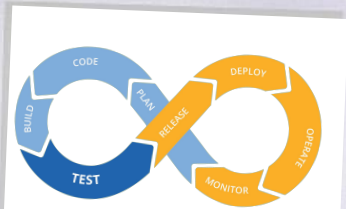
Ansible

Lev Denisov
Anna Turu Pi

Infrastructure as Code

- Historically, you would ensure that a system's state matched what you expected and then create a document certifying that this is the case.

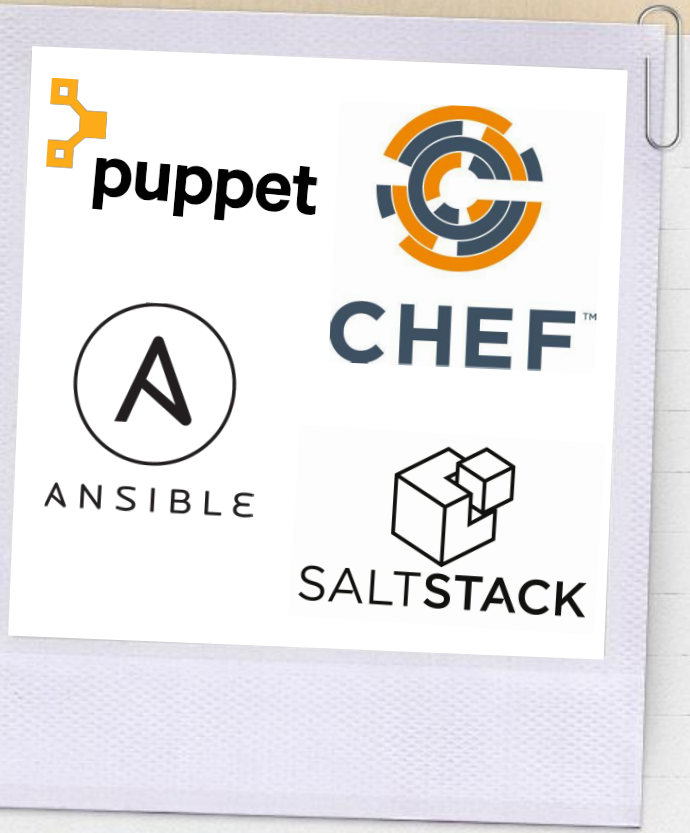
- Today, we write that document as a specification declaring the expected state of the system, then rely on tools such as Ansible to implement the transformations required.



Deployment Automation

Configuration Management Tools

- ✗ Puppet and Chef each use a central server to store the desired state of machines and any metadata associated with them.
- ✗ Ansible is agentless, it doesn't have any central server at all.
- ✗ Both Ansible and Salt use YAML syntax.
- ✗ Ansible uses SSH to connect from the host machine to other systems.



Installing Ansible

Linux

```
sudo apt-get install
python-software-properties # if required
sudo apt-add-repository ppa:ansible/ansible
sudo apt-get update
sudo apt-get install ansible
```

OS X

```
brew install ansible
```

Creating Your Environment

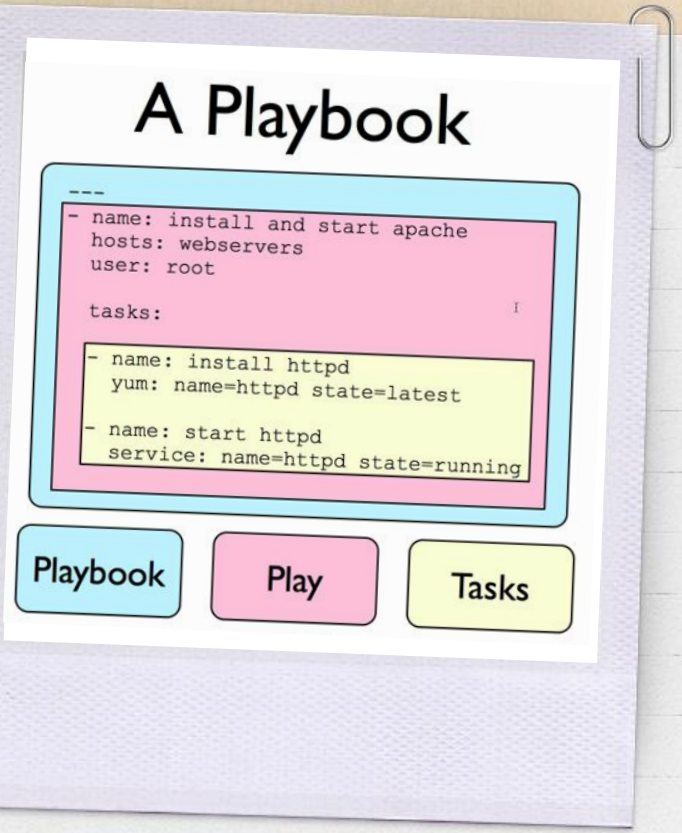
```
config.vm.provision "ansible_local" do |ansible|
    ansible.playbook = "provisioning/playbook.yml"
end
```

- ✗ Using **ansible** will run Ansible (which we installed earlier) on your local machine, while using **ansible_local** will log in to the virtual machine and run it there instead.

Simple playbook

provisioning/playbook.yml

```
---
- hosts: all
   tasks:
    - name: Make sure that we can
connect to the machine
       ping:
```



Simple playbook

Output:

```
TASK: [Make sure that we can connect to the machine]
**************************
ok: [default]
```

Minimal installation: pip and Django


```
---
- hosts: all
    become: true
    tasks:
    - name: Install pip
        apt: name=python3-pip state=present update_cache=yes
    - name: Install django
        pip:
            name: django
            executable: /usr/bin/pip3
```

Pip and Django installation playbook output

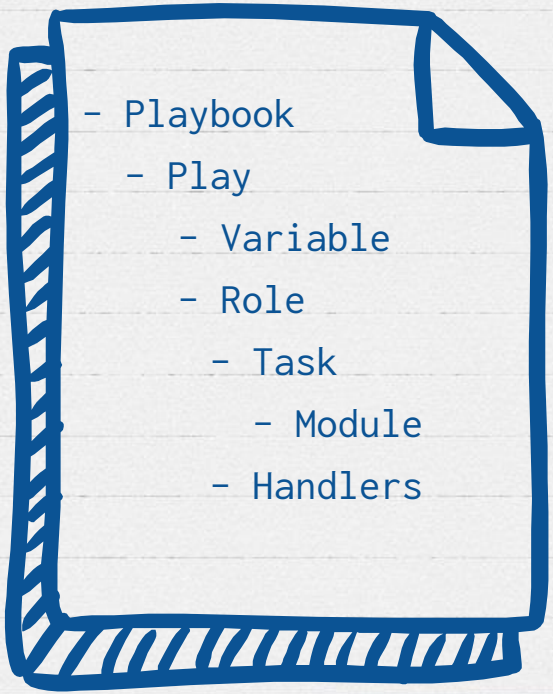
```
PLAY [all] *********************************************************

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

TASK [Install pip] *********************************************************
ok: [default]

TASK [Install django] *********************************************************
changed: [default]

PLAY RECAP *********************************************************************
default                    : ok=4    changed=1  unreachable=0   failed=0
```

Ansible terminology

- Controller Machine

- Inventory

```
host[1-3].example.com
host5.example.com:50822
192.168.9.29
```



Playbooks and Idempotency

- ✗ Idempotent means that you can do something multiple times and the outcome will be the same.
- ✗ Playbook is idempotent if you can run it multiple times and have the same machine state.
- ✗ Most of Ansible’s modules are idempotent.

Handlers

- ✗ Run tasks automatically
- ✗ Check if changes are required
- ✗ For example: restart nginx after the configuration file is updated

Inventory

```
ansible all -i /path/to/inventory.ini -m ping
```

inventory.ini:

```
host[1-3].example.com
host5.example.com:50822
192.168.9.29
```

Runs provisioning on every specified host

Variables

- ✗ Letters, numbers, underscores are valid names
- ✗ Always start with a letter
- ✗ {{ ansible_env }} pulls from environment
- ✗ Precedence rules exist

Roles

- ✗ “A playbook that is split up into multiple different files”
- ✗ You can’t run a role on its own; you need to include it inside a playbook along with the information about which hosts to run on
- ✗ Allow multinode orchestration
- ✗ Define behavior for server type
- ✗ Enable reuse and organization

Roles

Playbook:

```
---
- hosts: webservers
  roles:
     - common
     - webservers
```

```
# roles/example/tasks/main.yml
- name: added in 2.4, previously you used 'include'
  import_tasks: redhat.yml
  when: ansible_os_platform|lower == 'redhat'

# roles/example/tasks/redhat.yml
- yum:
    name: "httpd"
    state: present
```

Deployment

- ✗ Configure servers
- ✗ Set up [Python] environment
- ✗ Deploy code
- ✗ Migrate database schema & data
- ✗ Perform ad hoc tasks

Windows

Ansible is not officially supported in windows but there are ways to run it from Windows Subsystem for Linux (WSL). Vagrant allows to use ansible on the guest system to provision linux machines from windows.

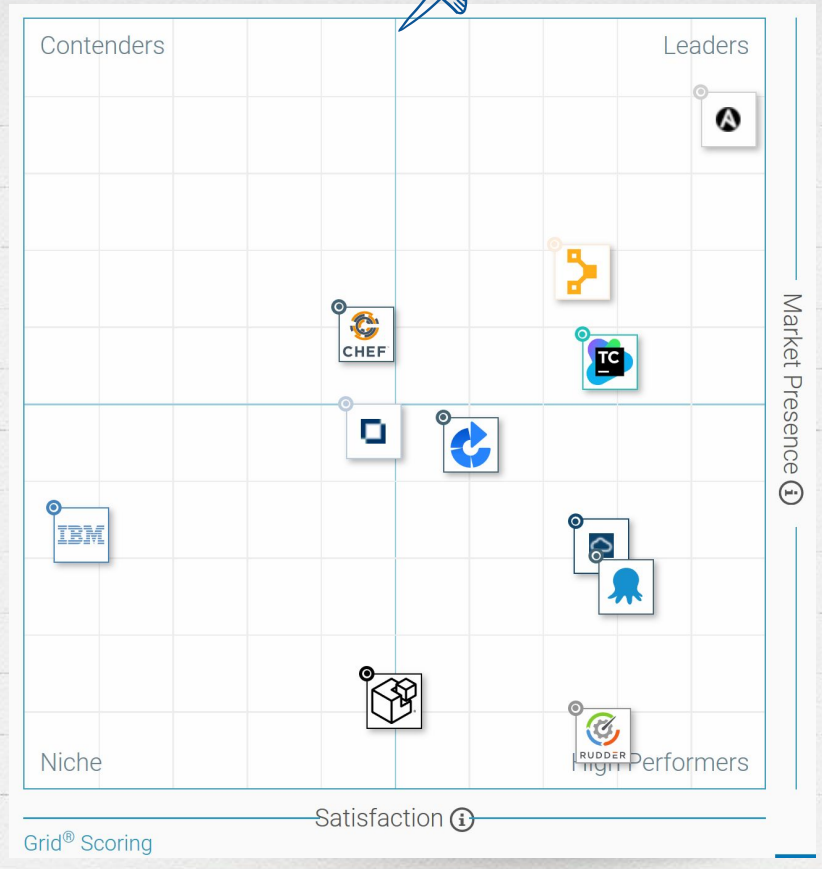
AWS with Ansible

Ansible can be used to create infrastructure on AWS

Use orchestration playbook for creating instances
Use provisioning playbook to configure the created instances

CM Tools Comparison

- ✗ **Market leader**
  - ✗ Easier to use
  - ✗ Fastest deployment
    - ✗ Agentless
    - ✗ Simplest syntax



References

- ✗ Ansible docs, http://www.ansibleworks.com/docs/
- ✗ Deploy Django with Ansible, https://github.com/mattmakai/sf-django
- ✗ Edureka, DevOps Tutorials, https://www.edureka.co/blog/what-is-ansible/
- ✗ Michael Heap, Ansible From Beginner to Pro, 2016