Автоматизация с Ansible

Продолжаем знакомиться с системой управления конфигурациями Ansible. Jinja2 как язык, используемый в шаблонах. Пишем плейбук для установки и настройки nginx, используя шаблоны и переменные Ansible в конфигурационных файлах nginx.

[Шаблоны Jinja2](#_nhceoe68m3jm)

[Переменные](#_b1ie7mm5jztk)

[Плейбук nginx](#_38lrs7i2i55)

[Домашнее задание](#_qk61dk4of60m)

[Дополнительные материалы](#_y937sk8fclye)

[Используемая литература](#_uvp6qax5r1ok)

# Шаблоны Jinja2

Шаблоны в ansible используются для тех же целей, что и в языках программирования. Используя переменные и различные условия, можно настроить много типовых конфигураций серверов.

Jinja2 изначально был разработан как шаблонизатор для Python, но также нашел очень удачное применение в Ansible.

Ansible ищет шаблоны в папке templates для каждой роли. Там и будем создавать шаблоны конфигурационных файлов для nginx. Принято всем шаблонам давать расширение j2, хоть это и не обязательно.

Создайте новую роль и плейбук для нее, как на предыдущем уроке, назовите ее *nginx.*

Для настройки самого nginx понадобятся 2 конфига:

1. nginx.conf.j2 – в нем хранятся глобальные настройки веб-сервера.
2. Шаблон для виртуальных хостов (для удобства назовем его vhosts.conf.j2).

Содержимое nginx.conf.j2:

|  |
| --- |
| user www-data;worker\_processes {{ ansible\_processor\_cores }};error\_log /var/log/nginx/error.log;pid /var/run/nginx.pid;worker\_rlimit\_nofile 65536;timer\_resolution 100ms;events { worker\_connections 5000; multi\_accept on; use epoll;}http { resolver 8.8.8.8; log\_format graylog 'HOST: $host, $remote\_addr - $remote\_user [$time\_local] $request ' '$status, BYTES: $body\_bytes\_sent, TIME: $request\_time, REFERER: $http\_referer ' 'USER-AGENT: $http\_user\_agent REAL-IP: $http\_x\_forwarded\_for'; access\_log /var/log/nginx/access.log graylog; include /etc/nginx/mime.types; client\_body\_buffer\_size 16K; client\_header\_buffer\_size 1K; client\_max\_body\_size 20M; large\_client\_header\_buffers 8 32K; open\_file\_cache max=5000 inactive=20s; open\_file\_cache\_valid 30s; open\_file\_cache\_min\_uses 2; open\_file\_cache\_errors on; client\_body\_timeout 5; client\_header\_timeout 5; keepalive\_timeout 35; send\_timeout 30; reset\_timedout\_connection on; types\_hash\_max\_size 2048; server\_names\_hash\_bucket\_size 256; server\_tokens off; sendfile on; tcp\_nopush on; tcp\_nodelay on; gzip on; gzip\_buffers 64 8k; gzip\_comp\_level 6; gzip\_min\_length 1000; gzip\_proxied expired no-cache no-store private auth; gzip\_http\_version 1.1; gzip\_types text/plain text/css application/json application/x-javascript text/xml application/xml application/xml+rss text/javascript; gzip\_disable "MSIE [1-6]\."; proxy\_next\_upstream error timeout invalid\_header http\_429 http\_500 http\_502 http\_503 http\_504; proxy\_temp\_path /var/cache/nginx/client\_temp; proxy\_set\_header Host $http\_host; proxy\_set\_header X-Real-IP $remote\_addr; proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for; proxy\_set\_header X-Forwarded-Proto $scheme; proxy\_buffers 32 16k; proxy\_connect\_timeout 30s; proxy\_send\_timeout 3s; proxy\_read\_timeout 3s; include /etc/nginx/conf.d/\*.conf;} |

Количество процессов nginx задается переменной ansible *{{ ansible\_processor\_cores }}* При запуске плейбука собираются так называемые «факты» о системе. Их можно использовать в шаблонах и плейбуках. Полный список можно посмотреть с помощью команды *ansible –m setup –u user –kK host,* где user – имя пользователя, под которым мы заходим на сервер, и host – сам сервер.

Содержимое vhosts.conf.j2:

|  |
| --- |
| server { listen 80; server\_name {{ item }}; root /var/www/html; index index.html index.htm index.nginx-debian.html; location ~\* \.(jpg|jpeg|gif|css|png|js|ico|html)$ { expires max; } location / { try\_files $uri $uri/ /index.html; } access\_log /var/log/nginx/{{ item }}.access.log graylog; error\_log /var/log/nginx/{{ item }}.error.log;} |

Здесь в *server\_name* и в настройках логирования в качестве параметра указано взять значение переменной *{{ item }}*.

## Переменные

Как известно из предыдущего урока, переменные могут храниться в нескольких местах. В нашем случае их нужно добавить в *defaults/main.yaml:*

|  |
| --- |
| ---nginx: version: 1.14\* domains: - domain1.test.lan - domain2.test.lan - domain3.test.lan |

В качестве версии укажите актуальную на данный момент в репозиториях. В переменной *domains* передается список доменов, для которых необходимо создать виртуальные хосты.

## Плейбук nginx

Пишем код для установки и настройки nginx в *tasks/main.yaml.*

Сначала добавим ключ для репозитория nginx и сам репозиторий, используя модули apt\_key и apt\_repository:

|  |
| --- |
| ---- name: Add Nginx repo apt key apt\_key: url=https://nginx.org/keys/nginx\_signing.key state=present- name: Add Nginx repo apt\_repository: repo: "deb http://nginx.org/packages/ubuntu/ xenial nginx" state: present update\_cache: yes |

Установим nginx определенной версии, используя уже известный модуль apt. Как раз здесь и используется переменная *version,* которую мы задали выше. Обращение ко вложенным переменным происходит через точку. Т. е. у нас есть общая переменная *nginx,* в которой есть еще две – *version* и *domains.* Сейчас нужно значение переменной *version.*

|  |
| --- |
| - name: Install Nginx apt: name="nginx={{ nginx.version }}" state=present update\_cache=yes |

Осталось только скопировать шаблоны конфигурационных файлов на сервер. В главном конфиге nginx у нас указана директива *include /etc/nginx/conf.d/\*.conf;.* Поэтому сначала проверим, есть ли такая папка, и если нет, то создадим. Для проверки есть модуль stat, который использует одноименную утилиту. Результат надо куда-то записать для дальнейшего использования. Для этого используется конструкция *register: имя\_переменной.*

|  |
| --- |
| - name: Checking if Nginx includes folder exists stat: path=/etc/nginx/conf.d register: nginx\_includes |

Чтобы понимать, с чем мы имеем дело и как оттуда получить значение о присутствии или отсутствии папки, посмотрим, что внутри этой переменной. Используем для этого модуль debug.

|  |
| --- |
| - name: Lets see what inside debug: var=nginx\_includes |

А внутри json-структура, в которой содержится все, что отдал модуль stat. Нам оттуда нужны значения переменных *nginx\_includes.stat.exists* и *nginx\_includes.stat.path.* Теперь, когда мы знаем значение, для проверки можно и создать папку, если ее нет. Для создания файлов, папок, ссылок в ansible есть модуль file. Также используем новые модули для форматирования вывода и условия выполнения этого блока.

|  |
| --- |
| - name: Creating Nginx includes folder file: path="{{ nginx\_includes.stat.path }}" state=directory owner=root group=root mode=0755 recurse=yes when: nginx\_includes.stat.exists == False |

Наконец-то скопируем шаблоны nginx на сервер с помощью модуля *template.* В src указываем название шаблона в папке templates, в dest – куда его положить на сервере.

|  |
| --- |
| - name: Copy nginx.conf template to server template: src=nginx.conf.j2 dest=/etc/nginx/nginx.conf owner=root group=root mode=0644 notify: restart nginx |

Внимание! После копирования конфига с глобальными настройками надо перезапустить nginx. Это, конечно, можно сделать в следующем блоке с помощью модуля *shell* или *systemd* с условием, что конечный файл изменился. Однако есть способ проще: в Ansible существуют триггеры, которые срабатывают, если при выполнении данного блока что-то было изменено. Именно триггер здесь и указан в *notify: restart nginx.* Чтобы его использовать, сначала его нужно создать. Все триггеры создаются в *handlers/main.yaml.* Добавим его в этот файл и сразу же добавим триггер для перезагрузки nginx.

|  |
| --- |
| ---- name: restart nginx systemd: name=nginx state=restarted enabled=yes- name: reload nginx systemd: name=nginx state=reloaded enabled=yes |

Отрабатывают триггеры в самом конце плейбука в том порядке, в котором они описаны в файле. Если есть необходимость выполнить триггер сразу после какого-то блока, то после него надо добавить блок со следующим содержимым:

|  |
| --- |
| - name: Force all notified handlers to run at this meta: flush\_handlers |

Теперь скопируем последний шаблон с виртуальными хостами. И здесь понадобится использовать переменную *nginx.domains,* а точнее то, что в ней лежит. Чтобы понять, что в ней лежит, посмотрите это с помощью модуля debug, как делали это выше. После добавления или изменения виртуальных хостов нужно перезагрузить nginx, для чего используем триггер, добавленный выше.

|  |
| --- |
| - name: Copy Nginx vhosts configs to server template: src=vhosts.conf.j2 dest="/etc/nginx/conf.d/{{ item }}.conf" owner=root group=root mode=0644 loop: "{{ nginx.domains }}" loop\_control: label: "Copy vhost config {{ item }}.conf to Nginx server" notify: reload nginx |

Когда мы указали *loop: "{{ nginx.domains }}",* эта переменная внутри ansible в данном блоке будет доступна как переменная item. Исходя из этого, в dest указываем *{{ item }},* что можно рассматривать как *nginx.domains.0, nginx.domains.1* и т. д. (это массив значений, и через loop происходит итерация).

Модуль *loop\_control* используется для форматирования вывода ansible в консоль. Его удобно использовать, когда слишком много параметров выводятся в консоль и превращаются в кашу.

Немного улучшим nginx-сервер, добавив в него ssl-сертификаты. Создадим скрипт, который будет генерировать самоподписанные сертификаты, и положим его в *files/create\_cert.sh* текущей роли.

|  |
| --- |
| #!/bin/bashopenssl genrsa -out rootCA.key 4096openssl req -x509 -new -key rootCA.key -days 10000 -out rootCA.crt -subj "/C=RU/O=Unknown, OOO/OU=Certification Services Division/OU=(c) 2018 Unknown Org. -For authorized use only/CN=geekbrains"chmod 0600 rootCA.key |

Его нужно скопировать на сервер не как шаблон, а как обычный файл. Используем для этого модуль copy. Чтобы было куда копировать, нужно создать папку для сертификатов и скрипта, например /etc/nginx/ssl. Это сделайте сами сразу после блока с установкой nginx по примеру создания /etc/nginx/conf.d. Теперь копируем скрипт на сервер:

|  |
| --- |
| - name: Copy ssl certificate creation script to server copy: src=create\_cert.sh dest="/etc/nginx/ssl/create\_cert.sh" owner=root group=root mode=0700 |

И генерируем сертификаты. Для этого используем модуль shell. Но если в него добавить запуск этого скрипта, сертификаты будут генерироваться каждый раз при запуске плейбука, а нам это не нужно. На этот случай у модуля shell есть параметры args (<http://docs.ansible.com/ansible/latest/modules/command_module.html>). Из-за наличия creates: *rootCA.crt* bash-команда будет выполняться, только если нет файла rootCA.crt. *Chdir: /etc/nginx/ssl* нужно, чтобы перейти в эту папку и там запустить скрипт, иначе сертификаты сгенерируются там, где вы этого совсем не ожидаете.

|  |
| --- |
| - name: Generating if not exists ssl certificate on server shell: "bash /etc/nginx/ssl/create\_cert.sh" args: chdir: /etc/nginx/ssl creates: rootCA.crt notify: restart nginx |

Конфиг nginx для виртуальных хостов поменяйте сами так, чтобы он поддерживал ssl-сертификаты.

Осталось только создать файл плейбука и указать в нем эту роль с хостами или группой хостов, на которые она будет применяться. Это сделайте сами, как на предыдущем уроке.

# Домашнее задание

1. Повторить действия, выполненные на уроке.
2. Создать файл плейбука и указать в нем роль с хостами или группой хостов, на которые она будет применяться.

# Дополнительные материалы

1. <https://ru.wikipedia.org/wiki/Jinja>
2. [http://docs.ansible.com/ansible/latest/modules/command\_module.htm](http://docs.ansible.com/ansible/latest/modules/command_module.html)

# Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. [http://docs.ansible.com/ansible/latest/modules/command\_module.htm](http://docs.ansible.com/ansible/latest/modules/command_module.html)