Bachelor of Technology

Computer Engineering Sem : 7

01CE0717 – DevOps Essentials

Marwadi University
Marwadi Chandarana Group

NAAC A+

Unit : 4 – Configuration Management Using Ansible

Content

- Ansible Introduction,
- Installation,
- Ansible master/slave configuration,
- YAML basics, Ansible modules,
- Ansible Inventory files,
- Ansible playbooks,
- Ansible Roles,
- Adhoc commands in ansible

Ansible

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

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

Ansible

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

Ansible

- Ansible is designed for multi-tier deployment. Ansible does not manage one system at time, it models IT infrastructure by describing all of your systems are interrelated.

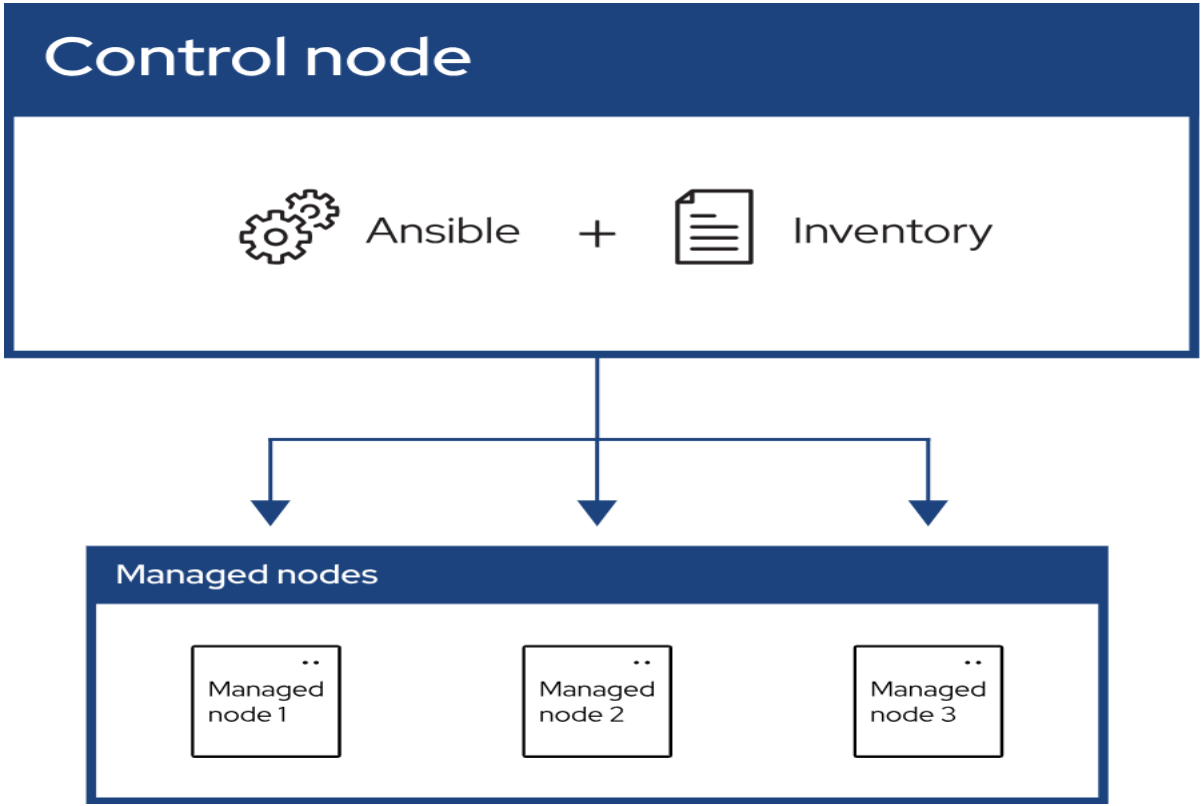
- Ansible is completely agentless which means Ansible works by connecting your nodes through ssh(by default). But if you want other method for connection like Kerberos, Ansible gives that option to you.

Ansible

- After connecting to your nodes, Ansible pushes small programs called as “Ansible Modules”. Ansible runs that modules on your nodes and removes them when finished.

- Ansible manages your inventory in simple text files (These are the hosts file). Ansible uses the hosts file where one can group the hosts and can control the actions on a specific group in the playbooks

Ansible


Control node
Ansible
+
Inventory
Managed nodes
Managed node 1
Managed node 2
Managed node 3

Ansible Installation

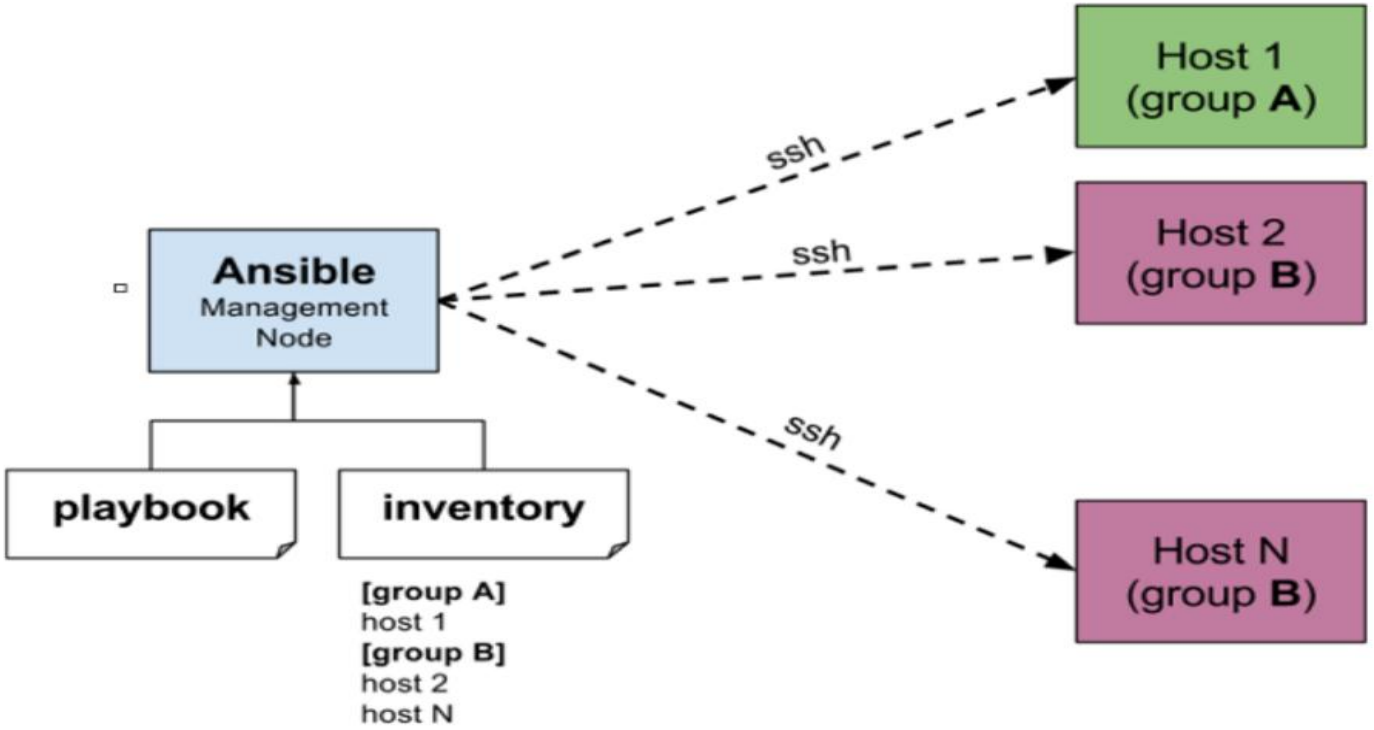
- https://phoenixnap.com/kb/install-ansible-on-windows

- https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

Ansible master/slave configuration

- Ansible works by connecting to your nodes and pushing out small programs, called "Ansible modules" to them. Ansible then executes these modules (over SSH by default), and removes them when finished. Your library of modules can reside on any machine, and there are no servers, daemons, or databases required.

Ansible master/slave configuration



Ansible master/slave configuration

- The management node in the above picture is the controlling node (managing node) which controls the entire execution of the playbook. It's the node from which you are running the installation.
- The inventory file provides the list of hosts where the Ansible modules needs to be run and the management node does a SSH connection and executes the small modules on the hosts machine and installs the product/software.

Ansible master/slave configuration

- Beauty of Ansible is that it removes the modules once those are installed so effectively it connects to host machine , executes the instructions and if it’s successfully installed removes the code which was copied on the host machine which was executed.

Ansible Inventory Files

- The inventory file is also called as host fie.
- It is a text file that contains the list of managed nodes. These nodes can be arranged in different host groups. This can be useful for organizing your hosts and for running tasks on specific groups of hosts.
- The inventory file is an essential component of ansible architecture.
- In this file, the IP address of each node is specified.
- The inventory file is used by ansible to determine which hosts to run tasks on.
- When ansible runs a playbook, it will first look for the inventory file in the current working directory. If the inventory file is not found, ansible will look for it in the default location (/etc/ansible/hosts).

Ansible Modules

- Ansible modules are used to perform various tasks on remote hosts.
- These are basically small, standalone scripts that ansible uses to communicate with the remote hosts in order to execute specific actions.
- The modules can be executed from the command line or can be written in the playbooks.
- They interact with local machines, APIs, or remote systems to perform tasks like managing system resources, installing packages, manipulating files, etc.
- Modules provide a defined interface, accept arguments, and return information to Ansible by printing JSON to stdout before exiting.

Ansible Modules

- The modules can be executed from the command line or can be written in the playbooks.
- While writing the module we use the options such as –m and –a.
  - The –m is for module and –a is for argument, after the –a option we specify the executable command in double quotes.
- Commonly used modules
  - Command: executes a command on remote hosts
    - i.e.: $ansible all –m command –a “echo ‘Hello Hosts’ ”
  - User: It is used to create, modify or delete users.
    - i.e.: $ansible all –m user –a “name=new_user state=present”

Ansible Modules

- Commonly used modules (continue…)
  - Ping: It is used to connectivity to remote hosts
    - i.e.: $ansible all –m ping
  - Copy: It is used to copy file from control machine to remote hosts
    - i.e.: $ansible dev –m copy –a “src=/home/ansible_user/myfile.txt dest=/…./myfile.txt”
  - Package: used to install, remove or upgrade the packages.
    - i.e.: $ansible package –a “name=nginx state=present”
  - Shell: used to run shell commands with more flexibility on remote hosts

Ansible Modules

- Commonly used modules (continue…)
  - Shell: used to run shell commands with more flexibility on remote hosts
    - i.e.: $ansible dev –m shell –a “cmd=ls”
  - Service: used to start, stop or restart a specific service on remote hosts
    - i.e.: $ansible dev –m service –a “name=apache2 state=started”
    - State can be started / stopped / restarted / reloaded.

Yaml Basics

- The YAML is human – readable data serialization language which is used to write ansible playbook.
- YAML stands for yet another markup language or YAML ain't markup language (a recursive acronym), which emphasizes that YAML is for data, not document.
- YAML is very popular because it is easy to read and understand.
- YAML files use a .yaml or .yml extension and follow specific syntax rules.
- Every YAML file starts with 3 dashes (---) while the file ends with 3 dots (…).

Yaml Basics

- **Indentation and whitespaces:** In YAML Python-style indentation is used to determine the structure and indicate nesting. The indentation level must be consistent throughout the document. The indentation is typically done using number of spaces. Normally 2 spaces, 4 spaces per level of indentation is done.

- **Comments:** Comments can be identified with a pound or hash symbol (#). YAML does not support multi-line comment, each line needs to be suffixed with the pound character.

```
# This line is YAML comment.
```

Yaml Basics

- **Key value pair:** The YAML script is based on key value pair. The value can be string, number of any other data type. The key value pair is used to store data, for defining the task or for specifying the configuration.
  - Note − There should be space between : and value.

    name: “Samir”

    msg: “Welcome”
- **List:** In YAML, we can declare a list or arrays. The list is used to define multiple items under a single key.

  Days:

  - Sunday

  - Monday

  - Tuesday

Yaml Basics

- **Dictionaries:** Dictionaries or maps can be created with the help of key value pair. The dictionaries are represented as mappings. The mappings are basically the key value pairs. For example –

```
student:
        name: Rahul
        std: 10
        address:
                street: College Road
                city: Rajkot
                state: Gujarat
```

Yaml Basics

- **Mult-line values:** We can define the multi – line values in YAML by using | character at the start of value followed by newline. For example

```
OS: |
        Windows
        Linux
        MacOS
```

- **Folded style:** Use the > character to start the value and then end the value with a newline character (\n). This is useful for writing long paragraphs of text without introducing line breaks. Newlines are converted to spaces.
- For example

```
OS: >
        Windows
        Linux
        MacOS
```

Yaml Basics

- **Data Types:** YAML supports various scalar data types, including strings, numbers, booleans, and null values. Scalars don't require special syntax; you can simply write them as plain values. For example -

  name: Samir
  age: 35
  is_married: yes

- **YAML is a superset of JSON:** As YAML is a superset of JSON, any valied JSON document should be valid YAML.
- For Example:

  YAML to JSON

  https://onlineyamltools.com/convert-yaml-to-json

  JSON to YAML

  https://onlineyamltools.com/convert-json-to-yaml

Yaml Basics

- **Benefits of YAML**
  - Human-readable
  - Simple to use
  - Fast
  - Unambiguous
  - Portable
  - Matching with popular languages
  - Secure Implementation

Yaml Playbook

- An ansible playbook is a set of instructions that ansible uses to automate tasks on remote hosts.
- Playbooks are written in YAML, a human-readable format.
- They are the core component of any Ansible configuration.

What they do?

- Playbooks automate tasks such as installing packages, configuring services, and copying files. They can also be used to provision infrastructure, manage configurations, and deploy applications.

How they work?

- Ansible reads the playbook file and parses the YAML syntax. The playbook is made up of one or more plays, which are collections of tasks that are run in sequence.

Yaml Playbook

How to use them?

- To run a playbook, an administrator runs the *ansible-playbook* command against the target machines.

How to create them?

- Playbooks can be created by writing a file in YAML that defines the tasks and configurations to be applied to devices.

How to use variables

- Variables can be defined using a *vars* block at the beginning of a play, or in external YAML files.

Yaml Playbook

- Each playbook contains following important elements:
- **Hosts:** In a play, we can specify the target hosts or groups of hosts where you want to apply the tasks.
- **Play:** A play is the core unit of Ansible execution. It contains a list of tasks, variables, and maps that link the tasks to specific hosts.
- **Task:** A task is a unit of work that is executed on a target host. It calls an Ansible module, which is responsible for an action and its configuration parameters.
- **Module:** A module is a type of plugin that executes tasks on a target. Each module has its own set of options and properties.
- **Variable**: A variable can store values that can be used across multiple tasks and playbooks. Variables can be defined in the playbook or in separate files.

Yaml Playbook

- Ex.1: Create a playbook to gather the information about the remote hosts (i.e. localhost).

  test1.yaml

```
- hosts: localhost
  user: ansible_user123
  connection: ssh
  become: yes
  gather_facts: yes
```

Yaml Playbook

- Ex.2: Create a playbook to create an empty file on remote hosts.

test2.yaml

```
- hosts: localhost
  tasks:
  - name: Creates an empty file
    file:
        path: "/mnt/c/Users/Admin/newFileName.txt"
        state: touch
```

Yaml Playbook

- Ex.3: Create a playbook to copy the content in file on remote hosts.

test3.yaml

```
- hosts: localhost
  tasks:
    - name: copy content in file
      copy:
        content: |
          <html>
          <head><title>Welcome to NGINX on Amazon Linux 2!</title></head>
          <body>
          <h1>Success! NGINX is installed and running!</h1>
          </body>
          </html>
        dest: /mnt/c/Users/Admin/another_file23.html
        mode: '0644'
```

Ansible Roles

- Ansible roles are a powerful feature that allows you to organize and reuse automation tasks in a structured way. They enable you to break down complex playbooks into smaller, more manageable components, making it easier to maintain and share your automation code.
- Roles provide a framework for fully independent, or interdependent collections of variables, tasks, files, templates, and modules.
- ***Definition****:* Ansible roles are a way to group related tasks, variables, files, templates, and handlers into a single unit of organization. This makes it easier to manage and reuse automation code across different projects.
- ***Structure:*** Each role has a defined directory structure that typically includes directories for tasks, handlers, variables, files, and templates. This organization helps keep your code clean and modular.

Ansible Roles

- Roles are not playbooks. Roles are small functionality which can be independently used but have to be used within playbooks. There is no way to directly execute a role. Roles have no explicit setting for which host the role will apply to.
- Top-level playbooks are the bridge holding the hosts from your inventory file to roles that should be applied to those hosts.
- The directory structure for roles is essential to create a new role.
- Roles have a structured layout on the file system. The default structure can be changed but for now let us stick to defaults.
- Each role is a directory tree in itself. The role name is the directory name within the /roles directory.

Ansible Roles

Standard Role Directory Structure

A typical Ansible role directory structure looks like this:

text

```
my_role/
├── defaults
│   └── main.yml      # Default variables for the role
├── files
│   └── some_file.txt # Static files to be deployed
```

To be contd…..

Ansible Roles


```
├── handlers
│   └── main.yml      # Handlers for the role
├── meta
│   └── main.yml      # Metadata about the role
├── tasks
│   └── main.yml      # Main tasks for the role
├── templates
│   └── some_template.j2 # Jinja2 templates to be rendered
└── vars
    └── main.yml      # Variables specific to the role
```

Ansible Roles

Benefits of Using Roles

Modularity: Roles allow you to break down your automation tasks into smaller, reusable components.

Reusability: Once a role is created, it can be reused across multiple playbooks and projects.

Organization: Roles provide a clear structure for organizing tasks, variables, and files, making it easier to understand and maintain your automation code.

Collaboration: Different team members can work on separate roles simultaneously without interfering with each other's work.

Utilizing Roles in Ansible Playbooks

Adhoc commands in ansible

- Ansible ad hoc commands are a powerful feature that allows you to execute quick tasks on one or more managed nodes without the need to write a full playbook. They are particularly useful for performing one-off tasks or for testing purposes.

- Ad hoc commands are quick, one-liner commands that you can run using the ansible command-line tool. They are not stored for future use and are ideal for performing tasks that do not require the complexity of a playbook. Common use cases include managing packages, rebooting servers, copying files, and gathering system information.

Adhoc commands in ansible

Syntax of Ad Hoc Commands

The general syntax for running an ad hoc command is:

bash

ansible [hosts] -m [module_name] -a "[module options]"

[hosts]: This specifies the target hosts or groups defined in your inventory.

-m [module_name]: This specifies the Ansible module to use (e.g., ping, copy, yum, etc.).

-a "[module options]": This specifies the arguments to pass to the module.

Adhoc commands in ansible

Examples of Ansible Ad Hoc Commands

1. Pinging Hosts

To check connectivity to all hosts in the inventory:

bash

ansible all -m ping

2. Rebooting Servers

To reboot all servers in a group called webservers:

bash

ansible webservers -m command -a "/sbin/reboot"

Adhoc commands in ansible

3. Copying Files
To copy a file from the control machine to all hosts in a group called webservers:
bash
ansible webservers -m copy -a "src=/local/path/to/file dest=/remote/path/to/file“

4. Managing Packages
To ensure a package is installed on all hosts:
bash
ansible all -m yum -a "name=httpd state=present“

To remove a package:
bash
ansible all -m yum -a "name=httpd state=absent"

Adhoc commands in ansible

5. Creating Directories

To create a directory on all hosts:

bash

ansible all -m file -a "path=/tmp/mydir state=directory mode=0755"

THANK YOU