Введение в администрирование Linux: Ansible

Семинар #2 “Ansible: Написание сценариев”

Пантелеев Александр

Введение в администрирование Linux: Ansible

План занятия

- Факты и переменные
- Фильтры
- Циклы
- Условные выражения
- Шифрование секретов

Вопросы просьба задавать в конце подтемы.

Ссылка на материалы для занятия

ТЕНЗОР

Факты и переменные

Когда Ansible выполняет сценарий, до запуска первой задачи происходит сбор информации о сервере.

Эта информация сохраняется в переменных, называемых фактами. Обращение к фактам работает как обращение к любым другим переменным.

$ ansible *groupname* -m setup | tee nodefacts.txt

```
---
- name: get facts playbook
  hosts: nodes
  gather_facts: true
  tasks:
  - name: output all facts
    debug:
      msg: "{{ ansible_facts }}"

  - name: print OS distribution fact
    debug:
      msg: "{{ ansible_facts.distribution }}"
```

Ссылка на документацию

ТЕНЗОР

Локальные факты

Ansible позволяет ассоциировать факты с конечным хостом. Ansible обнаружит файлы с фактами в директории /etc/ansible/facts.d если они отвечают одному из следующих требований:

- Имя файла заканчивается на .fact
- Имеют формат .ini
- Имеют формат JSON/YAML
- Являются Executable файлами, не принимающими аргументов и возвращающими результат в формате JSON/YAML

Эти факты доступны в виде ключей особой переменной ansible_local.

```yaml
---
- name: get facts playbook
  hosts: nodes
  gather_facts: true
  tasks:
  - name: ensure that facts.d directory exists
    file:
      path: /etc/ansible/facts.d
      state: directory
      owner: ansible
      group: ansible
      mode: 0755
    become: true

  - name: ensure that local facts exist
    copy:
      src: ./02b_local_facts.yaml
      dest: /etc/ansible/facts.d/ansible_course.fact
      owner: ansible
      group: ansible
      mode: 0644
    become: true

  - name: print ansible local
    debug:
      msg: "{{ ansible_local }}"

  - name: print var2
    debug:
      msg: "{{ ansible_local.ansible_course.grouped_facts1.var2 }}"

```

```ini
[grouped_facts1]
var1: value1
var2: value2
var3: value3

[grouped_facts2]
var4: value4
var5: value5
var6: value6
```

ТЕНЗОР

Кэширование фактов

Если кэширование фактов включено, Ansible сохранит факты в кэше, полученные после первого подключения к хостам. В последующих попытках выполнить сценарий Ansible будет извлекать факты из кэша, не обращаясь к удаленным хостам. Такое положение вещей сохраняется до истечения времени хранения кэша.

```
[defaults]
gathering = smart
# кэш остается действительным 24 часа
fact_caching_timeout = 86400
# обязательно указать реализацию кэширования фактов
fact_caching = jsonfile
fact_caching_connection = /tmp/ansible_fact_cache
```

ТЕНЗОР

Модуль set_fact

Ansible позволяет устанавливать факты (по сути, создавать новые переменные) в задачах с помощью модуля set_fact.

```yaml
---
- name: set facts playbook
  hosts: nodes
  gather_facts: false
  tasks:
  - name: set facts task
    set_fact:
      fact1: "value1"

  - name: output fact1
    debug:
      msg: "fact1 - {{ fact1 }}"
```

Ссылка на документацию

Модуль register

Опция register позволяет зарегистрировать в отдельную словарь информацию о выполненной задаче.
У каждого модуля разные ключи словаря.
Важное замечание: Обязательно проверьте ситуации, когда состояние хоста изменяется и когда оно не изменяется. В противном случае сценарий завершится с ошибкой, попытавшись обратиться к отсутствующему ключу, зарегистрированной переменной.

```yaml
---
- name: show return value of whoami command
  hosts: nodes
  tasks:
    - name: register output of whoami command
      command: whoami
      register: login

    - name: debug
      debug:
        msg: "{{ login }}"

    - name: debug
      debug:
        msg: "Current session user is {{ login.stdout }}"

```

[Ссылка на документацию]

ТЕНЗОР

Встроенные переменные

Параметр	Описание
hostvars	Словарь, ключи которого — имена хостов Ansible, а значения — словари, отображающие имена переменных в их значения
inventory_hostname	FQDN текущего хоста, как оно задано в Ansible
inventory_hostname_short	Имя текущего хоста, как оно задано в Ansible, без имени домена
group_names	Список всех групп, в которые входит текущий хост
groups	Словарь, ключи которого — имена групп в Ansible, а значения — списки имен хостов, входящих в группы.
ansible_check_mode	Логическая переменная, принимающая истинное значение, когда сценарий выполняется в тестовом режиме
ansible_play_batch	Список имен хостов из реестра, активных в текущем пакете
ansible_play_hosts	Список имен хостов из реестра, участвующих в текущей операции
ansible_version	Словарь с информацией о версии Ansible

Ссылка на документацию

Hostvars и inventory_hostname

inventory_hostname – это имя текущего хоста, как оно задано в реестре Ansible. Если мы определяли псевдоним для хоста, тогда это – псевдоним.

С помощью переменной hostvars и inventory_hostname можно вывести все переменные, связанные с текущим хостом, следующим образом:

```yaml
- name: get facts playbook
  hosts: nodes
  gather_facts: true
  tasks:
  - name: list all nodes participating in play
    debug:
      msg: "{{ inventory_hostname }}"

  - name: get facts by hostvars
    debug:
      msg: "{{ hostvars[inventory_hostname].ansible_default_ipv4.address }}"

```

Установка переменных из командной строки

Переменные, установленные передачей параметра команде ansible-playbook, имеют наивысший приоритет и могут заменять ранее определенные переменные

$ ansible-playbook variable_override.yml -e 'greeting_msg="Привет, друг"‘

Вместо отдельных переменных мы можем передать в Ansible файл с переменными, указав имя файла

$ ansible-playbook variable_override.yml -e @06b_file.yaml

```yaml
- name: pass the message to command line
  hosts: nodes
  gather_facts: true
  vars:
    greeting_msg: "greeting message is not specified"
  tasks:
  - name: message output
    debug:
      msg: "{{ greeting_msg }}"

```

ТЕНЗОР

Приоритеты переменных

- role/defaults — самый низкий приоритет. Идеальное место для размещения переменных, которые надо переопределять.
- group_vars/all — переменные группы all наименее приоритетные
- group_vars/other_groups
- host_vars
- Переменные, собираемые через gather_facts: true (host_facts)
- Переменные уровня play
- Переменные уровня block
- Переменные уровня task
- set_fact/register.
- Переменные передаваемые через -e — самый высокий приоритет

Ссылка на документацию

Пост на хабре про переменные Ansible

ТЕНЗОР

Фильтры

- Позволяют обрабатывать значения переменных, чтобы извлечь информацию, преобразовать ее или использовать для вычисления нового значения.
- Могут потребовать использование дополнительных аргументов или опций в круглых скобках
- Возможность создать цепочку из нескольких фильтров в одном выражении

Важно: Фильтры не изменяют значение, хранящееся в переменной. Выражение Jinja2 обрабатывает это значение и использует результат, не изменяя саму переменную

```yaml
---
- name: common filters
  hosts: localhost
  connection: local
  gather_facts: false
  vars:
    my_value: 5
    my_value2: Null
    my_default: 10
  tasks:
    - name: mandatory filter
      debug:
        msg: >-
          'my_value | mandatory' result is
          {{ my_value | mandatory }}

    - name: default filter
      debug:
        msg: >-
          'my_value2 | default(my_default)' result is
          {{ my_value2 | default(my_default) }}

    - name: default filter with null override
      debug:
        msg: >-
          'my_value2 | default(my_default, True)' result is
          {{ my_value2 | default(my_default, True) }}
```

[Ссылка на документацию]

ТЕНЗОР

Фильтры управления списками

- max, min и sum
- length, first, last
- sort, reverse
- flatten

```yaml
---
- name: list filters
  hosts: localhost
  connection: local
  gather_facts: false
  tasks:
    - name: max, min and sum filters
      debug:
        msg:
          - "'[ 2, 4, 6, 8, 10, 12 ] | max' result is
            {{ [ 2, 4, 6, 8, 10, 12 ] | max }}"
          - "'[ 2, 4, 6, 8, 10, 12 ] | min' result is
            {{ [ 2, 4, 6, 8, 10, 12 ] | min }}"
          - "'[ 2, 4, 6, 8, 10, 12 ] | sum' result is
            {{ [ 2, 4, 6, 8, 10, 12 ] | sum }}"

    - name: length, first and last filters
      debug:
        msg:
          - "'[ 2, 4, 6, 8, 10, 12 ] | length' result is
            {{ [ 2, 4, 6, 8, 10, 12 ] | length }}"
          - "'[ 2, 4, 6, 8, 10, 12 ] | first' result is
            {{ [ 2, 4, 6, 8, 10, 12 ] | first }}"
          - "'[ 2, 4, 6, 8, 10, 12 ] | last' result is
            {{ [ 2, 4, 6, 8, 10, 12 ] | last }}"
```

```yaml
- name: reverse and sort filters
  debug:
    msg:
      - "'[ 2, 4, 6, 8, 10 ] | reverse' result is
        {{ [ 2, 4, 6, 8, 10 ] | reverse }}"
      - "'[ 4, 8, 10, 6, 2 ] | sort' result is
        {{ [ 4, 8, 10, 6, 2 ] | sort }}"

- name: flatten filter
  debug:
    msg:
      - "'[ 2, [4, [6, 8]], 10 ] | flatten' result is
        {{ [ 2, [4, [6, 8]], 10 ] | flatten }}"
```

ТЕНЗОР

Фильтры управления словарями

- Объединение словарей
- Преобразование словарей

```yaml
---
- name: dict filters
  hosts: localhost
  connection: local
  gather_facts: false
  vars:
    characters_dict:
      Douglas: Human
      Marvin: Robot
      Arthur: Human
    characters_items:
      - key: Douglas
        value: Human
      - key: Marvin
        value: Robot
      - key: Arthur
        value: Human
  tasks:
    - name: print characters_dict
      debug:
        msg: "{{ characters_dict }}"

    - name: characters_items
      debug:
        msg: "{{ characters_items }}"

    - name: dict2items and items2dict filters
      debug:
        msg:
          - "'characters_dict | dict2items' result is
            {{ characters_dict | dict2items }}"
          - "'characters_items | items2dict' result is
            {{ characters_items | items2dict }}"

    - name: dict2items and items2dict filters
      assert:
        that:
          - "{{ characters_dict | dict2items }} is eq( characters_items )"
          - "{{ characters_items | items2dict }} is eq( characters_dict )"
```

```yaml
  tasks:
    - name: combine filter
      debug:
        msg: "{{ {'A':1, 'B':2} | combine({'B':4, 'C':5}) }}"
```

ТЕНЗОР

Фильтры хэширования и кодирования

- Хеширование строк и паролей
- Кодирование строк

```yaml
---
- name: other filters
  hosts: localhost
  connection: local
  gather_facts: false
  vars:
    my_string: some string with spaces
  tasks:
    - name: SHA1 hash
      debug:
        msg: "'password' in SHA1 -
          {{ 'password' | hash('sha1') }}"

    - name: SHA512 hash
      debug:
        msg: "'password' in SHA512 -
          {{ 'password' | password_hash('sha512') }}"

    - name: base64encode filters
      debug:
        msg: "âÉïôú encoded -
          '{{ 'âÉïôú' | b64encode }}'"

    - name: base64decode filter
      debug:
        msg: "w6LDicOvw7TDug== decoded -
          '{{ 'w6LDicOvw7TDug==' | b64decode }}'"

```

ТЕНЗОР

Фильтры обработки строк

- Обернуть в кавычки
- Управление регистром

```
28
29      - name: quote filter
30        debug:
31          msg:
32            - "{{ my_string }}"
33            - "{{ my_string | quote }}"
34
35      - name: Change case of characters
36        debug:
37          msg:
38            - "'Test | lower' result is {{ 'Test' | lower }}"
39            - "'Test | upper' result is {{ 'Test' | upper }}"
40            - "'test | capitalize' result is {{ 'test' | capitalize }}'"
41
```

Циклы

loop
~~with_<lookup>(list, items, dict и т.д.)~~
until

[Ссылка на документацию]

ТЕНЗОР

Циклы по спискам, loop_control

Задача будет выполняться пять раз, что равно количеству элементов в простом списке.

Loop_control:

- pause
- index_var
- loop_var
- extended

```yaml
---
- name: print list
  hosts: localhost
  connection: local
  gather_facts: false
  vars:
    prime: [2, 3, 5, 7, 11]
  tasks:
    - name: Show five first prime numbers
      debug:
        msg: "{{ prime_number }}"
      loop: "{{ prime }}"
      loop_control:
        loop_var: prime_number

```

Ссылка на документацию по loop_control

Циклы по словарям

Изначально loop работает только со списками. Ansible выдаст ошибку, если попытаться перебрать значения словаря. Для вывода значения словаря необходимо пропустить переменную цикла через фильтр dict2items, после чего Ansible выдаст нам каждую пару ключ/значение из словаря

```yaml
---
- name: Print Dictionary
  hosts: localhost
  connection: local
  gather_facts: false
  vars:
    employee:
      name: "Elliot Alderson"
      title: "Cybersecurity engineer"
      company: "Allsafe Cybersecurity"
  tasks:
    - name: Print employee dictionary
      debug:
        msg: "{{ employee_info }}"
      loop: "{{ employee | dict2items }}"
      loop_control:
        loop_var: employee_info

```

Циклы со сложными списками

Благодаря поддержке jinja2, мы можем производить проход по сложным спискам
В примере мы получаем декартово произведение по всем элементам и выводим его.

```yaml
---
- name: Jinja magic in loop
  hosts: localhost
  connection: local
  gather_facts: false
  tasks:
    - name: Cartesian product example
      debug:
        msg: "{{ cartesian[0] + cartesian[1] }}"
      loop: "{{ ['abc', 'def'] | product(['123', '456', '789']) | list }}"
      loop_control:
        loop_var: cartesian

```

Условные выражения

Условные выражения применяются при помощи ключевого слова when

- 1 задача выполняется
- 2 задача пропускается
- 3 задача выполняется, условия объединяются как при операторе and

```yaml
---
- name: test
  hosts: localhost
  connection: local
  gather_facts: true
  tasks:
    - name: show ansible version success
      debug:
        msg: "ansible version is {{ ansible_version }}"
      when: ansible_version.full == '2.9.14'

    - name: show ansible version skipped
      debug:
        msg: "ansible version is {{ ansible_version }}"
      when: ansible_version.full == '2.9.15'

    # комбинированные выражения
    - name: show ansible version mixed condition
      debug:
        msg: >-
          os distribution is {{ ansible_facts.distribution }}
          {{ ansible_facts['distribution_major_version'] }}
      when:
        - ansible_facts.distribution | lower == 'centos'
        - ansible_facts['distribution_major_version'] == "7"
```

[Ссылка на документацию]

Условные выражения в циклах

Ansible предоставляет возможность комбинировать условные выражения с циклами. В таком случае условное выражение будет проверяться для каждого элемента цикла отдельно

```yaml
---
- name: loops with conditionals
  hosts: localhost
  connection: local
  gather_facts: false
  tasks:
    - name: return items > 5
      debug:
        msg: "{{ even_number }}"
      loop: [ 0, 2, 4, 6, 8, 10 ]
      loop_control:
        loop_var: even_number
      when: even_number > 5
```

changed_when

Условное выражение changed_when позволяет самостоятельно определить, при каких условиях задача должна иметь состояние changed.

```yaml
---
- name: changed_when example
  hosts: localhost
  connection: local
  gather_facts: false
  become: true
  tasks:
    - name: changed_when test
      shell: echo "Sample text"

    - name: changed_when test
      shell: echo "Sample text"
      changed_when: false

```

failed_when

Условное выражение failed_when позволяет самостоятельно определить, при каких условиях сценарий должен завершиться с ошибкой.

```yaml
---
- name: failed_when play
  hosts: localhost
  connection: local
  gather_facts: false
  tasks:
  - name: run non-existing program
    command: /opt/myprogramm
    register: result
    ignore_errors: true

  - name: show registered variable
    debug:
      msg: "{{ result }}"

  - name: stop task if program failed
    debug:
      msg: "Stop running if myprogramm failed"
    failed_when: result is failed

  - name: unreachable task
    debug:
      msg: "this message never shows up"

```

Шифрование секретов (ansible-vault)

Утилита командной строки ansible-vault позволяет создавать и редактировать зашифрованный файл, который ansible-playbook будет автоматически распознавать и расшифровывать с помощью пароля.

```
1  password: HESOYAM
```

Зашифровать и расшифровать уже существующий файл.

- $ ansible-vault encrypt secrets.yaml
- $ ansible-vault decrypt secrets.yaml

```
1  $ANSIBLE_VAULT;1.1;AES256
2  30313036316634346436306162303065373561666430343538623561396530656134306265373366
3  643334643438393436663139633430626263666131636320a643764333065636233346433623533
4  356263303562636265383666663833343764393666613635646339646337343138393364306361 34
5  623036323830616165 0a303232353430613761636534333437326630623830373666333533373434
6  663638336461633365353935313438653338373434396363633931646362623561 38
7
```

Создать новый зашифрованный файл. Откроется текстовый редактор из переменной окружения $EDITOR

- $ ansible-vault create secrets2.yml

К файлу, зашифрованному через ansible-vault можно обращаться в секции vars_files.

```
1  ---
2  - name: get facts playbook
3    hosts: localhost
4    connection: local
5    gather_facts: false
6    vars_files:
7      - 01b_secret_encrypted.yaml
8    tasks:
9      - name: get facts task
10       debug:
11         msg: "{{ password }}"
12
```

При обращении к зашифрованному файлу нужно указать параметр --ask-vault-pass.

- $ ansible-playbook playbook.yaml --ask-vault-pass

[Ссылка на документацию]

ТЕНЗОР

Домашнее задание

1 задача: Добавить 4 пользователей на удаленный хост, используя loop. Дополнительно переопределить имя loop-переменной из item в user, назначить паузу в 5 секунд между добавлениями пользователей. К каждому пользователю добавить комментарий «Ansible-generated user». Учетные данные (логин и пароль) вынести во внешний файл, зашифровать его через Ansible-vault.

Если Ansible будет выдавать предупреждения в ходе выполнения задач — найти способ исправить ошибку

В сообщении с ДЗ жду сам playbook, зашифрованный файл с учетными данными, текстовый файл, в котором будет лежать ключ от зашифрованного файла и команда, которой вы запускаете плейбук.

Домашнее задание

2 задача: Требуется дописать сценарий установки БД MariaDB на удаленном сервере

В репозитории с материалами для лекции находится директория HW, с проектом db_init. В файле main.yaml описан порядок задач и указаны названия модулей, которые потребуется использовать, но которые мы не разбирали на занятиях.

Необходимые переменные находятся в директории vars.

ТЕНЗОР

Научились управлять фактами и переменными, фильтрами, циклами и условными выражениями. Вопросы?