Training Course

Ansible

Course Overview

- **This course helps you to understand how to Ansible works basic**
- **Prerequisites**
  - ✓ Some Linux system administration experience
  - ✓ Basic familiarity with Linux command line
  - ✓ Knowledge of popular development and scripting languages

Course Overview

- Module 1: Introduction to Ansible
- Module 2: Ansible Inventory
- Module 3: Ansible-playbook
- Module 4: Ansible Galaxy
- Module 5: Ansible Tower

Course Overview

Ansible Document and References

https://docs.ansible.com/

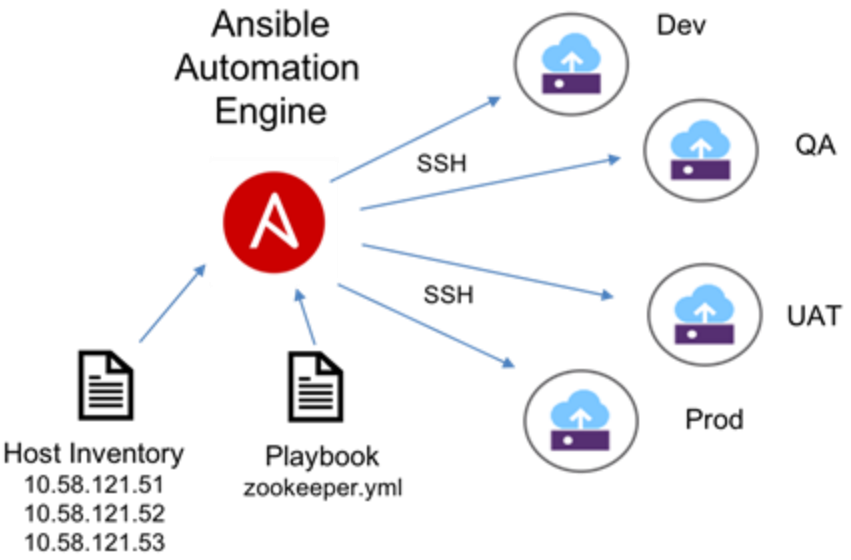
https://udemy.com

Module 1:
Introduction Ansible

What is Ansible?

- Ansible is an open-source software provisioning, configuration management, continuous delivery, application-deployment and security compliance tool enabling Infrastructure as Code.

- It uses SSH to connect to servers and run the configured Tasks. Ansible lets you control and configure nodes from a single machine.

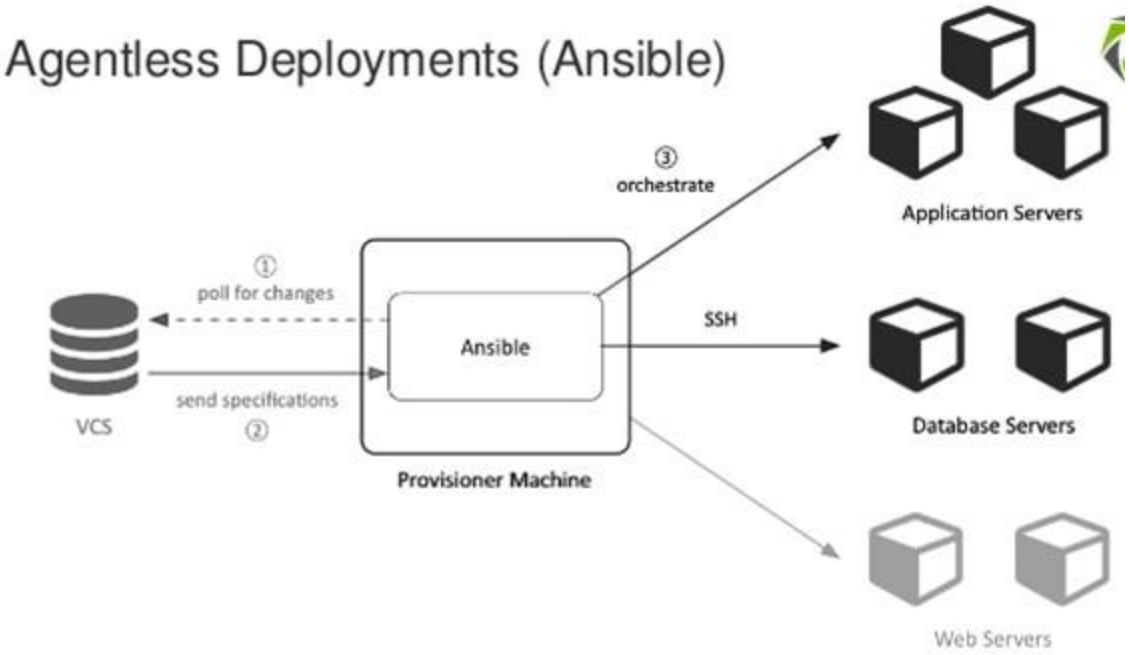


What is Ansible?

- **Simple and easy IT automation**
  Push model via OpenSSH, agentless
- **Provisioning**
  PXE booting and kickstarting bare-metal servers or VMs, or creating virtual or cloud instances from templates
- **Configuration management**
  Idempotent server state definitions
- **Application deployment**
  One-command standard deployments to update applications across many machines
- **Continuous delivery**
  Require sequential success of multiple processes
- **Security automation**
  Automate and standardize threat-scanning and firewall updates
- **Orchestration**
  Define how multiple configurations interact and ensure the disparate pieces can be managed as a whole

Why Ansible?

- **Agentless**: Ansible manages machines in an agent-less manner
- **Tiny Learning Curve**: Ansible is quite easy to learn



How does Ansible work?

- **Agentless**

Tasks are executed from your machine

- **Automated**

Reliable and less error prone

- **Playbooks**

Configuration/Installation/Deployment in a single YAML file, doubles as notes

- **Inventories**

Tasks can be executed on groups of machines

- **Module**

Atomic task e.g. Copy file

- **Task**

Uses a module with arguments

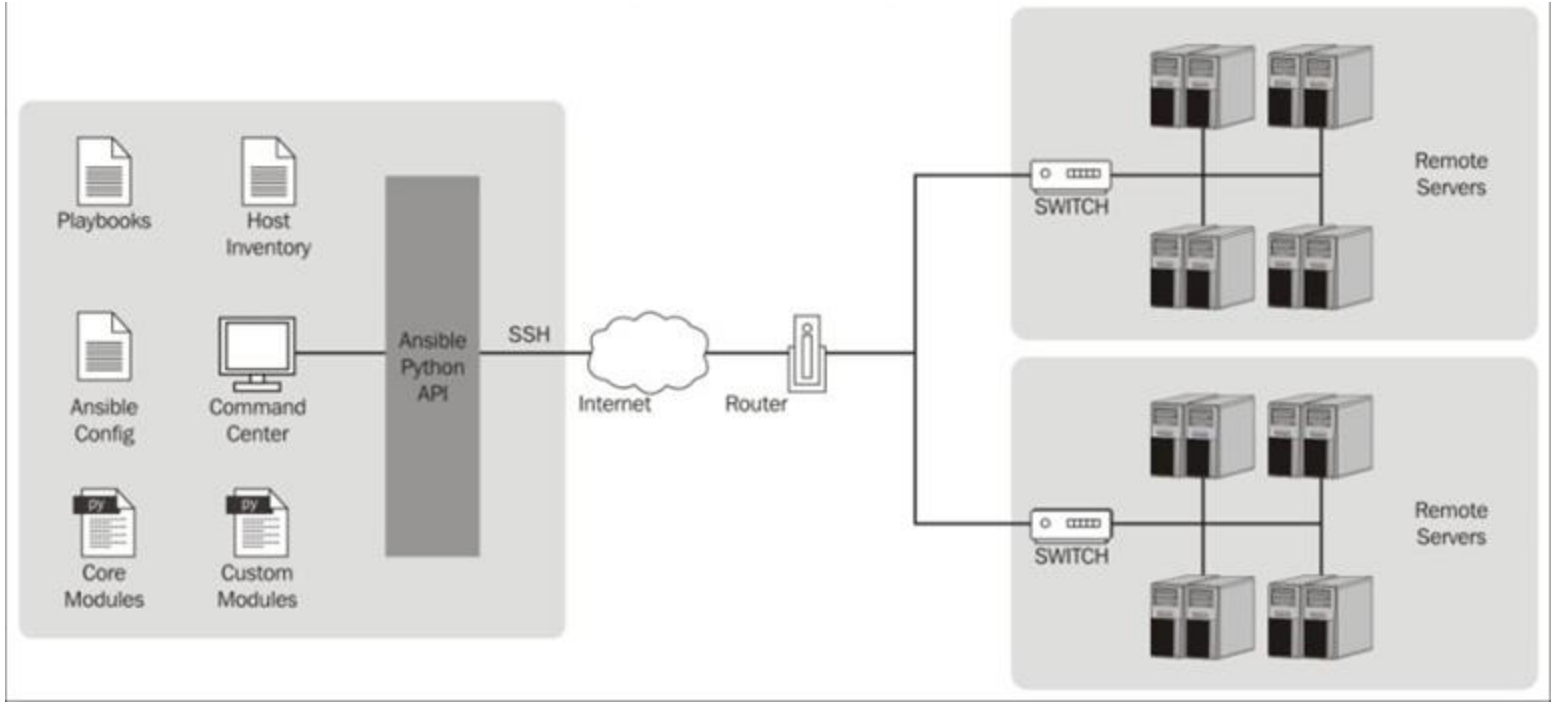
- **Play**

A series of tasks with designated hosts and user

- **Playbook**

A series of plays

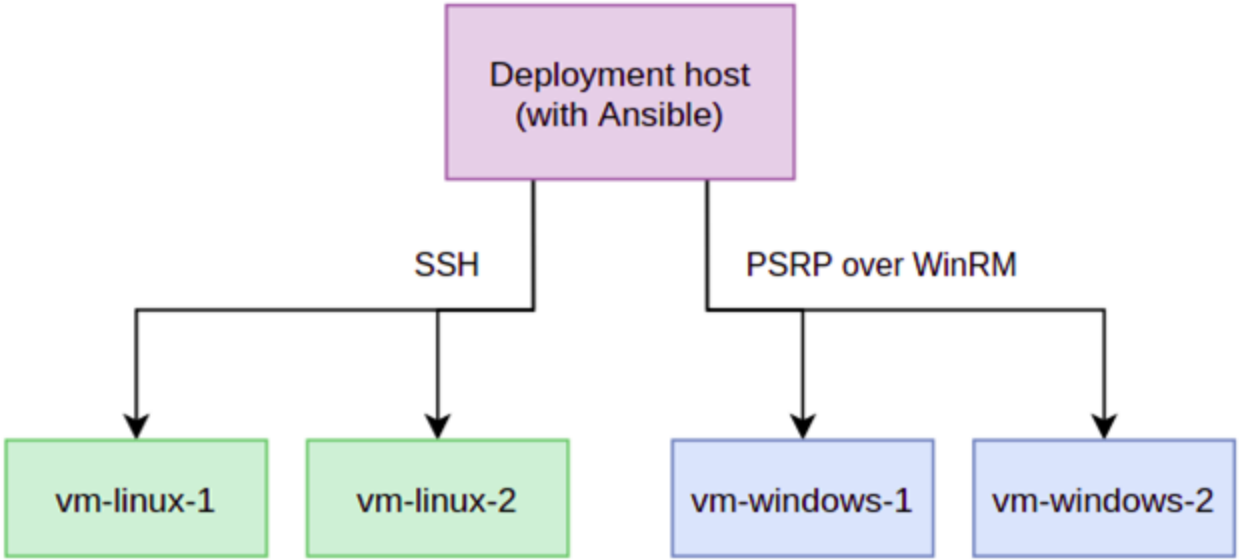
Architecture of Ansible



Ansible connectors

They traditionally use **different remote** access protocols:

- Linux: SSH
- Windows: PSRP / WinRM



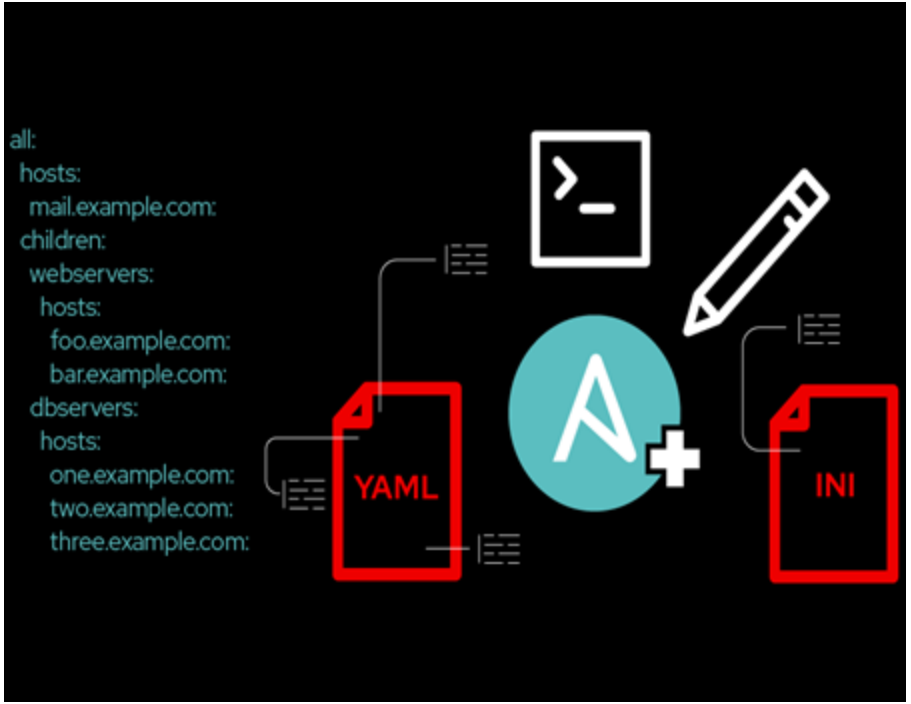
Install Ansible

sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository --yes --update
ppa:ansible/ansible
sudo apt install ansible
Gen key and copy to remote host
ssh-keygen
ssh-copy-id -i id_rsa.pub ubuntu@ip

Module 2:
Ansible Inventory

What is Ansible Inventory?

- The Inventory is a description of the nodes that can be accessed by Ansible.
- It is described by a configuration file, whose default location is in /etc/ansible/hosts.
- The configuration file lists either the IP address or hostname of each node that is accessible by Ansible.
- Every host is assigned to a group such as web servers, db servers etc.
- The Inventory file can be in one of many formats such as yaml, INI etc



Example different between INI and YAML

INI	YAML

mail.example.com
 	
[webservers]
foo.example.com
bar.example.com
 	
[databaseservers]
one.example.com
two.example.com
three.example.com
	all:
	  hosts:
	  mail.example.com
	  children:
	  webservers:
	  hosts:
	  foo.example.com:
	  bar.example.com:
	  databaseservers:
	  hosts:
	  one.example.com:
	  two.example.com:
	  three.example.com:

Module 2: Ansible Inventory

Describe hosts via variables:

INI	YAML
server1 ansible_host=192.1.2.3	hosts: server1: ansible_host: 192.1.2.3

Lists host name:

```
[webservers]
www.example[01:50].com
```

Define alphabetic ranges:

```
[databases]
Db-[a:f].example.com
```

Select the connection type and user on a per host basis:

```
[targets]
localhost              ansible_connection=local
other1.example.com     ansible_connection=ssh     ansible_user=ubuntu
other2.example.com     ansible_connection=ssh     ansible_user=ubuntu
```

Host variables

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

Group variables

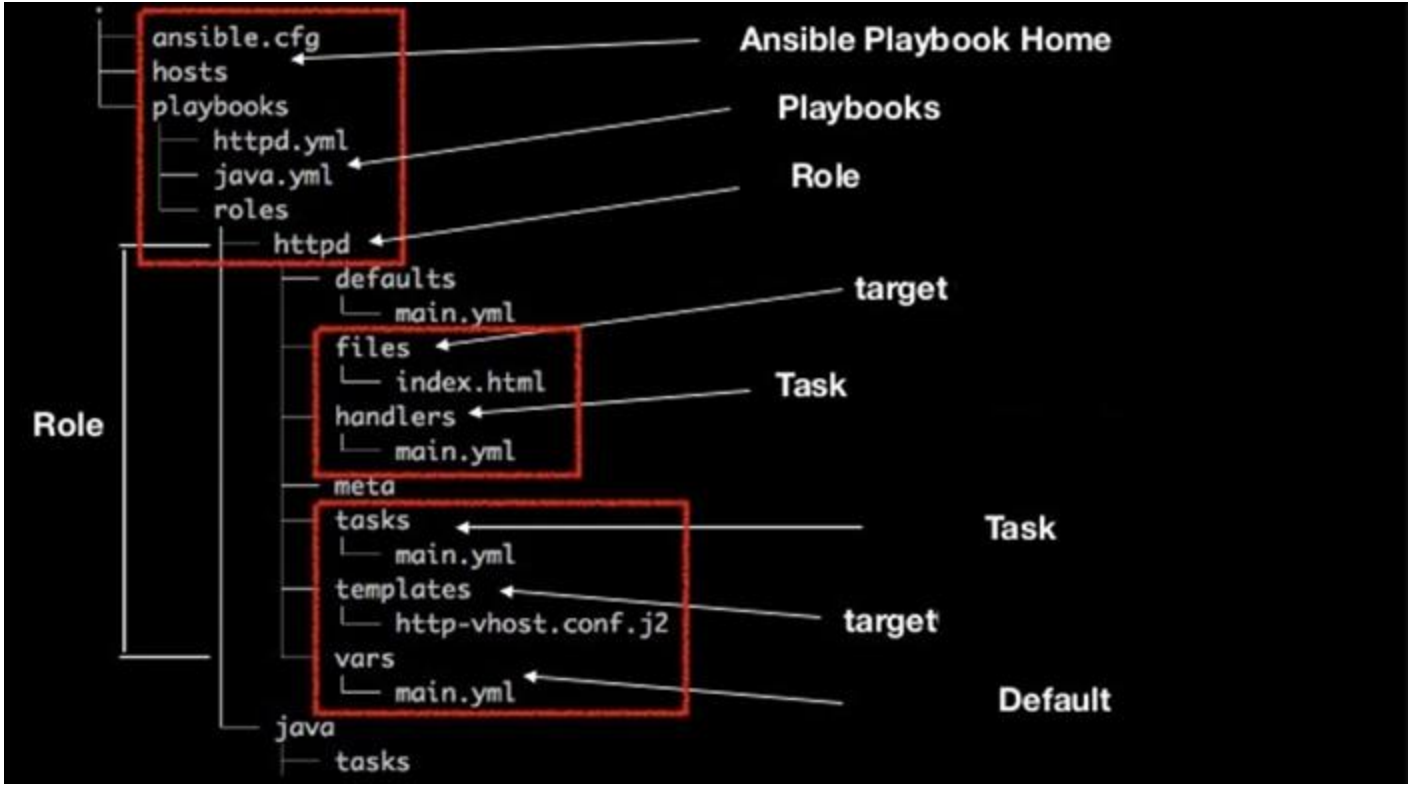
INI	YAML

[atlanta]
host1
host2
 	
[atlanta:vars]
ntp_server=ntp.atlanta.example.com
proxy=proxy.atlanta.example.com
	atlanta:
	  hosts:
	  host1:
	  host2:
	  vars:
	  ntp_server: ntp.atlanta.example.com
	  proxy: proxy.atlanta.example.com

Groups of groups and group variables

INI	YAML
[atlanta] host1 host2 [raleigh] host2 host3 [southeast:children] atlanta raleigh [southeast:vars] some_server=foo.southeast.example.com halon_system_timeout=30 self_destruct_countdown=60 escape_pods=2 [usa:children] southeast northeast southwest northwest	all: children: usa: children: southeast: children: atlanta: hosts: host1: host2: raleigh: hosts: host2: host3: vars: some_server: foo.southeast.example.com halon_system_timeout: 30 self_destruct_countdown: 60 escape_pods: 2 northeast: northwest: southwest:

Ansible Directory Structure



List of Behavioral Inventory Parameters

Host connection:

- **ansible_connection**: Connection type to the host. This can be the name of any of ansible connection plugins. SSH protocol types are smart, ssh or paramiko. The default is smart. Non-SSH based types are described in the next section.
- **ansible_host**: The name of the host to connect to, if different from the alias you wish to give to it.
- **ansible_port**: The ssh port number, if not 22.
- **ansible_user**: The default ssh username to use.

List of Behavioral Inventory Parameters

Specific to the SSH connection:

- **ansible_ssh_pass**: The ssh password to use (never store this variable in plain text; always use a vault).
- **ansible_ssh_private_key_file**: Private key file used by ssh. Useful if using multiple keys and you don't want to use SSH agent.
- **ansible_ssh_extra_args**: This setting is always appended to the default ssh command line.

  ex: ansible_ssh_extra_args='-o StrictHostKeyChecking=no'
- **ansible_ssh_pipelining**: Determines whether or not to use SSH pipelining. This can override the pipelining setting in ansible.cfg.
- **ansible_ssh_executable** (added in version 2.2): This setting overrides the default behavior to use the system ssh. This can override the ssh_executable setting in ansible.cfg.

List of Behavioral Inventory Parameters

Privilege escalation (see Ansible Privilege Escalation for further details):

- **ansible_become**: Equivalent to ansible_sudo or ansible_su, allows to force privilege escalation.
- **ansible_become_method**: which privilege escalation method should be used
- **ansible_become_user**: Equivalent to ansible_sudo_user or ansible_su_user, allows to set the user you become through privilege escalation.
- **ansible_become_pass**: Equivalent to ansible_sudo_pass or ansible_su_pass, allows you to set the privilege escalation password (never store this variable in plain text; always use a vault).
- **ansible_become_exe**: Equivalent to ansible_sudo_exe or ansible_su_exe, allows you to set the executable for the escalation method selected.

List of Behavioral Inventory Parameters

Remote host environment parameters:

- **ansible_shell_type**: The shell type of the target system.
- **ansible_python_interpreter**: The target host python path.
- **ansible_*_interpreter**: Works for anything such as ruby or perl and works just like ansible_python_interpreter.
- **ansible_shell_executable**: This sets the shell the ansible controller will use on the target machine, overrides executable in ansible.cfg which defaults to /bin/sh.

Examples from an Ansible-INI host file:

```
some_host         ansible_port=22     ansible_user=ubuntu
aws_host          ansible_ssh_private_key_file=/home/example/.ssh/aws.pem
freebsd_host      ansible_python_interpreter=/usr/local/bin/python
ruby_module_host  ansible_ruby_interpreter=/usr/bin/ruby.1.9.3
```

List of Behavioral Inventory Parameters wih Windows:

- ansible_connection: method connect to windows hosts (winrm)
- ansible_winrm_transport: basic authentication
- ansible_winrm_cert_pem: path to public key
- ansible_winrm_cert_key_pem: path to private key
- ansible_port: port connection, winrm http(5985) https(5986)

```
USERNAME="username"

cat > openssl.conf << EOL
distinguished_name = req_distinguished_name
[req_distinguished_name]
[v3_req_client]
extendedKeyUsage = clientAuth
subjectAltName = otherName:1.3.6.1.4.1.311.20.2.3;UTF8:$USERNAME@localhost
EOL

export OPENSSL_CONF=openssl.conf
openssl req -x509 -nodes -days 3650 -newkey rsa:2048 -out cert.pem -outform PEM -keyout cert_key.pem -subj "/CN=$USERNAME" -extensions v3_req_client
```

Module 3: Ansible-playbook

Introduction to YAML

- A YAML file is a text document that contains data formatted using YAML (YAML Ain't Markup Language), a human-readable data format used for data serialization.
- It is used for reading and writing data independent of a specific programming language. YAML files are often configuration files, used to define the settings of a program or application.
- All YAML files can optionally begin with “---” and end with “…”

```yaml
---
- hosts: master
  become: yes
  tasks:

    - name: Update
      shell: apt update -y

    - name: Start the cluster
      become: yes
      shell: kubeadm init --pod-network-cidr=192.168.10.0/24
      register: kubeadm_init

    - debug:
        msg: "{{ kubeadm_init.stdout }}"

    - name: create .kube directory
      become: yes
      become_user: ubuntu
      file:
        path: $HOME/.kube
        state: directory
        mode: 0755
```

Introduction to YAML

YAML type

```
---
# A list of tasty fruits
- Apple
- Orange
- Strawberry
-  Mango
...
```

```
# A dictionary is in a simple key: value
martin:
    name: Martin D'vloper
    job: Developer
    skill: Elite
```

Introduction to YAML

YAML type

Lists of dictionaries or mix both

```
- martin:
    name: Martin D'vloper
    job: Developer
    skills:
      - python
      - perl
      - pascal
- tabitha:
    name: Tabitha Bitumen
    job: Developer
    skills:
      - lisp
      - fortran
      - erlang
```

Dictionaries and lists can also be represented in an abbreviated form

```
---
martin: {name: Martin D'vloper, job: Developer, skill: Elite}
['Apple', 'Orange', 'Strawberry', 'Mango']
...
```

Specify a boolean value (true/false)

```
create_key: yes
needs_agent: no
knows_oop: True
likes_emacs: TRUE
uses_cvs: false
```

Introduction to YAML

Values can span multiple lines using | or >:

- Spanning multiple lines using a "Literal Block Scalar" | will include the newlines and any trailing spaces.
- Using a "Folded Block Scalar" > will fold newlines to spaces.

```
include_newlines: |
        exactly as you see
        will appear these three
        lines of poetry
```

```
fold_newlines: >
        this is really a
        single line of text
        despite appearances
```

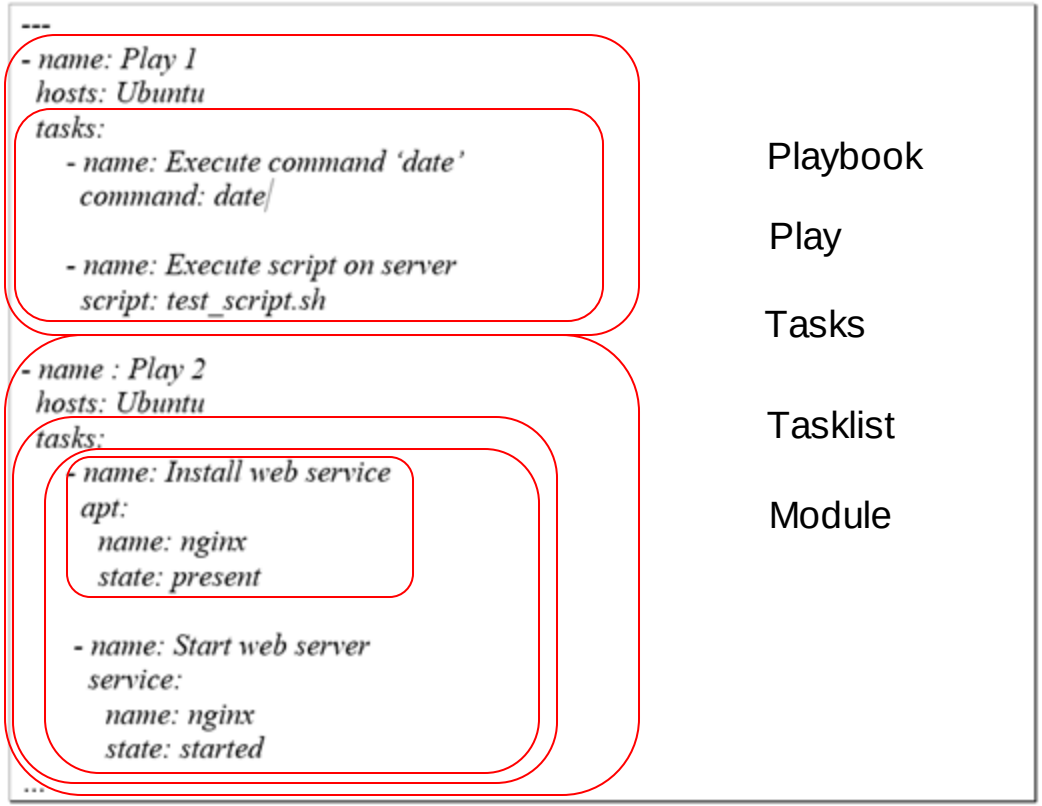
Playbook

- Playbook – A single YAML file
  - Play – Defines a set of activities (tasks) to be run on hosts
    - Task – An action to be performed on the host
      - ➢ Execute a command
      - ➢ Run a script
      - ➢ Install a package
      - ➢ Shutdown/Restart

playbook.yml

```yaml
---
- hosts: master
  become: yes
  tasks:

    - name: Update
      shell: apt update -y

    - name: Start the cluster
      become: yes
      shell: kubeadm init --pod-network-cidr=192.168.10.0/24
      register: kubeadm_init

    - debug:
        msg: "{{ kubeadm_init.stdout }}"

    - name: create .kube directory
      become: yes
      become_user: ubuntu
      file:
        path: $HOME/.kube
        state: directory
        mode: 0755
```

Playbook



- name: Play 1
hosts: Ubuntu
tasks:
- name: Execute command 'date'
command: date
- name: Execute script on server
script: test_script.sh
- name : Play 2
hosts: Ubuntu
tasks:
- name: Install web service
apt:
name: nginx
state: present
- name: Start web server
service:
name: nginx
state: started
...
Playbook
Play
Tasks
Tasklist
Module

Playbook

playbook

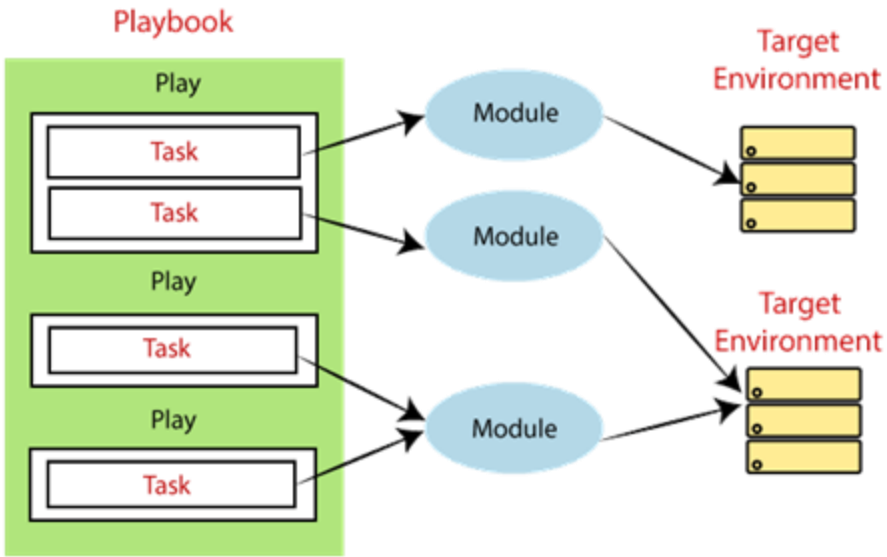
```yaml
---
- hosts: master
  become: yes
  tasks:
    - name: Update
      shell: apt update -y

- hosts: worker
  become: yes
  tasks:
    - name: Install Prometheus
      shell: |
          apt install prometheus -y
          systemctl enable prometheus
...
```

inventory

```
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 Modules

- ✓ There are over 1000 modules provided by Ansible to automate every part of the environment. Modules are like plugins that do the actual work in Ansible, they are gets executed in each playbook task.
- ✓ Each module is mostly standalone and can be written in a standard scripting language. One of the guiding properties of modules is idempotency, which means that even if an operation is repeated multiple times, it will always place the system into the same state



Modules

> ansible -m ping all

ansible – command line tool

ping – built-in Ansible module that does exactly what it says

all – all remote hostname in hosts file

Module examples

- copy – copy files from control node to target
- user – manage users and passwords
- package - install, update, remove tools using target package manager
- service – manage target system services using target init system
- firewalld – manage firewall configuration
- file – set permissions and ownership
- lineinfile – manage single lines on existing files
- command – allows for arbitrary commands, best practice is to avoid this

Ansible Modules Linux

common modules

```
- name: Install a list of packages
  apt:
    name:
     - nginx
      - postgresql
      - postgresql-server
    state: present
```

```
- name: Restart network service for interface eth0
  service:
    name: network
    state: restarted
    args: eth0
```

```
- name: Copy file with owner and permission,
using symbolic representation
  copy:
    src: scripts.sh
    dest: /home/ubuntu/
    owner: ubuntu
    group: ubuntu
    mode: ‘0644’
```

Ansible Modules Linux

common modules

```
- git:
    repo: https://github.com/ansible/ansible-examples.git
    dest: /src/ansible-examples
    separate_git_dir: /src/ansible-examples.git
```

```
- name: Install nginx
  apt:
    name: nginx
    state: present
    register: msg
- debug:
     msg: "{{ msg }}"
```

```
- name: Add a line to a file
  lineinfile:
    path: /etc/resolv.conf
    line: 192.168.1.99 foo.lab.net foo
    create: yes
```

Ansible Modules Linux

common modules

```
- name: configurable backup path
  cli_config:
    config: "{{ lookup('template', 'basic/config.j2') }}"
    backup: yes
    backup_options:
      filename: backup.cfg
      dir_path: /home/user
```

```
- name: Create a directory
  file:
    path: /etc/some_directory
    state: directory
    mode: '0755'
```

```
- name: Create a bz2 archive of multiple
files, rooted at /path
  archive:
    path:
    - /path/to/foo
    - /path/wong/foo
    dest: /path/file.tar.bz2
    format: bz2
```

Ansible Modules Windows

common modules

```
- name: Copy a single file
  win_copy:
    src: /srv/myfiles/foo.conf
    dest: C:\Temp\renamed-foo.conf
```

```
- name: Install git from specified repository
  win_chocolatey:
    name: git
    source: https://someserver/api/v2/
```

```
- name: Set an environment variable for all users
  win_environment:
    state: present
    name: TestVariable
    value: Test value
    level: machine
```

Ansible Modules Windows

common modules

```
- name: Set the owner of root directory
  win_owner:
    path: C:\apache
    user: SYSTEM
    recurse: no

- name: Set the log on user to a domain account
  win_service:
    name: service name
    state: restarted
    username: DOMAIN\User
    password: Password
```

```
- name: Example from an Ansible Playbook
  win_ping:
```

Refer:
https://docs.ansible.com/ansible/2.9/modules/list_of_windows_modules.html

Ansible Variables

- Variable stores information that varies with each host.
- Variable names should be letters, numbers, and underscores. Variables should always start with a letter.
  - ✓ foo_port is a great variable. foo5 is fine too.
  - ✓ foo-port, foo port, foo.port and 12 are not valid variable names.

```
1  #Sample Inventory File
2  web inter_ip_range=192.168.1.2
```

source: {{ inter_ip_range }} -> wrong
source: '{{ inter_ip_range }}' -> true

Conditionals

```
---
- name: Install nginx
  hosts: all
  tasks:
   - name: Install nginx on Debian
     apt:
        name: nginx
        state: present
     when: ansible_os_family == “Debian” and
              ansible_distribution_version ==”16.04”

   - name: Install nginx on Redhat
     yum:
        name: nginx
          state: present
     when: ansible_os_family == “Redhat” or
              ansible_os_family == “SUSE”
```

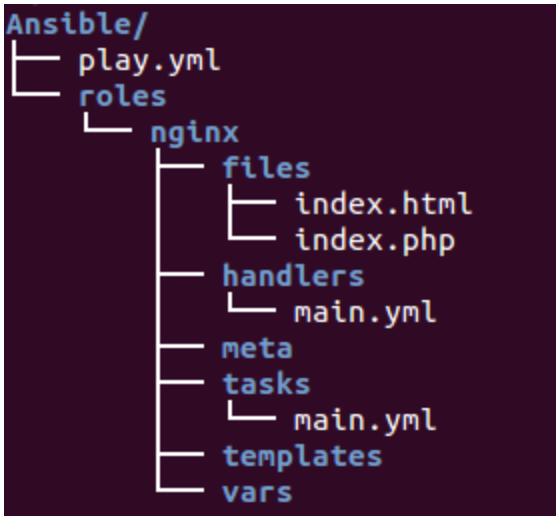
Conditionals in Loop

```
---
- name: Install Software
  hosts: all
  vars:
       packages:
            - name: nginx
              required: True
            - name: mysql
              required: True
            - name: apache
              required: False
  tasks:
  - name: Install “{{ item.name }}” on Debian
    apt:
       name: “{{ item.name }}”
       state: present
    when: item.required == True
    loop: “{{ packages }}”
```

Roles

- Roles are a way to group tasks together into one container. We could have a role for setting up MySQL, another one for configuring iptables etc.
- Roles makes it easy to configure hosts. Any role can be performed on any host or group of hosts such as:

```
- hosts: all
  roles:
    - nginx
```



Run your first Ansible playbook

Playbook.yml

```
---
- hosts: Ubuntu
  become: yes
  tasks:
     - name: Update
       shell: apt update -y
     - name: Install web service
       apt:
         name: nginx
         state: present
     - name: Start web server
       service:
         name: nginx
         state: started
...
```

```
hosts
server ansible_host=$IP ansible_user="ubuntu"
```

```
# check playbook
$ ansible-playbook –check path/playbook.yml
# run playbook
$ ansible-playbook –i path/hosts path/playbook.yml
```

Thank you