Ansible

As automation tool

DigitalOnUs
Accelerating Digital Business

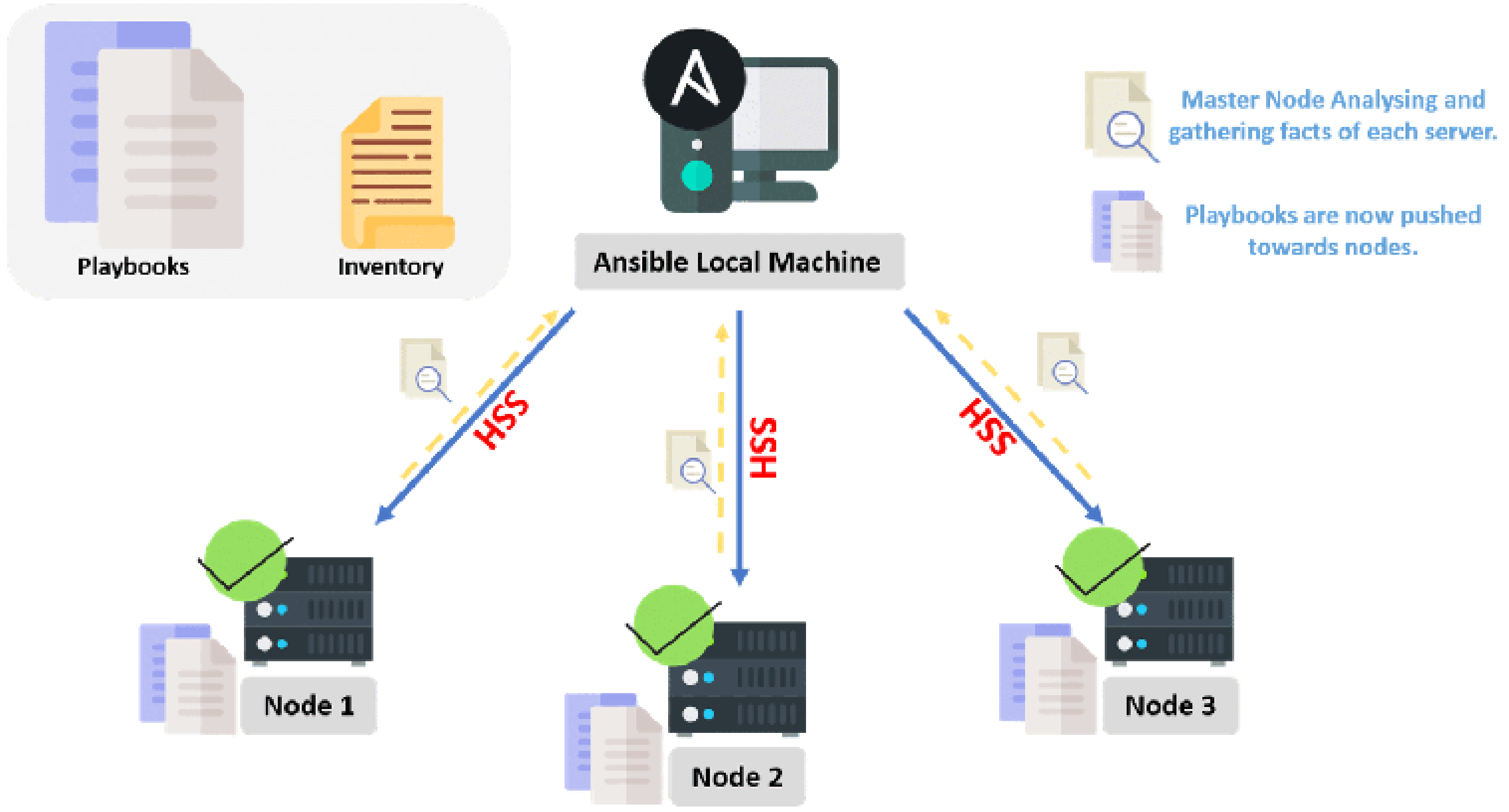
Agenda

- Ansible basics
- Idempotency and orchestration
- Syntax
- Installation Mac / GNU Linux
- More theory
- Live examples
- Preparing environment
- Exercises
- Learning material

What is Ansible?

- Tool for:
  - Server provisioning
  - Application deployment.
  - Any IT task with “Plays” and “Playbooks” using idempotency and orchestration, along with consistency.
  - Uses paramiko, a powerful python module to generate a ssh connection under hosts.

Ansible´s structure


Playbooks
Inventory
Ansible Local Machine
Master Node Analysing and gathering facts of each server.
Playbooks are now pushed towards nodes.
SSH
SSH
SSH
Node 1
Node 2
Node 3

Idempotency

- Operation is idempotent if the result of performing once is exactly the same as the result of performing it repeatedly without any intervening actions.

Orchestration

- **<u>For ansible, orchestration is </u>**the handler for an orchestra. Datacenters playing many parts; web, db, load b., monitoring servers all of them need to be touched in particular order.
- Some systems may perform some steps, then others. Other systems may processed other steps. Ansible orchestration tries to model that system steps.

Ansible playbook syntax

- Playbooks are formed by one of more “plays”.

-
  ```
  - name: play 1
     hosts: all
     become: true
      pre_tasks:
         name: do something before roles
         debug: msg="this is run before a role"
  ```

Installation

- Create 2 virtual machines, (preferent CentOS 7+) under virtualbox.
  - 2 networks, 1 as bridge and other as NAT.
- Create your user with your name and elevate user for sudo access.
- Use your own machine to install ansible, using pip or your prefered engine.
  - yum –y install ansible
  - pip install ansible

Inventory

- File used to place host, by hostname OR IP.
- When ansible is run with ansible-playbook and ansible adhoc mode, inventory file is needed.

```
[webservers]
foo.example.com
bar.example.com

[dbservers]
one.example.com
two.example.com
three.example.com
```

How to limit inventory

- Ansible has different options to limit the playbook execution only for a few nodes:
  - NOT recommended:
    - --limit “groupnme”
  - Recommended:
    - Limit into playbook

ADHOC

- Run ansible modules in one line.
- ansible all –i “ip-INV file” –m “module name” syntax…

Ansible Playbook

- Playbooks are the language by which Ansible orchestrates, configures, administers, or deploys systems. They are called playbooks partially because it's a sports analogy, and it's supposed to be fun using them. They aren't workbooks :)
- Same ideology as puppet.

Example of playbook

```
---
- hosts: all
  remote_user: whatever_you_want
  become: yes
  vars_prompt:
    - name: "INC"
      prompt: "INC # "
      private: no
  tasks:
  - name: Get timestamp from server.
    shell: "date +%d%m%Y"
    register: time_lnx
```

Tasks

- Task are executed in order, one by one. If some of the nodes are not reachable , it will be rejected from the list and the execution won´t stop.
- Every task should have a name, that must be DESCRIPTIVE according to the main job.

More about tasks

- What happen if some task are not logical correct or fails? Let´s check in a live example:

Tasks: Use of notify

- notify is an special instruction under some tasks. They will be executed only one time, doesn´t matter if notify is called 10 or 1000 times.

```
notify:
     - restart apache
```

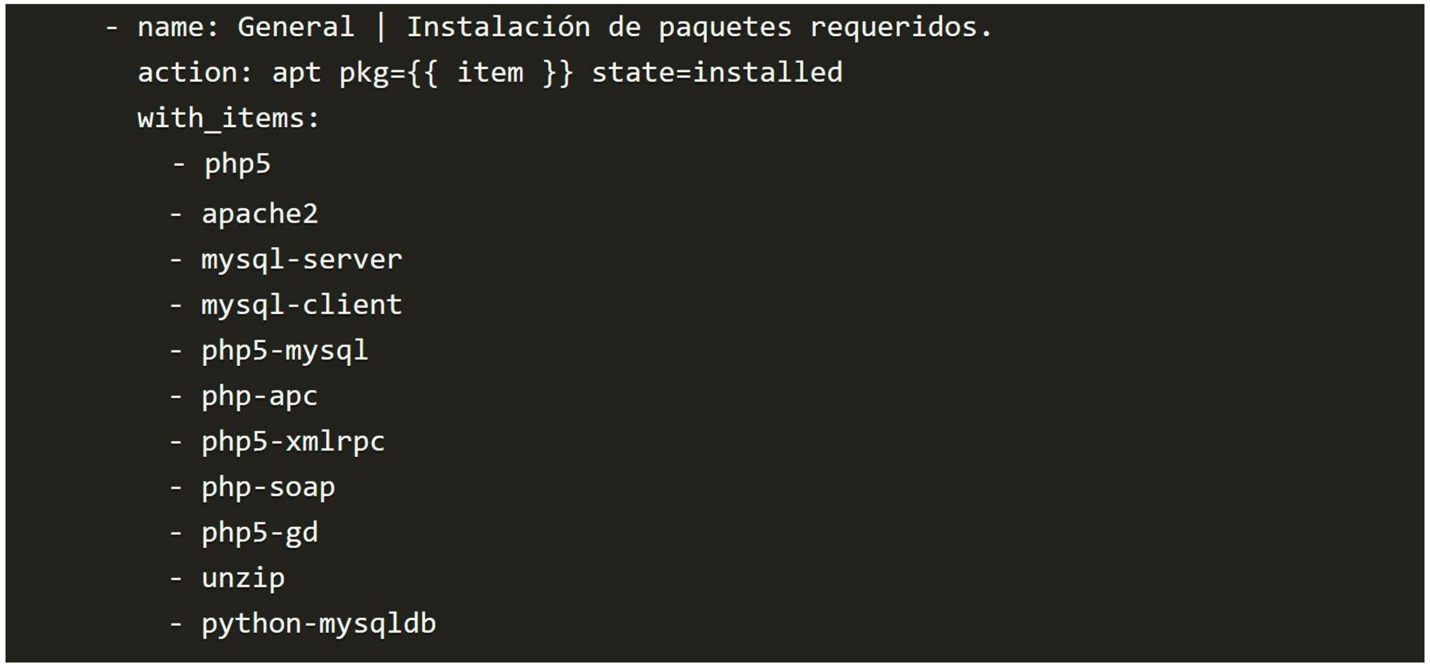
Notify as a handler

- notify will be executed only if the file/daemon had changes until the execution.
    - Differences to do in this way vs writing task that restart the service?

Task: Item

- Some times, we will need install a list of packages, or do some action over a list of elements. Item is the worker that can help us accomplishing those tasks:

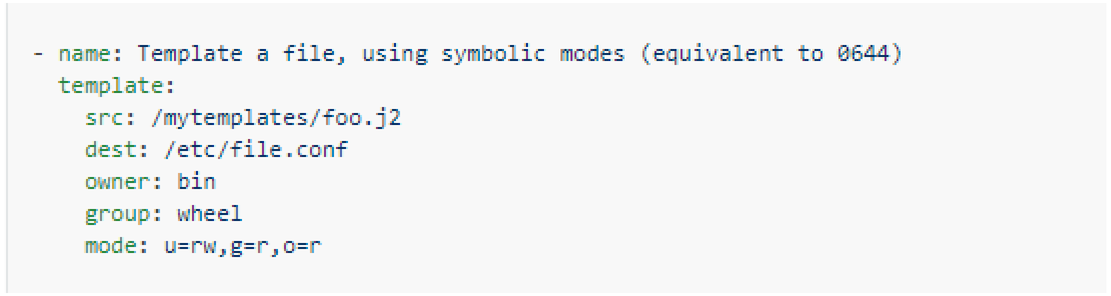
Item Example



```
- name: General | Instalación de paquetes requeridos.
  action: apt pkg={{ item }} state=installed
  with_items:
    - php5
    - apache2
    - mysql-server
    - mysql-client
    - php5-mysql
    - php-apc
    - php5-xmlrpc
    - php-soap
    - php5-gd
    - unzip
    - python-mysqldb
```

Variables

- Variables as any other language, stores information that we can use after in the execution of the playbook. We can declare them:
  - Inside the playbook
  - J2 files (templates)



```
- name: Template a file, using symbolic modes (equivalent to 0644)
  template:
    src: /mytemplates/foo.j2
    dest: /etc/file.conf
    owner: bin
    group: wheel
    mode: u=rw,g=r,o=r
```

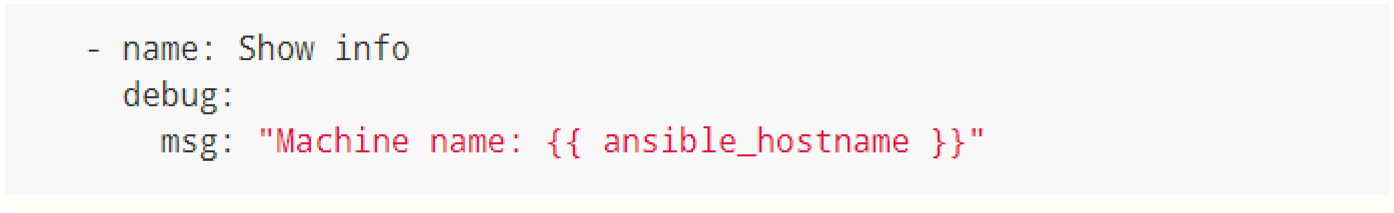
Loops: with_items

- Loops are useful with repetitive Jobs that are executed in the playbook like add users to some group…

Example

- Create a playbook that add user test1 and test2 to group Wheel.

Ansible facts



```
- name: Show info
  debug:
    msg: "Machine name: {{ ansible_hostname }}"
```

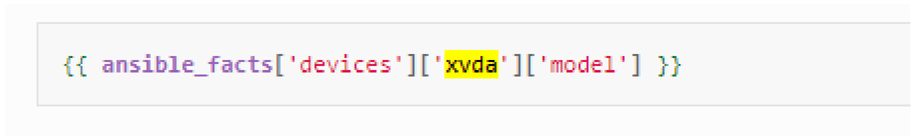
Facts are used to apply commands to specific hosts. Return a Python dictionary with the information reported by host.

Useful cases:

In a play where I need to get the following information:

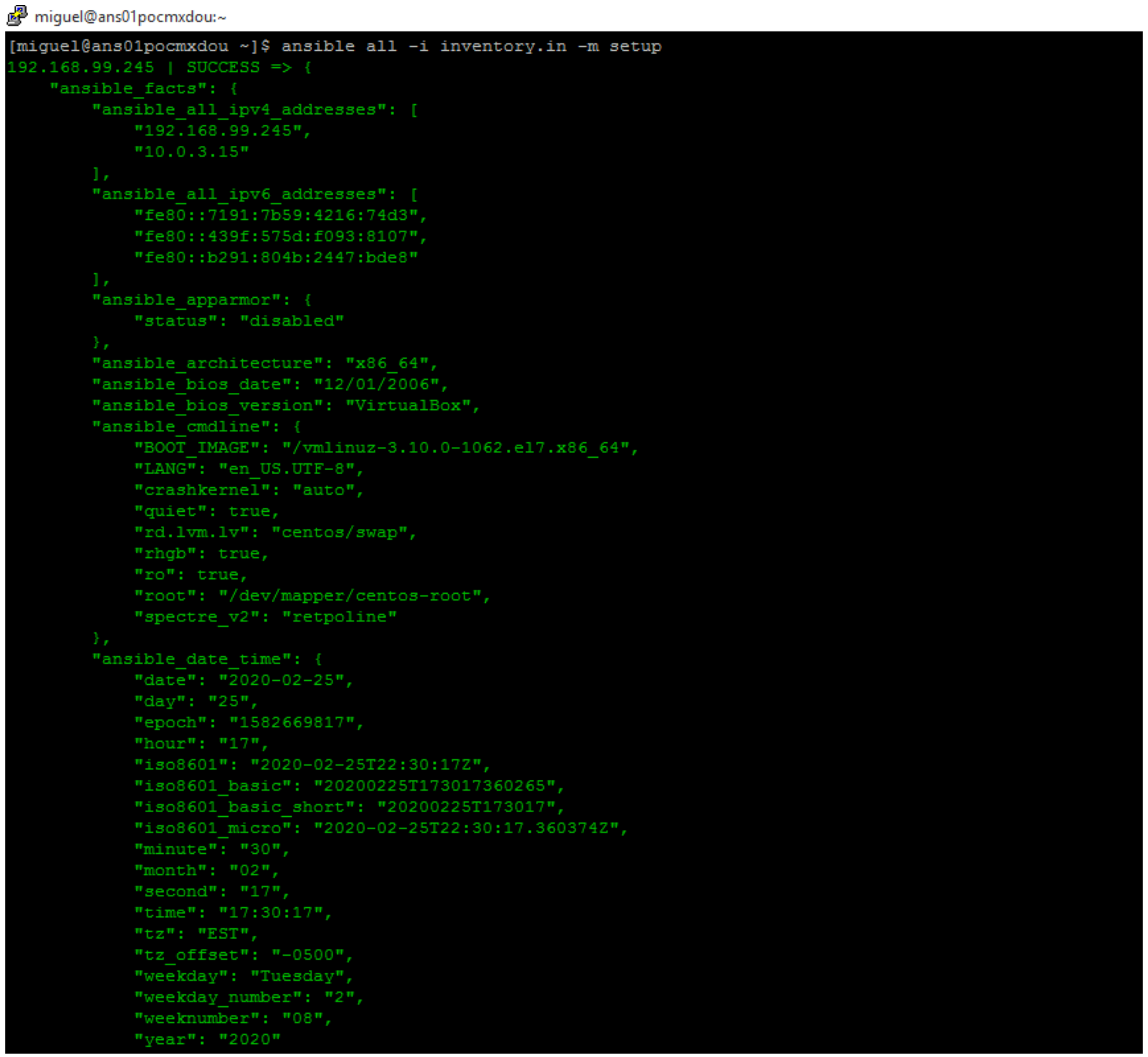
- Get the dmidecode information only from physical nodes.
- Install spacewalk client only on virtualmachines.

How to Access to sub facts:



```
{{ ansible_facts['devices']['xvda']['model'] }}
```

Ansible Facts example



```
[miguel@ans01pocmxdou ~]$ ansible all -i inventory.in -m setup
192.168.99.245 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "192.168.99.245",
            "10.0.3.15"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::7191:7b59:4216:74d3",
            "fe80::439f:575d:f093:8107",
            "fe80::b291:804b:2447:bde8"
        ],
        "ansible_apparmor": {
            "status": "disabled"
        },
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "12/01/2006",
        "ansible_bios_version": "VirtualBox",
        "ansible_cmdline": {
            "BOOT_IMAGE": "/vmlinuz-3.10.0-1062.el7.x86_64",
            "LANG": "en_US.UTF-8",
            "crashkernel": "auto",
            "quiet": true,
            "rd.lvm.lv": "centos/swap",
            "rhgb": true,
            "ro": true,
            "root": "/dev/mapper/centos-root",
            "spectre_v2": "retpoline"
        },
        "ansible_date_time": {
            "date": "2020-02-25",
            "day": "25",
            "epoch": "1582669817",
            "hour": "17",
            "iso8601": "2020-02-25T22:30:17Z",
            "iso8601_basic": "20200225T173017360265",
            "iso8601_basic_short": "20200225T173017",
            "iso8601_micro": "2020-02-25T22:30:17.360374Z",
            "minute": "30",
            "month": "02",
            "second": "17",
            "time": "17:30:17",
            "tz": "EST",
            "tz_offset": "-0500",
            "weekday": "Tuesday",
            "weekday_number": "2",
            "weeknumber": "08",
            "year": "2020"
```

Must read

- Since this presentation is only for the really basics of ansible, following topics must be followed in order to get more background:
  - Using variables: https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html
  - Using roles: https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html
  - Templates: https://docs.ansible.com/ansible/latest/modules/template_module.html
  - Modules documentation: https://docs.ansible.com/ansible/latest/modules/modules_by_category.html

One other

- Ansible has a lot of good practices, below document is a must read also.

Exercises 1

- 1. Create a adhoc command that display the first 10 users located in /etc/passwd. Use only adhoc command, find the way to pass parameters for the module used.
- This adhoc command must be applied for all of your vms listed in your inventory file.

Exercises 2

- You must provisioning a new server. Requiered packages to install in all nodes:
  - cups
  - vim
  - httpd
  - gcc
- Enable and start the service of httpd and cups, using the correct module for your distribution (service or systemd)
- Create playbook that install all of those packages using loops and the system configuration also must be done using loops.

Exercises 3

- Create a new playbook and take in consideration the following:
  - You were assigned to create 1 printer (point to /dev/null)
  - Set the status of that printer to disabled.
  - Send Jobs to that disabled printer. (Tip: Use foor loop bash)
    - Extra points if that is created with playbooks :P
  - PLAYBOOK start here:
  - Under the main play, create a new folder called: **DOU + local time of your server (with date command)** and take the backup of /var/spool*, /var/cache/*, /etc/cups/* . Each directory must be separated and must be compressed with gzip.
  - Get the output of lpstat –o and redirect to a new file called lpstat_o
  - Get the output of lpstat –t and redirect to a new file called lpstat_t
    - Send both files to the first folder.
  - Tar and compress the final folder which must have lpstat files and spool directories, place in tmp or any other directory.

Exercises 4

- Your manager wants to disable NetworkManager service from your nodes and use the public dns of google. Create a new playbook that disable that daemon from start and copy from your template folder the new resolv.conf – place into /etc/resolv.conf