**Проект инфраструктуры веб-приложения**

Компоненты системы:

- Демонстрационное веб-приложение, написанное на языке Python и фреймворке Flask;

- контейнеризация Docker;

- WSGi Gunicorn;

- веб-сервер Nginx;

- PostgreSQL;

- Системы мониторинга Prometheus, cAdvisor;

- CI/CD Jenkins;

Структура директорий веб-приложения:

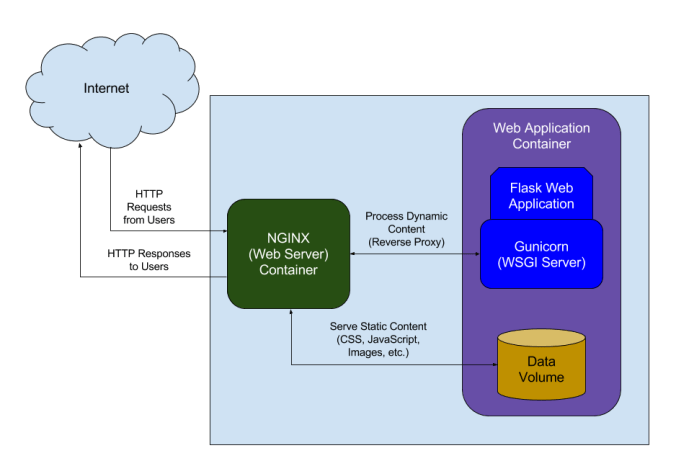
* unicorn/backend/
* unicorn /docker-compose.yml
* unicorn/env-postgresql
* unicorn/nginx
* unicorn/prometheus

Проект на GitHub: <https://github.com/NormanJean/unicorn.git>

В директории веб-приложения «backend/» располагается Dockerfile, содержащий инструкции по сборке образа. В текущей директории расположен файл зависимостей приложения (requirements.txt), который при запуске контейнера обновляется и устанавливает необходимые зависимости при сборке Flask-приложения.

Ввиду того, что приложение подразумевает использование баз данных PostgreSQL, балансировку нагрузки нескольких экземпляров, а также развертывание в продуктовой среде, наилучшим решением является контейнеризация каждого компонента на платформе Docker. Это обеспечит унификацию сборки с последующим развертыванием в другом окружении, обеспечит экономию аппаратных ресурсов и единый мониторинг как контейнера, так и каждого компонента отдельно.

Далее изображена схема, описывающая порядок взаимодействия всех компонентов контейнеризированного веб-приложения:



Файл «unicorn/docker-compose.yml» описывает процедуру оркестрации контейнеров Flasк-приложения.

Директива «flask:» собирает образ на основе Dockerfile из директории «backend/». Ввиду того, что веб-сервер Flask предназначен для разработки, в качестве WSGI будет использоваться сервер Gunicorn, прослушивающий стандартный для Flask-приложения порт (5000). Также, мы создадим 3 экземпляра приложения директивой «replicas:», для последующей балансировки при помощи Nginx.

Директива «nginx:» собирает образ на основе Dockerfile из директории «nginx/». В этой же директории расположен файл конфигурации веб-сервера. Nginx будет обрабатывать внешние запросы на 80-ый порт и проксировать их на Gunicorn. Помимо этого, в файле конфигурации мы укажем три экземпляра flasк-приложения с параметрами «max\_fails=3 fail\_timeout=5s». Это обеспечит балансировку между экземплярами а также проверку доступности (максимальное количество сбоев и время ожидания после сбоя).

Директива «postgres:» стартует контейнер с образом «postgres:14», принимая параметры с переменными из «env-file», разворачивая БД в директории «/var/lib/postgresql/data».

Директива «networks:» содержит описание docker-сетей для контейнеров. Все сети используют драйвер «bridge».

Для сбора логов с flask и nginx в формате «JSON», будем использовать стандартный драйвер платформы Docker Logging Driver c директивой «driver: json-file». Для пересылки логов в стек ELK планируется использовать Filebeat, ввиду своей лёгкости и возможности централизованного сбора данных журналов.

В качестве системы мониторинга будет использоваться Provetheus в связке с cAdvisor. Данная система также контейнеризирована, файлы «docker-compose.yml» и «prometheus.yml» расположены в директории «unicorn/prometheus/». После запуска контейнеров Prometheus и cAdvisor, метрики, собираемые cAdvisor пересылаются в Prometheus.

Некоторые запросы сбора метрик PromQL:

container\_network\_receive\_packets\_total{image=~" new-app-nginx "}

container\_network\_transmit\_packets\_total{image=~" new-app-nginx "}

- для отслеживание входящих и отправленных пакетов;

container\_memory\_usage\_bytes{image=~"postgres:14"}

container\_cpu\_usage\_seconds\_total{image=~"postgres:14"}

container\_memory\_usage\_bytes{image=~" new-app-flask "}

container\_cpu\_usage\_seconds\_total{image=~" new-app-flask "}

- для отслеживания нагрузки на ресурсы;

http\_requests\_total{image=~"postgres:14"}

http\_requests\_total{image=~"new-app-flask"}

http\_requests\_total{image=~"new-app-nginx"}

- общее количество запросов;

Также, по адресу <http://localhost:8090> можно отслеживать метрики от cAdvisor в режиме реального времени.

P.S. Вся инфраструктура собиралась на тестовом стенде, не всё удалось реализовать из-за нехватки времени и практического опыта. Многое пришлось изучить с нуля, по многим моментам есть сомнения, правильно ли я всё сделал. Т.к. «приложение» возвращает только «Привет, мир» в окно браузера и инфраструктура не имеет никакой нагрузки, есть сомнения насчет того, правильно ли я настроил балансировку, логирование, мониторинг. Но из того, что есть у меня в проекте всё стартануло.

В любом случае, это было интересно! Спасибо)