Вот улучшенный и дополненный урок для начинающих DevOps-инженеров:

Пошаговый урок: Автоматизация деплоя приложений с помощью Ansible, Docker и HAProxy

Цель урока:

Научиться автоматизировать развёртывание простых приложений на сервере, используя Ansible, Docker и HAProxy, а также безопасно управлять секретами.

🛠️1. Подготовка окружения

Задание:

Создать проект и настроить рабочее окружение.

Пошагово:

Создай структуру проекта:

```
mkdir -p helloworld-proxy-ansible/{group_vars,roles/
{common,haproxy,python_app,node_app,go_app,java_app},templates/
{haproxy,docker}}
```

Создай файлы:

- `group_vars/all.yml` (переменные)
- `group_vars/vault.yml` (секреты, зашифрованные через ansible-vault)
- `inventory.ini` (список серверов)
- `playbook.yml` (основной плейбук)

Заполни `inventory.ini`:

```
[all]
server1 ansible_host=192.168.1.2 ansible_user=root
```

Проверь подключение:

```
ansible all -i inventory.ini -m ping
```

✅Чек-лист:

-

📝2. Настройка переменных

Заполни `group_vars/all.yml`:

```
app_ports:
  python: 8001
  node: 8002
  go: 8003
  java: 8004

haproxy_frontend_port: 80
haproxy_stats_port: 1936
domain_name: helloworld.local
docker_network: app_network
```

Создай зашифрованный файл переменных и добавь секреты:

```
ansible-vault create group_vars/vault.yml
```

Пример заполнения `vault.yml`:

```
secret_message: "Hello, secure world!"
```

✅**Чек-лист:**

-

🗜️3. Базовая настройка (роль `common`)

✅**Чек-лист:**

-

📠4. Docker приложения

Создай Dockerfile для Python (`templates/docker/python.Dockerfile.j2`):

```
FROM python:3.9
WORKDIR /app
COPY . .
RUN pip install flask
EXPOSE 5000
```

```
ENV SECRET_MESSAGE={{ secret_message }}
CMD ["python", "app.py"]
```

Создай приложение (`app.py`):

```
from flask import Flask
import os

app = Flask(__name__)

@app.route('/')
def hello():
    secret = os.getenv('SECRET_MESSAGE', 'no secret provided')
    return f"Secret message: {secret}"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

✅**Чек-лист:**

-

⚙️5. Настройка HAProxy

Создай шаблон конфигурации (`templates/haproxy/haproxy.cfg.j2`):

```
global
  log /dev/log local0
  maxconn 2000

defaults
  mode http
  timeout connect 5000
  timeout client 50000
  timeout server 50000

frontend http_front
  bind *:{{ haproxy_frontend_port }}
  stats uri /haproxy?stats
  default_backend http_back

backend http_back
  balance roundrobin
  server python_app localhost:{{ app_ports.python }} check
```

Создай роль `haproxy`:

```
- name: Настройка HAProxy
  template:
    src: ../../templates/haproxy/haproxy.cfg.j2
    dest: /etc/haproxy/haproxy.cfg
  notify: reload haproxy

- name: Перезагрузка HAProxy
  service:
    name: haproxy
    state: reloaded
```

Запусти плейбук с тегами:

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

✅**Чек-лист:**

-

🕐6. Главный плейбук

Заполни `playbook.yml`:

```
- hosts: all
  become: true
  vars_files:
    - group_vars/all.yml
    - group_vars/vault.yml
  roles:
    - common
    - python_app
    - haproxy
  handlers:
    - name: reload haproxy
      service:
        name: haproxy
        state: reloaded
```

Запусти весь плейбук:

```
ansible-playbook -i inventory.ini playbook.yml --ask-vault-pass
```

✅**Чек-лист:**

-

🔎7. Проверка работы

Проверь работу приложения:

```
curl http://<SERVER_IP>/
```

Проверь работу HAProxy:

```
curl http://<SERVER_IP>/haproxy?stats
```

Убедись, что секретное сообщение отображается на веб-странице.

✅**Чек-лист:**

-

‼️ **Поздравляю! Ты успешно автоматизировал развёртывание приложений и научился безопасно управлять секретами с помощью Ansible!**