

Running Ad Hoc Commands

Ansible Fundamentals

Agenda

- Ansible Configuration File
- Ansible command syntax
- Run ad-hoc commands
- Authenticating Ansible Connections

Ansible Configuration File overview

Introduction to Ansible

Configuration File

- **Unified Settings:** The Ansible configuration file, `ansible.cfg`, provides a consolidated location to define settings that dictate how Ansible operates and interacts with different systems.
- **Hierarchy of Precedence:** Ansible can have multiple configuration files, and they are processed in a specific order of precedence, allowing for both global and project-specific settings.
- **INI Format:** The configuration file uses the INI format, making it easy to read and edit. Sections are defined using square brackets, and key-value pairs within those sections set various configurations.
- **Default Location:** By default, Ansible looks for the configuration file at `/etc/ansible/ansible.cfg`, but this can be overridden by user-specific or project-specific configuration files.
- **Flexibility:** The configuration file allows users to modify a wide range of parameters, from specifying the inventory path, adjusting parallel task execution, setting timeout values, to defining custom plugins or modules paths.

Places for Configuration

- **ANSIBLE_CONFIG** Environment Variable
 - An environment variable that points to the location of the config file.
- Current Directory
 - **ansible.cfg** in the current directory from which ansible or ansible-playbook is run.
- Home Directory
 - **.ansible.cfg** in the user's home directory.
- Global Configuration
 - **/etc/ansible/ansible.cfg**

Some configurations

- **inventory**: Specifies the location of the inventory file, which contains a list of the nodes that Ansible manages.
 - Example: **inventory = /etc/ansible/hosts**
- **remote_user**: Default username used to connect to target machines.
 - Example: **remote_user = admin**
- **host_key_checking**: Determines if Ansible checks the remote host's SSH key. Turning this off is useful for managing a large number of hosts without initial manual intervention.
 - Example: **host_key_checking = False**
- **forks**: Defines the number of parallel processes to use when communicating with remote hosts. It essentially controls parallelism.
 - Example: **forks = 10**
- **log_path**: Specifies the location where Ansible should write its log file. This is useful for troubleshooting and auditing purposes.
 - Example: **log_path = /var/log/ansible.log**

Ansible command syntax

Introduction to Ansible

CLI Commands

- `ansible`
- `ansible-config`
- `ansible-console`
- `ansible-doc`
- `ansible-inventory`
- `ansible-playbook`
- `ansible-pull`
- `ansible-vault`

CLI Commands: **ansible**

- This command allows you to execute ad-hoc commands on target hosts
- Example: This pings all hosts in your inventory to check if they are reachable.



```
ansible all -m ping
```

CLI Commands: **ansible-config**

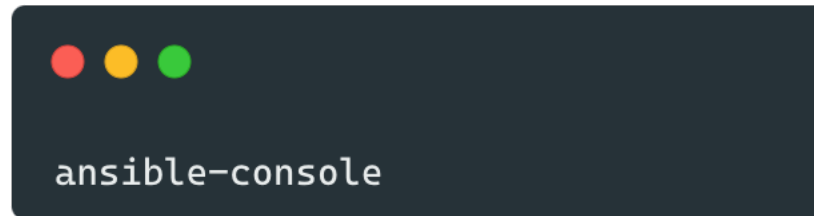
- Allows you to view, list, and manage Ansible configuration
- Example: This displays the current Ansible configuration



```
ansible-config view
```


CLI Commands: **ansible-console**

- Provides an interactive REPL (Read-Eval-Print Loop) interface for executing ad-hoc commands
- Example: This starts the interactive Ansible console



CLI Commands: **ansible-doc**


- Provides documentation on Ansible modules
- Example: This shows documentation for the **yum** module.



```
ansible-doc yum
```

CLI Commands: **ansible-inventory**

- Allows you to view and manage the Ansible inventory
- Example: This lists all hosts in the current inventory



```
ansible-inventory --list
```

CLI Commands: **ansible-playbook**

- Executes Ansible playbooks, which are scripts that define a set of tasks to be run on target hosts
- Example: This runs the **deploy_app.yml** playbook.



```
ansible-playbook deploy_app.yml
```

CLI Commands: **ansible-pull**

- A mode that inverts the default push architecture of Ansible into a pull architecture, which can be useful for scalable or decentralized setups
- Example: This pulls a playbook from a Git repository and executes it locally



```
ansible-pull -U https://github.com/user/repo.git
```

CLI Commands: **ansible-pull**

- Provides encryption and decryption capabilities for sensitive data in playbooks or variables files
- Example: This encrypts the **secrets.yml** file.



```
ansible-vault encrypt secrets.yml
```


Run ad-hoc commands

Introduction to Ansible

Ad-Hoc Commands

- An ad hoc command uses the **/usr/bin/ansible** command-line tool to automate a single task on one or more managed nodes.
- Ad hoc commands are quick and easy, but they are not reusable.
- Ad hoc commands demonstrate the simplicity and power of Ansible
- The concepts you learn here will port over directly to the playbook language. Before reading and executing these examples, please read

Why Using Ad-hoc Commands?

- Ad hoc commands are great for tasks you repeat rarely
- For example, if you want to power off all the machines in your lab for Christmas vacation, you could execute a quick one-liner in Ansible without writing a playbook
- An ad hoc command looks like this:

```
$ ansible [pattern] -m [module] -a "[module options]"
```

- The **-a** option accepts options either through the **key=value** syntax or a JSON string starting with { and ending with } for more complex option structure

How to identify machines

- Ad hoc commands use inventories to define the machines to be reached
- Inventory may be referenced in different ways following this precedence list
 - Command Line **-i** Option
 - **ANSIBLE_INVENTORY** Environment Variable
 - **ansible.cfg** Configuration File:
 - Default Inventory Path: **/etc/ansible/hosts**
- Inventory file defines groups of hosts and identifies them with a name
- When executing the command you use patterns to define which machines you will run the command

Common Patterns

Description	Pattern(s)	Targets
All hosts	all (or *)	
One host	host1	
Multiple hosts	host1:host2 (or host1,host2)	
One group	webserver	
Multiple groups	webserver:dbserver	All hosts in webserver plus all hosts in dbserver
Excluding groups	webserver:!atlanta	All hosts in webserver except those in atlanta
Intersection of groups	webserver:&staging	Any hosts in webserver that are also in staging
Control node	localhost	Run command on control node only

Simple ad-hoc commands

- Reboot all servers on **web** group on inventory



```
$ ansible web -a "/sbin/reboot"
```


- Ansible runs 5 execution simultaneous. If you want to execute 10, you use flag **-f**



```
$ ansible web -a "/sbin/reboot" -f 10
```

Simple ad-hoc commands

- If you want to execute the command with another username, you may use **-u** flag



```
$ ansible atlanta -a "/sbin/reboot" -f 10 -u username
```

- And sometimes you need to elevate your user



```
$ ansible atlanta -a "/sbin/reboot" -f 10 -u username --become [--ask-become-pass]
```

Run ad-hoc commands using modules

- On previous samples, you're using the **ansible.builtin.command**
- This is the default module and you don't need to specify on every command
- The flag **-a** represent the argument for module command, that on this case represent the command itself
- To specify a different module you may use **-m** flag
- Module name can be composed by several blocks using a dot to separate them
- When you do use this full name, the module will be related with **ansible.builtin**


Run ad-hoc commands using modules

- Ping all machines



```
$ ansible all -m ping
```

- Copy **/etc/hosts** to **/tmp/hosts** on atlanta group within the inventory



```
ansible atlanta -m ansible.builtin.copy -a "src=/etc/hosts dest=/tmp/hosts"
```

Run ad-hoc commands using modules

- You may create folders with several parameters

```
ansible webservers -m file -a "dest=/path mode=600 owner=user group=group"
```

- Restart **httpd** service on **webservers** group within the inventory

```
$ ansible webservers -m service -a "name=httpd state=restarted"
```

Run ad-hoc commands with explicit inventory

- Using **-i** flag to define inventory



```
$ ansible -i inventory/dev.yml webserver -m service -a "name=httpd state=restarted"
```

Modules names

- In modern Ansible practices, it's recommended to use the fully qualified collection name (FQCN) for modules
- This practice helps in clearly distinguishing between core modules and those from other collections, ensuring clarity and avoiding conflicts.
- For example, instead of using just `copy`, `ping`, or `file`, you would use:
 - **`ansible.builtin.copy`**
 - **`ansible.builtin.ping`**
 - **`ansible.builtin.file`**
- This is especially useful in environments where custom modules or modules from other collections are also being used.

Run Ad-Hoc Commands

Demo

Authenticating Ansible Connections

Introduction to Ansible

How to authenticate

- SSH Keys

- Ansible primarily uses SSH keys for authentication
- It's the most common method where the control machine has a private key and the managed nodes have the corresponding public key in the authorized keys list

- Username & Password

- While less secure and not recommended for production, Ansible can use SSH with a username and password for authentication
- This method can be useful in scenarios where key-based authentication isn't feasible
- You need to have **sshpass** installed on control node

How to authenticate

- Become (Privilege Escalation)
 - For tasks that require elevated privileges, Ansible uses the become method.
 - This can leverage tools like sudo, su, pbrun, and others to gain higher-level permissions.
- SSH Configuration & ssh-agent
 - Ansible can leverage existing SSH configurations and keys loaded into the ssh-agent for authentication.
 - This allows users to use jump hosts, non-default ports, and other SSH settings.
- Vault for Encrypted Credentials
 - Ansible Vault can encrypt sensitive data, including authentication credentials.
 - This ensures that secrets are stored securely but can be decrypted by Ansible during playbook runs.

SSH Key

- Ansible leverages the native SSH mechanism for authentication, which means it uses the SSH keys already defined on the control node (the machine running Ansible) to authenticate to remote servers
- Default Private Key
 - By default, Ansible uses the private key located at `~/.ssh/id_rsa`
- Specifying a Different Key
 - If you want to use a different private key, you can specify it using the `--private-key` option or set it in the Ansible configuration file (`ansible.cfg`) using the `private_key_file` setting.
- ssh-agent
 - If you're using **ssh-agent** on the control node, Ansible can leverage it.
 - When you add your private keys to the agent using **ssh-add**, Ansible will use the identities loaded into the agent for authentication.
- Ansible Vault
 - If you need to store SSH private keys securely, you can encrypt them using Ansible Vault.
 - When running playbooks, you can decrypt the key on-the-fly, ensuring that sensitive keys are not exposed in plaintext.

Authentication

Demo

