# Success your DevOps tasks With Ansible

HAMIDA TRIMECHE

# Agenda

1. Configuration management tools
2. Ansible:
    1. Inventory
    2. Playbook
    3. Variables
3. Template module (Jinja2)
4. Roles

# 1 Configuration management tools

DevOps is evolving and gaining traction as organizations discover how it enables them to produce better applications and reduce their software products' time to market. DevOps' core values are Culture, Automation, Measurement, and Sharing (CAMS), and an organization's adherence to them influences how successful it is.

**Culture**

brings people and processes together

**Automation**

creates a fabric for Dev

**Measurement**

permits improvements

**Sharing**

enables the feedback loop in the CAMS cycle

Another DevOps concept is the idea that almost everything can be managed in code: servers, databases, networks, log files, application configurations, documentation, automated tests, deployment processes, and more.

# advantage of ansible:


## Free

Ansible is an open-source tool. Very simple to set up and use: No special coding skills are necessary to use Ansible's playbooks (more on playbooks later).

## Powerful

Ansible lets you model even highly complex IT workflows.

## Flexible

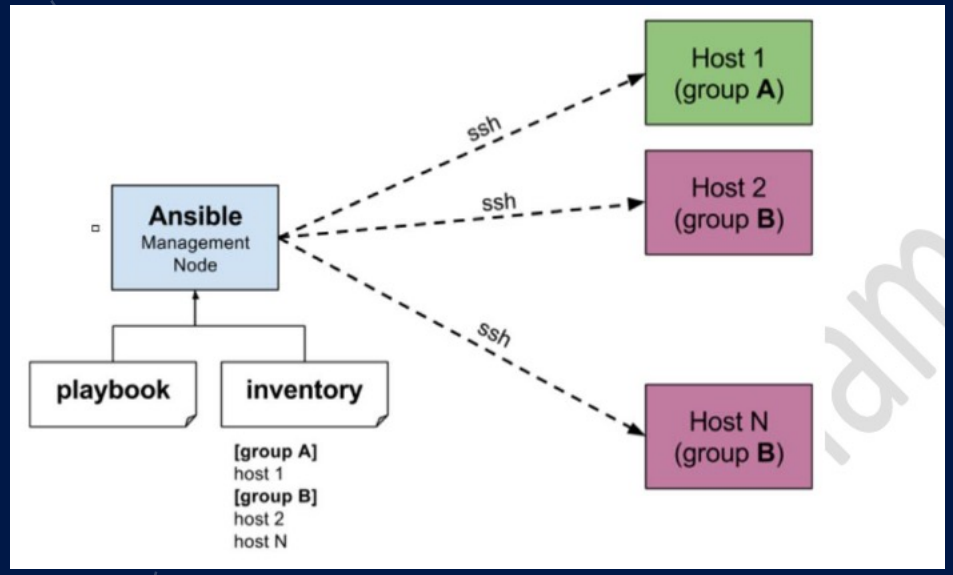
You can orchestrate the entire application environment no matter where it's deployed. You can also customize it based on your needs.

## Agentless

You don't need to install any other software or firewall ports on the client systems you want to automate. You also don't have to set up a separate management structure.

# How Ansible work

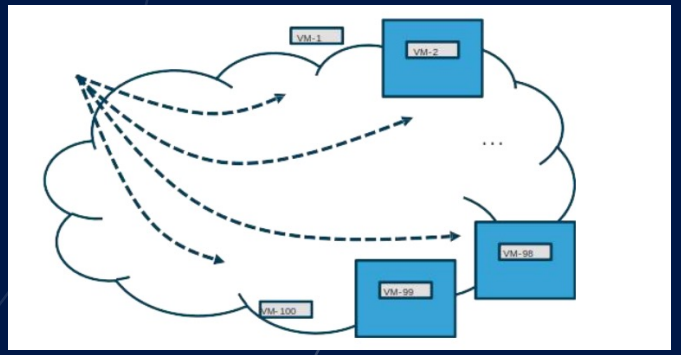
Ansible works by connecting to your nodes and pushing out small programs, called "Ansible Modules" to them. Ansible then executes these modules (over SSH by default) and removes them when finished. Your library of modules can reside on any machine, and there are no servers, daemons, or databases required.



# 2 Ansible

## Why Ansible: Real-Time Remote Execution of commands

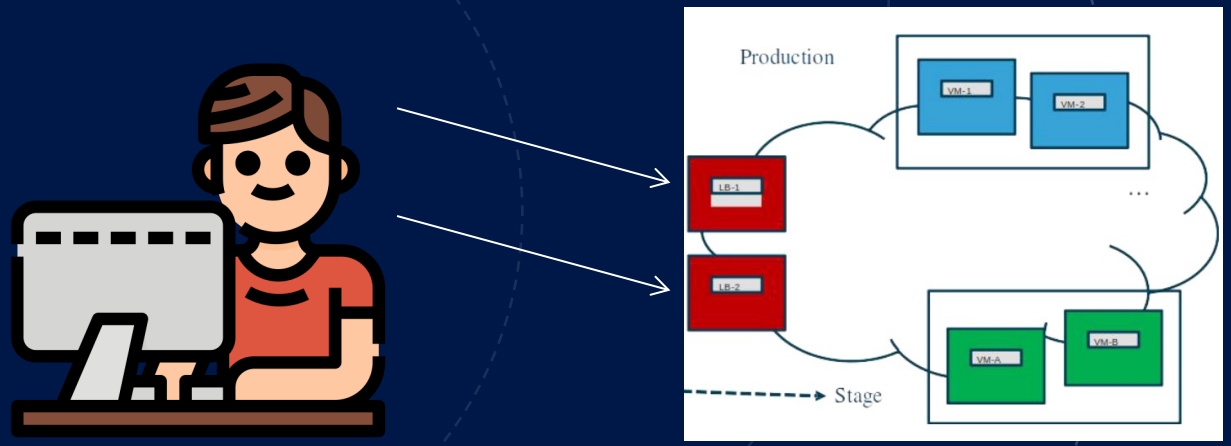
1.Audit routes on all virtual machines: $ ansible -m shell -a “netstat -rn” datacenter-east




2.Updates routes required for consistency: $ ansible -m shell -a “route add X.X.X.X” datacenter-east

# Why Ansible: Change Control Workflow Orchestration

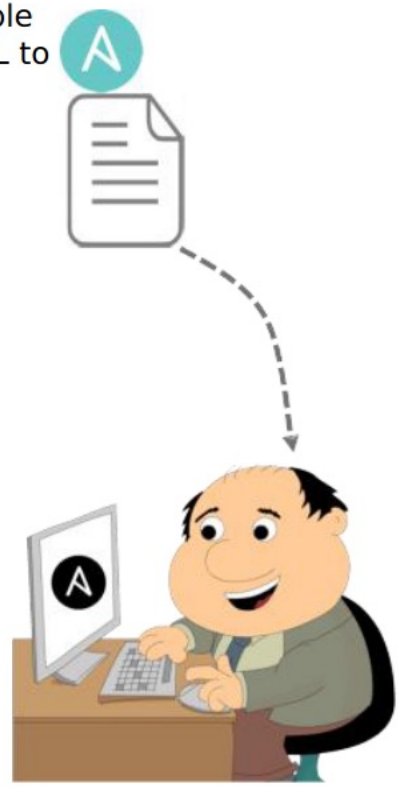
1. Update load balancer pools to point to stage



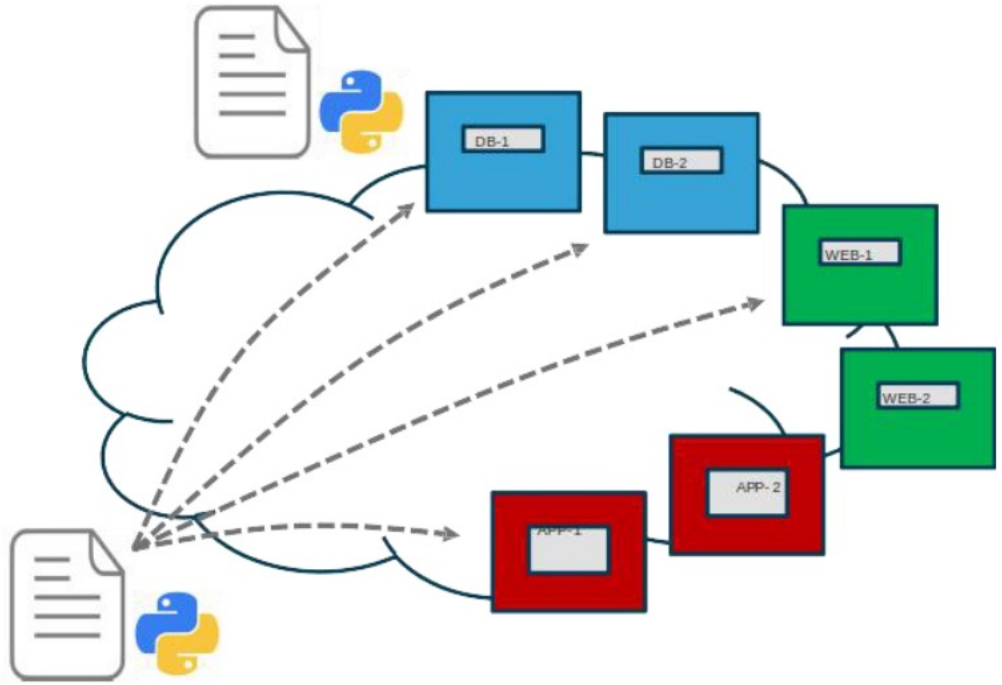

2. Deploy application change to stage and verify

# How does Ansible work

1. Engineers deploy Ansible playbooks written in YAML to a control station



Ansible Control Station




2. Ansible copies modules typically written in Python to remote hosts to execute tasks

# Ansible Configuration File

- Linux host with a Python and the Ansible installed
- Support transport to remote hosts
    - Typically SSH but could use an API
- Ansible Components
    - Ansible configuration file
    - Inventory files
    - Ansible modules
    - Playbooks

# Control operation of Ansible

- Control operation of Ansible
- Default configuration
    - /etc/ansible/ansible.cfg
- Override default settings
    - ANSIBLE_CONFIG ENV
    - ansible.cfg in current directory
    - .ansible.cfg in home directory

```
DevNet$ cat ansible.cfg

# config file for ansible
# override global certain global settings

[defaults]
# default to inventory file of ./hosts  inventory
= ./hosts

# disable host checking to automatically add  # hosts to
known_hosts
host_key_checking = False

# set the roles path to the local directory  roles_path
= ./
```

# Ansible Authentication Basics

Typically, Ansible uses SSH for authentication and assumes keys are in place

- Setting up and transferring SSH keys allows playbooks to be run automatically
- Using passwords is possible
- Network Devices often use passwords

# Ansible Inventory File

Inventory file identifies hosts, and groups of hosts under management

- Hosts can be IP or FQDN
- Groups enclosed in

- Can include host specific parameters as well
- Example: Instructing Ansible to use the active Python Interpreter when using Python Virtual Environments : $ansible_python_interpreter="/usr/bin/env python

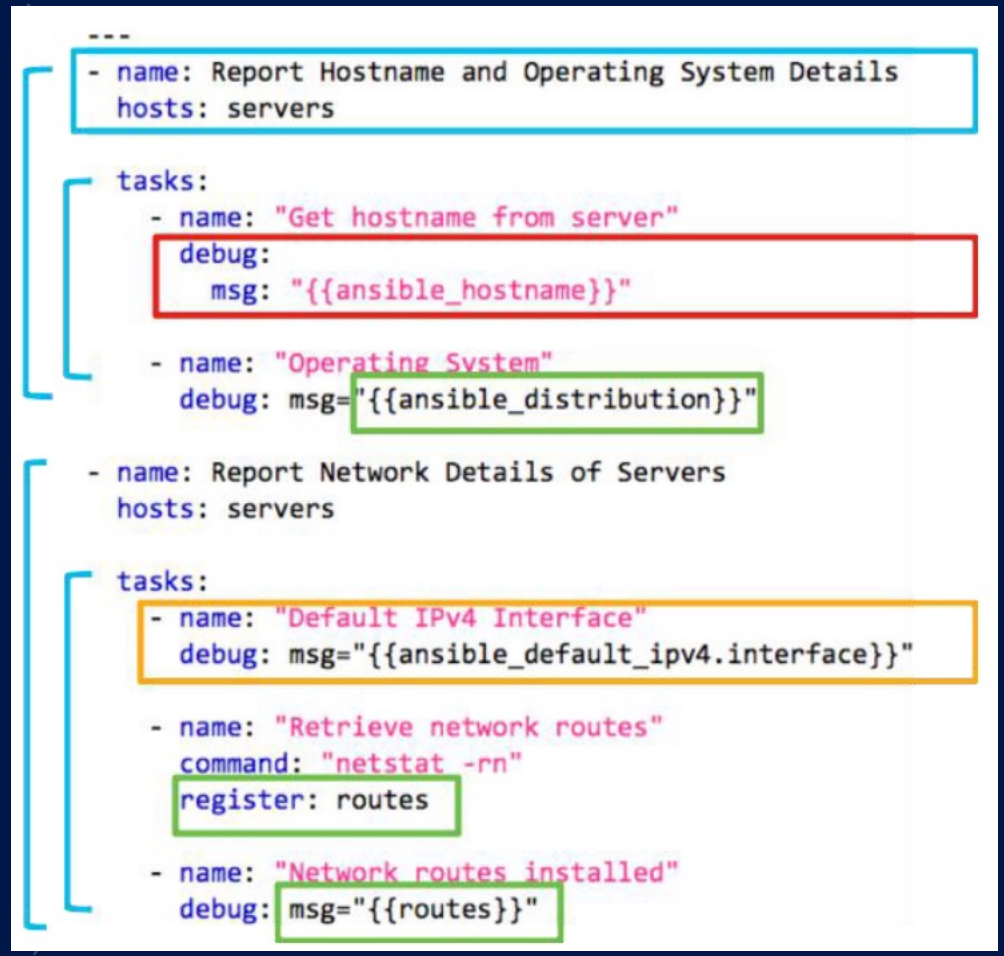
# Using Ansible CLI for ad-hoc Commands

- Quickly run a command against a set of hosts
- Specify the module with –m module
- Specfiy the username to use with –u user, default is to use local username
- Specify the server or group to target
- Provide module arguments with –a argument

```
DevNet$ ansible -m setup -u root servers

10.10.20.20 | SUCCESS => {
    "ansible_facts": {  "ansible_all_ipv4_addresses": [
        "10.10.20.20",
        "172.17.0.1"
    ],
        "ansible_all_ipv6_addresses":
            [  "fe80::250:56ff:febb:3a3f"
    ],
        "ansible_apparmor": {  "status":
            "disabled"
    },
    "ansible_architecture": "x86_64",
.
.
```

# Ansible playbook

- Written in YAML
- One or more plays that contain hosts and tasks
- Tasks have a name & module keys.
- Modules have parameters
- Variables referenced with {{name}}
    - Ansible gathers “facts”
    - Create your own by register-ing output from another task



# Ansible playbook

```
DevNet$ ansible-playbook -u root example1.yaml

PLAY [Report Hostname and Operating System Details]
*********************************************************************************************

TASK [Gathering Facts]
*********************************************************************************************** ok: [10.10.20.20]

TASK [Get hostname from server]
*********************************************************************************************** ok: [10.10.20.20] => {
    "msg": "localhost"
}

PLAY [Report Network Details of Servers]
*********************************************************************************************

TASK [Network routes installed]
*********************************************************************************************** ok: [10.10.20.20] => {
        "stdout_lines": [
                        "Kernel IP routing table",









            "Destination          Gateway             Genmask             Flags       MSS Window      irtt  Iface",
            "0.0.0.0              10.10.20.254        0.0.0.0             UG            0 0             0  ens160",
            "10.10.20.0           0.0.0.0             255.255.255.0       U             0 0             0  ens160",


            "172.16.30.0          10.10.20.160        255.255.255.0       UG            0 0             0  ens160",
        ]

PLAY RECAP
*********************************************************************************************************************
*******************
10.10.20.20                : ok=7    changed=1    unreachable=0    failed=0
```

# Using Variable Files and loops with Ansible

- Include external variable files using
- vars_files: filename.yaml
- Reference variables with {{name}}
- YAML supports lists and hashes (ie key/value)
- Loop to repeat actions with with_items: variable

example2_vars.yaml

```
---
company_name: "DevNet"
quotes:
  - "DevNet Rocks!"
  - "Programmability is amazing"
  - "Ansible is easy to use"
  - "Lists are fun!"
```

```
---
- name: Illustrate Variables
  hosts: servers
  gather_facts: false
  vars_files:
    - example2_vars.yaml

  tasks:
    - name: "Print Company Name from Variable"
      debug: msg="Hello {{company_name}}"

    - name: "Loop over a List"
      with_items: "{{quotes}}"
      debug: msg="{{item}}"
```

# 3 Template module (Jinja2)

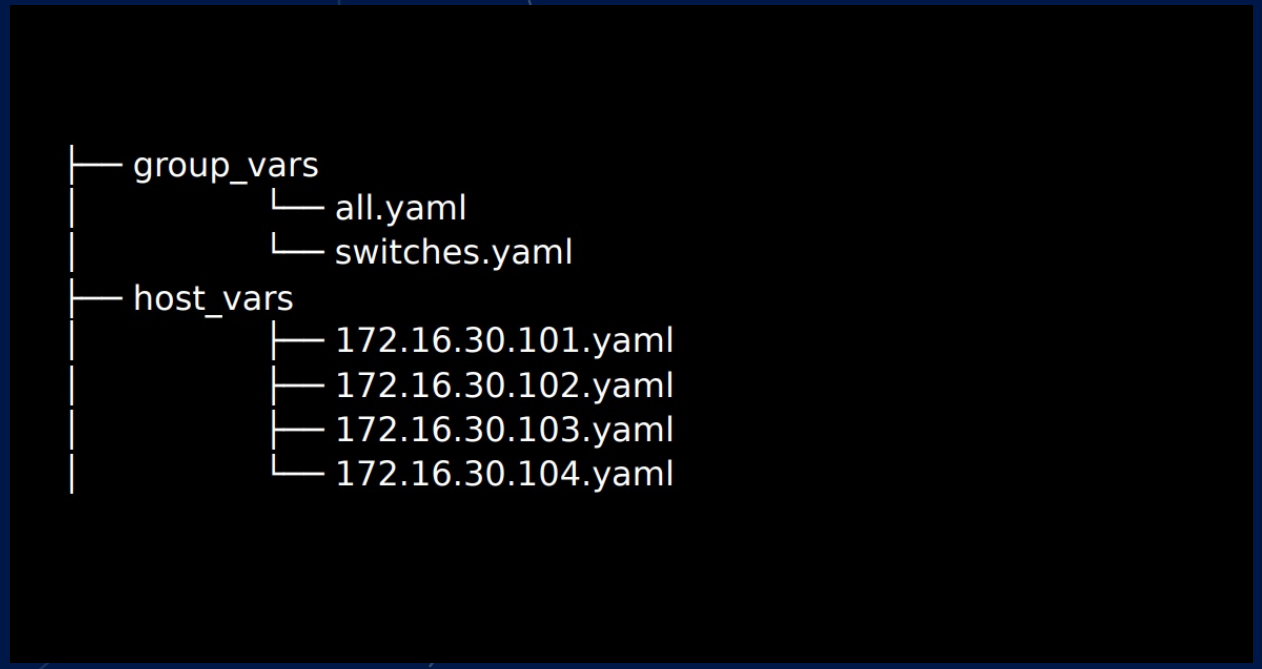
```
DevNet$ ansible-playbook -u root example3.yaml
PLAY [Generate Configuration from Template]
*******************************

TASK [Generate config]
*************************************************
changed: [localhost]

PLAY RECAP
*************************************************************
*******  localhost          : ok=1       changed=1
                unreachable=0          failed=0

DevNet$ cat example3.conf  feature bgp
router bgp 65001
router-id 10.10.10.1
```

# Jinja2 Templating – Variables to the Max!

```
DevNet$ ansible-playbook -u root example3.yaml
PLAY [Generate Configuration from Template]
*******************************

TASK [Generate config]
*************************************************
changed: [localhost]

PLAY RECAP
*************************************************************
*******  localhost          : ok=1       changed=1
               unreachable=0          failed=0

DevNet$ cat example3.conf  feature bgp
router bgp 65001
router-id 10.10.10.1
```

# Jinja2 Templating – Variables to the Max!

- Ansible allows for Group and Host specific variables
  - group_vars/groupname.yaml
  - host_vars/host.yaml
- Variables automatically available



# 4 Roles

## Using Ansible Roles

- Roles declares any playbooks defined within a role must be executed.

```
- hosts: dcloud-servers
  roles:
    - { role: geerlingguy.apache }
```

against hosts

Roles promote playbook reuse

```
$ cd geerlingguy.apache/
$ ls
LICENSE   README.md defaults  handlers  meta      tasks     templates tests     vars
```

Roles contain playbooks, templates, and variables to complete a workflow (e.g. installing Apache

# Thank You

HAMIDA TRIMECHE