Лабораторная работа №5

Управление конфигурациями хостов с помощью Ansible

Теоретическая часть

Конфигурация множества хостов – одна из важных задач системного администрирования. Предположим, перед системным администратором стоит задача установить на все пользовательские машины организации web сервер и сконфигурировать его. Наиболее очевидное решение – подключиться к каждому из компьютеров и вручную проделать все необходимые действия. Но правильным решением является написание скрипта, который будет запущен на всех машинах и проведет необходимую настройку за вас. Централизованным решением данной задачи является система управлением конфигурациями, наиболее распространенной из которых является Ansible.

Ansible — система управления конфигурациями, предназначенная для решения широкого круга задач по автоматизации и сопровождению IT инфраструктуры организации. Простыми словами, Ansible предназначен для автоматизации обслуживания большого числа машин. Главное отличие Ansible от существующих аналогов — не нужна установка дополнительного ПО на машины клиентов – взаимодействие происходит посредством SSH.

Алгоритм работы Ansible, в общем случае, выглядит так:

- Серверная машина соединяется с клиентами, информация о которых берется из файла инвентаризации
- Сервер отправляет на клиенты небольшие программы (модули).
- Отправленные модули с заданными параметрами выполняются клиентах.
- По завершению работы модули на клиентах удаляются.

Настройка Ansible на рабочих станциях.

Для установки программы потребуется выполнить следующую команду:

```
#Установка пакета Ansible (если не установлен)
sudo apt-get install ansible
```

Работа с Ansible ведется через SSH соединение. Поэтому перед использованием, необходимо проверить доступность клиентских машин с помощью протокола ssh и в случае необходимости настроить подключение. Самым простым способом является настройка подключения без использования пароля при создании открытого ключа.

```
#Установка openssh-server (если не установлен)
sudo apt-get install openssh-server
#Создание ключа для SSH соединения (без пароля)
ssh-keygen
#Копирование ключа на машину клиента
ssh-copy-id hostname
```

Настройка файла инвентаризации

Для задания адресов машин, на которых необходимо выполнить задачу используется файл инвентаризации. Данный файл называется hosts и располагается в директории /etc/ansible/ Чтобы добавить список хостов, на которых будут выполняться команды, необходимо перечислить их ip адреса в столбик.

```
192.168.122.1
192.168.122.2
###########
```

Машины возможно объединять в группы. Для этого необходимо перед адресами указать название, под которым они будут объединены.

```
[server]
192.168.122.1
[clients]
192.168.122.2
192.168.122.3
192.168.122.4
```

Далее возможно будет вызывать машины все вместе с помощью ключевого слова all или отдельно, по названию группы.

Запуск скриптов Ansible из командной строки

Самым простым способом отправить команду на все машины – указать её при запуске команды ansible. Например, для того, чтобы проверить соединение со всеми хостами необходимо выполнить команду:

```
ansible -m ping all
```

В случае, если необходимо выполнить команду только на группе машин, то вместо ключевого слова all указывается название группы

```
ansible -m ping server
```

Для выполнения команды на удаленных клиентах возможно воспользоваться ключом -a после которого будет следовать команда оболочки bash. Например:

```
ansible -a 'cat /etc/astra_version' all
```

Для того, чтобы запустить команду от имени администратора, необходимо добавить ключи -b и -K.

```
ansible -m shell -a 'cat /etc/astra_version' –b -K
-b – запустить программу от админа
-K – запросить пароль админа при запуске скрипта
```

Создание и выполнение сценариев (playbook)

Для решения более сложных задач по конфигурации существует возможность описывать их решение в файле сценария (в английской литературе playbook). Сценарии пишутся на языке YAML. Рассмотрим подробно пример сценария

```
- name: Simple playbook
  hosts: all
  become: no
  tasks:
        - name: Whoami
          shell: whoami
```

Первые три строчки сценария относятся ко всему файлу – в них задается комментарий, описывающий операцию (name), на каких машинах будет выполняться (all) и требуется ли выдача прав администратора (no).
Далее, после ключевого слова tasks идет перечисление выполняемых задач. В указанном примере она одна – запуск команды pwd в оболочке каждого из хостов. Вместо ключевого слова shell возможно использовать название одного из поддерживаемых ansible модулей. Ниже приведен список наиболее популярных модулей:

- apt (yum) - управление ПО
- copy - копирование файла
- template - тиражирование шаблонных файлов
- file - создание, удаление файлов, изменение атрибутов файлов
- lineinfile - вставка, замена, удаление строки в текстовом файле
- service - управление службами
- user - управление учетными записями пользователей
- group - управления учетными записями групп
- debug - вывод отладочной информации, значений переменных
- command, shell - выполнение внешних команд ОС

Обратите внимание на синтаксис YAML файлов. Файлы YAML начинаются с трех дефисов, обозначающих начало документа. Однако Ansible не посчитает ошибкой, если вы забудете указать три дефиса в начале сценария. Комментарии начинаются со знака «решетка» и продолжаются до конца строки, как в сценариях на языке командной оболочки, Python и Ruby. Обычно строки в YAML не заключаются в кавычки, даже если они включают пробелы.

В YAML существует понятие списка – перечисление, начинающееся со знака дефис. В данном случае, в начале нашего сценария находится знак «–» для обозначения первого перечисления. Далее указываются переменные и их значения через знак двоеточия. Такой объект называется отображением или словарем.

Обратите внимание, что для отделения задач от описания заголовочной части сценария происходит с помощью **пробелов**, не табуляции!

Для запуска разработанного сценария необходимо выполнить команду:

```
ansible-playbook script.yml
```

В случае успешного запуска команды мы получим сообщение приблизительно следующего содержания:

```
PLAY [Simple playbook]
**********************************************************************
```

```
TASK [Gathering Facts]
*********************************************************************
ok: [192.168.122.2]

TASK [Whoami]
*********************************************************************
changed: [192.168.122.2]

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

Сценарий был выполнен успешно. Однако обратите внимание, результат его выполнения не был возвращен на экран. Для этого необходимо использовать переменные.

Переменные в ansible

Переменные в ansible возможно задать с помощью ключа *vars*. Переменные могут указываться для конкретного хоста или группы хостов внутри файла *inventory*:

- Переменная для хоста указывается после имени хоста в виде:
  переменная=значение;
- Переменные для группы задаются внутри секции
  [имя_группы:vars] в формате переменная=значение.

Для вывода значений переменных необходимо создать отдельную задачу с ключом *debug*. Обращение к переменной происходит с помощью двойных фигурных скобок.
Чтобы получить значение из выполненного на другом хосте скрипта, используется ключ *register*, после которого следует название переменной, в которую сохраняется значение.
Рассмотрим следующий пример:

```
- name: Simple playbook
  hosts: client
  become: no
  vars:
    X: Result
  tasks:
  - name: Whoami
    shell: whoami
    register: output_data

  - name: Print result
    debug:
     msg: “{{ X }} = {{ output_data.stdout }}”
```

В данном примере создается переменная X, содержащая строку «Result». Её значение подставляется в задаче Print result. Результат выполнения команды *whoami* записывается в переменную *output_data*. Кроме непосредственно самого результата выполнения, в переменную заносится информация об успешности выполнения операции и некоторые другие служебные данные. Т.к. нас интересует только результат – то вы обращаемся к полю *stdout*.

Результат выполнения скрипта:

```
PLAY [Simple playbook]
*********************************************************************

TASK [Gathering Facts]
*********************************************************************
ok: [192.168.122.2]

TASK [Whoami]
*********************************************************************
changed: [192.168.122.2]

TASK [Print result]
*********************************************************************
ok: [192.168.122.2] => {
    "msg": "Result = adminstd"
}

PLAY RECAP
*********************************************************************
192.168.122.2              : ok=3    changed=1    unreachable=0    failed=0
```

Обратите внимание, что было выполнено две задачи - Whoami и Print result.

К строкам возможно обращаться как к массивам, в которых переменные отделены определенным символом разделителем. Для получения определенного значения необходимо выполнить метод split('/'), где в скобках указан символ разделения, а далее обратиться к переменной, как к элементу массива с помощью номера в квадратных скобках. Например: `result.stdout.split('.')[2]`

Примеры сценариев

Рассмотрим несколько примеров сценариев Ansible.

Создание директории

```
        - name: Create directory
          file:
                path: ~/work
                state: directory
```

Создание файла

```
        - name: Create file
          file:
                path: ~/test
                state: touch
```

Копирование файла на клиенте

```
        - name: Copy file on client
          copy:
                src: ~/test
                dest: ~/work/test
                remote_src: yes
```

Условный оператор

В ansible возможно выполнять или пропускать некоторые задачи в зависимости от выполнения условия. Для этого используется ключевое слово when.

```
- name: Simple playbook
  hosts: client
  become: no
  vars:
    X: 10
  tasks:
    - name: if/when X > 5
      debug:
        msg: "> 5"
      when: X|int > 5

    - name: if/when X < 5
      debug:
        msg: "< 5"
      when: X|int < 5
```

Обратим внимание на строку «when: X|int > 5». В данном случае с помощью конструкции X|int мы приводим переменную X к целому типу и далее сравниваем с числом 5. Т.к. значение X=10 > 5, то данное задание выполняется. Следующее задание напротив, будет пропущено.

```
TASK [if/when X > 5]
*********************************************************************
ok: [192.168.122.2] => {
    "msg": "> 5"
}

TASK [if/when X < 5]
*********************************************************************
skipping: [192.168.122.2]
```

Редактирование текста

Для редактирования текстовых файлов используется ключ lineinfile

Добавить строку в файл:

```
    - name: Place line
      lineinfile:
        line: Hello World
        path: hello.txt
        create: true
```

Удалить строку из файла:

```
     - name: Remove line
       lineinfile:
         line: Hello World
         path: hello.txt
         state: absent
```

Изменение файла:

```
- name: Change file
  lineinfile:
   line: Passwords no
   regexp: ^Passwords
   path: ~/work/secret_file
```

Предположим, существует файл, содержащий строку Passwords yes. Для изменения значения этой строки найдем строку, начинающуюся со слова Passwords (regexp: ^Passwords) и изменим значение на Passwords no (line: Passwords yes)

Практическая работа

1. Настройте ansible на серверной машине. В качестве клиентов выберете обе машины – сервер и клиент.
2. Проверьте доступность всех устройств с помощью команды ping используя запуск скрипта ansible
3. Используя ansible, запустите на машине клиента скрипт, выводящий объем свободной оперативной машины
4. Создайте playbook, выполняющий следующие задания:
    1.1. Создайте директории ServerВАШИИНИЦИАЛЫ на сервере и ClientВАШИИНИЦИАЛЫ на машине клиента соответственно. Данные директории создаются в домашней директории пользователя.
    2.1. Создайте файлы с названием info в домашней директории. Добавьте проверку на существование файла. В случае его наличия файл повторно не создается.
    3.1. Заполните данные файлы информацией о системе, включающей в себя имя машины, вашу фамилию, ip адрес, объем занятой оперативной памяти (в Mb), среднюю нагрузку на последние 15 минут работы (см. файл /proc/loadavg). Формат записи: astra001 | Ivanov | 192.168.122.1 | 722 | 1.58
    4.1. Скопируйте данный файл в созданную в п. 1.1 директорию.
    5.1. Измените в перемещенном файле значение вашей фамилии на ваше имя.
    6.1. В зависимости от значения нагрузки в файле выведите сообщение на экран. Если нагрузка больше 1: state NAME_MACHINE bad. Если меньше 1, то state NAME_MACHINE good.

Вопросы для проверки:

1. Какая команда используется для проигрывания сценариев?
2. Как называется ключ в play, в котором указываются хосты для выполнения заданий (task)?
3. Что нужно сделать для того, чтобы задания в сценарии выполнялись с привилегиями администратора?

Список литературы

[1] Л. Хоштейн и Р. Мозер, Запускаем Ansible, Москва: ДМК, 2018.

[2] «Документация ansible,» 07 11 2023. [В Интернете]. Available: https://docs.ansible.com/.