perplexity

Решение через Ansible с подробными объяснениями для начинающего DevOps

Структура проекта

```
helloworld-proxy-ansible/
├── group_vars/
│   └── all.yml            # Общие переменные
├── roles/
│   ├── common/            # Базовая настройка
│   ├── python_app/        # Flask приложение
│   ├── node_app/          # Express приложение
│   ├── go_app/            # Go HTTP сервер
│   ├── java_app/          # Spring Boot приложение
│   ├── nginx_proxy/       # Конфигурация NGINX
│   └── ssl_cert/          # SSL сертификаты
├── inventory.ini          # Файл инвентаризации
├── playbook.yml           # Главный плейбук
└── templates/             # Шаблоны конфигураций
    ├── nginx/
    │   └── helloworld.j2
    └── systemd/           # unit-файлы для systemd
```

1. Файл переменных (`group_vars/all.yml`)

```
# Параметры приложений
app_ports:
  python: 8001
  node: 8002
  go: 8003
  java: 8004

# SSL параметры
ssl_dir: /etc/nginx/ssl
ssl_key: "{{ ssl_dir }}/selfsigned.key"
ssl_cert: "{{ ssl_dir }}/selfsigned.crt"

# Доменное имя
domain_name: helloworld.alinadobs

# Пользователь для запуска приложений
app_user: "{{ ansible_user }}"
```

Объяснение:
Здесь мы определяем переменные, которые будут использоваться во всех ролях. Это:

- Порты для каждого приложения
- Пути для SSL сертификатов
- Доменное имя для NGINX
- Системный пользователь (используем того же, что и для подключения Ansible)

2. Роль `common` (базовая настройка)

```
# roles/common/tasks/main.yml
- name: Установка системных зависимостей
  apt:
    name:
      - python3
      - python3-pip
      - golang
      - default-jdk
      - nodejs
      - npm
      - curl
      - nginx
      - openssl
    state: present
    update_cache: yes
  tags: dependencies
```

Объяснение:
Эта роль устанавливает все необходимые пакеты. Ключевые моменты:

- `apt` - модуль для управления пакетами в Debian/Ubuntu
- `update_cache: yes` - обновляет кэш пакетов перед установкой
- `tags` - позволяет запускать только эту часть командой `ansible-playbook --tags dependencies`

3. Роль `python_app` (Flask приложение)

```
# roles/python_app/tasks/main.yml
- name: Создание директории для приложения
  file:
    path: /opt/python_app
    state: directory
    owner: "{{ app_user }}"
    group: "{{ app_user }}"

- name: Установка Flask
  pip:
    name: flask
    executable: pip3

- name: Копирование Python приложения
```

```
  copy:
    content: |
      from flask import Flask
      app = Flask(__name__)
      @app.route('/')
      def hello():
          return "Hello from Python Flask!"
      if __name__ == '__main__':
          app.run(host='0.0.0.0', port={{ app_ports.python }})
    dest: /opt/python_app/app.py
    owner: "{{ app_user }}"
    group: "{{ app_user }}"

- name: Настройка systemd службы
  template:
    src: ../../templates/systemd/python_app.service.j2
    dest: /etc/systemd/system/python_app.service
  notify: restart_python_app

- name: Запуск и включение службы
  systemd:
    name: python_app
    state: started
    enabled: yes
```

Шаблон systemd службы (`templates/systemd/python_app.service.j2`):

```
[Unit]
Description=Python Hello World App

[Service]
User={{ app_user }}
ExecStart=/usr/bin/python3 /opt/python_app/app.py
WorkingDirectory=/opt/python_app
Restart=always

[Install]
WantedBy=multi-user.target
```

Объяснение:

1. Создаем директорию для приложения
2. Устанавливаем Flask через pip
3. Копируем код приложения напрямую (без отдельного файла)
4. Создаем systemd службу для автоматического запуска
5. Запускаем службу и добавляем в автозагрузку

4. Роль `nginx_proxy` (настройка NGINX)

```
# roles/nginx_proxy/tasks/main.yml
- name: Настройка конфигурации NGINX
  template:
    src: ../../templates/nginx/helloworld.j2
    dest: /etc/nginx/sites-available/helloworld.conf

- name: Активация конфигурации
  file:
    src: /etc/nginx/sites-available/helloworld.conf
    dest: /etc/nginx/sites-enabled/helloworld.conf
    state: link

- name: Проверка конфигурации NGINX
  command: nginx -t
  register: nginx_test
  changed_when: false
  notify: restart_nginx

- name: Обновление /etc/hosts
  lineinfile:
    path: /etc/hosts
    line: "127.0.0.1 {{ domain_name }}"
    state: present
```

Шаблон NGINX (`templates/nginx/helloworld.j2`):

```
server {
    listen 443 ssl;
    server_name {{ domain_name }};

    ssl_certificate     {{ ssl_cert }};
    ssl_certificate_key {{ ssl_key }};

    gzip on;
    gzip_types text/plain application/json application/javascript text/css;

    access_log /var/log/nginx/helloworld_access.log;
    error_log  /var/log/nginx/helloworld_error.log;

    location /python/ {
        proxy_pass http://127.0.0.1:{{ app_ports.python }}/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    location /node/ {
        proxy_pass http://127.0.0.1:{{ app_ports.node }}/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    location /go/ {
```

```
        proxy_pass http://127.0.0.1:{{ app_ports.go }}/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    location /java/ {
        proxy_pass http://127.0.0.1:{{ app_ports.java }}/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

Объяснение:

1. Генерируем конфиг NGINX из шаблона с подстановкой переменных
2. Активируем конфиг через симлинк
3. Проверяем корректность конфигурации
4. Добавляем запись в /etc/hosts для локального тестирования

5. Главный плейбук (`playbook.yml`)

```
- hosts: all
  become: true
  vars_files:
    - group_vars/all.yml
  roles:
    - common
    - role: ssl_cert
      tags: ssl
    - python_app
    - node_app
    - go_app
    - java_app
    - nginx_proxy
  handlers:
    - name: restart_nginx
      service:
        name: nginx
        state: restarted

    - name: restart_python_app
      systemd:
        name: python_app
        state: restarted

    # Аналогичные handlers для других приложений
```

6. Запуск плейбука

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

Проверка работы:

```
curl -k https://helloworld.alinadobs/python/
curl -k https://helloworld.alinadobs/node/
```

Ключевые концепции Ansible для начинающих

1. **Идемпотентность**
   Ansible гарантирует, что повторный запуск плейбука не изменит систему, если конфигурация уже соответствует желаемому состоянию.
2. **Модули**
   Каждое действие (установка пакетов, копирование файлов) выполняется специализированными модулями (`apt`, `copy`, `template`).
3. **Handlers**
   Специальные задачи, которые выполняются только при изменениях (например, перезапуск служб).
4. **Шаблоны Jinja2**
   Позволяют создавать конфигурационные файлы с подстановкой переменных (как в примере с NGINX).
5. **Система ролей**
   Позволяет разбивать сложную конфигурацию на переиспользуемые компоненты.
6. **Теги (tags)**
   Позволяют запускать только часть плейбука:
   `ansible-playbook --tags ssl playbook.yml`

Что изучит менти

- Автоматизация развертывания инфраструктуры
- Управление конфигурацией через код
- Работа с systemd службами
- Настройка обратного прокси в NGINX
- Генерация SSL сертификатов
- Основы Ansible (переменные, шаблоны, handlers)

Решение полностью соответствует заданию и использует best practices Ansible, подходящие для начинающих DevOps-инженеров.

⁂