**GENERAL**

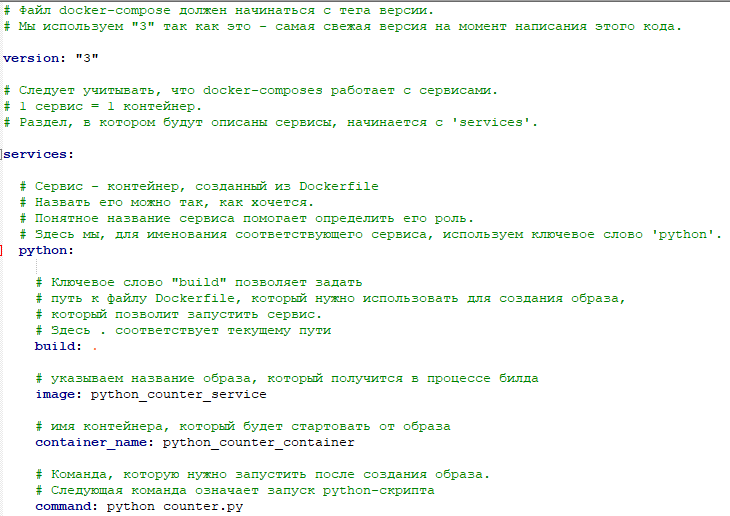
**Docker compose** упрощает развертывание нескольких контейнеров

Например, если нужно развернуть веб сервис на контейнерах, то нам потребуется 3 образа:

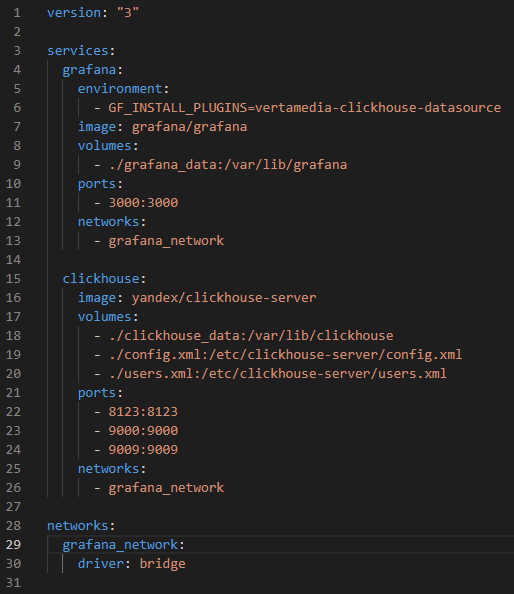
* Веб сервера
* БД
* Языка программирование

Каждый из них нужно запустить со своими настройками, а еще не забыть про сеть между ними

Файл который отвечает за настройку – **файл.yml**



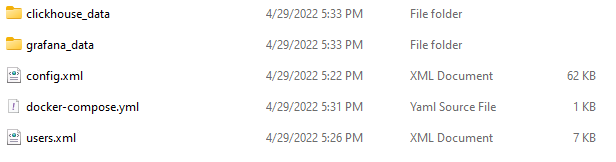
Yml для поднятия clickhouse + grafana с плагином для подключения clickhouse + добавление их в общую сеть + подключение volume для сохранения данных



**environment** *(LIST)* – по сути docker run -e something=something. Здесь мы указываем логин, пароль для подключения. Связь с БД, установка плагинов и тд

**volumes** *(LIST).* Связать папки в контейнере с папками на хосте. Что бы в случае выхода из контейнера и повторного захода, данные не удалялись

Поддерживает относительный путь, от нашего **docker-compose.yml** файла. Только нужно заранее создать там эти данные. В случае папок, то просто создаем их пустыми. А вот если нужны конфигурационные файлы, то нужно их заранее скачать. В последующем они будут просто изменяться. НЕЛЬЗЯ создать пустой файл и связать его, иначе ничего не заработает



**ports** *(LIST)* – по сути docker run -p. Мапим порты сервиса контейнера к хосту, что бы зайти по localhost:порт

**network** *(LIST)* – по сути docker run --net. Создаем общую сеть для контейнеров. После всех контейнеров нужно будет указать настройки сети. По дефолту, docker сам создает общую сеть между контейнерами, созданные в docker-compose. Но мы можем это сделать и сами

**depends\_on** *(LIST)*– запустит прописанные контейнеры до того, как запустит этот

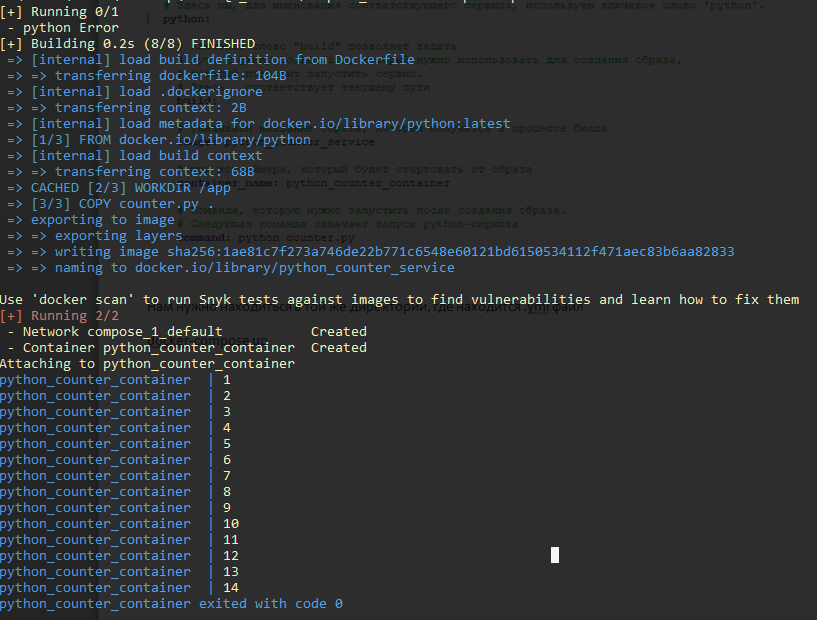
**ЗАПУСК**

Нам нужно находиться в той же директории, где находится .yml файл

**docker-compose up**

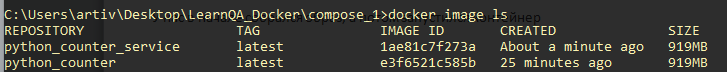
Если же в директории несколько .yaml файлов, то указываем явно

**docker-compose -f** [file\_name.yml] **up**



У нас в начале собрался образ, а потом запустился контейнер

Вот наглядно видно что у нас остался созданный образ – “**python\_counter\_service**”



А так же есть запущенный контейнер



Если мы внесем изменения в скрипт python и опять запустим команду docker-compose up, то внесенные изменения не отобразятся. Для того что бы изменения внеслись, то нужно указать ключ

docker-compose up --build

**docker-compose -f** [file\_name.yml] **down** – выключить все контейнеры, созданные через docker-compose

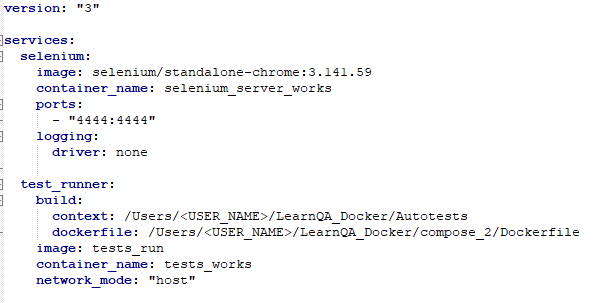
**docker-compose -f** [file\_name.yml] **start** – включить контейнеры, которые были созданы через docker-compose

docker-compose rm – удаляет все контейнеры, которые были созданы командой docker-compose up, но при этом не удаляет образы

docker-compose down --rmi all – удаляет все образы, контейнеры, тома которые были созданы с помощью команды docker-compose up

docker-compose up ps – посмотреть список контейнеров, которые создавались с помощью docker-compose up

.yml файл для поднятия 2 контейнеров (с тестами с Selenium Server) и настройки сети между ними



**Если не знаем что писать в docker-compose**

1. Идет в гугл и ищем “github docker compose [нужный нам образ]”
2. Там можно подсмотреть какие порты нужны, какие папки мапить и тд