EBook Gratis

APRENDIZAJE ansible

Free unaffiliated eBook created from
Stack Overflow contributors.

#ansible

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: ansible

It is an unofficial and free ansible ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official ansible.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con ansible

Observaciones

Esta sección proporciona una descripción general de qué es ansible y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema grande dentro de ansible, y vincular a los temas relacionados. Dado que la Documentación para ansible es nueva, es posible que deba crear versiones iniciales de los temas relacionados.

Examples

Hola Mundo

Crea un directorio llamado `ansible-helloworld-playbook`

```
mkdir ansible-helloworld-playbook
```

Crea un archivo de `hosts` y agrega sistemas remotos como quieras gestionar. Como ansible confía en ssh para conectar las máquinas, debe asegurarse de que ya estén disponibles para usted en ssh desde su computadora.

```
192.168.1.1
192.168.1.2
```

Pruebe la conexión a sus sistemas remotos utilizando el módulo de ping ansible.

```
ansible all -m ping -k
```

En caso de éxito debería devolver algo así.

```
192.168.1.1| SUCCESS => {
    "changed": false,
    "ping": "pong"
}
192.168.1.2| SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

En caso de error debe volver.

```
192.168.1.1| UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh.",
    "unreachable": true
```

```
}
```

Probar el acceso de sudo con

```
ansible all -m ping -k -b
```

Probar conexión y configuración con ping

```
ansible -i hosts -m ping targethost
```

`-i hosts` define la ruta al archivo de inventario
`targethost` es el nombre del host en el archivo `hosts`

Inventario

El inventario es la forma más fácil de rastrear todos los sistemas en su infraestructura. Aquí hay un simple archivo de inventario estático que contiene un solo sistema y las credenciales de inicio de sesión de Ansible.

```
[targethost]
192.168.1.1 ansible_user=mrtuovinen ansible_ssh_pass=PassW0rd
```

Escriba estas líneas, por ejemplo, en el archivo `hosts` y pase el archivo a `ansible-playbook` comando `ansible` o `ansible-playbook` con la `--inventory-file -i` / `--inventory-file` .

Ver inventario estático e inventario dinámico para más detalles.

Aprovisionamiento de máquinas remotas con Ansible.

Podemos aprovisionar sistemas remotos con Ansible. Debe tener un par de claves SSH y debe llevar su clave pública SSH al archivo ~ / .ssh / authorized_keys de la máquina. El porpuse es que puede iniciar sesión sin ninguna autorización.

Requisitos previos:

- Ansible

Necesita un archivo de inventario (por ejemplo: development.ini) donde determine el host que desea usar:

```
[MACHINE_NAME]
MACHINE_NAME hostname=MACHINE_NAME ansible_ssh_host=IP_ADDRESS ansible_port=SSH_PORT
ansible_connection=ssh ansible_user=USER ansible_ssh_extra_args="-o StrictHostKeyChecking=no -
o UserKnownHostsFile=/dev/null"
```

- nombre de host - el nombre de host de la máquina remota
- ansible_ssh_host - la ip o dominio del host remoto
- ansible_port - el puerto del host remoto que suele ser 22

- ansible_connection - la conexión donde configuramos, queremos conectarnos con ssh
- ansible_user - el usuario ssh
- ansible_ssh_extra_args - argumentos extra lo que desea especificar para la conexión ssh

Argumentos adicionales requeridos para ssh:

- StrictHostKeyChecking: puede pedir a una clave que verifique qué espera de un sí o un no. El Ansible no puede responder a esta pregunta, luego arroja un error, el host no está disponible.
- UserKnownHostsFile: necesario para la opción StrictHostKeyChecking.

Si tiene este archivo de inventario, puede escribir una prueba playbook.yml:

```
---
- hosts: MACHINE_NAME
  tasks:
    - name: Say hello
      debug:
        msg: 'Hello, World'
```

A continuación, puede iniciar la disposición:

```
ansible-playbook -i development.ini playbook.yml
```

ansible.cfg

Este es el ansible.cfg predeterminado de Ansible github .

```
# config file for ansible -- http://ansible.com/
# ==============================================

# nearly all parameters can be overridden in ansible-playbook
# or with command line flags. ansible will read ANSIBLE_CONFIG,
# ansible.cfg in the current working directory, .ansible.cfg in
# the home directory or /etc/ansible/ansible.cfg, whichever it
# finds first

[defaults]

# some basic default values...

#inventory      = /etc/ansible/hosts
#library        = /usr/share/my_modules/
#remote_tmp     = $HOME/.ansible/tmp
#local_tmp      = $HOME/.ansible/tmp
#forks          = 5
#poll_interval  = 15
#sudo_user      = root
#ask_sudo_pass = True
#ask_pass      = True
#transport      = smart
#remote_port    = 22
#module_lang    = C
#module_set_locale = False
```

```
# plays will gather facts by default, which contain information about
# the remote system.
#
# smart - gather by default, but don't regather if already gathered
# implicit - gather by default, turn off with gather_facts: False
# explicit - do not gather by default, must say gather_facts: True
#gathering = implicit

# by default retrieve all facts subsets
# all - gather all subsets
# network - gather min and network facts
# hardware - gather hardware facts (longest facts to retrieve)
# virtual - gather min and virtual facts
# facter - import facts from facter
# ohai - import facts from ohai
# You can combine them using comma (ex: network,virtual)
# You can negate them using ! (ex: !hardware,!facter,!ohai)
# A minimal set of facts is always gathered.
#gather_subset = all

# some hardware related facts are collected
# with a maximum timeout of 10 seconds. This
# option lets you increase or decrease that
# timeout to something more suitable for the
# environment.
# gather_timeout = 10

# additional paths to search for roles in, colon separated
#roles_path    = /etc/ansible/roles

# uncomment this to disable SSH key host checking
#host_key_checking = False

# change the default callback
#stdout_callback = skippy
# enable additional callbacks
#callback_whitelist = timer, mail

# Determine whether includes in tasks and handlers are "static" by
# default. As of 2.0, includes are dynamic by default. Setting these
# values to True will make includes behave more like they did in the
# 1.x versions.
#task_includes_static = True
#handler_includes_static = True

# change this for alternative sudo implementations
#sudo_exe = sudo

# What flags to pass to sudo
# WARNING: leaving out the defaults might create unexpected behaviours
#sudo_flags = -H -S -n

# SSH timeout
#timeout = 10

# default user to use for playbooks if user is not specified
# (/usr/bin/ansible will use current user as default)
#remote_user = root

# logging is off by default unless this path is defined
```

```
# if so defined, consider logrotate
#log_path = /var/log/ansible.log

# default module name for /usr/bin/ansible
#module_name = command

# use this shell for commands executed under sudo
# you may need to change this to bin/bash in rare instances
# if sudo is constrained
#executable = /bin/sh

# if inventory variables overlap, does the higher precedence one win
# or are hash values merged together?  The default is 'replace' but
# this can also be set to 'merge'.
#hash_behaviour = replace

# by default, variables from roles will be visible in the global variable
# scope. To prevent this, the following option can be enabled, and only
# tasks and handlers within the role will see the variables there
#private_role_vars = yes

# list any Jinja2 extensions to enable here:
#jinja2_extensions = jinja2.ext.do,jinja2.ext.i18n

# if set, always use this private key file for authentication, same as
# if passing --private-key to ansible or ansible-playbook
#private_key_file = /path/to/file

# If set, configures the path to the Vault password file as an alternative to
# specifying --vault-password-file on the command line.
#vault_password_file = /path/to/vault_password_file

# format of string {{ ansible_managed }} available within Jinja2
# templates indicates to users editing templates files will be replaced.
# replacing {file}, {host} and {uid} and strftime codes with proper values.
#ansible_managed = Ansible managed: {file} modified on %Y-%m-%d %H:%M:%S by {uid} on {host}
# This short version is better used in templates as it won't flag the file as changed every
run.
#ansible_managed = Ansible managed: {file} on {host}

# by default, ansible-playbook will display "Skipping [host]" if it determines a task
# should not be run on a host.  Set this to "False" if you don't want to see these "Skipping"
# messages. NOTE: the task header will still be shown regardless of whether or not the
# task is skipped.
#display_skipped_hosts = True

# by default, if a task in a playbook does not include a name: field then
# ansible-playbook will construct a header that includes the task's action but
# not the task's args.  This is a security feature because ansible cannot know
# if the *module* considers an argument to be no_log at the time that the
# header is printed.  If your environment doesn't have a problem securing
# stdout from ansible-playbook (or you have manually specified no_log in your
# playbook on all of the tasks where you have secret information) then you can
# safely set this to True to get more informative messages.
#display_args_to_stdout = False

# by default (as of 1.3), Ansible will raise errors when attempting to dereference
# Jinja2 variables that are not set in templates or action lines. Uncomment this line
# to revert the behavior to pre-1.3.
#error_on_undefined_vars = False
```

```
# by default (as of 1.6), Ansible may display warnings based on the configuration of the
# system running ansible itself. This may include warnings about 3rd party packages or
# other conditions that should be resolved if possible.
# to disable these warnings, set the following value to False:
#system_warnings = True

# by default (as of 1.4), Ansible may display deprecation warnings for language
# features that should no longer be used and will be removed in future versions.
# to disable these warnings, set the following value to False:
#deprecation_warnings = True

# (as of 1.8), Ansible can optionally warn when usage of the shell and
# command module appear to be simplified by using a default Ansible module
# instead.  These warnings can be silenced by adjusting the following
# setting or adding warn=yes or warn=no to the end of the command line
# parameter string.  This will for example suggest using the git module
# instead of shelling out to the git command.
# command_warnings = False


# set plugin path directories here, separate with colons
#action_plugins     = /usr/share/ansible/plugins/action
#cache_plugins      = /usr/share/ansible/plugins/cache
#callback_plugins   = /usr/share/ansible/plugins/callback
#connection_plugins = /usr/share/ansible/plugins/connection
#lookup_plugins     = /usr/share/ansible/plugins/lookup
#inventory_plugins  = /usr/share/ansible/plugins/inventory
#vars_plugins       = /usr/share/ansible/plugins/vars
#filter_plugins     = /usr/share/ansible/plugins/filter
#test_plugins       = /usr/share/ansible/plugins/test
#strategy_plugins   = /usr/share/ansible/plugins/strategy

# by default callbacks are not loaded for /bin/ansible, enable this if you
# want, for example, a notification or logging callback to also apply to
# /bin/ansible runs
#bin_ansible_callbacks = False


# don't like cows?  that's unfortunate.
# set to 1 if you don't want cowsay support or export ANSIBLE_NOCOWS=1
#nocows = 1

# set which cowsay stencil you'd like to use by default. When set to 'random',
# a random stencil will be selected for each task. The selection will be filtered
# against the `cow_whitelist` option below.
#cow_selection = default
#cow_selection = random

# when using the 'random' option for cowsay, stencils will be restricted to this list.
# it should be formatted as a comma-separated list with no spaces between names.
# NOTE: line continuations here are for formatting purposes only, as the INI parser
#       in python does not support them.
#cow_whitelist=bud-frogs,bunny,cheese,daemon,default,dragon,elephant-in-snake,elephant,eyes,\
#              hellokitty,kitty,luke-
koala,meow,milk,moofasa,moose,ren,sheep,small,stegosaurus,\
#              stimpy,supermilker,three-eyes,turkey,turtle,tux,udder,vader-koala,vader,www

# don't like colors either?
# set to 1 if you don't want colors, or export ANSIBLE_NOCOLOR=1
#nocolor = 1
```

```
# if set to a persistent type (not 'memory', for example 'redis') fact values
# from previous runs in Ansible will be stored.  This may be useful when
# wanting to use, for example, IP information from one group of servers
# without having to talk to them in the same playbook run to get their
# current IP information.
#fact_caching = memory


# retry files
# When a playbook fails by default a .retry file will be created in ~/
# You can disable this feature by setting retry_files_enabled to False
# and you can change the location of the files by setting retry_files_save_path

#retry_files_enabled = False
#retry_files_save_path = ~/.ansible-retry

# squash actions
# Ansible can optimise actions that call modules with list parameters
# when looping. Instead of calling the module once per with_ item, the
# module is called once with all items at once. Currently this only works
# under limited circumstances, and only with parameters named 'name'.
#squash_actions = apk,apt,dnf,package,pacman,pkgng,yum,zypper

# prevents logging of task data, off by default
#no_log = False

# prevents logging of tasks, but only on the targets, data is still logged on the
master/controller
#no_target_syslog = False

# controls whether Ansible will raise an error or warning if a task has no
# choice but to create world readable temporary files to execute a module on
# the remote machine.  This option is False by default for security.  Users may
# turn this on to have behaviour more like Ansible prior to 2.1.x.  See
# https://docs.ansible.com/ansible/become.html#becoming-an-unprivileged-user
# for more secure ways to fix this than enabling this option.
#allow_world_readable_tmpfiles = False

# controls the compression level of variables sent to
# worker processes. At the default of 0, no compression
# is used. This value must be an integer from 0 to 9.
#var_compression_level = 9

# controls what compression method is used for new-style ansible modules when
# they are sent to the remote system.  The compression types depend on having
# support compiled into both the controller's python and the client's python.
# The names should match with the python Zipfile compression types:
# * ZIP_STORED (no compression. available everywhere)
# * ZIP_DEFLATED (uses zlib, the default)
# These values may be set per host via the ansible_module_compression inventory
# variable
#module_compression = 'ZIP_DEFLATED'

# This controls the cutoff point (in bytes) on --diff for files
# set to 0 for unlimited (RAM may suffer!).
#max_diff_size = 1048576

[privilege_escalation]
#become=True
#become_method=sudo
#become_user=root
```

```
#become_ask_pass=False

[paramiko_connection]

# uncomment this line to cause the paramiko connection plugin to not record new host
# keys encountered.  Increases performance on new host additions.  Setting works independently
of the
# host key checking setting above.
#record_host_keys=False

# by default, Ansible requests a pseudo-terminal for commands executed under sudo. Uncomment
this
# line to disable this behaviour.
#pty=False

[ssh_connection]

# ssh arguments to use
# Leaving off ControlPersist will result in poor performance, so use
# paramiko on older platforms rather than removing it, -C controls compression use
#ssh_args = -C -o ControlMaster=auto -o ControlPersist=60s

# The path to use for the ControlPath sockets. This defaults to
# "%(directory)s/ansible-ssh-%%h-%%p-%%r", however on some systems with
# very long hostnames or very long path names (caused by long user names or
# deeply nested home directories) this can exceed the character limit on
# file socket names (108 characters for most platforms). In that case, you
# may wish to shorten the string below.
#
# Example:
# control_path = %(directory)s/%%h-%%r
#control_path = %(directory)s/ansible-ssh-%%h-%%p-%%r

# Enabling pipelining reduces the number of SSH operations required to
# execute a module on the remote server. This can result in a significant
# performance improvement when enabled, however when using "sudo:" you must
# first disable 'requiretty' in /etc/sudoers
#
# By default, this option is disabled to preserve compatibility with
# sudoers configurations that have requiretty (the default on many distros).
#
#pipelining = False

# if True, make ansible use scp if the connection type is ssh
# (default is sftp)
#scp_if_ssh = True

# if False, sftp will not use batch mode to transfer files. This may cause some
# types of file transfer failures impossible to catch however, and should
# only be disabled if your sftp version has problems with batch mode
#sftp_batch_mode = False

[accelerate]
#accelerate_port = 5099
#accelerate_timeout = 30
#accelerate_connect_timeout = 5.0

# The daemon timeout is measured in minutes. This time is measured
# from the last activity to the accelerate daemon.
#accelerate_daemon_timeout = 30
```

```
# If set to yes, accelerate_multi_key will allow multiple
# private keys to be uploaded to it, though each user must
# have access to the system via SSH to add a new key. The default
# is "no".
#accelerate_multi_key = yes

[selinux]
# file systems that require special treatment when dealing with security context
# the default behaviour that copies the existing context or uses the user default
# needs to be changed to use the file system dependent context.
#special_context_filesystems=nfs,vboxsf,fuse,ramfs

# Set this to yes to allow libvirt_lxc connections to work without SELinux.
#libvirt_lxc_noseclabel = yes

[colors]
#highlight = white
#verbose = blue
#warn = bright purple
#error = red
#debug = dark gray
#deprecate = purple
#skip = cyan
#unreachable = red
#ok = green
#changed = yellow
#diff_add = green
#diff_remove = red
#diff_lines = cyan
```

Coloque esta configuración en la raíz de los directorios de su rol para cambiar el comportamiento de Ansible cuando use ese rol. Por ejemplo, puedes configurarlo para que detenga la creación de `playbook.retry` en las ejecuciones fallidas de playbook o para apuntar a vars secretos que no deseas en tu repositorio de git.

Lea Empezando con ansible en línea: https://riptutorial.com/es/ansible/topic/826/empezando-con-ansible

Capítulo 2: Ansible Group Vars

Examples

Ejemplo de group_vars / development, y por qué

Estructura del proyecto

```
project/
  group_vars/
     development
  inventory.development
  playbook.yaml
```

Estas variables se aplicarán a los hosts en el grupo de desarrollo debido al nombre de archivo.

```
---
## Application
app_name: app
app_url: app.io
web_url: cdn.io
app_friendly: New App
env_type: production
app_debug: false

## SSL
ssl: true
ev_ssl: false

## Database
database_host: 127.0.0.1
database_name: app
database_user: sql

## Elasticsearch
elasticsearch_host: 127.0.0.1
```

Lea Ansible Group Vars en línea: https://riptutorial.com/es/ansible/topic/6226/ansible-group-vars

Capítulo 3: Ansible instalar mysql

Introducción

Cómo usar ansible para instalar el archivo binario mysql

Examples

Cómo usar ansible para instalar el archivo binario mysql

- hosts: tareas mysql:
    - nombre: Agregar usuario mysql usuario: nombre: mysql shell: / sbin / nologin
    - nombre: instale la última versión de libselinux-python yum: nombre: libselinux-python estado: última
    - nombre: instalar perl yum: nombre: estado de perl: más reciente
    - nombre: eliminar el paquete mysql-libs yum: nombre: mysql-libs estado: ausente

```
- name: download and unarchive tar
  unarchive:
     src=/tmp/mysql-5.6.35-linux-glibc2.5-x86_64.tar.gz
     dest=/tmp
     copy=yes


- name: Move mysql paceage to specified directory
  command: creates="/usr/local/mysql" mv /tmp/mysql-5.6.35-linux-glibc2.5-x86_64
/usr/local/mysql

- name: chown mysql mysql /usr/local/mysql
  file: path=/usr/local/mysql owner=mysql group=mysql recurse=yes


- name: Add lib to ld.so.conf
  lineinfile: dest=/etc/ld.so.conf line="/usr/local/mysql/lib/"

- name: ldconfig
  command: /sbin/ldconfig

- name: Mkdir mysql_data_dir
  file: path=/data/mysql/3306/{{ item }} state=directory owner=mysql group=mysql
  with_items:
    - data
    - logs
    - tmp

- name: Copy mysql my.cnf
  copy: src=/etc/my.cnf dest=/etc/my.cnf
```

```
- name: Copy mysql my.cnf
  copy: src=/etc/my.cnf dest=/usr/local/mysql/my.cnf


- name: Init mysql db
  command: /usr/local/mysql/scripts/mysql_install_db  \
    --user=mysql  \
    --basedir=/usr/local/mysql \
    --datadir=/data/mysql/3306/data

- name: Add mysql bin to profile
  lineinfile: dest=/etc/profile line="export PATH=$PATH:/usr/local/mysql/bin/"

- name: Source profile
  shell: executable=/bin/bash source /etc/profile

- name: Copy mysqld to init when system start
  command: cp -f /usr/local/mysql/support-files/mysql.server /etc/init.d/mysqld

- name: Add mysqld to system start
  command: /sbin/chkconfig --add mysqld

- name: Add mysql to system start when init 345
  command: /sbin/chkconfig --level 345 mysqld on

- name: Retart mysql
  service: name=mysqld state=restarted
```

Lea Ansible instalar mysql en línea: https://riptutorial.com/es/ansible/topic/10920/ansible-instalar-mysql

Capítulo 4: Ansible: bucle

Examples

with_items - lista simple

Se puede `with_items` bucle `with_items` en ansible para hacer un bucle fácilmente sobre los valores.

```
- name: Add lines to this file
  lineinfile: dest=/etc/file line={{ item }} state=present
  with_items:
    - Line 1
    - Line 2
    - Line 3
```

with_items - lista predefinida

También puede recorrer una lista de variables.

Desde vars:

```
favorite_snacks:
  - hotdog
  - ice cream
  - chips
```

y luego el bucle:

```
- name: create directories for storing my snacks
  file: path=/etc/snacks/{{ item }} state=directory
  with_items: '{{ favorite_snacks }}'
```

Si está utilizando Ansible 2.0+, debe usar comillas alrededor de la llamada a la variable.

with_items - diccionario predefinido

Es posible crear bucles más complejos con diccionarios.

Desde vars:

```
packages:
  - present: tree
  - present: nmap
  - absent: apache2
```

entonces el bucle:

```
- name: manage packages
```

```
  package: name={{ item.value }} state={{ item.key }}
  with_items: '{{ packages }}'
```

O, si no te gusta usar el valor clave:

vars

```
packages:
  - name: tree
    state: present
  - name: nmap
    state: present
  - name: apache2
    state: absent
```

entonces el bucle:

```
- name: manage packages
  package: name={{ item.name }} state={{ item.state }}
  with_items: '{{ packages }}'
```

with_items - diccionario

Puedes usar un diccionario para un bucle un poco más complejo.

```
- name: manage packages
  package: name={{ item.name }} state={{ item.state }}
  with_items:
    - { name: tree, state: present }
    - { name: nmap, state: present }
    - { name: apache2, state: absent }
```

Bucles anidados

Puedes crear bucles anidados usando `with_nested` .

de vars:

```
keys:
  - key1
  - key2
  - key3
  - key4
```

entonces el bucle:

```
- name: Distribute SSH keys among multiple users
  lineinfile: dest=/home/{{ item[0] }}/.ssh/authorized_keys line={{ item[1] }} state=present
  with_nested:
    - [ 'calvin', 'josh', 'alice' ]
    - '{{ keys }}'
```

Esta tarea recorrerá cada usuario y llenará su archivo `authorized_keys` con las 4 teclas definidas en la lista.

Lea Ansible: bucle en línea: https://riptutorial.com/es/ansible/topic/6414/ansible--bucle

Capítulo 5: Ansible: Bucles y Condicionales

Observaciones

Los documentos oficiales explican los condicionales del playbook.

- http://docs.ansible.com/ansible/playbooks_conditionals.html

Ansible (github)

- https://github.com/marxwang/ansible-learn-resources

Examples

¿Qué tipo de condicionales usar?

Use Conditionals via (la sintaxis está entre `[brackets]`):

- cuando [**cuando:**]

```
Task:
   - name: run if operating system is debian
     command: echo "I am a Debian Computer"
     when: ansible_os_family == "Debian"
```

- bucles [**with_items:**]
- bucles [**with_dicts:**]
- Datos personalizados [**cuando:** my_custom_facts == '1234']
- Importaciones condicionales
- Seleccionar archivos y plantillas basados en variables

[Cuando] Condición: `ansible_os_family` Listas

Uso común

- cuando: ansible_os_family == "CentOS"
- cuando: ansible_os_family == "Redhat"
- cuando: ansible_os_family == "Darwin"
- cuando: ansible_os_family == "Debian"
- cuando: ansible_os_family == "Windows"

Todas las listas

basado en discutir aquí http://comments.gmane.org/gmane.comp.sysutils.ansible/4685

```
OS_FAMILY = dict(
            RedHat = 'RedHat',
            Fedora = 'RedHat',
            CentOS = 'RedHat',
            Scientific = 'RedHat',
            SLC = 'RedHat',
            Ascendos = 'RedHat',
            CloudLinux = 'RedHat',
            PSBM = 'RedHat',
            OracleLinux = 'RedHat',
            OVS = 'RedHat',
            OEL = 'RedHat',
            Amazon = 'RedHat',
            XenServer = 'RedHat',
            Ubuntu = 'Debian',
            Debian = 'Debian',
            SLES = 'Suse',
            SLED = 'Suse',
            OpenSuSE = 'Suse',
            SuSE = 'Suse',
            Gentoo = 'Gentoo',
            Archlinux = 'Archlinux',
            Mandriva = 'Mandrake',
            Mandrake = 'Mandrake',
            Solaris = 'Solaris',
            Nexenta = 'Solaris',
            OmniOS = 'Solaris',
            OpenIndiana = 'Solaris',
            SmartOS = 'Solaris',
            AIX = 'AIX',
            Alpine = 'Alpine',
            MacOSX = 'Darwin',
            FreeBSD = 'FreeBSD',
            HPUX = 'HP-UX'
        )
```

Cuando la condicion

Uso básico

Utilice la condición when para controlar si una tarea o rol se ejecuta o se omite. Esto normalmente se usa para cambiar el comportamiento de juego basado en hechos del sistema de destino. Considera este libro de jugadas:

```
- hosts: all
  tasks:
    - include: Ubuntu.yml
      when: ansible_os_family == "Ubuntu"
```

```
    - include: RHEL.yml
      when: ansible_os_family == "RedHat"
```

Donde `Ubuntu.yml` y `RHEL.yml` incluyen alguna lógica de distribución específica.

Otro uso común es limitar los resultados a aquellos en ciertos grupos de inventario de Ansible. Considere este archivo de inventario:

```
[dbs]
mydb01

[webservers]
myweb01
```

Y este libro de jugadas:

```
- hosts: all
  tasks:
    - name: Restart Apache on webservers
      become: yes
      service:
        name: apache2
        state: restarted
      when: webservers in group_names
```

Esto está usando la variable mágica `group_names` .

Sintaxis condicional y lógica

Condicion individual

Sintaxis

```
when: (condition)
```

Ejemplo

- `when: ansible_os_family == "Debian"`
- `when: ansible_pkg_mgr == "apt"`
- `when: myvariablename is defined`

Filtro booleano

Ejemplo

```
when: result|failed
```

Múltiples condiciones

Sintaxis

```
When: condition1 and/or condition2
```

Ejemplo (simple)

```
when: ansible_os_family == "Debian" and ansible_pkg_mgr == "apt"
```

Ejemplo (complejo)

Use paréntesis para mayor claridad o para controlar la precedencia. "Y" tiene una prioridad más alta que "O".

Las cláusulas pueden abarcar líneas:

```
when:
  ansible_distribution in ['RedHat', 'CentOS', 'ScientificLinux'] and
  (ansible_distribution_version|version_compare('7', '<') or
  ansible_distribution_version|version_compare('8', '>='))
  or
  ansible_distribution == 'Fedora'
  or
  ansible_distribution == 'Ubuntu' and
  ansible_distribution_version|version_compare('15.04', '>=')
```

Tenga en cuenta el uso de paréntesis para agrupar "o" en la primera comprobación de distribución.

Obtenga `ansible_os_family` y` ansible_pkg_mgr` con la configuración

Podemos obtener datos (`ansible_os_family` , `ansible_pkg_mgr`) con el comando Ad-Hoc del módulo de configuración y el filtro.

- ansible_os_family:

```
$ ansible all -m setup -a 'filter=ansible_os_family'
ra.local | SUCCESS => {
    "ansible_facts": {
        "ansible_os_family": "Debian"
    },
    "changed": false
}
```

- ansible_pkg_mgr:

```
$ ansible all -m setup -a 'filter=ansible_pkg_mgr'
debian.local | SUCCESS => {
    "ansible_facts": {
        "ansible_pkg_mgr": "apt"
    },
    "changed": false
}
```

Ejemplo (s) simple de "cuándo"

Dado:

```
---
variable_name: True
```

Entonces, estas tareas siempre se ejecutan.

```
- name: This is a conditional task
  module: src=/example/ dest=/example
  when: variable_name

- name: This is a conditional task
  module: src=/example/ dest=/example
  when: True
```

Esta tarea nunca se ejecutará.

```
- name: This is a conditional task
  module: src=/example/ dest=/example
  when: False
```

Usando hasta para un reintento.

Este es un ejemplo del uso de hasta / reintentos / retraso para implementar una comprobación activa de una aplicación web que se está iniciando. Se supone que habrá algún período de tiempo (hasta 3 minutos) en el que la aplicación web rechace las conexiones de socket. Después de eso, comprueba la página / viva de la palabra "Aceptar". También delega la recuperación de la URL al localhost que se ejecuta ansible. Esto tiene sentido como la tarea final en un libro de jugadas de implementación.

```
---
- hosts: my-hosts
  tasks:
  - action: uri url=http://{{ ansible_all_ipv4_addresses }}:8080/alive return_content=yes
    delegate_to: localhost
    register: result
    until: "'failed' not in result and result.content.find('OK') != -1"
    retries: 18
    delay: 10
```

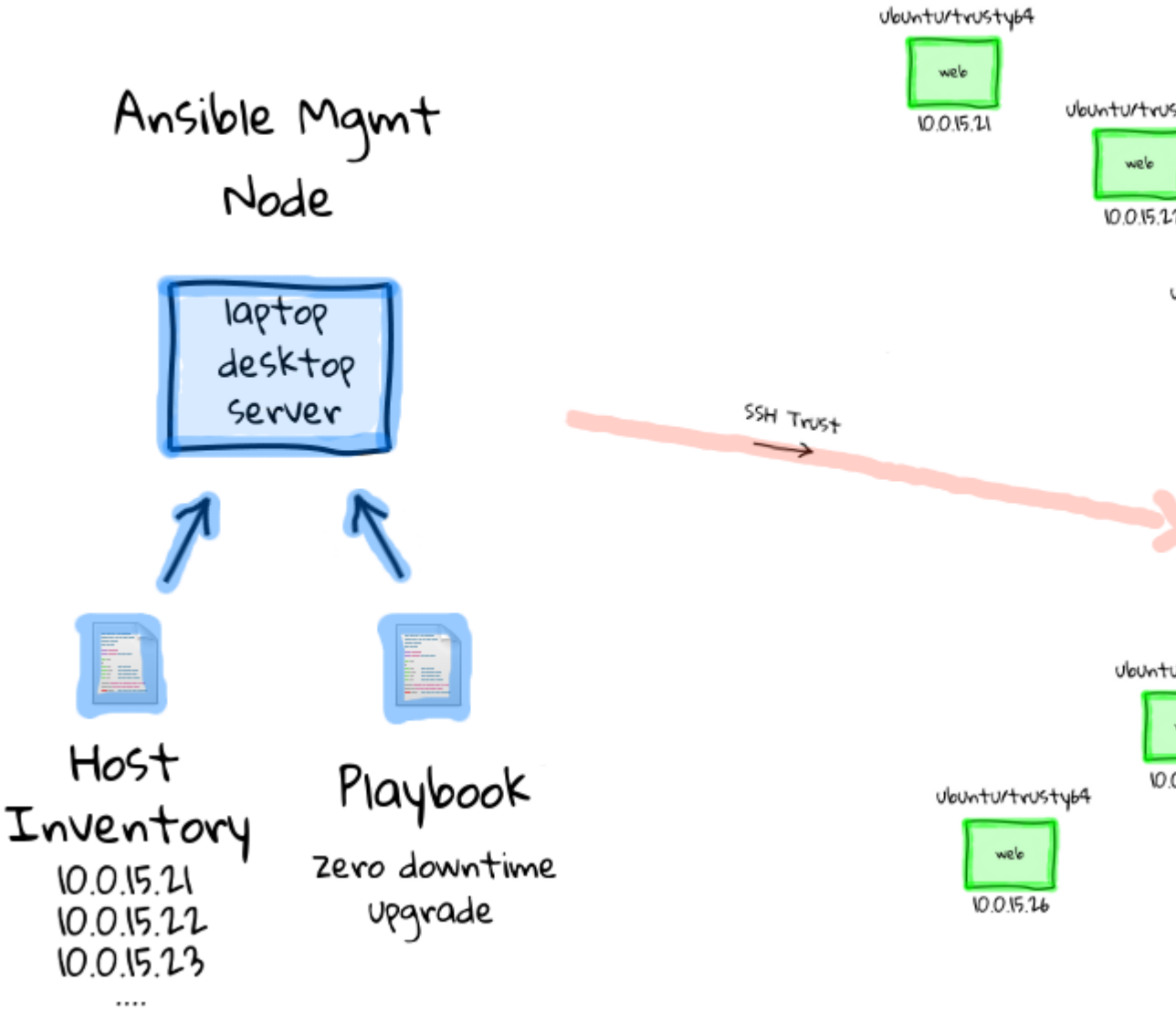
El patrón hasta reintentar se puede usar con cualquier acción; La documentación ansible proporciona un ejemplo de espera hasta que un determinado comando de shell devuelve un resultado deseado: http://docs.ansible.com/ansible/playbooks_loops.html#do-until-loops .

Lea Ansible: Bucles y Condicionales en línea: https://riptutorial.com/es/ansible/topic/3555/ansible--bucles-y-condicionales

Capítulo 6: Arquitectura ansible

Examples

Entendiendo la arquitectura ansible



La idea es tener una o más máquinas de control desde donde puede emitir comandos ad-hoc a máquinas remotas (a través de la herramienta `ansible`) o ejecutar un conjunto de instrucciones en secuencia a través de los libros de jugadas (a través `ansible-playbook` herramienta `ansible-playbook`).

Básicamente, usamos la máquina de control Ansible, que normalmente será su computadora de escritorio, computadora portátil o servidor. Luego, a partir de ahí, utiliza Ansible para eliminar los cambios de configuración, a través de ssh.

El archivo de inventario del host determina las máquinas de destino donde se ejecutarán estas

reproducciones. El archivo de configuración de Ansible se puede personalizar para reflejar la configuración de su entorno.

Lea Arquitectura ansible en línea: https://riptutorial.com/es/ansible/topic/7659/arquitectura-ansible

Capítulo 7: Bucles

Examples

Copia múltiples archivos en una sola tarea

```
- name: copy ssl key/cert/ssl_include files
  copy: src=files/ssl/{{ item }} dest=/etc/apache2/ssl/
  with_items:
    - g_chain.crt
    - server.crt
    - server.key
    - ssl_vhost.inc
```

Instalar múltiples paquetes en una sola tarea

```
- name: Installing Oracle Java and support libs
  apt: pkg={{ item }}
  with_items:
    - python-software-properties
    - oracle-java8-installer
    - oracle-java8-set-default
    - libjna-java
```

Lea Bucles en línea: https://riptutorial.com/es/ansible/topic/6095/bucles

Capítulo 8: Cifrado secreto

Observaciones

Ansible ofrece Vault (¡no debe confundirse con HashiCorp Vault !) Para manejar el cifrado de datos confidenciales. Vault se dirige principalmente a cifrar cualquier dato estructurado, como variables, tareas, manejadores.

Examples

Encriptación de datos estructurados sensibles.

Primero, cree un archivo clave, por ejemplo, `vault_pass_file` , que idealmente contiene una larga secuencia de caracteres aleatorios. En los sistemas linux, puede usar `pwgen` para crear un archivo de contraseña aleatorio:

```
pwgen 256 1 > vault_pass_file
```

Luego, use este archivo para cifrar datos confidenciales, por ejemplo, `groups_vars/group.yml` :

```
ANSIBLE_VAULT_PASSWORD_FILE=vault_pass_file ansible-vault encrypt group_vars/group.yml
```

A partir de ahora, para ejecutar un libro de jugadas, necesita el `vault_pass_file` :

```
ANSIBLE_VAULT_PASSWORD_FILE=vault_pass_file ansible-playbook -i inventories/nodes my-
playbook.yml
```

Tenga en cuenta que también puede usar el indicador `--vault-password-file vault_pass_file` lugar de configurar la variable de entorno `ANSIBLE_VAULT_PASSWORD_FILE` .

Para editar o descifrar el secreto en el disco, puede utilizar la `ansible-vault edit ansible-vault decrypt` respectivamente.

Uso de tuberías de búsqueda para descifrar datos encriptados no almacenados en bóveda

Con Vault también puede cifrar datos no estructurados, como archivos de clave privada y aún así poder descifrarlos en su juego con el módulo de `lookup` .

```
---

- name: Copy private key to destination
  copy:
    dest=/home/user/.ssh/id_rsa
    mode=0600
    content=lookup('pipe', 'ANSIBLE_VAULT_PASSWORD_FILE=vault_pass_file ansible-vault view
```

```
keys/private_key.enc')
```

Usando local_action para descifrar plantillas encriptadas en bóveda

Puede ejecutar una `local_action` que se basa en plantillas encriptadas en bóveda utilizando el módulo `local_action` .

```
---

- name: Decrypt template
  local_action: "shell {{ view_encrypted_file_cmd }} {{ role_path }}/templates/template.enc >
{{ role_path }}/templates/template"
  changed_when: False

- name: Deploy template
  template:
    src=templates/template
    dest=/home/user/file

- name: Remove decrypted template
  local_action: "file path={{ role_path }}/templates/template state=absent"
  changed_when: False
```

Tenga en cuenta el `changed_when: False` . Esto es importante en caso de que ejecute pruebas de idempotencia con sus roles de disponibilidad, de lo contrario, cada vez que ejecute el libro de jugadas se notificará un cambio. En `group_vars/all.yml` puede establecer un comando de descifrado global para reutilizarlo, por ejemplo, como `view_encrypted_file_cmd` .

group_vars / all.yml

```
---

view_encrypted_file_cmd: "ansible-vault --vault-password-file {{ lookup('env',
'ANSIBLE_VAULT_PASSWORD_FILE') }} view"
```

Ahora, al ejecutar una reproducción, debe configurar la variable de entorno `ANSIBLE_VAULT_PASSWORD_FILE` para que apunte a su archivo de contraseña de bóveda (idealmente con una ruta absoluta).

Lea Cifrado secreto en línea: https://riptutorial.com/es/ansible/topic/3355/cifrado-secreto

Capítulo 9: Cómo crear un servidor DreamHost Cloud a partir de un libro de jugadas ansible

Examples

Instalar la biblioteca de sombra

Shade es una biblioteca desarrollada por OpenStack para simplificar las interacciones con las nubes OpenStack, como DreamHost.

sombra de instalación de $ pip

Escribe un libro de jugadas para lanzar un servidor

Cree un archivo llamado `launch-server.yaml` , que será nuestro libro de jugadas.

La primera parte del libro de jugadas es una lista de hosts en los que se ejecutará su libro de jugadas, solo tenemos uno, localhost.

```
- hosts: localhost
```

Luego necesitamos definir una lista de tareas a realizar en este libro de jugadas. Solo tendremos uno que lance un servidor Ubuntu Xenial en DreamCompute.

```
tasks:
  - name: launch an Ubuntu server
```

La siguiente parte del libro de jugadas usa el `os_server` (OpenStack Server). Esto define cómo debe verse el servidor en DreamCompute.

```
os_server:
```

El primer paso es autenticar en DreamCompute; sustituya `{username}` con su nombre de usuario de DreamCompute, `{password}` con su contraseña de DreamCompute y `{project}` con su proyecto de DreamCompute. Los encontrarás en el archivo OpenStack RC .

```
  auth:
    auth_url: https://iad2.dream.io:5000
    username: {username}
    password: {password}
    project_name: {project}
```

Las siguientes líneas definen algunos elementos del nuevo servidor.

```
  state: present
  name: ansible-vm1
  image: Ubuntu-16.04
  key_name: {keyname}
  flavor: 50
  network: public
  wait: yes
```

Vamos a desglosar las pocas líneas anteriores:

- `state` es el estado del servidor, los valores posibles están `present` o `absent`
- `name` es el nombre del servidor a crear; puede ser cualquier valor
- `image` es la imagen desde la cual arrancar el servidor; los valores posibles son visibles en el panel web de DreamHost Cloud ; la variable acepta el nombre de la imagen o UUID
- `key_name` es el nombre de la clave pública que se agregará al servidor una vez que se haya creado; esta puede ser cualquier clave que ya se haya agregado a DreamCompute.
- `flavor` es el sabor del servidor para arrancar; esto define la cantidad de RAM y CPU que tendrá su servidor; la variable acepta el nombre de un sabor (gp1.semisonic) o el ID (50, 100, 200, etc.)
- `network` es la red para poner su servidor en. En el caso de DreamHost Cloud es la red `public` .
- `wait` set to yes obliga al playbook a esperar a que se cree el servidor antes de continuar.

Corriendo el libro de jugadas

Ejecutar el libro de jugadas de Ansible:

```
$ ansible-playbook launch-server.yaml
```

Deberías ver la salida como

```
PLAY [localhost]
*****************************************************************

TASK [setup]
*****************************************************************
ok: [localhost]

TASK [launch an Ubuntu server]
*********************************************
changed: [localhost]

PLAY RECAP
*********************************************************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0
```

Ahora, si verifica el tablero de DreamHost Cloud , debería ver una nueva instancia llamada "ansible-vm1"

Lea Cómo crear un servidor DreamHost Cloud a partir de un libro de jugadas ansible en línea: https://riptutorial.com/es/ansible/topic/4689/como-crear-un-servidor-dreamhost-cloud-a-partir-de-

un-libro-de-jugadas-ansible

Capítulo 10: Conviértete (Privilegio Escalada)

Introducción

A menudo, necesita ejecutar comandos con un usuario diferente u obtener privilegios de *root* . Esas opciones le permiten **convertirse en** otro usuario en el sistema invitado.

Sintaxis

- `become` : se puede establecer en verdadero o sí y activa la configuración de escalado del usuario.
- `become_user` : establecido en el usuario deseado en el host remoto.
- `become_method` : especifique el comando utilizado para iniciar sesión y cambiar de usuario.
- `become_flags` : cambiar los parámetros de inicio de sesión. Generalmente se utiliza cuando se desea cambiar a un usuario del sistema sin privilegios de shell.

Examples

Solo en una tarea

```
- name: Run script as foo user
  command: bash.sh
  become: true
  become_user: foo
```

Ejecutar todas las tareas de rol como root

```
- hosts: all
  become: true

- name: Start apache
  service: apache2
  state: started
```

Ejecutar un rol como root

```
- hosts: all
  roles:
    - { role: myrole, become: yes }
    - myrole2
```

Lea Conviértete (Privilegio Escalada) en línea:
https://riptutorial.com/es/ansible/topic/8328/conviertete--privilegio-escalada-

Capítulo 11: Galaxia

Examples

Compartiendo roles con Ansible Galaxy

También es posible compartir roles fácilmente con la comunidad o descargar roles creados por otros miembros de la comunidad con Ansible Galaxy .

Ansible se envía con una herramienta de línea de comandos llamada `ansible-galaxy` que se puede usar para instalar roles en el directorio de roles definido en el archivo `ansible.cfg` :

```
ansible-galaxy install username.rolename
```

También puede usar la herramienta Ansible Galaxy para descargar roles de otras ubicaciones, como GitHub, creando un archivo de texto con la ubicación definida como `src` :

```
- src: https://github.com/username/rolename
```

Y luego instale los roles en el archivo de texto así:

```
ansible-galaxy install -r requirements.txt
```

También puede usar la herramienta `ansible-galaxy` para crear el "andamiaje" de roles:

```
ansible-galaxy init rolename
```

Una vez que haya creado un rol y lo haya cargado en GitHub, puede compartirlo en Ansible Galaxy mediante el enlace a su repo de GitHub en Ansible Galaxy después de iniciar sesión.

Más ejemplos en el tema Galaxy .

Lea Galaxia en línea: https://riptutorial.com/es/ansible/topic/6599/galaxia

Capítulo 12: Galaxia

Examples

Comandos básicos

Rol de búsqueda en Ansible Galaxy

```
ansible-galaxy search role_name
```

Instalar el rol de Ansible Galaxy

```
ansible-galaxy install role_name
```

Más ayuda

```
ansible-galaxy --help
```

Lea Galaxia en línea: https://riptutorial.com/es/ansible/topic/6656/galaxia

Capítulo 13: Instalación

Introducción

Instalar Ansible en cualquier sistema operativo, incluido Windows usando Virtual Box y Vagrant. También está disponible una solución alternativa si solo desea practicar comandos y libros de juego ad-hoc y no desea configurar el entorno local.

Examples

Instalando Ansible en Ubuntu

Ansible mantiene un repositorio de PPA que se puede usar para instalar los archivos binarios de Ansible:

```
sudo apt-add-repository ppa:ansible/ansible -y
sudo apt-get update && sudo apt-get install ansible -y
```

Para instalar una versión específica, use `pip` . El PPA puede estar desactualizado.

Instalando Ansible en MacOS

Hay dos formas principales de instalar Ansible en OS X, ya sea utilizando Homebrew o el administrador de paquetes Pip.

Si tiene Homebrew, la última Ansible se puede instalar usando el siguiente comando:

```
brew install ansible
```

Para instalar la rama Ansible 1.9.X use el siguiente comando:

```
brew install homebrew/versions/ansible19
```

Para instalar la rama Ansible 2.0.X use el siguiente comando:

```
brew install homebrew/versions/ansible20
```

Para instalar usando pip, use el siguiente comando: `pip install ansible` .

Para instalar una versión específica, use `pip install ansible=<required version>` .

Instalación en sistemas basados en Red Hat.

Ansible se puede instalar en CentOS u otros sistemas basados en Red Hat. En primer lugar debe instalar los requisitos previos:

```
sudo yum -y update
sudo yum -y install gcc libffi-devel openssl-devel python-pip python-devel
```

Luego instale Ansible con pip:

```
sudo pip install ansible
```

Puedo recomendarle que actualice las herramientas de configuración después de la instalación:

```
sudo pip install --upgrade setuptools
```

También puede utilizar el Administrador de paquetes local también:

```
yum install ansible
```

Instalación desde la fuente

Ansible es el **mejor uso** de un pago y envío.

Se ejecuta como usted (no root) y tiene mínimas dependencias de python.

La dependencia de Python pip instala con pip:

```
sudo pip install paramiko PyYAML Jinja2 httplib2 six
```

A continuación, clona el repositorio Ansible de GitHub:

```
cd ~/Documents
git clone git://github.com/ansible/ansible.git --recursive
cd ansible
```

Finalmente, agregue la línea de script de inicialización ansible a su ~ / .bashrc o ~ / .zshrc:

```
source ~/Documents/ansible/hacking/env-setup
```

Reinicie su sesión de terminal y pruebe con

```
ansible --version
```

Instalación en Amazon Linux desde git repo

Amazon Linux es una variante de RHEL, por lo que las instrucciones de Red Hat deberían funcionar en su mayor parte. Hay, sin embargo, al menos una discrepancia.

Hubo una instancia en la que el paquete **python27-devel** , a diferencia de **python-devel** , fue explícitamente necesario.

Aquí, lo instalaremos desde la fuente.

```
sudo yum -y update
sudo yum -y install python27 python27-devel openssl-devel libffi-devel gcc git

git clone https://github.com/ansible/ansible/<search the github for a preferable branch>

cd ansible
sudo python setup.py build
sudo python setup.py install
```

Instalación de la máquina de Ansible On Any OS (windows) con Virtual Box + Vagrant

Mi computadora portátil tiene Windows 10. Aquí estoy dando pasos que puede seguir para probar y aprender Ansible.

Alguna teoria

Para Ansible, necesita una máquina de control y un host (o hosts) para ejecutar el libro de jugadas.

- **Control Machine** debe estar basado en Linux o MacOS (Windows no permitido) y necesita Python (versión 2.6 o superior). Aquí se instalará Ansible.
- **La máquina de destino** (host / nodo) puede ser Linux / MacOS / windows. Esto solo necesita Python para ser instalado. No se requiere software de agente.

PREPARAR

Paso 1: **Instalar Virtual Box**

Virtual box es un software para crear computadoras virtuales de diferentes sistemas operativos. Es como tener varias computadoras cada una o diferentes sistemas operativos y versiones diferentes.

Descargue Virtual Box de acuerdo con el sistema operativo de su sistema e instálelo.

Paso 2: **Instala Vagrant**

Vagrant es la interfaz de línea de comandos para crear máquinas virtuales en una caja virtual. Esto facilita las cosas. Necesitas aprender comandos básicos de Vagrant.

Paso 3: **Crea una carpeta donde quieras tu máquina virtual**

Paso 4: **Crea una máquina virtual usando Vagrant**

Abra el terminal y vaya a la ruta donde creó la carpeta, y ejecute los dos comandos siguientes.

Necesitas seleccionar **Virtual Box** . Estoy instalando Ubuntu por ejemplo. Puedes elegir cualquier cosa de la lista. `vagrant init ubuntu/trusty64` ejecutar estos dos comandos en **la** categoría " **caja virtual** ": `vagrant init ubuntu/trusty64` y `vagrant init ubuntu/trusty64 vagrant up --provider virtualbox` . Otras categorías pueden ser: hyperv, vmware_desktop, etc. (esto llevará algún tiempo, ya que descargará los archivos necesarios)

Paso 4: Instalar Ansible

Para UbuntuOS: `sudo apt-get install ansible`

Solución alternativa :

Puedes usar **Katacoda** para practicar ansible. No es necesario instalar ni configurar nada. Ejecute dos comandos dados en el paso 2 y después de eso, estará listo para comenzar.

Lea Instalación en línea: https://riptutorial.com/es/ansible/topic/4906/instalacion

Capítulo 14: Introducción a los libros de jugadas

Examples

Visión general

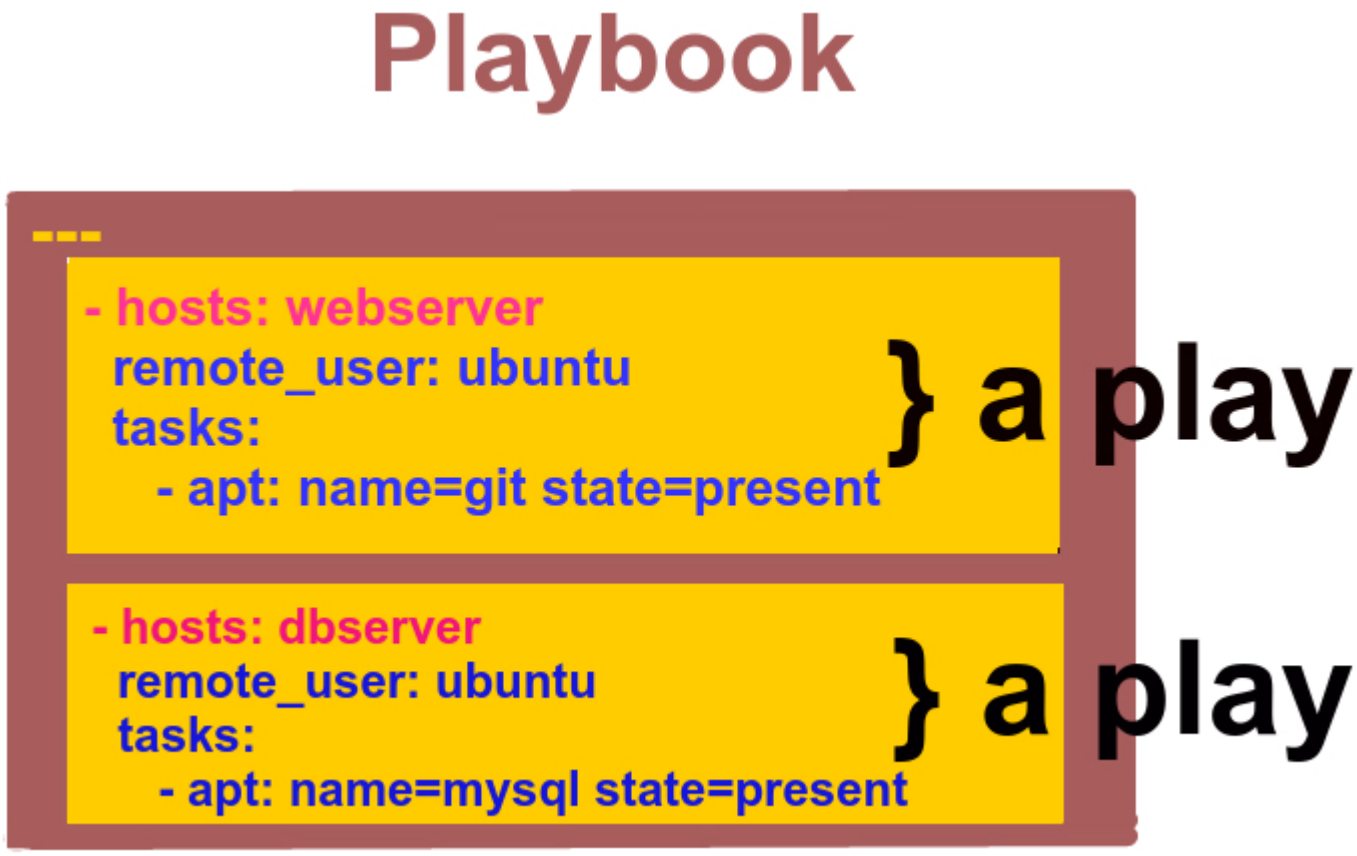
En Ansible, un libro de jugadas es un archivo YAML que contiene la definición de cómo debe verse un servidor. En un libro de jugadas, usted define qué acciones debe realizar Ansible para que el servidor se encuentre en el estado que desea. Solo se hace lo que usted define.

Este es un manual básico de Ansible que instala git en todos los hosts que pertenecen al grupo `web` :

```
---
- name: Git installation
  hosts: web
  remote_user: root
  tasks:
    - name: Install Git
      apt: name=git state=present
```

Estructura del libro de jugadas

El formato de un libro de jugadas es bastante sencillo, pero estricto en términos de espaciado y diseño. Un libro de jugadas consiste en juegos. Un juego es una combinación de hosts de destino y las tareas que queremos aplicar en estos hosts, por lo que el dibujo de un libro de jugadas es el siguiente:



Para ejecutar este libro de jugadas, simplemente ejecutamos:

```
ansible-playbook -i hosts my_playbook.yml
```

Estructura del juego

Aquí hay una jugada simple:

```
- name: Configure webserver with git
  hosts: webserver
  become: true
  vars:
    package: git
  tasks:
    - name: install git
      apt: name={{ package }} state=present
```

Como dijimos anteriormente, cada juego debe contener:

- Un conjunto de hosts para configurar
- Una lista de tareas a ejecutar en esos hosts.

Piense en un juego como lo que conecta a los hosts con las tareas. Además de especificar hosts y tareas, los juegos también admiten una serie de configuraciones opcionales. Dos comunes son:

- `name` : un comentario que describe de qué trata la obra. Ansible imprimirá esto cuando la

obra comience a correr
- `vars` : una lista de variables y valores

Etiquetas

El juego contiene varias tareas, que se pueden etiquetar:

```
- name: Install applications
  hosts: all
  become: true
  tasks:
    - name: Install vim
      apt: name=vim state=present
      tags:
        - vim
    - name: Install screen
      apt: name=screen state=present
      tags:
        - screen
```

La tarea con la etiqueta 'vim' se ejecutará cuando se especifique 'vim' en las etiquetas. Puede especificar tantas etiquetas como desee. Es útil usar etiquetas como 'instalar' o 'config'. Luego puede ejecutar el libro de jugadas especificando etiquetas o etiquetas de omisión. por

```
ansible-playbook my_playbook.yml --tags "tag1,tag2"
ansible-playbook my_playbook.yml --tags "tag2"
ansible-playbook my_playbook.yml --skip-tags "tag1"
```

Por defecto, Ansible ejecuta todas las etiquetas.

Lea Introducción a los libros de jugadas en línea:
https://riptutorial.com/es/ansible/topic/3343/introduccion-a-los-libros-de-jugadas

Capítulo 15: Inventario

Parámetros

Parámetro	Explicación
ansible_connection	Tipo de conexión al host. Este puede ser el nombre de cualquiera de los complementos de conexión de ansible. Los tipos de protocolo SSH son `smart` , `ssh` o `paramiko` . El valor predeterminado es inteligente. Los tipos no basados en SSH se describen en la siguiente sección.
ansible_host	El nombre del host al que se conectará, si es diferente del alias que desea darle.
ansible_port	El número de puerto ssh, si no 22
usuario ansible	El nombre de usuario ssh predeterminado a usar.
ansible_ssh_pass	La contraseña ssh a usar (esto es inseguro, recomendamos encarecidamente usar `--ask-pass` o SSH keys)
ansible_ssh_private_key_file	Archivo de clave privada utilizado por ssh. Es útil si usa varias teclas y no quiere usar el agente SSH.
ansible_ssh_common_args	Esta configuración siempre se agrega a la línea de comandos predeterminada para **sftp** , **scp** y **ssh** . Útil para configurar un `ProxyCommand` para un determinado host (o grupo).
ansible_sftp_extra_args	Esta configuración siempre se agrega a la línea de comando **sftp** predeterminada.
ansible_scp_extra_args	Esta configuración siempre se agrega a la línea de comandos de **scp** predeterminada.
ansible_ssh_extra_args	Esta configuración siempre se agrega a la línea de comandos predeterminada de **ssh** .
ansible_ssh_pipelining	Determina si utilizar o no la canalización SSH. Esto puede anular la configuración de la `pipelining` en `ansible.cfg` .
ansible_become	Equivalente a `ansible_sudo` o `ansible_su` , permite forzar la escalada de privilegios
ansible_become_method	Permite establecer el método de escalado de privilegios.
ansible_become_user	Equivalente a `ansible_sudo_user` o `ansible_su_user` , permite

Parámetro	Explicación
	configurar el usuario en el que se convierte a través de la escalada de privilegios
ansible_become_pass	Equivalente a `ansible_sudo_pass` o `ansible_su_pass` , le permite establecer la contraseña de escalada de privilegios
ansible_shell_type	El tipo de shell del sistema de destino. No debe usar esta configuración a menos que haya establecido el `ansible_shell_executable` en un shell no compatible con Bourne (sh). Por defecto, los comandos están formateados usando la sintaxis de estilo `sh` . Establecer esto en `csh` o `fish` hará que los comandos ejecutados en los sistemas de destino sigan la sintaxis de esos shell.
ansible_python_interpreter	La ruta de destino de python host. Esto es útil para sistemas con más de un Python o que no están ubicados en **/ usr / bin / python** como * BSD, o donde **/ usr / bin / python** no es un Python de la serie 2.X. No utilizamos el mecanismo **/ usr / bin / env** , ya que requiere que la ruta del usuario remoto se configure correctamente y también supone que el ejecutable de **python** se denomina python, donde el ejecutable podría llamarse algo así como **python2.6** .
ansible _ * _ intérprete	Funciona para cualquier cosa como ruby o perl y funciona como `ansible_python_interpreter` . Esto reemplaza a shebang de módulos que se ejecutarán en ese host.
ansible_shell_executable	Esto establece el shell que el controlador ansible usará en la máquina de destino, reemplaza el `executable` en `ansible.cfg` que por defecto es **/ bin / sh** . Realmente solo debería cambiarlo si no es posible usar **/ bin / sh** (es decir, **/ bin / sh** no está instalado en la máquina de destino o no puede ejecutarse desde sudo). Nuevo en la versión 2.1.

Examples

Inventario con nombre de usuario y contraseña.

El inventario es la forma más fácil de rastrear todos los sistemas en su infraestructura. Aquí hay un archivo de inventario simple que contiene un solo sistema y las credenciales de inicio de sesión de Ansible.

```
[targethost]
192.168.1.1 ansible_user=mrtuovinen ansible_ssh_pass=PassW0rd
```

Inventario con clave privada personalizada

```
[targethost]
192.168.1.1 ansible_user=mrtuovinen ssh_private_key_file=~/.ssh/custom_key
```

Inventario con puerto SSH personalizado

```
[targethost]
192.168.1.1 ansible_user=mrtuovinen ansible_port=2222
```

Pasar el inventario estático a ansible-playbook

```
ansible-playbook -i path/to/static-inventory-file -l myhost myplaybook.yml
```

Pasar inventario dinámico a libro de jugabilidad

```
ansible-playbook -i path/to/dynamic-inventory-script.py -l myhost myplaybook.yml
```

Ver inventario dinámico para más detalles.

Inventario, Vars de grupo, y usted

Estructura del proyecto (buenas prácticas ansible).

```
project/
  group_vars/
     development
  inventory.development
  playbook.yaml
```

todo comienza con el inventario.

```
[development]
dev.fakename.io

[development:vars]
ansible_host: 192.168.0.1
ansible_user: dev
ansible_pass: pass
ansible_port: 2232

[api:children]
development
```

que te permite enlazar a group_vars. Mantenga los datos 'específicos' para ese entorno ...

```
---
app_name: NewApp_Dev
app_url: https://dev.fakename.io
```

```
app_key: f2390f23f01233f23f
```

que permite ejecutar el siguiente libro de jugadas CONTRA el archivo de inventario:

```
---
- name: Install api.
  hosts: api
  gather_facts: true
  sudo: true
  tags:
    - api
  roles:
    - { role: api,         tags: ["api"]         }
```

con el siguiente runline:

```
ansible-playbook playbook.yaml -i  inventory.development
```

Archivo de hosts

El archivo host se utiliza para almacenar conexiones para libros de jugadas de Anisble. Hay opciones para definir parámetros de conexión:

`ansible_host` es el nombre de host o la dirección IP

`ansible_port` es el puerto que usa la máquina para SSH

`ansible_user` es el usuario remoto para conectarse como

`ansible_ssh_pass` si usa una contraseña para SSH

`ansible_ssh_private_key_file` si necesita usar varias claves que son específicas de los hosts

Estas son las opciones más utilizadas. Más se puede encontrar en la documentación oficial de Ansible .

Aquí hay un ejemplo de archivo `hosts` :

```
# Consolidation of all groups
[hosts:children]
web-servers
offsite
onsite
backup-servers

[web-servers]
server1 ansible_host=192.168.0.1 ansible_port=1600
server2 ansible_host=192.168.0.2 ansible_port=1800

[offsite]
server3 ansible_host=10.160.40.1 ansible_port=22 ansible_user=root
server4 ansible_host=10.160.40.2 ansible_port=4300 ansible_user=root
```

```
# You can make groups of groups
[offsite:children]
backup-servers

[onsite]
server5 ansible_host=10.150.70.1 ansible_ssh_pass=password

[backup-servers]
server6 ansible_host=10.160.40.3 ansible_port=77
```

Lea Inventario en línea: https://riptutorial.com/es/ansible/topic/1764/inventario

Capítulo 16: Inventario dinámico

Observaciones

Las variables de entorno en el inventario dinámico no funcionan, fe

```
"ansible_ssh_private_key_file": $HOME/.ssh/key.pem"
```

Si el servidor del inventario dinámico pasa `$HOME` por ejemplo, reemplace la variable en el código del cliente (Python):

```
json_input.replace("$HOME", os.environ.get("HOME"))
```

Examples

Inventario dinámico con credenciales de inicio de sesión.

Pase el inventario dinámico a `ansible-playbook` :

```
ansible-playbook -i inventory/dyn.py -l targethost my_playbook.yml
```

`python inventory/dyn.py` debería imprimir algo como esto:

```
{
  "_meta": {
    "hostvars": {
      "10.1.0.10": {
        "ansible_user": "vagrant",
        "ansible_ssh_private_key_file": "/home/mrtuovinen/.ssh/id_rsa",
        "ansible_port": 22
      },
      "10.1.0.11": {
        "ansible_user": "ubuntu",
        "ansible_ssh_private_key_file": "/home/mrtuovinen/.ssh/id_rsa",
        "ansible_port": 22
      },
      "10.1.0.12": {
        "ansible_user": "steve",
        "ansible_ssh_private_key_file": "/home/mrtuovinen/.ssh/key.pem",
        "ansible_port": 2222
      }
    }
  },
  "vagrantbox": [
    "10.1.0.10"
  ],
  "ubuntubox": [
    "10.1.0.11"
  ],
  "osxbox": [
```

```
    "10.1.0.12"
  ]
}
```

Lea Inventario dinámico en línea: https://riptutorial.com/es/ansible/topic/1758/inventario-dinamico

Capítulo 17: Roles

Examples

Usando roles

Ansible utiliza el concepto de roles para permitir mejor el código modular y evitar que se repita.

Un rol es simplemente una estructura de carpetas que Ansible sabe desde dónde cargar archivos vars, tareas y controladores desde. Un ejemplo podría parecer algo como esto:

```
apache/
├── defaults
│   └── main.yml
├── files
│   ├── mod-pagespeed-stable_current_i386.deb
│   ├── mod-pagespeed-stable_current_i386.rpm
│   ├── mod-pagespeed-stable_current_amd64.deb
│   └── mod-pagespeed-stable_current_x86_64.rpm
├── tasks
│   ├── debian.yml
│   ├── main.yml
│   └── redhat.yml
├── templates
│   ├── httpd.conf.j2
│   └── sites-available
│       └── virthualhost.conf.j2
└── vars
    ├── debian
    └── redhat
```

Luego puedes usar el rol con un libro de jugadas básico que se ve así:

```
- hosts: webservers
  roles:
     - apache
```

Cuando ejecute Ansible contra este libro de jugadas, se dirigirá a todos los hosts del grupo de `webservers` y ejecutará la función de `apache` definida anteriormente, cargando automáticamente las variables predeterminadas para la función y ejecutando todas las tareas incluidas en `tasks/main.yml` . Ansible también sabe buscar ciertos tipos de archivos en ubicaciones amigables con los roles:

- Si existen roles / x / tasks / main.yml, las tareas enumeradas allí se agregarán al juego
- Si los roles / x / handlers / main.yml existen, los manejadores enumerados allí se agregarán a la obra
- Si existen roles / x / vars / main.yml, las variables enumeradas allí se agregarán a la obra
- Si existen roles / x / meta / main.yml, las dependencias de los roles que se enumeran allí se

agregarán a la lista de roles (1.3 y posteriores)

- Cualquier copia, script, plantilla o tareas de inclusión (en el rol) pueden hacer referencia a archivos en roles / x / {archivos, plantillas, tareas} / (el directorio depende de la tarea) sin tener que realizar una ruta de acceso relativa o absoluta

Dependencias de roles

Los roles también le permiten definir otros roles como una dependencia al crear un archivo `meta/main.yml` con un bloque de `dependencies` :

```
dependencies:
  - role: common
```

También es posible pasar un valor a un parámetro / variable en el rol dependiente:

```
dependencies:
  - { role: common, some_parameter: 3 }
```

O incluso ejecutar el rol dependiente condicionalmente:

```
dependencies:
  - { role: common, some_parameter: 3 }
  - { role: sshd, enable_sshd: false,
                        when: environment == 'production' }
```

Los roles dependientes siempre se ejecutan antes que los roles que dependen de ellos. Además, solo se ejecutan una vez. Si dos roles establecen el mismo que su dependencia, solo se ejecutará la primera vez.

Imagine los roles role1, role2 y role3 con los siguientes `meta/main.yml` 's:
role1 / meta / main.yml:

```
dependencies:
    - role: role3
```

role2 / meta / main.yml:

```
dependencies:
    - role: role3
```

Al ejecutar role1 y role2 en el mismo libro de jugadas (con role1 llamado antes de role2), el orden de ejecución sería el siguiente:

```
role3 -> role1 -> role2
```

Puede anular este comportamiento especificando `allow_duplicates: yes` en `meta/main.yml` de role1 y role2. La orden de ejecución resultante sería:

```
role3 -> role1 -> role3 -> role2
```

Separar tareas y variables específicas de distribución dentro de un rol

Podemos separar fácilmente tareas y variables específicas de distribución en diferentes archivos .yml dedicados. Ansible nos ayuda a identificar automáticamente la distribución de los hosts de destino a través de `{{ ansible_distribution }}` y `{{ ansible_distribution_version }}` , por lo que solo tenemos que nombrar los archivos .yml dedicados a la distribución en consecuencia.

Para Ubuntu Xenial, el árbol de dir de rol básico se vería así:

```
role
├── tasks
│   ├── main.yml
│   └── Ubuntu16.04.yml
└── vars
    └── Ubuntu16.04.yml
```

Dentro de las `tasks/main.yml` ahora podemos incluir automáticamente las variables y tareas adecuadas para la distribución de los hosts de destino.

tareas / main.yml

```
---

- name: include distribution specific vars
  include_vars: "{{ ansible_distribution }}{{ ansible_distribution_version }}.yml"

- name: include distribution specific install
  include: "{{ ansible_distribution }}{{ ansible_distribution_version }}.yml"
```

Dentro de las `tasks/Ubuntu16.06.yml` y `vars/Ubuntu16.04.yml` ahora podemos definir tareas y variables para Ubuntu Xenial respectivamente.

Lea Roles en línea: https://riptutorial.com/es/ansible/topic/3396/roles

Capítulo 18: Usando Ansible con Amazon Web Services

Observaciones

ejemplo-2: Esto sirve como ejemplo, así que simplemente no lo copie / pegue. En su lugar, para satisfacer sus necesidades, debe personalizar sus variables; ansible_key, reglas de grupo de seguridad etc.

ejemplo-1: Para deshabilitar la comprobación estricta de la clave de host ssh, un comportamiento que no queremos al automatizar tareas, lo configuramos como `no` en el archivo `ansible.cfg` . es decir: `StrictHostKeyChecking=no`

El archivo `ec2.py` es un script de Python que ejecuta y devuelve sus recursos de AWS basados en el `ec2.ini` que es el archivo de configuración que necesita personalizar si desea limitar el alcance de su proyecto a algunas regiones particulares, etiquetas específicas, etc. ...

Examples

Cómo iniciar la instancia de EC2 desde las AMI oficiales de Amazon, modificarla y almacenarla como nueva AMI

Este es un flujo de trabajo muy común cuando se utiliza Ansible para aprovisionar una instancia de AWS EC2. Esta publicación supone un conocimiento básico de Ansible y, lo que es más importante, supone que la ha configurado correctamente para conectarse a AWS.

Como insiste la documentación oficial de Ansible , vamos a utilizar cuatro roles:

1- **ami_find** para obtener el ID de ami en función del cual **lanzaremos** nuestra instancia de EC2.

2- **ec2_ami_creation** para lanzar efectivamente la instancia de EC2.

3- **code_deploy** para modificar la instancia; Esto podría ser cualquier cosa, así que simplemente transferiremos un archivo a la máquina de destino.

4- **build_ami** para construir nuestra nueva imagen basada en la instancia de ec2 en ejecución.
Esta publicación asume que estás en el nivel superior de tu proyecto de Ansible: `my_ansible_project`

El primer rol: **ami_find**

```
cd my_ansible_project/roles && ansible-galaxy init ami_find
```

En este rol vamos a usar el módulo ec2_ami_find y, como ejemplo, buscaremos una máquina Ubuntu y obtendremos su **ami_id** (ami-xxxxxxxx). Ahora edite el

`my_ansible_project/roles/ami_find/tasks/main.yml` :

```
---
- ec2_ami_find:
    name: "ubuntu/images/hvm-ssd/ubuntu-trusty-14.04-amd64-server-*"
    sort: name
    sort_order: descending
    sort_end: 1
    region: "{{ aws_region }}"
  register: ami_find
- set_fact: ami_ubuntu="{{ ami_find.results[0].ami_id }}"
```

El segundo rol: **ec2_ami_creation**

Aquí, usaremos el `ami_id` que obtuvimos del primer rol y luego `ami_id` nuestra nueva instancia basada en él:

```
cd my_ansible_project/roles && ansible-galaxy init ec2_ami_creation
```

En este rol, vamos a utilizar lo más importante el módulo ec2 para lanzar nuestra instancia. Ahora edite `my_ansible_project/roles/ec2_ami_creation/tasks/main.yml` file:

```
---
- ec2_vpc_subnet_facts:
    region: "{{aws_region}}"
  register: vpc
- name: creation of security group of the ec2 instance
  ec2_group:
    name: example
    description: an example EC2 group
    region: "{{ aws_region }}"
    rules:
      - proto: tcp
        from_port: 22
        to_port: 22
        cidr_ip: 0.0.0.0/0
    state: present
  register: ec2_sg

- name: create instance using Ansible
  ec2:
    key_name: "{{ ansible_key }}"
    group: example
    vpc_subnet_id: "{{vpc.subnets[0].id}}"
    instance_type: "{{ instance_type }}"
    ec2_region: "{{ aws_region }}"
    image: "{{ base_image }}"
    assign_public_ip: yes
    wait: yes
  register: ec2

- set_fact: id={{ec2.instances[0].id}}

- name: adding the newly created instance to a temporary group in order to access it later
from another play
  add_host: name={{ item.public_ip }} groups=just_created
  with_items: ec2.instances
```

```
- name: Wait for SSH to come up
  wait_for: host={{ item.public_dns_name }} port=22 delay=10 timeout=640 state=started
  with_items: ec2.instances
```

El tercer rol: **code_deploy**

Aquí, aprovisionaremos esta instancia, que se agregó a un grupo llamado `just_created`

```
cd my_ansible_project/roles && ansible-galaxy init code_deploy
```

En este rol, vamos a utilizar el template_module para transferir un archivo y escribir el nombre de host de la máquina en él. Ahora edite el `my_ansible_project/roles/code_deploy/tasks/main.yml` :

```
---
- template: src=my_file.txt.j2 dest=/etc/my_file.txt
```

Luego, muévase a la carpeta de plantillas dentro de su rol:

`cd my_ansible_project/roles/templates` y agregue un archivo llamado `my_file.txt.j2` contiene:

```
my name is {{ ansible_hostname }}`
```

El cuarto papel: `build_ami`

Ahora crearemos una imagen de la instancia en ejecución usando el módulo ec2_ami . Mueve a tu carpeta de proyectos y:

```
 cd my_ansible_project/roles && ansible-galaxy init build_ami
```

Ahora edite el `my_ansible_project/roles/build_ami/tasks/main.yml` :

```
---
- ec2_ami:
    instance_id: "{{ instance_id }}"
    wait: yes
    name: Base_Image
```

Ahora, creo que te has estado preguntando cómo organizar todos estos roles. Estoy en lo cierto Si es así, sigue leyendo.

Escribiremos un libro de jugadas, compuesto por tres jugadas: la primera jugada aplicable en `localhost` llamará a nuestras dos primeras funciones, la segunda jugada se aplicará a nuestro grupo **just_created** . El último rol será aplicable en `localhost` . ¿Por qué `localhost` ? Cuando queremos **administrar** algunos recursos de AWS, usamos nuestra máquina local, tan simple como eso. A continuación, usaremos un archivo vars en el que pondremos nuestras variables: `ansible_key` , `aws_region` , etc.

cree una carpeta de infraestructura en la parte superior de su proyecto y agregue un archivo

dentro llamado `aws.yml` :

```
---
aws_region: ap-southeast-2
ansible_key: ansible
instance_type: t2.small
```

Así que en la parte superior de tu proyecto crea `build_base_image.yml` y agrega esto:

```
---
   - hosts: localhost
     connection: local
     gather_facts: False
     vars_files:
       - infrastructure/aws.yml
     roles:
       - ami_find
       - { role: ec2_creation, base_image: "{{ ami_ubuntu }}"}

   - hosts: just_created
     connection: ssh
     gather_facts: True
     become: yes
     become_method: sudo
     roles:
       - code_deploy

   - hosts: localhost
     connection: local
     gather_facts: False
     vars_files:
       - infrastructure/aws.yml
     roles:
       - { role: new_image, instance_id: "{{ id }}"}
```

Eso es todo. No olvide eliminar sus recursos después de probar esto, o por qué no crear una función para eliminar la instancia en ejecución :-)

Cómo configurar correctamente Ansible para conectarse a los servicios web de Amazon

La administración de los recursos de AWS que escalan hacia arriba y hacia abajo se encuentra dentro de los límites del archivo host de inventario estático, es por eso que necesitamos algo dinámico. Y para eso están los inventarios dinámicos . Empecemos:

Descargue estos archivos ec2.ini y ec2.py en la carpeta de su proyecto:

```
cd my_ansible_project
wget https://raw.githubusercontent.com/ansible/ansible/devel/contrib/inventory/ec2.py
wget https://raw.githubusercontent.com/ansible/ansible/devel/contrib/inventory/ec2.ini
```

Una vez hecho esto, haga el archivo `ec2.py` ejecutable:

```
chmod +x ec2.py
```

Ahora, exporte su clave de acceso y secreto de AWS como variables de entorno:

```
export AWS_ACCESS_KEY_ID='ABCDEFGHIJKLM'
export AWS_SECRET_ACCESS_KEY='NOPQRSTUVWXYZ'
```

Para usar el script `ec2.py` necesitamos el SDK de Python AWS, `boto` , por lo que necesita instalarlo:

```
sudo pip install boto
```

Para probar si todo está bien, intente ejecutar el `ec2.py` enumerando sus recursos:

```
./ec2.py --list
```

Deberías ver algo similar a:

```
{
  "_meta": {
    "hostvars": {}
  }
}
```

Ahora queremos usar el inventario dinámico junto con nuestro archivo de hosts estáticos. Primero, cree una carpeta llamada `inventory` , agregue `ec2.py` , `ec2.ini` y nuestro archivo de `hosts` y luego diga a Ansible que use esa carpeta como un archivo de inventario:

```
mkdir inventory
mv ec2.py inventory/ec2.py
mv ec2.ini inventory/ec2.ini
mv hosts inventory/hosts
```

A continuación, debemos definir la configuración a nivel de proyecto para Ansible creando un archivo de configuración de Ansible en su carpeta de proyecto llamada `ansible.cfg` y agregando esto:

```
[defaults]
hostfile = inventory
[ssh_connection]
pipelining = False
ssh_args = -o ControlMaster=auto -o ControlPersist=30m -o StrictHostKeyChecking=no
```

A continuación, debemos configurar Ansible para usar una clave SSH para autenticar el acceso a nuestras instancias de EC2. Usar un agente de SSH es la mejor manera de autenticarse con recursos, ya que esto facilita la administración de claves:

```
ssh-agent bash
ssh-add ~/.ssh/keypair.pem
```

¡Eso es! Si siguió esto, puede probarlo utilizando el módulo `ping` y luego verá las instancias en

ejecución que se han configurado para usar su clave respondiendo con pong:

```
ansible -m ping all
11.22.33.44 | success >> {
    "changed": false,
    "ping": "pong"
}
```

Lea Usando Ansible con Amazon Web Services en línea:
https://riptutorial.com/es/ansible/topic/3302/usando-ansible-con-amazon-web-services

Capítulo 19: Usando Ansible con OpenStack

Introducción

OpenStack es una plataforma de software de código abierto para cloud computing. Las instancias de Linux se pueden iniciar / detener de forma manual mediante la interfaz web gráfica o automatizadas gracias al módulo de nube openstack de ansible.

La configuración de ansible puede ser complicada, pero una vez bien configurada, su uso es realmente fácil y potente para las pruebas y el entorno de integración continua.

Parámetros

parámetros	Comentarios
anfitriones: localhost	Los comandos de OpenStack se inician desde nuestro host local
recopilación de datos: falso	No necesitamos recopilar información en nuestro localhost
auth_url: https://openstack-identity.mycompany.com/v2.0	usar la URL V2.0
estado: presente	'presente' / 'ausente' para crear / eliminar la instancia
validate_certs: False	útil si https usa certificados autofirmados
red: "{{nombre de red}}"	(Opcional)
auto_ip: si	(Opcional)

Observaciones

- Ponemos la URL de autenticación directamente en el libro de jugadas, no en una variable. La URL usada en vars debe ser escapada.
- Tenga cuidado con la versión URL de autenticación, use V2.0 en lugar de V3 en https://openstack-identity.mycompany.com/v2.0 .
- En archivos yml, tenga mucho cuidado al copiar / pegar desde el navegador. Comprueba dos veces los espacios según se tengan en cuenta.
- Más detalles en: http://docs.ansible.com/ansible/list_of_cloud_modules.html#openstack

Examples

Revisa tu versión Ansible

Verifique que estén instaladas las versiones correctas de software:

- ansible> = 2.0
- python> = 2.6
- módulo de sombra para python

```
$ansible --version
ansible 2.2.0.0

$python --version
Python 2.7.5
```

Instale 'sombrear' el componente de Python usado para pilotar OpenStack.

```
$pip install shade
```

Nota: si utiliza un proxy de la empresa, siempre es útil conocer el pip synthax correcto.

```
$pip install --proxy proxy_ip:proxy_port shade
```

Recopile información de OpenStack GUI para configurar Ansible

---

Nuestro inquilino openstack ya está establecido:

- un lan virtual da instancias de IP privada
- un enrutador virtual mapea IP pública a IP privada
- Se ha generado una clave de seguridad.
- Tenemos configuración de firewall por defecto para ssh y el puerto 80
- Podemos iniciar una instancia gracias a la interfaz web de OpenStack.

Deje recopilar toda la información necesaria de esta interfaz web.

Las informaciones de autenticación se pueden encontrar en el archivo openstack.rc. este archivo se puede descargar utilizando la interfaz web de OpenStack en [acceso y seguridad / acceso a API].

```
$cat openstack.rc
#!/bin/bash

# To use an OpenStack cloud you need to authenticate against the Identity
# service named keystone, which returns a **Token** and **Service Catalog**.
# The catalog contains the endpoints for all services the user/tenant has
# access to - such as Compute, Image Service, Identity, Object Storage, Block
# Storage, and Networking (code-named nova, glance, keystone, swift,
# cinder, and neutron).
#
# *NOTE*: Using the 2.0 *Identity API* does not necessarily mean any other
# OpenStack API is version 2.0. For example, your cloud provider may implement
```

```
# Image API v1.1, Block Storage API v2, and Compute API v2.0. OS_AUTH_URL is
# only for the Identity API served through keystone.
export OS_AUTH_URL=https://openstack-identity.mycompany.com/v3

# With the addition of Keystone we have standardized on the term **tenant**
# as the entity that owns the resources.
export OS_TENANT_ID=1ac99fef77ee40148d7d5ba3e070caae
export OS_TENANT_NAME="TrainingIC"
export OS_PROJECT_NAME="TrainingIC"

# In addition to the owning entity (tenant), OpenStack stores the entity
# performing the action as the **user**.
export OS_USERNAME="UserTrainingIC"

# With Keystone you pass the keystone password.
echo "Please enter your OpenStack Password: "
read -sr OS_PASSWORD_INPUT
export OS_PASSWORD=$OS_PASSWORD_INPUT

# If your configuration has multiple regions, we set that information here.
# OS_REGION_NAME is optional and only valid in certain environments.
export OS_REGION_NAME="fr"
# Don't leave a blank variable, unset it if it was empty
if [ -z "$OS_REGION_NAME" ]; then unset OS_REGION_NAME; fi
```

Obtenemos OS_AUTH_URL, OS_TENANT_NAME, OS_USERNAME.

Versión de la API de autenticación: OS_AUTH_URL

Cuidado con la versión de la API de autenticación. Por defecto v3 está activado, pero ansible necesita la v2.0. Obtenemos la url y configuramos V2.0 en lugar de V3: https://openstack-identity.mycompany.com/v2.0

Informaciones de VM

Cree una instancia utilizando la interfaz web de OpenStack y obtenga el nombre de la imagen, el sabor, la clave, la red y el grupo de seguridad.

Cree un archivo ./group_vars/all con toda la información necesaria.

```
$vi ./group_vars/all
# Authentication
AuthUserName: UserTrainingIC
AuthPassword: PasswordTrainingIC
TenantName: TrainingIC

# VM infos
ImageName: CentOS-7-x86_64-GenericCloud-1607
FlavorName: m1.1cpu.1gb
InfraKey: KeyTrainingIC
NetworkName: NetPrivateTrainingIC
SecurityGroup: default
```

Escribe el libro de jugadas ansible para crear la instancia.

Deje usar el comando 'os_server' del módulo 'Cloud' [

http://docs.ansible.com/ansible/os_server_module.html] . Las variables se definen en ./group_vars/all.

```
$vi launch_compute.yml
- name: launch a compute instance
  hosts: localhost
  gather_facts: False
  tasks:
  - name: Create and launch the VM
    os_server:
      auth:
        auth_url: https://openstack-identity.mycompany.com/v2.0
        username: "{{ AuthUserName }}"
        password: "{{ AuthPassword }}"
        project_name: "{{ TenantName }}"
      state: present
      validate_certs: False
      name: "MyOwnPersonalInstance"
      image: "{{ ImageName }}"
      key_name: "{{ InfraKey }}"
      timeout: 200
      flavor:   "{{ FlavorName }}"
      security_groups: "{{ SecurityGroup }}"
      network: "{{ NetworkName }}"
      auto_ip: yes
```

```
$ ansible-playbook  -s launch_compute.yml
[WARNING]: provided hosts list is empty, only localhost is available
PLAY [launch a compute instance] ***********************************************
TASK [Create and launch the VM] ************************************************
changed: [localhost]
PLAY RECAP *********************************************************************
localhost                  : ok=1    changed=1    unreachable=0    failed=0
```

Recopila información sobre nuestra nueva instancia.

Utilice el comando 'os_server_facts' del módulo 'Cloud' [http://docs.ansible.com/ansible/os_server_module.html] . Las variables se definen en ./group_vars/all y el nombre de la instancia está en el servidor: "MyOwnPersonalInstance".

```
$vi get_compute_info.yml
- name: Get and print instance IP
  hosts: localhost
  gather_facts: False
  tasks:
  - name: Get VM infos
    os_server_facts:
      auth:
        auth_url: https://openstack-identity.mygroup/v2.0
        username: "{{ AuthUserName }}"
        password: "{{ AuthPassword }}"
        project_name: "{{ TenantName }}"
      validate_certs: False
      server: "MyOwnPersonalInstance"

  - name: Dump all
    debug:
```

```
      var: openstack_servers
```

```
$ansible-playbook  -s get_compute_info.yml
[WARNING]: provided hosts list is empty, only localhost is available
PLAY [Get and print instance IP] ***********************************************
TASK [Get VM IP] ***************************************************************
ok: [localhost]
TASK [Affichage] ***************************************************************
ok: [localhost] => {
    "openstack_servers": [
        {
            "OS-DCF:diskConfig": "MANUAL",
            "OS-EXT-AZ:availability_zone": "fr",
            "OS-EXT-STS:power_state": 1,
            "OS-EXT-STS:task_state": null,
[...]
            "volumes": []
        }
    ]
}

PLAY RECAP *********************************************************************
localhost                  : ok=2    changed=0    unreachable=0    failed=0
```

Esto es muy detallado. Se muestra mucha información. Por lo general, solo se necesita la dirección IP para acceder a la nueva instancia a través de SSH.

Obtenga su nueva instancia de IP pública

En lugar de imprimir toda la información, imprimimos solo la dirección IP de la primera instancia cuyo nombre es "MyOwnPersonalInstance". Usualmente es todo lo que necesitamos.

```
$vi get_compute_ip.yml
- name: Get and print instance IP
  hosts: localhost
  gather_facts: False
  tasks:
  - name: Get VM infos
    os_server_facts:
      auth:
        auth_url: https://openstack-identity.mycompany.com/v2.0
        username: "{{ AuthUserName }}"
        password: "{{ AuthPassword }}"
        project_name: "{{ TenantName }}"
      validate_certs: False
      server: "MyOwnPersonalInstance"

  - name: Dump IP
    debug:
      var: openstack_servers[0].interface_ip
```

Borra nuestra instancia

Para eliminar nuestra instancia, reutilice el comando os_server con toda la información de autenticación y simplemente reemplace 'estado: presente' por 'estado: ausente'.

```
$vi stop_compute.yml
- name: launch a compute instance
  hosts: localhost
  gather_facts: False
  tasks:
  - name: Create and launch the VM
    os_server:
      auth:
        auth_url: https://openstack-identity.mygroup/v2.0
        username: "{{ AuthUserName }}"
        password: "{{ AuthPassword }}"
        project_name: "{{ ProjectName }}"
      state: absent
      validate_certs: False
      name: "{{ TPUser }}"
      timeout: 200
```

Lea Usando Ansible con OpenStack en línea: https://riptutorial.com/es/ansible/topic/8712/usando-ansible-con-openstack

Capítulo 20: Variables del grupo ansible

Examples

Agrupar variables con inventario estático.

Se sugiere que defina grupos según el propósito del host (roles) y también la ubicación geográfica o del centro de datos (si corresponde):

Archivo de `inventory/production`

```
[rogue-server]
192.168.1.1

[atlanta-webservers]
www-atl-1.example.com
www-atl-2.example.com

[boston-webservers]
www-bos-1.example.com
www-bos-2.example.com

[atlanta-dbservers]
db-atl-1.example.com
db-atl-2.example.com

[boston-dbservers]
db-bos-1.example.com

# webservers in all geos
[webservers:children]
atlanta-webservers
boston-webservers

# dbservers in all geos
[dbservers:children]
atlanta-dbservers
boston-dbservers

# everything in the atlanta geo
[atlanta:children]
atlanta-webservers
atlanta-dbservers

# everything in the boston geo
[boston:children]
boston-webservers
boston-dbservers
```

Archivo `group_vars/all`

```
---
apache_port: 80
```

Archivo `group_vars/atlanta-webservers`

```
---
apache_port: 1080
```

Archivo `group_vars/boston-webservers`

```
---
apache_port: 8080
```

Archivo `host_vars/www-bos-2.example.com`

```
---
apache_port: 8111
```

Después de ejecutar `ansible-playbook -i inventory/hosts install-apache.yml` (los hosts en el libro de jugadas serían `hosts: all`)

Los puertos serían

Dirección	Puerto
`192.168.1.1`	80
`www-atl-1.example.com`	1080
`www-atl-2.example.com`	1080
`www-bos-1.example.com`	8080
`www-bos-2.example.com`	8111

Lea Variables del grupo ansible en línea: https://riptutorial.com/es/ansible/topic/6544/variables-del-grupo-ansible

Creditos

S. No	Capítulos	Contributors
1	Empezando con ansible	activatedgeek, Alex, baptistemm, calvinmclean, Community, Jake Amey, jasonz, jscott, Michael Duffy, mrtuovinen, Pants, PumpkinSeed, tedder42, thisguy123, ydaetskcoR
2	Ansible Group Vars	Nick, Peter Mortensen
3	Ansible instalar mysql	Fernando
4	Ansible: bucle	calvinmclean
5	Ansible: Bucles y Condicionales	A K, Chu-Siang Lai, Jordan Anderson, marx, Mike, mrtuovinen, Nick, Rob H, wolfaviators
6	Arquitectura ansible	Jordan Anderson, Yogesh Darji
7	Bucles	marx, mrtuovinen
8	Cifrado secreto	fishi
9	Cómo crear un servidor DreamHost Cloud a partir de un libro de jugadas ansible	Stefano Maffulli
10	Conviértete (Privilegio Escalada)	Jordan Anderson, Willian Paixao
11	Galaxia	mrtuovinen, ydaetskcoR
12	Instalación	ca2longoria, Jake Amey, Michael Duffy, mrtuovinen, Nick, PumpkinSeed, Raj, tedder42, ydaetskcoR
13	Introducción a los libros de jugadas	32cupo, Abdelaziz Dabebi, ydaetskcoR
14	Inventario	calvinmclean, mrtuovinen, Nick
15	Inventario dinámico	mrtuovinen
16	Roles	Chu-Siang Lai, fishi, mrtuovinen, winston, ydaetskcoR

17	Usando Ansible con Amazon Web Services	Abdelaziz Dabebi, another geek, ydaetskcoR
18	Usando Ansible con OpenStack	BANANENMANNFRAU, Sebastien Josset
19	Variables del grupo ansible	mrtuovinen