Использование роли и плейбука Ansible на примере установки и настройки NGINX

В данных материалых разберем использование Ansible для установки веб-сервера NGINX. Мы попробуем продемонстрировать максимум возможностей системы управления — плейбуки (playbook), роли (role), переменные (vars), шаблоны (templates), обработчики (handlers). Также мы сделаем развертывание веб-сервера на 2 разных семейства операционных систем — на основе deb и RPM.

Группы хостов

Создадим тестовую группу хостов, на которой будем тестировать удаленную установку приложения.

Открываем файл хостов ansible:

vi /etc/ansible/hosts

Добавляем группу хостов или редактируем ее:

[redhat-servers]  
192.168.0.11  
  
[debian-servers]  
192.168.0.12

*\* в данном примере мы создаем 2 группы хостов —****redhat-servers****для серверов семейства Red Hat и****debian-servers****— для deb-серверов. В каждую группу входят по одному серверу —****192.168.0.11****и****192.168.0.12****.*

Проверить доступность хостов для ansible можно командой:

ansible -m ping all -u ansible -k

*\* данной командой мы пропингуем все хосты из инвентаризационного файла hosts.*

... мы должны получить, примерно, такой ответ:

192.168.0.11 | SUCCESS => {  
    "ansible\_facts": {  
        "discovered\_interpreter\_python": "/usr/bin/python3"  
    },   
    "changed": false,   
    "ping": "pong"  
}  
192.168.0.12 | SUCCESS => {  
    "ansible\_facts": {  
        "discovered\_interpreter\_python": "/usr/bin/python"  
    },   
    "changed": false,   
    "ping": "pong"  
}

Создание плейбука

Создаем файл для playbook:

vi /etc/ansible/play.yml

---  
- hosts: redhat-servers  
  become:  
    true  
  become\_method:  
    su  
  become\_user:  
    root  
  remote\_user:  
    ansible  
  roles:  
   - epel  
   - nginx  
  
- hosts: debian-servers  
  become:  
    true  
  become\_method:  
    sudo  
  become\_user:  
    root  
  remote\_user:  
    ansible  
  roles:  
   - nginx

*\* где:*

* ***---****— начало файла YAML. Данный формат имеет строгую структуру  — важен каждый пробел;*
* ***hosts****— группа хостов, к которым будут применяться правила плейбука (если мы хотим, чтобы правила применялись ко всем хостам, указываем****hosts: all****);*
* ***become****— указывает на необходимость эскалации привилегий;*
* ***become\_method****— метод эскалации привилегий;*
* ***become\_user****— пользователь под которым мы заходим с помощью****become\_method****;*
* ***remote\_user****— пользователь, под которым будем подключаться к удаленным серверам;*
* ***roles****— список ролей, которые будут применяться для плейбука.*

*\* В данном случае мы задействуем нашу группы хостов, которые создали в самом начале; повышаем привилегии методом****su****под пользователем****root****(su - root) для группы****redhat-servers****и методом****sudo****для****debian-servers****; подключение к серверам выполняется от пользователя****ansible****; используем созданную нами роль****nginx****(саму роль мы создадим позже). Для систем RPM сначала выполним роль****epel****— она будет отвечать за установку репозитория EPEL, так как в стандартном репозитории nginx нет.*

Стоит отдельно уделить внимание вопросу повышения привилегий. IT-среды могут применять разные политики относительно безопасности. Например, на серверах CentOS, по умолчанию, нет sudo и повышать привилегии нужно с помощью su. В Ubuntu наоборот — по умолчанию есть sudo, а su не работает, так как пароль на root не устанавливается. В данном случае есть несколько путей при работе с Ansible:

1. Как в нашем примере, разделить группы хостов на разные семейства операционных систем и применять к ним разные методы повышения привилегий. Чтобы данный плейбук корректно отработал, учетная запись, под которой идет удаленное подключение к серверу (в нашем примере ansible) должна иметь такой же пароль, как у пользователей root на серверах семейства Red Hat. Данный метод удобен с точки зрения отсутствия необходимости приводить к единому виду безопасность серверов разного семейства. Однако, с точки зрения безопасности лучше, чтобы пароли у root и ansible были разные.
2. Использовать метод для создания плейбука, как описан выше, но запускать его с ключом --limit, например, **ansible-playbook --limit=debian-servers ...** — таким образом, мы запустим отдельные задания для каждого семейства операционных систем и сможем ввести индивидуальные пароли для каждого случая.
3. Мы можем на всех серверах deb установить пароль для пользователя root, таким образом, получив возможность для become\_method: su.
4. И наконец, можно для серверов Red Hat установить sudo и проходить become с помощью метода sudo.

Создание роли

Роли в Ansible используются для логического разделения плейбука. Они имеют строгий синтаксис и файловую структуру. Таким образом, конфигурация становится более упорядоченной и понятной для дальнейшей поддержки.

И так, для ролей должна быть четкая файловая структура — создаем каталоги:

mkdir -p /etc/ansible/roles/nginx/tasks

mkdir -p /etc/ansible/roles/epel/tasks

*\* в данном случае мы создали каталоги****nginx****,***epel***и****tasks****внутри каталога****roles****. В ansible это означает, что мы создали роли с названием****nginx****и****epel****, а файл****main.yml****, который мы поместим в каталоги****tasks****будет описывать задачи данных ролей.*

Создаем файл с описанием задач для роли nginx:

vi /etc/ansible/roles/nginx/tasks/main.yml

---  
- name: Install Nginx Web Server on RedHat Family  
  yum:  
    name=nginx  
    state=latest  
  when:  
    ansible\_os\_family == "RedHat"  
  notify:  
    - nginx systemd  
  
- name: Install Nginx Web Server on Debian Family  
  apt:  
    name=nginx  
    state=latest  
  when:  
    ansible\_os\_family == "Debian"  
  notify:  
    - nginx systemd

*\* где*

* ***---****— начало файла YAML;*
* ***name****— название для роли (может быть любым);*
* ***yum/apt****— используемый модуль для установки приложения;*
* ***yum/apt name****— название пакета, которое мы устанавливаем;*
* ***yum/apt state****— состояние пакета, которое должно контролироваться ролью;*
* ***when****— условие, при котором данная роль будет выполняться;*
* ***notify****— обработчик, который будет вызван в случае успешного выполнения задачи. При необходимости, можно задать несколько обработчиков;*

*\* В данном примере мы создали простую задачу для роли по развертыванию****nginx****. На системы RPM установка выполняется с помощью модуля****yum****, на deb —****apt****. Версия должна быть последней (****latest****); после установки пакета, необходимо разрешить автозапуск сервиса и запустить его.  
\* при установке пакетов также следует учитывать, что некоторые могут иметь разные названия в разных системах. Например, Apache в RMP-системах называется****httpd****, в deb —****apache2****.*

Создаем файл с описанием задач для роли epel:

vi /etc/ansible/roles/epel/tasks/main.yml

---  
- name: Install EPEL Repo  
  yum:  
    name=epel-release  
    state=present

Обратите внимание, что в плейбуке выше мы задействовали notify, но не задали handlers — в качестве примера, мы вынесем его в отдельный файл:

mkdir /etc/ansible/roles/nginx/handlers

vi /etc/ansible/roles/nginx/handlers/main.yml

---  
- name: nginx systemd  
  systemd:  
    name: nginx  
    enabled: yes  
    state: started

*\*****handlers****— описание обработчика, который может быть вызван с помощью notify;****systemd****— модуль для управления юнитами systemd;****systemd enabled****— разрешает или запрещает сервис;****systemd state****— состояние, которое должно быть у сервиса. В данном примере мы указываем, что у демона****nginx****должно быть состояние****started****и разрешен автозапуск (****enabled****).*

Запуск плейбука

Запускаем наш плейбук:

ansible-playbook /etc/ansible/play.yml -kK

После ввода данной команды система запросит первый пароль для учетной записи, от которой мы подключаемся к удаленным серверам (в нашем примере мы задали ее в самом плейбуке, опции remote\_user):

SSH password:

После ввода пароля, будет запрошен второй пароль — для повышения привилегий на самой удаленной системе:

BECOME password[defaults to SSH password]:

... в итоге мы должны увидеть следующую картину:

192.168.0.11    : ok=2    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0     
192.168.0.12    : ok=2    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0

Настройка NGINX с помощью шаблона ansible

После установки веб-сервера nginx необходимо его настроить. Процесс настройки мы разделим на создание переменных и шаблонов конфигурационных файлов, а также настройки роли для формирования конфигурационных файлов на основе шаблона.

Создаем общие переменные

Переменные задаются в файле main.yml, который находится в каталоге vars, который, в свою очередь, находится в каталоге роли. И так, создаем каталог для переменных:

mkdir /etc/ansible/roles/nginx/vars

... и сам файл main.yml:

vi /etc/ansible/roles/nginx/vars/main.yml

worker\_processes: auto  
worker\_connections: 2048  
client\_max\_body\_size: 512M

*\* где представлены некоторые опции настройки NGINX:*

* ***worker\_processes****— определяет количество рабочих процессов. Чаще всего, оптимальнее выставлять автоматическое конфигурирование.*
* ***worker\_connections****— максимальное количество соединений одного рабочего процесса.*
* ***client\_max\_body\_size****— максимальный размер загружаемых данных.*

Шаблон для nginx.conf

Создаем каталог для хранения шаблонов:

mkdir /etc/ansible/roles/nginx/templates

... и создаем первый шаблон для конфигурационного файла nginx.conf:

vi /etc/ansible/roles/nginx/templates/nginx.conf

user  {{ nginx\_user }};  
worker\_processes  {{ worker\_processes }};  
worker\_priority     -1;  
  
error\_log  /var/log/nginx/error.log warn;  
pid        /var/run/nginx.pid;  
  
events {  
    worker\_connections  {{ worker\_connections }};  
}  
  
http {  
    include       /etc/nginx/mime.types;  
    default\_type  application/octet-stream;  
  
    log\_format  main  '$remote\_addr - $remote\_user [$time\_local] "$request" '  
                      '$status $body\_bytes\_sent "$http\_referer" '  
                      '"$http\_user\_agent" "$http\_x\_forwarded\_for"';  
    access\_log  /var/log/nginx/access.log  main;  
  
    sendfile on;  
    tcp\_nopush on;  
    tcp\_nodelay on;  
    keepalive\_timeout  65;  
    reset\_timedout\_connection  on;  
    client\_body\_timeout        35;  
    send\_timeout               30;  
  
    gzip on;  
    gzip\_min\_length     1000;  
    gzip\_vary on;  
    gzip\_proxied        expired no-cache no-store private auth;  
    gzip\_types          text/plain text/css application/json application/x-javascript text/xml application/xml application/xml+rss text/javascript application/javascript;  
    gzip\_disable        "msie6";  
  
    types\_hash\_max\_size 2048;  
    client\_max\_body\_size {{ client\_max\_body\_size }};  
    proxy\_buffer\_size   64k;  
    proxy\_buffers   4 64k;  
    proxy\_busy\_buffers\_size   64k;  
    server\_names\_hash\_bucket\_size 64;  
  
    include /etc/nginx/modules-enabled/\*.conf;  
    include /etc/nginx/conf.d/\*.conf;  
    include /etc/nginx/sites-enabled/\*;  
}

*\* это пример рабочего конфигурационного файла. Переменных могло быть и больше, но для демонстрации их использования, вполне, достаточно. Сами переменные заключены в двойные фигурные скобки и были нами определены в файле выше (кроме переменной****nginx\_user****, которая будет определена в настройках плейбука, так как для каждого типа операционной системы она должна быть своя).*

Теперь настраиваем нашу роль для использования шаблона:

vi /etc/ansible/roles/nginx/tasks/main.yml

Добавляем:

- name: Replace nginx.conf  
  template:  
    src=templates/nginx.conf  
    dest=/etc/nginx/nginx.conf

*\* данная роль определяет, каким файлом на сервере ansible (****src****) необходимо заменить файл на удаленной системе (****dest****).*

И последнее, открываем файл плейбука:

vi /etc/ansible/play.yml

... вставляем:

- hosts: redhat-servers  
  vars:  
    nginx\_user: nginx  
    ...  
  
- hosts: debian-servers  
  vars:  
    nginx\_user: www-data  
    ...

*\* в данном примере мы создаем переменную****nginx\_user****. Так как для разных семейств операционных систем используются разные пользователя для nginx по умолчанию, мы также сделали переменную с разными значениями для RPM и deb.*

Готово — запускаем плейбук:

ansible-playbook /etc/ansible/play.yml -kK

... и вводим дважды пароли. На наших серверах должен появиться конфигурационной файл nginx.conf в соответствии с нашим шаблоном.

Шаблон для создания виртуальных доменов в NGINX

Создадим виртуальные домены при помощи ansible. Сначала создадим переменные — для каждого сервера должны быть свои домены — такие переменные можно задать в файле hosts, котором мы создавали хосты для ansible:

vi /etc/ansible/hosts

Добавляем к нашим строкам:

[redhat-servers]  
192.168.0.11 virtual\_domain=domain1.dmosk.ru  
  
[debian-servers]  
192.168.0.12 virtual\_domain=domain2.dmosk.ru

*\* мы создали переменную****virtual\_domain****, у которой будут свои значения для каждого хоста.*

Создаем шаблон:

vi /etc/ansible/roles/nginx/templates/nginx\_vhosts.conf

server {  
    listen       80;  
    server\_name  {{ virtual\_domain }} www.{{ virtual\_domain }};  
    root /var/www/{{ virtual\_domain }};  
  
    access\_log /var/log/nginx/{{ virtual\_domain }}\_access\_log;  
    error\_log /var/log/nginx/{{ virtual\_domain }}\_error\_log;  
  
    location / {  
        fastcgi\_pass unix:{{ fastcgi\_pass\_path }};  
        proxy\_redirect     off;  
        proxy\_set\_header   Host             $host;  
        proxy\_set\_header   X-Forwarded-Proto $scheme;  
        proxy\_set\_header   X-Real-IP        $remote\_addr;  
        proxy\_set\_header   X-Forwarded-For  $proxy\_add\_x\_forwarded\_for;  
    }  
      
    location ~\* ^.+\.(jpg|jpeg|gif|png|css|zip|tgz|gz|rar|bz2|doc|docx|xls|xlsx|exe|pdf|ppt|tar|wav|bmp|rtf|js)$ {  
            expires modified +1w;  
    }  
}

*\* в данном примере мы создаем простой конфигурационный файл для создания виртуального домена. В качестве самого домена берется значение переменной****virtual\_domain****, которую мы задали через файл****hosts****. Переменная****fastcgi\_pass\_path****будет создана для каждой группы систем своя и определяем путь до сокетного файла для взаимодействия nginx с fastcgi.*

Настраиваем нашу роль — создаем 3 задачи:

vi /etc/ansible/roles/nginx/tasks/main.yml

Добавляем:

---  
...  
  
- name: Create home directory for www  
  file:  
    path: /var/www/{{ virtual\_domain }}  
    state: directory  
  
- name: Add virtual domain in NGINX for RPM  
  vars:  
    fastcgi\_pass\_path: /var/run/php-fpm/php5-fpm.sock  
  template:  
    src=templates/nginx\_vhosts.conf  
    dest=/etc/nginx/conf.d/{{ virtual\_domain }}.conf  
  when:  
    ansible\_os\_family == "RedHat"  
  notify:  
    - nginx restart  
  
- name: Add virtual domain in NGINX for Deb  
  vars:  
    fastcgi\_pass\_path: /run/php/php7.2-fpm.sock  
  template:  
    src=templates/nginx\_vhosts.conf  
    dest=/etc/nginx/sites-enabled/{{ virtual\_domain }}.conf  
  when:  
    ansible\_os\_family == "Debian"  
  notify:  
    - nginx restart

*\* первая задача нужна для создания каталога, в котором будут находиться файлы сайта (без данного каталога наш конфигурационный сайт выдаст ошибку при попытке перезапустить сервис nginx). Вторая задача создаем конфигурационный файл для виртуального домена, который находится в каталоге****/etc/nginx/conf.d****; источником данных служит созданный нами шаблон****nginx\_vhosts.conf****; также мы задаем переменную****fastcgi\_pass\_path****, которая будет указывать путь до сокетного файла. Третья задача также создает конфигурацию для виртуального домена на базе созданного шаблона; в отличие от второй задачи, третья создает конфигурационный файл в каталоге****/etc/nginx/sites-enabled****, а также имеет другое значение переменной****fastcgi\_pass\_path****. После успешного выполнения задачи, будет отправлен сигнал на перезапуск службы nginx для применения настроек (некоторые сервисы принимают более предпочтительный вариант —****reloaded****).Также мы добавили условие — вторая задача будет применяться к система на базе Red Hat, третья — Debian.  
\* на самом деле, мы могли две последние задачи объединить в одну, а переменную****fastcgi\_pass\_path****задать в файле плейбука. Но мы так поступили намеренно, чтобы продемонстрировать возможность использования переменных при описании роли.*

Открываем обработчик:

vi /etc/ansible/roles/nginx/handlers/main.yml

Дописываем:

---  
...  
  
- name: nginx restart  
  service:  
    name=nginx  
    state=restarted

Запускаем плейбук:

ansible-playbook /etc/ansible/play.yml -kK

Теги

В нашем примере нам не довелось использовать теги — еще одну удобную возможность управления запуском плейбука.

Теги позволяют отметить роль и при запуске плейбука запустить именно ее, проигнорировав другие роли. Например, есть такой плейбук:

...  
  roles:  
    - role: nginx  
      tags: web1  
    - role: apache  
      tags: web2

*\* в данном плейбуке есть две роли — одна называется****nginx****, вторая —****apache****; для каждой роли мы задали свой тег.*

Теперь, чтобы запустить плейбук, но выполнить в нем определенную роль, нам достаточно указать тег:

ansible-playbook --tags web2 /etc/ansible/play.yml -kK

*\* данная команда запустит плейбук с выполнением только роли с тегом****web2****(в нашем примере,****apache****).*

Ansible Galaxy

На последок хочется сказать пару слов о Ansible Galaxy. Грубо говоря, это репозиторий готовых плейбуков и ролей. Найти файлы, соответствующую задаче, а также документацию можно на портале [galaxy.ansible.com](https://galaxy.ansible.com/). Установка выполняется командой ansible-galaxy, например:

ansible-galaxy install geerlingguy.apache

*\* данная команда создаст в каталоге пользователя папку****.ansible/roles/geerlingguy.apache****— в нее поместит файлы с описанием роли.*

Готовые библиотеки можно использовать для выполнения задач или просто как шпаргалки.