GIT: https://github.com/samich80/docker\_python.git

Приложение Python в контейнере Docker со связью с каталогами хост-машины.

Для создания такой связи необходимо подготовить каталоги хаст-машины:

Создаем каталоги:

|  |  |
| --- | --- |
| /work | Рабочий каталог между приложением и контейнером Docker |
| /work/config | Каталог для файла конфигурации приложения в контейнере |
| /work/infiles | Каталог файлов для оработки |
| /work/outfiles | Каталог выходных файлов |

Создаем файл для сборки docker-контейнера:

|  |  |
| --- | --- |
| # Version: 0.0.1 | Информация о версии сборки |
| FROM ubuntu:18.04 | Указываем базовый дистрибутив |
| MAINTAINER user <user@email.com> | Информация о создателе сборки |
| RUN apt-get update  RUN apt-get upgrade -y  RUN apt-get install -y apt-utils | Обновляем apt-get внутри сборки |
| RUN apt-get install -y language-pack-ru | Устанавливаем поддержку кириллицы внутри сборки |
| RUN apt purge -y python2.7 | Удаляем старую версию Python |
| RUN apt-get install -y python3 | Устанавливаем новую версию Python |
| RUN apt install -y python3-pip | Устанавливаем менеджер пакетов Python |
| RUN ln -s /usr/bin/python3 /usr/bin/python  RUN ln -s /usr/bin/pip3 /usr/bin/pip | Создаем линки для правильной работы Python |
| ENV LANGUAGE ru\_RU.UTF-8  ENV LANG ru\_RU.UTF-8  ENV LC\_ALL ru\_RU.UTF-8  RUN locale-gen ru\_RU.UTF-8 && dpkg-reconfigure locales | Настройка locale для поддержки кириллицы UTF-8 |
| WORKDIR / | Устанавливаем рабочий каталог |
| RUN mkdir work  RUN mkdir program  RUN mkdir /work/infiles  RUN mkdir /work/outfiles  RUN mkdir /work/config | Создаем необходимые каталоги внутри сборки |
| VOLUME ["/work"] | Указываем «общий» каталог между хост-машиной и docker-контейнером |
| COPY program/Changer.py /program/ | Копируем скрипт Python внутрь сборки |

Тестовый контейнер с приложением.

Приложение следит за каталогом /work/infiles хост-машины. Если там появляется файл с именем input.txt, содержащий любой текст на кириллице. Обрабатывает его по правилам. Записанным в файле /work/config/config.conf

Обработка заключается в замене символов по словарю:

'й':'q','ц':'w','у':'e','к':'r','е':'t','н':'y','г':'u','ш':'i','щ':'o','з':'p','х':'[','ъ':']','ф':'a','ы':'s','в':'d','а':'f','п':'g','р':'h','о':'j','л':'k','д':'l','ж':';','э':'"','я':'z','ч':'x','с':'c', 'м':'v','и':'b','т':'n','ь':'m','б':',','ю':'.','ё':'`'

Создать словарь можно с помощью скрипта в util/MakeDict.py. заменив пары символов на свои. Затем запустить скрипт и скопировать файл config.conf в каталог config.

**Установка и запуск**.

Перейти в каталог с файлом Dockerfile.

Запустить *sudo docker build -t user/my\_python\_app /home/sysadmin/container/*

(если файлы расположены в /home/sysadmin/container/, иначе, заменить на свой путь к каталогу.

После создания контейнера, запустить его:

*sudo docker run -v /work:/work -d user/my\_python\_app*

Подключится к контейнеру можно командой:

*sudo docker exec -i -t user/my\_python\_app bash*

С учетом рекомендуемой структуры каталога, запустить скрипт можно командой:

*python /program/Changer.py*

Без входа в консоль, выполнить команду можно командой:

*docker exec -d user/my\_python python /program/Changer.py*

**Описание структуры работы и взаимодействия контейнеров.**

В рамках одно хост-сервера запущены и работают три контейнера:

1. Контейнер со скриптом Python
2. Контейнер с базой данных MySql
3. Контейнер со средством администрирования сервера MySql

С точки зрения безопасности сети, такое разделение контейнеров является оптимальным. Разнесены сервисы базы данных и средства администрирования базы данных.

Принцип работы такой схемы: запускается контейнер со скриптом Python, обрабатывает файлы, находящиеся на хост машине в директории /work/infiles/ . Обработанные файлы складывает в директорию /work/outfiles/ хост машины. Такой вариант работы обеспечивается связью между контейнером docker и файловой системой хост-машины. Далее, скрипт подключается к контейнеру с БД MySql, использую внутреннюю сеть docker, которая расположена в подсети 172.17.0.0/24 и записывает в БД имена обработанных файлов. По-умолчанию сеть работает в режиме bridge. К контейнеру можно обратиться двумя способами:

1. По псевдониму внутри docker-инфраструктуры
2. По IP адресу bridge 172.17.0.1 из приложений, работающих на хост машине.

И третий контейнер со средством управления базой данных.

**Реализация.**

Для осуществления описанных задач, последовательно подготовим хост-машину и контейнеры docker.

На хост-машине (используем Ubuntu в качестве операционной системы хост-машины):

1. Обновляем репозитории и добавляем репозиторий с docker:

*sudo apt-get update*

*sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common*

*curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -*

*sudo apt-key fingerprint 0EBFCD88*

*sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"*

*sudo apt-get update*

1. Устанавливаем docker:

*sudo apt-get install docker-ce docker-ce-cli containerd.io*

1. Обновляем docker:

*sudo apt-get update*

Создание контейнера со скриптом Python описана выше.

Установка контейнера с MySql:

*docker run --name mysql -e MYSQL\_ROOT\_PASSWORD=e1C403b178\* -d mysql*

параметр ***-e MYSQL\_ROOT\_PASSWORD*** устанавливает пароль для сервера MySql внутри контенера. ***–d mysql*** указывает запустить контейнер в режиме сервиса (в фоновом режиме) и задать ему внутреннее имя ***mysql*** по которому можно обратиться к контейнеру.

Установка средства администрирования сервера MySql.

*docker run --link mysql:db -d -p 8080:8080 adminer*

*--link mysql:db* связывает два контейнера (контейнер с БД Mysql и контейнер с администрированием) и указывает, что внутри контейнера мы можем использовать псевдоним ***db*** для связанного контейнера.

*-p 8080:8080* указывает, что контейнеру нужно использовать порт 8080 вне docker.

К *adminer* можно подключиться по адресу: <http://localhost:8080> внутри хост-машины или по адресу: <http://host:8080> с другой машины в локальной сети.