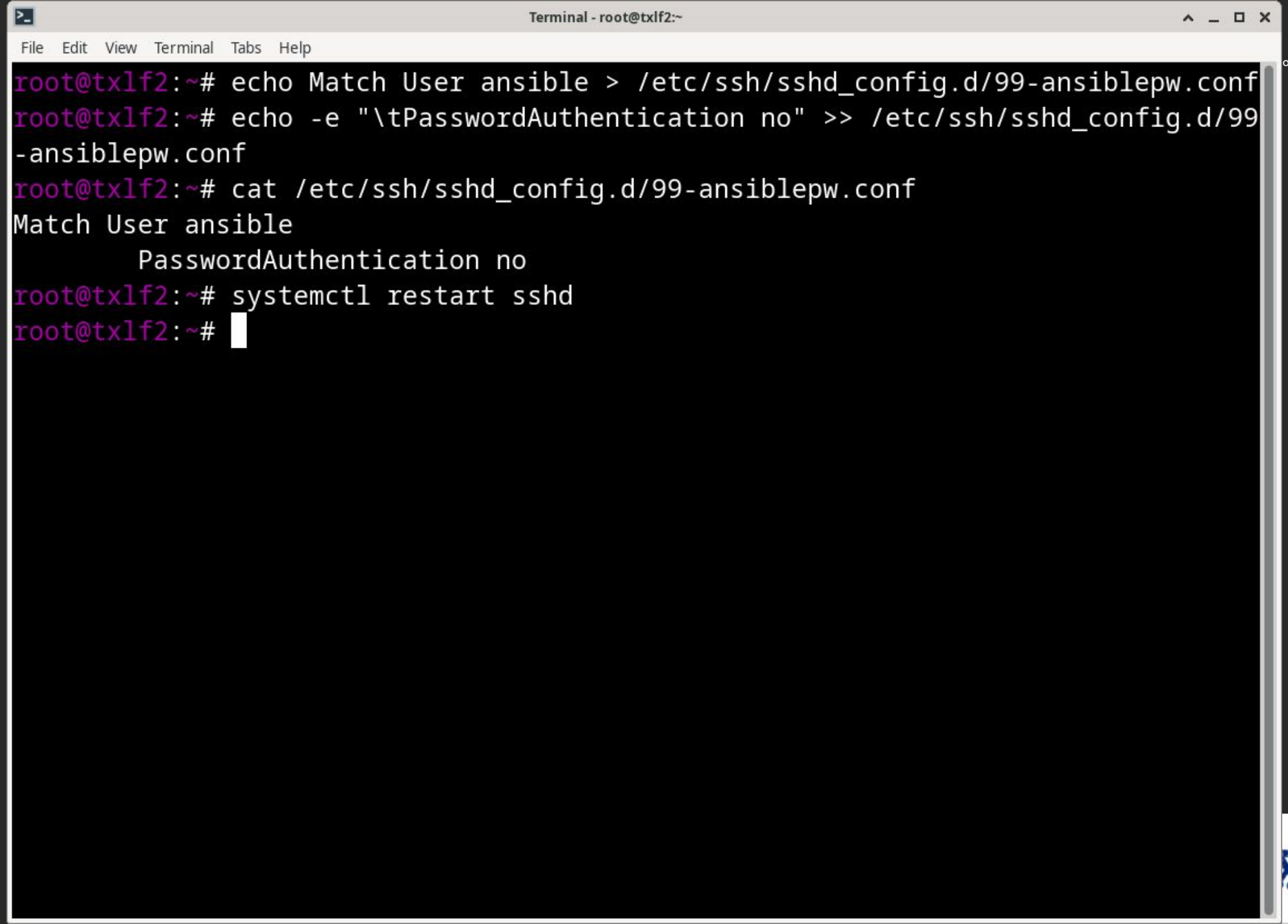
Requirements for Ansible

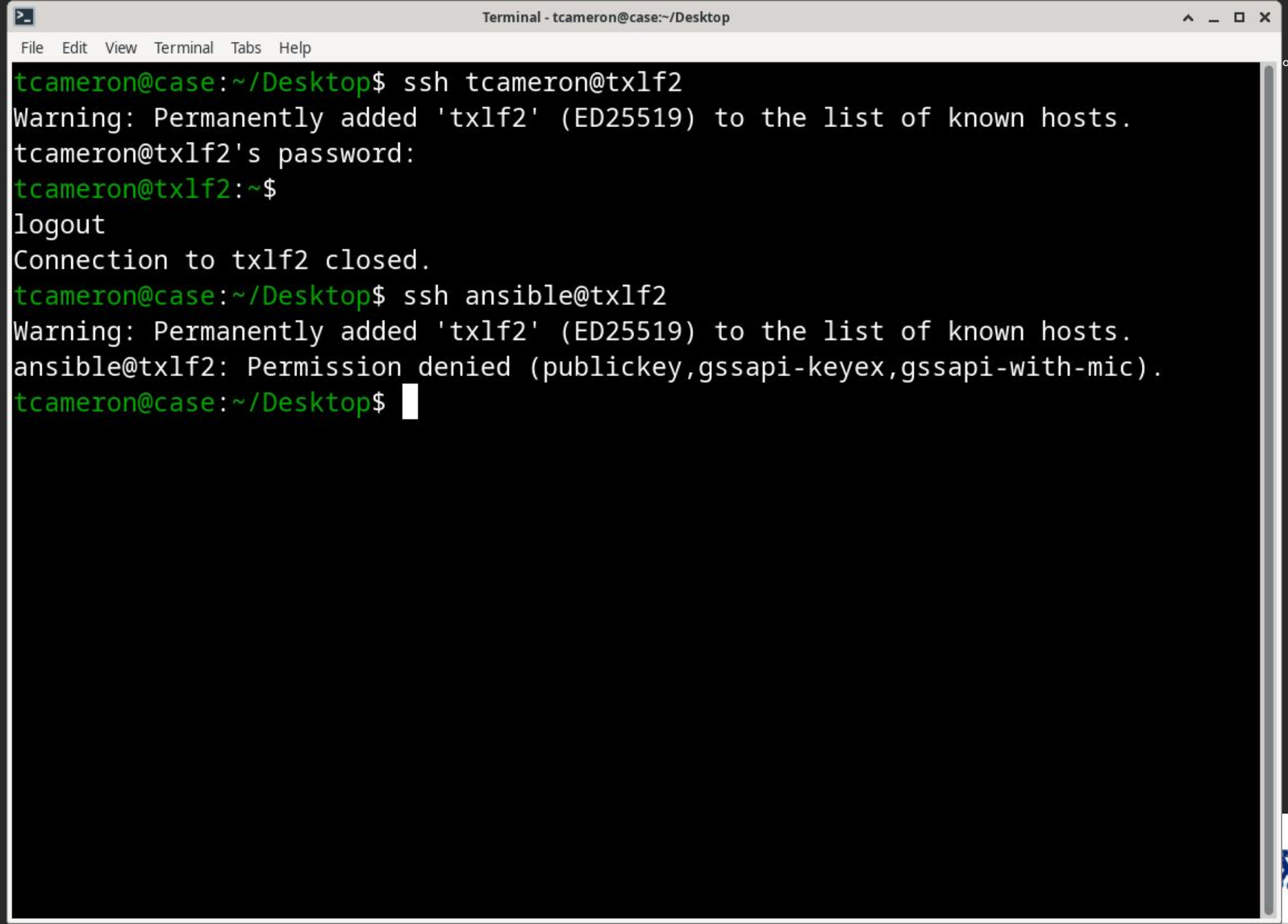
Optionally:

- **AFTER** the ssh keys are distributed, disable password logins for the ansible user.

```
echo Match User ansible > /etc/ssh/sshd_config.d/99-ansiblepw.conf
echo -e "\tPasswordAuthentication no" >> /etc/ssh/sshd_config.d/99-ansiblepw.conf
```


Source:
Insert source data here
Insert source data here





Requirements for Ansible

Optionally:

- You can get **REALLY** crazy with setting firewall rules such that you cannot ssh into the managed server except from the management machine.
  - `sudo firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source address="192.168.1.25" port port="22" protocol="tcp" accept'`
  - `sudo firewall-cmd --permanent --remove-service=ssh`
  - `sudo firewall-cmd --reload`

Source:
Insert source data here
Insert source data here

Red Hat TXLF

Set Up Inventory

Per directory or global

Ansible Inventory

The inventory file is a list of the machines you want to manage. It can be YAML or INI formatted

- For this session, we'll use INI format

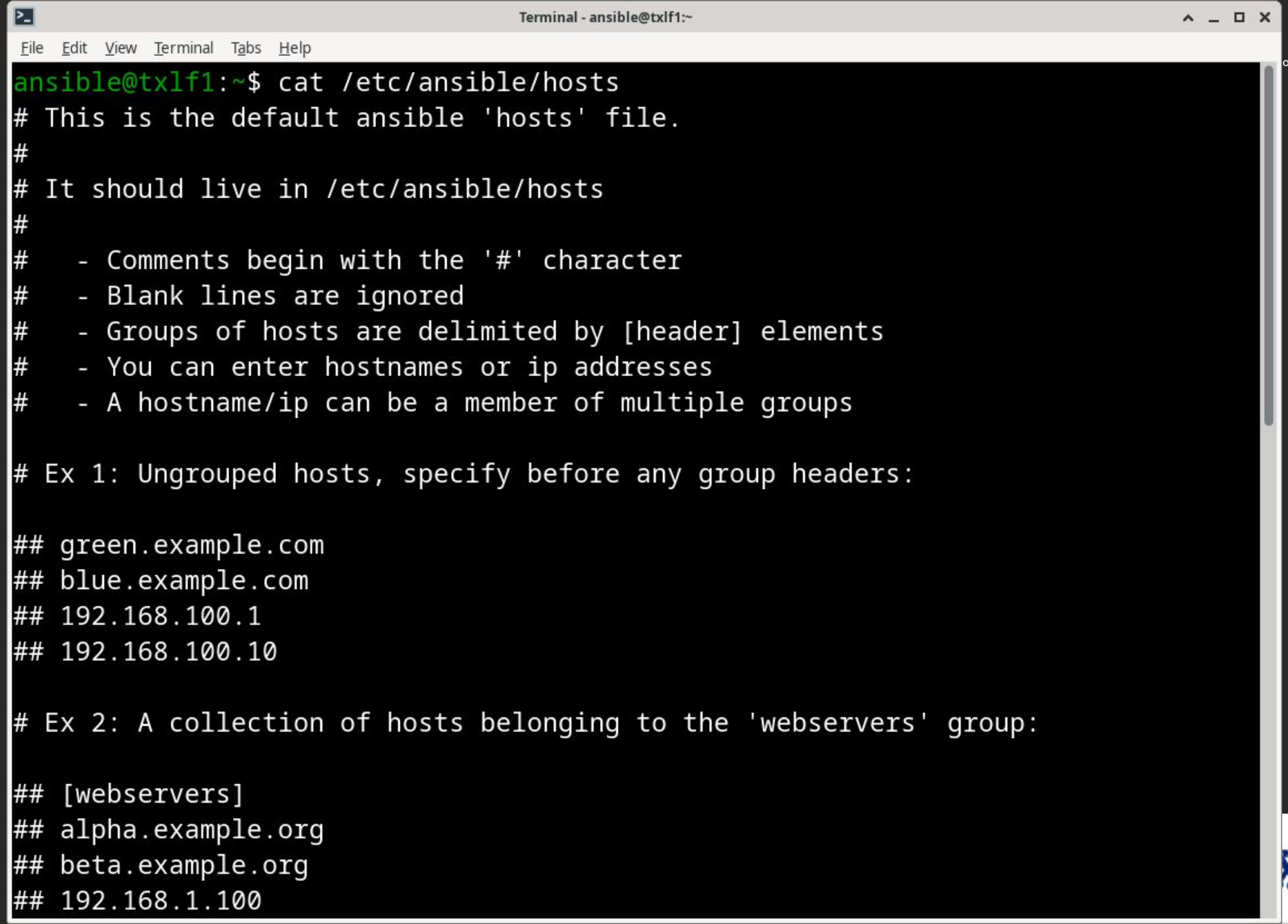
Source:

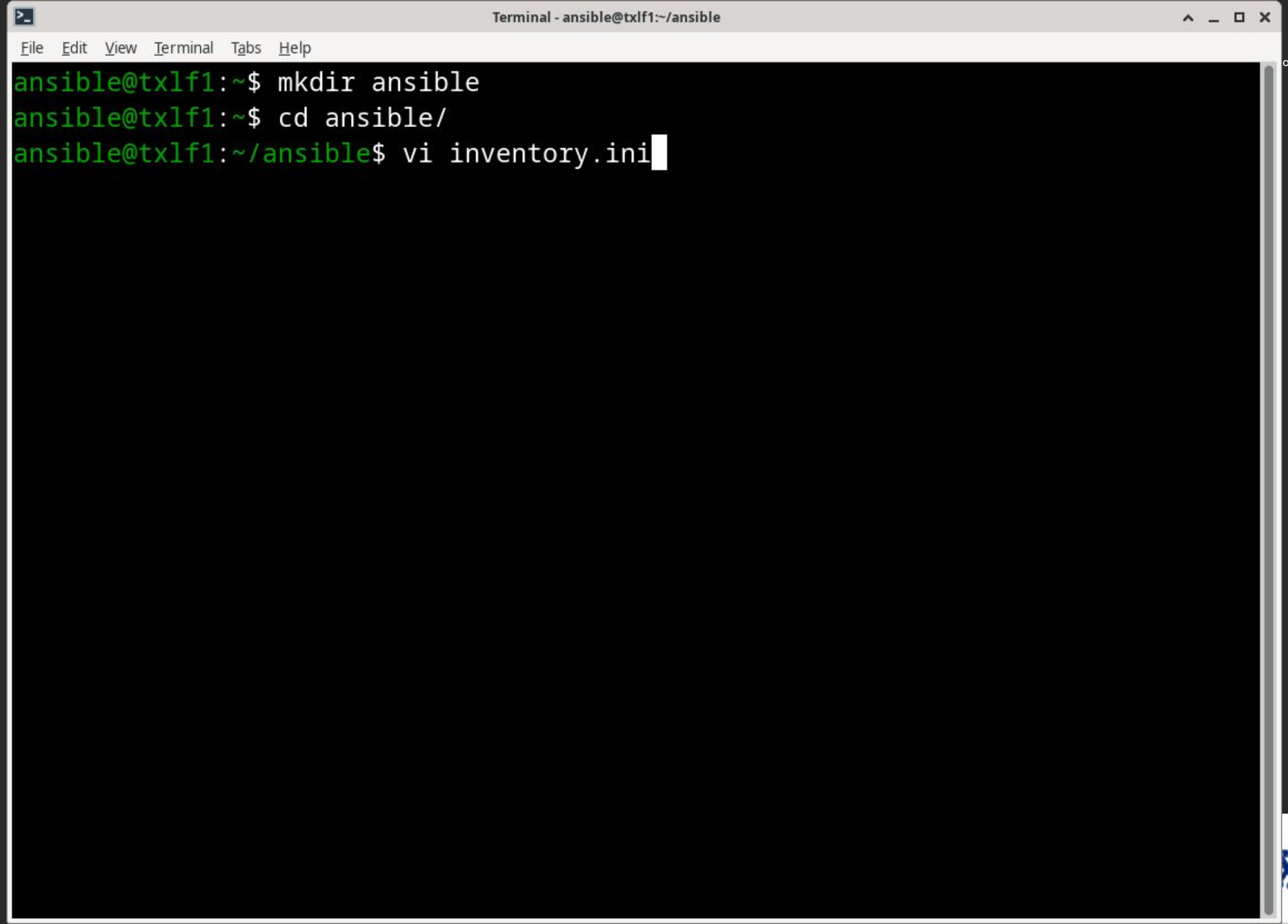
Ansible Inventory

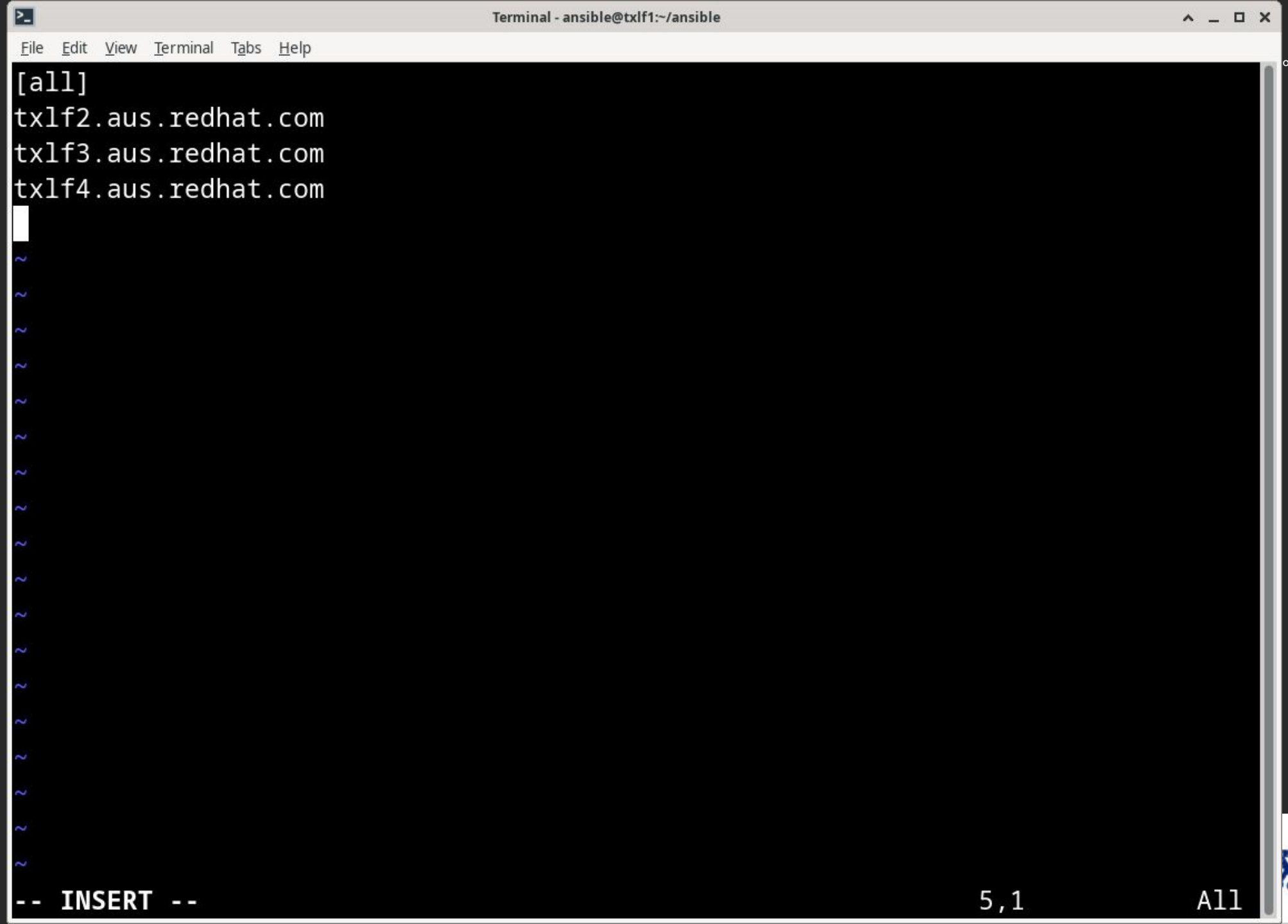
You can use the global inventory file in /etc/ansible/hosts

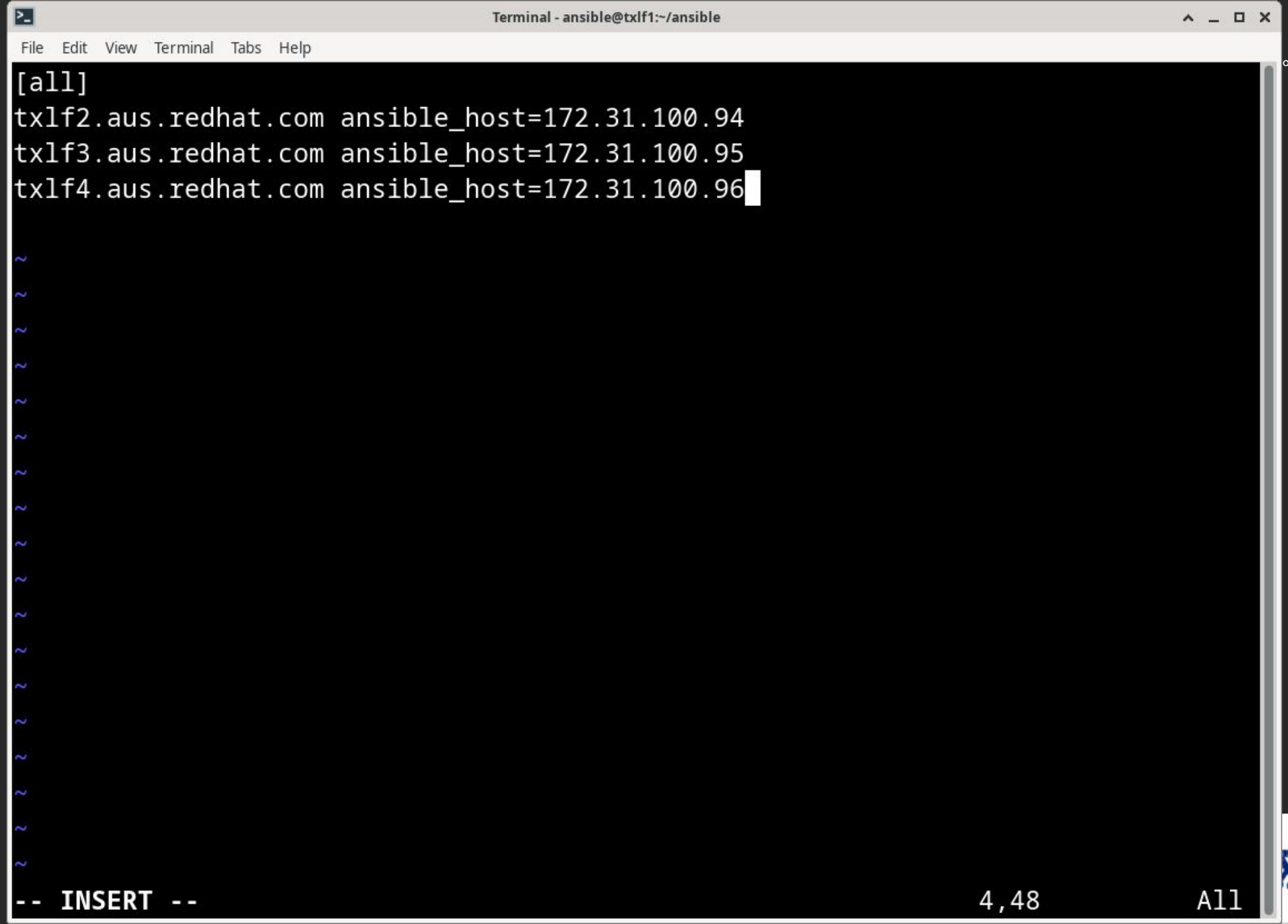
Or, you can use an inventory file in the working directory you're running Ansible commands from.

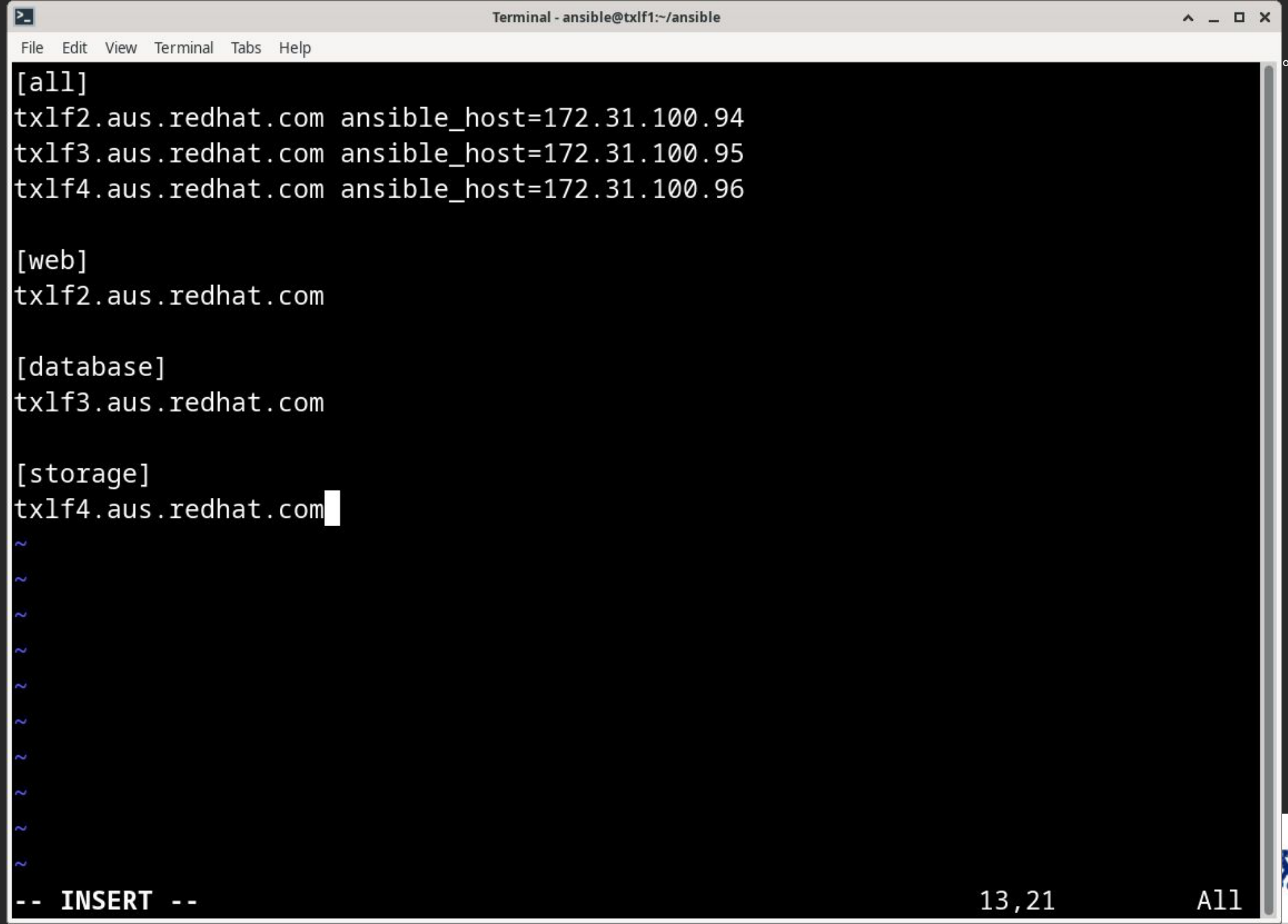
Source:











Ansible Configuration File

There’s a cheat to set it up

Ansible Configuration File

The Ansible configuration file is used to define settings when you run Ansible commands.

- As with inventories, there is a global one and you can create a local one.
  - /etc/ansible/ansible.cfg
  - ./ansible.cfg

Source:

Ansible Configuration File

You can create a new one using ansible-config init

- When you run ansible-config init it will create a config and pipe it to your defined pager (usually less).
  - Note that this config has the defaults **enabled**.
  - If you want to dump it to a config file, use
    - `ansible-config init > ansible.cfg`

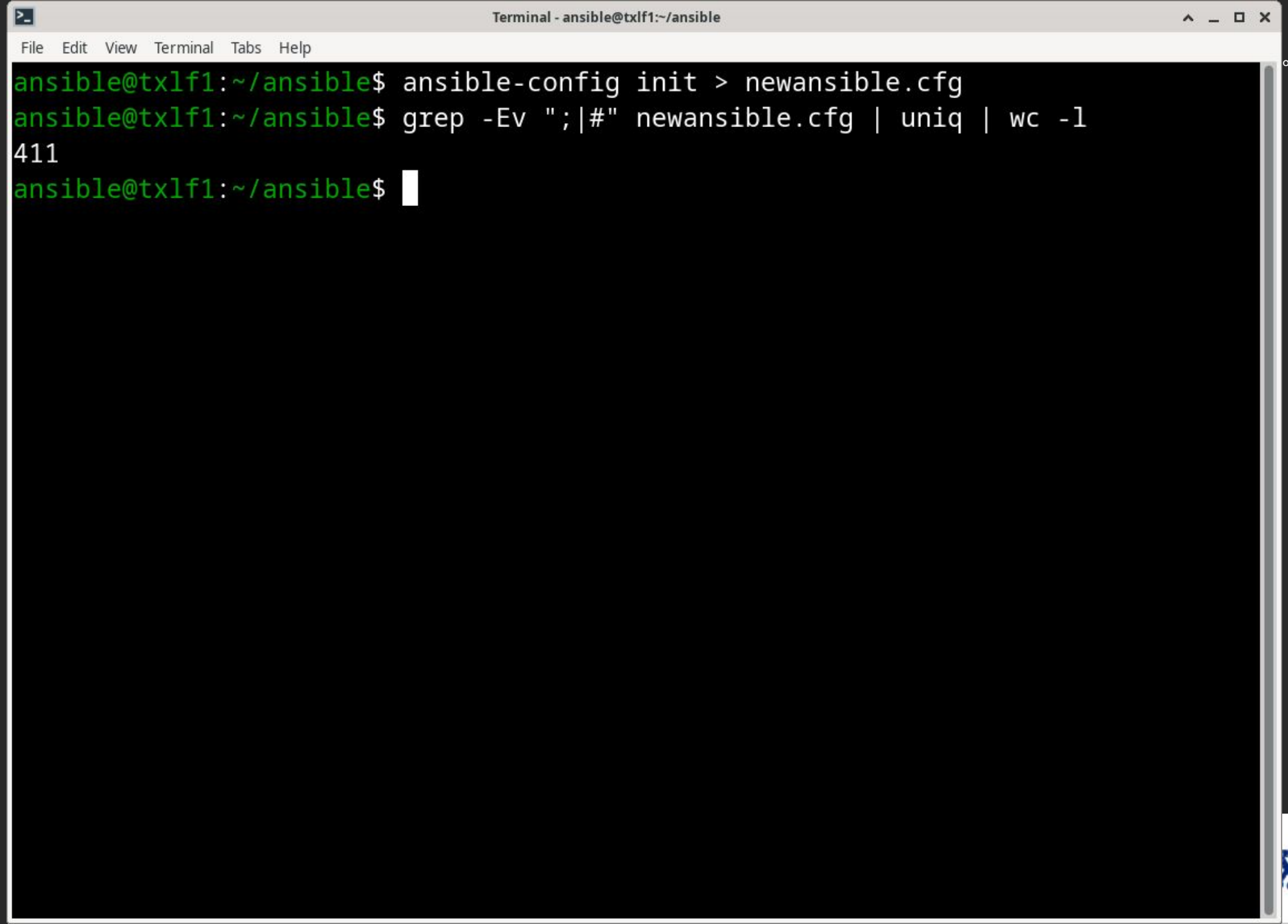
Source:

Ansible Configuration File

You can create a new one using ansible-config init

- When you run ansible-config init it will create a config and pipe it to your defined pager (usually less).
  - Note that this config has the defaults **enabled**.
  - If you want to dump it to a config file, use
    - `ansible-config init > ansible.cfg`
  - **I don't recommend this.**

Source:

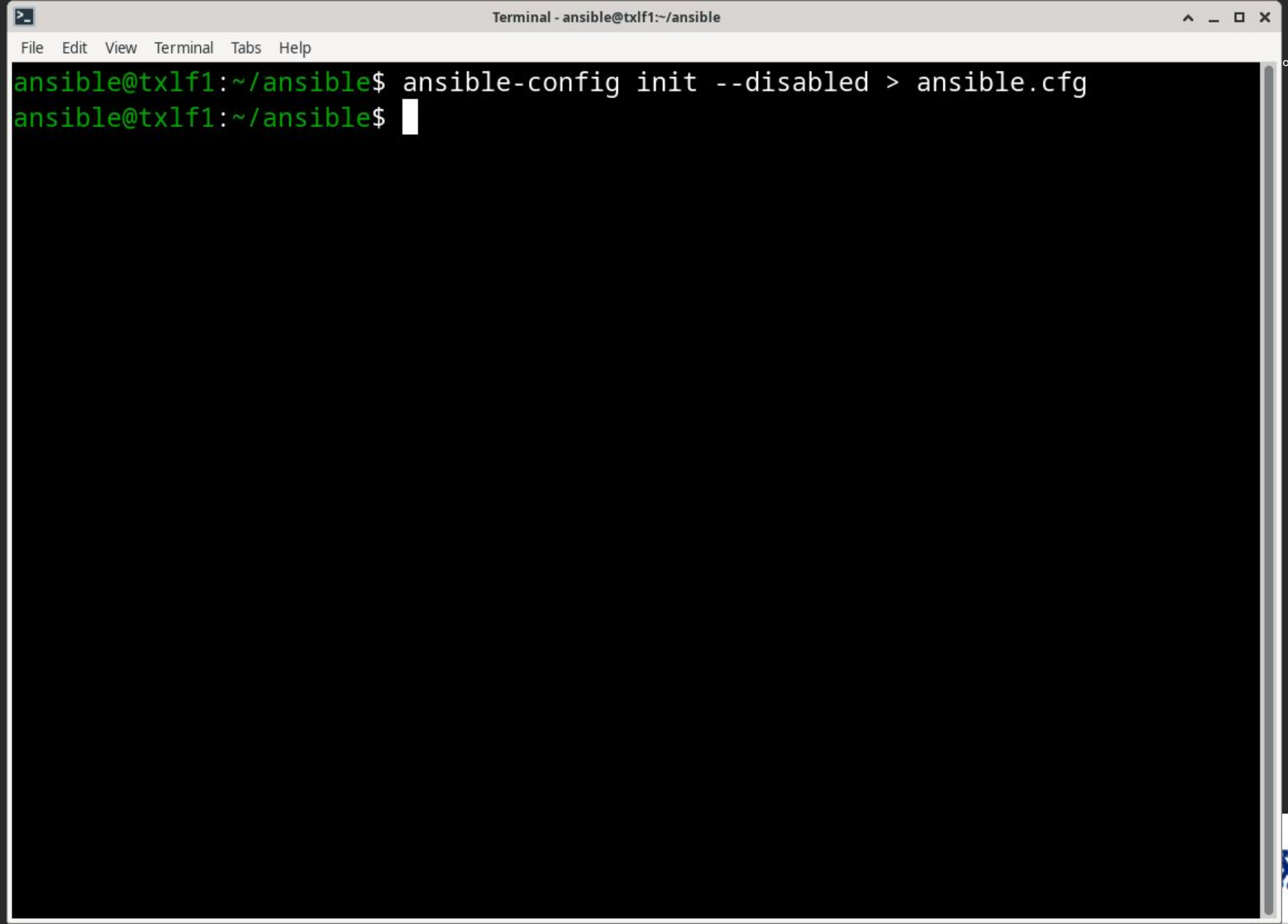


Ansible Configuration File

You can create a new one using ansible-config init

- I recommend you create a config with everything **disabled** and dump it to a file:
  - `ansible-config init --disabled > ansible.cfg`

Source:



Ansible Configuration File

There are 4 fields you need to set at a minimum

- become
  - Do you want privilege escalation?
- become_method
  - What privilege escalation tool do you use?
- inventory
  - What is the path to the inventory file?
- remote_user
  - What is the login name on the managed servers?

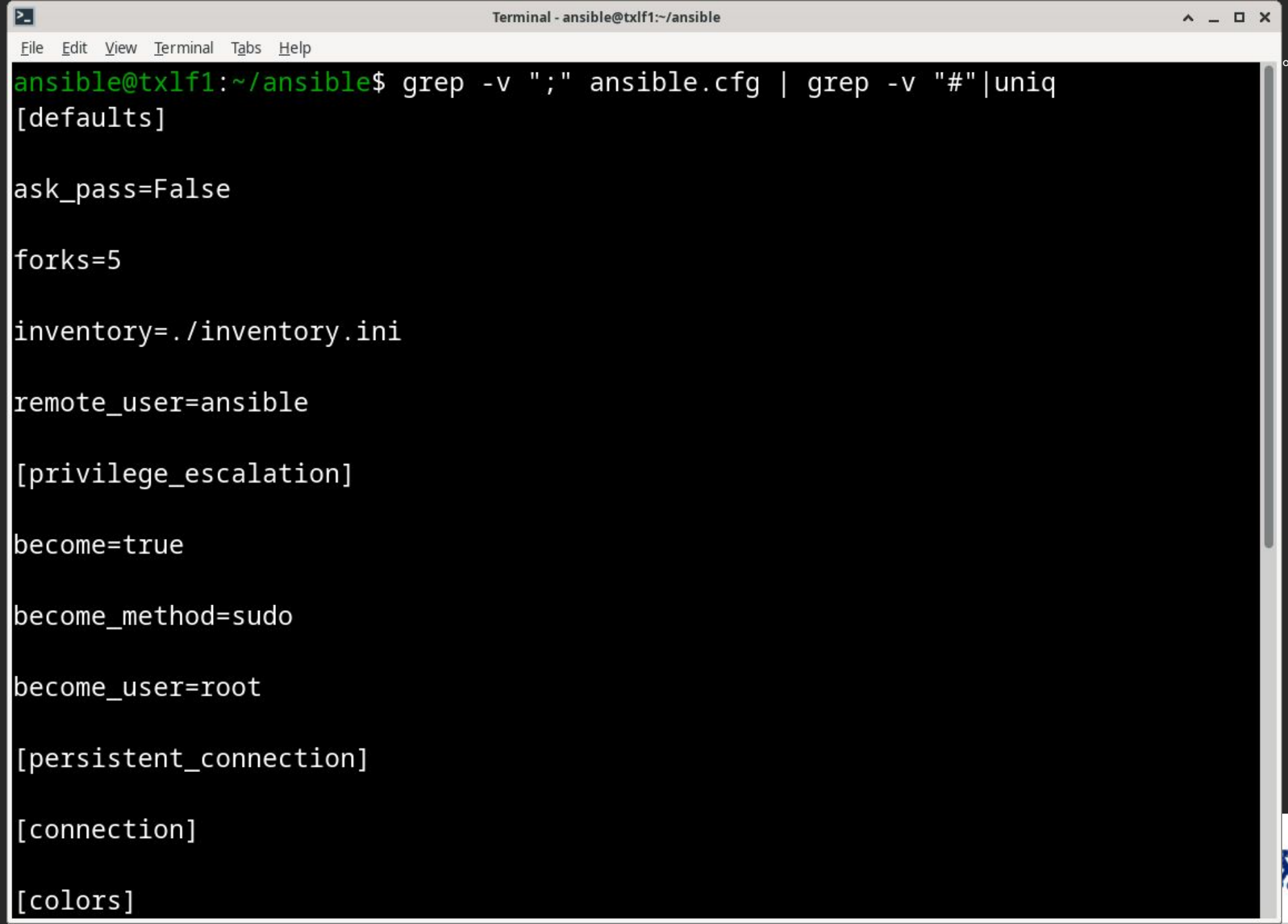
Source:

Ansible Configuration File

There are 3 other fields you might want to set

- become_user
  - If you use sudo, it defaults to root. If you want another user, set it here
- ask_pass
  - If you're using ssh keys, you don't need to set this.
- forks
  - How many ssh sessions do you want spawn? If you have a beefy management node, bump it up. On my 64GB workstation, I have it set to 20
  - On this demo machine, 5

Source:



Playbooks

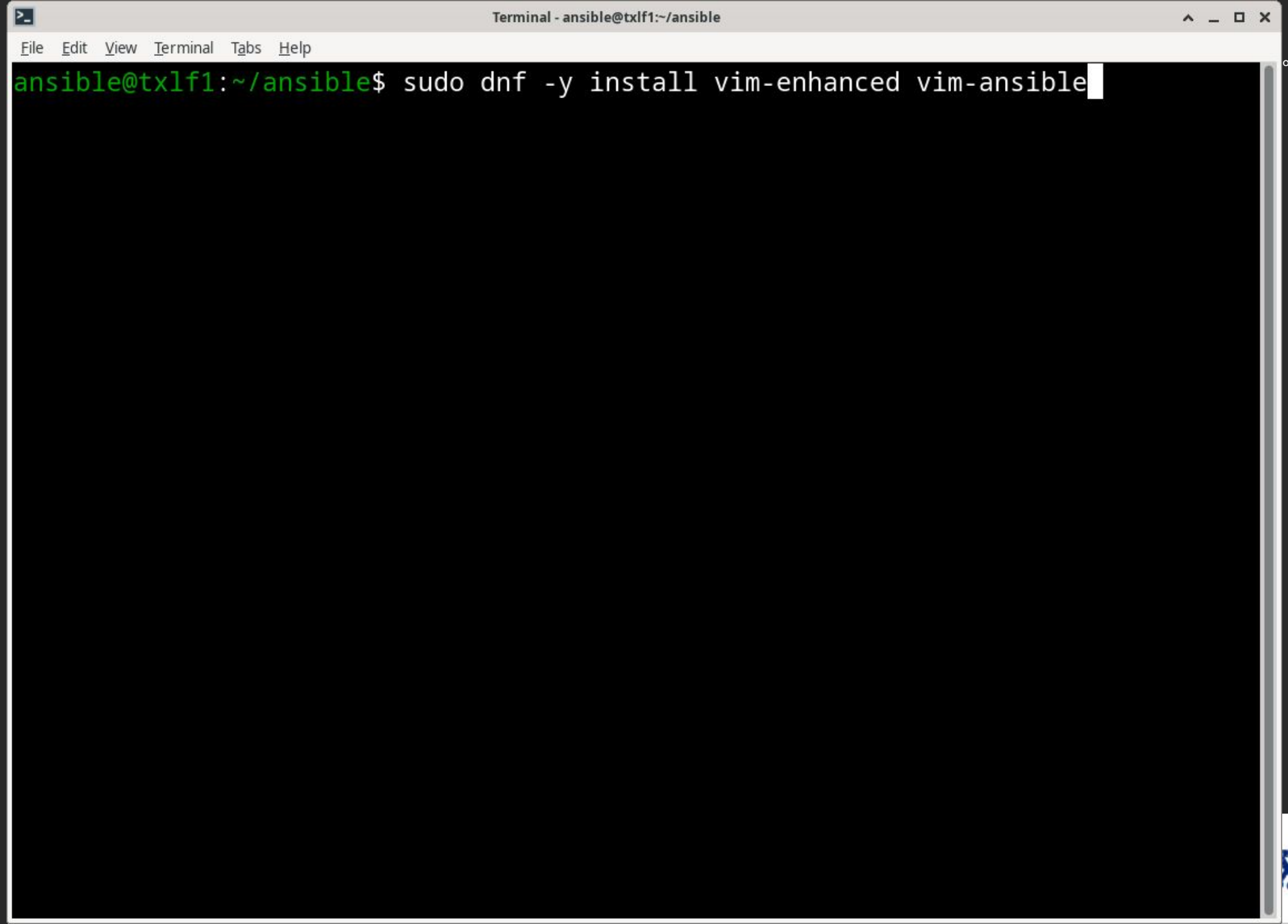
The basics

Playbooks

Playbooks are just YAML files. There are some basics that will be the same almost every time.

- Since it's YAML, it should be named with either `*.yml` or `*.yaml`. For instance, `myplaybook.yml` or `myplaybook.yaml`.
  - If you're a vim user, I strongly recommend you install vim-enhanced and vim-ansible (if available).
  - There are other packages for other editors

Source:



```
Total size of inbound packages is 10 MiB. Need to download 10 MiB.
After this operation, 42 MiB extra will be used (install 42 MiB, remove 0 B).
[1/7] vim-ansible-0:3.4-4.fc42.noarch   100% |  67.9 KiB/s |  16.8 KiB |  00m00s
[2/7] gpm-libs-0:1.20.7-51.fc42.x86_64  100% |  70.6 KiB/s |  20.3 KiB |  00m00s
[3/7] libsodium-0:1.0.20-4.fc42.x86_64  100% | 970.2 KiB/s | 175.6 KiB |  00m00s
[4/7] vim-filesystem-2:9.1.1775-1.fc42. 100% |  88.2 KiB/s |  15.4 KiB |  00m00s
[5/7] vim-enhanced-2:9.1.1775-1.fc42.x8 100% |   2.7 MiB/s |   2.0 MiB |  00m01s
[6/7] xxd-2:9.1.1775-1.fc42.x86_64      100% | 235.8 KiB/s |  31.4 KiB |  00m00s
[7/7] vim-common-2:9.1.1775-1.fc42.x86_ 100% |   7.8 MiB/s |   8.1 MiB |  00m01s
--------------------------------------------------------------------------------
[7/7] Total                             100% |   5.8 MiB/s |  10.4 MiB |  00m02s
Running transaction
[1/9] Verify package files              100% |  70.0   B/s |   7.0   B |  00m00s
[2/9] Prepare transaction               100% |  61.0   B/s |   7.0   B |  00m00s
[3/9] Installing vim-filesystem-2:9.1.1 100% | 337.1 KiB/s |   4.7 KiB |  00m00s
[4/9] Installing xxd-2:9.1.1775-1.fc42. 100% |   1.1 MiB/s |  38.3 KiB |  00m00s
[5/9] Installing vim-common-2:9.1.1775- 100% |  65.4 MiB/s |  37.2 MiB |  00m01s
[6/9] Installing libsodium-0:1.0.20-4.f 100% |  42.3 MiB/s | 390.2 KiB |  00m00s
[7/9] Installing gpm-libs-0:1.20.7-51.f 100% |   5.3 MiB/s |  32.4 KiB |  00m00s
[8/9] Installing vim-enhanced-2:9.1.177 100% |  79.2 MiB/s |   4.2 MiB |  00m00s
[9/9] Installing vim-ansible-0:3.4-4.fc 100% |   9.1 KiB/s |  27.0 KiB |  00m03s
Complete!
ansible@txlf1:~/ansible$ vi playbook.yml
```

Playbooks

Playbooks are just YAML files. There are some basics that will be the same almost every time.

- **Tabs are two spaces.** This is very important. And not a little bit frustrating.
  - If you're using vim-ansible or vim-enhanced, it is smart enough to set your tabs correctly. Otherwise, there are plenty of examples of .vimrc files for working with YAML online

Source:

Playbooks

Playbooks are just YAML files. There are some basics that will be the same almost every time.

- Playbooks use key:value pairs, as in:
  - `setting: {true|false}`

Source:

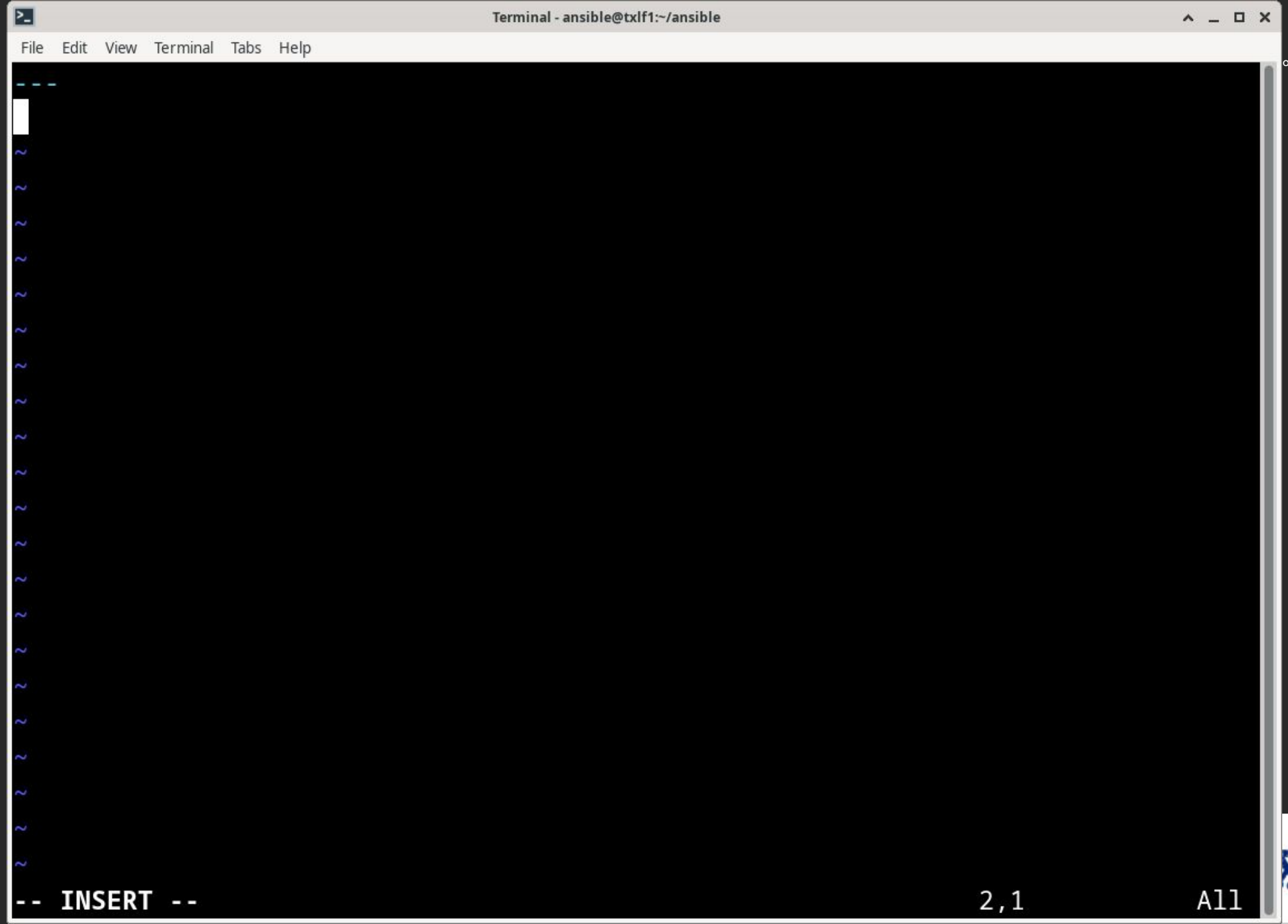
Playbooks

Playbooks are just YAML files. There are some basics that will be the same almost every time.

- The playbook always starts with three dashes:

```
---
```


Source:



Playbooks

Playbooks are just YAML files. There are some basics that will be the same almost every time.

- Next, you'll define the name of the playbook:
  - - name: A logical name for the playbook

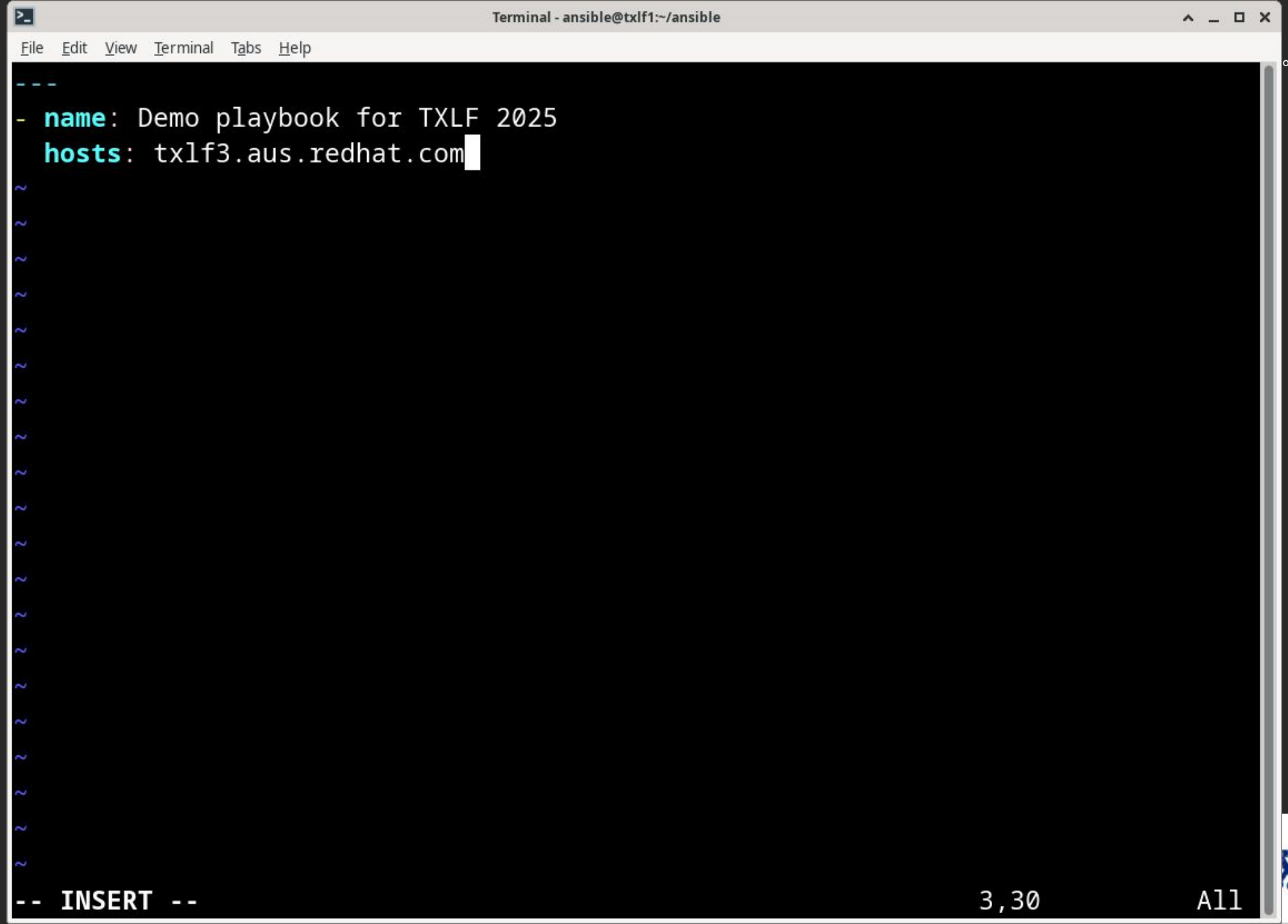
Source:

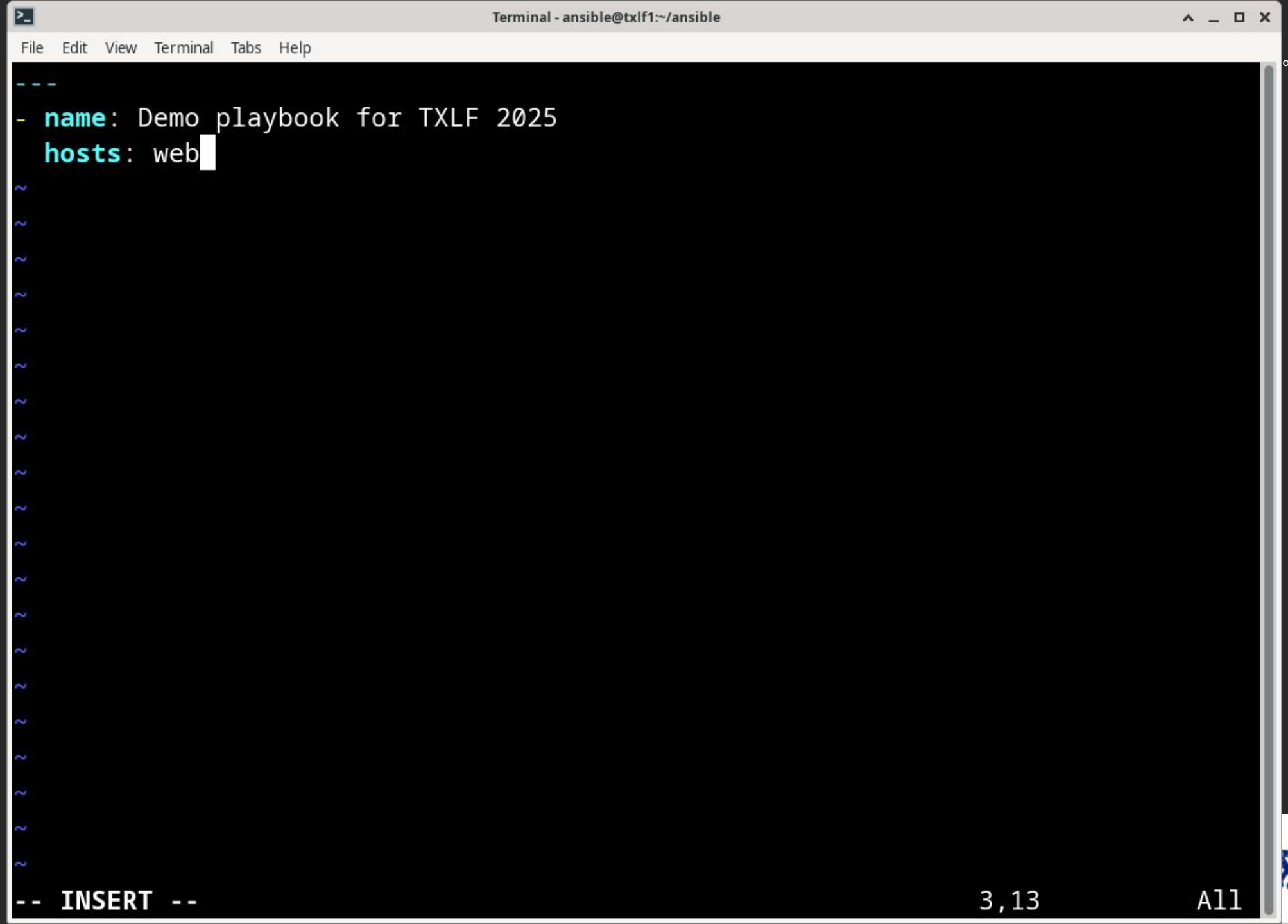
Playbooks

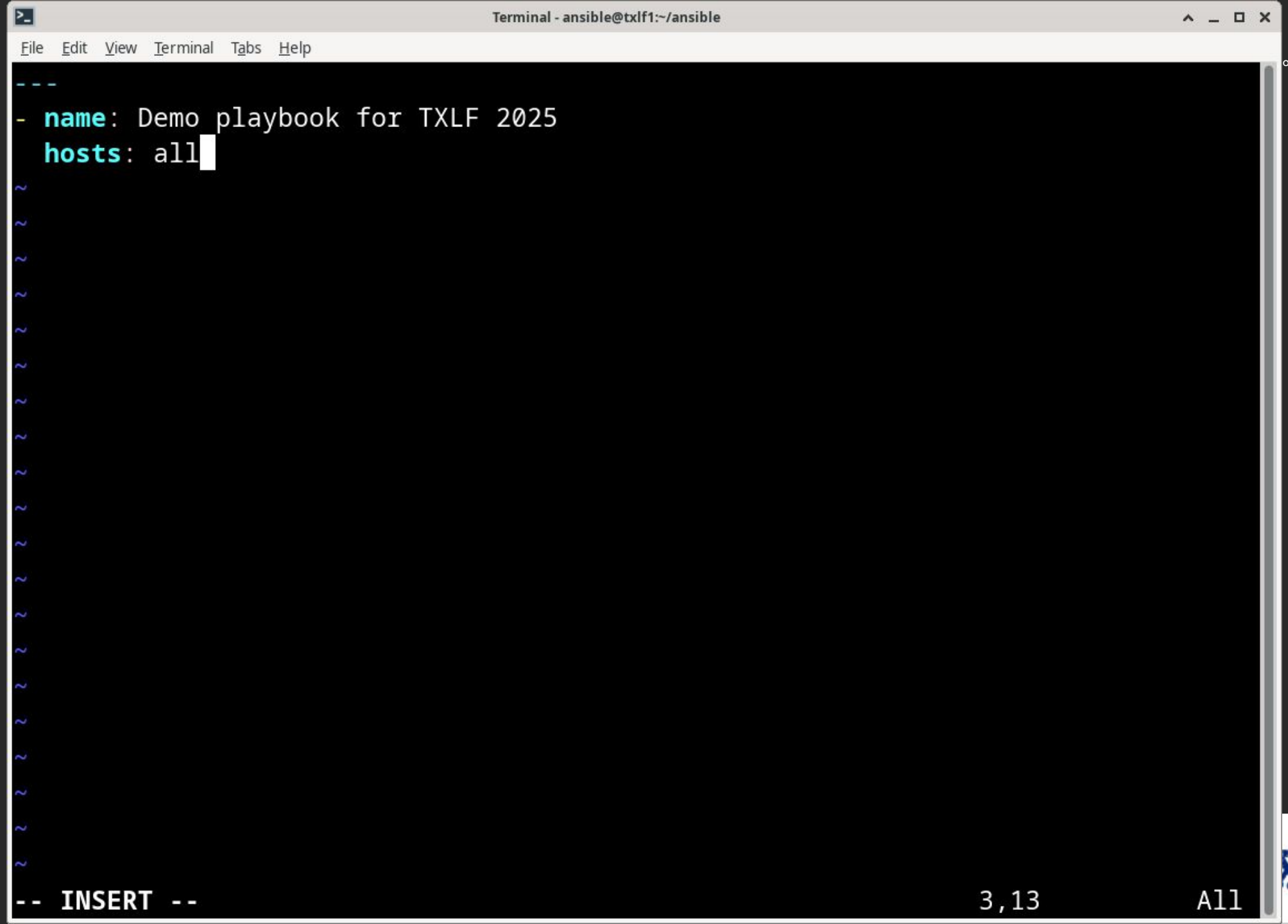
There are a number of fields you can use next. The basics include:

- hosts:
  - The hosts you want to run the playbook on. Use either the hostnames or groups in your inventory

Source:







Playbooks

There are a number of fields you can use next. The basics include:

- gather_facts:
  - When Ansible connects, it launches the `setup` module using python3 on the managed node

Source:

Playbooks

There are a number of fields you can use next. The basics include:

- gather_facts:
  - Setup gathers information about:
    - OS type and version
    - Network interfaces
    - CPU, memory, and storage info

Source:

Playbooks

There are a number of fields you can use next. The basics include:

- gather_facts:
  - Setup gathers information about:
    - Hostname, DNS domain, and architecture
    - Environment variables, python3 version

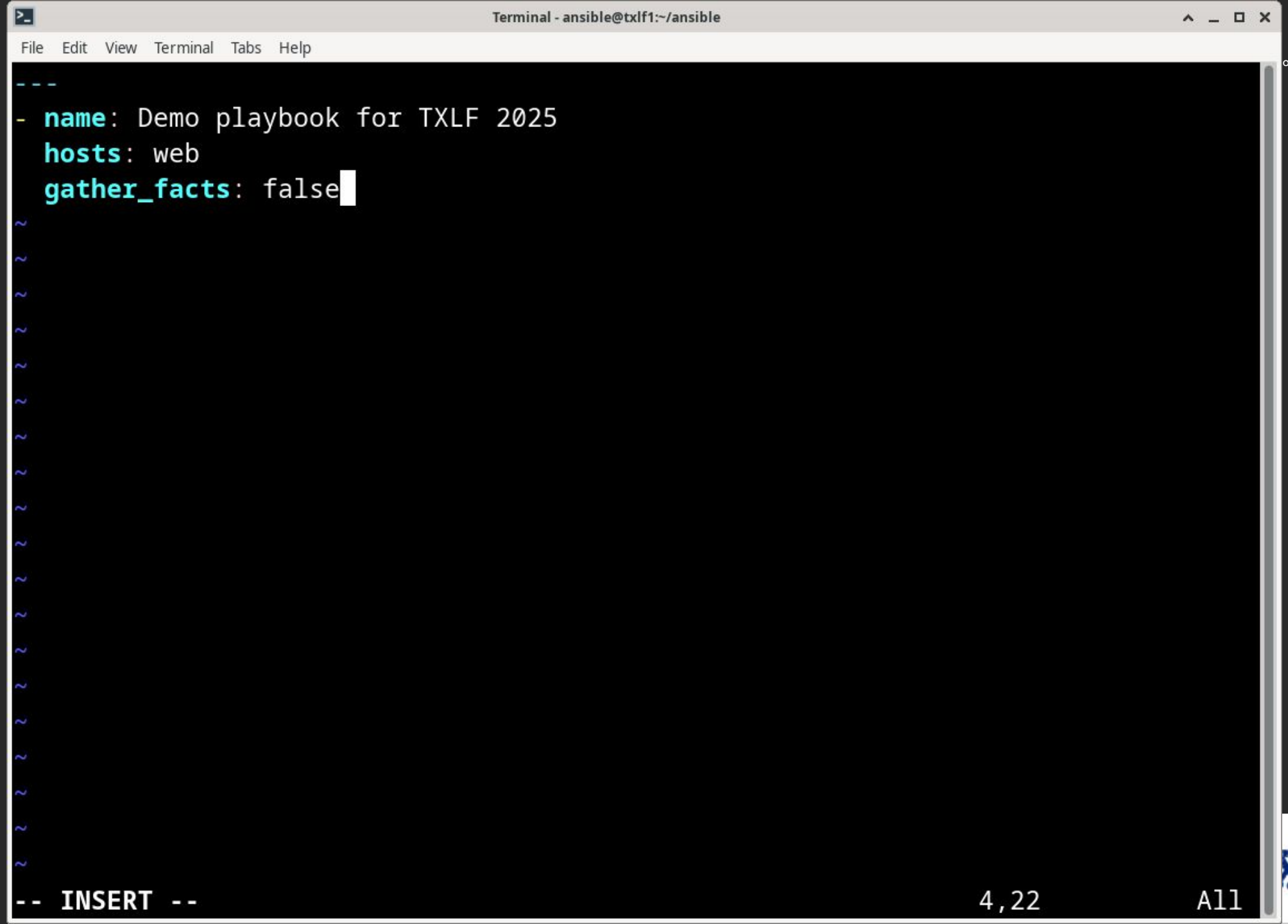
Source:

Playbooks

There are a number of fields you can use next. The basics include:

- gather_facts:
  - If you don't need to gather that information, you can set `gather_facts: false`
  - Note that the information gathered is really useful with more advanced playbooks. More on that later.

Source:

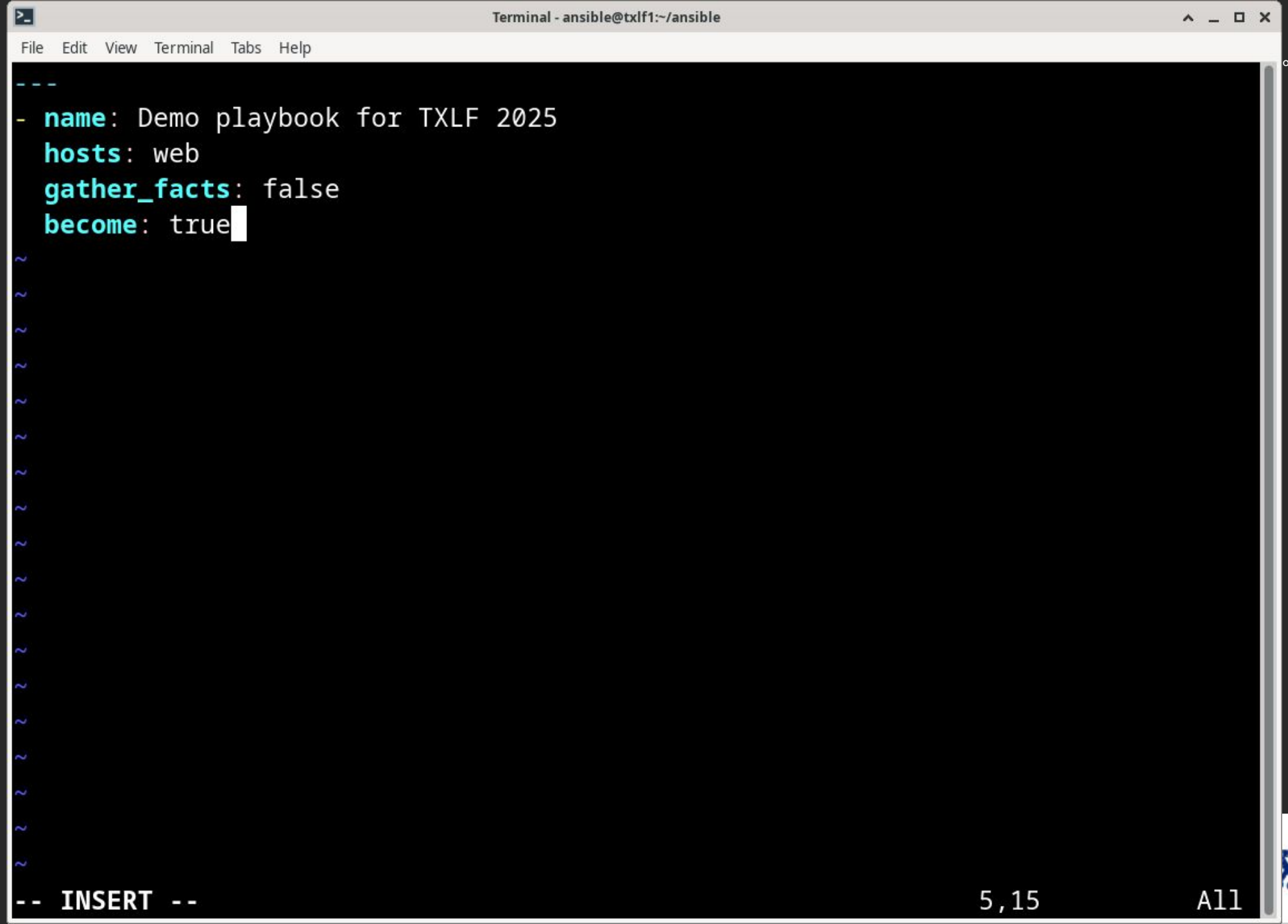


Playbooks

There are a number of fields you can use next. The basics include:

- become:
  - Do you want to escalate privileges
  - If you've already defined `become: true` in your ansible.cfg, this is optional! I just wanted to point out that you can enable or disable values in either ansible.cfg or the playbook.

Source:

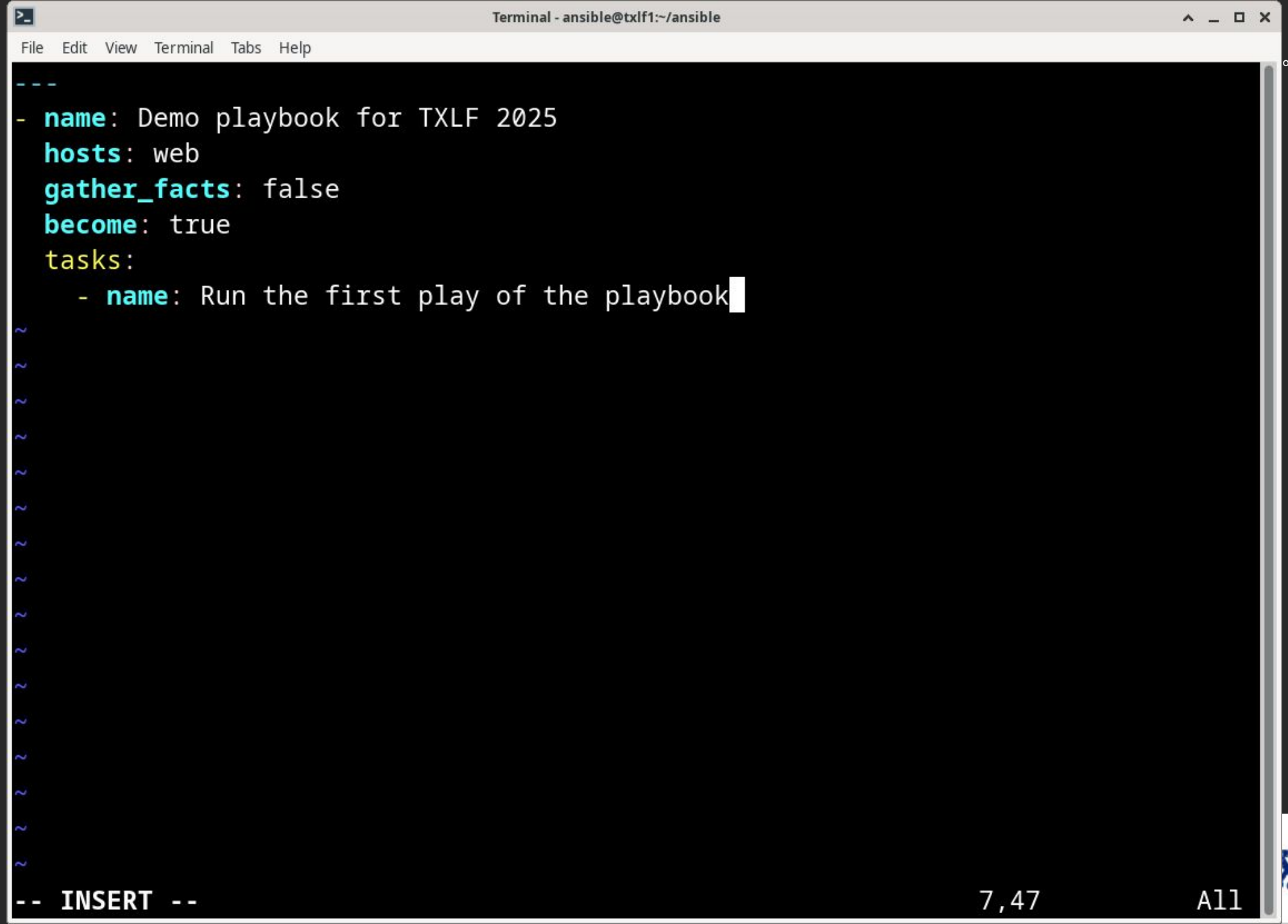


Playbooks

There are a number of fields you can use next. The basics include:

- tasks:
  - This is where we start calling individual plays. These plays make up the playbook.
  - Each play will have its own name

Source:



Playbooks

There are a number of fields you can use next. The basics include:

- module:
  - This is where you define what the play is going to do.
  - In the next section, I'm just going to use the Ansible ping module

Source:

Testing with Ansible ping

This actually tests a few things

Using Ansible ping

Using Ansible ping is similar to using ping from the command line, but it is NOT an ICMP ping. It is an ssh session, so it tests a few things.

- Does the managed system have ssh set up correctly?
- Is the account set up correctly?
- Does the managed system have python3 installed?
- Optionally, does privilege escalation work correctly?

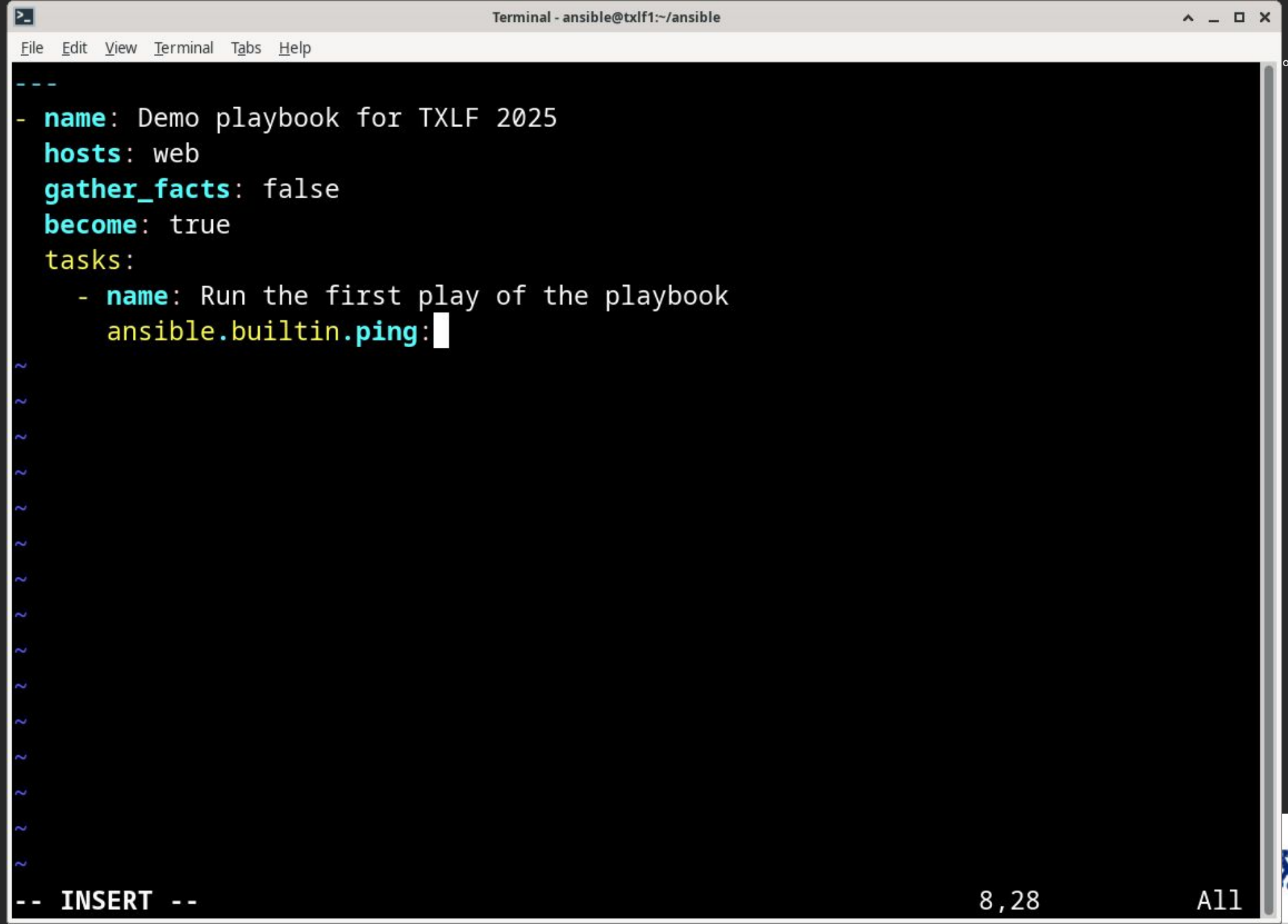
Source:
Insert source data here
Insert source data here

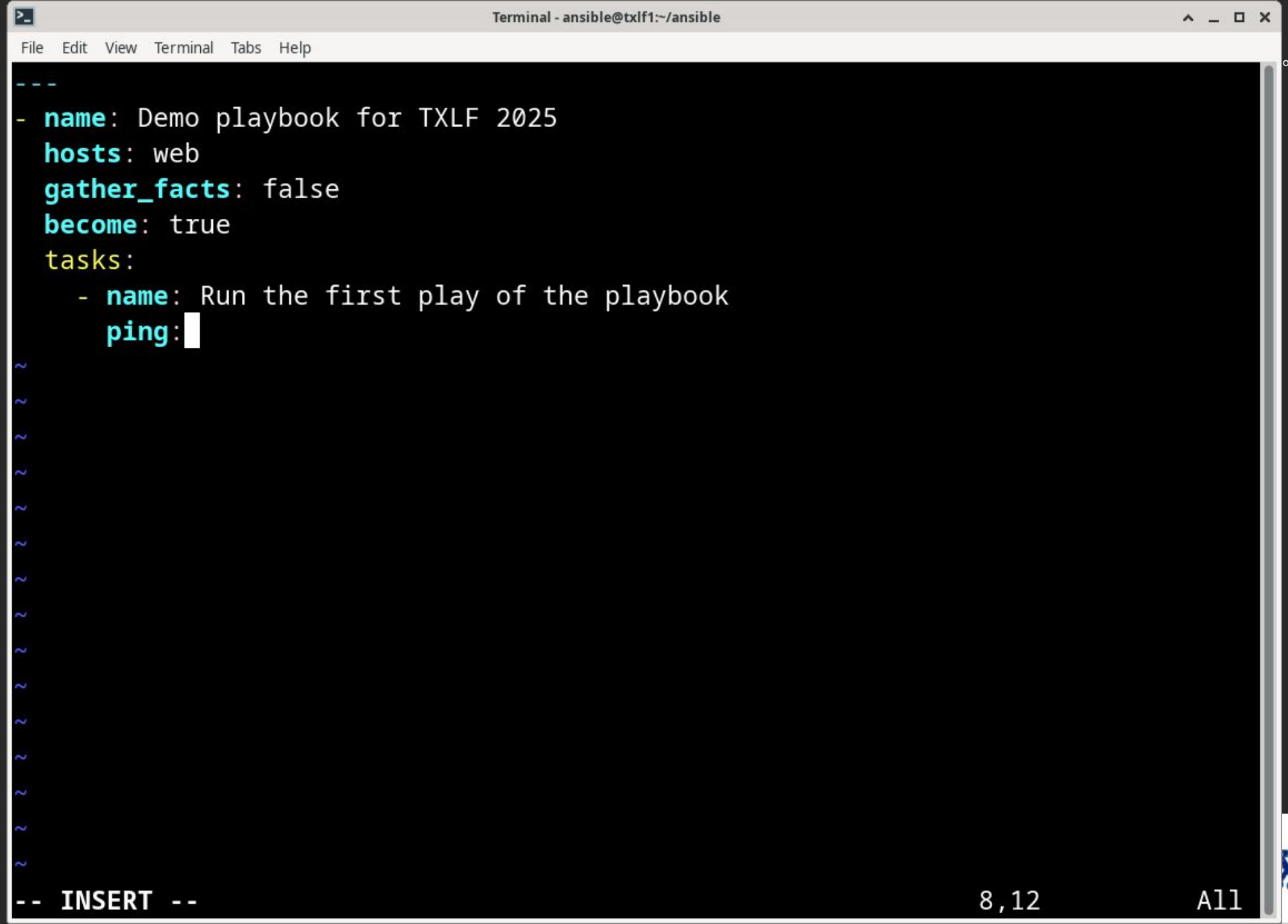
Using Ansible ping

You will define the collection after the `name` section of the play

- You can use the full name of the collection, as in `ansible.builtin.ping`
- You can also shorten the collection to the last string.
  - In this case, `ping`
- When you're first starting out, I recommend you use the full collection name. Repetition helps absorb knowledge.

Source:
Insert source data here
Insert source data here



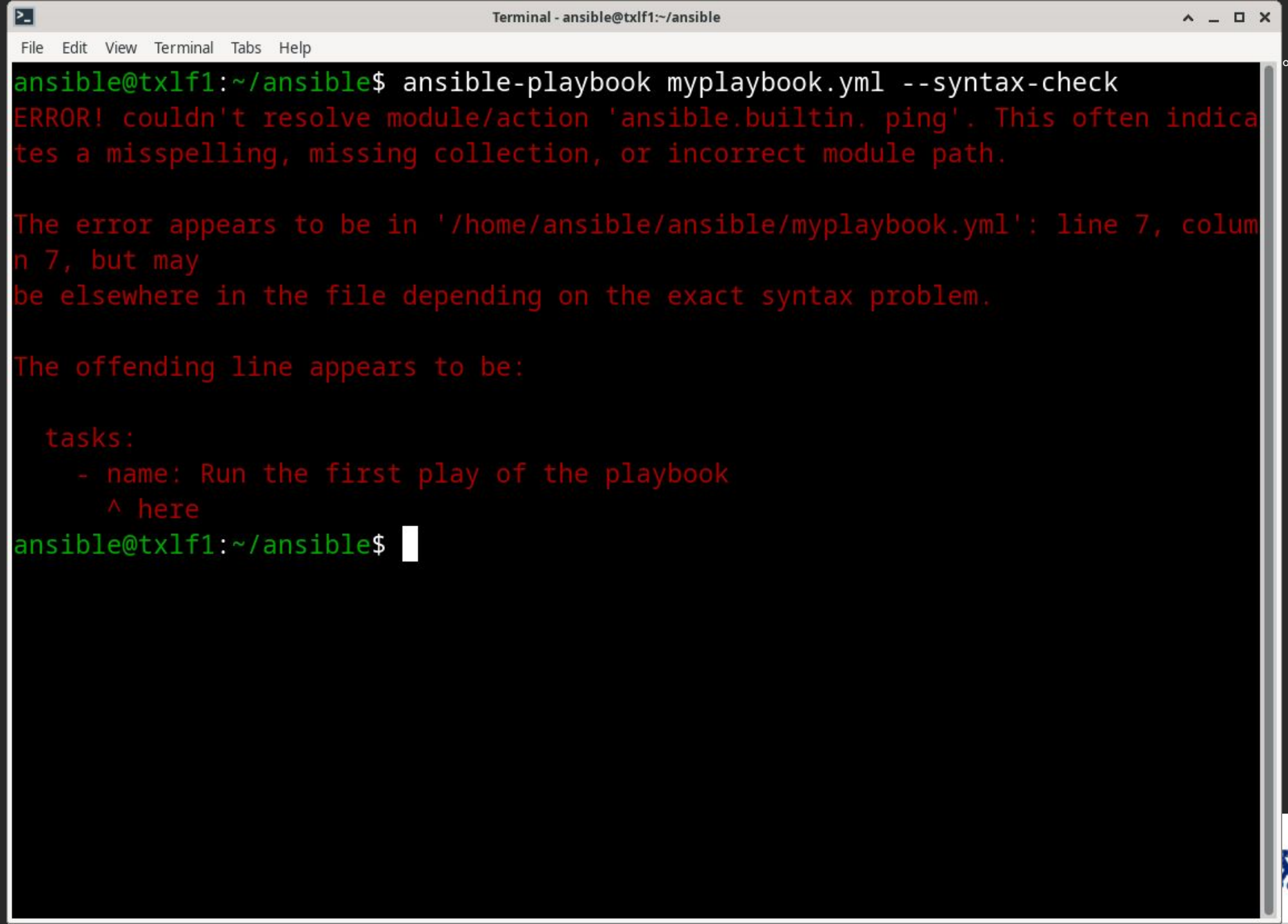


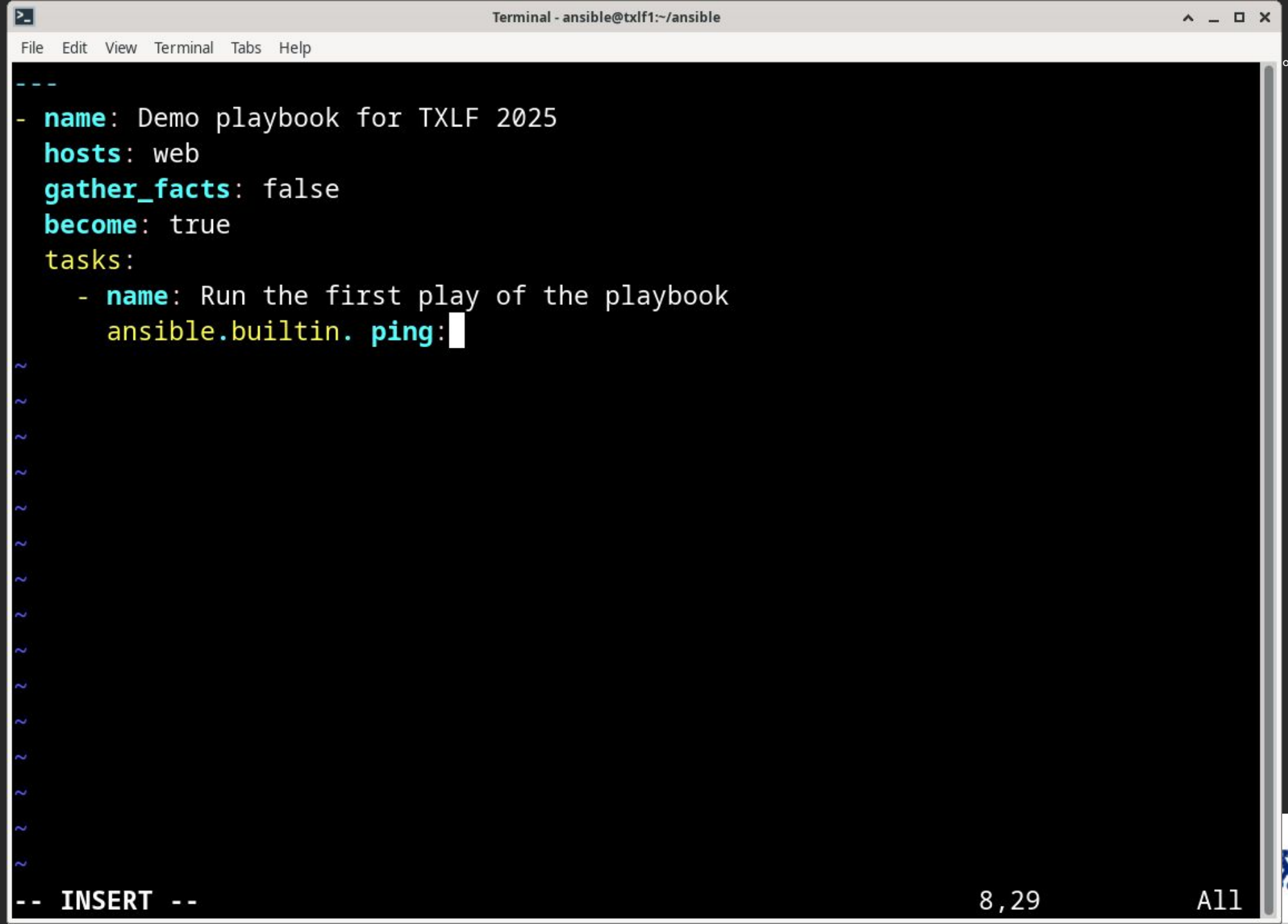
Using Ansible ping

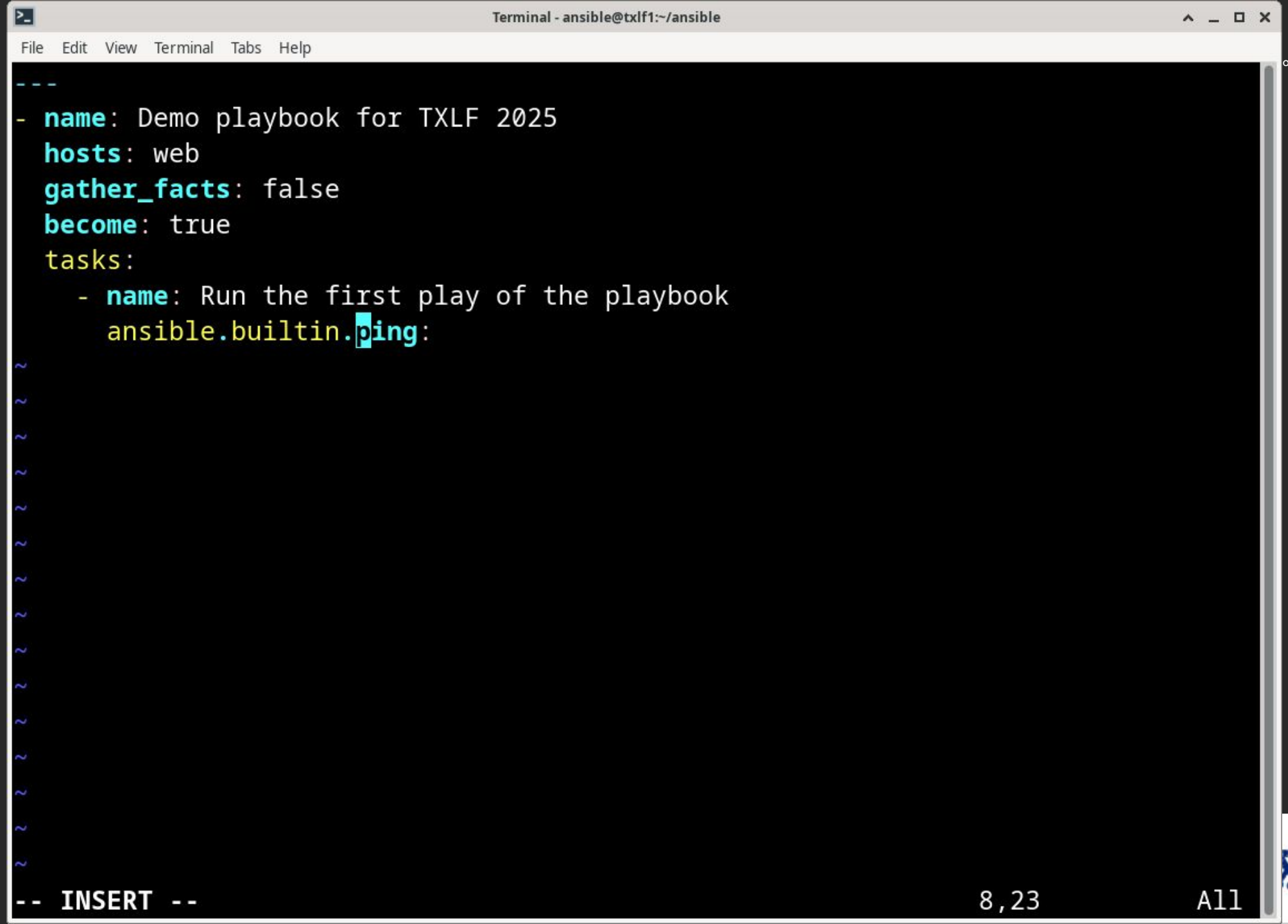
Now you can do a syntax check!

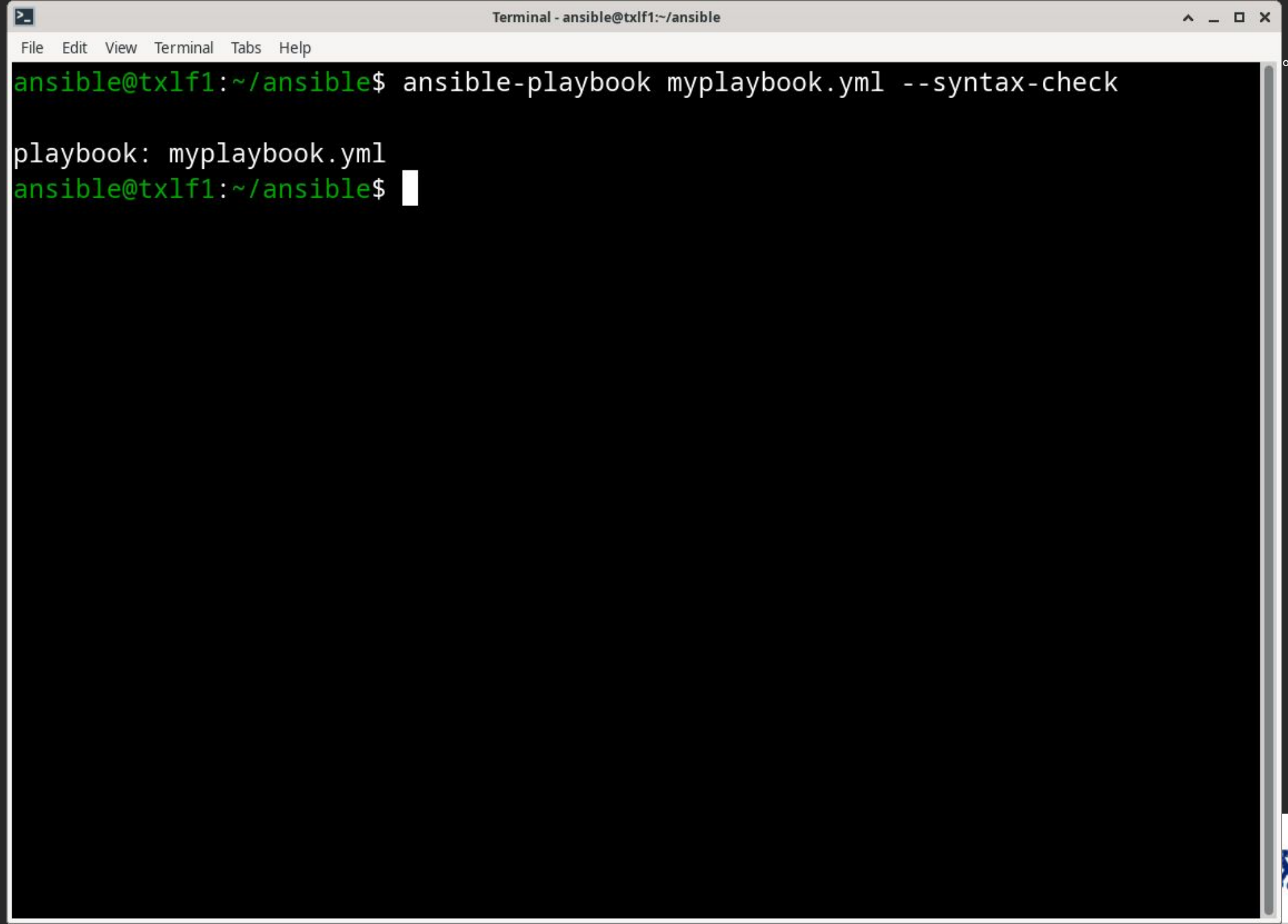
- To launch the playbook, you use
  - `ansible-playbook [playbook]`
- But you can also pass --syntax-check to the command to make sure your playbook is formatted correctly

Source:
Insert source data here
Insert source data here







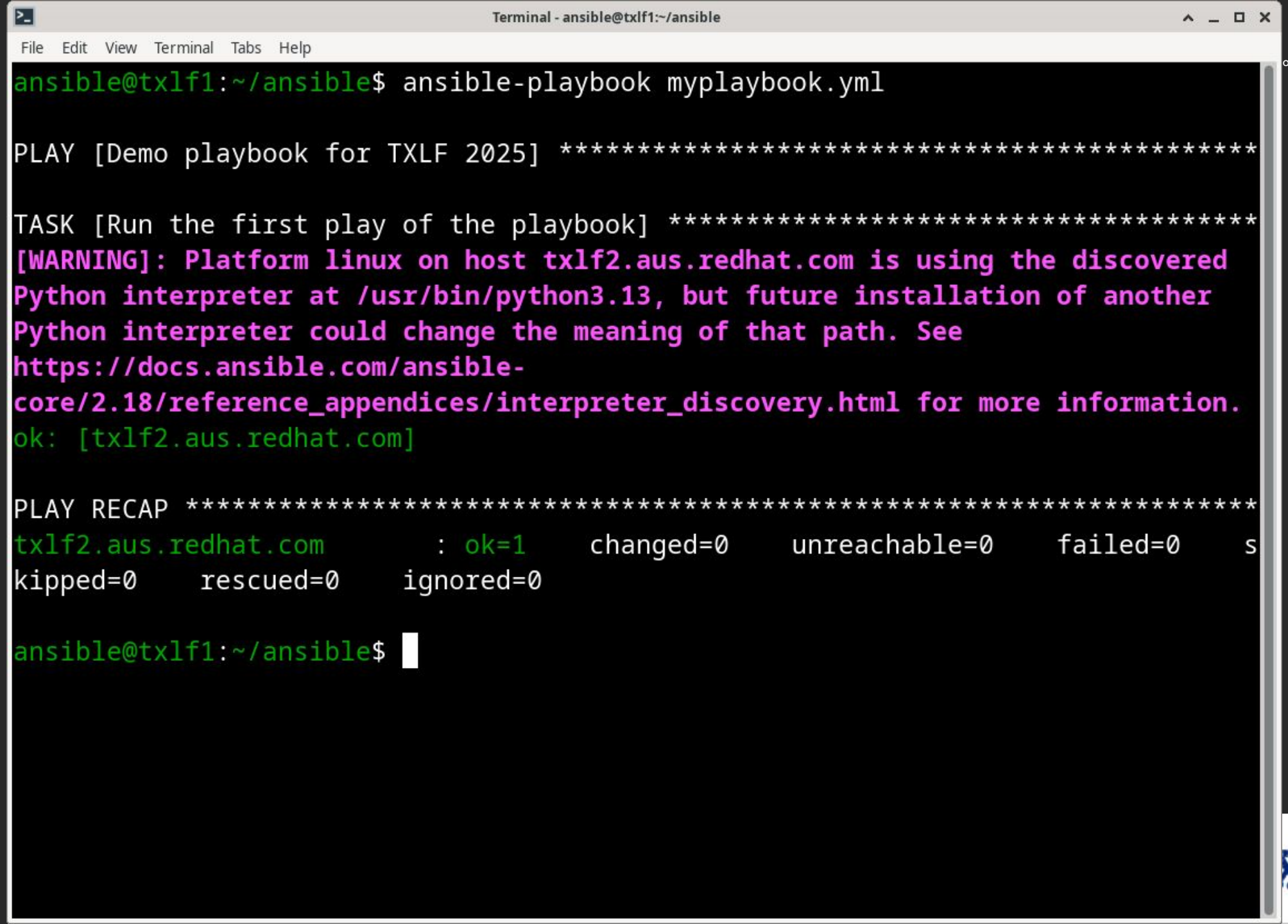


Using Ansible ping

Time to run the playbook!

- To launch the playbook, you use
  - `ansible-playbook [playbook]`
- You can safely ignore the warning about the python version

Source:
Insert source data here
Insert source data here

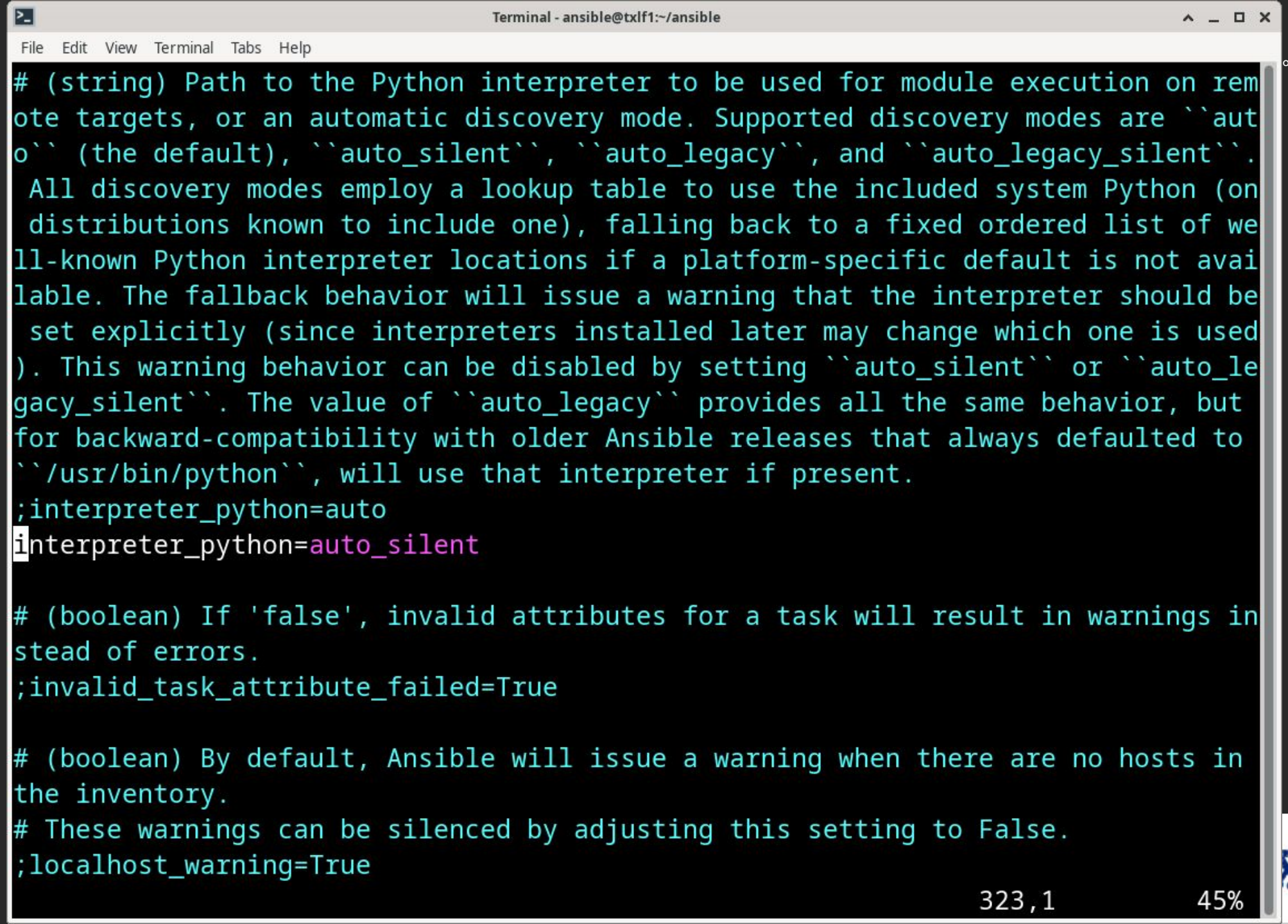


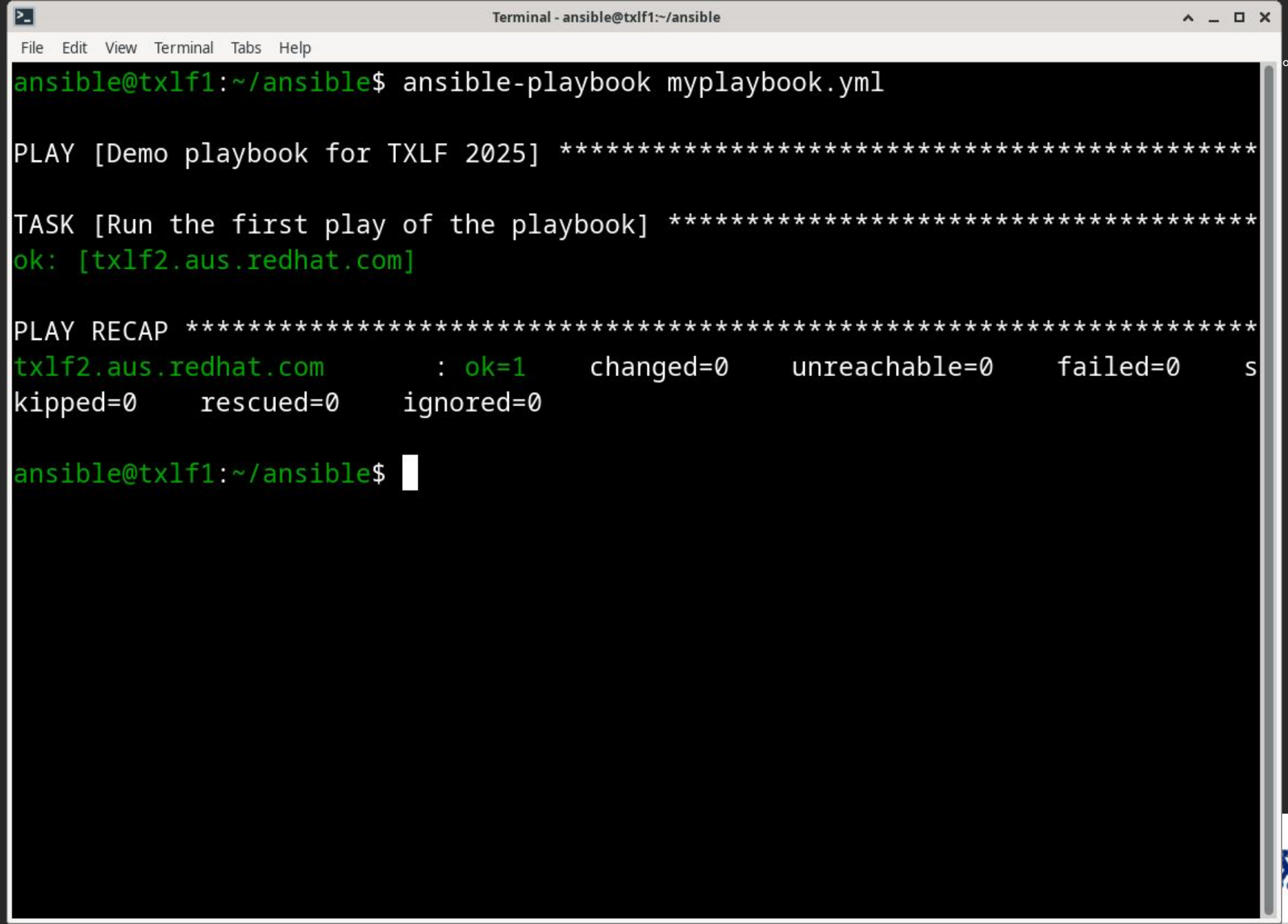
Using Ansible ping

Time to run the playbook!

- To silence the warning, add the following line to your `ansible.cfg`
  - `interpreter_python=auto_silent`

Source:
Insert source data here
Insert source data here





Using Ansible ping

This has tested three important things:

- That your management machine can connect to the managed node via ssh
- That the account is set up with the correct ssh key
- That privilege escalation worked

Source:
Insert source data here
Insert source data here

Background

How You Install And Run A Service Manually

Pop Quiz!

Without automation, how do you make a service like the Apache http server run on a Linux machine?

Source:
Insert source data here
Insert source data here

Pop Quiz!

Without automation, how do you make a service like the Apache http server run on a Linux machine?

- Install the software

Source:
Insert source data here
Insert source data here

Red Hat

TXLF

Pop Quiz!

Without automation, how do you make a service like the Apache http server run on a Linux machine?

- Install the software
- Enable the service

Source:
Insert source data here
Insert source data here

Pop Quiz!

Without automation, how do you make a service like the Apache http server run on a Linux machine?

- Install the software
- Enable the service
- Start the service

Source:
Insert source data here
Insert source data here

Pop Quiz!

Without automation, how do you make a service like the Apache http server run on a Linux machine?

- Install the software
- Enable the service
- Start the service
- Open the firewall ports

Source:
Insert source data here
Insert source data here

Automating These Steps

Let’s dive in!

Automating these steps

We'll go through each of the steps

- Install the software
- Enable the service
- Start the service
- Open the firewall ports

Source:
Insert source data here
Insert source data here

Automating Installation

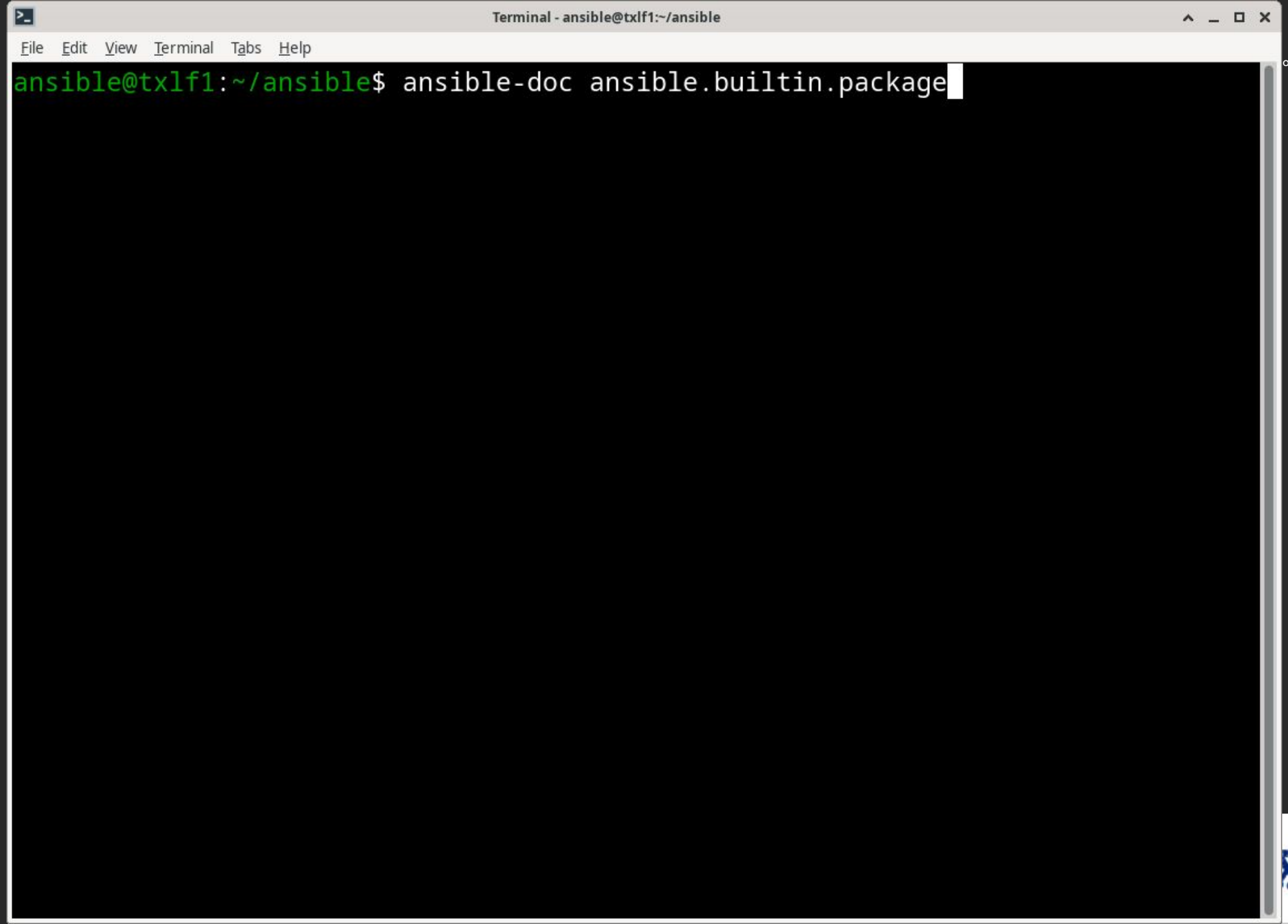
There are a couple of ways

Automating Installation

There are a couple of ways you can automate package installation.

- The generic `ansible.builtin.package` module
  - The benefit of this is it works across Linux distros
  - The downside is that it is somewhat basic, not using any specific capabilities of the underlying package manager like apt or dnf

Source:
Insert source data here
Insert source data here



Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

```
> MODULE ansible.builtin.package (/usr/lib/python3.13/site-packages/ansible/mod>

  This modules manages packages on a target without specifying a
  package manager module (like ansible.builtin.dnf,
  ansible.builtin.apt, ...). It is convenient to use in an
  heterogeneous environment of machines without having to create a
  specific task for each package manager.
  ansible.builtin.package calls behind the module for the
  package manager used by the operating system discovered by the
  module ansible.builtin.setup.  If
  ansible.builtin.setup was not yet run,
  ansible.builtin.package will run it.
  This module acts as a proxy to the underlying package manager
  module. While all arguments will be passed to the underlying module,
  not all modules support the same arguments. This documentation only
  covers the minimum intersection of module arguments that all
  packaging modules support.
  For Windows targets, use the ansible.windows.win_package
  module instead.

  * note: This module has a corresponding action plugin.

OPTIONS (red indicates it is required):
:
```

Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

```
name    Package name, or package specifier with version.
        Syntax varies with package manager. For example `name-1.0'
        or `name=1.0'.
        Package names also vary with package manager; this module
        will not "translate" them per distribution. For example
        `libyaml-dev', `libyaml-devel'.
        To operate on several packages this can accept a comma
        separated string of packages or a list of packages,
        depending on the underlying package manager.

state   Whether to install (`present'), or remove (`absent') a
        package.
        You can use other states like `latest' ONLY if they are
        supported by the underlying package module(s) executed.

use     The required package manager module to use (`dnf', `apt',
        and so on). The default `auto' will use existing facts or
        try to auto-detect it.
        You should only use this field if the automatic selection
        is not working for some reason.
        Since version 2.17 you can use the
        `ansible_package_use' variable to override the
:
```

Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

```
NOTES:
      * While ansible.builtin.package abstracts
        package managers to ease dealing with multiple
        distributions, package name often differs for the same
        software.

REQUIREMENTS:  Whatever is required for the package plugins
        specific for each system.


AUTHOR: Ansible Core Team

EXAMPLES:
- name: Install ntpdate
  ansible.builtin.package:
    name: ntpdate
    state: present

# This uses a variable as this changes per distribution.
- name: Remove the apache package
  ansible.builtin.package:
    name: "{{ apache }}"
    state: absent
:
```

Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

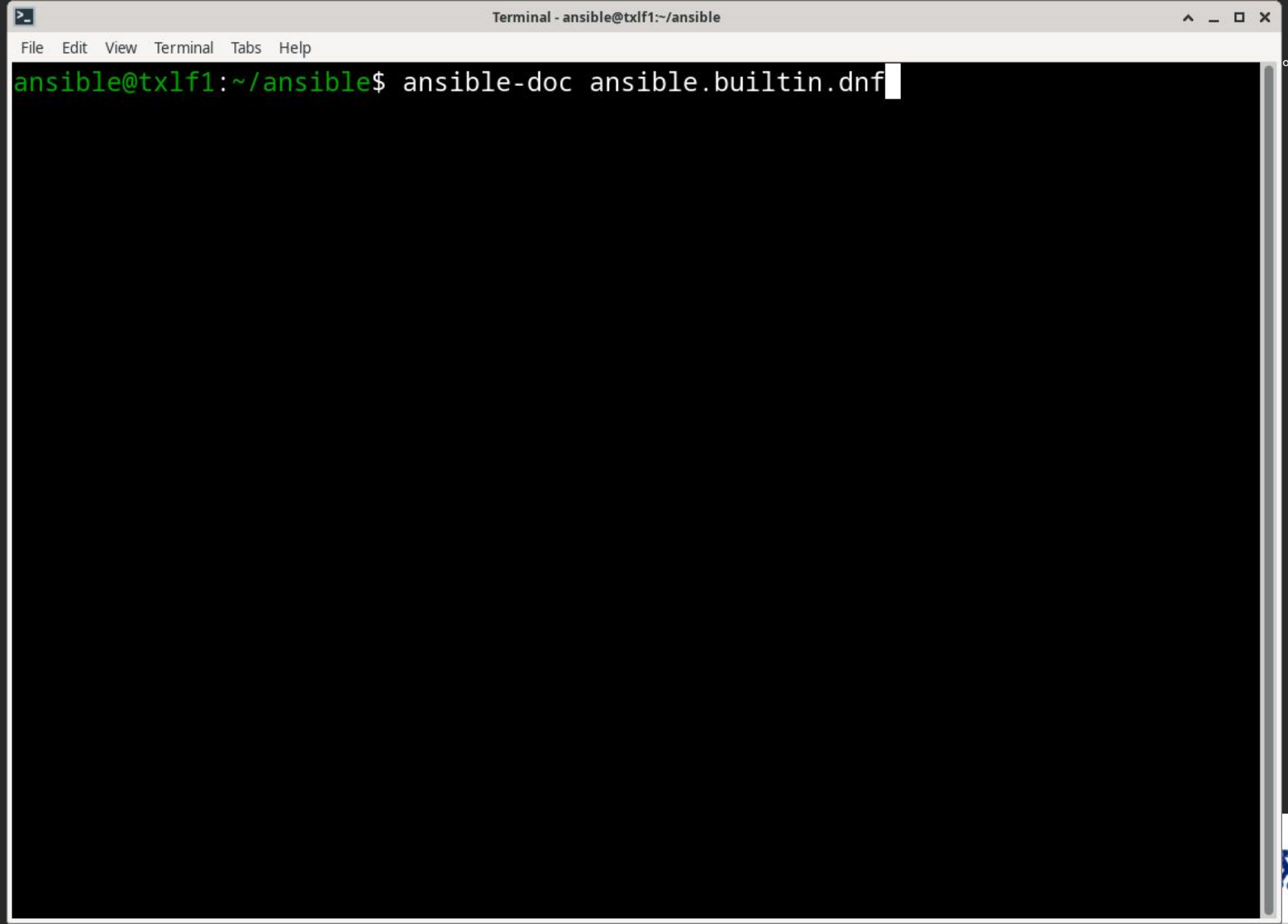
```
AUTHOR: Ansible Core Team

EXAMPLES:
- name: Install ntpdate
  ansible.builtin.package:
    name: ntpdate
    state: present

# This uses a variable as this changes per distribution.
- name: Remove the apache package
  ansible.builtin.package:
    name: "{{ apache }}"
    state: absent

- name: Install the latest version of Apache and MariaDB
  ansible.builtin.package:
    name:
      - httpd
      - mariadb-server
    state: latest

(END)
```

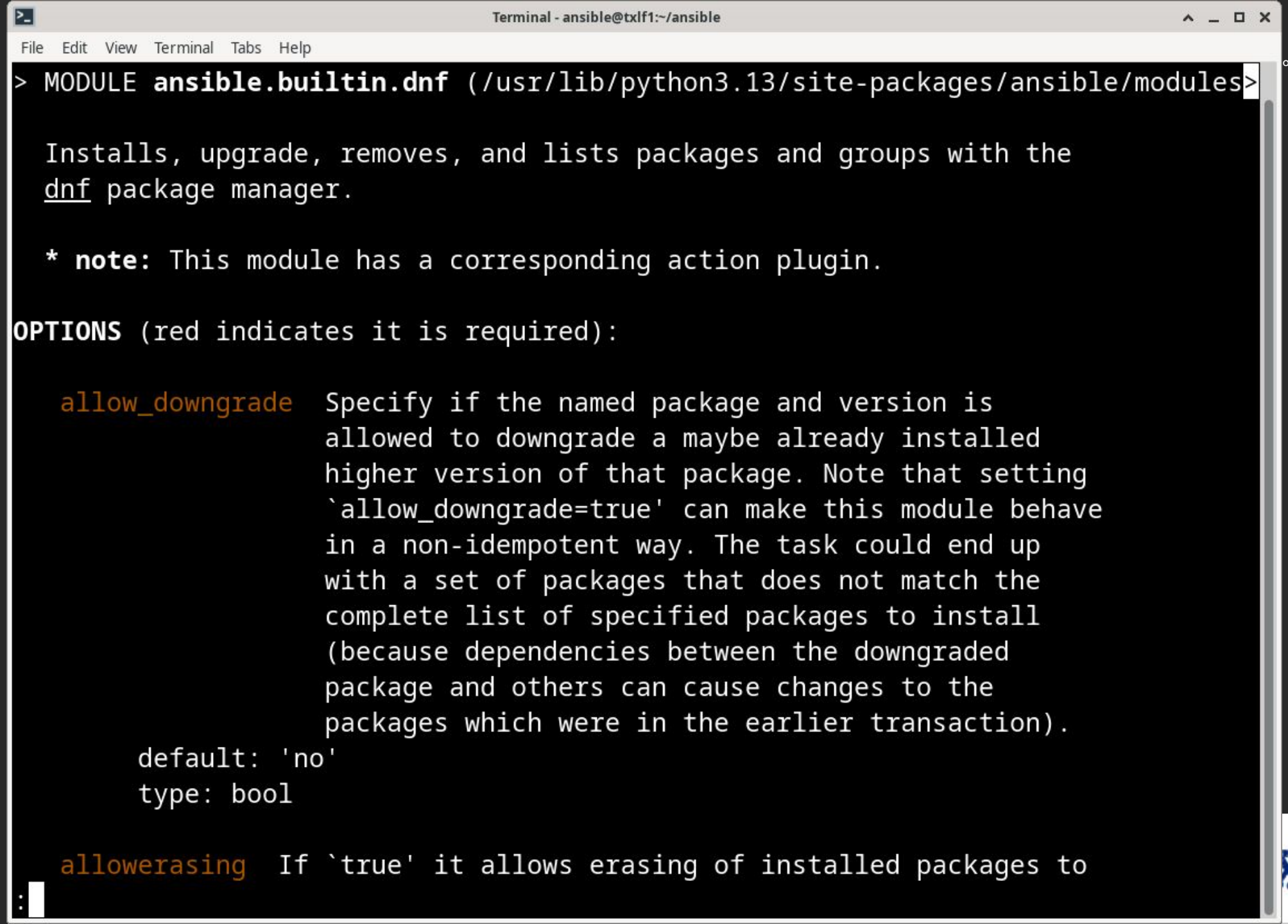
Automating Installation

Or you can use distro-specific

- Red Hat family `ansible.builtin.dnf`
  - The benefit of this is it is specific to distros which use DNF
  - It has a **bunch** of extra options for DNF

Source:
Insert source data here
Insert source data here





Terminal - ansible@txlf1:~/ansible

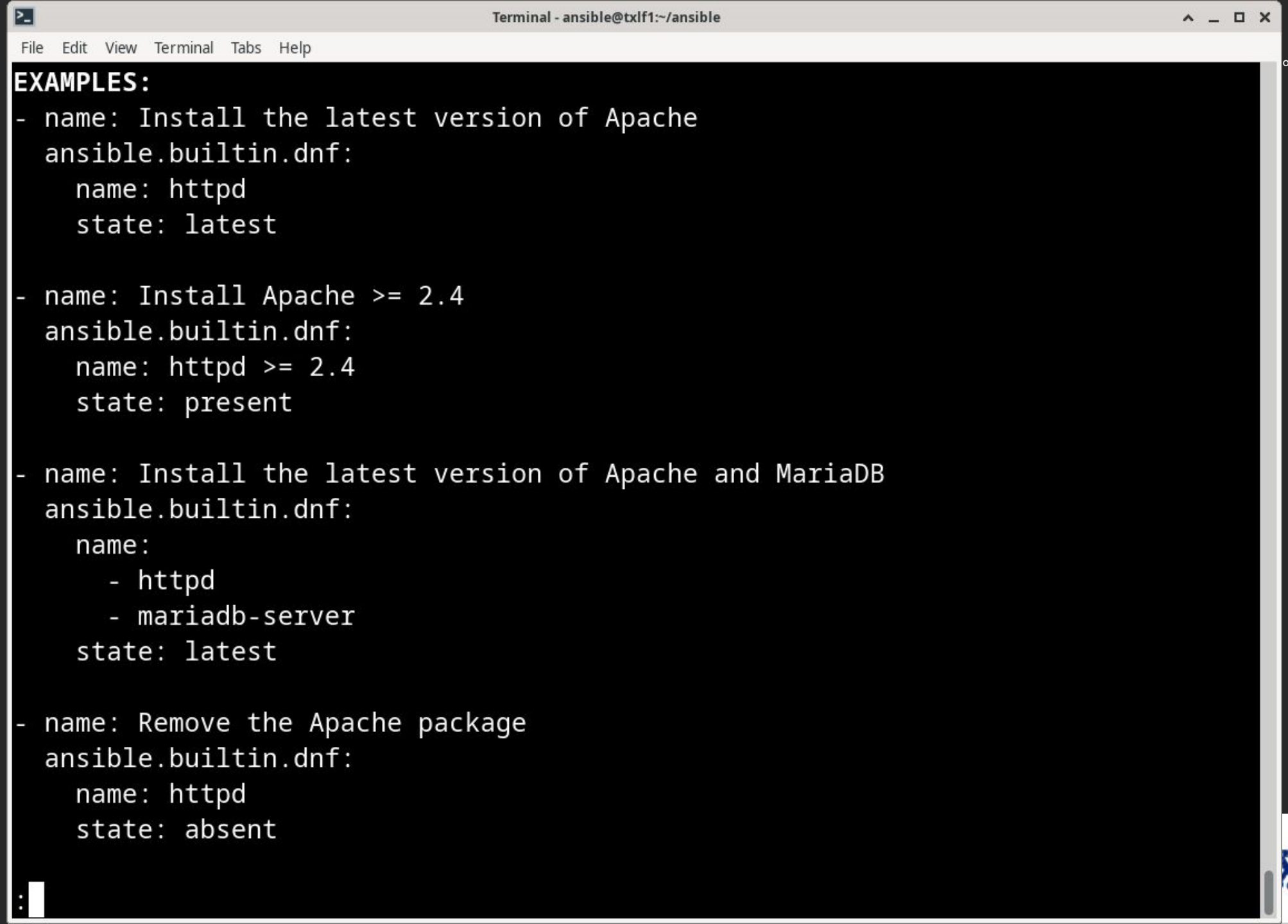
File Edit View Terminal Tabs Help

```
              resolve dependencies.
        default: 'no'
        type: bool

    autoremove  If `true', removes all "leaf" packages from the system
                that were originally installed as dependencies of
                user-installed packages but which are no longer
                required by any such package. Should be used alone or
                when `state=absent'.
        default: 'no'
        type: bool

    best    When set to `true', either use a package with the highest
            version available or fail.
            When set to `false', if the latest version cannot be
            installed go with the lower version.
            Default is set by the operating system distribution.
        default: null
        type: bool

    bugfix  If set to `true', and `state=latest' then only installs
            updates that have been marked bugfix related.
            Note that, similar to `dnf upgrade-minimal',
:
```

Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

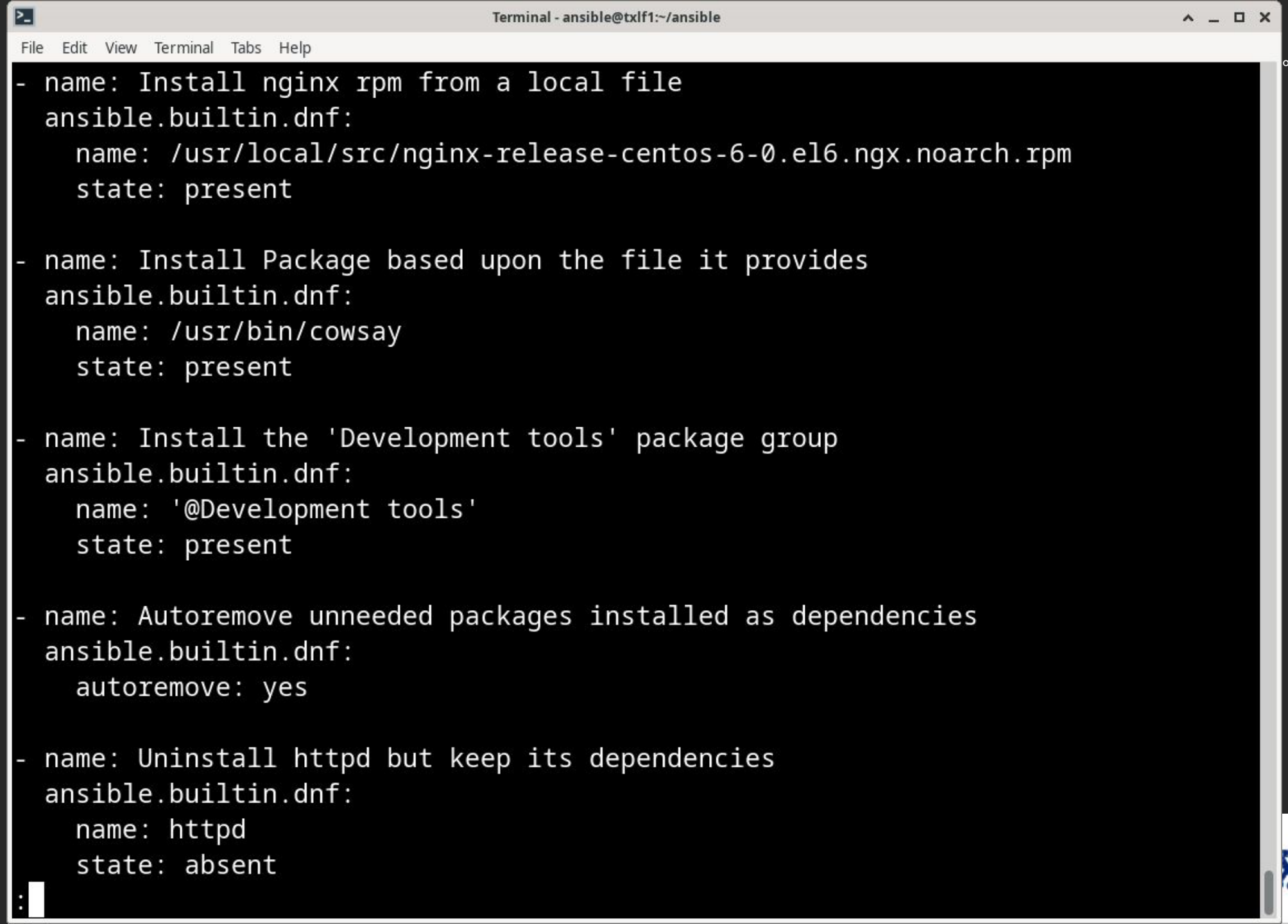
```
          this filter applies to dependencies as well.
      default: 'no'
      type: bool

  cacheonly  Tells dnf to run entirely from system cache; does not
             download or update metadata.
      default: 'no'
      type: bool

  conf_file  The remote dnf configuration file to use for the
             transaction.
      default: null
      type: str

  disable_excludes  Disable the excludes defined in DNF config files.
                    If set to `all', disables all excludes.
                    If set to `main', disable excludes defined in
                    `[main]' in `dnf.conf'.
                    If set to `repoid', disable excludes defined for
                    given repo id.
      default: null
      type: str

:
```



Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

```
- name: Install the latest version of Apache from the testing repo
  ansible.builtin.dnf:
    name: httpd
    enablerepo: testing
    state: present

- name: Upgrade all packages
  ansible.builtin.dnf:
    name: "*"
    state: latest

- name: Update the webserver, depending on which is installed on the system. Do>
  ansible.builtin.dnf:
    name:
      - httpd
      - nginx
    state: latest
    update_only: yes

- name: Install the nginx rpm from a remote repo
  ansible.builtin.dnf:
    name: 'http://nginx.org/packages/centos/6/noarch/RPMS/nginx-release-centos-6>
    state: present
:
```



Terminal - ansible@txlf1:~/ansible

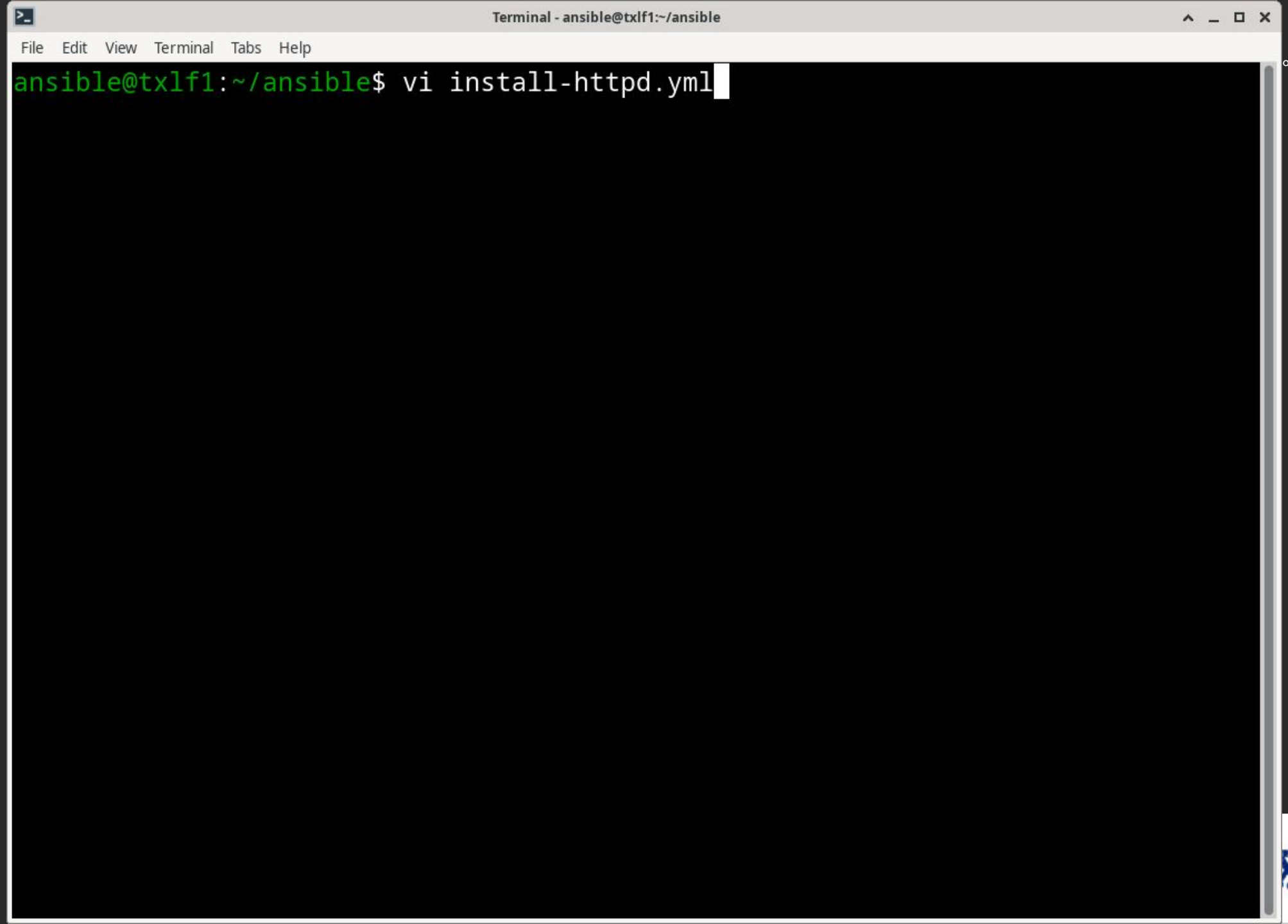
File Edit View Terminal Tabs Help

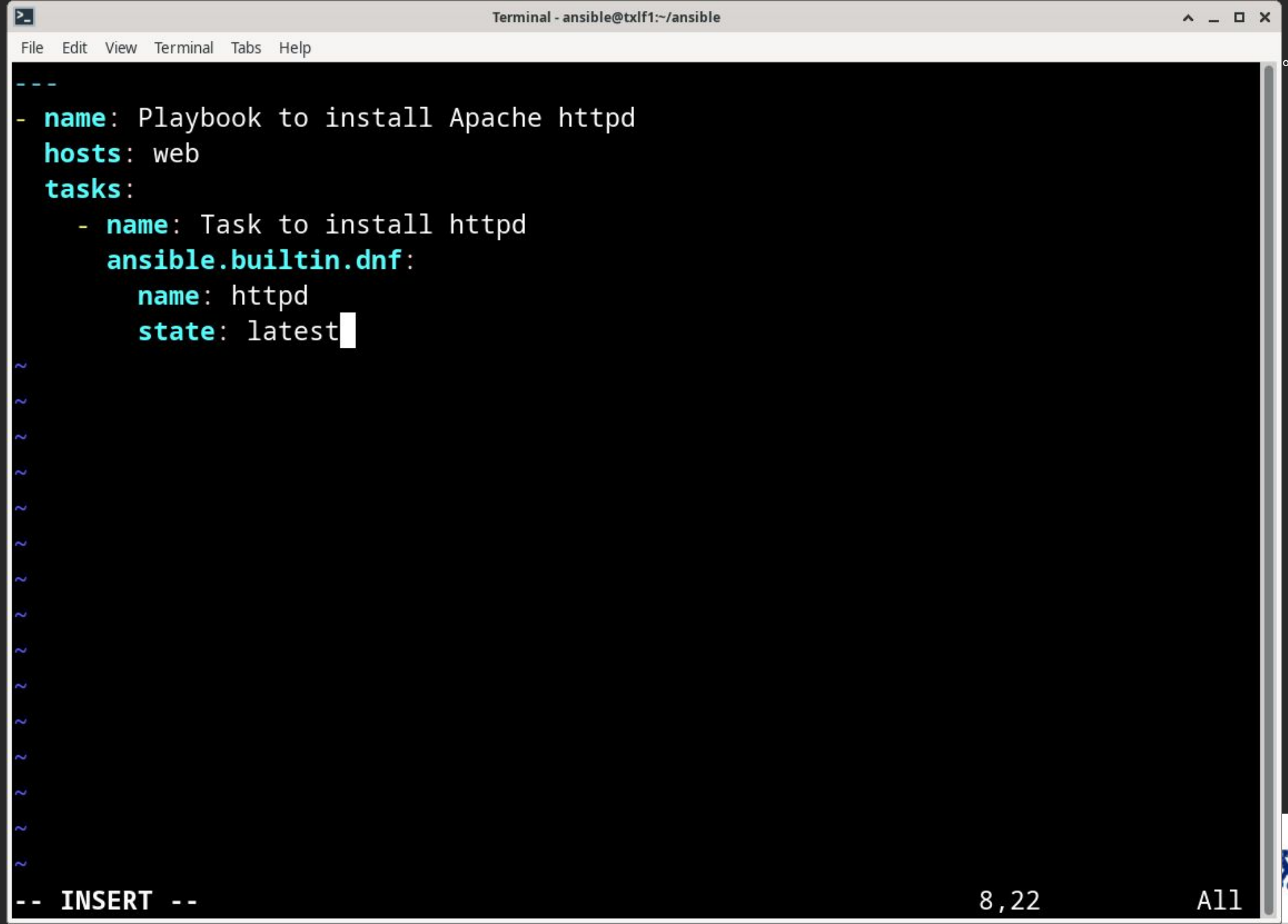
```
    autoremove: yes

- name: Uninstall httpd but keep its dependencies
  ansible.builtin.dnf:
    name: httpd
    state: absent
    autoremove: no

- name: Install a modularity appstream with defined stream and profile
  ansible.builtin.dnf:
    name: '@postgresql:9.6/client'
    state: present

- name: Install a modularity appstream with defined stream
  ansible.builtin.dnf:
    name: '@postgresql:9.6'
    state: present

- name: Install a modularity appstream with defined profile
  ansible.builtin.dnf:
    name: '@postgresql/client'
    state: present

(END)
```

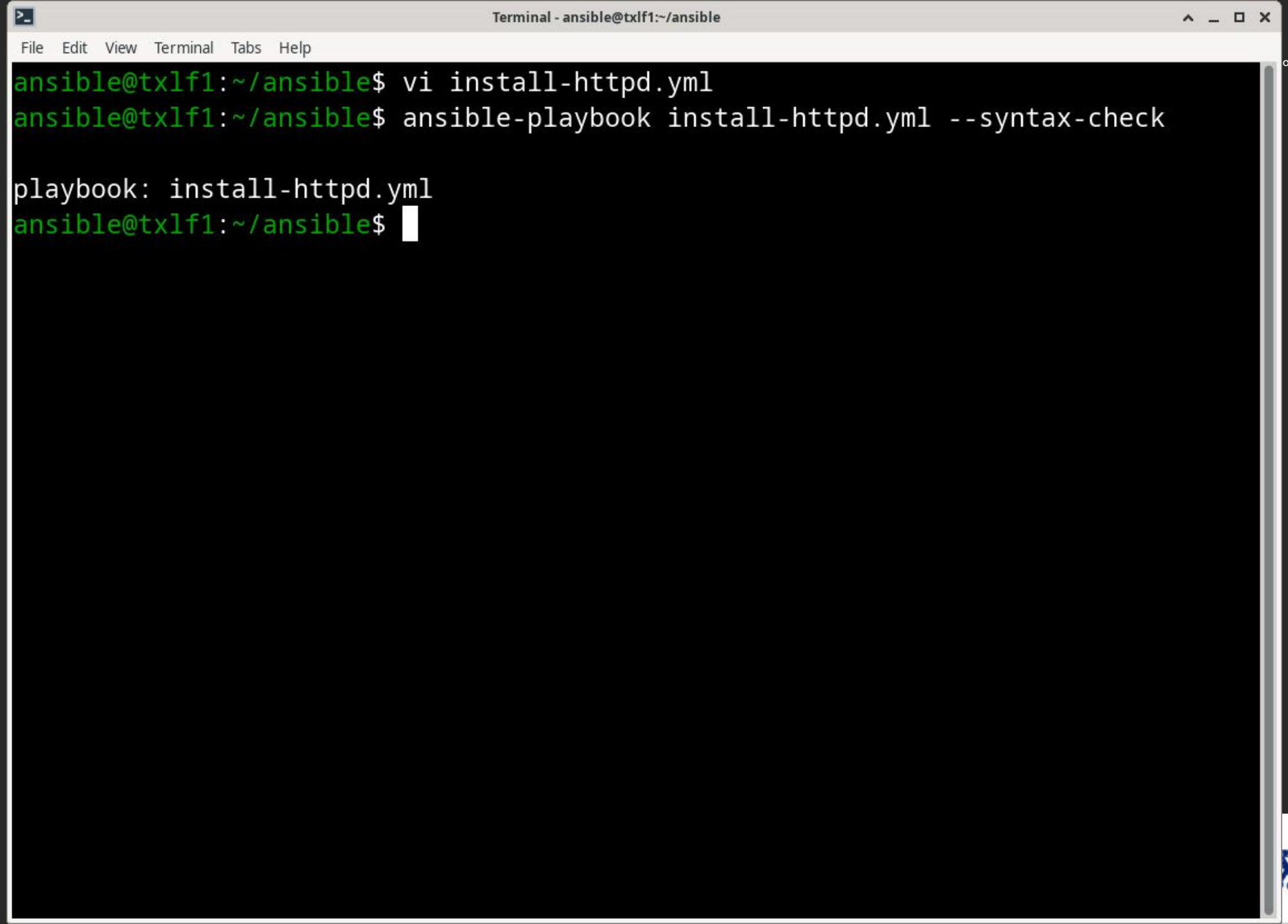
Automating Installation

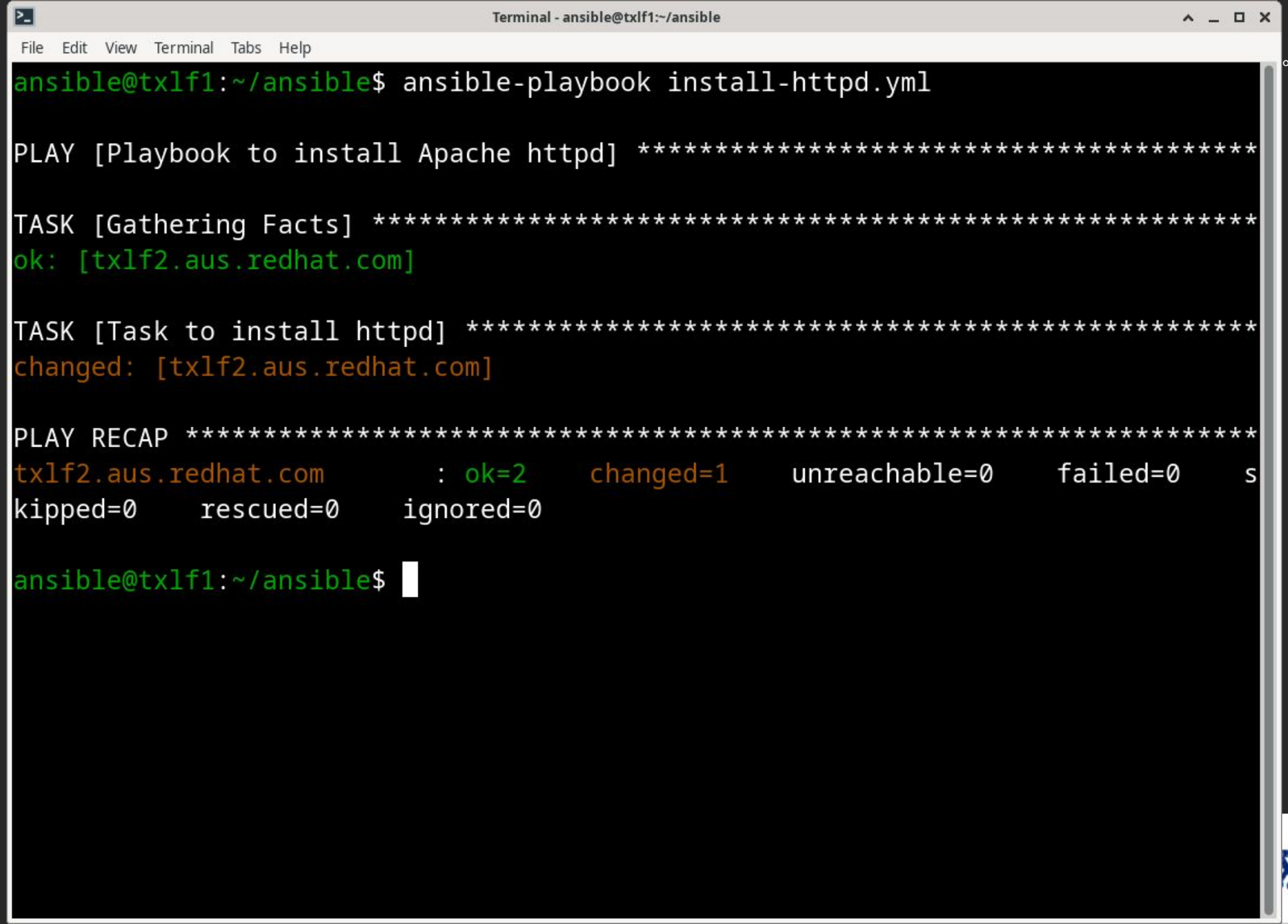
So let’s install httpd and mod_ssl on a Red Hat distro

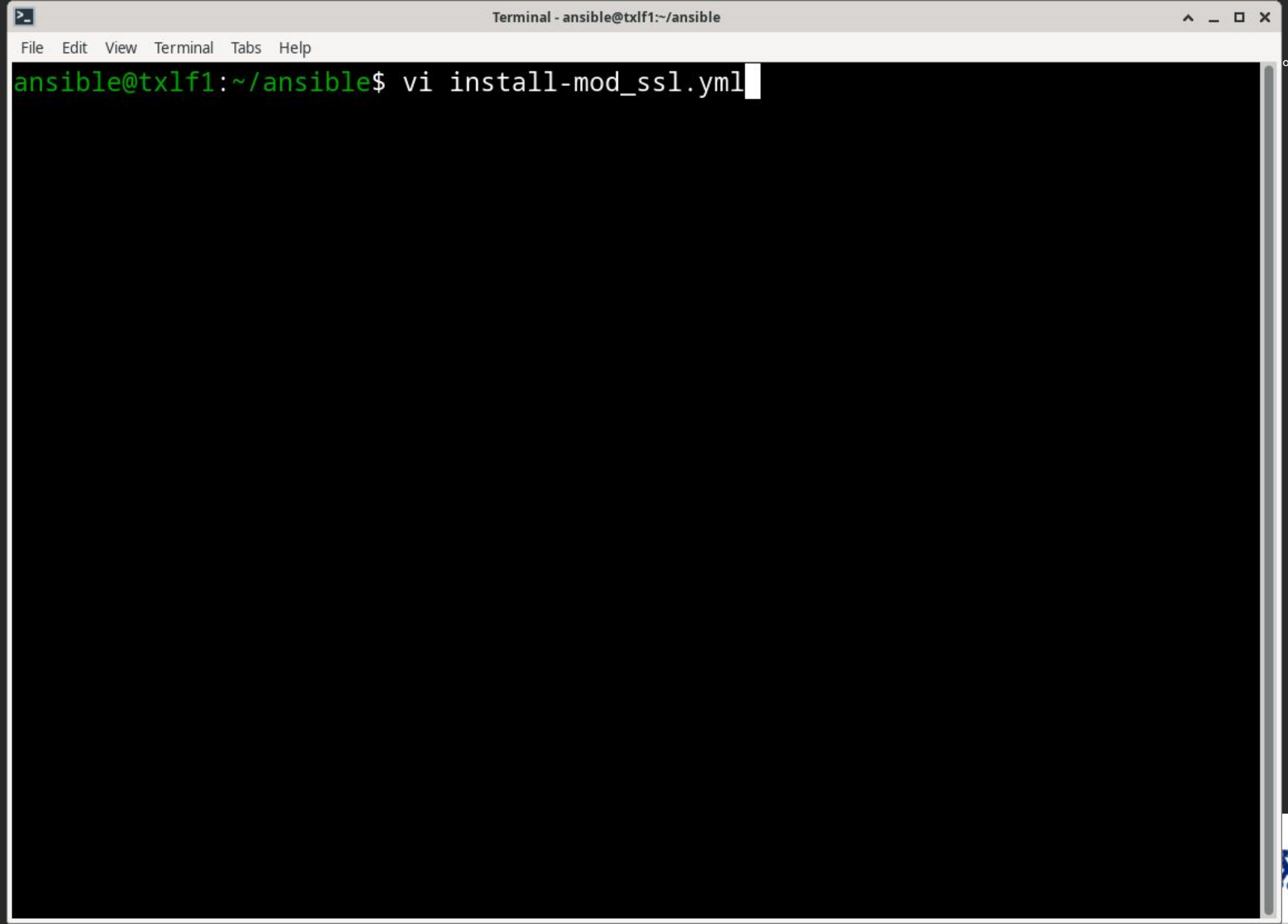
Source:
Insert source data here
Insert source data here

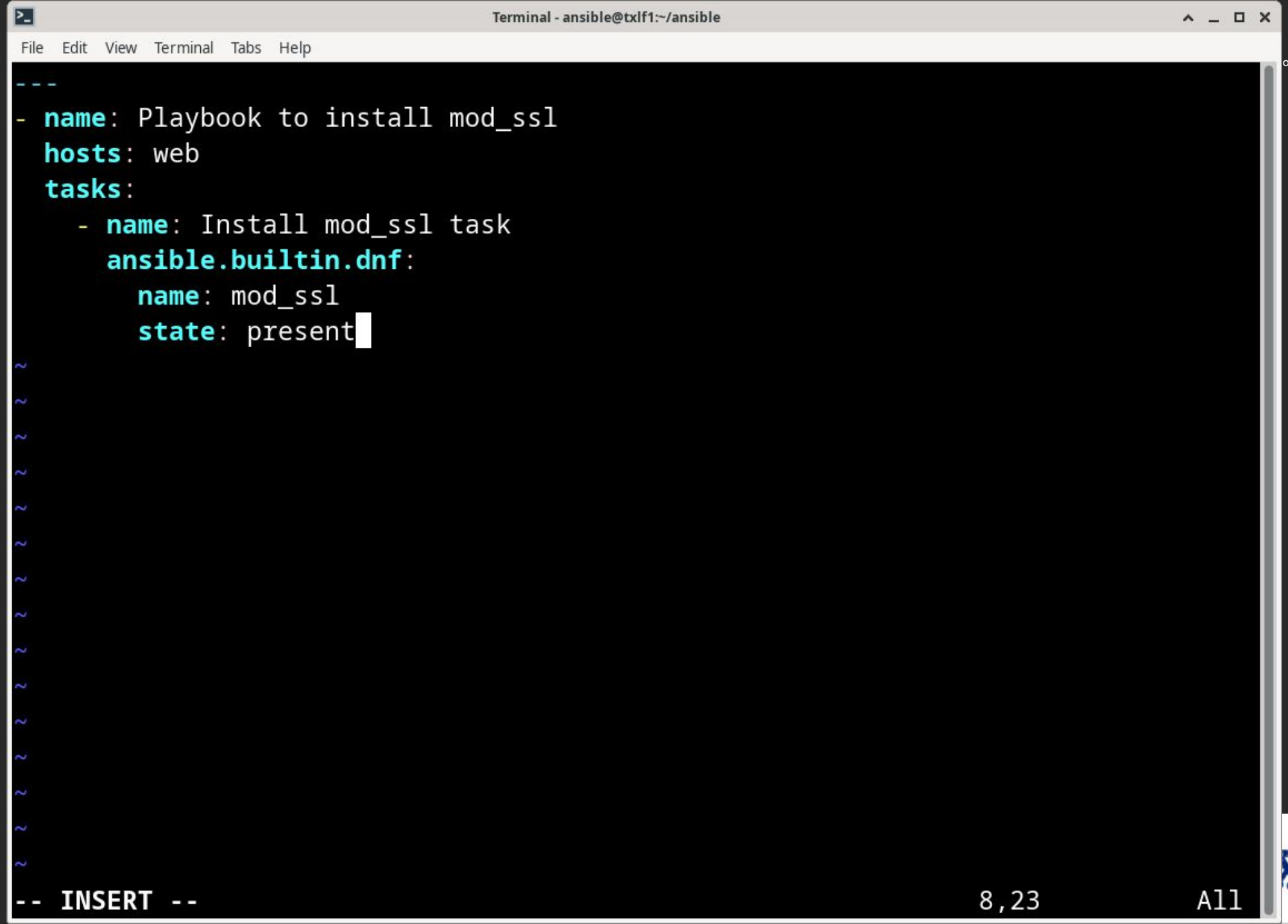












Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

```
ansible@txlf1:~/ansible$ vi install-mod_ssl.yml
ansible@txlf1:~/ansible$ ansible-playbook install-mod_ssl.yml

PLAY [Playbook to install mod_ssl] *********************************************

TASK [Gathering Facts] *********************************************************
ok: [txlf2.aus.redhat.com]

TASK [Install mod_ssl task] ****************************************************
changed: [txlf2.aus.redhat.com]

PLAY RECAP *********************************************************************
txlf2.aus.redhat.com       : ok=2    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0

ansible@txlf1:~/ansible$
```

Automating Service Management

Again, there are a couple of ways

Enabling the Service

Remember we said you have to install the package, enable the service, start the service, and open the firewall ports? Our next step is to enable the service!

- Remember, **you know how to do this.** All we're learning is how to automate what you already know.

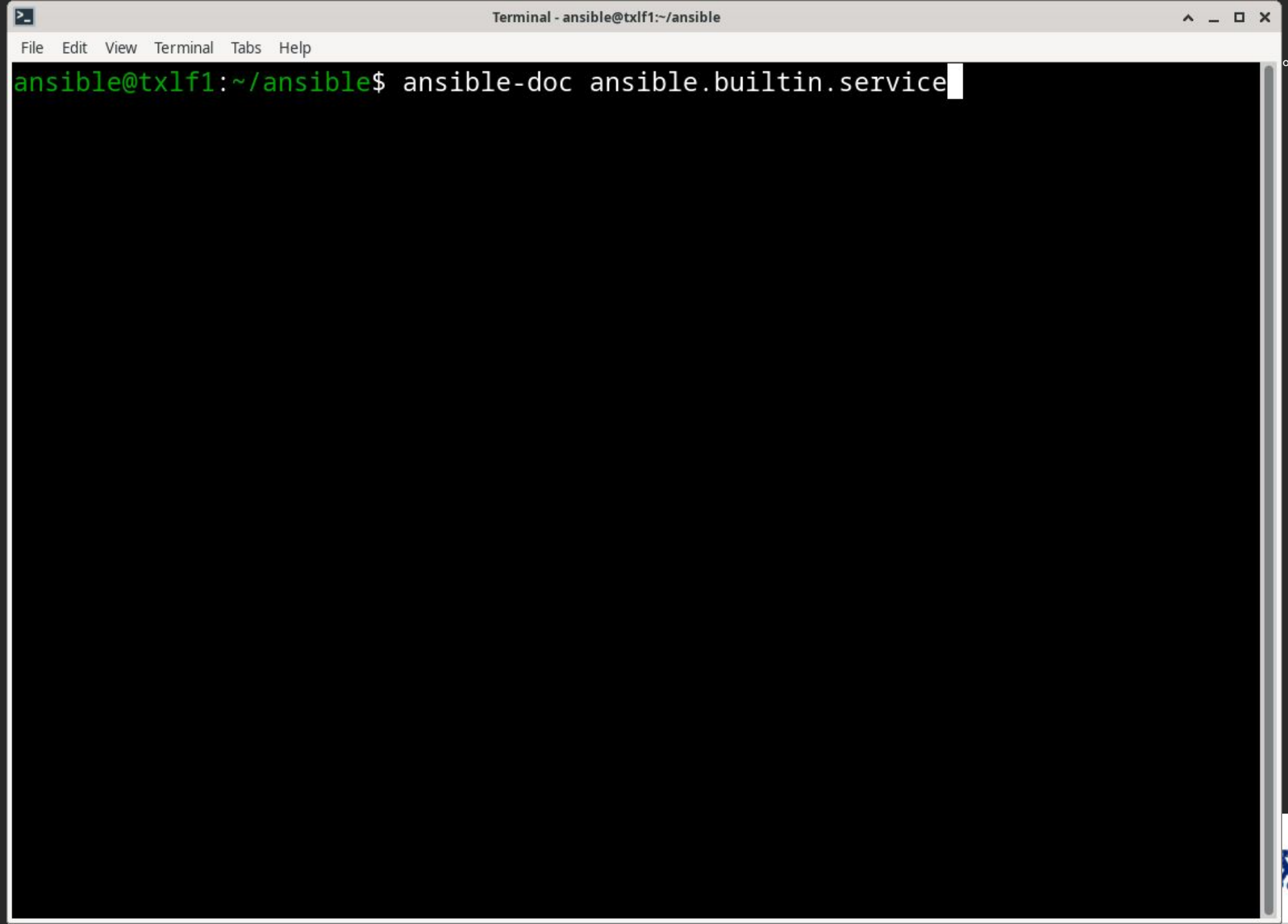
Source:
Insert source data here
Insert source data here

Enabling the Service

Just like there is a “generic” package module, there is a “generic” ansible service module, `ansible.builtin.service`.

- `ansible.builtin.service` is portable across multiple distros, whether they use systemd or not. It does not have a lot of advanced features, since it has to abstract service management from the OS
- `ansible.builtin.systemd_service` (formerly `ansible.builtin.systemd`) is specifically designed for distros which use systemd. It offers advanced features.

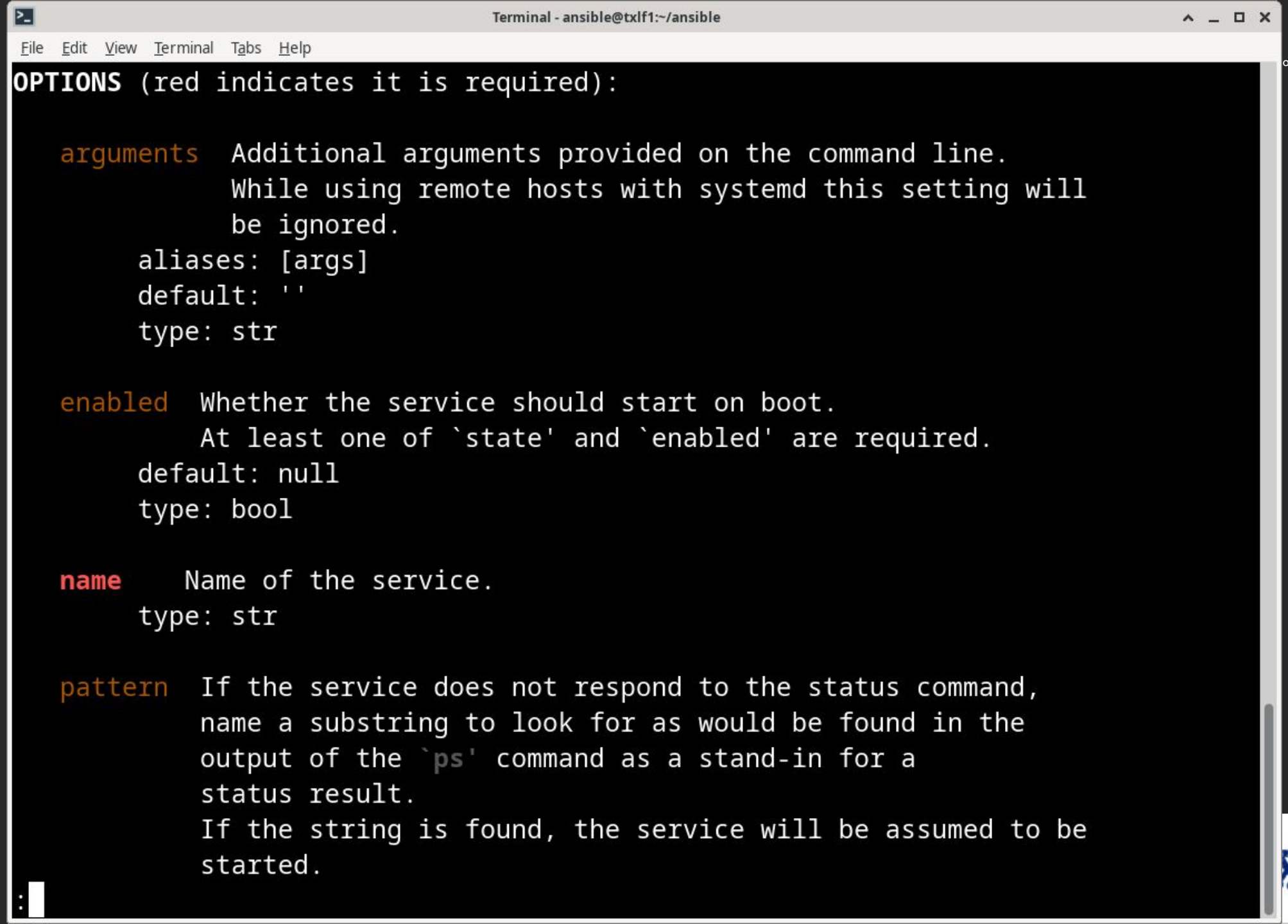
Source:
Insert source data here
Insert source data here

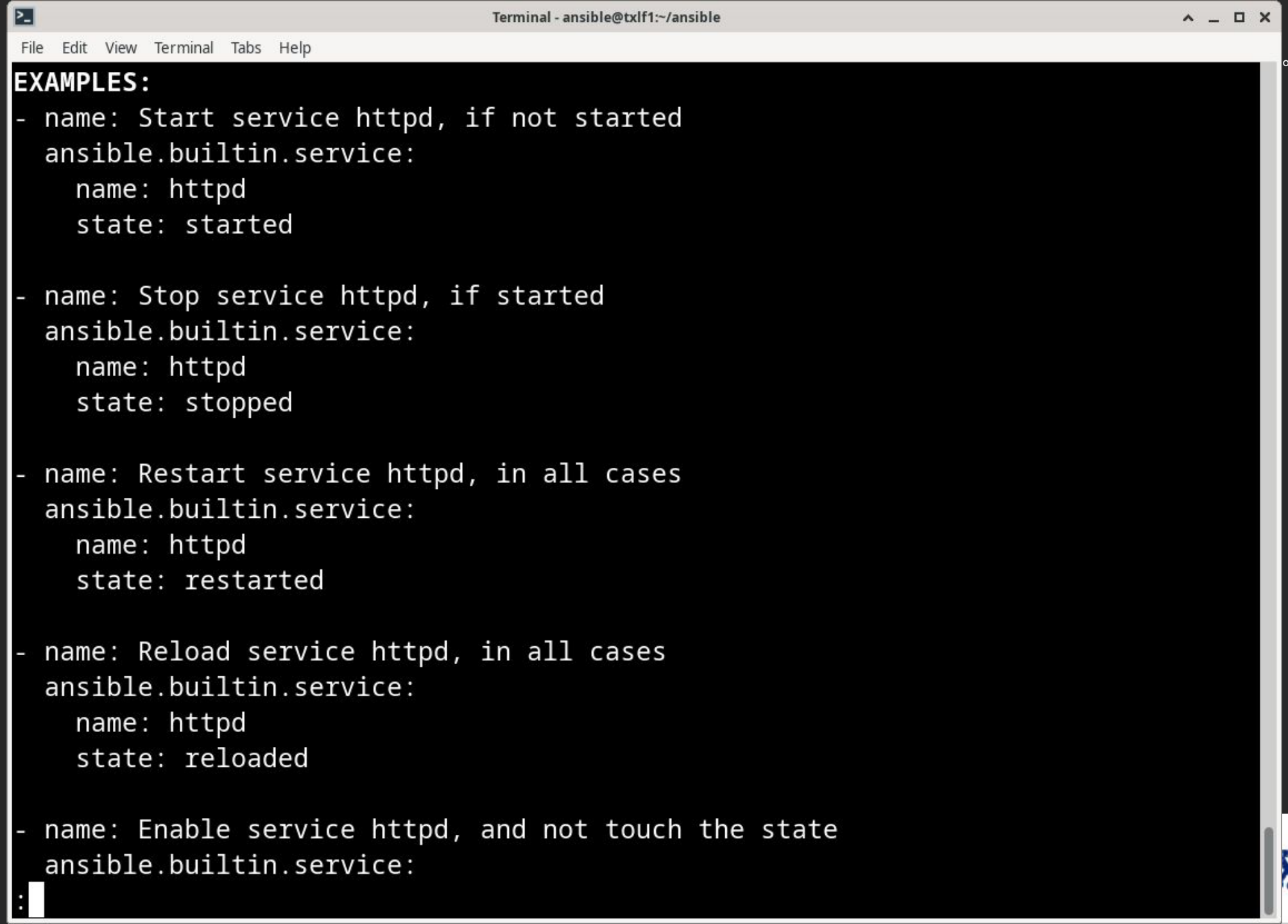


Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

```
> MODULE ansible.builtin.service (/usr/lib/python3.13/site-packages/ansible/mod>

  Controls services on remote hosts. Supported init systems include
  BSD init, OpenRC, SysV, Solaris SMF, systemd, upstart.
  This module acts as a proxy to the underlying service manager
  module. While all arguments will be passed to the underlying module,
  not all modules support the same arguments. This documentation only
  covers the minimum intersection of module arguments that all service
  manager modules support.
  This module is a proxy for multiple more specific service manager
  modules (such as ansible.builtin.systemd and
  ansible.builtin.sysvinit). This allows management of a
  heterogeneous environment of machines without creating a specific
  task for each service manager. The module to be executed is
  determined by the `use' option, which defaults to the service
  manager discovered by ansible.builtin.setup.  If
  ansible.builtin.setup was not yet run, this module may
  run it.
  For Windows targets, use the ansible.windows.win_service
  module instead.

  * note: This module has a corresponding action plugin.

:
```

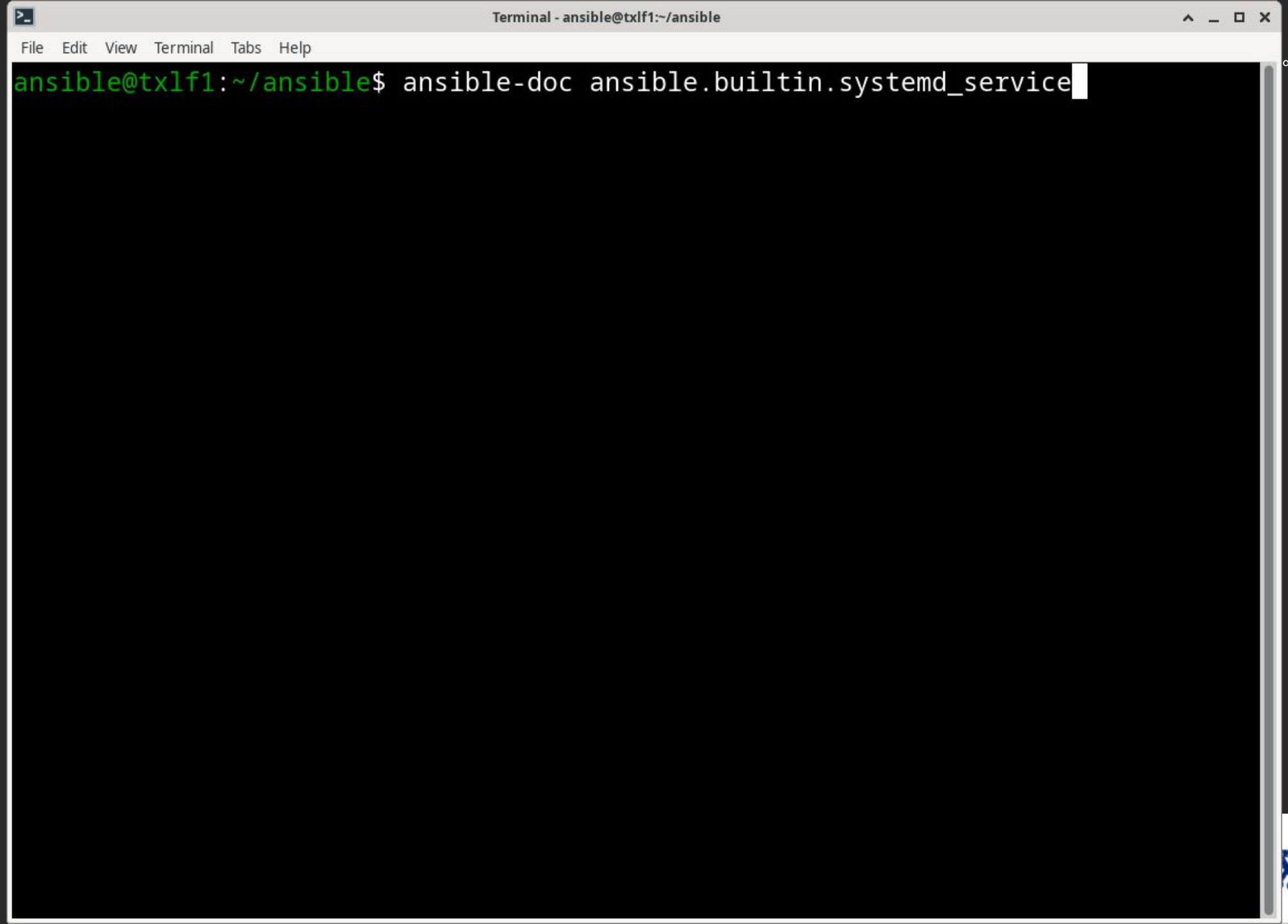




Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

```
- name: Reload service httpd, in all cases
  ansible.builtin.service:
    name: httpd
    state: reloaded

- name: Enable service httpd, and not touch the state
  ansible.builtin.service:
    name: httpd
    enabled: yes

- name: Start service foo, based on running process /usr/bin/foo
  ansible.builtin.service:
    name: foo
    pattern: /usr/bin/foo
    state: started

- name: Restart network service for interface eth0
  ansible.builtin.service:
    name: network
    state: restarted
    args: eth0

(END)
```



Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

```
> MODULE ansible.builtin.systemd_service (/usr/lib/python3.13/site-packages/ans>

  Controls systemd units (services, timers, and so on) on remote
  hosts.
  ansible.builtin.systemd is renamed to
  ansible.builtin.systemd_service to better reflect the
  scope of the module. ansible.builtin.systemd is kept as
  an alias for backward compatibility.

OPTIONS (red indicates it is required):

   daemon_reexec  Run daemon_reexec command before doing any other
                  operations, the systemd manager will serialize the
                  manager state.
        aliases: [daemon-reexec]
        default: false
        type: bool

   daemon_reload  Run `daemon-reload' before doing any
                  other operations, to make sure systemd has read any
                  changes.
                  When set to `true', runs `daemon-reload'
                  even if the module does not start or stop anything.
:
```

Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

```
        aliases: [daemon-reload]
        default: false
        type: bool

    enabled  Whether the unit should start on boot. At least one of
             `state' and `enabled' are required.
             If set, requires `name'.
        default: null
        type: bool

    force    Whether to override existing symlinks.
        default: null
        type: bool

    masked  Whether the unit should be masked or not. A masked unit is
            impossible to start.
            If set, requires `name'.
        default: null
        type: bool

    name    Name of the unit. This parameter takes the name of exactly
            one unit to work with.
            When no extension is given, it is implied to a
:
```

Terminal - ansible@txlf1:~/ansible

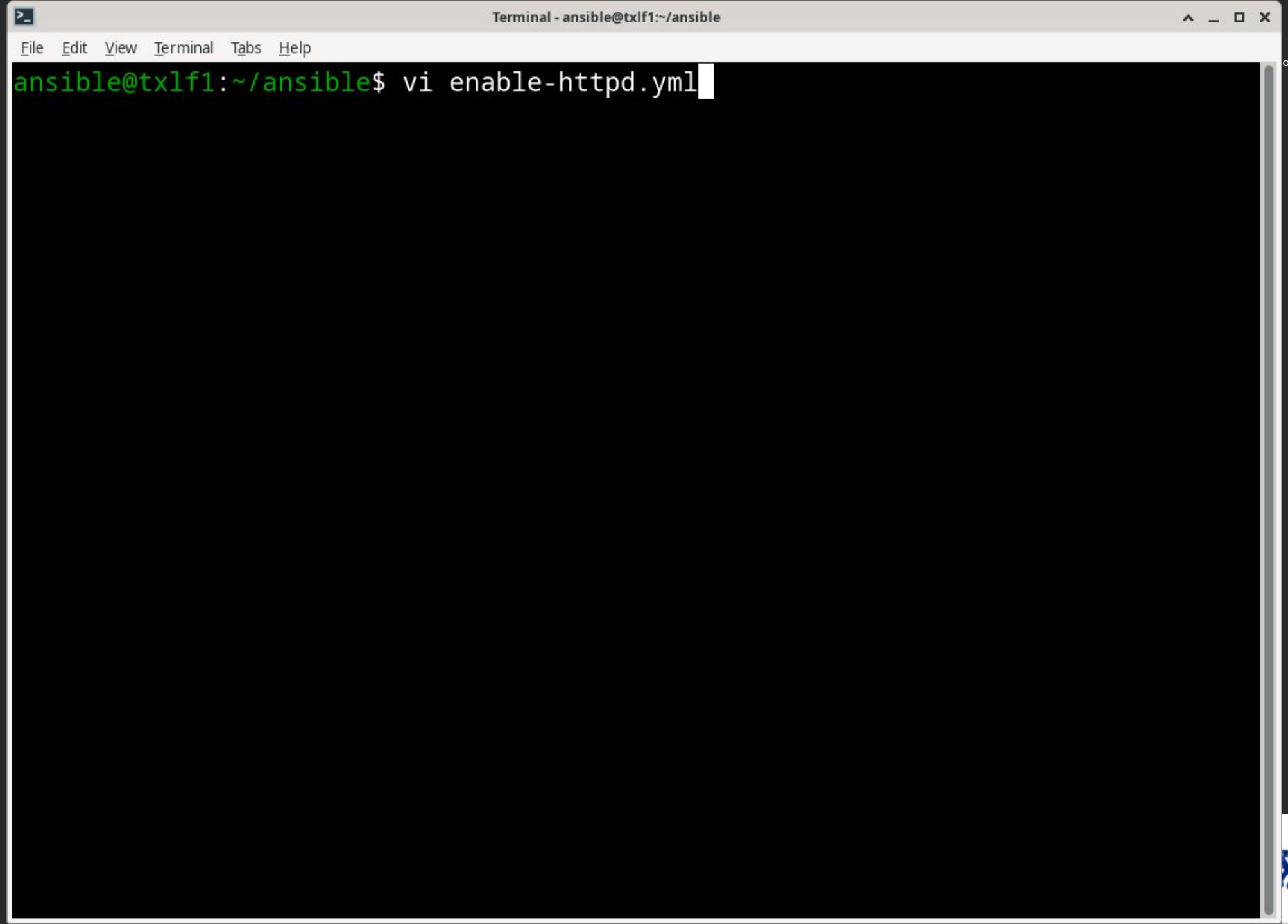
File Edit View Terminal Tabs Help

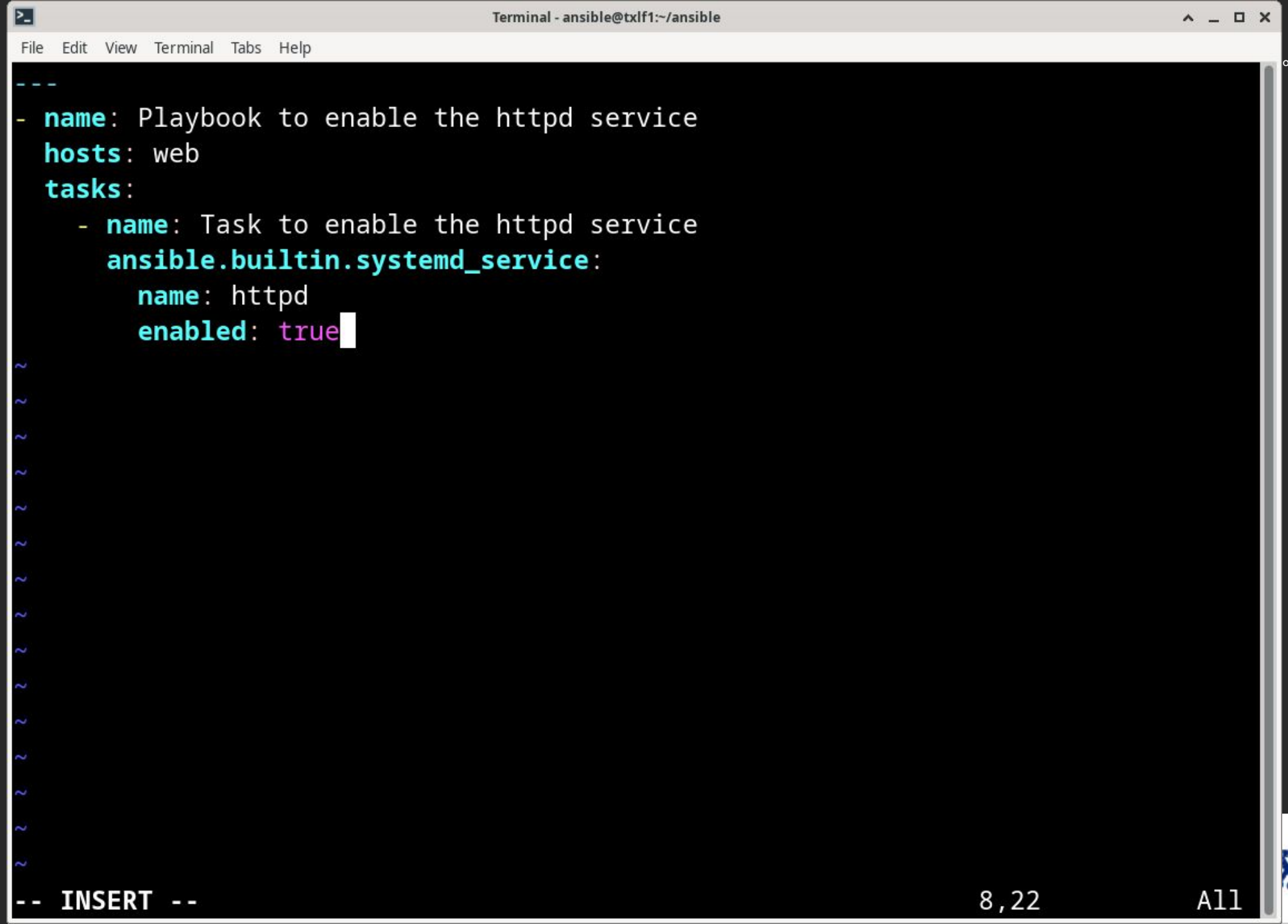
```
EXAMPLES:
- name: Make sure a service unit is running
  ansible.builtin.systemd_service:
    state: started
    name: httpd

- name: Stop service cron on debian, if running
  ansible.builtin.systemd_service:
    name: cron
    state: stopped

- name: Restart service cron on centos, in all cases, also issue daemon-reload >
  ansible.builtin.systemd_service:
    state: restarted
    daemon_reload: true
    name: crond

- name: Reload service httpd, in all cases
  ansible.builtin.systemd_service:
    name: httpd.service
    state: reloaded

- name: Enable service httpd and ensure it is not masked
:
```

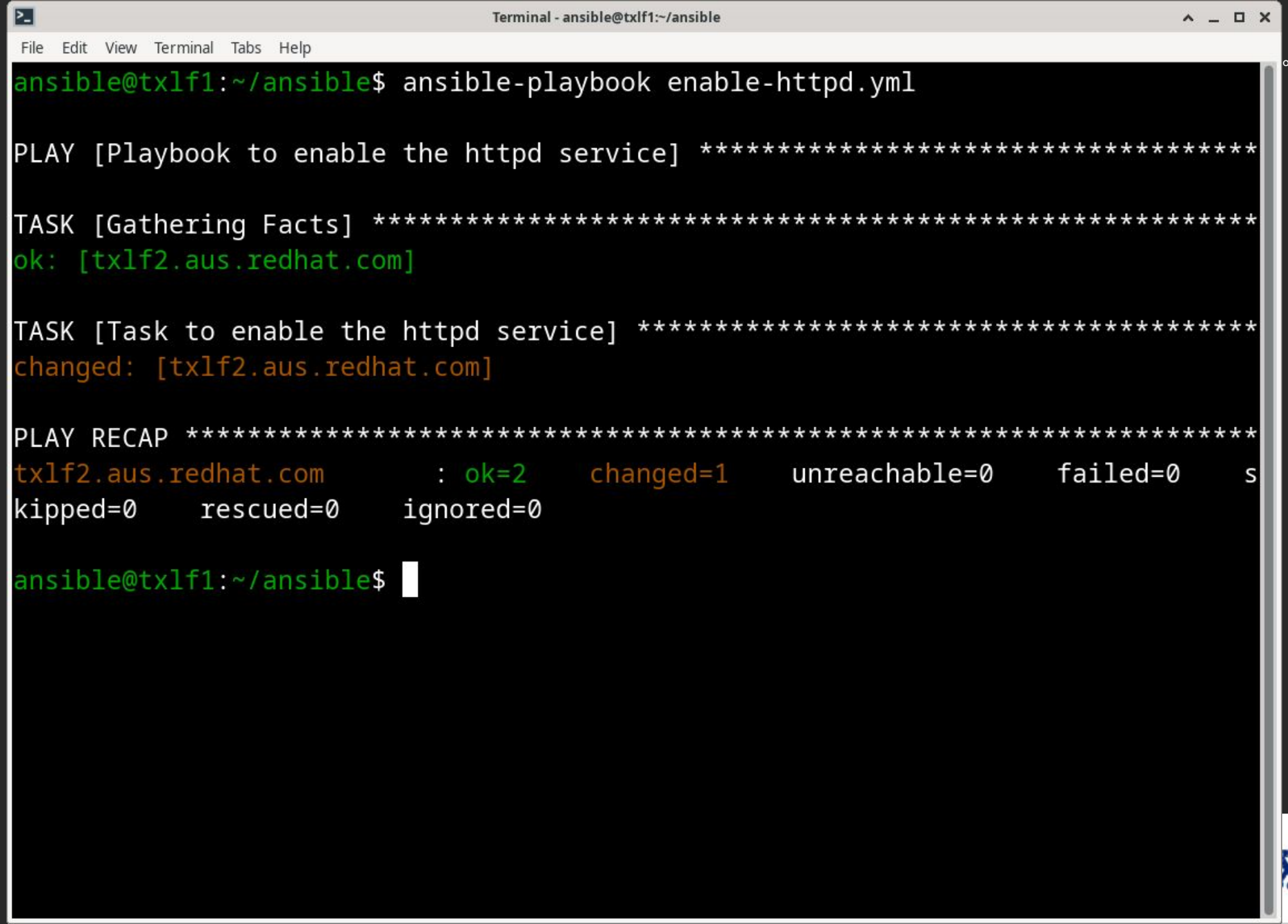
Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

```
  ansible.builtin.systemd_service:
    name: httpd
    enabled: true
    masked: no

- name: Enable a timer unit for dnf-automatic
  ansible.builtin.systemd_service:
    name: dnf-automatic.timer
    state: started
    enabled: true

- name: Just force systemd to reread configs (2.4 and above)
  ansible.builtin.systemd_service:
    daemon_reload: true

- name: Just force systemd to re-execute itself (2.8 and above)
  ansible.builtin.systemd_service:
    daemon_reexec: true

- name: Run a user service when XDG_RUNTIME_DIR is not set on remote login
  ansible.builtin.systemd_service:
    name: myservice
    state: started
:
```







Starting the Service

Remember we said you have to install the package, enable the service, start the service, and open the firewall ports? Let's start the service. This is kind of a silly example, but I want to keep this as simple as possible.

- We're going to create a playbook to start the service.
- If you noticed in the ansible-doc output, you **can actually enable and start a service at the same time**.
- But for today, baby steps!

Source:
Insert source data here
Insert source data here

Starting the Service

Also, I'm gonna be brutally honest here. This is an example of Ansible playbooks sometimes having syntax that doesn't make sense to my neurodivergent mind!

- For some reason, I always feel like it should be `state: enabled`, but it's not!
- It's
  - `enabled: true`
  - `state: started`

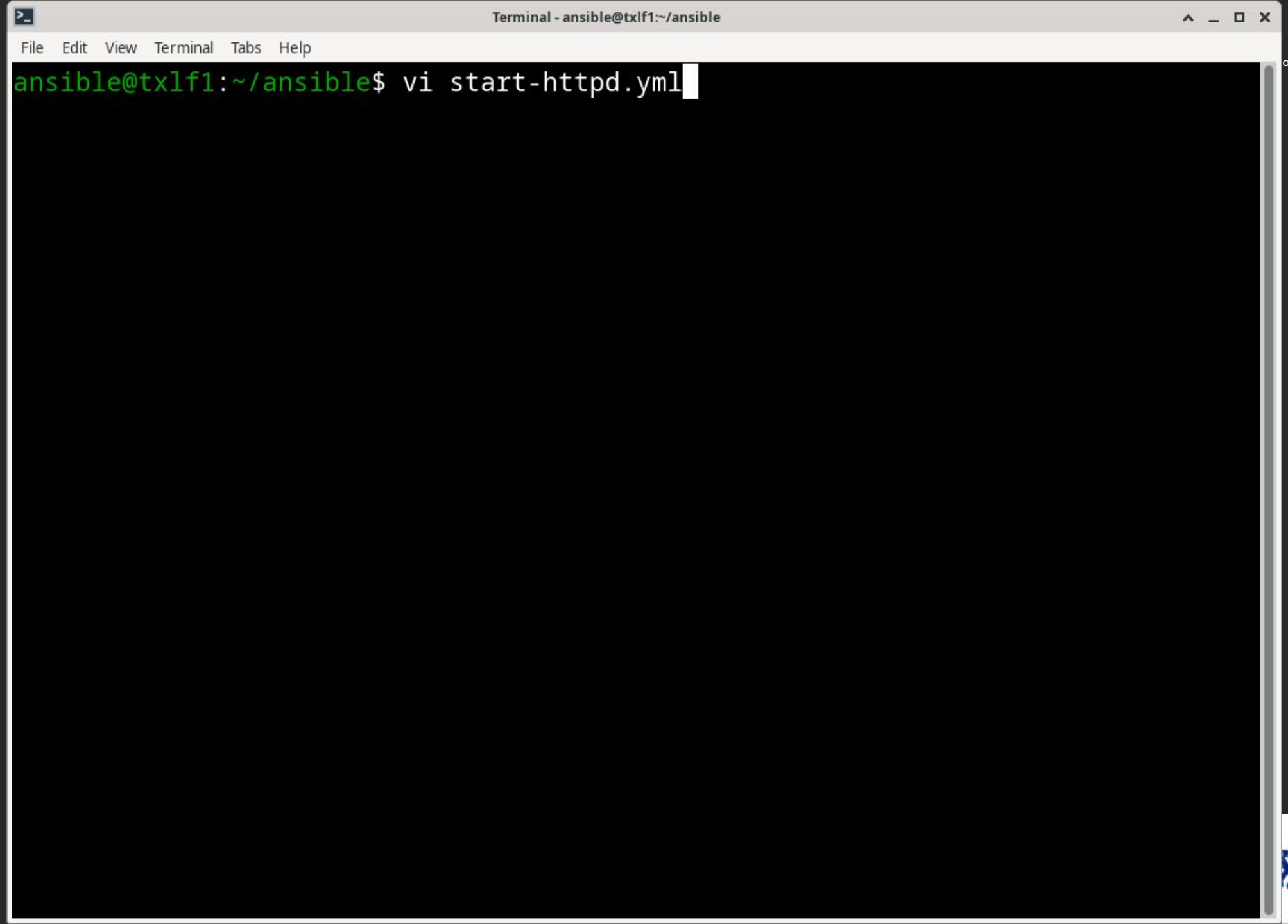
Source:
Insert source data here
Insert source data here

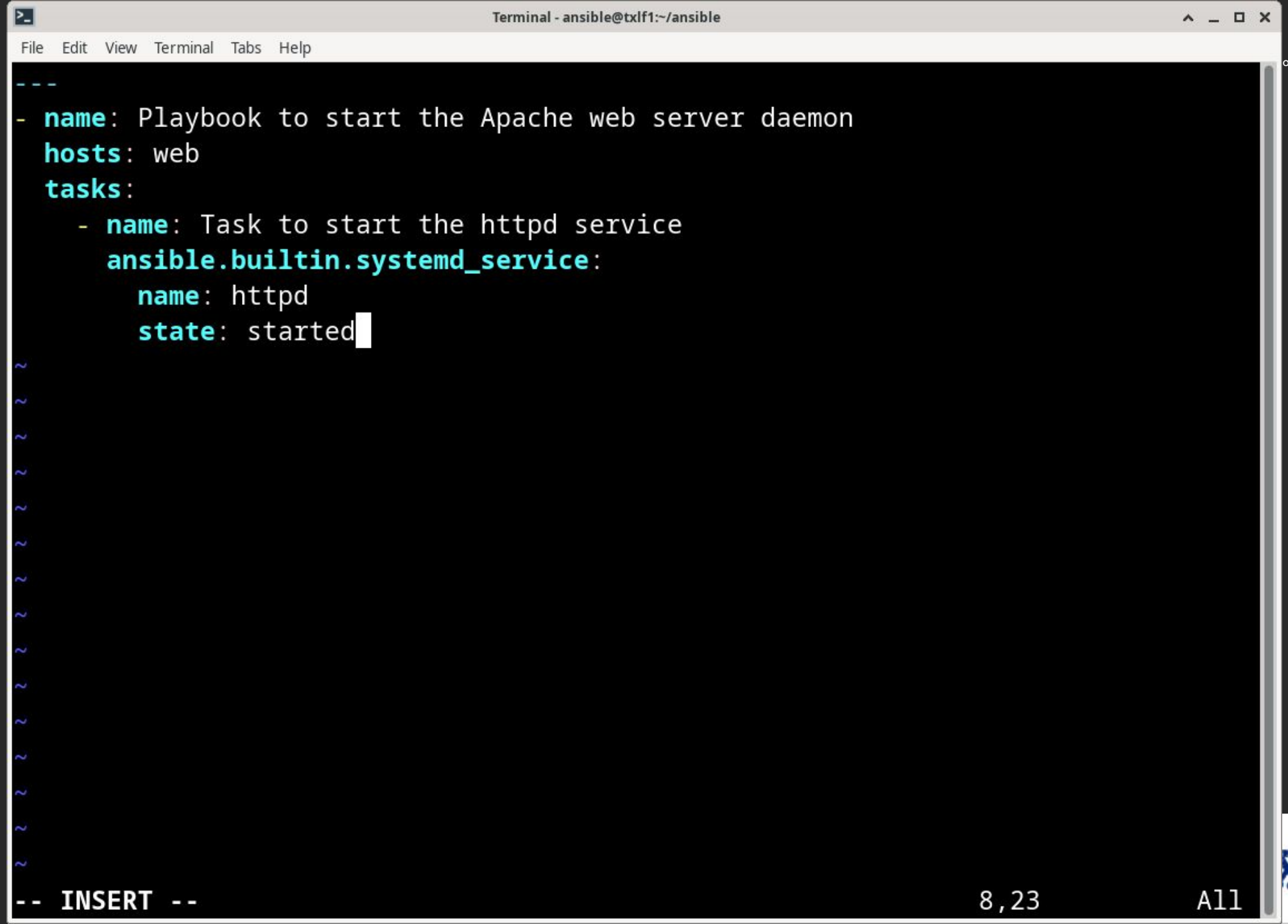
Starting the Service

Also, I'm gonna be brutally honest here. This is an example of Ansible playbooks sometimes having syntax that doesn't make sense to my neurodivergent mind!

- This is why I constantly show you the ansible-doc output. Get used to using ansible-doc to learn playbook syntax and the oddities!

Source:
Insert source data here
Insert source data here





Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

```
ansible@txlf1:~/ansible$ vi start-httpd.yml
ansible@txlf1:~/ansible$ ansible-playbook start-httpd.yml

PLAY [Playbook to start the Apache web server daemon] **************************

TASK [Gathering Facts] *********************************************************
ok: [txlf2.aus.redhat.com]

TASK [Task to start the httpd service] *****************************************
changed: [txlf2.aus.redhat.com]

PLAY RECAP *********************************************************************
txlf2.aus.redhat.com       : ok=2    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0

ansible@txlf1:~/ansible$
```

Automating Firewall Rules

How to open ports

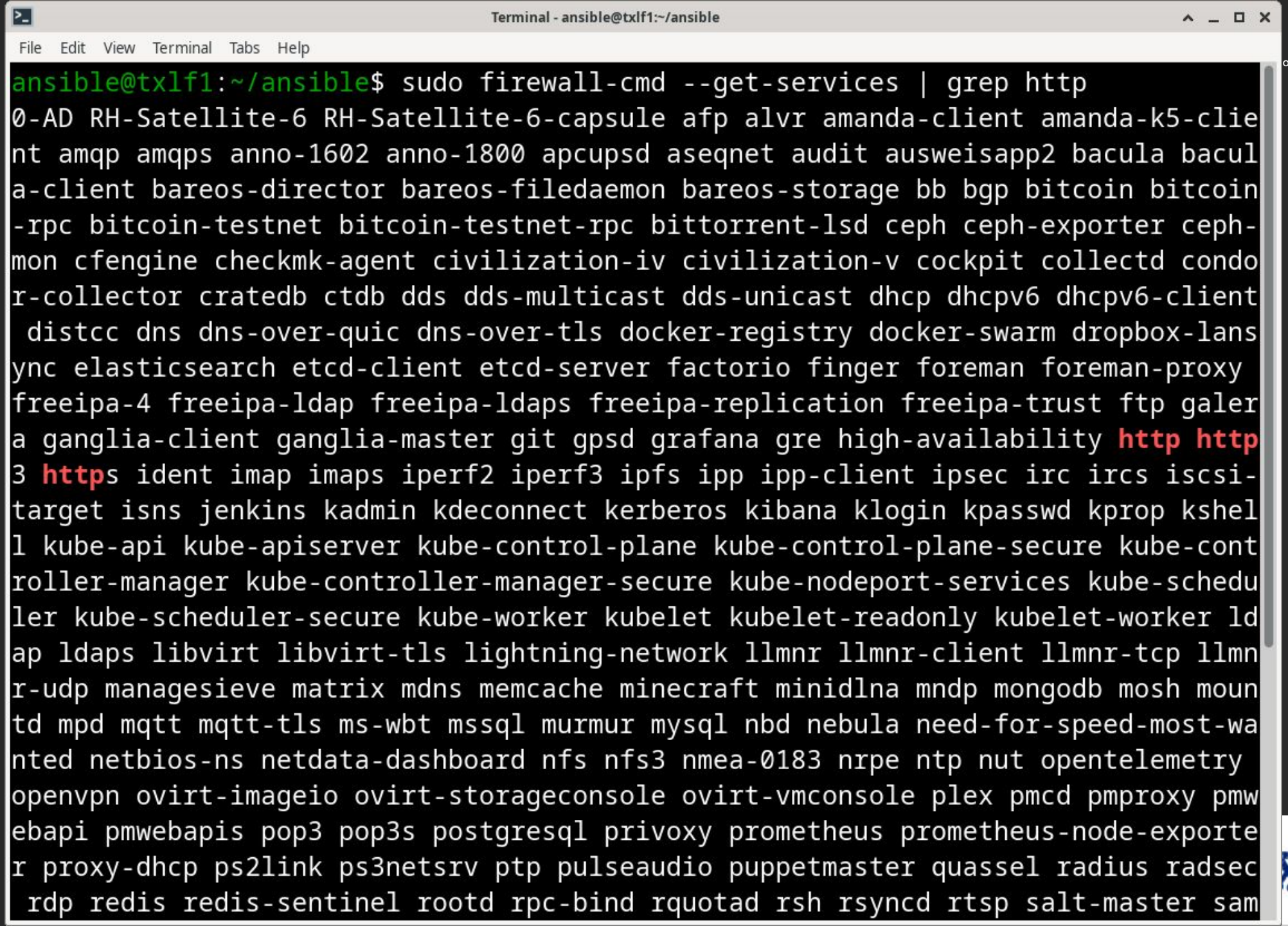
Red Hat TXLF

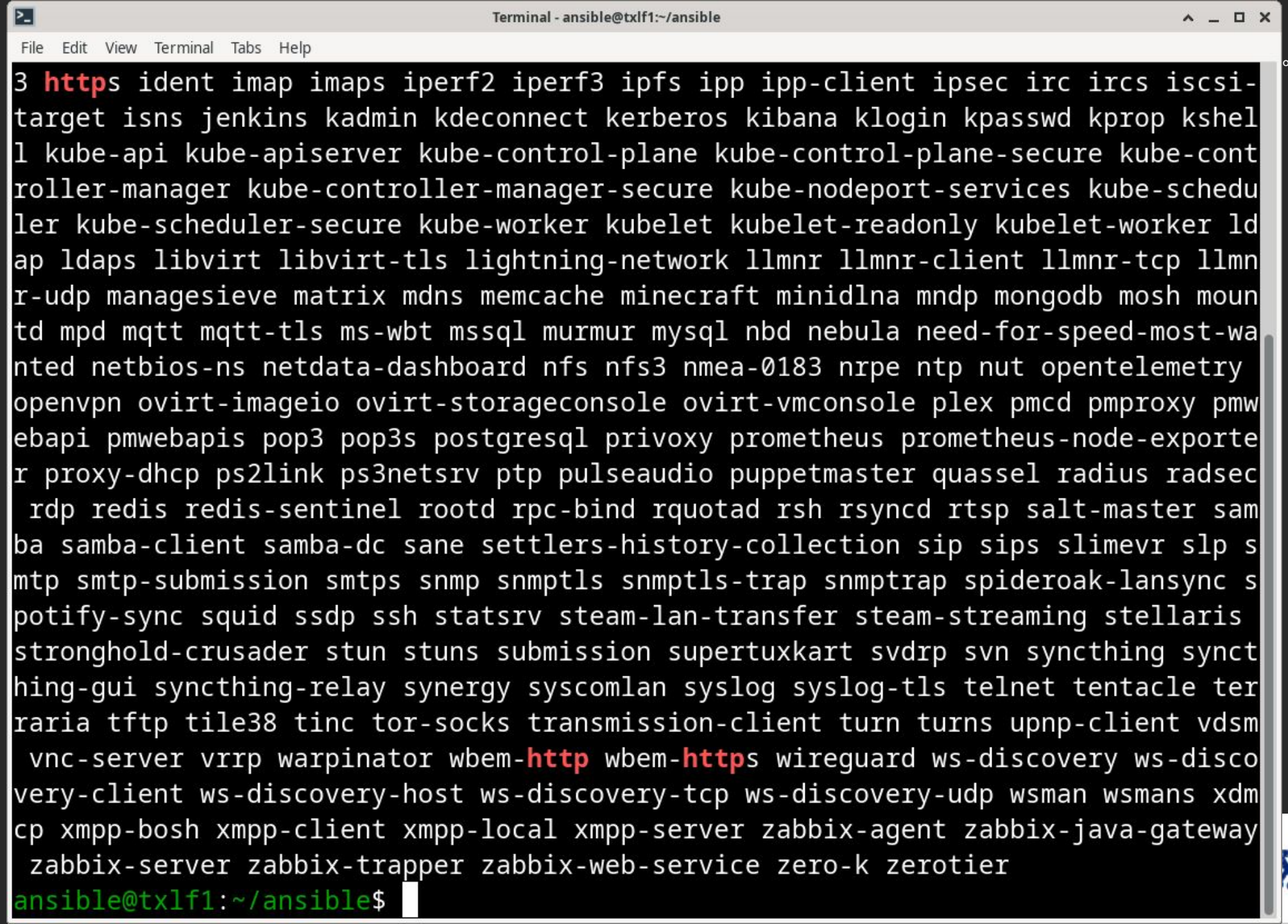
Automating firewall rules

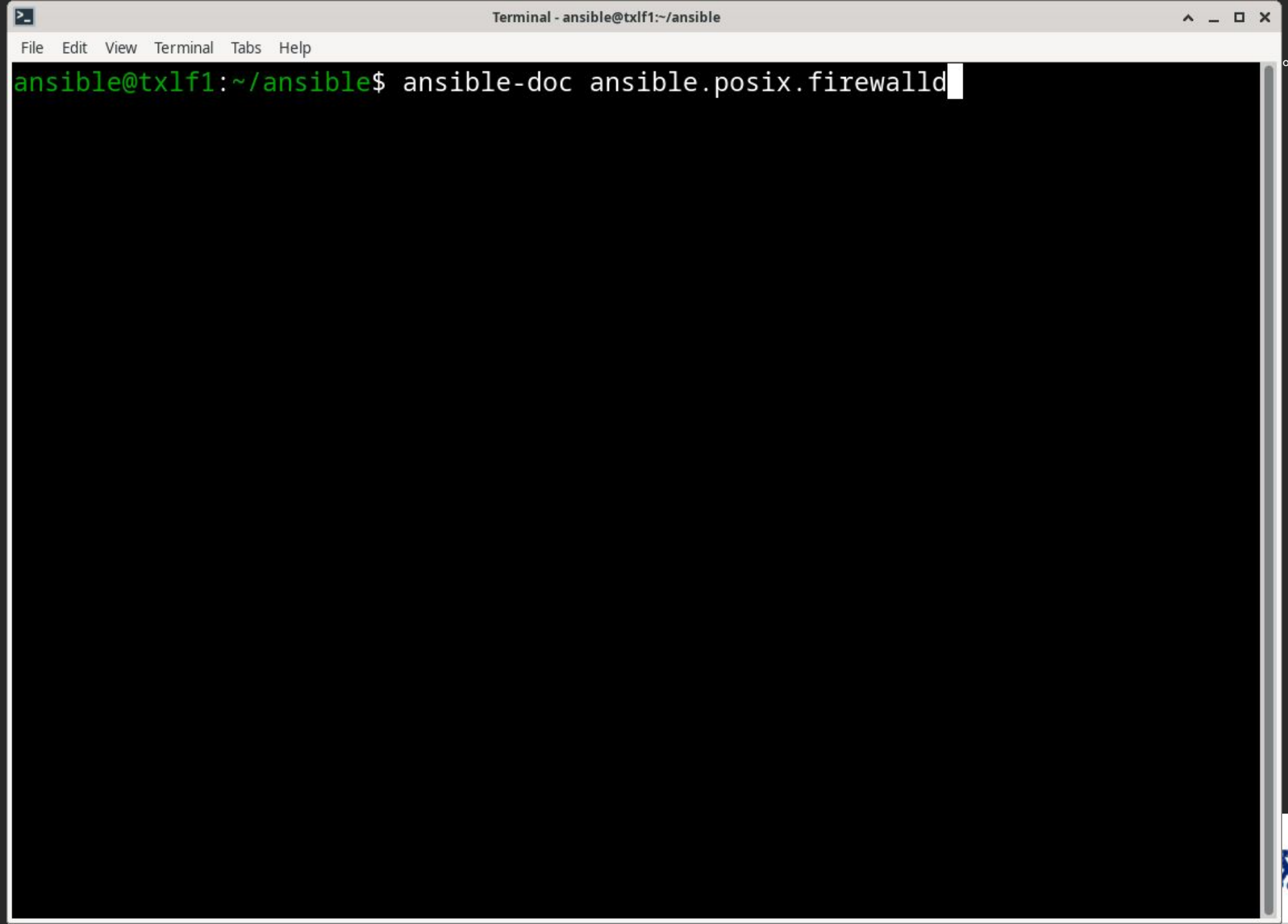
The majority of Linux distros use either firewalld or ufw.

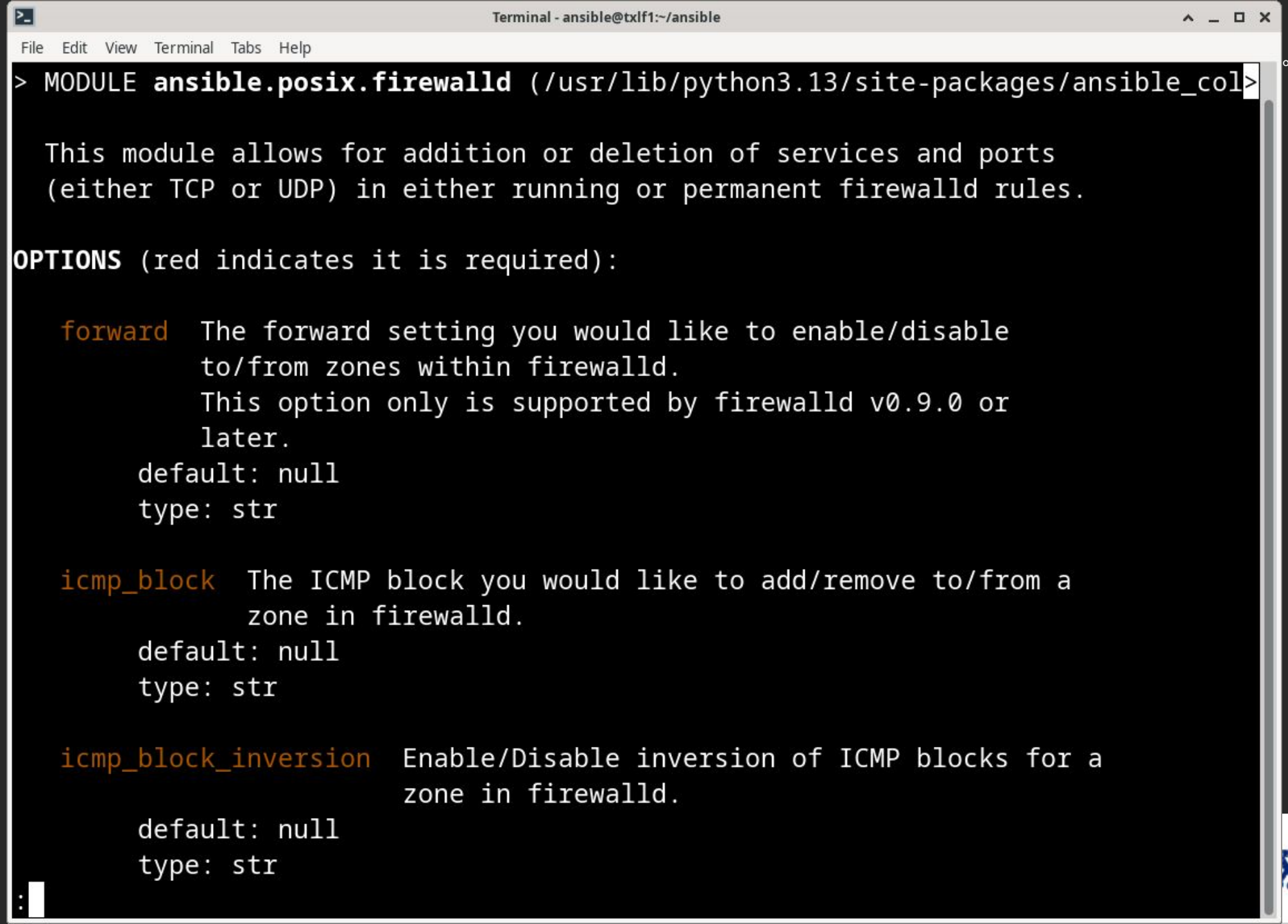
- For this example, we'll use the `ansible.posix.firewalld` collection
- My assumption is that you understand how `firewalld` and `firewall-cmd` work

Source:
Insert source data here
Insert source data here









Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

```
  immediate  Whether to apply this change to the runtime firewalld
             configuration.
             Defaults to `true' if `permanent=false'.
       default: false
       type: bool

  interface  The interface you would like to add/remove to/from a
             zone in firewalld.
       default: null
       type: str

  masquerade  The masquerade setting you would like to enable/disable
              to/from zones within firewalld.
       default: null
       type: str

  offline  Ignores `immediate' if `permanent=true' and firewalld is
           not running.
       default: false
       type: bool

  permanent  Whether to apply this change to the permanent firewalld
:
```

Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

```
            configuration.
            As of Ansible 2.3, permanent operations can operate on
            firewalld configs when it is not running (requires
            firewalld >= 0.3.9).
            Note that if this is `false', `immediate=true' by
            default.
        default: false
        type: bool

    port    Name of a port or port range to add/remove to/from
            firewalld.
            Must be in the form PORT/PROTOCOL or PORT-PORT/PROTOCOL for
            port ranges.
        default: null
        type: str

    port_forward  Port and protocol to forward using firewalld.
        default: null
        elements: dict
        type: list
        suboptions:

            port  Source port to forward from.
:
```

Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

```
  service  Name of a service to add/remove to/from firewalld.
           The service must be listed in output of
           `firewall-cmd --get-services'.
       default: null
       type: str

  source  The source/network you would like to add/remove to/from
          firewalld.
       default: null
       type: str

  state   Enable or disable a setting.
          For ports: Should this port accept (`enabled') or reject
          (`disabled') connections.
          The states `present' and `absent' can only be used in zone
          level operations (i.e. when no other parameters but zone
          and state are set).
       choices: [absent, disabled, enabled, present]
       type: str

  target  firewalld Zone target.
          If `state=absent', this will reset the target to default.
       choices: [default, ACCEPT, DROP, '%%REJECT%%']
:
```

Terminal - ansible@txlf1:~/ansible

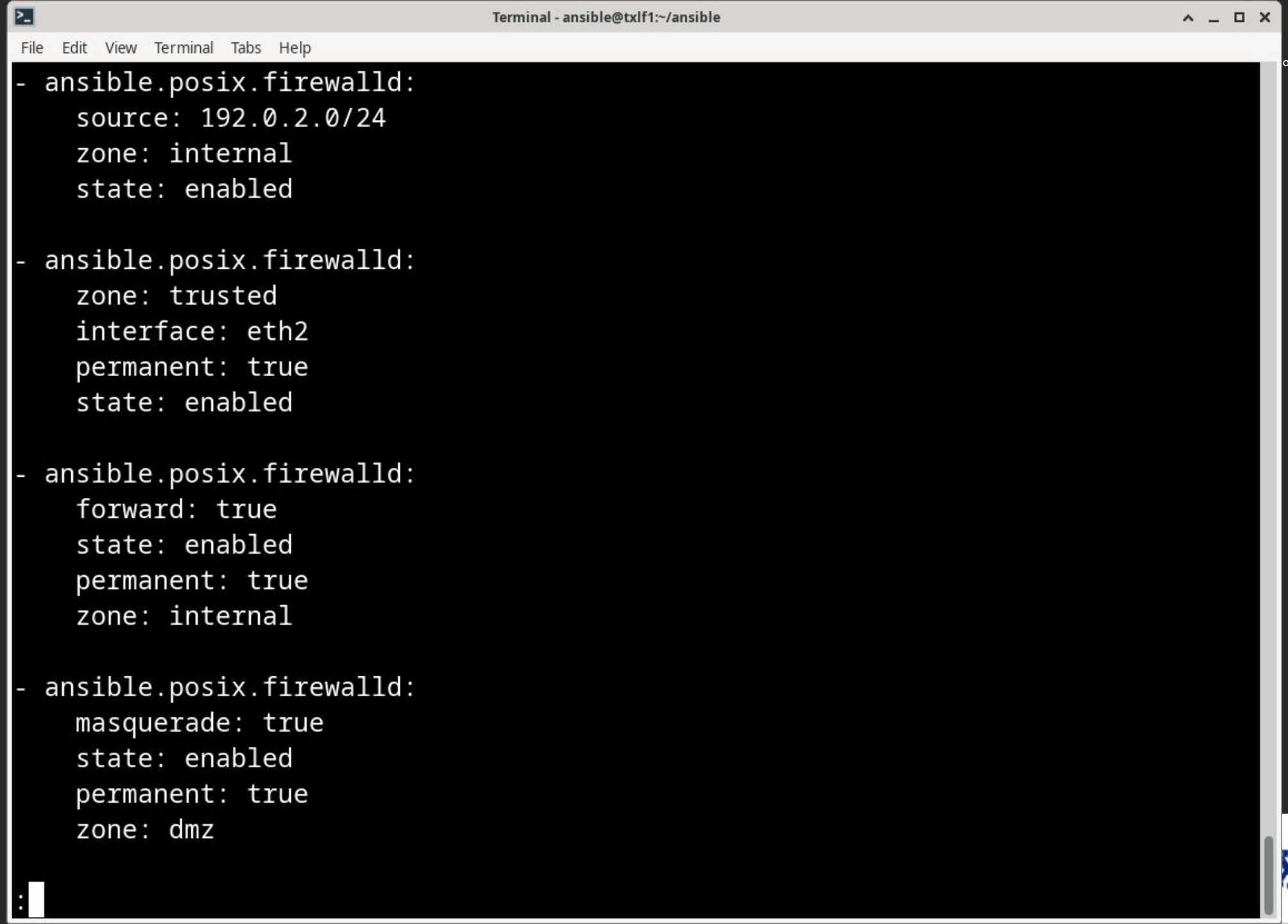
File Edit View Terminal Tabs Help

```
EXAMPLES:
- name: permanently enable https service, also enable it immediately if possible
  ansible.posix.firewalld:
    service: https
    state: enabled
    permanent: true
    immediate: true
    offline: true

- name: permit traffic in default zone for https service
  ansible.posix.firewalld:
    service: https
    permanent: true
    state: enabled

- name: permit ospf traffic
  ansible.posix.firewalld:
    protocol: ospf
    permanent: true
    state: enabled

- name: do not permit traffic in default zone on port 8081/tcp
  ansible.posix.firewalld:
:
```

Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

```
- name: do not permit traffic in default zone on port 8081/tcp
  ansible.posix.firewalld:
    port: 8081/tcp
    permanent: true
    state: disabled

- ansible.posix.firewalld:
    port: 161-162/udp
    permanent: true
    state: enabled

- ansible.posix.firewalld:
    zone: dmz
    service: http
    permanent: true
    state: enabled

- ansible.posix.firewalld:
    rich_rule: rule service name="ftp" audit limit value="1/m" accept
    permanent: true
    state: enabled

- ansible.posix.firewalld:
:
```

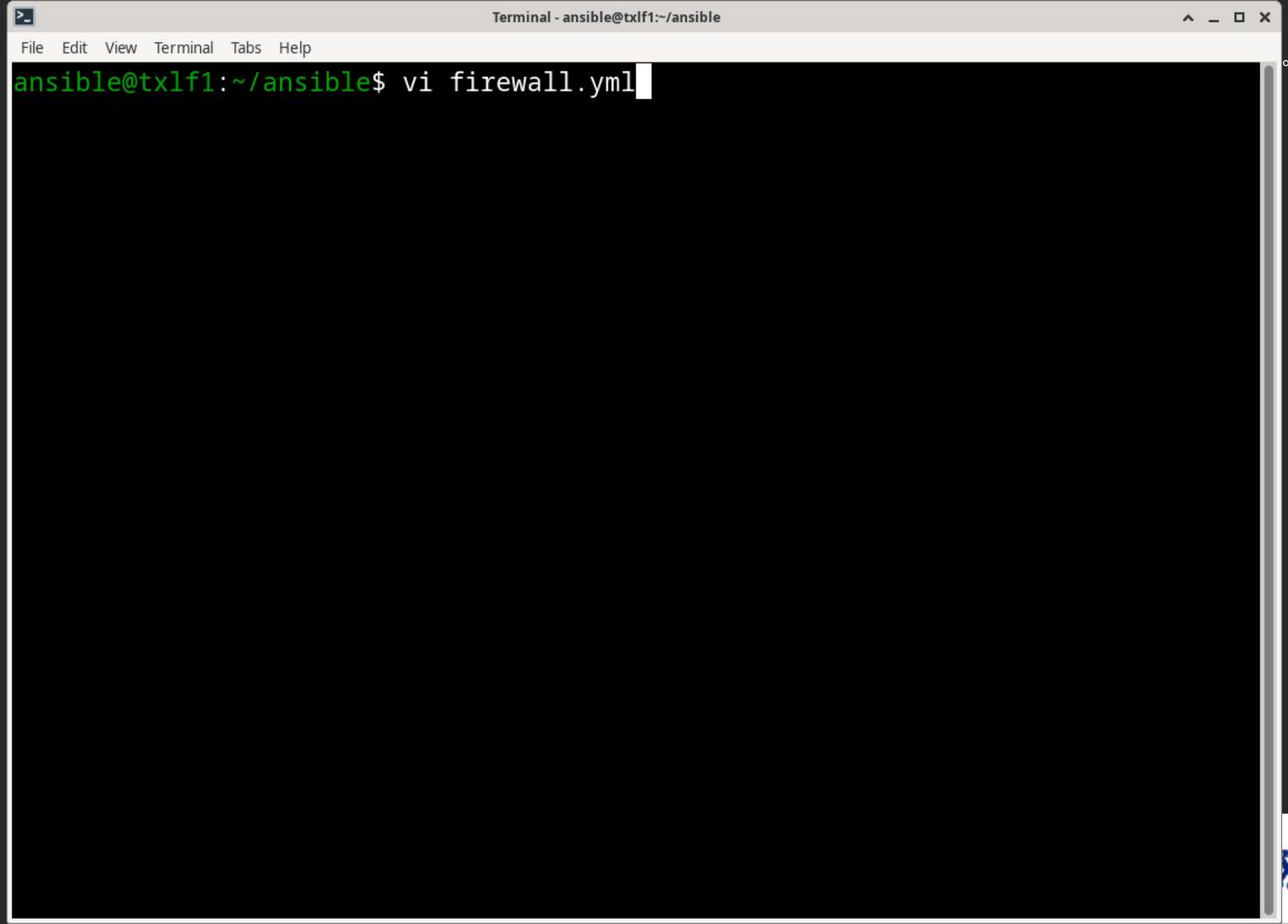


Automating firewall rules

Y'all are getting the hang of this!

- I'm going to make this playbook a little more complicated.
- Instead of one task, we're going to combine a couple of tasks in the playbook

Source:
Insert source data here
Insert source data here



Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

```
---
- name: Playbook to open firewall ports for the web server
  hosts: web
  tasks:
    - name: Task to allow access to the http service
      ansible.posix.firewalld:
        service: http
        state: enabled
        permanent: true
        immediate: true
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                    10,24         All
```

Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

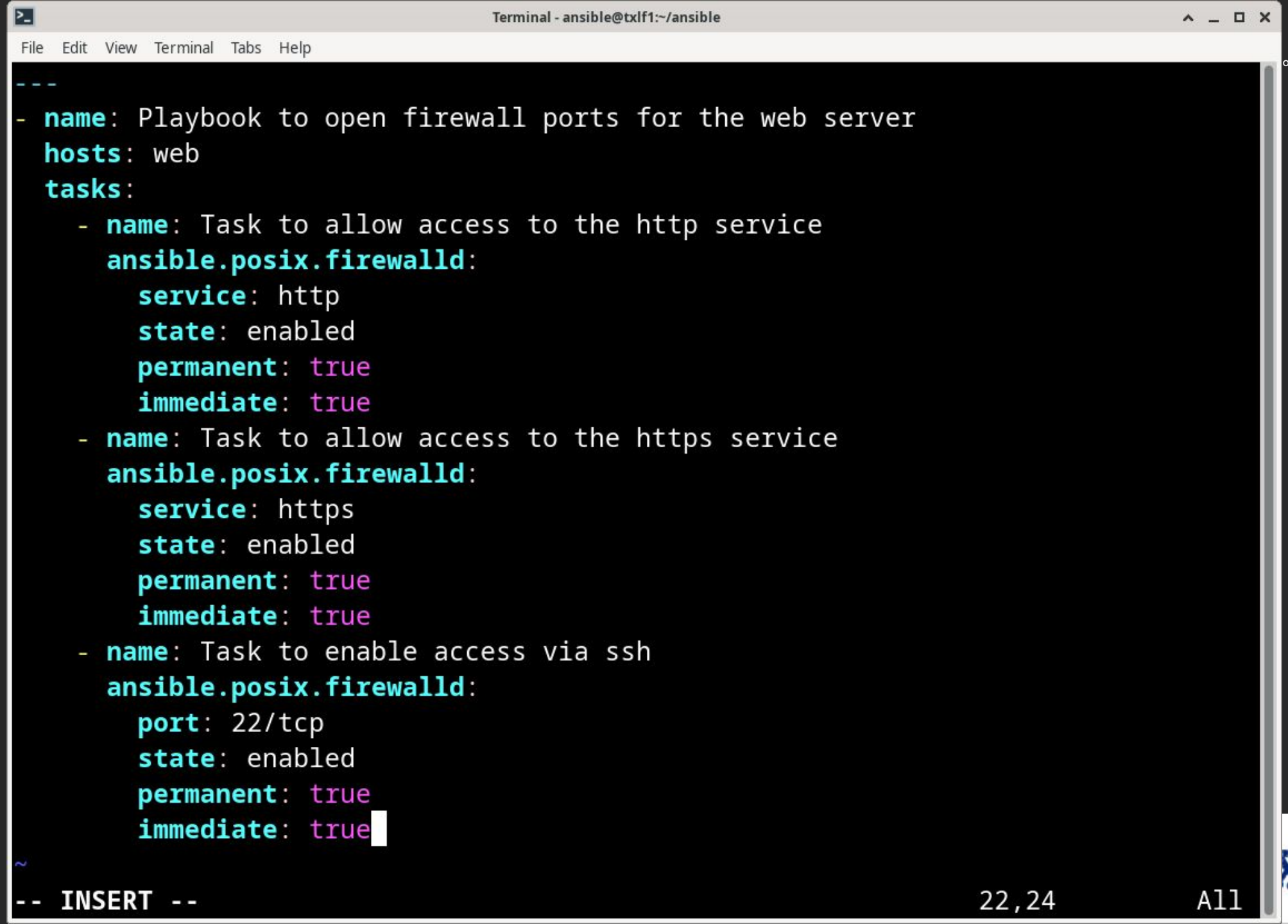
```
---
- name: Playbook to open firewall ports for the web server
  hosts: web
  tasks:
    - name: Task to allow access to the http service
      ansible.posix.firewalld:
        service: http
        state: enabled
        permanent: true
        immediate: true
    - name: Task to allow access to the https service
      ansible.posix.firewalld:
        service: https
        state: enabled
        permanent: true
        immediate: true
~
~
~
~
~
~
~
~
-- INSERT --                                    16,24         All
```

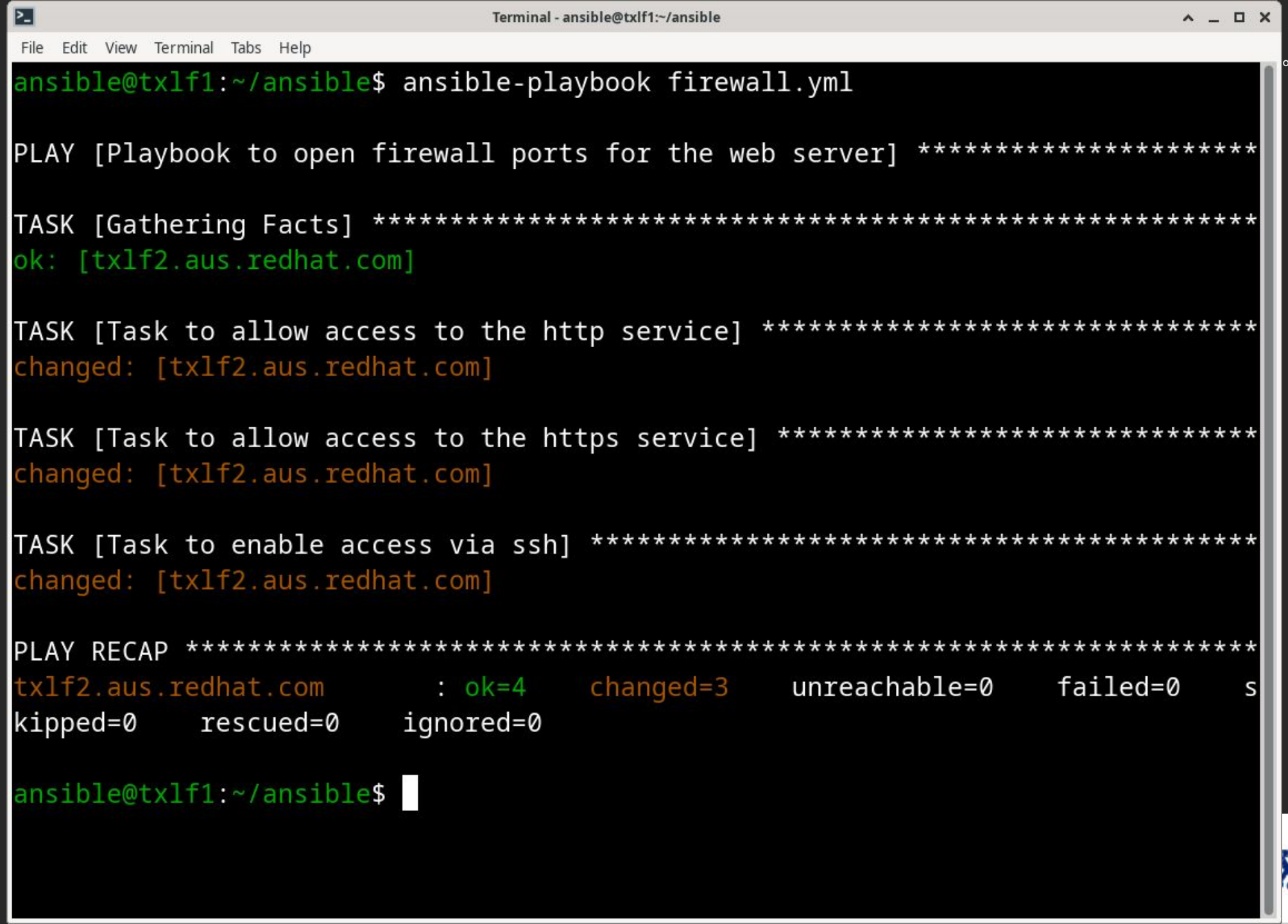
Automating firewall rules

You can use the service name you get from `firewall-cmd --get-services` (preferred), or you can open individual ports

- In this playbook, I'm showing you how to open ports.
- For uniformity, I recommend you consistently use service names

Source:
Insert source data here
Insert source data here





Fedora Webserver Test Page

If you can read this page, it means that the web server installed at this site is working properly, but has not yet been configured.

If you are a member of the general public:

The website you just visited is either **experiencing problems** or **undergoing routine maintenance**.

To let the administrators of this website know that you are seeing this page and not what you were expecting, an e-mail addressed to "webmaster" at the website's domain should reach an appropriate person. For example, if you saw this page while visiting **www.example.com**, you could send e-mail to "**webmaster@example.com**".

Fedora is a distribution of Linux, a popular computer operating system. It is commonly used by hosting companies because it is free, and includes free web server software. This "test page" is shown instead of the expected website if they do not set up their web server correctly.

Accordingly, please keep these facts in mind:

- Neither the Fedora Project or Red Hat has any affiliation

If you are the website administrator:

You may now add content to the webroot directory. Note that until you do so, people visiting your website will see this page, and not your content.

For systems using Apache Webserver: You may now add content to the directory `/var/www/html/`. Note that until you do so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the file `/etc/httpd/conf.d/welcome.conf`.

For systems using Nginx: You should now put your content in a location of your choice and edit the `root` configuration directive in the **nginx** configuration file `/etc/nginx/nginx.conf`.

For systems using Caddy: You should now put your content in a location of your choice and edit the root configuration directive in the Caddy configuration file `/etc/caddy/Caddyfile`.

Important note: Idempotency

Ansible is **idempotent**

Idempotency

Idempotence is the property of certain operations in mathematics and computer science whereby they can be applied multiple times without changing the result beyond the initial application.

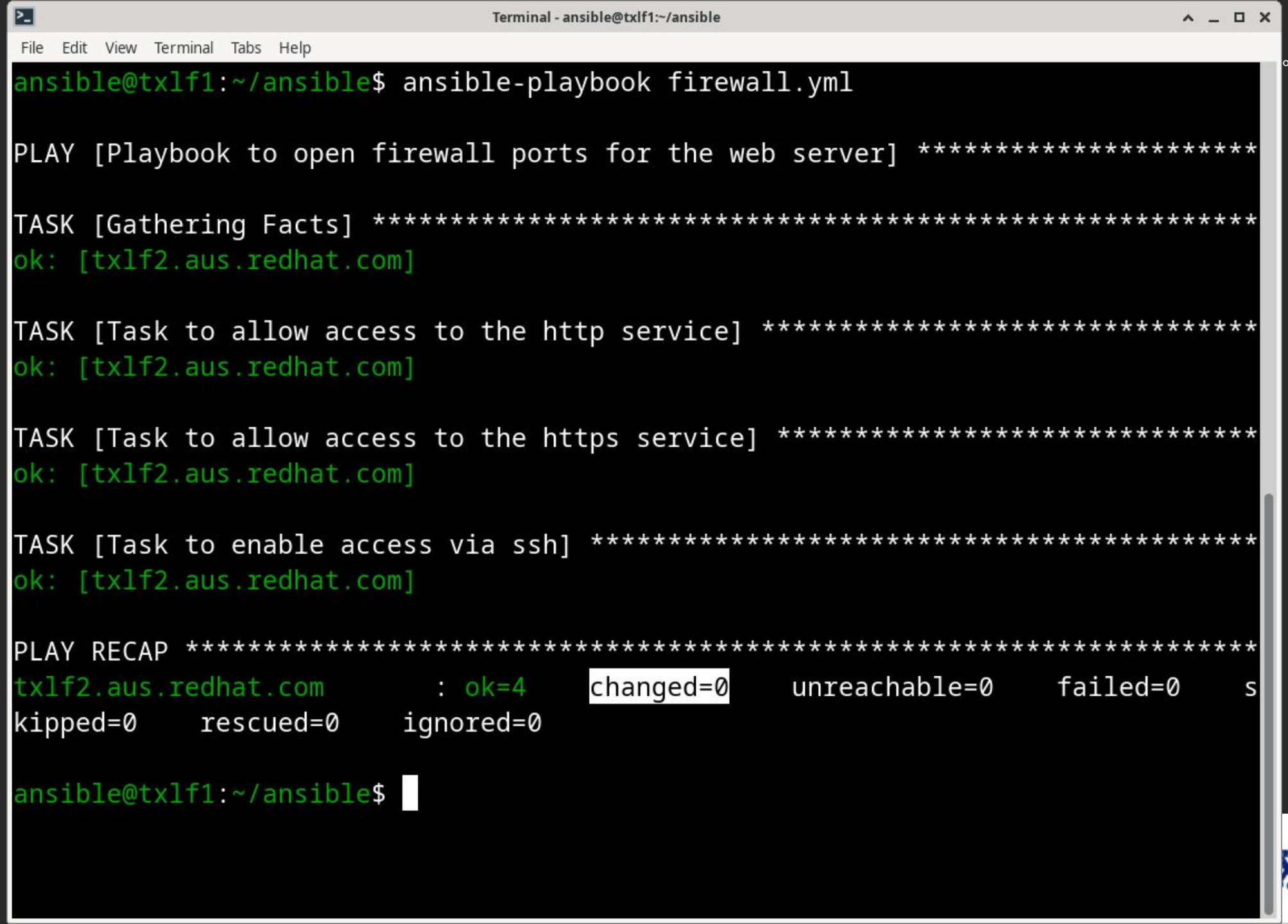
- Ansible is idempotent.
- In other words, you can run Ansible playbooks multiple times and it won't make changes unless needed

Source:
https://en.wikipedia.org/wiki/Idempotence

Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

```
ansible@txlf1:~/ansible$ ls
ansible.cfg       firewall.yml       install-mod_ssl.yml  myplaybook.yml
enable-httpd.yml  install-httpd.yml  inventory.ini        start-httpd.yml
ansible@txlf1:~/ansible$ ansible-playbook install-httpd.yml

PLAY [Playbook to install Apache httpd] ****************************************

TASK [Gathering Facts] *********************************************************
ok: [txlf2.aus.redhat.com]

TASK [Task to install httpd] ***************************************************
ok: [txlf2.aus.redhat.com]

PLAY RECAP *********************************************************************
txlf2.aus.redhat.com       : ok=2    changed=0    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0

ansible@txlf1:~/ansible$
```



Tying it all together

You can create playbooks with multiple plays!

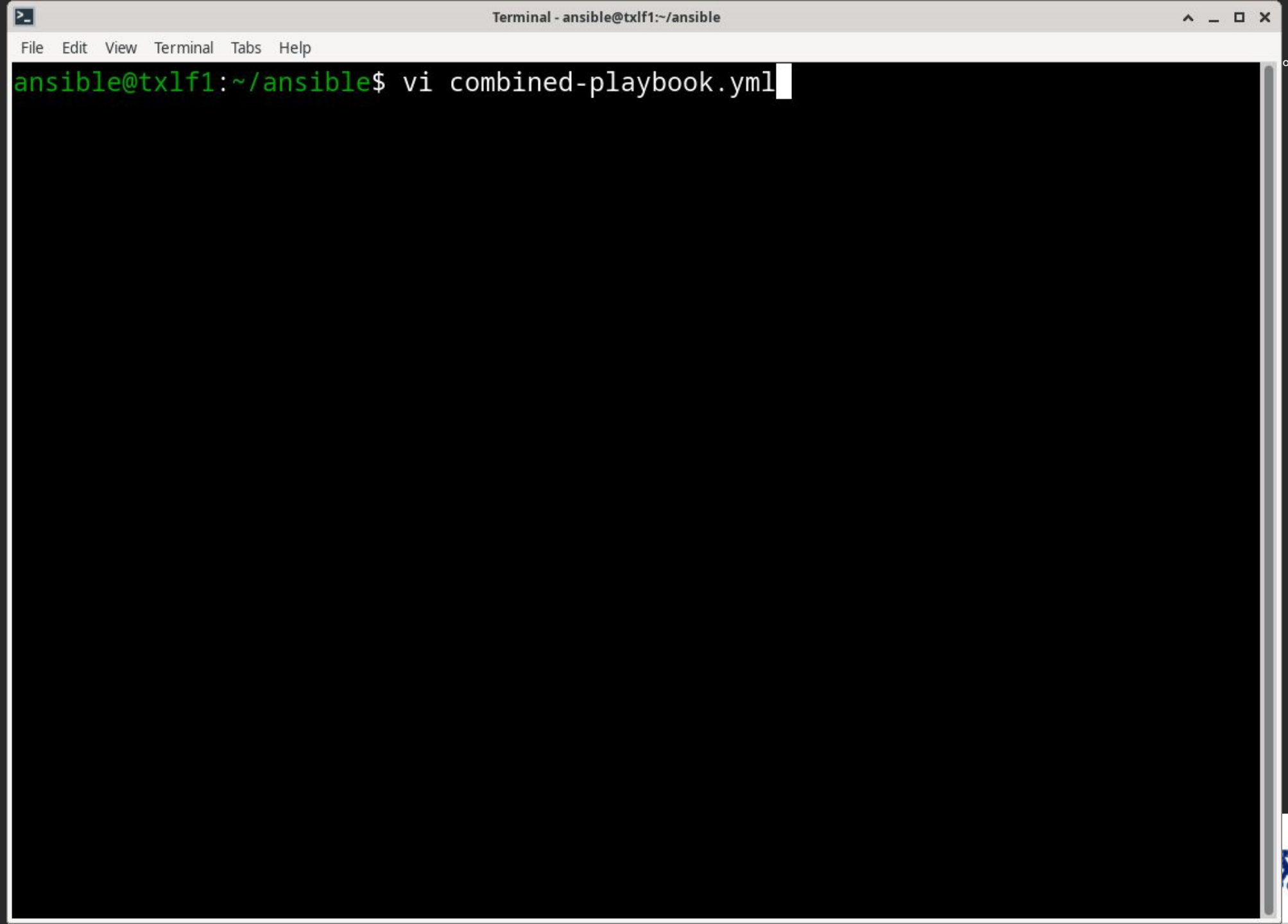
Tying it all together

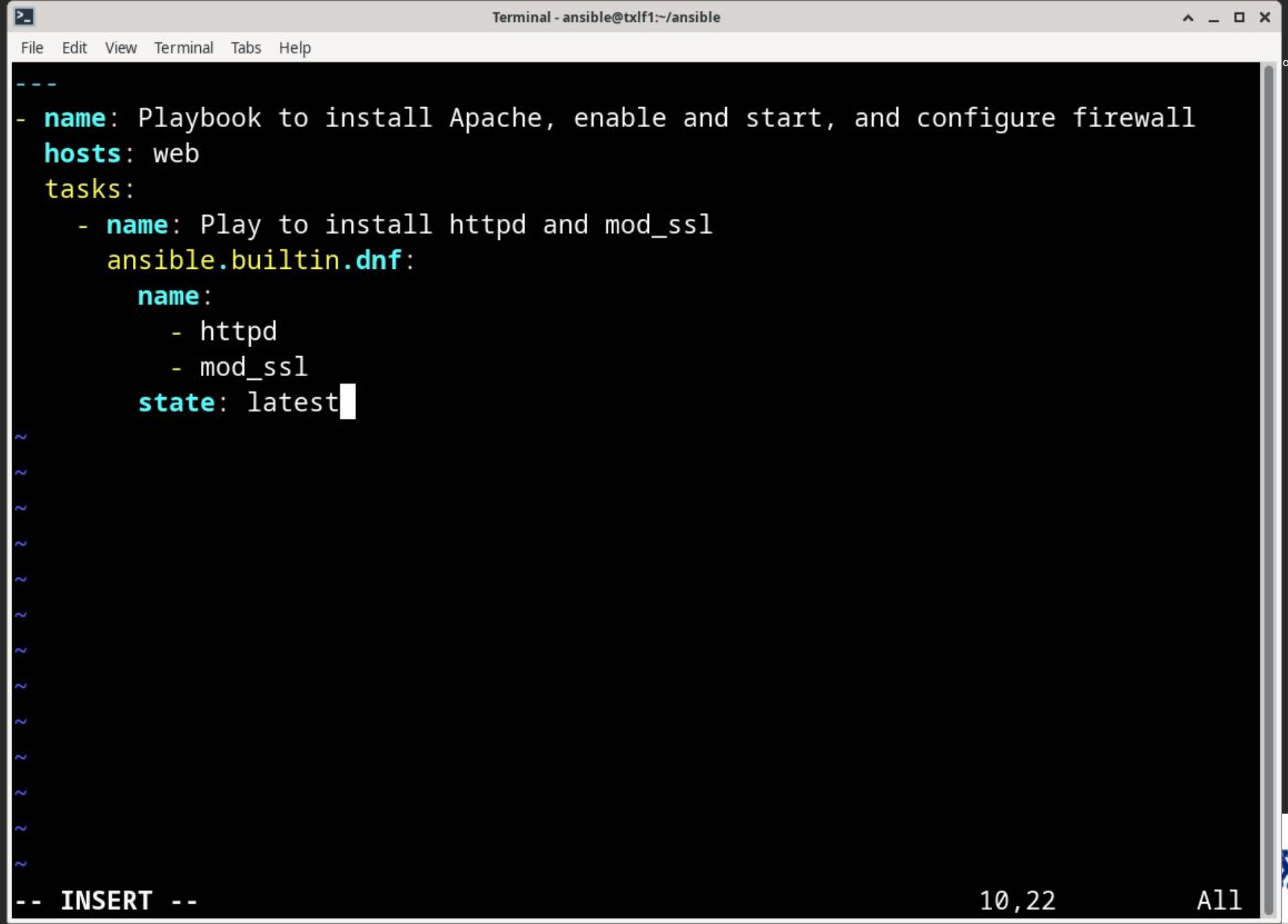
I have intentionally broken this down into little bite-sized morsels.

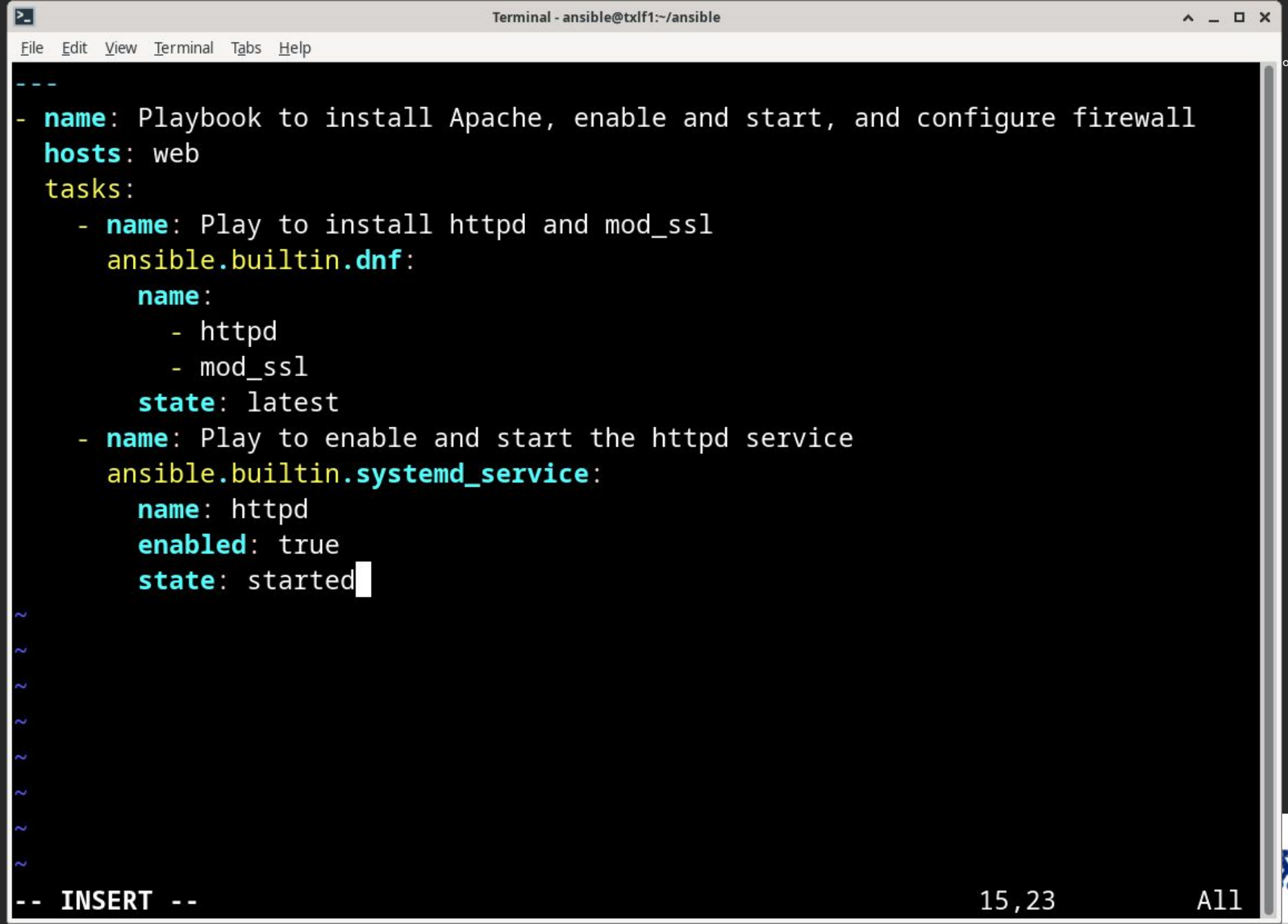
- You saw me do one individual task per playbook.
- Then you saw me add a couple of tasks in the firewall playbook.
- You can create a playbook with ALL of the steps we've covered

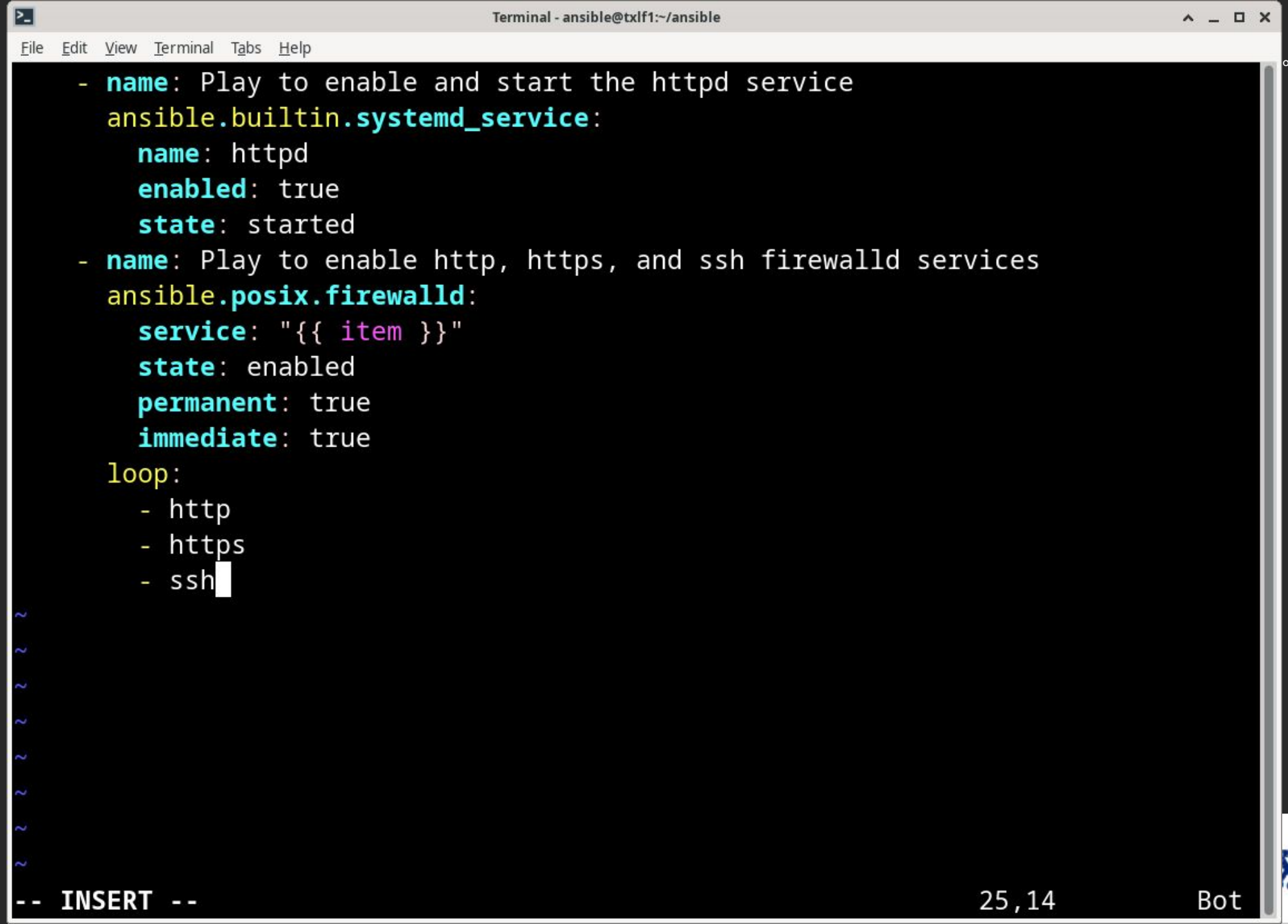
Source:
https://en.wikipedia.org/wiki/Idempotence

Red Hat TXLF









Terminal - ansible@txlf1:~/ansible

File Edit View Terminal Tabs Help

```
ansible@txlf1:~/ansible$ ansible-playbook combined-playbook.yml --syntax-check

playbook: combined-playbook.yml
ansible@txlf1:~/ansible$ ansible-playbook combined-playbook.yml

PLAY [Playbook to install Apache, enable and start, and configure firewall] ****

TASK [Gathering Facts] *********************************************************
ok: [txlf2.aus.redhat.com]

TASK [Play to install httpd and mod_ssl] ***************************************
changed: [txlf2.aus.redhat.com]

TASK [Play to enable and start the httpd service] ******************************
changed: [txlf2.aus.redhat.com]

TASK [Play to enable http, https, and ssh firewalld services] ******************
changed: [txlf2.aus.redhat.com] => (item=http)
changed: [txlf2.aus.redhat.com] => (item=https)
ok: [txlf2.aus.redhat.com] => (item=ssh)

PLAY RECAP *********************************************************************
txlf2.aus.redhat.com       : ok=4    changed=3    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

Any Questions?

Feel free to ask! If you already knew this, you wouldn't be here!

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.


linkedin.com/company/red-hat


youtube.com/user/RedHatVideos


facebook.com/redhatinc


twitter.com/RedHat