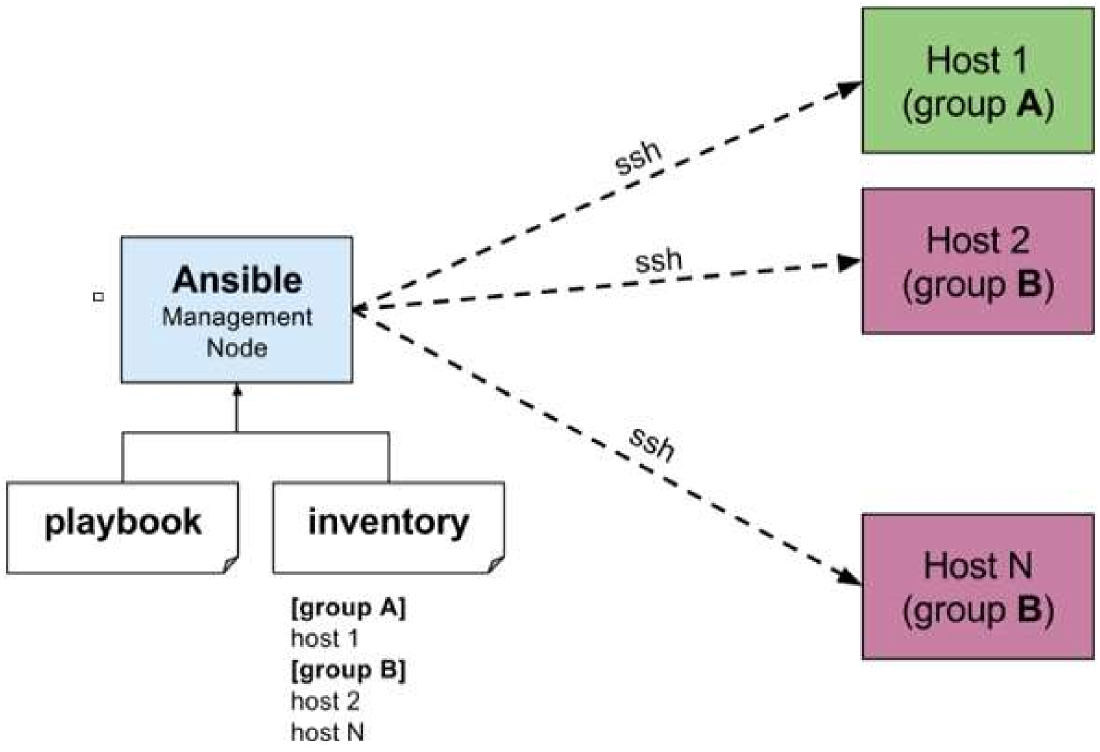
Ansible

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

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.



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.

Mainly, there are two types of machines when we talk about deployment –

Control machine – Machine from where we can manage other machines.

Remote machine – Machines which are handled/controlled by control machine.

There can be multiple remote machines which are handled by one control machine. So, for managing remote machines we have to install Ansible on control machine.

Control Machine Requirements

Ansible can be run from any machine with Python 2 (versions 2.6 or 2.7) or Python 3 (versions 3.5 and higher) installed.

Note — Windows does not support control machine. By default, Ansible uses ssh to manage remote machine.

Ansible does not add any database. It does not require any daemons to start or keep it running. While managing remote machines, Ansible does not leave any software installed or running on them. Hence, there is no question of how to upgrade it when moving to a new version.

Installation through Apt on Ubuntu Machine

For installing Ansible you have to configure PPA on your machine. For this, you have to run the following line of code —

```
$ sudo apt-get update

$ sudo apt-get install software-properties-common

$ sudo apt-add-repository ppa:ansible/ansible $ sudo apt-get update

$ sudo apt-get install ansible
```

After running the above line of code, you are ready to manage remote machines through Ansible.

Some common words related to Ansible.

Service/Server — A process on the machine that provides the service.

Machine — A physical server, vm(virtual machine) or a container.

Target machine — A machine we are about to configure with Ansible.

Task — An action(run this, delete that) etc managed by Ansible.

Playbook — The yml file where Ansible commands are written and yml is executed on a machine.

Managing Packages

The Ad-hoc commands are available for yum and apt. Following are some Ad-hoc commands using yum. The following command checks if yum package is installed or not, but does not update it.

$ Ansible abc -m yum -a "name = demo-tomcat-1 state = present"

The following command check the package is not installed.

$ Ansible abc -m yum -a "name = demo-tomcat-1 state = absent"

The following command checks the latest version of package is installed.

$ Ansible abc -m yum -a "name = demo-tomcat-1 state = latest"

Gathering Facts

Facts can be used for implementing conditional statements in playbook. You can find adhoc information of all your facts through the following Ad-hoc command –

$ Ansible all -m setup

Playbooks are the files where Ansible code is written. Playbooks are written in YAML format. YAML stands for Yet Another Markup Language. Playbooks are one of the core features of Ansible and tell Ansible what to execute. They are like a to-do list for Ansible that contains a list of tasks.

Playbooks contain the steps which the user wants to execute on a particular machine. Playbooks are run sequentially. Playbooks are the building blocks for all the use cases of Ansible.

Playbook Structure

Each playbook is an aggregation of one or more plays in it. Playbooks are structured using Plays. There can be more than one play inside a playbook.

The function of a play is to map a set of instructions defined against a particular host.

YAML is a strict typed language; so, extra care needs to be taken while writing the YAML files. There are different YAML editors but we will prefer to use a simple editor like notepad++. Just open notepad++ and copy and paste the below yaml and change the language to YAML (Language → YAML).

A YAML starts with --- (3 hyphens)

Create a Playbook

```
---
  name: install and configure DB
  hosts: testServer
  become: yes

  vars:
    oracle_db_port_value : 1521

  tasks:
  -name: Install the Oracle DB
    yum: <code to install the DB>

  -name: Ensure the installed service is enabled and running
  service:
    name: <your service name>
```

Save the above content in a file as **test.yml**.

Roles

Roles provide a framework for fully independent, or interdependent collections of variables, tasks, files, templates, and modules.

In Ansible, the role is the primary mechanism for breaking a playbook into multiple files. This simplifies writing **complex playbooks**, and it makes them easier to reuse.

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.

Creating a New Role

Each role is a directory tree in itself. The role name is the directory name within the /roles directory.

$ ansible-galaxy -h

$ ansible-galaxy init --force --offline vivekrole

- vivekrole was created successfully

$ tree vivekrole/

```
vivekrole/

├── defaults

│   └── main.yml

├── files ├── handlers

│   └── main.yml

├── meta

│   └── main.yml

├── README.md ├── tasks

│   └── main.yml

├── templates ├── tests │   ├── inventory

│   └── test.yml

└── vars

   └── main.yml


8 directories, 8 files
```

Utilizing Roles in Playbook

This code is of the playbook vivek_orchestrate.yml. We have defined the hosts: tomcat-node and called the two roles – install-tomcat and start-tomcat.

```
- hosts: tomcat-node
roles:
  - {role: install-tomcat}
  - {role: start-tomcat}
```

There is a tasks directory under each directory and it contains a main.yml. The main.yml contents of install-tomcat are −

```
---
#Install vivek artifacts
-
  block:
    - name: Install Tomcat artifacts
      action: >
        yum name = "demo-tomcat-1" state = present
      register: Output

  always:
    - debug:
      msg:
        - "Install Tomcat artifacts task ended with message: {{Output}}"
        - "Installed Tomcat artifacts - {{Output.changed}}"
```

The contents of main.yml of the start tomcat are −

```
#Start Tomcat
-
```

```
  block:
    - name: Start Tomcat
    command: <path of tomcat>/bin/startup.sh"
    register: output
    become: true

  always:
    - debug:
      msg:
        - "Start Tomcat task ended with message: {{output}}"
        - "Tomcat started - {{output.changed}}"
```

<u>Ran the command to run the playbook.</u>

-vvv option for verbose output – verbose output

```
$ cd vivek-playbook/
```

This is the command to run the playbook

```
$ sudo ansible-playbook -i hosts vivek_orchestrate.yml –vvv
```

<u>Loops</u>

To loop, the 'with_items' syntax is being used.

with_items: "{{output.stdout_lines}}"

--> output.stdout_lines gives us the line by line output and then we loop on the output with the with_items command of Ansible.

<u>Conditionals</u>

```
---
#Tsting
```

```
- hosts: all
  vars:
    test1: "Hello Vivek"
  tasks:
    - name: Testing Ansible variable
    debug:
      msg: "Equals"
      when: test1 == "Hello Vivek"
```

Common Playbook Issues

- *Quoting*
- *Indentation*

Playbook is written in yaml format and the above two are the most common issues in yaml/playbook.

Yaml does not support tab based indentation and supports space based indentation, so one needs to be careful about the same.

Note – once you are done with writing the yaml , open this site(*https://editor.swagger.io/*) and copy paste your yaml on the left hand side to ensure that the yaml compiles properly. Swagger qualifies errors in warning as well as error.