

ANSIBLE PARA DEV+OPS

DÍA 6

QUE VEREMOS HOY

- ¿Qué es Ansible Tower?
- Aprovisionamiento con Vagrant
- Interacción con servicios cloud
- Interacción con docker
- Labs:
 - Interacción con AWS
 - Generación de contenedores mediante ansible

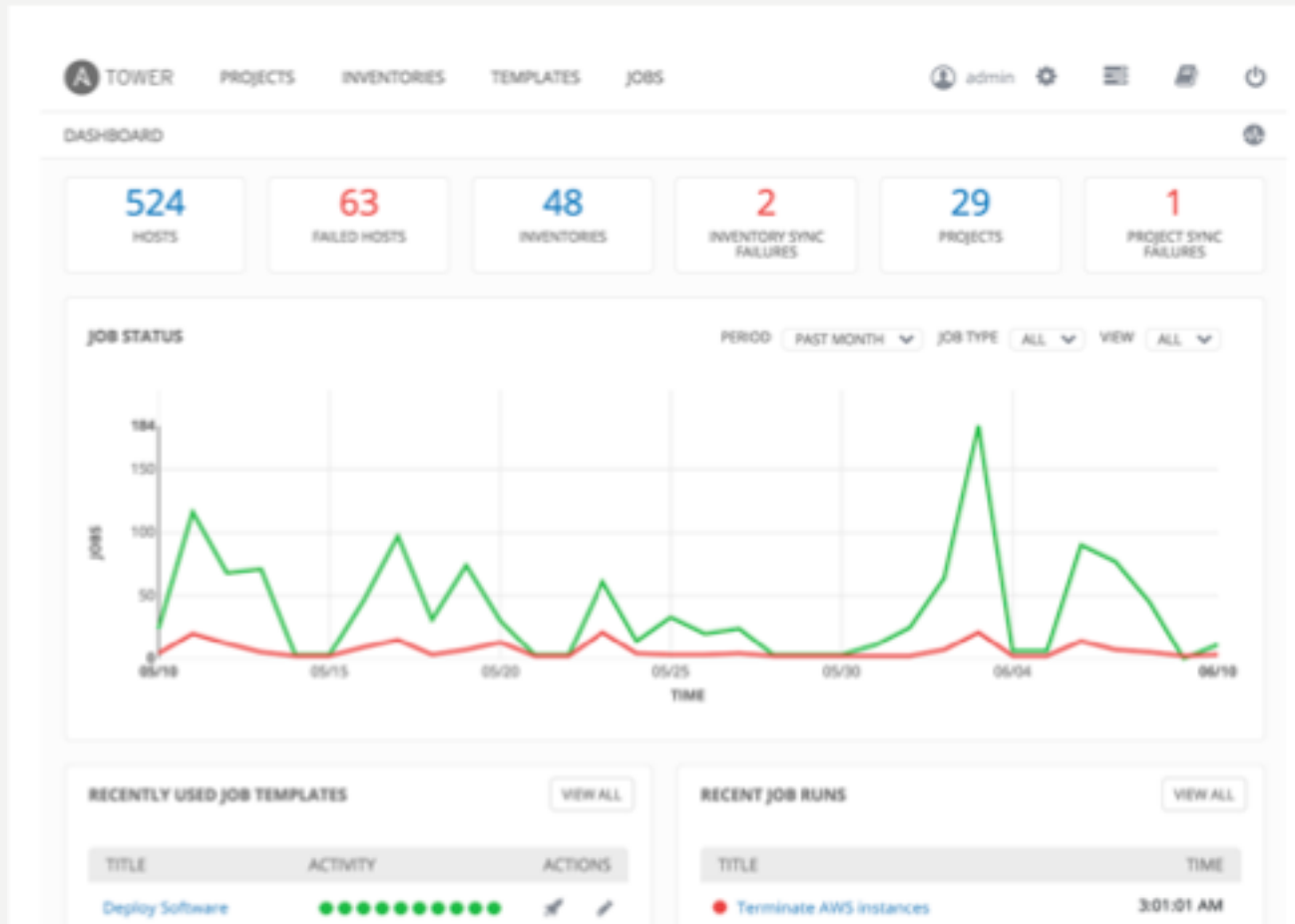
ANSIBLE TOWER



¿QUÉ ES ANSIBLE TOWER?

- Plataforma de CI/CD para ansible
 - Gestión de roles
 - Control de versiones
 - Reporting
-
- NEW!! Ansible Tower Opensource -> <https://github.com/ansible/awx>

TOWER DASHBOARD



JOB STATUS

DETAILS

STATUS

Successful

STARTED

1:00:07 AM

FINISHED

1:00:12 AM

TEMPLATE

Update License Server

JOB TYPE

Playbook Run

LAUNCHED BY

User

INVENTORY

License Server

PROJECT

License

REVISION

00000000

PLAYBOOK

store.yml

CREDENTIAL

License Server Deployment

LIMIT

Store

VERBOSITY

Update License Server

EXTRA VARIABLES

1

expire_time: 1453164676

2

vmware_host: cent7issue

3

4

LABELS

UPDATES

OK

REMOVE VMWARE HOST

OK 1

PLAYS 1

TASKS 8

HOSTS 15

ELAPSED 00:00:00

SEARCH

KEY

1

PLAY [Remove VMWare Host] *****

00:00:05

2

3

GATHERING FACTS *****

00:00:01

15

16

TASK: [ansiblelicense | install required packages via yum] *****

00:00:01

17

ok: [74.207.226.226]

18

ok: [74.207.226.226]

19

20

TASK: [ansiblelicense | update setuptools] *****

00:00:01

28

29

TASK: [ansiblelicense | update pip] *****

00:00:01

35

36

TASK: [ansiblelicense | create unprivileged user for ansiblelicense] *****

00:00:01

37

skipping: [74.207.226.226]

38

skipping: [74.207.226.226]

39

40

TASK: [ansiblelicense | configure ansiblelicense directory permissions] *****

00:00:01

41

changed: [74.207.226.226]

42

changed: [74.207.226.226]

43

44

TASK: [ansiblelicense | enable maintenance page] *****

00:00:01

65

66

TASK: [ansiblelicense | check ssh connection to github] *****

00:00:01

67

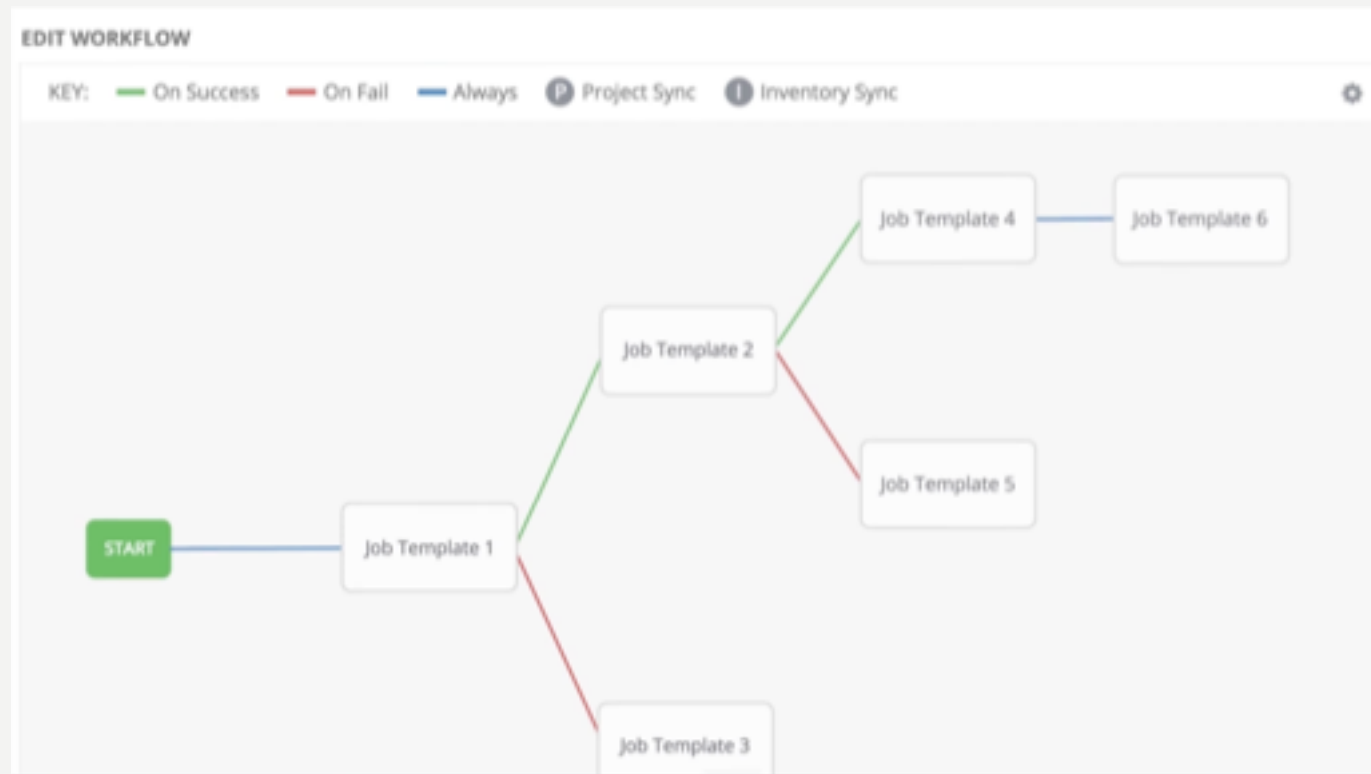
ok: [74.207.226.226]

68


69


PLAY RECAP *****


MULTI PLAYBOOK WORKFLOW




REMOTE COMMAND


 TOWER




EXECUTE COMMAND 


*MODULE 

yum


HOST PATTERN 

host1:host2


☒ ENABLE PRIVILEGE ESCALATION 


FORKS 


0

ARGUMENTS 

name=openssl state=latest

*MACHINE CREDENTIAL 

 Staging ssh key

*VERBOSITY 

0 (Normal)

VAGRANT

- Herramienta para el despliegue rápido de máquinas virtuales basado en plantillas
- Plantillas basadas en VM precocinadas para
 - VirtualBox
 - Vmware
 - Parallels
- <http://vagrant.up>

VAGRANT – 1 SOLA VM

- `vagrant init centos/7`
- `vagrant up`

VAGRANTFILE

- Permite personalizar una imagen de vagrant
 - Provisioner para shell scripts, ansible, chef...
 - Características de RAM y CPU
- Permite hacer despliegues de más de una VM
- Es un fichero ruby, que usa una DSL específica
- <https://www.vagrantup.com/docs/vagrantfile/>

VAGRANT - 3 VMS

```
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  # Use the same key for each machine
  config.ssh.insert_key = false

  config.vm.define "vagrant1" do |vagrant1|
    vagrant1.vm.box = "ubuntu/trusty64"
    vagrant1.vm.network "forwarded_port", guest: 80, host: 8080
    vagrant1.vm.network "forwarded_port", guest: 443, host: 8443
  end

  config.vm.define "vagrant2" do |vagrant2|
    vagrant2.vm.box = "ubuntu/trusty64"
    vagrant2.vm.network "forwarded_port", guest: 80, host: 8081
    vagrant2.vm.network "forwarded_port", guest: 443, host: 8444
  end

  config.vm.define "vagrant3" do |vagrant3|
    vagrant3.vm.box = "ubuntu/trusty64"
    vagrant3.vm.network "forwarded_port", guest: 80, host: 8082
    vagrant3.vm.network "forwarded_port", guest: 443, host: 8445
  end
end
```

ANSIBLE + CLOUD SERVICES

- Cosas que veremos
 - Ansible y AWS
 - Ansible y vSphere
- Conceptos a tener en cuenta
 - Ejecución en el control host (host: localhost)
 - Delegación de tareas al control host

ANSIBLE + AWS

- Ansible tiene una serie de módulos que permiten integrar AWS con Ansible para aprovisionamiento y configuración
- http://docs.ansible.com/ansible/latest/guide_aws.html
- Depende de boto (librería de python)
- Requieren clave de API
- No guardan idempotencia en un fichero, si no que comprueban a cada ciclo
 - Tiempo de ejecución alto
 - Algunos componentes (como los security groups) no tienen bien resuelta la idempotencia

ANSIBLE + AWS: CREAR INSTANCIAS EC2

```
- name: Provision a set of instances
  ec2:
    key_name: my_key
    group: test
    instance_type: t2.micro
    image: "{{ ami_id }}"
    wait: true
    exact_count: 5
    count_tag:
      Name: Demo
    instance_tags:
      Name: Demo
    register: ec2
```

ANSIBLE + AWS: ANSIBLE VS CLOUDFORMATION

- CF mantiene idempotencia de forma más controlable que Ansible
- CF tiene una sintaxis más confusa que Ansible
- Si ya se tiene desarrollado la capa de infraestructura con CF, Ansible tiene un módulo para ejecutar scripts en CF

ANSIBLE + VSPHERE

- Módulo vsphere_guest
 - http://docs.ansible.com/ansible/latest/vsphere_guest_module.html
- Muy complejo
- Debe ejecutarse en un nodo con visibilidad directa al cluster de vsphere
- Depende de pvmomi(módulo de python)

DONDE EJECUTAR AWS/VSPHERE

- Generalmente se ejecutarán en el nodo de control
- Los Playbooks entonces deberán apuntar a host: localhost para todas las tareas
- También existe la posibilidad de que las tareas individuales relacionadas con infraestructura estén delegadas mediante la opción “delegate_to:”