Caltech | Center for Technology & Management Education

# Post Graduate Program in DevOps

Powered by simplilearn

Caltech | Center for Technology & Management Education

# Configuration Management with Ansible and Terraform

Powerd by simplilearn

# Ansible Configuration

# A Day in the Life of a DevOps Engineer

You are working as a Senior DevOps Engineer in an organization where recently a few freshers joined your team. The manager of your team requested you to demonstrate different ways for configuring Ansible.

You are requested to include the following important information to pass on to the new employees:

- What is the option they have to govern the behavior of all interactions performed by the Ansible Server?
- What can be used to configure settings in Ansible?
- What determines the order of precedence within the playbook keyword?
- Where are the hostnames or private IP addresses of all the nodes or hosts connected with the ansible server stored?

# A Day in the Life of a DevOps Engineer

- What are the nodes connected with the Ansible server?
- What types of formats are supported in Ansible environments?
- What does Ansible use to communicate with distant machines?

To achieve all the above, along with some additional features, you will be learning a few concepts in this lesson that will help find a solution for the given scenario.

# Learning Objectives

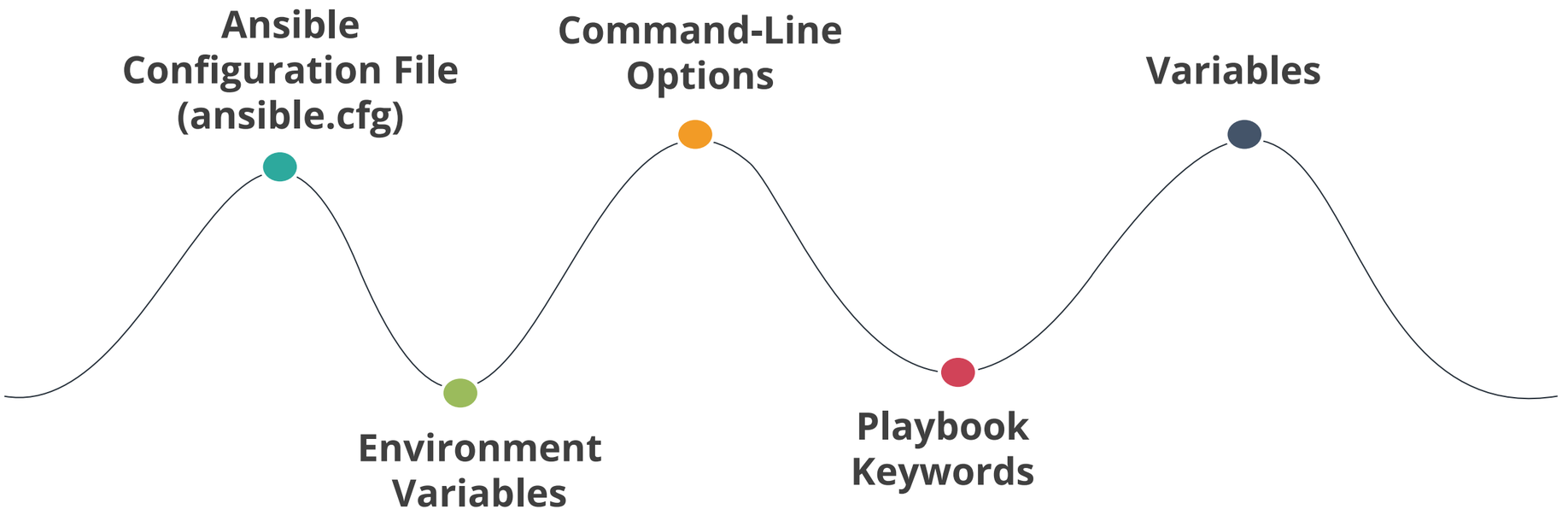
By the end of this lesson, you will be able to:

- Configure ansible server using different ways
- Create Ansible inventory
- Define hosts & groups and select them using patterns
- Setup a remote connection between ansible server and other nodes (hosts and groups)

# How to Configure Ansible?

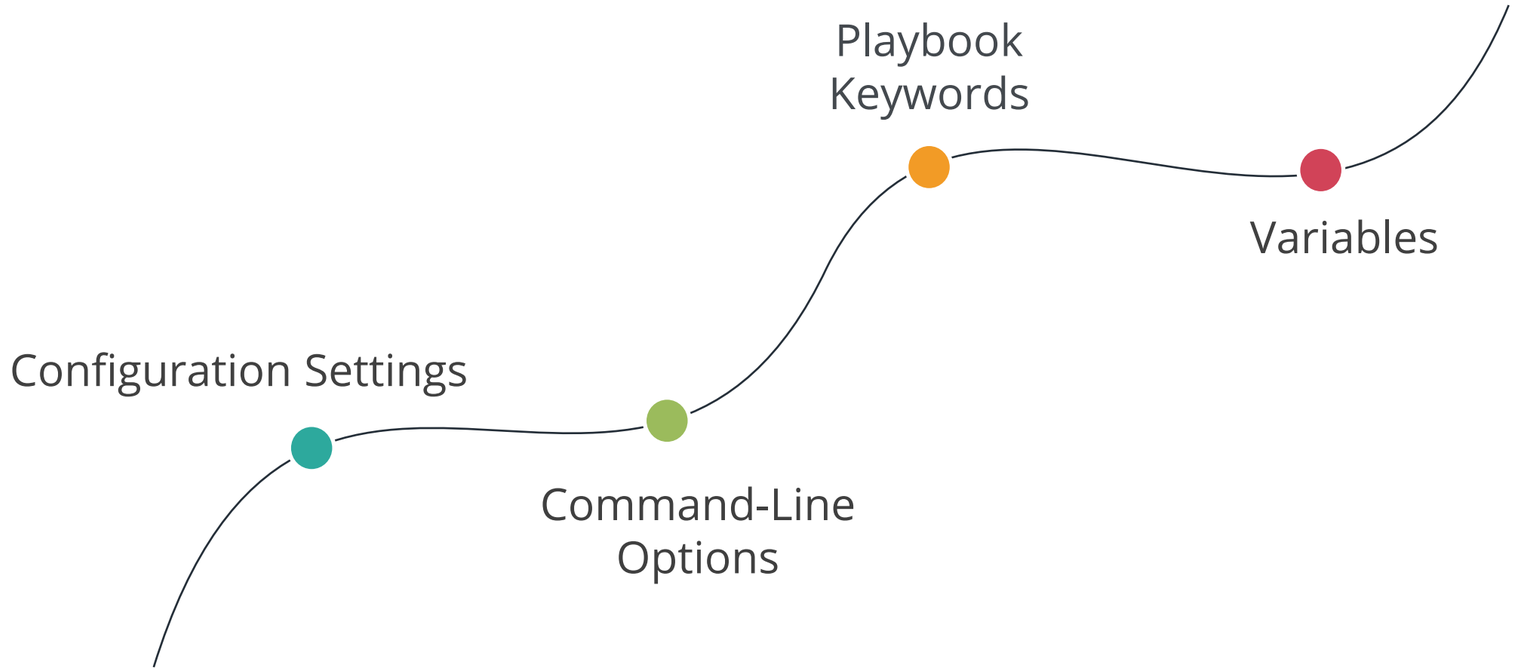
# Ways to Configure Ansible

Ansible supports several ways for configuring its behavior, which are as follows:



# Precedence Rules

Ansible has four options for infrastructure configuration. The options are listed in order of precedence from lowest (most easily overridden) to highest (overrides all others):



# Configuration Settings

Configuration settings include both values from the ansible.cfg file and environment variables.

01 The values set in configuration files have a lower priority in this category.

02 Ansible ignores all other ansible.cfg files and utilizes the first one it finds.

# Understanding Ansible Configuration File

# Ansible Configuration File

Ansible configuration file or ansible.cfg governs the behaviour of all interactions performed by the Ansible Server.

01 Ansible comes with a default configuration file when a user first installs it.

02 The default configuration file for Ansible is ansible.cfg, which may be found in **/etc/ansible/ansible.cfg.**

# Ansible Configuration File

Ansible configuration file (ansible.cfg) example:

```
# nearly all parameters can be overridden in ansible-playbook
# or with command line flags. ansible will read ANSIBLE_CONFIG,
# ansible.cfg in the current working directory, .ansible.cfg in
# the home directory or /etc/ansible/ansible.cfg, whichever it
# finds first
[defaults]
# it some basic default values...
#inventory = /etc/ansible/hosts flibrary = /usr/share/my_modules/
#moduleutils = iusr/share/mymodule_utils/ lremote_tmp - -/.ansible/tmp
#local_tmp = -/.ansible/tmp
#plugin_filters_cfg = /etc/ansibleiplugin_filters.yml
#forks = 5
#poll_interval = 15
#sudo user = root
#ask_sudo_pass = True
#ask_pass = True
#transport = smart
#remote_port = 22
#module_lang = C
#modulesetlocale = False
"/etc/ansible/ansible.cfq" 490L, 19985C
```

An Ansible configuration file uses an INI format to store its configuration data.

# Ansible Configuration File

Ansible configuration file is divided into ten different sections as shown below:

```
 [loaner@loaner ansibleJ$ grep '^\[' ansible.cfg | nl

1 [defaults]
2 [inventory]
3 [privilege_escalation]
4 [paramiko_connection]
5 [ssh_connection]
6 [persistent_connection]
7 [accelerate]
8 [selinux]
9 [cotors]
10 [diff]

[loaner@loaner ansible]S cat ansible.cfg | we -l
 490
```

# Ansible Configuration File

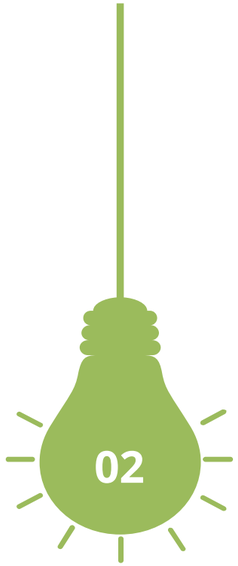
01 Each section denoted within the square brackets gives users an idea about this massive configuration file.

02 It is highly recommended that user should browse through the default configuration file to see the different settings that are available to them.

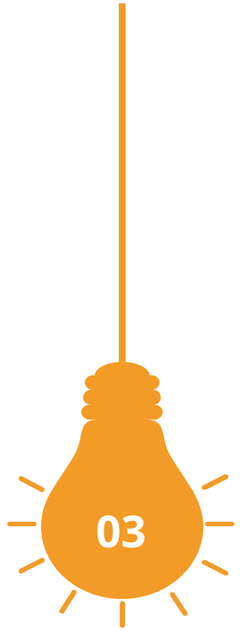
# Configuration File Locations

Ansible looks for the ansible.cfg file in the following locations, in this order:

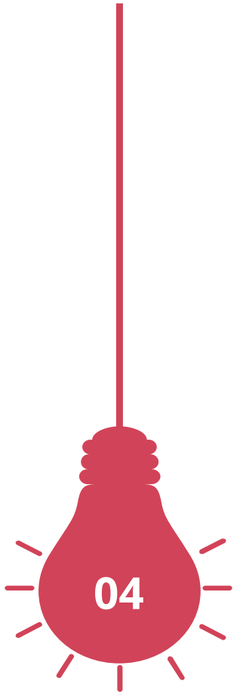

ANSIBLE_CONFIG (if Environment Variable is set)




ansible.cfg (in the current directory)




~/.ansible.cfg (in the home directory)

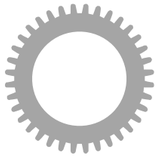



/etc/ansible/ansible.cfg

# Configuration File Installation Options

The latest configurations are given below:

If the user uses package management to install ansible, the most recent ansible.cfg should be in **/etc/ansible**, maybe as a ".rpmnew" file (or something else) if it is updated.



If the users have installed Ansible from pip or source, then they can create this file to override default settings in Ansible.

# Environmental Configurations

Environment variables can be used to configure settings in Ansible. If these environment variables are set, they will override any configuration file settings.

# Command Line Options

Important points regarding command line options are:

01 Only the most useful and popular configuration options are available through the command line.

02 The settings in the command line take precedence over those in the configuration file and environment.

# Playbook Keywords

The playbook determines the order of precedence within playbook keywords. The more precise keywords take precedence over the more general:

01 play (most general)

02 Blocks, includes, imports, or roles (optional and can contain tasks and each other)

03 tasks (most specific)

# Playbook Keywords

Example of configuring ansible using playbook keywords:

```
- hosts: all
  connection: ssh
  tasks:
     - name: This task uses ssh.
       ping:

       - name: This task uses ssh.
      connection: paramiko
       ping:
```

Source: www.docs.ansible.com

# Variables

Any variable can override any playbook keyword, command-line option, or configuration setting.

01 Connection variables are variables that have comparable playbook keywords, command-line options, and configuration settings.

02 Tasks (most specific) are the component of a playbook.

# Variables

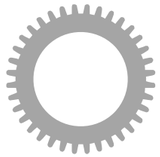
Example of configuring ansible using variables:

```
-  hosts: cloud
gather_facts:false
 become: yes
 vars:
    ansible_become_user: admin
 tasks:
   - name: This task uses admin as the become user.
   dnf:
     name: some-service
     state: latest
   - block:
       - name: This task uses admin as the become user.
       - name: This task uses admin as the become user.
     vars:
       ansible_become_user:service-admin
   - name: This task (outside of the block) uses admin as the become user.
     service:
       name: some-service
       state: restarted
```

# Variables

Use of **-e** extra variables on the command line

Extra variables can be used to override all other parameters in all other categories: on the command line —extra-vars or -e.

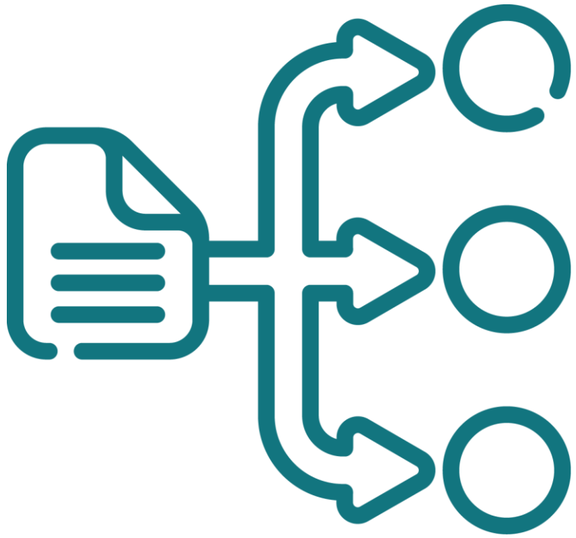


Values passed with **-e** are variables, not command-line options. They override configuration settings, command-line options, and playbook keywords as well as variables set elsewhere.

# Managing Ansible Inventory

# Ansible Inventory

Ansible Inventory file stores the hostnames or private IP address of all the nodes or hosts connected with the ansible server.



The default location for Ansible inventory file is **/etc/ansible/hosts**

# Ansible Inventory

Example of an inventory file in INI format:

```
 mail.example.com


[webservers]
foo.example.com
bar.example.com

[Observers]
one.example.com
two.example.com
three.example.com
```

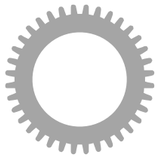
# Ansible Inventory

Example of an inventory file in YAML format:

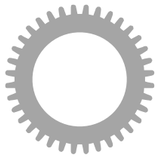
```
all:
  hosts:
       mail.example.com:
   children:
       webservers:
            hosts:
                foo.example.com:
                bar.example.com:
         dbservers:
              hosts:
                one.example.com:
                two.example.com:
               three.example.com:
```

# Ansible Inventory

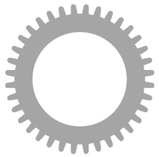
Key points regarding ansible inventory are:



The place where hosts or groups are defined is known as inventory, Ansible runs against various managed nodes or "hosts" in the infrastructure at once.



After defining the inventory users can use patterns to select the hosts or groups against which they want Ansible to run.



Users can specify a different inventory file at the command line using the -i <path> option.

# Types of Ansible Inventory

There are two types of Ansible Inventory:

**Static Inventory**

A static inventory file is a plain text file that contains a list of managed hosts defined by hostnames or IP addresses under a host group.

**Dynamic Inventory**

A script developed in any high-level programming language is known as a dynamic inventory. It's useful in cloud setups like AWS, where virtual server IP addresses change when they're stopped and restarted.

# Ansible Inventory

Example of a Static inventory file in YAML:

```
[webservers]
 173.82.115.165

[database_servers]
173.82.220.239

[datacenter:children]
webservers
database_servers
```

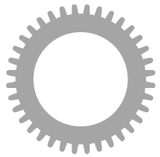
***Note***

The use of managed hosts in a host group is optional. Users can just use their hostnames or IP addresses to make a list of them.

# Host and Groups

Host and Groups are the nodes connected with the Ansible server.

The Host is the single node connected with the ansible server.



Groups are the collection of hosts and are named as a single entity.

# Adding Variables to Inventory

01 Inventory stores variable values relating to a specific host or group.

02 Inventory can contain variable values linked to a specific host or group.

03 Users can add additional managed nodes to their Ansible inventory, then they will probably want to keep variables in distinct host and group variable files.

# Adding Variables to Inventory

Example of assigning a variable to one machine: host variables

```
 [atlanta]
host1 http_port=80 maxRequestsPerChild=808
host2 http_port=303maxRequestsPerChild=909
```

In INI format

# Adding Variables to Inventory

Example of assigning a variable to one machine: host variables

```
atlanta:
    hosts:
        hosts1:
            http_port  80
            maxRequestsPerChild: 808
         host2:
            http_port: 303
            maxRequestsPerChild: 909
```

In YAML format

# Adding Variables to Inventory

Example of assigning a variable to one machine: group variables

```
[atlanta]
 host1
 host2

[atlanta:vars]
 ntp_server=ntp.atlanta.example.com
 proxy=proxy.atlanta.example.com
```

In INI format

# Adding Variables to Inventory

Example of assigning a variable to one machine: group variables

```
 atlanta:
     hosts:
          host1:
          host2:

     vars:
          ntp_server:
ntp.atlanta.example.com
          proxy: proxy.atlanta.example.com
```

In YAML format

# Organizing Host and Group Variables

01 Variables can be stored in the main inventory file, keeping separate host and group variables files may make it easier to arrange the variable values.

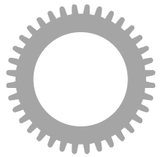
02 YAML syntax is required for host and group variable files.

03 A file can have extensions such as ‘.yml', ‘.yaml', and ‘.json' or no extensions at all. Both file types are valid.

# Patterns in Ansible

Patterns are a set of expressions in Ansible that lets user specify which node a playbook or an ad hoc command must be applied to.

Ansible pattern can be used to refer to a single host, an IP address, an inventory group, a collection of inventory groups, or all hosts in your inventory.



Commands and playbooks can be run against specific hosts and/or groups in your inventory using patterns.

# Patterns in Ansible

Patterns can be used almost any time user executes an ad hoc command or a playbook. The pattern is the only element of an ad hoc command that has no flag and usually it is the second element

**Syntax**

```
ansible <pattern> -m <name_of_module> -a "<module options>"
```

**Example**

```
ansible dbservers[0] -m service -a "name=httpd state=restarted"
```

***Note***

Here dbservers is an arbitrary group specified in the ansible server where dbservers[0] is the first host of that group

# Common Patterns

Common patterns for targeting inventory hosts and groups

| Description | Pattern(s) | Targets |
|---|---|---|
| All hosts | all (or *) | |
| One host | host1 | |
| Multiple hosts | host1:host2 (or host1,host2) | |
| One group | webservers | |
| Multiple groups | webservers:dbservers | all hosts in webservers plus all hosts in dbservers |
| Excluding groups | webservers:!atlanta | all hosts in webservers except those in atlanta |
| Intersection of groups | webservers:&staging | any hosts in webservers that are also in staging |

***Note***

User can use either a comma (,) or a colon (:) to separate a list of hosts

# Advanced Pattern Options

**Using group position in patterns**

User can define a host or subset of hosts by its position in the group.

Eample of a group

```
[dbservers]

localhost:42006
10.4.0.1
10.4.0.2
```

Selecting hosts by using their group position and range

```
dbservers[0]        # == localhost

dbservers[-1]       # ==10.4.0.2

dbservers[0:2]      # == localhost, 10.4.0.1, 10.4.0.2

dbservers[0],dbservers[1] # == localhost,10.4.0.1

dbservers[1:]       # == 10.4.0.1, 10.4.0.2

dbservers[:3]       # == localhost, 10.4.0.1, 10.4.0.2
```

# Assisted Practice

## Creating Static Host Inventory

**Duration: 15 Min.**

**Problem Statement:**

You've been assigned a task to create a static host inventory which will allow you to add nodes which can be then managed by ansible server.

# Assisted Practice: Guidelines

**Steps to be followed:**

1. Install Ansible
2. Generate ssh key on ansible server
3. Establish SSH connection between ansible server and the nodes
4. Add groups and hosts in the inventory file present in ansible server
5. Check the connectivity between ansible server and the nodes specified in inventory file

# Inventory File Formats

# INI and YAML

Ansible environment supports two types of formats.

01 INI

02 YAML

# INI and YAML

Example of Inventory in INI format

```
server1.example.com
server1.example.com
[linux]
server3.example.com
server4.example.com
[windows]
server5.example.com
server6.example.com
```

Source www.freekb.net

# INI and YAML

Example of Inventory in YAML format

```
linux:
     hosts:
           server1.example.com:
           server2.example.com:
           server3.example.com:
 windows:
      hosts:
           server1.example.com:
           server2.example.com:
           server3.example.com:
```

Source www.freekb.net

# Assisted Practice

## Host and Groups Configuration

**Duration: 15 Min.**

**Problem Statement:**

You've been assigned a task to configure hosts individually and by adding them in a group, making groups makes the job of DevOps easier in terms of configuring them

# Assisted Practice: Guidelines

**Steps to be followed:**

1. Install Ansible on the node.
2. Generate SSH key on the ansible server.
3. Copy the SSH key on the hosts specified in Ansible Inventory.
4. Establish SSH connectivity between Ansible Server and the nodes specified in inventory file.
5. Check the connectivity by using ping command.
6. Add/delete files on the hosts or groups using ansible.
7. Update the hosts and groups using ansible.

# Managing Ansible Remote Connections

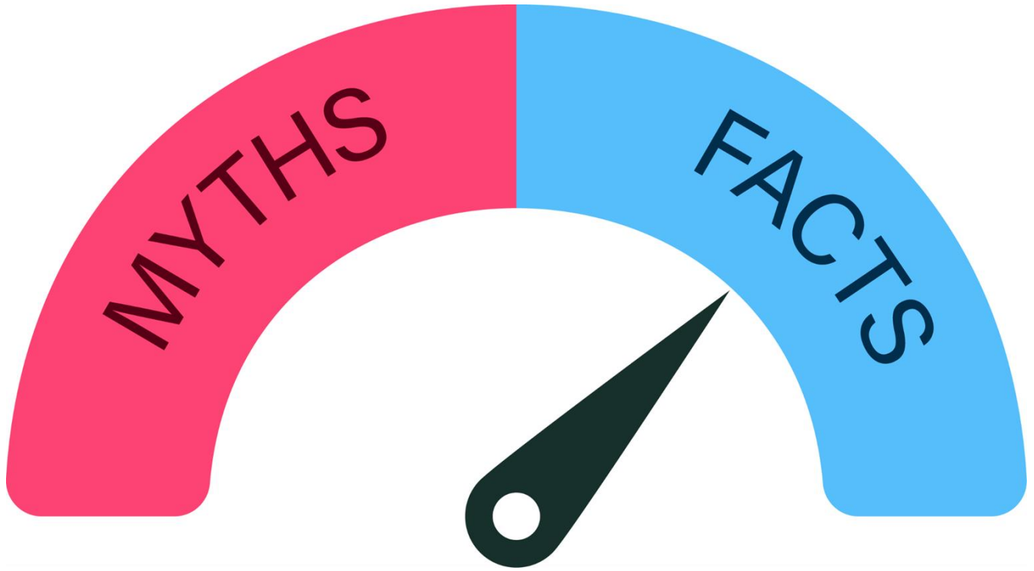
# SSH for Linux or Unix

Ansible uses the SSH protocol to communicate with remote machines. By default, Ansible connects to remote machines using native OpenSSH and user's current username, much like SSH.


Verify that users can connect to all of the nodes in their inventory using SSH with the same username.

# Ansible for Windows Hosts

Ansible and its supported platforms are often believed to be available only for Linux or Unix. Though Ansible is not natively available for Windows yet, users can use it to manage their Windows PCs.

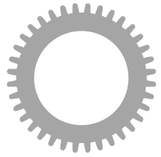



Users need to establish some settings on their Windows PC so that Ansible can communicate with it, just like Linux-managed nodes and do automated operations.

# Configure Windows Hosts Using WinRM

Host Requirements:

Powershell and .NET Framework upgrading

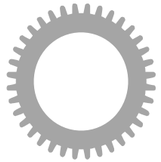


WinRM Memory Hotfix

# Configure Windows Hosts Using WinRM

WinRM setup:

WinRM Listener



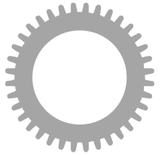
WinRM Service Options

Common WinRM Issues

# Configure Windows Hosts Using WinRM

WinRM Listener:

Setup WinRM Listener

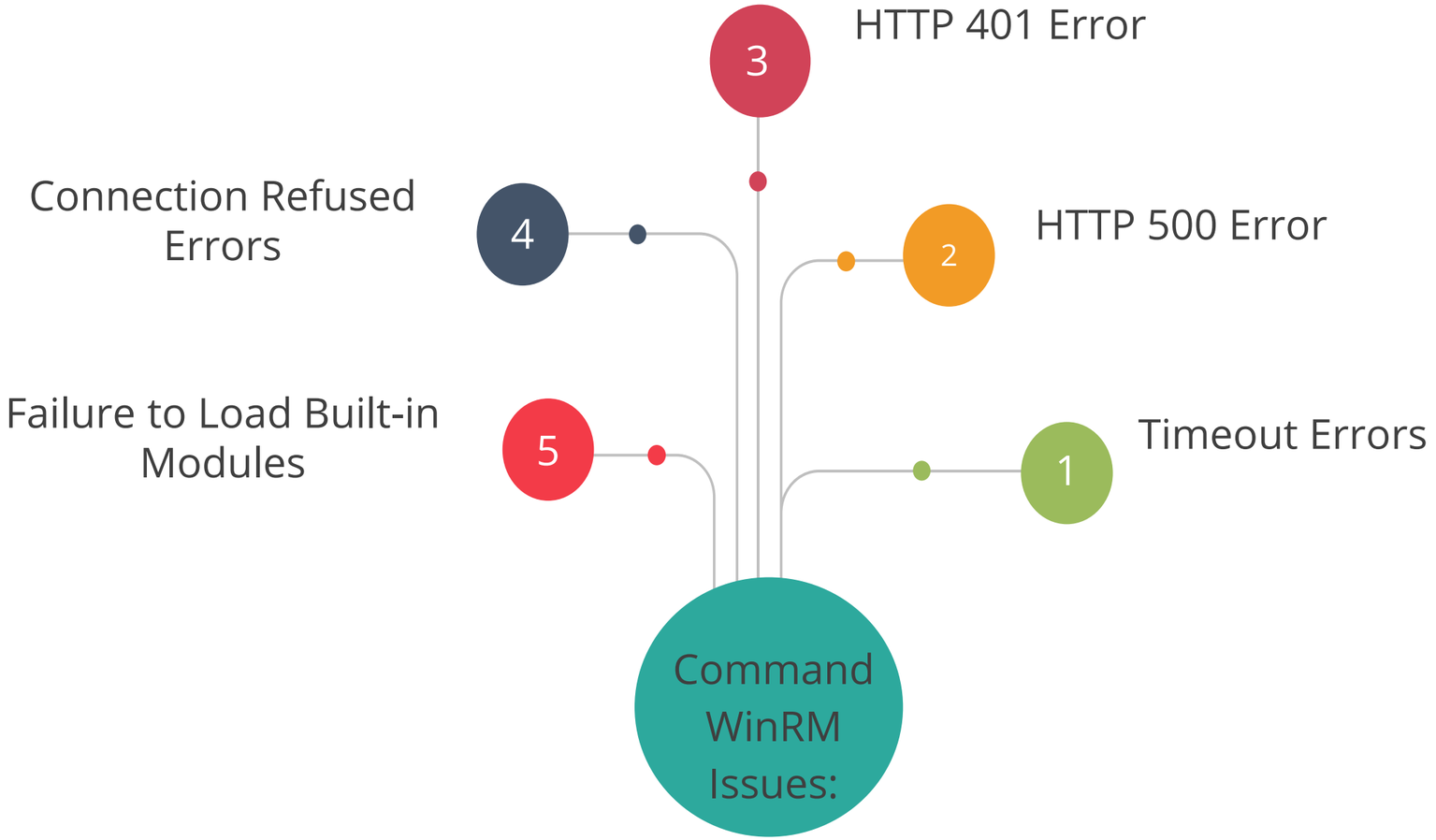


Delete WinRM Listener

# Configure Windows Hosts Using WinRM

WinRM Service Option:
Set authentication choices and memory settings to modify the behavior of the WinRM service component.

# Configure Windows Hosts Using WinRM



# Key Takeaways

- Ansible server can be configured in five different ways.
- The default location for Ansible inventory file is **/etc/ansible/hosts**.
- Any variable overrides any playbook term, command-line option, or configuration setting.
- Ansible can manage Windows OS.

## Simplification of Inventory with ranges

**Duration: 25 Min.**

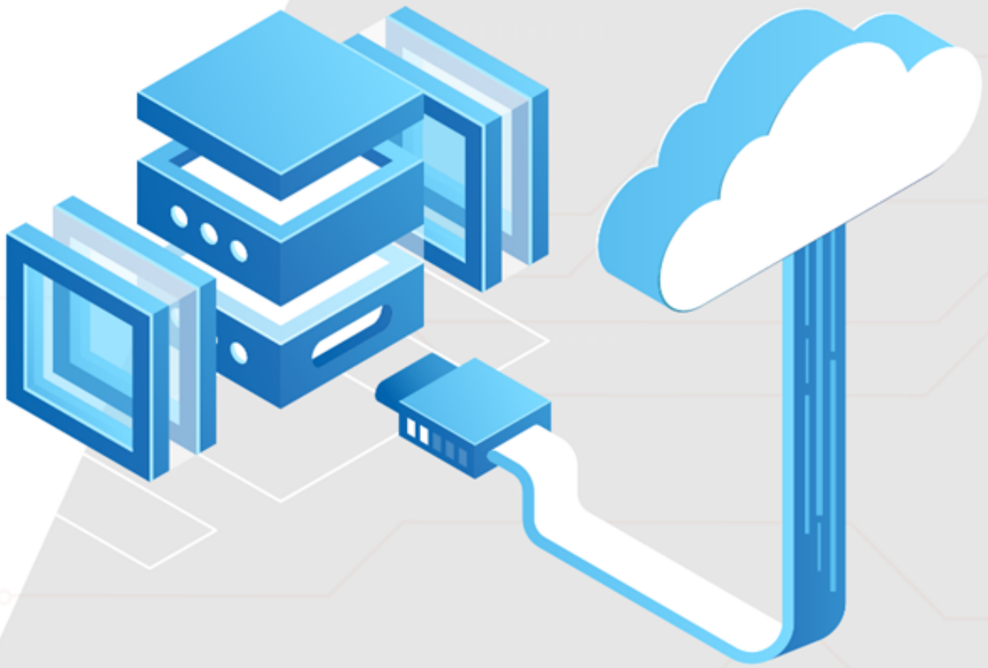
**Project agenda:** To do simplification of inventory with ranges

**Description:** Ansible works against multiple managed nodes or hosts in your infrastructure at the same time, using a list or group of lists known as inventory. Once your inventory is defined, you use ranges to select the hosts you want Ansible to run against.

**Perform the following:**

1. Install Ansible on main node (Ansible server)
2. Add host and groups to hosts file
3. Establish the connectivity between the hosts specified in the host file and the ansible server
4. Configuring the hosts by using range-based access

LESSON-END PROJECT



# Thank you