Бесплатная электронная книга

УЧУСЬ

ansible

Free unaffiliated eBook created from
Stack Overflow contributors.

#ansible

1

1: 2

2

Examples 2

, 2

ping 3

3

Ansible 3

ansible.cfg 4

2: Loops 11

Examples 11

11

11

3: playbooks 12

Examples 12

12

Playbook 12

13

14

4: 15

Examples 15

Ansible Galaxy 15

5: 16

Examples 16

16

6: 17

17

Examples 17

17

7: 19

19

Examples 21
21
21
SSH- 21
21
21
, 21
22
8: Ansible OpenStack 24
24
24
24
Examples 25
Ansible 25
GUI OpenStack, Ansible 25
27
27
IP- 28
29
9: Ansible - Amazon 30
30
Examples 30
EC2 AMI Amazon, AM 30
Ansible Amazon Web Services 33
10: mysql 36
36
Examples 36
mysql 36
11: DreamHost Ansible Playbook 38
Examples 38
Shade 38
38

Playbook 39
12: 41
41
Examples 41
Ansible Ubuntu 41
Ansible MacOS 41
Red Hat 42
42
Amazon Linux git repo 43
Ansible (Windows) Virtual Box + 43
: 44
13: : 45
45
Examples 45
? 45
[] : `ansible_os_family` 45
45
46
46
46
47
47
47
48
`ansible_os_family` ` ansible_pkg_mgr` 48
«» () 49
49
14: : 51
Examples 51
with_items - 51
with_items - 51
with_items - 51

with_items - 52
52
15: Vars 54
Examples 54
group_vars / development 54
16: 55
Examples 55
55
17: 57
Examples 57
57
58
59
18: 60
60
Examples 60
60
, 60
local_action 61
19: 62
Examples 62
62
20: ( ) 64
64
64
Examples 64
64
root 64
root 64
66

Около

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

глава 1: Начало работы с помощью

замечания

В этом разделе представлен обзор того, что возможно, и почему разработчик может захотеть его использовать.

Следует также упомянуть о любых крупных предметах в пределах возможного и связать их с соответствующими темами. Поскольку Documentation for ansible является новым, вам может потребоваться создать начальные версии этих связанных тем.

Examples

Привет, мир

Создайте каталог под названием `ansible-helloworld-playbook`

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

Создайте `hosts` файлов и добавьте удаленные системы, как хотите управлять. Поскольку ssh полагается на ssh для подключения компьютеров, вы должны убедиться, что они уже доступны вам в ssh с вашего компьютера.

```
192.168.1.1
192.168.1.2
```

Проверьте соединение с удаленными системами с помощью модуля Ansible ping .

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

В случае успеха он должен вернуть что-то подобное

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

В случае ошибки он должен вернуться

```
192.168.1.1| UNREACHABLE! => {
    "changed": false,
```

```
    "msg": "Failed to connect to the host via ssh.",
    "unreachable": true
}
```

Проверить доступ к sudo с помощью

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

Проверить соединение и настройку с помощью ping

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

`-i hosts` определяет путь к файлу инвентаризации
`targethost` - это имя хоста в файле `hosts`

инвентарь

Инвентарь - это Ansible способ отслеживать все системы в вашей инфраструктуре. Вот простой статический файл инвентаризации, содержащий одну систему и учетные данные для входа в Ansible.

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

Написать эти строки, например , для `hosts` файла и передать файл `ansible` или `ansible-playbook` команду с `-i` / `--inventory-file` флагом.

Для получения более подробной информации см. Статические запасы и динамические ресурсы .

Предоставление удаленных машин с помощью Ansible

Мы можем предоставить удаленные системы Ansible. У вас должна быть пара ключей SSH, и вы должны использовать открытый ключ SSH для файла ~ / .ssh / authorized_keys машины. Вы можете войти в систему без авторизации.

Предпосылки:

- анзибль

Вам нужен файл инвентаризации (например, development.ini), в котором вы определяете хост, что вы хотите использовать:

```
[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"
```

- hostname - имя хоста удаленной машины
- ansible_ssh_host - ip или домен удаленного хоста
- ansible_port - порт удаленного хоста, который обычно равен 22
- ansible_connection - соединение, в котором мы устанавливаем, мы хотим подключиться к ssh
- ansible_user - пользователь ssh
- ansible_ssh_extra_args - дополнительные аргументы, которые вы хотите указать для подключения ssh

Требуемые дополнительные аргументы для ssh:

- StrictHostKeyChecking - он может запросить ключ, проверяющий, что ждет да или нет. Ansible не может ответить на этот вопрос, а затем выбросить ошибку, хост недоступен.
- UserKnownHostsFile - требуется для параметра StrictHostKeyChecking.

Если у вас есть этот файл инвентаря, вы можете написать тестовый playbook.yml:

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

то вы можете начать предоставление:

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

ansible.cfg

Это ansible.cfg по умолчанию из 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
```

Поместите эту конфигурацию в корень ваших каталогов ролей, чтобы изменить поведение Ansible при использовании этой роли. Например, вы можете установить его, чтобы остановить создание `playbook.retry` при неудачных играх с книгами или указать на секретные вары, которые вы не хотите в своем репозитории git.

Прочитайте Начало работы с помощью онлайн: https://riptutorial.com/ru/ansible/topic/826/начало-работы-с-помощью

глава 2: Loops

Examples

Скопировать несколько файлов в одну задачу

```
- 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
```

Установка нескольких пакетов в одной задаче

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

Прочитайте Loops онлайн: https://riptutorial.com/ru/ansible/topic/6095/loops

глава 3: Введение в playbooks

Examples

обзор

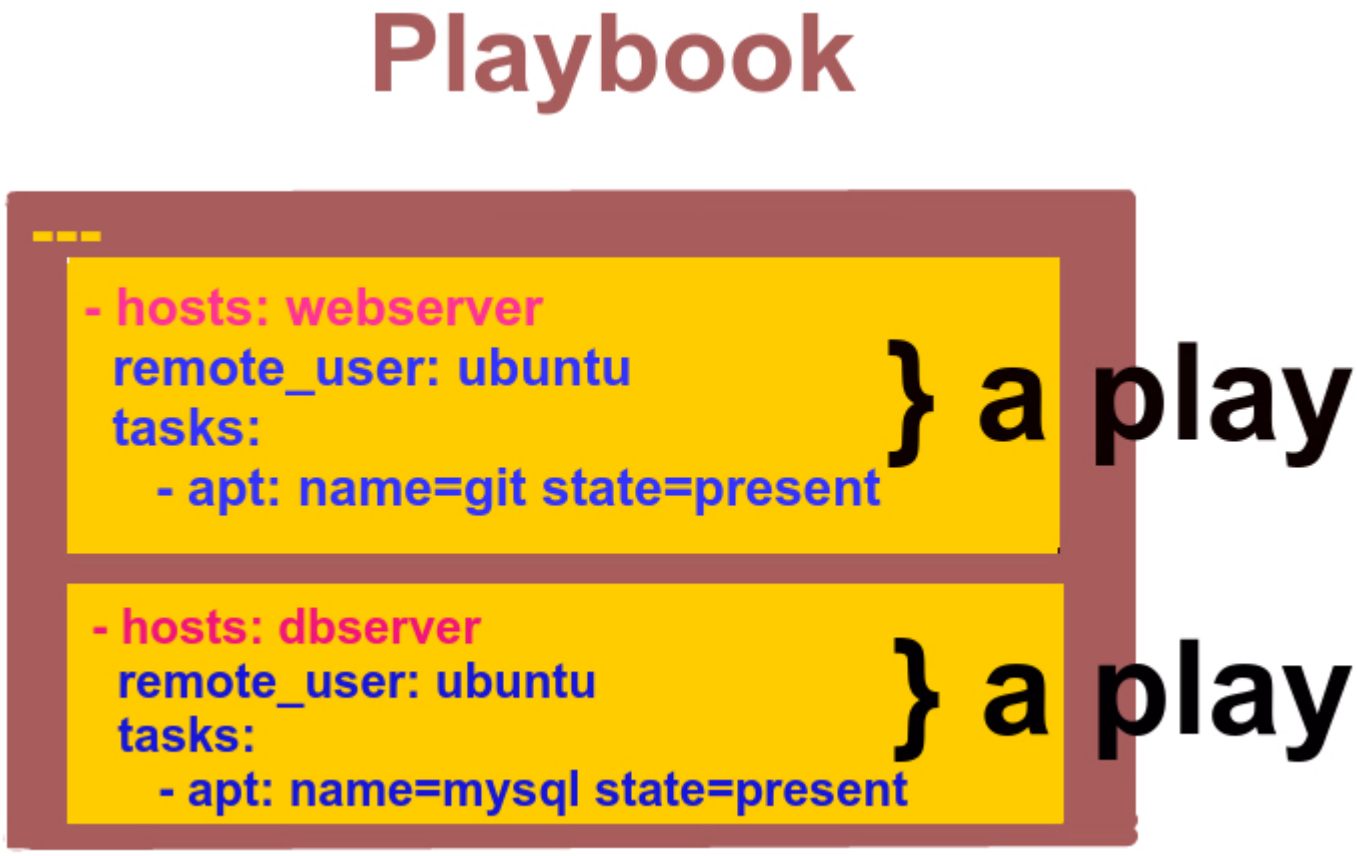
В Ansible, playbook - это файл YAML, содержащий определение того, как должен выглядеть сервер. В игровой книге вы определяете, какие действия должны предпринять Ansible, чтобы получить сервер в нужном вам состоянии. Выполняется только то, что вы определяете.

Это базовая загрузочная игра Ansible, которая устанавливает git на каждый хост, принадлежащий к `web` группе:

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

Структура Playbook

Формат пьесы довольно прост, но строгий с точки зрения расстояния и макета. Плей-лист состоит из пьес. Игра представляет собой комбинацию целевых хостов и задач, которые мы хотим применить на этих хостах, поэтому рисунок учебника:



Для выполнения этой пьесы мы просто запускаем:

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

Структура игры

Вот простая игра:

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

Как мы говорили ранее, каждая игра должна содержать:

- Набор хостов для настройки
- Список задач, которые должны выполняться на этих хостах

Подумайте о игре как о том, что связывает хосты с задачами. Помимо указания хостов и задач, игры также поддерживают ряд дополнительных настроек. Два общих:

- `name`

: комментарий, который описывает, о чем идет речь. Ansible распечатает это, когда начнется воспроизведение

- `vars` : список переменных и значений

Теги

Игра содержит несколько задач, которые могут быть отмечены:

```
- 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
```

Задача с тегом «vim» будет выполняться, когда «vim» указан в тегах. Вы можете указать столько тегов, сколько хотите. Полезно использовать теги, такие как «install» или «config». Затем вы можете запустить playbook с указанием тегов или пропусков. За

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

По умолчанию Ansible запускает все теги

Прочитайте Введение в playbooks онлайн: https://riptutorial.com/ru/ansible/topic/3343/введение-в-playbooks

глава 4: галактика

Examples

Совместное использование роли в Ansible Galaxy

Также можно легко разделить роли с сообществом или роли загрузки, которые были созданы другими членами сообщества с Ansible Galaxy .

Необычные корабли с инструментом командной строки, называемым `ansible-galaxy` которые могут быть использованы для установки ролей в каталоге ролей, определенном в файле `ansible.cfg` :

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

Вы также можете использовать инструмент Ansible Galaxy для загрузки ролей из других мест, таких как GitHub, путем создания текстового файла с местоположением, определенным как `src` :

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

Затем установите роли в текстовом файле следующим образом:

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

Вы также можете использовать инструмент `ansible-galaxy` для создания роли «строительные леса»:

```
ansible-galaxy init rolename
```

После того, как вы создали роль и загрузили ее в GitHub, вы можете поделиться ею в Ansible Galaxy, связавшись с вашим репо GitHub в Ansible Galaxy после входа в систему.

Дополнительные примеры в разделе Галактики .

Прочитайте галактика онлайн: https://riptutorial.com/ru/ansible/topic/6599/галактика

глава 5: галактика

Examples

Основные команды

Роль поиска в Ansible Galaxy

```
ansible-galaxy search role_name
```

Установить роль из Ansible Galaxy

```
ansible-galaxy install role_name
```

Дополнительная помощь

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

Прочитайте галактика онлайн: https://riptutorial.com/ru/ansible/topic/6656/галактика

глава 6: Динамический инвентарь

замечания

Переменные среды в динамических ресурсах не будут работать.

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

Если, например, сервер динамического инвентаризации передает `$HOME` , замените переменную в коде клиента (Python):

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

Examples

Динамическая инвентаризация с учетными данными

Передача динамических ресурсов в `ansible-playbook` :

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

`python inventory/dyn.py` должен распечатывать что-то вроде этого:

```
{
  "_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"
  ]
}
```

Прочитайте Динамический инвентарь онлайн: https://riptutorial.com/ru/ansible/topic/1758/динамический-инвентарь

глава 7: инвентарь

параметры

параметр	объяснение
ansible_connection	Тип подключения к хосту. Это может быть имя любого из подключаемых подключаемых модулей. Типы протоколов SSH являются `smart` , `ssh` или `paramiko` . Значение по умолчанию - умное. Типы, не относящиеся к SSH, описаны в следующем разделе.
ansible_host	Имя хоста для подключения, если оно отличается от псевдонима, который вы хотите ему передать.
ansible_port	Номер порта ssh, если не 22
ansible_user	Использовать имя пользователя ssh по умолчанию.
ansible_ssh_pass	Пароль ssh для использования (это небезопасно, мы настоятельно рекомендуем использовать `--ask-pass` или SSH)
ansible_ssh_private_key_file	Файл закрытого ключа, используемый ssh. Полезно, если вы используете несколько ключей, и вы не хотите использовать SSH-агент.
ansible_ssh_common_args	Этот параметр всегда добавляется к командной строке по умолчанию для **sftp** , **scp** и **ssh** . Полезно настраивать `ProxyCommand` для определенного хоста (или группы).
ansible_sftp_extra_args	Этот параметр всегда добавляется к командной строке **sftp** по умолчанию.
ansible_scp_extra_args	Этот параметр всегда добавляется к командной строке **scp** по умолчанию.
ansible_ssh_extra_args	Этот параметр всегда добавляется к командной строке **ssh** по умолчанию.
ansible_ssh_pipelining	Определяет, следует ли использовать конвейерную обработку SSH. Это может переопределить настройку `pipelining` в `ansible.cfg` .

параметр	объяснение
ansible_become	Эквивалент `ansible_sudo` или `ansible_su` , позволяет принудительно повысить эскалацию привилегий
ansible_become_method	Позволяет установить способ эскалации привилегий
ansible_become_user	Эквивалентен `ansible_sudo_user` или `ansible_su_user` , позволяет установить пользователя, `ansible_sudo_user` `ansible_su_user` эскалации привилегий
ansible_become_pass	Эквивалент `ansible_sudo_pass` или `ansible_su_pass` , позволяет вам установить пароль эскалации привилегий
ansible_shell_type	Тип оболочки целевой системы. Вы не должны использовать этот параметр, если только вы не установили `ansible_shell_executable` в `ansible_shell_executable` отличную от Bourne (sh). По умолчанию команды отформатированы с использованием синтаксиса `sh` style. Установка этого параметра в `csh` или `fish` заставит команды, выполняемые в целевых системах, вместо этого следовать синтаксису оболочки.
ansible_python_interpreter	Целевой путь python хоста. Это полезно для систем с более чем одним Python или не находится в **/ usr / bin / python,** таких как * BSD, или где **/ usr / bin / python** не является **питоном** серии 2.X. Мы не используем механизм **/ USR / бен** / **ENV** , как требует путь удаленного пользователя должен быть установлен правильно , а также предполагает , что **питон** исполняемый называется питон, где исполняемый файл может называться что - то вроде **python2.6.**
анзибль _ * _ переводчик	Работает для чего угодно, такого как ruby или perl, и работает так же, как `ansible_python_interpreter` . Это заменяет shebang модулей, которые будут запускаться на этом хосте.
ansible_shell_executable	Это устанавливает оболочку, которую может использовать ansible controller на целевой машине, переопределяет `executable` в `ansible.cfg` который по умолчанию имеет значение **/ bin / sh** . Вы действительно должны только изменить его, если невозможно использовать **/ bin / sh** (т.е. **/ bin / sh** не установлен на целевой машине или не может быть запущен из sudo.).

параметр	объяснение
	Новое в версии 2.1.

Examples

Инвентарь с именем пользователя и паролем

Инвентарь - это Ansible способ отслеживать все системы в вашей инфраструктуре. Вот простой файл инвентаризации, содержащий одну систему и учетные данные для входа в Ansible.

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

Инвентарь с пользовательским закрытым ключом

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

Инвентарь с пользовательским SSH-портом

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

Пропустить статическую инвентаризацию в загружаемую книгу

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

Передача динамических ресурсов в загружаемую книгу

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

Подробнее см. Динамический инвентарь .

Инвентаризация, групповые войны и вы

структура проекта (приемлемая передовая практика).

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

все начинается с inventory.development

```
[development]
dev.fakename.io

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

[api:children]
development
```

который позволяет вам ссылаться на group_vars. Удерживайте данные «специфично» в этой среде ...

```
---
app_name: NewApp_Dev
app_url: https://dev.fakename.io
app_key: f2390f23f01233f23f
```

который позволяет запускать следующий плей-лист ПРОТИВ файла инвентаризации:

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

со следующей строкой:

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

Файл хостов

Файл хоста используется для хранения подключений для загрузочных плееров Anisble. Существуют опции для определения параметров подключения:

`ansible_host` - имя хоста или IP-адрес

`ansible_port` - это порт, который машина использует для SSH

`ansible_user` - удаленный пользователь для подключения как

`ansible_ssh_pass` при использовании пароля для SSH

`ansible_ssh_private_key_file` если вам нужно использовать несколько ключей, специфичных для хостов

Это наиболее часто используемые варианты. Более подробную информацию можно найти в официальной документации Ansible .

Вот пример файла `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
```

Прочитайте инвентарь онлайн: https://riptutorial.com/ru/ansible/topic/1764/инвентарь

глава 8: Использование Ansible с OpenStack

Вступление

OpenStack - это программная платформа с открытым исходным кодом для облачных вычислений. Экземпляры Linux могут запускаться / останавливаться вручную с использованием графического веб-интерфейса или автоматизированного благодаря незащищенному облачному модулю.

Конфигурирование может быть сложным, но как только он хорошо сконфигурирован, он очень прост и эффективен для тестирования и непрерывной интеграции.

параметры

параметры	Комментарии
хосты: localhost	Команды OpenStack запускаются с нашего локального хоста
gather_facts: False	Нам не нужно собирать информацию о нашем локальном хосте
auth_url: https://openstack-identity.mycompany.com/v2.0	используйте URL-адрес V2.0
состояние: настоящее	'present' / 'absent' для создания / удаления экземпляра
validate_certs: False	полезно, если https использует самоподписанные сертификаты
network: "{{NetworkName}}"	(необязательный)
auto_ip: да	(необязательный)

замечания

- Мы помещаем URL-адрес проверки подлинности непосредственно в playbook, а не в переменную. URL, используемый в vars, должен быть экранирован.
- Будьте осторожны с версией URL-адреса аутентификации, используйте V2.0 вместо

V3 в https://openstack-identity.mycompany.com/v2.0 .
- В yml-файлах будьте очень осторожны при копировании / вставке из браузера. Проверяйте в два раза пробелы по мере их учета.
- Подробнее см .: http://docs.ansible.com/ansible/list_of_cloud_modules.html#openstack

Examples

Проверьте версию Ansible

Проверьте правильность версий программного обеспечения:

- ansible> = 2.0
- python> = 2.6
- модуль тени для python

```
$ansible --version
ansible 2.2.0.0

$python --version
Python 2.7.5
```

Установите 'shade' компонент python, используемый для запуска openstack.

```
$pip install shade
```

Примечание: если вы используете прокси-сервер компании, всегда полезно знать правильный пин синтакс

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

Соберите информацию из GUI OpenStack, чтобы настроить Ansible

---

Наш арендатор-арендатор уже установлен:

- виртуальный язык предоставляет экземпляры частного IP-адреса
- виртуальный маршрутизатор, открытый публичный IP-адрес для частного IP-адреса
- был создан ключ безопасности
- у нас есть конфигурация брандмауэра по умолчанию для ssh и порта 80
- мы можем запустить экземпляр благодаря веб-интерфейсу OpenStack

Позвольте собрать всю необходимую информацию из этого веб-интерфейса.

Информацию об аутентификации можно найти в файле openstack.rc. этот файл можно загрузить с помощью веб-интерфейса OpenStack в [access and security / API Access].

```
$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
```

Мы получаем OS_AUTH_URL, OS_TENANT_NAME, OS_USERNAME.

Версия API аутентификации: OS_AUTH_URL

Остерегайтесь версии API проверки подлинности. По умолчанию v3 активирован, но для него требуется v2.0. Мы получаем URL-адрес и устанавливаем V2.0 вместо V3: https://openstack-identity.mycompany.com/v2.0

Информация о ВМ

Создайте экземпляр, используя веб-интерфейс OpenStack, и получите имя для образа, вкуса, ключа, сети, группы безопасности.

Создайте файл ./group_vars/all со всей необходимой информацией.

```
$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
```

Записывайте воспроизводимую книгу для создания экземпляра

Пусть используется команда «os_server» из модуля «Cloud» [http://docs.ansible.com/ansible/os_server_module.html] . Переменные определены в ./ 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
```

Соберите информацию о нашем новом экземпляре

Используйте команду «os_server_facts» из модуля «Cloud» [http://docs.ansible.com/ansible/os_server_module.html] . Переменные определены в ./ group_vars/all, а имя экземпляра находится на сервере: «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
```

Это очень много. Отображается много информации. Обычно для доступа к новому экземпляру через SSH требуется только IP-адрес.

Получите новый публичный IP-адрес экземпляра

Вместо того, чтобы печатать всю информацию, мы печатаем только IP-адрес первого экземпляра, чье имя «MyOwnPersonalInstance». Это обычно все, что нам нужно.

```
$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
```

Удалить наш экземпляр

Чтобы удалить наш экземпляр, повторно используйте команду os_server со всей информацией об аутентификации и просто замените «state: present» на «state: absent».

```
$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
```

Прочитайте Использование Ansible с OpenStack онлайн:
https://riptutorial.com/ru/ansible/topic/8712/использование-ansible-с-openstack

глава 9: Использование Ansible с веб-службами Amazon

замечания

пример-2: Это служит примером, поэтому просто не копируйте / не пропустите его. Вместо этого, чтобы удовлетворить ваши потребности, вы должны настроить свои переменные; ansible_key, правила группы безопасности и т. д.

Пример-1: Для того, чтобы отключить SSH строгой проверки ключа хоста, поведение мы не хотим при автоматизации задач, мы не установить его `no` в `ansible.cfg` файле. т.е.: `StrictHostKeyChecking=no`

Файл `ec2.py` - это скрипт python, который выполняет и возвращает ваши ресурсы AWS на основе `ec2.ini` который является конфигурационным файлом, который вам нужно настроить, если вы хотите ограничить область вашего проекта в определенных регионах, определенных тегах и т. Д. ...

Examples

Как запустить экземпляр EC2 из официальных AMI Amazon, изменить его и сохранить в новом AMI

Это очень обычный рабочий процесс при использовании Ansible для создания экземпляра AWS EC2. Это сообщение предполагает базовое понимание Ansible и, самое главное, предполагает, что вы правильно настроили его для подключения к AWS.

Как утверждает настоящая официальная документация Ansible , мы собираемся использовать четыре роли:

1- **ami_find,** чтобы получить идентификатор ami, на основе которого мы запустим экземпляр EC2.

2- **ec2_ami_creation** для эффективного запуска экземпляра EC2.

3- **code_deploy** для модификации экземпляра; это может быть что угодно, поэтому мы просто переносим файл на целевую машину.

4- **build_ami** для создания нашего нового изображения на основе исполняемого экземпляра ec2. Этот пост предполагает, что вы находитесь на верхнем уровне своего проекта Ansible: `my_ansible_project`

Первая роль: **ami_find**

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

В этой роли мы собираемся использовать модуль ec2_ami_find, и в качестве примера мы будем искать машину Ubuntu и получить ее **ami_id** (ami-xxxxxxxx). Теперь отредактируйте `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 }}"
```

Вторая роль: **ec2_ami_creation**

Здесь мы будем использовать `ami_id` мы получили от первой роли, а затем `ami_id` наш новый экземпляр на основе этого:

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

В этой роли мы будем использовать, самое главное, ec2_module для запуска нашего экземпляра. Теперь отредактируйте `my_ansible_project/roles/ec2_ami_creation/tasks/main.yml` :

```
---
- 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
```

Третья роль: **code_deploy**

Здесь мы предоставим этот экземпляр, который был добавлен в группу с именем `just_created`

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

В этой роли мы будем использовать template_module для передачи файла и записи имени машины в нем. Теперь отредактируйте `my_ansible_project/roles/code_deploy/tasks/main.yml` :

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

затем перейдите в папку шаблонов внутри вашей роли:

`cd my_ansible_project/roles/templates` и добавьте файл `my_file.txt.j2` содержащий:

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

Четвертая роль: `build_ami`

Теперь мы создадим образ работающего экземпляра с помощью модуля ec2_ami . Перейдите в папку проекта и:

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

Теперь отредактируйте `my_ansible_project/roles/build_ami/tasks/main.yml` :

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

Теперь, я думаю, вам было интересно, как организовать все эти роли. Я прав? Если да, продолжайте чтение.

Мы напишем книгу, состоящую из трех пьес: первая игра, применимая на `localhost` будет называть наши первые две роли, вторая игра применима к нашей **команде just_created** .

последняя роль будет применяться на `localhost` . Почему `localhost` ? Когда мы хотим **управлять** некоторыми ресурсами AWS, мы используем нашу локальную машину так же просто. Затем мы будем использовать файл vars, в который будут помещены наши переменные: `ansible_key` , `aws_region` и т. Д.

создайте папку инфраструктуры в верхней части вашего проекта и добавьте в нее файл под названием `aws.yml` :

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

Поэтому в верхней части вашего проекта создайте `build_base_image.yml` и добавьте следующее:

```
---
   - 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 }}"}
```

Вот и все, не забудьте удалить свои ресурсы после тестирования, или почему бы не создать роль для удаления исполняемого экземпляра :-)

Как правильно настроить Ansible для подключения к Amazon Web Services

Управление ресурсами AWS, которые масштабируются вверх и вниз, попадает в рамки статического файла хоста инвентаря, поэтому нам нужно что-то динамическое. И для этого нужны динамические запасы . Давайте начнем:

Загрузите файлы ec2.ini и ec2.py в папку проекта:

```
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
```

После этого сделайте `ec2.py` файл `ec2.py` :

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

Теперь экспортируйте свой ключ AWS Secret и Access в качестве переменных окружения:

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

Для использования скрипта `ec2.py` нам нужен Python AWS SDK, `boto` поэтому вам нужно его установить:

```
sudo pip install boto
```

Чтобы проверить, все ли хорошо, попробуйте выполнить `ec2.py` , указав свои ресурсы:

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

вы должны увидеть нечто похожее на:

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

Теперь мы хотим использовать динамический инвентарь вместе с нашим статическим файлом hosts. Сначала создайте папку под названием `inventory` , добавьте `ec2.py` , `ec2.ini` и наш файл `hosts` затем скажите Ansible, чтобы использовать эту папку в качестве файла инвентаризации:

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

Затем мы должны определить конфигурацию уровня проекта для Ansible, создав файл конфигурации Ansible в папке проекта с именем `ansible.cfg` и добавив следующее:

```
[defaults]
hostfile = inventory
[ssh_connection]
```

```
pipelining = False
ssh_args = -o ControlMaster=auto -o ControlPersist=30m -o StrictHostKeyChecking=no
```

Затем нам нужно настроить Ansible для использования ключа SSH для аутентификации доступа к нашим экземплярам EC2. Использование агента SSH - лучший способ аутентификации с помощью ресурсов, поскольку это упрощает управление ключами:

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

Это оно! Если вы следовали этому, вы можете протестировать его с помощью модуля `ping` а затем вы увидите, что ваши запущенные экземпляры, которые были настроены для использования вашего ключа, отвечающего на понг:

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

Прочитайте Использование Ansible с веб-службами Amazon онлайн:
https://riptutorial.com/ru/ansible/topic/3302/использование-ansible-с-веб-службами-amazon

глава 10: Исправлена установка mysql

Вступление

Как использовать возможность установки бинарного файла mysql

Examples

Как использовать возможность установки бинарного файла mysql

- hosts: задачи mysql:
    - name: Добавить пользователя mysql user: name: mysql shell: / sbin / nologin
    - name: установить последнюю версию libselinux-python yum: имя: libselinux-python состояние: последнее
    - name: install perl yum: имя: perl состояние: последнее
    - name: удалить пакет mysql-libs yum: name: mysql-libs состояние: отсутствует

```
- 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
```

Прочитайте Исправлена установка mysql онлайн: https://riptutorial.com/ru/ansible/topic/10920/исправлена---установка-mysql

глава 11: Как создать облачный сервер DreamHost из файла Ansible Playbook

Examples

Установка библиотеки Shade

Shade - это библиотека, разработанная OpenStack для упрощения взаимодействия с облаками OpenStack, например DreamHost.

$ pip установить оттенок

Создание книги для запуска сервера

Создайте файл с именем `launch-server.yaml` , который будет нашей игрой.

Первая часть playbook - это список хостов, на которых будет работать ваша плейбук, у нас есть только один, localhost.

```
- hosts: localhost
```

Затем нам нужно определить список задач, которые будут выполняться в этой пьесе. У нас будет только тот, который запускает сервер Xenial Ubuntu на DreamCompute.

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

Следующая часть книги использует `os_server` (OpenStack Server). Это определяет, как должен выглядеть сервер в DreamCompute.

```
os_server:
```

Первый шаг - аутентификация DreamCompute; замените `{username}` своим именем DreamCompute, `{password}` с паролем DreamCompute и `{project}` с проектом DreamCompute. Вы найдете их в файле OpenStack RC .

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

Следующие строки определяют некоторые элементы нового сервера.

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

Давайте разложим предыдущие несколько строк:

- `state` - это состояние сервера, возможные значения `present` или `absent`
- `name` - имя создаваемого сервера; может быть любым значением
- `image` - это изображение для загрузки сервера; возможные значения отображаются на веб-панели DreamHost Cloud ; переменная принимает имя изображения или UUID
- `key_name` - это имя открытого ключа для добавления на сервер после его создания; это может быть любой ключ уже добавлен в DreamCompute.
- `flavor` - это вкус сервера для загрузки; это определяет, сколько оперативной памяти и процессора будет иметь ваш сервер; переменная принимает либо название аромата (gp1.semisonic), либо идентификатор (50, 100, 200 и т. д.),
- `network` - это сеть для включения вашего сервера. В случае DreamHost Cloud это `public` сеть.
- `wait` set to yes заставляет плейер ждать, пока сервер будет создан, прежде чем продолжить.

Запуск Playbook

Запустите программу воспроизведения Ansible:

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

Вы должны увидеть вывод как

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

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

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

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

Теперь, если вы проверите панель инструментов DreamHost Cloud, вы увидите новый экземпляр с именем «ansible-vm1»,

Прочитайте Как создать облачный сервер DreamHost из файла Ansible Playbook онлайн: https://riptutorial.com/ru/ansible/topic/4689/как-создать-облачный-сервер-dreamhost-из-файла-ansible-playbook

глава 12: Монтаж

Вступление

Установка Ansible в любой ОС, включая Windows с использованием Virtual Box и Vagrant. Альтернативное решение также доступно, если вы просто хотите практиковать незаменимые специальные команды и проигрыватели и не хотите настраивать локальную среду.

Examples

Установка Ansible на Ubuntu

Ansible поддерживает репозиторий PPA, который можно использовать для установки бинарных файлов Ansible:

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

Чтобы установить определенную версию, используйте `pip` . PPA может быть устаревшим.

Установка Ansible на MacOS

Существует два основных способа установки Ansible в OS X, либо с помощью диспетчера пакетов Homebrew или Pip.

Если у вас есть доморощенный, последний Ansible можно установить, используя следующую команду:

```
brew install ansible
```

Чтобы установить ветку Ansible 1.9.X, используйте следующую команду:

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

Чтобы установить ветку Ansible 2.0.X, используйте следующую команду:

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

Чтобы установить с помощью pip, используйте следующую команду: `pip install ansible` .

Чтобы установить определенную версию, используйте `pip install ansible=<required version>` .

Установка в системах на базе Red Hat

Ansible может быть установлен на CentOS или других системах на базе Red Hat. Во-первых, вы должны установить предварительные условия:

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

затем установите Ansible with pip:

```
sudo pip install ansible
```

Я могу порекомендовать вам обновить setuptools после установки:

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

Вы также можете использовать локальный диспетчер пакетов:

```
yum install ansible
```

Установка из источника

Ansible **лучше всего использовать** в кассе.

Он работает как вы (не root), и имеет минимальные зависимости python.

Настройка зависимости Python pip от pip:

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

Затем, клонируйте Ansible repo из GitHub:

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

Наконец, добавьте строку сценария инициализации с помощью строки ~ / .bashrc или ~ / .zshrc:

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

Перезагрузите сеанс терминала и проверьте

```
ansible --version
```

Установка на Amazon Linux из git repo

Amazon Linux - вариант RHEL, поэтому инструкции Red Hat должны работать по большей части. Однако существует, по крайней мере, одно несоответствие.

Был случай, когда пакет **python27-devel** , в отличие от **python-devel** , был явно необходим.

Здесь мы будем устанавливать из источника.

```
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
```

Установка Ansible для любой операционной системы (Windows) с использованием Virtual Box + Vagrant

У моего ноутбука есть Windows 10. Здесь я даю шаги, которые вы можете выполнить для тестирования и изучения Ansible.

НЕКОТОРЫЕ ТЕОРИИ

Для Ansible вам нужна машина управления и хост (или хосты) для запуска Playbook.

- **Контрольная машина** должна быть на базе Linux или MacOS (окна не разрешены), а Python (версия 2.6 или более поздняя). Здесь будет установлен Ansible.
- **Целевой компьютер** (хост / узел) может быть Linux / MacOS / windows. Для этого необходимо установить только Python. Никакое программное обеспечение агента не требуется.

НАСТРОИТЬ

Шаг 1. **Установка виртуальной коробки.**

Виртуальная коробка - это программное обеспечение для создания виртуальных компьютеров разных ОС. Это похоже на наличие нескольких компьютеров в каждой или разных ОС и разных версий.

Загрузите Virtual Box в соответствии с ОС в вашей системе и установите его.

Шаг 2: **Установите бродягу**

Vagrant - это интерфейс командной строки для создания виртуальных машин в виртуальной коробке. Это облегчает задачу. Вам нужно изучить основные команды бродяг.

Шаг 3. **Создайте папку, в которой вы хотите, чтобы ваша виртуальная машина**

Шаг 4. **Создание виртуальной машины с использованием бродяг**

Откройте терминал и перейдите на путь, где была создана папка, и выполните следующие две команды.

Вам нужно выбрать **Virtual Box** . Например, я устанавливаю Ubuntu. Вы можете выбрать что-нибудь из списка. Вам нужно запустить эти две команды под категорией « **виртуальный блок** »: `vagrant init ubuntu/trusty64` и `vagrant up --provider virtualbox` . Другие категории могут быть: hyperv, vmware_desktop и т. Д. (Это займет некоторое время, так как оно загрузит необходимые файлы)

Шаг 4: Установите Ansible

Для UbuntuOS: `sudo apt-get install ansible`

Альтернативное решение :

Вы можете использовать **Katacoda** для практического применения. Не нужно ничего устанавливать или настраивать. Запустите две команды, приведенные в шаге 2, и после этого вы хорошо пойдете.

Прочитайте Монтаж онлайн: https://riptutorial.com/ru/ansible/topic/4906/монтаж

глава 13: Несоблюдение: петли и условные обозначения

замечания

Официальные документы объясняют условные обозначения.

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

Нескончаемый (github)

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

Examples

Какие условные обозначения использовать?

Использование условных выражений через (синтаксис находится в `[brackets]`):

- когда [**когда:**]

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

- петли [**with_items:**]
- циклы [**with_dicts:**]
- Пользовательские факты [**когда:** my_custom_facts == '1234']
- Условный импорт
- Выбор файлов и шаблонов на основе переменных

[Когда] Условие: `ansible_os_family` Списки

Общего пользования

- когда: ansible_os_family == "CentOS"
- когда: ansible_os_family == "Редхат"
- когда: ansible_os_family == "Дарвин"

- когда: ansible_os_family == "Debian"
- когда: ansible_os_family == "Windows"

Все списки

на основе обсуждения здесь 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'
        )
```

Когда условие

Основное использование

Используйте условие when, чтобы контролировать, выполняется или не выполняется задача или роль. Обычно это используется для изменения игрового поведения на основе фактов из системы назначения. Рассмотрим этот учебник:

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

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

Где `Ubuntu.yml` и `RHEL.yml` включают некоторую логику, специфичную для дистрибутива.

Другим распространенным применением является ограничение результатов для тех, кто находится в определенных группах инвентаризации Ansible. Рассмотрим этот файл инвентаризации:

```
[dbs]
mydb01

[webservers]
myweb01
```

И этот плейбук:

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

Это использует магическую переменную `group_names` .

Условный синтаксис и логика

Единственное условие

Синтаксис

```
when: (condition)
```

пример

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

Булевский фильтр

пример

```
when: result|failed
```

Несколько условий

Синтаксис

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

Пример (простой)

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

Пример (сложный)

Для обеспечения четкости или для управления приоритетом используйте круглые скобки. «И» имеет более высокий приоритет, чем «ИЛИ».

Классы могут охватывать линии:

```
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', '>=')
```

Обратите внимание на использование круглых скобок для группировки «или» в первой проверке распределения.

Получите `ansible_os_family` и` ansible_pkg_mgr` с настройкой

Мы можем получить факты (`ansible_os_family` , `ansible_pkg_mgr`) с командой Ad-Hoc модуля настройки и фильтра.

- 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
}
```

Простые «Когда» Пример (ы)

Дано:

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

Затем эти задачи всегда выполняются.

```
- 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
```

Эта задача никогда не будет выполняться.

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

Использование до повторной проверки цикла

Это пример использования до / retries / delay для реализации живой проверки для запуска webapp. Он предполагает, что будет некоторый период времени (до 3 минут), когда webapp отказывается от соединений сокетов. После этого он проверяет / живую страницу для слова «OK». Он также делегирует поиск URL-адреса для запуска localhost. Это имеет смысл в качестве конечной задачи в учебнике для развертывания.

```
---
- 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
```

Пока шаблон повторения не может использоваться с любым действием; Несвязанная

документация обеспечивает пример ожидания, пока некоторая команда оболочки не вернет желаемый результат: http://docs.ansible.com/ansible/playbooks_loops.html#do-until-loops .

Прочитайте Несоблюдение: петли и условные обозначения онлайн: https://riptutorial.com/ru/ansible/topic/3555/несоблюдение--петли-и-условные-обозначения

глава 14: Несоблюдение: Цикл

Examples

with_items - простой список

Цикл `with_items` in ansible может использоваться для простого `with_items` значений.

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

with_items - предопределенный список

Вы также можете перебирать список переменных.

Из vars:

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

а затем цикл:

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

Если вы используете Ansible 2.0+, вы должны использовать кавычки вокруг вызова переменной.

with_items - предопределенный словарь

Можно создавать более сложные циклы со словарями.

Из vars:

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

то цикл:

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

Или, если вам не нравится использовать значение ключа:

вары:

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

то цикл:

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

with_items - словарь

Вы можете использовать словарь для немного более сложного цикла.

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

Вложенные петли

Вы можете создавать вложенные циклы, используя `with_nested` .

от варсов:

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

то цикл:

```
- 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 }}'
```

Эта задача будет зацикливаться на каждом пользователе и заполнить файл `authorized_keys` четырьмя ключами, определенными в списке.

Прочитайте Несоблюдение: Цикл онлайн: https://riptutorial.com/ru/ansible/topic/6414/несоблюдение--цикл

глава 15: Несчастные группы Vars

Examples

Пример group_vars / development и почему

Структура проекта

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

Эти переменные будут применяться к хостам в группе разработки из-за имени файла.

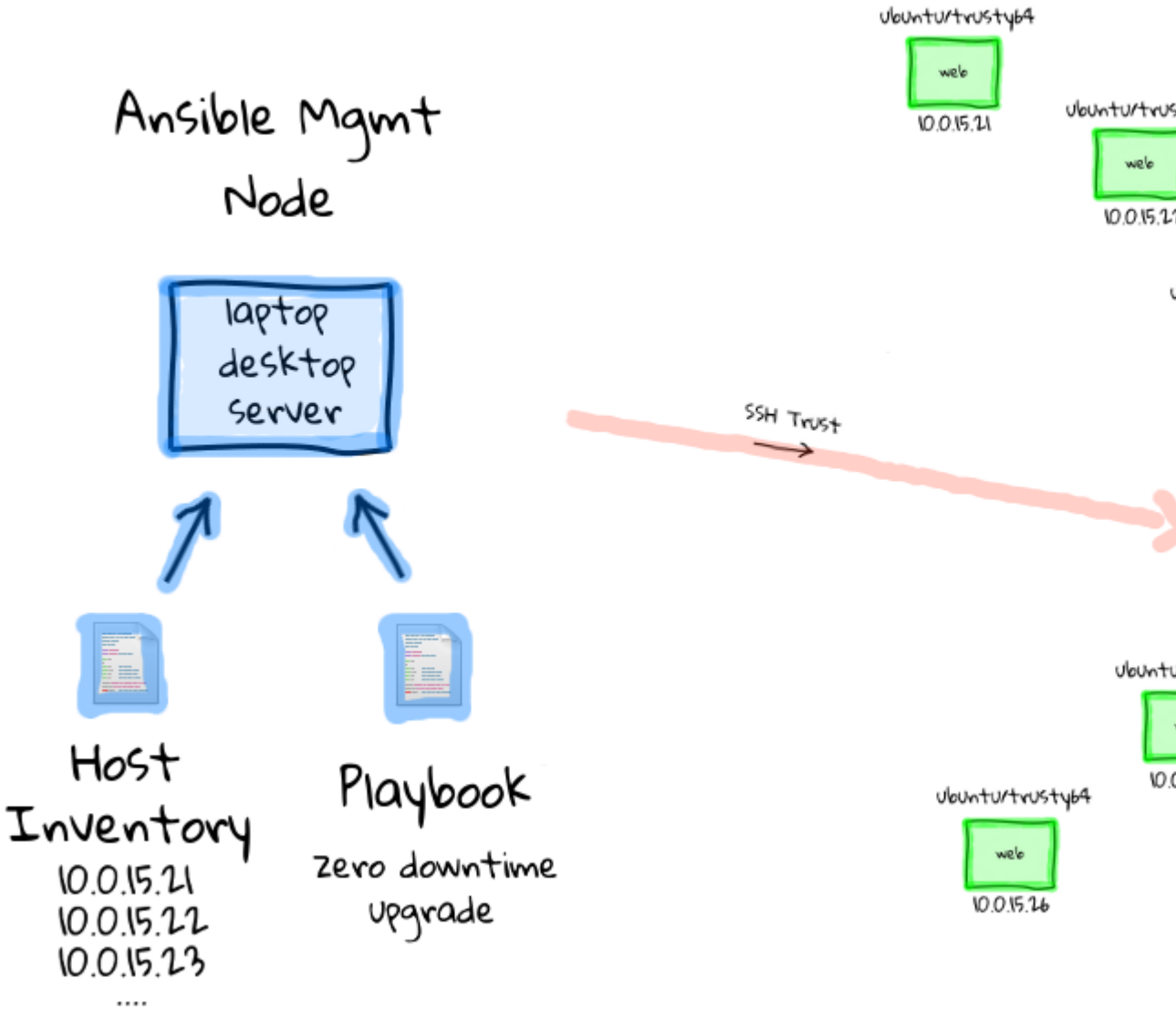
```
---
## Application
app_name: app
app_url: app.io
web_url: cdn.io
app_friendly: New App
env_type: production
app_debug: false

## 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
```

Прочитайте Несчастные группы Vars онлайн: https://riptutorial.com/ru/ansible/topic/6226/несчастные-группы-vars

глава 16: Прозрачная архитектура

Examples

Понимание прочной архитектуры



Идея состоит в том, чтобы иметь одну или несколько управляющих машин, из которых вы можете отправлять специальные команды на удаленные компьютеры (через `ansible` инструмент) или запускать последовательность инструкций, устанавливаемую с помощью проигрывателей (с помощью инструмента для `ansible-playbook`).

В принципе, мы используем Ansible control machine, обычно это ваш рабочий стол, ноутбук или сервер. Затем оттуда вы можете использовать Ansible для изменения конфигурации, используя ssh.

Файл инвентаризации хоста определяет целевые компьютеры, в которых будут выполняться эти игры. Файл конфигурации Ansible можно настроить, чтобы отображать настройки в вашей среде.

Прочитайте Прозрачная архитектура онлайн: https://riptutorial.com/ru/ansible/topic/7659/прозрачная-архитектура

глава 17: Роли

Examples

Использование ролей

Ansible использует концепцию ролей, чтобы лучше допускать модульный код и избегать повторения.

Роль - это просто структура папок, которая Ansible знает, где загрузить файлы, задачи и обработчики vars. Пример может выглядеть примерно так:

```
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
```

Затем вы можете использовать эту роль в основной игровой книге, которая выглядит следующим образом:

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

Когда вы запускаете Ansible против этой пьесы, он будет нацеливаться на всех хостов в группе `webservers` и запускать `apache` выше роль `apache` , автоматически загружая любые переменные по умолчанию для этой роли и запуская все задачи, включенные в `tasks/main.yml` . Ansible также знает, как искать определенные типы файлов в дружественных ролях местах:

- Если существуют роли / x / tasks / main.yml, в игру будут добавлены перечисленные в них задачи
- Если существуют роли / x / handlers / main.yml, в игру будут добавлены обработчики,

перечисленные в нем.

- Если существуют роли / x / vars / main.yml, в игру будут добавлены переменные, перечисленные в ней
- Если существуют роли / x / meta / main.yml, любые перечисленные в нем зависимые роли будут добавлены в список ролей (1.3 и более поздних)
- Любая копия, сценарий, шаблон или включенные задачи (в роли) могут ссылаться на файлы в ролях / x / {файлах, шаблонах, задачах} / (dir зависит от задачи), не требуя их относительно или абсолютно

Ролевые зависимости

Роли также позволяют вам определять другие роли как зависимость, создавая файл `meta/main.yml` с блоком `dependencies` :

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

Также возможно передать значение параметру / переменной в зависимой роли:

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

Или даже выполнить зависимую роль условно:

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

Зависимые роли всегда выполняются перед ролями, которые зависят от них. Кроме того, они выполняются только один раз. Если две роли заявляют то же, что и их зависимость, она выполняется только в первый раз.

Представьте себе роли role1, role2 и role3 с текстом `meta/main.yml` :
role1 / мета / main.yml:

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

роль2 / мета / main.yml:

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

При выполнении role1 и role2 в одной и той же playbook (с ролью1, вызываемой до role2), порядок выполнения будет следующим:

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

Вы можете переопределить это поведение, указав `allow_duplicates: yes` в `meta/main.yml` роли1 и role2. Результирующий порядок выполнения будет следующим:

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

Разделение конкретных задач и переменных распределения внутри роли

Мы можем легко разделить конкретные задачи и переменные дистрибутива на разные выделенные файлы .yml. Ansible помогает нам автоматически идентифицировать распределение целевых хостов через `{{ ansible_distribution }}` и `{{ ansible_distribution_version }}` , поэтому мы просто должны назвать соответствующие дистрибутивы .yml-файлам соответственно.

Для Ubuntu Xenial основная роль dir tree будет выглядеть примерно так:

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

Внутри `tasks/main.yml` теперь мы можем автоматически включать правильные переменные и задачи для распределения целевых хостов.

Задачи / 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"
```

Внутри `tasks/Ubuntu16.06.yml` и `vars/Ubuntu16.04.yml` теперь мы можем определить задачи и переменные для Ubuntu Xenial соответственно.

Прочитайте Роли онлайн: https://riptutorial.com/ru/ansible/topic/3396/роли

глава 18: Секретное шифрование

замечания

Ansible предлагает Vault (чтобы не ошибиться с HashiCorp Vault !) Для обработки конфиденциального шифрования данных. Сейф в первую очередь предназначен для шифрования любых структурированных данных, таких как переменные, задачи, обработчики.

Examples

Шифрование конфиденциальных структурированных данных

Сначала создайте файл ключа, например, `vault_pass_file` , который идеально содержит длинную последовательность случайных символов. В системах linux вы можете использовать `pwgen` для создания файла случайного пароля:

```
pwgen 256 1 > vault_pass_file
```

Затем используйте этот файл для шифрования конфиденциальных данных, например `groups_vars/group.yml` :

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

С этого `vault_pass_file` для запуска учебника вам нужен `vault_pass_file` :

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

Обратите внимание: вы также можете использовать `--vault-password-file vault_pass_file` вместо установки переменной среды `ANSIBLE_VAULT_PASSWORD_FILE` .

Для того , чтобы изменить или расшифровать секрет на диске , вы можете использовать `ansible-vault edit` и `ansible-vault decrypt` соответственно.

Использование поисковых каналов для дешифрования неструктурированных данных, хранящихся в хранилищах

С помощью Vault вы также можете шифровать неструктурированные данные, такие как файлы секретных ключей, и все еще иметь возможность расшифровывать их в своей игре с помощью модуля `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')
```

Использование local_action для дешифрования зашифрованных зашифрованных шаблонов

Вы можете запустить игру, которая полагается на зашифрованные шаблонами шаблоны, используя модуль `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
```

Обратите внимание, что `changed_when: False` . Это важно, если вы запускаете тесты idempotence с вашими незаменимыми ролями, иначе каждый раз, когда вы запускаете playbook, выдается сообщение об изменении. В `group_vars/all.yml` вы можете установить глобальную команду decrypt для повторного использования, например, как `view_encrypted_file_cmd` .

group_vars / all.yml

```
---

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

Теперь при запуске воспроизведения вам необходимо установить `ANSIBLE_VAULT_PASSWORD_FILE` среды `ANSIBLE_VAULT_PASSWORD_FILE` чтобы указать на ваш файл паролей хранилища (в идеале с абсолютным путем).

Прочитайте Секретное шифрование онлайн: https://riptutorial.com/ru/ansible/topic/3355/секретное-шифрование

глава 19: Сильные групповые переменные

Examples

Групповые переменные со статическими ресурсами

Предполагается, что вы определяете группы в зависимости от назначения хоста (ролей), а также географии или местоположения центра данных (если применимо):

Файловый `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
```

Файл `group_vars/all`

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

Файл `group_vars/atlanta-webservers`

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

Файл `group_vars/boston-webservers`

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

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

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

После запуска исполняемого `ansible-playbook -i inventory/hosts install-apache.yml` (хосты в playbook будут `hosts: all`)

Порты будут

Адрес	порт
`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

Прочитайте Сильные групповые переменные онлайн:
https://riptutorial.com/ru/ansible/topic/6544/сильные-групповые-переменные

глава 20: Станьте (Привилегированная эскалация)

Вступление

Часто вам нужно выполнять команды под другим пользователем или получать привилегии *root* . Эти параметры позволяют вам **стать** другим пользователем в гостевой системе.

Синтаксис

- `become` : может быть установлено значение true или yes и запускает настройки эскалации пользователя.
- `become_user` : установить желаемый пользователь на удаленном хосте.
- `become_method` : указать команду, используемую для входа в систему и изменения пользователя.
- `become_flags` : изменить параметры входа. В основном используется, когда вы хотите изменить системного пользователя без привилегий оболочки.

Examples

Только в задаче

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

Запуск всех ролевых задач как root

```
- hosts: all
  become: true

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

Запуск роли root

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

Прочитайте Станьте (Привилегированная эскалация) онлайн:
https://riptutorial.com/ru/ansible/topic/8328/станьте--привилегированная-эскалация-

кредиты

S. No	Главы	Contributors
1	Начало работы с помощью	activatedgeek, Alex, baptistemm, calvinmclean, Community, Jake Amey, jasonz, jscott, Michael Duffy, mrtuovinen, Pants, PumpkinSeed, tedder42, thisguy123, ydaetskcoR
2	Loops	marx, mrtuovinen
3	Введение в playbooks	32cupo, Abdelaziz Dabebi, ydaetskcoR
4	галактика	mrtuovinen, ydaetskcoR
5	Динамический инвентарь	mrtuovinen
6	инвентарь	calvinmclean, mrtuovinen, Nick
7	Использование Ansible с OpenStack	BANANENMANNFRAU, Sebastien Josset
8	Использование Ansible с веб-службами Amazon	Abdelaziz Dabebi, another geek, ydaetskcoR
9	Исправлена установка mysql	Fernando
10	Как создать облачный сервер DreamHost из файла Ansible Playbook	Stefano Maffulli
11	Монтаж	ca2longoria, Jake Amey, Michael Duffy, mrtuovinen, Nick, PumpkinSeed, Raj, tedder42, ydaetskcoR
12	Несоблюдение: петли и условные обозначения	A K, Chu-Siang Lai, Jordan Anderson, marx, Mike, mrtuovinen, Nick, Rob H, wolfaviators
13	Несоблюдение: Цикл	calvinmclean

14	Несчастные группы Vars	Nick, Peter Mortensen
15	Прозрачная архитектура	Jordan Anderson, Yogesh Darji
16	Роли	Chu-Siang Lai, fishi, mrtuovinen, winston, ydaetskcoR
17	Секретное шифрование	fishi
18	Сильные групповые переменные	mrtuovinen
19	Станьте (Привилегированная эскалация)	Jordan Anderson, Willian Paixao