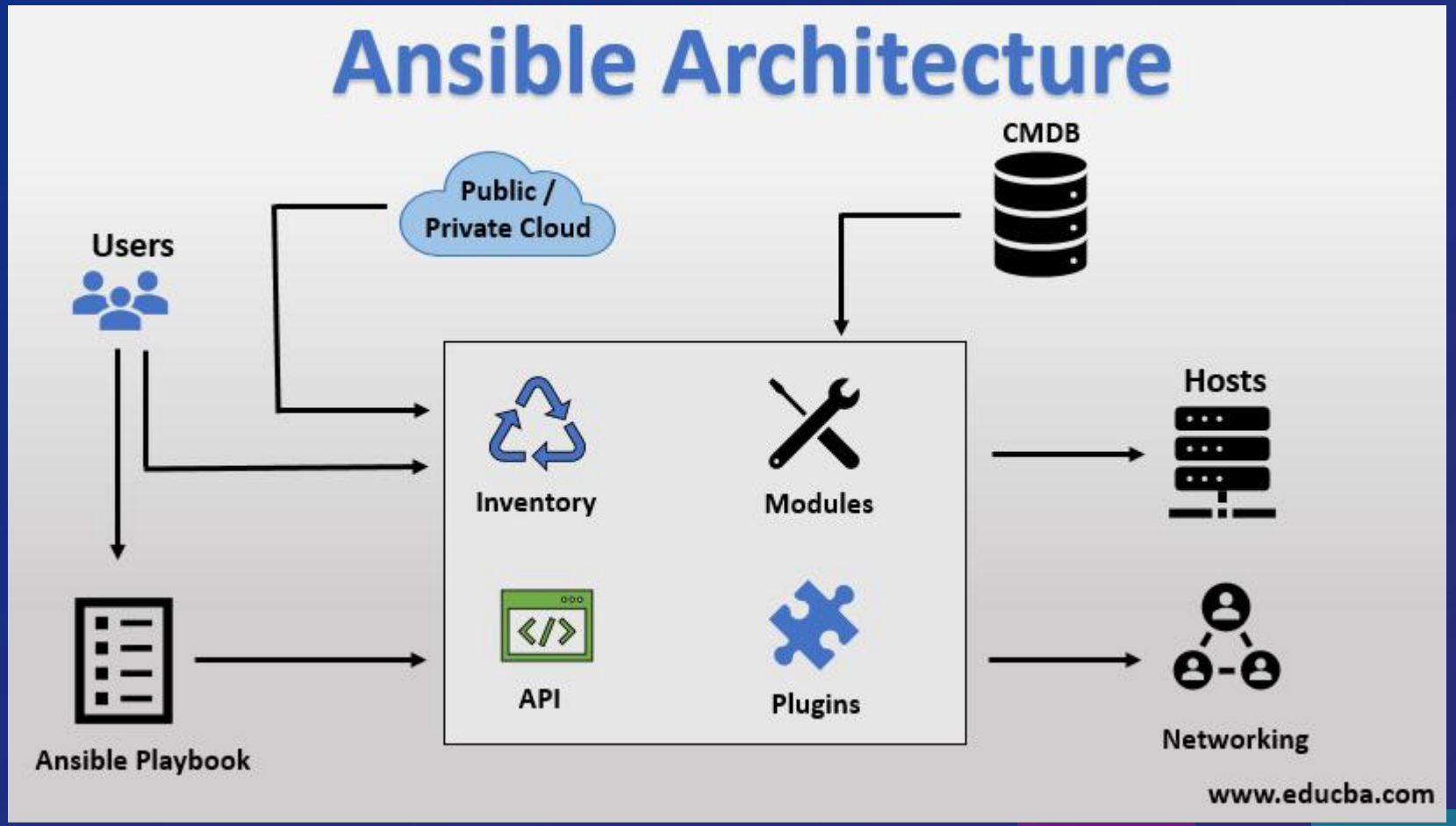

Automation for
thousands of hosts

SSH paramiko anywhere



Ansible + WSL2 +Vagrant

Windows (native) cannot be a controller

Prerequirement

01

WSL2

02

Vagrant

03

Ansible

04

Environment

Step by step setup

Step 01

Install WSL2

Step 02

Install ansible on controller

Step 03

Install vagrant on WSL2

Step 04

Edit Vagrantfile

Step 05

Edit inventory

Step 06

First try

Install WSL2

- Windows 11 integrated WSL2 as a Feature
  - Virtual Machine Platform
  - Windows SubSystem for Linux
- Install a ubuntu image by default

```
$ wsl --install
```

```
$ wsl --status
```

Install ansible

- Add python extension

```
$ pip3 install ansible
```

- version

```
$ ansible --version
```

Configure vagrant env

- Windows Access

```
export VAGRANT_WSL_ENABLE_WINDOWS_ACCESS="1"
```

- Path modifications

```
export PATH="$PATH:/mnt/c/Program Files/Oracle/VirtualBox"
```

Note: append cmd.exe and powershell path too

Share boxes with windows(optional)

- Windows
  `c:\Users\yangwawa\.vagrant.d\boxes`
- Linux WSL2
  `~/.vagrant.d/boxes`
- Custom Windows system home path

```
export
VAGRANT_WSL_WINDOWS_ACCESS_USER_HOME_PATH="/mnt/c/User
s/yangwawa"
```

Vagrant + VirtualBox

- Install vagrant plugin

```
$ vagrant plugin install virtualbox_WSL2
```

Vagrantfile

- Initialize a Vagrantfile with Centos7 box.

```
vagrant init generic/centos7
```

- Add public manageable network

```
config.vm.network "public_network", ip: "192.168.0.101"
config.vm.hostname  = "server01"
```

Inventory

- mapping the hostname/domain in /etc/hosts
- `WinIni` / `yml` format
- [all] and [ungrouped]
- for example:

```
inspecthost01
[webserver]
server01 ansible_user=vagrant
server02
server03
```

```
$ ansible-inventory -i inventory --list --yaml
```

Ansible.cfg

- generate ansible.cfg

```
$ ansible-config  init --disabled > ansible.cfg
```

- Preference (highest to lowest)
  - ANSIBLE_CONFIG
  - ./ansible.cfg
  - ~/.ansible.cfg
  - /etc/ansible/ansible.cfg

First try

- Testing ping/pong

```
$ ansible -i inventory -m ping
```

- Login with public/private key only

```
--private-key  --key-file
$ ansible --key-file=~/private_key-learning-
ansible learning -m ping -u vagrant
```

ansible command

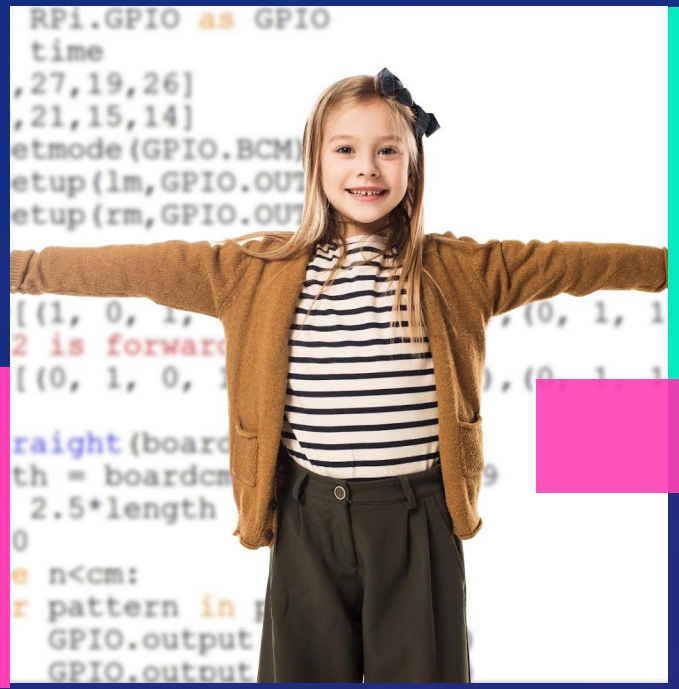
ansible-doc

- Module docs

```
ansible-doc -l | grep builtin.file
ansible-doc file
ansible machinename -m file -a
'path=/tmp/testing state=absent'
```

Playbook

Write a playbook file



/etc/wsl.conf

```
[automount]
enabled=true
root = /
options = "metadata,umask=0033"
```

ansible-playbook

```
ansible-playbook -i inventory --key-
 file .vagrant/machines/default/virtualbox/
 private_key  playbook.yml

-k  --ask-pass
-K  --ask-sudo
```

playbook format

- target section
- variable section (optional)
- task section
- handler section (optional)

playbook.yml example

```
---
- hosts: webservers
  user: root
  vars:
      apache_version: 2.6
      motd_warning: 'WARNING: Use by ACME Employees ONLY'
      testserver: yes
  tasks:
      - name: setup a MOTD
        copy:
           dest: /etc/motd
           content: "{{ motd_warning }}"
```

target section

- hosts: *webservers*
  user: *vagrant*
  sudo / become : yes
  sudo_user / become_user

vars section

```
vars:
 apache_version: 2.6
 motd_warning: 'WARNING: Use by ACME
 Employees ONLY'
 testserver: yes
```

include vars file

```
vars_files:
 conf/country-AU.yml
 conf/datacenter-SYD.yml
 conf/cluster-mysql.yml
```

task section

```
tasks:
 - name: install apache
   action: yum name=httpd state=installed
 - name: configure apache
   copy: src=files/httpd.conf \
 dest=/etc/httpd/conf/httpd.conf
 - name: restart apache
   service:
       name: httpd
       state: restarted
```

handler section

- One handler can only call one module
- Notify can trigger multiple handler
- Only changed event/fact can trigger handler

```
- name: update to latest DHCP
  yum
    name: dhcp
    state: latest
    notify: restart dhcp
- name: copy the DHCP config
  copy:
    src: dhcp/dhcpd.conf
    dest: /etc/dhcp/dhcpd.conf
    notify: restart dhcp
```

```
handlers:
- name: restart dhcp
  service:
    name: dhcpd
    state: restarted
```

useful Playbook module

Most common used ansible playbook module

debug module

```
- name: print out system gateway
  debug:
     msg: “System {{ inventory_hostname }} has
 gateway {{ ansible_default_ipv4.gateway }}”
```

setup module

- collect every fact/info from remote host

tempalte module

- Jinja2 template

```
---
- name: Setup BIND
  host: allnames
  tasks:
    - name: configure BIND
      template: src=templates/named.conf.j2 \
dest=/etc/named.conf \
owner=root group=named mode=0640
```

- comment

```
  {# Variables for zone config #}
```

template if

```
{% if 'authorativenames' in group_names %}
 {% set zone_type = 'master' %}
 {% set zone_dir = 'data' %}
{% else %}
 {% set zone_type = 'slave' %}
 {% set zone_dir = 'slaves' %}
{% endif %}
```

template for loop

```
{% for ip in ansible_all_ipv4_addresses %}
 {{ ip }};
{% endfor %}
```

set_fact module

- gather machine fact and use in template or as variable

```
ansible -i inventory -m setup
```

- for example:

```
- name: Calculate InnoDB buffer pool size
  set_fact:
    innodb_buffer_pool_size_mb  : \
"{{ ansible_memtotal_mb/2 }}"
```

pause module

- use debug

```
- hosts: localhost
  tasks:
    - name: wait on user input
      pause:
        prompt: "Warning! Press ENTER to continue or CTRL-C to quit."
    - name: timed wait
      pause:
        seconds: 30
```

wait_for module

- poll a particular TCP port and not continue until that port accepts a remote connection

```
- name: Start Tomcat
  service:
    name: tomcat7
    state: started
```

```
- name: Wait for Tomcat to start
  wait_for:
    port: 8080
    state: started
```

assemble module

- combines several files

```
- name: Build the authorized_keys file
  assemble:
      src: /opt/sshkeys
      remote_src: false
      dest: /root/.ssh/authorized_keys
      owner: root
      group: root
      mode: 0700
```

slurp module

- grabs a file from the remote system register as var
- encodes it with base 64 (in template {{ *sshkey.content* | b64decode }})
- load into memory (no use for large file)

```
- name: Fetch a SSH key from a machine
  hosts: bastion01
  tasks:
    - name: Fetch key
      slurp:
         src: /root/.ssh/id_rsa.pub
      register: sshkey
```

ansible.builtin.iptables

```
- name: Insert a rule on line 5
  ansible.builtin.iptables:
    chain: INPUT
    protocol: tcp
    destination_port: 8080
    jump: ACCEPT
    action: insert
    rule_num: 5
```

module example

ansible.builtin.file

```
- name: Change file ownership, group and
  permissions
    ansible.builtin.file:
      path: /etc/foo.conf
      owner: foo
      group: foo
      mode: '0644'
```

ansible.builtin.file

```
- name: Create a directory if it does not
  exist
    ansible.builtin.file:
      path: /etc/some_directory
      state: directory
      mode: '0755'
```

ansible.builtin.file

```
- name: Touch again the same file, but do not change
  times this makes the task>
   ansible.builtin.file:
     path: /etc/foo.conf
     state: touch
     mode: u+rw,g-wx,o-rwx
     modification_time: preserve
     access_time: preserve
```

ansible.builtin.git

```
- name: checkout Qroud
  git:
    repo: 'https://github.com/yangwawa0323/learning-ansible.git'
    dest: /opt/apps/Qroud force=no
    update: no
```

shell module

- Copy the shell script to remote host and run
- Simplest bash command
- The args is same level as shell *****

```
- name: Change the working directory to somedir/
before executing the command
  shell: somescript.sh >> somelog.txt
  args:
    chdir: somedir/
```

replace module

```
- name: Replace old hostname with new
  hostname (requires Ansible >= 2.4)
    ansible.builtin.replace:
      path: /etc/hosts
      regexp:
  '(\s+)old\.host\.name(\s+.*)?$'
      replace: '\1new.host.name\2'
```

Advanced

Running operations in parallel

- use the async and poll keywords.
  - async: max timeout
  - poll: check interval

```
tasks:
  - name: Install mlocate
    yum: name=mlocate state=installed
  - name: Run updatedb
    command: /usr/bin/updatedb
    async: 300
    poll: 10
```

with_items loop

```
- name: Copy SSH keys over
    copy:
        src: "keys/{{ item }}.pub"
        dest: "/opt/sshkeys/{{ item }}.pub"
        owner: root
        group: root
        mode: 0600
    with_items:
        - dan
        - kate
        - mal
```

with_fileglob loop

```
- name: Upload public keys
  copy:
    src: "{{ item }}"
    dest: /root/.sshkeys
    mode: 0600
    owner: root
    group: root
  with_fileglob:
    - keys/*.pub
```

with_sequnce loop

```
with_sequence: start=1 end={{ hostcount }} format=webapp%02x
- name: Start GCE Nodes
  gce:
     image: centos-6
     name: "mysql-{{ item }}"
     tags: mysql
     zone: us-central1-a
  with_sequence: count=2
  register: nodes
```

Conditional execution

```
Note: use pause module debug
- name: Install VIM via yum
  yum:
      name: vim-enhanced
      state: installed
  when: ansible_os_family == "RedHat"
- name: Install VIM via apt
  apt:
     name: vim
     state: installed
  when: ansible_os_family == "Debian"
```

Task delegation

```
- name: Get config
  get_url:
    dest: "configs/{{ ansible_hostname }}"
    force: yes
    url:"http://{{ ansible_hostname }}/diagnostic/config"
  delegate_to: localhost
```

Extra variables

- The hostvars variable
  - {{ hostvars.*ns1*.ansible_default_ipv4.address }}
- The groups variable

```
- name: Create a user for all app servers
  with_items: “{{ groups[‘appservers’] }}”
  mysql_user:
     name: kate
     password: test
     host: "{{ hostvars[item].ansible_eth0.ipv4.address }}"
     state: present
```

Finding files with variables

```
- name: Get the best match for the machine
  copy:
     dest: /etc/apache.conf
     src: "{{ item }}"
  with_first_found:
     - "files/apache/{{ ansible_os_family }}-
  {{ ansible_architecture
}}.cfg"
     - "files/apache/default-{{ ansible_architecture }}.cfg"
     - files/apache/default.cfg
```

Environment variables

- any module can combined with environment

```
- name: Install cobbler
      ansible.builtin.package:
        name: cobbler
        state: present
      environment:
        http_proxy: http://proxy.example.com:8080
```

External data lookups

- lookup for localhost env

```
  - name: Download file
    get_url:
        dest: /var/tmp/file.tar.gz
        url: http://server/file.tar.gz
    environment:
        http_proxy: "{{ lookup('env',
'http_proxy') }}"
```

Storing data

- Every module output somethings
- register to var object

```
  - name: Get /tmp info
    file:
          dest: /tmp
          state: directory
     register: tmp
   - name: Set mode on /var/tmp
     file:
          dest: /tmp/subtmp
          mode: "{{ tmp.mode }}"
          state: directory
```

Processing data

- Jinja2 filters
- can apply multiple filters

```
- name: Create accounts
  user: name={{ item | lower }} state=present
  with_items:
     - Fred
     - John
     - DanielH
```

Debugging playbooks

- debug module
- --verbose,
  - -vv
  - -vvvv
- pause module

Large Project

Includes

- Variable includes
- Playbook includes
- Task includes
- Handler includes

Tasks include

- include is deprecated. Now is upgrade to include_tasks module

```
---
- hosts: ansibletest
  user: root
  tasks:
    - name: include usersetup.yml
      include_task:
         file: usersetup.yml
      vars:
        - user : “{{ item }}”
      with_items:
        - mal
        - dan
        - kate
```

```
# usersetup.yml
---
# Requires a user variable to specify
user to setup
- name: Create user account
  user:
     name: "{{ user }}"
     state: present
```

Handlers include

```
 hosts: mailers
 tasks:
 - name: update sendmail
   yum:
      name: sendmail
      state: latest
   notify: restart sendmail
handlers:
 - include_tasks: sendmailhandlers.yml
```

```
# sendmailhandlers.yml
---
- name: config aliases
  command: newaliases
  notify: reload sendmail
- name: reload sendmail
  service:
     name: sendmail
     state: reloaded
- name: restart sendmail
  service:
     name: sendmail
     state: restarted
```

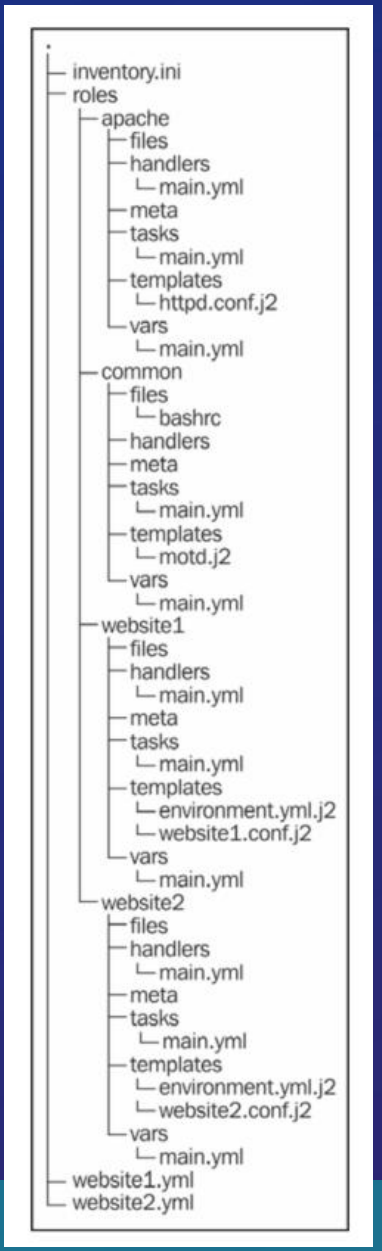
Playbook includes

```
- name: Include a play after another play
  import_playbook: otherplays.yaml

- name: Set variables on an imported playbook
  import_playbook: otherplays.yml
  vars:
    service: httpd
```

Roles

```
- name: Setup servers for website1.example.com
  hosts: website1
  roles:
      - common
      - apache
      - role: website1
        port: 80
```



Roles

- by default look in each directory main.yml file
- generate the empty role directory structure

  ```
  ansible-galaxy role init rolename
  ```

- copy module look in roles/*rolename*/files
- scripts also look in roles/*rolename*/files
- template module look in roles/*rolename*/templates

Roles default

- defaults/main.yml
- overridden by variables in the vars/main.yml file

```
---
port: 80
```

tags

- select partial of tasks to run

```
- name: install latest software
   yum:
      name:  apache
      state: latest
      notify: restart apache
   tags:
      - patch
```

- List and run tagged playbook

```
$ ansible-playbook webserver.yml --list-tags
$ ansible-playbook webservers.yml --tags deploy --tags ...
```

tags in roles

- tags can apply in roles

```
---
- hosts: website1
  roles:
      - common
      - { role: apache, tags: ["patch"] }
      - role: website2
        tags:
          - deploy
          - patch
```

Secrets

include sensitive data in your Ansible

Store secrets

- ansible-vault

```
$ ansible-vault create vars/staging.yml
$ ansible-vault encrypt vars/staging.yml
$ ansible-vault rekey vars/staging.yml
```

- --ask-vault-pass

```
$ ansible-playbook --ask-vault-pass encrypted.yml
```

- --vault-password-file
- vault_password_file to the ansible.cfg file