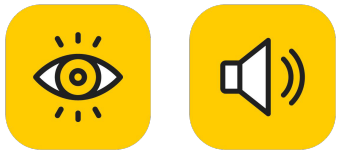
OTUS

Python QA Engineer

otus.ru

Меня хорошо видно & слышно?



Защита проекта
Тема: Проект по тестированию VPN

Седреева Галина

Должность Инженер

Компания "ООО С-Терра Си Эс Пи"

Цели проекта

1. Выполнять автоматическое тестирование с отчетностью в Allure.

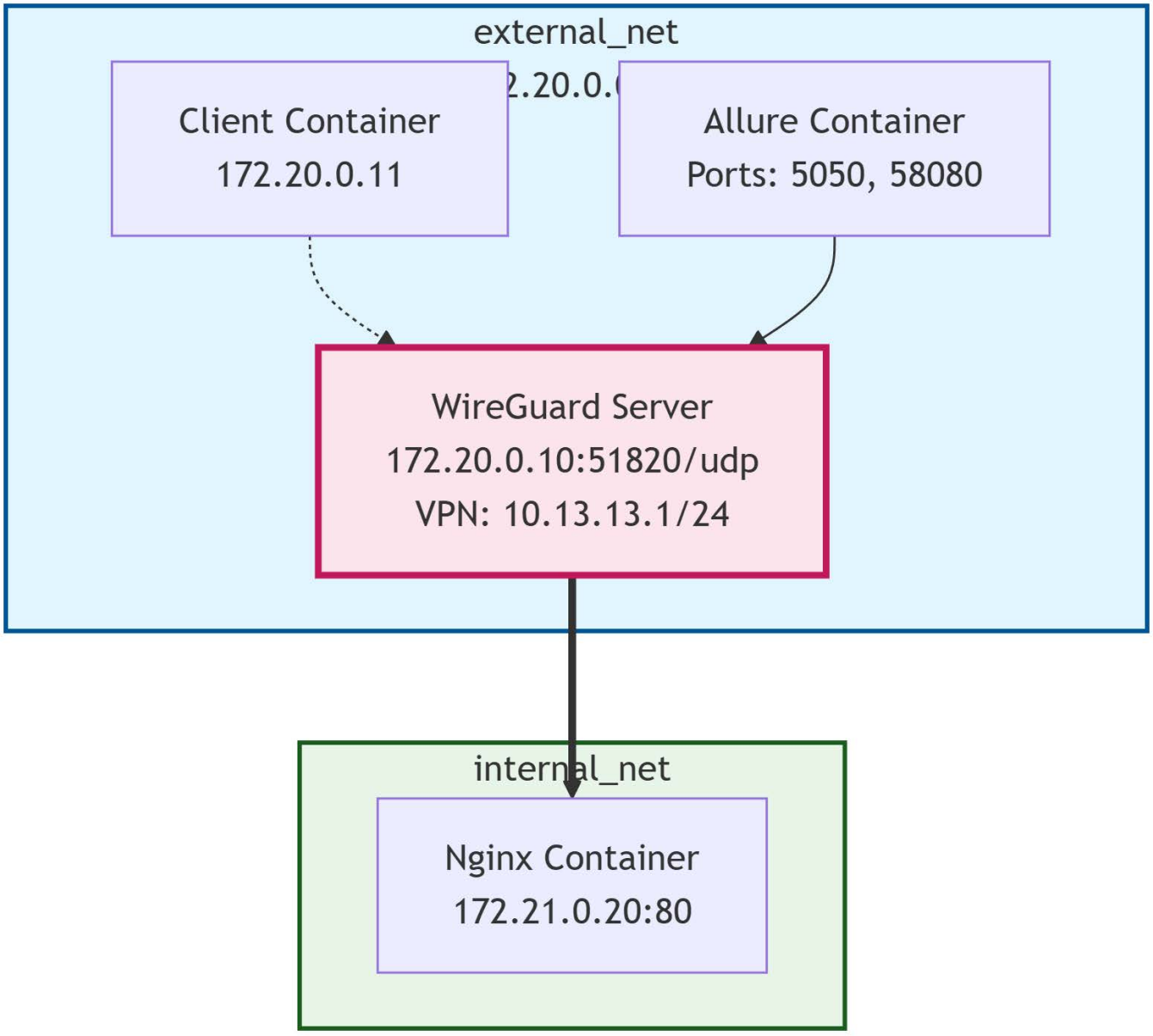
2. Этапы выполнения проекта:
    1. собрать стенд из docker-контейнеров и сетей между ними
    2. настроить в них VPN, SSH и nginx
    3. написать тесты
    4. настроить отчетность в Allure

3. Настройка серверов и клиентов происходила автоматически, безучастия инженера, в процессе развёртывания стенда, одной командой.

Структура проекта

```
project/
├── docker-compose.yml              # Файл для разворачивания стенда
├── Dockerfile.client               # Определяет образ клиента для тестов
├── Dockerfile.wireguard            # Определяет образ сервера WireGuard
├── nginx/
│   ├── default.conf                # Конфигурация Nginx
│   └── index.html                  # Простая страница для тестирования доступности
├── wireguard/
│   ├── server/
│   │   └── peer1/
│   │       └── peer1.conf          # Конфигурация клиента (для подключения)
│   └── client/
│       └── wg0.conf                # Скопированный конфиг клиента (настройка после сборки)
├── tests/
│   ├── conftest.py                 # Общие фикстуры для тестов
│   └── test_vpn.py                 # Автотесты
├── .gitignore                      # Игнорируемые файлы - чтобы не коммитить результаты тестов и Allure-отчёты.
├── README.md                       # Этот файл
└── allure-results/                 # Результаты тестов
└── allure-report/                  # Allure-отчёты
```

Схема стенда



Используемые технологии

- **Docker Compose** — для разворачивания стенда.
- **WireGuard** — для организации VPN.
- **Nginx** — в роли тестового сервиса в защищённой сети.
- **Python / Pytest** — для написания автотестов.
- **Paramiko** — для удалённого выполнения команд на сервере.
- **Allure** — для генерации HTML-отчётов по результатам тестирования

Как работает стенд

1. `docker-compose.yml` создаёт контейнеры:
   - `wireguard-server`: сервер WireGuard + Nginx + SSH.
   - `client-test`: клиент для тестов (на базе `Dockerfile.client`).
   - `nginx` (если не входит в `wireguard-server`): веб-сервер, доступный только через `internal_net`.
2. При старте:
   - Сервер WireGuard настраивается с `INTERNAL_SUBNET=10.13.13.0/24`.
   - Клиент получает IP из этой подсети при подключении.
   - Веб-сервер (`Nginx`) слушает на `10.13.13.1:80` (внутри `internal_net`).
3. **Тесты**:
   - Сначала проверяют доступность `http://10.13.13.1` **без VPN** → должна быть ошибка (например, `Connection refused`).
   - Затем подключаются к VPN через `wg quick up ...`.
   - После подключения проверяют `http://10.13.13.1` → должно быть `200 OK`.

Dockerfile.client > FROM

```
FROM linuxserver/wireguard:latest

ENV \
PUID=1000 \
PGID=1000 \
TZ=Europe/Moscow \
VPN_PROTOCOL=wireguard

RUN <<EOF
apk add --no-cache python3 py3-pip py3-virtualenv openssh-client curl wireguard-tools &&
virtualenv /venv &&
source /venv/bin/activate &&
pip3 install pytest allure-pytest paramiko requests
EOF
```

Dockerfile.wireguard > ...

```
FROM linuxserver/wireguard:latest

ENV \
PUID=1000 \
PGID=1000 \
TZ=Europe/Moscow \
SERVERURL=wireguard \
SERVERPORT=51820 \
PEERS=1 \
PEERDNS=auto \
INTERNAL_SUBNET=10.13.13.0/24 \
ALLOWEDIPS=10.13.13.0/24,172.21.0.0/24 \
LOG_CONFS=true

RUN <<EOF
apk add --no-cache openssh &&
ssh-keygen -A &&
mkdir -p /root/.ssh &&
sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config &&
echo 'root:123' | chpasswd
EOF
```

Фикстура

```python
@pytest.fixture(scope="function")
def set_vpn_connection(request):
    """Фикстура для установки нужного состояния VPN перед тестом."""
    mode = request.param
    wg_config_path = "/etc/wireguard/wg0.conf"

    # Убедимся, что конфиг существует
    if not os.path.exists(wg_config_path):
        pytest.fail(f"Конфиг WireGuard не найден по пути {wg_config_path}")

    def is_wireguard_active():
        result = subprocess.run(["ip", "link", "show", "wg0"],
                                stdout=subprocess.PIPE,
                                stderr=subprocess.PIPE)
        return result.returncode == 0

    # Очищаем текущее состояние
    if is_wireguard_active():
        subprocess.run(["wg-quick", "down", "wg0"], check=False)

    # Поднимаем соединение с сервером
    if mode == "enabled":
        subprocess.run(["chmod",  "600", "/config/wg_confs/wg0.conf", wg_config_path], check=True)
        subprocess.run(["wg-quick", "up", wg_config_path], check=True)
        time.sleep(2)  # даём время на установку соединения

    yield mode

    # Чистка после теста

    if is_wireguard_active():
        subprocess.run(["wg-quick", "down", "wg0"], check=False)
```

Примеры тестов

```python
@allure.description("Тест 1: Проверяем, что без VPN web-сервер недоступен")
@pytest.mark.parametrize("set_vpn_connection", ["disabled"], indirect=True)
@allure.title("Проверка недоступности веб-сервера без VPN")
@allure.feature("Проверка доступности")
def test_webserver_unreachable_without_vpn(set_vpn_connection):

    with allure.step("Проверяем, что веб-сервер недоступен"):
        with pytest.raises(requests.exceptions.RequestException) as exc_info:
            requests.get(WEB_SERVER_IP, timeout=R_TIMEOUT)

        exception = exc_info.value
        assert isinstance(exception, (
            requests.exceptions.ConnectionError,
            requests.exceptions.Timeout,
            requests.exceptions.RetryError
        )), f"Ожидалась ошибка подключения, получено: {type(exception).__name__}"


@allure.description("Тест 2: Запускаем VPN и проверяем, что клиент подключён")
@pytest.mark.parametrize("set_vpn_connection", ["enabled"], indirect=True)
@allure.title("Проверка подключения клиента через wg show")
@allure.feature("Соединение с VPN")
def test_vpn_client_connection(set_vpn_connection):

    with allure.step("Запрашиваем информацию о состоянии WireGuard (wg show)"):
        result = run_cmd("wg show")

    with allure.step("Проверяем успешное выполнение команды 'wg show'"):
        assert result.returncode == 0, f"Команда 'wg show' завершилась с ошибкой. STDERR: {result.stderr}"

    with allure.step("Проверяем наличие интерфейса wg0 и peer в выводе"):
        assert WG_INTERFACE in result.stdout, "Интерфейс wg0 не найден в выводе 'wg show'"
        assert "peer:" in result.stdout, "Поле 'peer:' отсутствует в выводе 'wg show'"
```