Настройка локального хранилища Docker Registry

date14.07.2021

user [Ivan](https://winitpro.ru/index.php/author/ivan/)

directory [DevOps](https://winitpro.ru/index.php/category/devops/), [Docker](https://winitpro.ru/index.php/category/virtualization/docker/), [Linux](https://winitpro.ru/index.php/category/linux/)

comments [комментариев 8](https://winitpro.ru/index.php/2021/03/03/nastrojka-lokalnogo-docker-registry/#comments)

В этой статье мы покажем, как настроить **Docker Registry** в качестве своего приватного репозиторий Docker. Docker Registry можно использовать для хранения и обмена образами docker. Отдельно мы покажем, как настроить аутентификацию для доступа к docker registry.

Давайте для начала разберем, из чего состоят docker image. Образ или image состоит из слоев, которые накладываются друг на друга. Когда мы собираем контейнер из образа, каждая команда создает новый слой, который накладывается на предыдущий. При запуске контейнера, все слои образа, на основе которого собран контейнер доступны на чтение, а сам слой контейнера на запись. На основе контейнера можно также создать новый image.

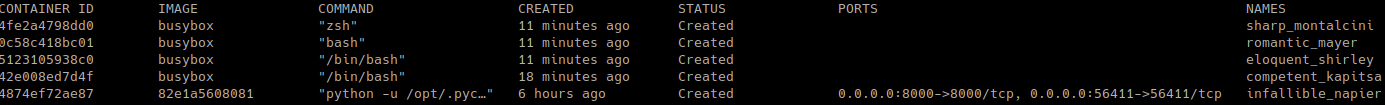
Для этого можно выполнить команду:

sudo docker commit <container-id>

Данной командой мы берем верхний уровень контейнера, тот, что доступен для записи и превращает его в слой для чтения.

Давайте попробуем запустить контейнер busybox, создать там файл и сделаем новый образ на основании данного контейнера.

1. Запустите контейнер на базе стандартного образа **busybox**и подключитесь в виртуальную консоль. Подробнее, основные команды, для работы с docker описаны в предыдущей статье: # sudo docker run -it busybox sh
2. Создайте файл внутри контейнера и выйдите из виртуальной консоли контейнера: # echo “test” >> 1.txt  
   # exit
3. Чтобы узнать id запущенного контейнера нужно выполнить: # sudo docker ps -a Вывод будет примерно такой:



На основе вывода можно получить id контейнера **4fe2a4798dd0.**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 4fe2a4798dd0 busybox "zsh" 11 minutes ago Created 0c58c418bc01 busybox "bash" 11 minutes ago Created 5123105938c0 busybox "/bin/bash" 11 minutes ago Created

1. Делаем коммит командой: # sudo docker commit 4fe2a4798dd0 mybusybox:v1

sha256:83ca3512880377a9fd03fd5cc80cbfd441b10ea71671ace0a72eeaa645128425

После id контейнера передается название нового image и тег.

В ответ команда вернет вам хеш созданного образа. Чтобы посмотреть информацию о образе, выполните команду:

# sudo docker image inspect 83ca3512880377a9fd03fd5cc80cbfd441b10ea71671ace0a72eeaa645128425

В выводе мы получим информацию о созданном образе.

[

{

"Id": "sha256:83ca3512880377a9fd03fd5cc80cbfd441b10ea71671ace0a72eeaa645128425",

"RepoTags": [

"mybusybox:v1"

],

"RepoDigests": [],

"Parent": "sha256:22667f53682a2920948d19c7133ab1c9c3f745805c14125859d20cede07f11f9",

"Comment": "",

"Created": "2021-02-16T18:29:58.408214348Z",

"Container": "4fe2a4798dd0dc3d9ae54f11db18dc8db9bf84f5d0ed3bdffc2a73eae9515186",

--------------

Вы можете просмотреть, какие команды были выполнены при сборке образа. Для это в консоли выполните команду:

sudo docker history mybusybox:v1

Вывод будет такой. На данном скриншоте видно, какие команды привели к созданию новых слоев в образе.

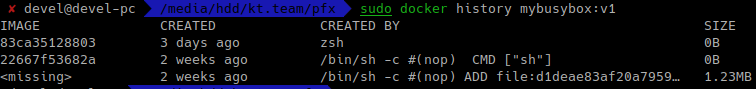


IMAGE CREATED CREATED BY SIZE

83ca35128803 3 days ago zsh 0B

22667f53682a 2 weeks ago /bin/sh -c #(nop) CMD ["sh"] 0B

<missing> 2 weeks ago /bin/sh -c #(nop) ADD file:d1deae83af20a7959… 1.23MB

Мы вкратце рассмотрели устройство образа docker. Данная информация будет полезна, для понимания механизмов работы docker registry.

Настройка и запуск Docker Registry

**Docker Registry** — это инструмент, хранения и обмена docker образами. Его образ можно скачать и запустить с официального репозитория **Docker Hub.**

Для запуска docker registry выполните в консоли:

# sudo docker run -d -p 5000:5000 --restart=always --name registry registry:2

Эта команда скачает **registry** и запустит контейнер на порту **5000** (ключ -p 5000:5000). Данный контейнер будет запускаться автоматически при старте docker.

В минимальной конфигурации мы запустили наш частный аналог **docker hub**. Давайте попробуем отправить свой образ в только что созданный репозиторий. Для это нам нужно протегировать образ. Возьмем для примера, образ, который мы создали [в прошлой статье](https://winitpro.ru/index.php/2021/02/17/sozdaem-prostoj-mikroservis-v-docker/). Либо можете использовать любой другой образ:

# sudo docker tag microsevice\_v1 <host ip>:5000/microsevice\_v1

Данной командой мы добавили тег для локального образа. Формат тега:

<hostname | ip >:post/<image name>:<tag>

В моем случае это будет:

# sudo docker tag microsevice\_v1 192.168.0.19:5000/microsevice\_v1

Можно указать **localhost:5000**, но тогда вы не сможем выполнить pull данного образа с другого компьютера в сети. Также можно указать hostname, если у вас настроен DNS.

После выполните:

# sudo docker push 192.168.0.19:5000/microsevice\_v1

Вывод консоли будет примерно таким:

The push refers to repository [192.168.0.19:5000/microsevice\_v1]

Get https://192.168.0.19:5000/v2/: http: server gave HTTP response to HTTPS client

Утилита docker ожидает, что registry работает по защищенному соединению HTTPS. Можно разрешить подключение по незащищенному протоколу, для в конфигурационный файл демона docker нужно добавить следующую строку:

# sudo nano /etc/docker/daemon.json

{

"insecure-registries": ["192.168.0.19:5000"]

}

После чего перезапустить демон docker.

# sudo systemctl restart docker

Еще раз попробуйте команду push.

Вывод будет примерно такой:



The push refers to repository [192.168.0.19:5000/microsevice\_v1]

e547e1a2483f: Layer already exists

efda78b6f2ad: Layer already exists

8c41c444e844: Layer already exists

042a80566ff4: Layer already exists

151a914ab9a4: Layer already exists

be340e26398b: Layer already exists

d7994f7c0aa0: Layer already exists

0bd71a837902: Layer already exists

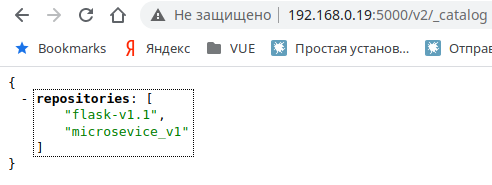
13cb14c2acd3: Layer already exists

latest: digest: sha256:0ae5b4cf9e0f6891b6b83ef62beeb6a97247fc9bd1057fba91476fea834b781e size: 2203

Ваш образ должен успешно отправился в ваш приватный репозиторий.

Чтобы посмотреть список images в репозитории можно воспользоваться curl или браузером:

# curl -X GET http://192.168.0.19:5000/v2/\_catalog



Теперь можно создать **Dockefile** и указать в качестве источника образа свой приватный репозиторий.

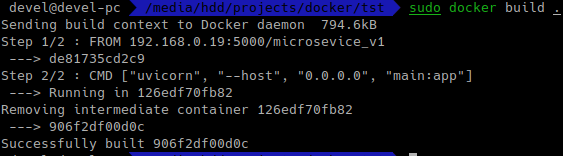
**Dockefile**

FROM 192.168.0.19:5000/microsevice\_v1

CMD ["uvicorn", "--host", "0.0.0.0", "main:app"]

Запустите сборку образа:

# sudo docker build .



Sending build context to Docker daemon 794.6kB

Step 1/2 : FROM 192.168.0.19:5000/microsevice\_v1

---> de81735cd2c9

Step 2/2 : CMD ["uvicorn", "--host", "0.0.0.0", "main:app"]

---> Running in 126edf70fb82

Removing intermediate container 126edf70fb82

---> 906f2df00d0c

Successfully built 906f2df00d0c

Видно, что образ подтянулся с локального репозитория и собрался контейнер.

То же самое можно сделать командой run, если вам нужно просто запустить контейнер, а не использовать его за основу.

sudo docker run -d 192.168.0.19:5000/microsevice\_v1

Данную команду можно выполнить на любом другом компьютере в сети, и тем самым мы обеспечили простой механизм переноса образов docker внутри сети. Docker Registry позволяет автоматизировать процесс deploy сервисов и их обмен при командной разработке.

Настройка авторизации в Docker Registry

Docker Registry поддерживает несколько способов аутентфикации, но в этой статье мы рассмотрим простую basic authentication.

Для начала установите утилиту, для генерации пароля:

sudo apt install apache2-utils

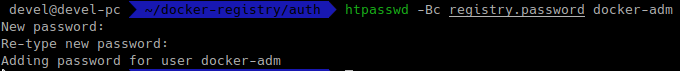
Затем создайте папку, для хранения настроек и Dockerfile.

# mkdir -p ~/docker-registry/auth/  
# cd ~/docker-registry/auth/

Для генерации пароля выполните:

# htpasswd -Bc registry.password docker-adm

Где **docker-adm**— это имя пользователя (его можно изменить на произвольное), ключ**-B** это принудительное шифрование пароля bcrypt, **-с** путь и имя создаваемого файла.

****

New password:

Re-type new password:

Adding password for user docker-adm

Утилита запросит пароль и подтверждение, после чего создаст файл

**registry.password.**

Перейдите на уровень выше в папку **~/docker-registry/**

# cd ~/docker-registry/

Запустите docker-registry со следующими переменными окружения:

docker run -d -p 5000:5000 -e REGISTRY\_AUTH=htpasswd -e REGISTRY\_AUTH\_HTPASSWD\_REALM=Registry -e REGISTRY\_AUTH\_HTPASSWD\_PATH=/auth/registry.password -e REGISTRY\_STORAGE\_FILESYSTEM\_ROOTDIRECTORY=/data -v "$PWD/data:/data" -v "$PWD/auth:/auth" --name registry registry:2

Ключи **-e** позволяют передать в контейнер переменные окружения(environment).

* **REGISTRY\_AUTH —**тип аутентификации, htpasswd;
* **REGISTRY\_AUTH\_HTPASSWD\_REALM —**это заголовок realm описанный в RFC 7235;
* **REGISTRY\_AUTH\_HTPASSWD\_PATH —**это путь до файла **registry.password**внутри контейнера, его не следует менять;
* **REGISTRY\_STORAGE\_FILESYSTEM\_ROOTDIRECTORY —**это путь до директории, где будет храниться образы внутри контейнера.

Команда длинная, лучше ее скопировать в текстовый файл, или в sh скрипт. Пример скрипта, можно взять [здесь](https://raw.githubusercontent.com/mesherinovivan/winitpro/docker-registry/start.sh)

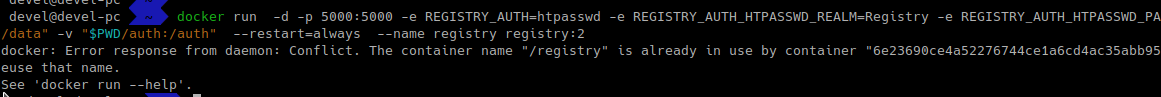
Ключ **-v** нужен, чтобы пробросить директории с host машины в контейнер. В нашем примере указано, что директория **«$PWD/auth»** — где $PWD это текущая директория, из которой производится запуск + **auth** будет смонтирована в контейнер в директорию **/auth.**Аналогично и второй ключ **-v.**

Вывод в консоль после выполнения команды:

**docker registry защита с помощью basic authentication**

Создался конейнер с hash: **6f430176572dfeca6e81be763206ea35d182facacd27ca9e732822b523d8906a**

Если вы видите следующую ошибку:



docker: Error response from daemon: Conflict. The container name "/registry" is already in use by container "6f430176572dfeca6e81be763206ea35d182facacd27ca9e732822b523d8906a". You have to remove (or rename) that container to be able to reuse that name.

See 'docker run --help'.

В данном случае контейнер уже запущен (мы его запустили в самом начале работы с docker registry). Можно остановить его и удалить контейнер командой:

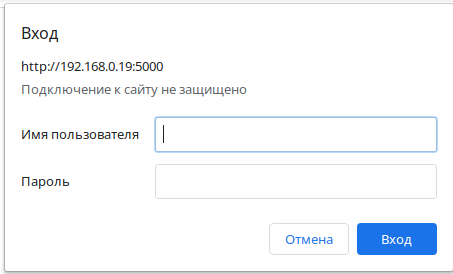
# sudo docker stop registry && sudo docker rm registry

Теперь можно повторить запуск registry.

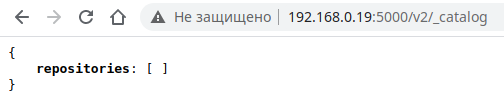
Для проверки аутентфикации, перейдите по адресу:

http://192.168.0.19:5000/v2/\_catalog

В окне нужно указать логин и пароль, который был указан при генерации файла **registry.password**.

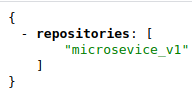


Если вы выполните аутентификацию, в окне браузера отобразится содержимое docker репозитория.



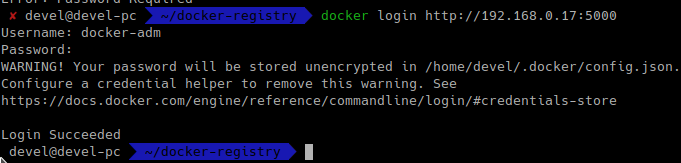
Наш репозиторий пуст.

Если выполнить команду push , указанную ранее, в registry появится один образ:



Но важно, теперь чтобы выполнить push в репозиторий, нужно сначала авторизоваться через консоль. Для этого нужно выполнить:

# sudo docker login login http://192.168.0.19:5000  
# sudo docker push 192.168.0.19:5000/microsevice\_v1



Username: docker-adm

Password:

WARNING! Your password will be stored unencrypted in /home/devel/.docker/config.json.

Configure a credential helper to remove this warning. See

https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded

В дальнейшем, мы будем использовать registry для автоматизации развертывания сервисов в [kubernetes](https://winitpro.ru/index.php/2021/07/14/ustanovka-i-nastrojka-klastera-kubernetes-k8s/).