САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Практическая работа 6
Настройка Gitlab CI/Github actions для автоматического развёртывания Node.JS-приложения

Выполнила:
Едигарева Дарья

Группа
К3339

Проверил:
Добряков Д. И.

Санкт-Петербург

2022 г.

Задача

Необходимо настроить автодеплой (с триггером на обновление кода в вашем репозитории, на определённой ветке) для вашего приложения на удалённый сервер с использованием Github Actions или Gitlab CI (любая другая CI-система также может быть использована).

Ход работы

Краткое описание архитектуры автодеплоя:

Репозиторий GitHub: пуш/мердж в ветку main, срабатывает workflow.
Runner: self-hosted на удалённом сервере.

Оркестрация: docker compose поднимает инфраструктуру: Postgres/RabbitMQ, auth-service, profiles-service, vacancies-service, api gateway (nginx).

Деплой: checkout кода на сервер, сборка образов/перезапуск контейнеров через Docker Compose - health-проверки.

1. Подготовка сервера

SSH

Ключ добавлен на сервер. В ~/.ssh/config настроен алиас.

2. Установка Docker/Compose

sudo apt-get update

sudo apt-get install -y ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

echo "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \

$(. /etc/os-release && echo
"${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install -y docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin

Проверка:

docker --version

docker compose version

3. Создание пользователя и права

sudo adduser github-runner

sudo usermod -aG docker,sudo github-runner

sudo visudo

добавила строку:

github-runner ALL=(ALL) NOPASSWD:ALL

4. Установка self-hosted GitHub Actions Runner

Логинимся под github-runner и устанавливаем раннер:

su - github-runner

mkdir -p ~/actions-runner && cd ~/actions-runner

скачать актуальный релиз (x64 для Ubuntu)

curl -o actions-runner-linux-x64.tar.gz -L https://github.com/actions/runner/releases/latest/download/actions-runner-linux-x64-<VER>.tar.gz

tar xzf actions-runner-linux-x64.tar.gz

получить TOKEN на странице Settings → Actions → Runners → New self-hosted runner

./config.sh \

--url https://github.com/<ORG_OR_USER>/<REPO> \

--token <TOKEN> \

--name prod-runner \

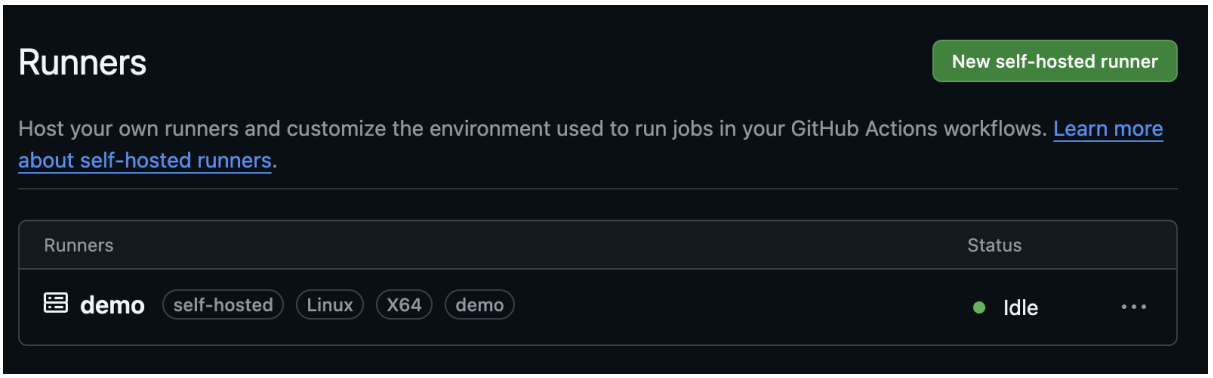
--labels self-hosted,prod \

--unattended

sudo ./svc.sh install

sudo ./svc.sh start

sudo ./svc.sh status



5. Подготовка репозитория

Структура:

homeworks/hw6/

├── docker-compose.micro.yml

├── gateway/nginx.conf

└── services/

├── auth-service/

├── profiles-service/

└── vacancies-service/

4.1. Docker Compose

Используется один compose-файл для всех сервисов; сеть/volumes создаю.

```yaml
services:
 db:
   image: postgres:17
   container_name: backend_hw2_db
   environment:
     POSTGRES_USER: postgres
     POSTGRES_PASSWORD: postgres
     POSTGRES_DB: postgres
   ports:
     - '5432:5432'
   volumes:
     - data:/var/lib/postgresql/data


 rabbitmq:
   image: rabbitmq:3.13-management
   container_name: message_broker
```

```
  environment:
    RABBITMQ_DEFAULT_USER: guest
    RABBITMQ_DEFAULT_PASS: guest
  ports:
    - '5672:5672'
    - '15672:15672'
  volumes:
    - rabbit-data:/var/lib/rabbitmq
  healthcheck:
    test: ["CMD", "rabbitmq-diagnostics", "status"]
    interval: 10s
    timeout: 10s
    retries: 6


auth-service:
  build:
    context: ./services/auth-service
    dockerfile: Dockerfile
  container_name: auth_service
  environment:
    NODE_ENV: production
    APP_HOST: 0.0.0.0
    APP_PORT: 8001
    APP_PROTOCOL: http
    DB_HOST: db
    DB_PORT: 5432
    DB_USER: postgres
    DB_PASSWORD: postgres
```

```
    DB_NAME: postgres
    JWT_SECRET_KEY: secret
    MESSAGE_BROKER_URL: amqp://rabbitmq:5672
  depends_on:
    db:
      condition: service_started
    rabbitmq:
      condition: service_healthy
  expose:
    - '8001'


vacancies-service:
  build:
    context: ./services/vacancies-service
    dockerfile: Dockerfile
  container_name: vacancies_service
  environment:
    NODE_ENV: production
    APP_HOST: 0.0.0.0
    APP_PORT: 8002
    APP_PROTOCOL: http
    DB_HOST: db
    DB_PORT: 5432
    DB_USER: postgres
    DB_PASSWORD: postgres
    DB_NAME: postgres
    JWT_SECRET_KEY: secret
    MESSAGE_BROKER_URL: amqp://rabbitmq:5672
```

```
  depends_on:
    db:
      condition: service_started
    rabbitmq:
      condition: service_healthy
  expose:
    - '8002'


profiles-service:
  build:
    context: ./services/profiles-service
    dockerfile: Dockerfile
  container_name: profiles_service
  environment:
    NODE_ENV: production
    APP_HOST: 0.0.0.0
    APP_PORT: 8003
    APP_PROTOCOL: http
    DB_HOST: db
    DB_PORT: 5432
    DB_USER: postgres
    DB_PASSWORD: postgres
    DB_NAME: postgres
    JWT_SECRET_KEY: secret
    MESSAGE_BROKER_URL: amqp://rabbitmq:5672
  depends_on:
    db:
      condition: service_started
```

```
    rabbitmq:
      condition: service_healthy
  expose:
    - '8003'


gateway:
  image: nginx:alpine
  container_name: api_gateway
  volumes:
    - ./gateway/nginx.conf:/etc/nginx/nginx.conf:ro
  depends_on:
    - auth-service
    - vacancies-service
    - profiles-service
  ports:
    - '8000:80'


volumes:
 data:
 rabbit-data:
```

4.Секреты/окружение

Файл.env хранится в репозитории. Пример переменных: DB_HOST, DB_USER, DB_PASSWORD, JWT_SECRET_KEY, NODE_ENV.

5. GitHub Actions — workflow автодеплоя

Файл: .github/workflows/deploy.yml

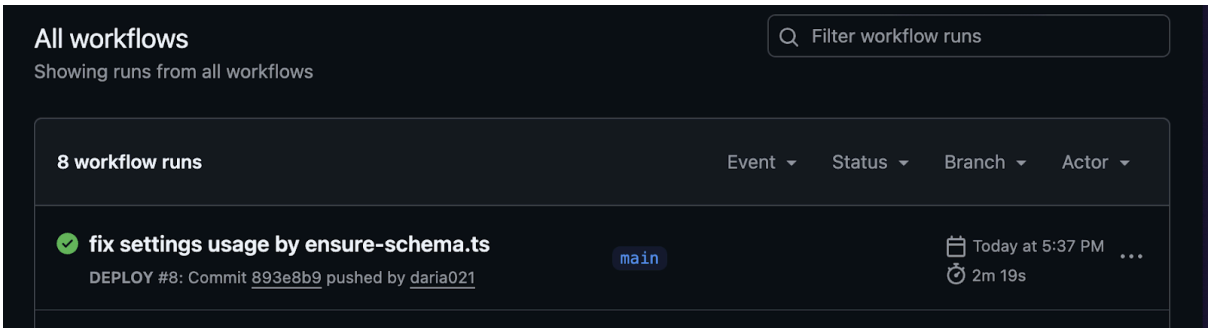
```
name: DEPLOY
```

```yaml
on:
 push:
   branches:
     - main
 workflow_dispatch:


jobs:
 deploy:
   name: Deploy all
   runs-on: [
     "self-hosted",
     "demo"
   ]
   defaults:
     run:
       working-directory: 'БР1.2/Едигарева Дарья/homeworks/hw6'
   steps:
     - name: Checkout
       uses: actions/checkout@v4


     - name: Compose up
       run: |
         sudo docker compose -f docker-compose.micro.yml stop
         sudo docker compose -f docker-compose.micro.yml up --build -d


     - name: Cleanup
       run: docker system prune -a -f
```

clean: false ускоряет деплой (сохраняется локальный кеш сборки Docker).

up -d --build --remove-orphans обновляет только изменившиеся сервисы.

Health-проверка фиксирует успешный деплой.



10. Выводы

Автодеплой настроен. Изменения в main автоматически собираются и выкатываются на сервер. Время деплоя — ~ 3 минуты.