



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!

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

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

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???



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



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!



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.

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.

About Ansible

Why you should use it?

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

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

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.

```
Terminal - tcameron@case:~/Desktop
File Edit View Terminal Tabs Help

tcameron@case:~/Desktop$ ansible-doc -l 2> /dev/null | awk -F \. '{ print $1"."$
2 }'| uniq -c | sort -g -r
1857 cisco.dnac
1196 fortinet.fortimanager
694 fortinet.fortios
619 cisco.meraki
601 community.general
445 cisco.ise
347 check_point.mgmt
342 azure.azcollection
315 community.network
261 cisco.aci
179 f5networks.f5_modules
176 community.vmware
171 google.cloud
155 netapp.ontap
141 community.aws
140 amazon.aws
134 vmware.vmware_rest
130 kaytus.ksmanage
130 ieisystem.inmanage
128 inspur.ispim
108 cisco.mso
105 dell EMC.openmanage
```

Confidentiality: **Public**

```
Terminal - tcameron@case:~/Desktop
File Edit View Terminal Tabs Help
105 dellemc.openmanage
90 openstack.cloud
88 netbox.netbox
87 theforeman.foreman
85 community.windows
84 cisco.nxos
74 hitachivantara.vspone_block
71 ansible.builtin
67 ansible.windows
66 purestorage.flasharray
65 dellemc.enterprise_sonic
58 ovirt.ovirt
56 community.digitalocean
55 netapp_eseries.santricity
53 ngine_io.cloudstack
52 purestorage.flashblade
51 sensu.sensu_go
48 ibm.storage_virtualize
47 awx.awx
45 community.zabbix
43 ibm.spectrum_virtualize
40 telekom_mms.icinga_director
39 junipernetworks.junos
39 cisco.ios
```

Confidentiality: **Public**

Why Ansible

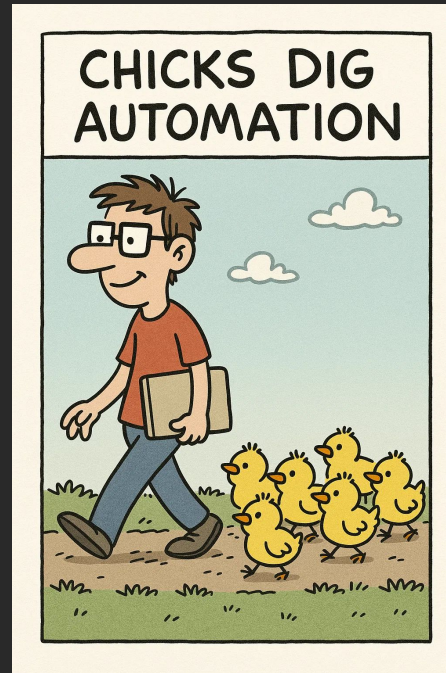
Why you should use it?

- ▶ It's COOL

Why Ansible

Why you should use it?

- ▶ It's COOL



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

Ansible Core (command line) vs. AAP

Ansible Automation Platform is designed for enterprise use

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

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

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

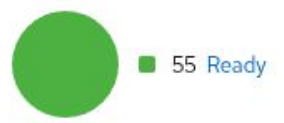
Welcome to the Ansible Automation Platform

[Manage view](#)

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

Resource Counts

55 Hosts



2 Projects



2 Inventories



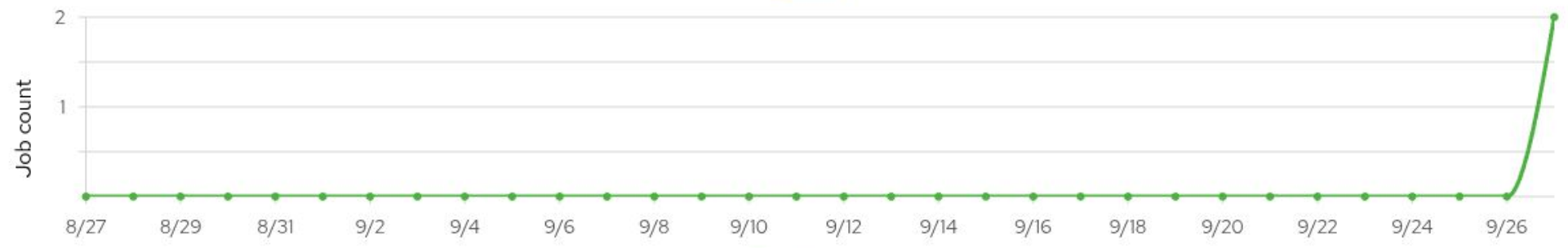
Job Activity



Past month

All job types

[View all Jobs](#)



2 Success

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

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.

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 "storageops" or similar. Grant them access via sudoers.

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 "storageops" or similar. Grant them access via sudoers.
- ▶ Create ssh keys using ssh-keygen

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

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
```

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

txlf1.aus.redhat.com on QEMU/KVM: molly

File Virtual Machine View Send Key



```
root@txlf1:~# useradd -G wheel ansible
root@txlf1:~# passwd ansible
New password:
Retype new password:
passwd: password updated successfully
root@txlf1:~# _
```

tiality: **Public**

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

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

ansible@txlf1:~/ansible$ sudo dnf list ansible-collection\*
Updating and loading repositories:
Repositories loaded.
Available packages
ansible-collection-ansible-netcommon.noarch                7.1.0-1.fc42
f
ansible-collection-ansible-netcommon-doc.noarch            7.1.0-1.fc42
f
ansible-collection-ansible-posix.noarch                     2.0.0-1.fc42
f
ansible-collection-ansible-utils.noarch                     5.0.0-2.fc42
f
ansible-collection-ansible-windows.noarch                   2.7.0-1.fc42
f
ansible-collection-awx-awx.noarch                           24.6.1-3.fc42
f
ansible-collection-chocolatey-chocolatey.noarch            1.5.3-2.fc42
f
ansible-collection-community-crypto.noarch                  2.22.1-2.fc42
f
ansible-collection-community-docker.noarch                   4.1.0-2.fc42
f
ansible-collection-community-general.noarch                  10.7.3-1.fc42
u
```

Confidentiality: **Public**

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

ansible-collection-community-general.noarch 10.7.3-1.fc42
u
ansible-collection-community-internal_test_tools.noarch 0.11.0-2.fc42
f
ansible-collection-community-kubernetes.noarch 2.0.1-11.fc42
f
ansible-collection-community-library_inventory_filtering_v1.noarch 1.0.1-2.fc42
f
ansible-collection-community-libvirt.noarch 1.3.1-2.fc42
f
ansible-collection-community-mysql.noarch 3.12.0-1.fc42
f
ansible-collection-community-postgresql.noarch 3.0.0-6.fc42
f
ansible-collection-community-rabbitmq.noarch 1.3.0-3.fc42
f
ansible-collection-containers-podman.noarch 1.16.3-7.fc42
f
ansible-collection-dellemc-openmanage.noarch 9.10.0-2.fc42
f
ansible-collection-kubernetes-core.noarch 2.3.2-10.fc42
f
ansible-collection-mdellweg-filters.noarch 0.0.6-2.fc42
f
```

Confidentiality: Public

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

f
ansible-collection-community-mysql.noarch 3.12.0-1.fc42
f
ansible-collection-community-postgresql.noarch 3.0.0-6.fc42
f
ansible-collection-community-rabbitmq.noarch 1.3.0-3.fc42
f
ansible-collection-containers-podman.noarch 1.16.3-7.fc42
f
ansible-collection-dellemc-openmanage.noarch 9.10.0-2.fc42
f
ansible-collection-kubernetes-core.noarch 2.3.2-10.fc42
f
ansible-collection-mdellweg-filters.noarch 0.0.6-2.fc42
f
ansible-collection-microsoft-sql.noarch 2.5.2-2.fc42
f
ansible-collection-pulp-pulp_installer.noarch 3.22.1-6.fc42
f
ansible-collection-pulp-pulp_installer-doc.noarch 3.22.1-6.fc42
f
ansible-collections-openstack.noarch 2.2.0-3.fc42
f
ansible@txlf1:~/ansible$
```

Confidentiality: Public

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

ansible@txlf1:~/ansible$ sudo dnf -y install ansible ansible-collections\*
Updating and loading repositories:
Repositories loaded.
Package Arch Version Reposito Size
Installing:
ansible noarch 11.6.0-1.fc42 updates 359.3 MiB
ansible-collections-openstack noarch 2.2.0-3.fc42 fedora 967.8 KiB
Installing dependencies:
ansible-core noarch 2.18.6-1.fc42 updates 13.9 MiB
git-core x86_64 2.51.0-2.fc42 updates 23.6 MiB
libtomcrypt x86_64 1.18.2-21.fc42 fedora 906.7 KiB
libtommath x86_64 1.3.1~rc1-5.fc42 fedora 130.4 KiB
python3-cryptography x86_64 44.0.0-3.fc42 fedora 5.0 MiB
python3-decorator noarch 5.1.1-14.fc42 fedora 78.5 KiB
python3-dogpile-cache noarch 1.3.3-2.fc42 fedora 568.6 KiB
python3-iso8601 noarch 2.1.0-3.fc42 fedora 50.8 KiB
python3-jinja2 noarch 3.1.6-1.fc42 fedora 2.9 MiB
python3-jsonpatch noarch 1.33-7.fc42 fedora 72.9 KiB
python3-jsonpointer noarch 2.4-4.fc42 fedora 45.3 KiB
python3-keystoneauth1 noarch 5.8.0-2.fc42 fedora 2.5 MiB
python3-mako noarch 1.2.3-9.fc42 fedora 701.2 KiB
python3-markupsafe x86_64 3.0.2-2.fc42 fedora 55.8 KiB
python3-netifaces x86_64 0.11.0-12.fc42 fedora 40.5 KiB
python3-openstacksdk noarch 4.0.0-2.fc42 fedora 5.6 MiB
```

Confidentiality: Public



```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

[10/31] Installing python3-paste-0:3.10 100% | 18.9 MiB/s | 2.7 MiB | 00m00s
[11/31] Installing libtommath-0:1.3.1~r 100% | 16.1 MiB/s | 131.5 KiB | 00m00s
[12/31] Installing libtomcrypt-0:1.18.2 100% | 42.2 MiB/s | 907.8 KiB | 00m00s
[13/31] Installing git-core-0:2.51.0-2. 100% | 101.1 MiB/s | 23.7 MiB | 00m00s
[14/31] Installing python3-pbr-0:6.0.0- 100% | 15.2 MiB/s | 639.8 KiB | 00m00s
[15/31] Installing python3-os-service-t 100% | 12.1 MiB/s | 124.3 KiB | 00m00s
[16/31] Installing python3-stevedore-0: 100% | 17.7 MiB/s | 325.6 KiB | 00m00s
[17/31] Installing python3-dogpile-cach 100% | 30.2 MiB/s | 587.4 KiB | 00m00s
[18/31] Installing python3-keystoneauth 100% | 35.9 MiB/s | 2.6 MiB | 00m00s
[19/31] Installing python3-resolvedlib-0 100% | 6.0 MiB/s | 98.8 KiB | 00m00s
[20/31] Installing ansible-core-0:2.18. 100% | 36.0 MiB/s | 14.4 MiB | 00m00s
[21/31] Installing python3-jsonpointer- 100% | 2.1 MiB/s | 48.1 KiB | 00m00s
[22/31] Installing python3-jsonpatch-0: 100% | 3.5 MiB/s | 75.6 KiB | 00m00s
[23/31] Installing python3-requestsexce 100% | 3.2 MiB/s | 19.9 KiB | 00m00s
[24/31] Installing python3-platformdirs 100% | 20.6 MiB/s | 168.4 KiB | 00m00s
[25/31] Installing python3-netifaces-0: 100% | 4.6 MiB/s | 42.6 KiB | 00m00s
[26/31] Installing python3-openstacksdk 100% | 29.5 MiB/s | 5.9 MiB | 00m00s
[27/31] Installing ansible-collections- 100% | 4.1 MiB/s | 993.9 KiB | 00m00s
[28/31] Installing ansible-0:11.6.0-1.f 100% | 39.4 MiB/s | 373.9 MiB | 00m09s
[29/31] Installing python3-crypto-0:2.6 100% | 7.4 MiB/s | 2.3 MiB | 00m00s
[30/31] Installing python3-beaker-0:1.1 100% | 10.9 MiB/s | 500.5 KiB | 00m00s
[31/31] Installing python3-libdnf5-0:5. 100% | 6.3 MiB/s | 9.2 MiB | 00m01s

Complete!
ansible@txlf1:~/ansible$
```

Confidentiality: Public



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

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
ansible@txlf1:~/ansible$ ansible-doc -l 2> /dev/null | wc -l
10632
ansible@txlf1:~/ansible$
```

Confidentiality: **Public**

You can also install collections from Ansible Galaxy

From the command line, use `ansible-galaxy`:

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

```
(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

Search

Collections

Namespaces

Roles

Documentation

Terms of Use

Collections

Keywords

Filter by keywords

Q

Name

12

11 - 20 of 3709



aci
Provided by [zjpeterperson](#)

Ansible collection for use with Cisco ACI, focused on the us...

0

0

0

2

Modules Roles Plugins Dependencies



aci
Provided by [jgomezve](#)

Ansible Plugin for EDA on ACI

0

0

0

1

Modules Roles Plugins Dependency



aci_fabric_manager
Provided by [lonemtntech](#)

provide interface to manage fabric policies in aci

0

0

0

1

Modules Roles Plugins Dependency



aci_fabric_policies
Provided by [lonemtntech](#)

provide interface to manage fabric policies in aci

0

0

0

1

Modules Roles Plugins Dependency



acklen_scripts
Provided by [acklendevloper](#)

Scripts used for automated pipelines in projects through

0

0

0

0

Modules Roles Plugins Dependencies



acklen_scripts
Provided by [ivandres73](#)

Scripts used for automated pipelines in projects through

0

0

0

0

Modules Roles Plugins Dependencies



acme
Provided by [t_systems_mms](#)

[DEPRECATED] This collection contains a role for issuing ssl...

0

1

0

4

Modules Role Plugins Dependencies



acme
Provided by [felixfontein](#)

Roles for ACME

0

3

1

6

Modules Roles Plugin Dependencies



acme
Provided by [telekom_mms](#)

This collection contains a role for issuing ssl certificates...

0

1

1

5

Modules Role Plugin Dependencies



acme
Provided by [rissson](#)

0

0

0

0

Modules Roles Plugins Dependencies

View more

→

11 - 20 of 3709

2 of 371

47

Requirements for Ansible

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

- ▶ Use ssh-keygen

```
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/Lbd13khD/zDVCr5vw2vjQUFNWhvF+JY/shA ansible@txlf1
The key's randomart image is:
+--[ED25519 256]--+
|                   .*=|
|                   ..0=|
|                  oo.|
|                 .  E  +o|
|                . + S . o o.o|
|               . o = . + = o.|
|              .  o .  = *. = .|
|             .  .o o= + **. .|
|            .... o..o o==+ |
+-----[SHA256]-----+
ansible@txlf1:~$
```

Confidentiality: **Public**

Requirements for Ansible

On the managed servers:

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

txlf2.aus.redhat.com on QEMU/KVM: molly

File Virtual Machine View Send Key

```
Fedora Linux 42 (Adams)
Kernel 6.16.8-200.fc42.x86_64 on x86_64 (tty1)

txlf2 login: root
Password:
Last login: Sun Sep 28 20:05:26 on tty1
root@txlf2:~# history
 1  grep wheel /etc/sudoers > /etc/sudoers.d/wheel
 2  vi /etc/sudoers.d/wheel
 3  useradd -G wheel ansible; echo MySecretPass1357 | passwd --stdin ansible
 4  dnf -y install python3
 5  history
root@txlf2:~# _
```

tiality: **Public**



Requirements for Ansible

On the management server:

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

```
Terminal - ansible@txlf1:~
File Edit View Terminal Tabs Help

ansible@txlf1:~$ ssh-copy-id txlf2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansible/.ssh/id_ed25519.pub"
The authenticity of host 'txlf2 (172.31.100.94)' can't be established.
ED25519 key fingerprint is SHA256:wQTCiUuIzBn9En8gDWICJ94kgsSrKPy0xDPUc7m2nK8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
ansible@txlf2's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'txlf2'"
and check to make sure that only the key(s) you wanted were added.

ansible@txlf1:~$
```

Confidentiality: **Public**

```
Terminal - ansible@txlf1:~
File Edit View Terminal Tabs Help

ansible@txlf1:~$ ssh-copy-id txlf3
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansible/.ssh/id_ed25519.pub"
The authenticity of host 'txlf3 (172.31.100.95)' can't be established.
ED25519 key fingerprint is SHA256:7RLKY+k70VrHs1NFDUkveCG8MTLhxVuIT+MPadgU9fA.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
ansible@txlf3's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'txlf3'"
and check to make sure that only the key(s) you wanted were added.

ansible@txlf1:~$
```

Confidentiality: **Public**

```
Terminal - ansible@txlf1:~
File Edit View Terminal Tabs Help

ansible@txlf1:~$ ssh-copy-id txlf4
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansible/.ssh/id_ed25519.pub"
The authenticity of host 'txlf4 (172.31.100.96)' can't be established.
ED25519 key fingerprint is SHA256:qTkPC1h5e4jq3ah2JmXMWzwmFudpRa4M+ivpEiMHVhg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
ansible@txlf4's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'txlf4'"
and check to make sure that only the key(s) you wanted were added.

ansible@txlf1:~$
```

Confidentiality: **Public**

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
```

```
Terminal - root@txlf2:~
File Edit View Terminal Tabs Help
root@txlf2:~# echo Match User ansible > /etc/ssh/sshd_config.d/99-ansiblepw.conf
root@txlf2:~# echo -e "\tPasswordAuthentication no" >> /etc/ssh/sshd_config.d/99-ansiblepw.conf
root@txlf2:~# cat /etc/ssh/sshd_config.d/99-ansiblepw.conf
Match User ansible
        PasswordAuthentication no
root@txlf2:~# systemctl restart sshd
root@txlf2:~#
```

Confidentiality: **Public**

```
Terminal - tcameron@case:~/Desktop
File Edit View Terminal Tabs Help

tcameron@case:~/Desktop$ ssh tcameron@txlf2
Warning: Permanently added 'txlf2' (ED25519) to the list of known hosts.
tcameron@txlf2's password:
tcameron@txlf2:~$
logout
Connection to txlf2 closed.
tcameron@case:~/Desktop$ ssh ansible@txlf2
Warning: Permanently added 'txlf2' (ED25519) to the list of known hosts.
ansible@txlf2: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
tcameron@case:~/Desktop$
```

Confidentiality: **Public**

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`

THEMARVJTHOMPSON

Confidentiality: **Public**



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

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.

```
Terminal - ansible@txlf1:~
File Edit View Terminal Tabs Help
ansible@txlf1:~$ cat /etc/ansible/hosts
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
# - Comments begin with the '#' character
# - Blank lines are ignored
# - Groups of hosts are delimited by [header] elements
# - You can enter hostnames or ip addresses
# - A hostname/ip can be a member of multiple groups
#
# Ex 1: Ungrouped hosts, specify before any group headers:
## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10
#
# Ex 2: A collection of hosts belonging to the 'webservers' group:
## [webservers]
## alpha.example.org
## beta.example.org
## 192.168.1.100
```

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
ansible@txlf1:~$ mkdir ansible
ansible@txlf1:~$ cd ansible/
ansible@txlf1:~/ansible$ vi inventory.ini
```

Confidentiality: **Public**

Confidentiality: **Public**

--	--

5,1

All

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`

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`

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.**

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
ansible@txlf1:~/ansible$ ansible-config init > newansible.cfg
ansible@txlf1:~/ansible$ grep -Ev ";|#" newansible.cfg | uniq | wc -l
411
ansible@txlf1:~/ansible$
```

Confidentiality: **Public**

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`

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
ansible@txlf1:~/ansible$ ansible-config init --disabled > ansible.cfg
ansible@txlf1:~/ansible$
```

Confidentiality: **Public**

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?

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

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
ansible@txlf1:~/ansible$ grep -v ";" ansible.cfg | grep -v "#" | uniq
[defaults]

ask_pass=False

forks=5

inventory=./inventory.ini

remote_user=ansible

[privilege_escalation]

become=true

become_method=sudo

become_user=root

[persistent_connection]

[connection]

[colors]
```

Confidentiality: **Public**

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 `*.yaml` or `*.yml`.
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

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
ansible@txlf1:~/ansible$ sudo dnf -y install vim-enhanced vim-ansible
```

Confidentiality: **Public**

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

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

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}`

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:

```
• ---
```


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

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

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

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

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

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.

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.

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

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
---
- name: Demo playbook for TXLF 2025
  hosts: web
  gather_facts: false
  become: true
  tasks:
    - name: Run the first play of the playbook

```

Confidentiality: **Public**



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

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?

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.

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
---
- name: Demo playbook for TXLF 2025
  hosts: web
  gather_facts: false
  become: true
  tasks:
    - name: Run the first play of the playbook
      ansible.builtin.ping:

```

Confidentiality: **Public**



```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
---
- name: Demo playbook for TXLF 2025
  hosts: web
  gather_facts: false
  become: true
  tasks:
    - name: Run the first play of the playbook
      ping:

```

Confidentiality: **Public**



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

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

ansible@txlf1:~/ansible$ ansible-playbook myplaybook.yml --syntax-check
ERROR! couldn't resolve module/action 'ansible.builtin.ping'. This often indicates a misspelling, missing collection, or incorrect module path.

The error appears to be in '/home/ansible/ansible/myplaybook.yml': line 7, column 7, but may
be elsewhere in the file depending on the exact syntax problem.

The offending line appears to be:

    tasks:
      - name: Run the first play of the playbook
        ^ here
ansible@txlf1:~/ansible$
```

Confidentiality: **Public**

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

---
- name: Demo playbook for TXLF 2025
  hosts: web
  gather_facts: false
  become: true
  tasks:
    - name: Run the first play of the playbook
      ansible.builtin.ping:

```

Confidentiality: **Public**

-- INSERT --

8,29

All




```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
ansible@txlf1:~/ansible$ ansible-playbook myplaybook.yml --syntax-check

playbook: myplaybook.yml
ansible@txlf1:~/ansible$
```

Confidentiality: **Public**

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

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

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

PLAY [Demo playbook for TXLF 2025] *****

TASK [Run the first play of the playbook] *****
[WARNING]: Platform linux on host txlf2.aus.redhat.com is using the discovered
Python interpreter at /usr/bin/python3.13, but future installation of another
Python interpreter could change the meaning of that path. See
https://docs.ansible.com/ansible-
core/2.18/reference\_appendices/interpreter\_discovery.html for more information.
ok: [txlf2.aus.redhat.com]

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

ansible@txlf1:~/ansible$
```

Confidentiality: Public

Using Ansible ping

Time to run the playbook!

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

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

# (string) Path to the Python interpreter to be used for module execution on remote targets, or an automatic discovery mode. Supported discovery modes are ``auto`` (the default), ``auto_silent``, ``auto_legacy``, and ``auto_legacy_silent``. All discovery modes employ a lookup table to use the included system Python (on distributions known to include one), falling back to a fixed ordered list of well-known Python interpreter locations if a platform-specific default is not available. The fallback behavior will issue a warning that the interpreter should be set explicitly (since interpreters installed later may change which one is used). This warning behavior can be disabled by setting ``auto_silent`` or ``auto_legacy_silent``. The value of ``auto_legacy`` provides all the same behavior, but for backward-compatibility with older Ansible releases that always defaulted to ``/usr/bin/python``, will use that interpreter if present.
;interpreter_python=auto
interpreter_python=auto_silent

# (boolean) If 'false', invalid attributes for a task will result in warnings instead of errors.
;invalid_task_attribute_failed=True

# (boolean) By default, Ansible will issue a warning when there are no hosts in the inventory.
# These warnings can be silenced by adjusting this setting to False.
;localhost_warning=True
```

Confidentiality: Public



```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

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

PLAY [Demo playbook for TXLF 2025] *****

TASK [Run the first play of the playbook] *****
ok: [txlf2.aus.redhat.com]

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

ansible@txlf1:~/ansible$
```

Confidentiality: **Public**

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

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?

Pop Quiz!

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

- ▶ Install the software

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

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

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

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

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

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
ansible@txlf1:~/ansible$ ansible-doc ansible.builtin.package
```

Confidentiality: **Public**

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
> MODULE ansible.builtin.package (/usr/lib/python3.13/site-packages/ansible/mod>

This module 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

(
```

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

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
ansible@txlf1:~/ansible$ ansible-doc ansible.builtin.dnf
```

Confidentiality: **Public**

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
> MODULE ansible.builtin.dnf (/usr/lib/python3.13/site-packages/ansible/modules>

Installs, upgrade, removes, and lists packages and groups with the
dnf package manager.

* note: This module has a corresponding action plugin.

OPTIONS (red indicates it is required):

    allow_downgrade Specify if the named package and version is
                        allowed to downgrade a maybe already installed
                        higher version of that package. Note that setting
                        `allow_downgrade=true` can make this module behave
                        in a non-idempotent way. The task could end up
                        with a set of packages that does not match the
                        complete list of specified packages to install
                        (because dependencies between the downgraded
                        package and others can cause changes to the
                        packages which were in the earlier transaction).
                        default: 'no'
                        type: bool

    allow_erasing If `true` it allows erasing of installed packages to
: |
```

```
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

EXAMPLES:

- name: Install the latest version of Apache
  ansible.builtin.dnf:
    name: httpd
    state: latest

- name: Install Apache >= 2.4
  ansible.builtin.dnf:
    name: httpd >= 2.4
    state: present

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

- name: Remove the Apache package
  ansible.builtin.dnf:
    name: httpd
    state: absent
```

```
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-
    state: present
:
```

Confidentiality: **Public**

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

- name: Install nginx rpm from a local file
  ansible.builtin.dnf:
    name: /usr/local/src/nginx-release-centos-6-0.el6ngx.noarch.rpm
    state: present

- name: Install Package based upon the file it provides
  ansible.builtin.dnf:
    name: /usr/bin/cowsay
    state: present

- name: Install the 'Development tools' package group
  ansible.builtin.dnf:
    name: '@Development tools'
    state: present

- name: Autoremove unneeded packages installed as dependencies
  ansible.builtin.dnf:
    autoremove: yes

- name: Uninstall httpd but keep its dependencies
  ansible.builtin.dnf:
    name: httpd
    state: absent

:
```

```
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

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
ansible@txlf1:~/ansible$ vi install-httpd.yml
```

Confidentiality: **Public**

~~~~~

All

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
ansible@txlf1:~/ansible$ vi install-httpd.yml
ansible@txlf1:~/ansible$ ansible-playbook install-httpd.yml --syntax-check

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

Confidentiality: **Public**

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

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] *****
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$
```

Confidentiality: **Public**

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
ansible@txlf1:~/ansible$ vi install-mod_ssl.yml
```

Confidentiality: **Public**

~~~~~

All

```
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$
```

Confidentiality: **Public**

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.

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.

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
ansible@txlf1:~/ansible$ ansible-doc ansible.builtin.service
```

Confidentiality: **Public**

```
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

OPTIONS (red indicates it is required):

arguments Additional arguments provided on the command line.
           While using remote hosts with systemd this setting will
           be ignored.
           aliases: [args]
           default: ''
           type: str

enabled Whether the service should start on boot.
         At least one of `state` and `enabled` are required.
         default: null
         type: bool

name Name of the service.
   type: str

pattern If the service does not respond to the status command,
        name a substring to look for as would be found in the
        output of the `ps` command as a stand-in for a
        status result.
        If the string is found, the service will be assumed to be
        started.
```

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

EXAMPLES:
- name: Start service httpd, if not started
  ansible.builtin.service:
    name: httpd
    state: started

- name: Stop service httpd, if started
  ansible.builtin.service:
    name: httpd
    state: stopped

- name: Restart service httpd, in all cases
  ansible.builtin.service:
    name: httpd
    state: restarted

- 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:
    state: enabled
```

```
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
ansible@txlf1:~/ansible$ ansible-doc ansible.builtin.systemd_service
```

Confidentiality: **Public**

```
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

:
```

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
ansible@txlf1:~/ansible$ vi enable-httpd.yml
```

Confidentiality: **Public**

~~~~~

All

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

ansible@txlf1:~/ansible$ ansible-playbook enable-httpd.yml

PLAY [Playbook to enable the httpd service] *****

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

TASK [Task to enable 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$
```

Confidentiality: **Public**

# 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!

# 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`

# 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!

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
ansible@txlf1:~/ansible$ vi start-httpd.yml
```

Confidentiality: **Public**

[illegible]

All

```
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$
```

Confidentiality: **Public**

---

# Automating Firewall Rules

How to open ports

# 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

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

ansible@txlf1:~/ansible$ sudo firewall-cmd --get-services | grep http
0-AD RH-Satellite-6 RH-Satellite-6-capsule afp alvr amanda-client amanda-k5-client amqp amqps anno-1602 anno-1800 apcupsd aseqnet audit ausweisapp2 bacula bacula-client bareos-director bareos-filedaemon bareos-storage bb bgp bitcoin bitcoin-rpc bitcoin-testnet bitcoin-testnet-rpc bittorrent-lsd ceph ceph-exporter ceph-mon cfengine checkmk-agent civilization-iv civilization-v cockpit collectd condor-collector cratedb ctdb dds dds-multicast dds-unicast dhcp dhcpv6 dhcpv6-client distcc dns dns-over-quic dns-over-tls docker-registry docker-swarm dropbox-lansync elasticsearch etcd-client etcd-server factorio finger foreman foreman-proxy freeipa-4 freeipa-ldap freeipa-ldaps freeipa-replication freeipa-trust ftp galeria ganglia-client ganglia-master git gpsd grafana gre high-availability http http3 https ident imap imaps iperf2 iperf3 ipfs ipp ipp-client ipsec irc ircs iscsi-target isns jenkins kadmin kdeconnect kerberos kibana klogin kpasswd kprop kshell kube-api kube-apiserver kube-control-plane kube-control-plane-secure kube-controller-manager kube-controller-manager-secure kube-nodeport-services kube-scheduler kube-scheduler-secure kube-worker kubelet kubelet-readonly kubelet-worker ldap ldaps libvirt libvirt-tls lightning-network llmnr llmnr-client llmnr-tcp llmnr-udp managesieve matrix mdns memcache minecraft minidlna mndp mongodb mosh mountd mpd mqtt mqtt-tls ms-wbt mssql murmur mysql nbd nebula need-for-speed-most-wanted netbios-ns netdata-dashboard nfs nfs3 nmea-0183 nrpe ntp nut opentelemetry openvpn ovirt-imageio ovirt-storageconsole ovirt-vmconsole plex pmcd pmproxy pmwebapi pmwebapis pop3 pop3s postgresql privoxy prometheus prometheus-node-exporter proxy-dhcp ps2link ps3netsrv ptp pulseaudio puppetmaster quassel radius radsec rdp redis redis-sentinel rootd rpc-bind rquotad rsh rsyncd rtsp salt-master sam
```

Confidentiality: Public

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

3 https ident imap imaps iperf2 iperf3 ipfs ipp ipp-client ipsec irc ircs iscsi-
target isns jenkins kadmin kdeconnect kerberos kibana klogin kpasswd kprop kshel
l kube-api kube-apiserver kube-control-plane kube-control-plane-secure kube-cont
roller-manager kube-controller-manager-secure kube-nodeport-services kube-schedu
ler kube-scheduler-secure kube-worker kubelet kubelet-readonly kubelet-worker ld
ap ldaps libvirt libvirt-tls lightning-network llmnr llmnr-client llmnr-tcp llmn
r-udp managesieve matrix mdns memcache minecraft minidlna mndp mongodb mosh moun
td mpd mqtt mqtt-tls ms-wbt mssql murmur mysql nbd nebula need-for-speed-most-wa
nted netbios-ns netdata-dashboard nfs nfs3 nmea-0183 nrpe ntp nut opentelemetry
openvpn ovirt-imageio ovirt-storageconsole ovirt-vmconsole plex pmcd pmproxy pmw
ebapi pmwebapis pop3 pop3s postgresql privoxy prometheus prometheus-node-exporte
r proxy-dhcp ps2link ps3netsrv ptp pulseaudio puppetmaster quassel radius radsec
rdp redis redis-sentinel rootd rpc-bind rquotad rsh rsyncd rtsp salt-master sam
ba samba-client samba-dc sane settlers-history-collection sip sips slimevr slp s
mtp smtp-submission smtps snmp snmptls snmptls-trap snmptrap spideroak-lansync s
potify-sync squid ssdp ssh statsrv steam-lan-transfer steam-streaming stellaris
stronghold-crusader stun stuns submission supertuxkart svdrp svn syncthing synct
hing-gui syncthing-relay synergy syscomlan syslog syslog-tls telnet tentacle ter
raria tftp tile38 tinc tor-socks transmission-client turn turns upnp-client vds
vnc-server vrrp warpinator wbem-http wbem-https wireguard ws-discovery ws-disco
very-client ws-discovery-host ws-discovery-tcp ws-discovery-udp wsman wsmans xdm
cp xmpp-bosh xmpp-client xmpp-local xmpp-server zabbix-agent zabbix-java-gateway
zabbix-server zabbix-trapper zabbix-web-service zero-k zerotier
ansible@txlf1:~/ansible$
```

Confidentiality: Public



```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
ansible@txlf1:~/ansible$ ansible-doc ansible.posix.firewalld
```

Confidentiality: **Public**

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
> MODULE ansible.posix.firewalld (/usr/lib/python3.13/site-packages/ansible_col>

This module allows for addition or deletion of services and ports
(either TCP or UDP) in either running or permanent firewalld rules.

OPTIONS (red indicates it is required):

forward The forward setting you would like to enable/disable
to/from zones within firewalld.
This option only is supported by firewalld v0.9.0 or
later.
default: null
type: str

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

icmp_block_inversion Enable/Disable inversion of ICMP blocks for a
zone in firewalld.
default: null
type: str
```

```
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%%']
```

Confidentiality: **Public**

```
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:
  :
```

Confidentiality: **Public**

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

- ansible.posix.firewalld:
  source: 192.0.2.0/24
  zone: internal
  state: enabled

- ansible.posix.firewalld:
  zone: trusted
  interface: eth2
  permanent: true
  state: enabled

- ansible.posix.firewalld:
  forward: true
  state: enabled
  permanent: true
  zone: internal

- ansible.posix.firewalld:
  masquerade: true
  state: enabled
  permanent: true
  zone: dmz
```

Confidentiality: **Public**

# 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

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
ansible@txlf1:~/ansible$ vi firewall.yml
```

Confidentiality: **Public**

$\frac{1}{\sqrt{2}}$

10,24

All

~~~~~

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

```
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
    - name: Task to enable access via ssh
      ansible.posix.firewalld:
        port: 22/tcp
        state: enabled
        permanent: true
        immediate: true

~
-- INSERT --
```

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

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

PLAY [Playbook to open firewall ports for the web server] *****

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

TASK [Task to allow access to the http service] *****
changed: [txlf2.aus.redhat.com]

TASK [Task to allow access to the https service] *****
changed: [txlf2.aus.redhat.com]

TASK [Task to enable access via ssh] *****
changed: [txlf2.aus.redhat.com]

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

ansible@txlf1:~/ansible$
```

Confidentiality: **Public**

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

```
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$
```

Confidentiality: Public

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

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

PLAY [Playbook to open firewall ports for the web server] *****

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

TASK [Task to allow access to the http service] *****
ok: [txlf2.aus.redhat.com]

TASK [Task to allow access to the https service] *****
ok: [txlf2.aus.redhat.com]

TASK [Task to enable access via ssh] *****
ok: [txlf2.aus.redhat.com]

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

ansible@txlf1:~/ansible$
```

Confidentiality: **Public**

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

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
ansible@txlf1:~/ansible$ vi combined-playbook.yml
```

Confidentiality: **Public**

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

---
- name: Playbook to install Apache, enable and start, and configure firewall
  hosts: web
  tasks:
    - name: Play to install httpd and mod_ssl
      ansible.builtin.dnf:
        name:
          - httpd
          - mod_ssl
        state: latest

-- INSERT --
```

Confidentiality: Public

```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help
---
- name: Playbook to install Apache, enable and start, and configure firewall
  hosts: web
  tasks:
    - name: Play to install httpd and mod_ssl
      ansible.builtin.dnf:
        name:
          - httpd
          - mod_ssl
        state: latest
    - name: Play to enable and start the httpd service
      ansible.builtin.systemd_service:
        name: httpd
        enabled: true
        state: started
~
~
~
~
~
~
~
~
~
~
-- INSERT --
```

Confidentiality: Public



```
Terminal - ansible@txlf1:~/ansible
File Edit View Terminal Tabs Help

- name: Play to enable and start the httpd service
  ansible.builtin.systemd_service:
    name: httpd
    enabled: true
    state: started
- name: Play to enable http, https, and ssh firewalld services
  ansible.posix.firewalld:
    service: "{{ item }}"
    state: enabled
    permanent: true
    immediate: true
  loop:
    - http
    - https
    - ssh

-- INSERT --
```

Confidentiality: Public

```
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 firewall 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
skipped=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