

Технология контейнеризации. Введение в Docker



План

- Создание docker host
- Создание своего образа
- Работа с Docker Hub

Проект `microservices` и проверка ДЗ

Создайте новую ветку в вашем `microservices` репозитории в организации **DevOps 2019-05** для выполнения данного ДЗ. Ветку назовите **`docker-2`**.

В репозитории **должна быть настроена** интеграция с `travis-ci` по аналогии с репозиторием `infra`.

Проверка данного ДЗ будет производиться через Pull Request ветки с ДЗ.

После того, как один из преподавателей сделает approve пул реквеста, ветку с ДЗ можно смерджить.

Директория **docker-** **monolith**

- В новой ветке создайте директорию **docker-monolith**

Устанавливаем Docker

- Docker – 17.06+
- docker-compose – 1.14+
- docker-machine – 0.12.0+

Устанавливаем Docker

- Ubuntu Linux: <https://docs.docker.com/engine/installation/linux/docker-ce/ubuntu/>
- Mac OS: <https://download.docker.com/mac/stable/Docker.dmg>
- Windows: <https://download.docker.com/win/stable/Docker%20for%20Windows%20Installer.exe>

Устанавливаем Docker

- `docker version` – версии `docker client` и `server`
- `docker info` – информация о текущем состоянии `docker daemon`

Устанавливаем Docker

Все ок:

```
> docker version
```

Client:

Version: 17.06.2-ce

API version: 1.30

Go version: go1.8.3

Git commit: cec0b72

Built: Tue Sep 5 20:12:06 2017

OS/Arch: darwin/amd64

Server:

Version: 17.06.2-ce

API version: 1.30 (minimum version 1.12)

Go version: go1.8.3

Git commit: cec0b72

Built: Tue Sep 5 19:59:19 2017

OS/Arch: linux/amd64

Experimental: true

Docker hello-world

Запустим первый контейнер

```
> docker run hello-world
```

```
Unable to find image 'hello-world:latest' locally
```

```
latest: Pulling from library/hello-world
```

```
5b0f327be733: Pull complete
```

```
Digest: sha256:b2ba691d8aac9e5ac3644c0788e3d3823f9e97f757f01d2ddc6eb5458df9d801
```

```
Status: Downloaded newer image for hello-world:latest
```

```
Hello from Docker!
```

```
<skip>
```

Что произошло?

- docker client запросил у docker engine запуск container из image hello-world
- docker engine не нашел image hello-world локально и скачал его с Docker Hub
- docker engine создал и запустил container из image hello-world и передал docker client вывод stdout контейнера

Docker ps

Список запущенных контейнеров

```
>docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

Список всех контейнеров

```
>docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
7113ae217dc6	hello-world	"/hello"	2 seconds ago	Exited (0) 1 seconds ago		tiny_ramanujan

Docker images

Список сохраненных образов

```
>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	05a3bd381fc2	2 weeks ago	1.848 kB

Docker run

- Команда run создает и запускает **контейнер из image**

```
>docker run -it ubuntu:16.04 /bin/bash  
root@8d0234c50f77:/# echo 'Hello world!' > /tmp/file  
root@8d0234c50f77:/# exit
```

```
>docker run -it ubuntu:16.04 /bin/bash  
root@4e727649fb85:/# cat /tmp/file  
cat: /tmp/file: No such file or directory  
root@4e727649fb85:/# exit
```

Docker run

- Docker run каждый раз запускает **НОВЫЙ** контейнер
- Если не указывать флаг --rm при запуске docker run, то после остановки контейнер вместе с содержимым остается на диске

Docker ps

- Найдем ранее созданный контейнер в котором мы создали /tmp/file
- Это должен быть предпоследний контейнер запущенный из образа ubuntu:16.04

```
>docker ps -a --format "table {{.ID}}\t{{.Image}}\t{{.CreatedAt}}\t{{.Names}}"
```

CONTAINER ID	IMAGE	CREATED AT	NAMES
4e727649fb85	ubuntu:16.04	2016-11-16 16:23:21 +0300 MSK	stupefied_montalcini
8d0234c50f77	ubuntu:16.04	2016-11-16 16:22:39 +0300 MSK	drunk_murdock
7113ae217dc6	hello-world	2016-11-16 16:02:51 +0300 MSK	tiny_ramanujan

- CONTAINER ID у всех разный, поэтому вместо **8d0234c50f77** далее будем писать <u_container_id>

Docker start & attach

- start запускает остановленный(уже созданный) **контейнер**
- attach подсоединяет терминал к созданному контейнеру

```
> docker start <u_container_id>  
> docker attach <u_container_id>  
ENTER
```

```
root@<u_container_id>:/#  
root@<u_container_id>:/# cat /tmp/file  
Hello world!
```

Ctrl + p, Ctrl + q

```
> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
<u_container_id>	ubuntu:16.04	"/bin/bash"	9 minutes ago	Up 9 minutes		drunk_murdock

Docker run vs start

- `docker run` = `docker create` + `docker start` + `docker attach`*
- `docker create` используется, когда не нужно стартовать контейнер сразу
- в большинстве случаев используется `docker run`

* при наличии опции `-i`

Docker run

- Через параметры передаются лимиты(cpu/mem/disk), ip, volumes
- -i – запускает контейнер в foreground режиме (docker attach)
- -d – запускает контейнер в background режиме
- -t создает TTY
- `docker run -it ubuntu:16.04 bash`
- `docker run -dt nginx:latest`

Docker exec

- Запускает новый процесс внутри контейнера
- Например, `bash` внутри контейнера с приложением

Docker exec

```
>docker exec -it <u_container_id> bash
```

```
root@<u_container_id>:/#
```

```
ps axf
```

PID	TTY	STAT	TIME	COMMAND
-----	-----	------	------	---------

12	?	Ss	0:00	bash
----	---	----	------	------

22	?	R+	0:00	_ ps axf
----	---	----	------	-----------

1	?	Ss+	0:00	/bin/bash
---	---	-----	------	-----------

```
root@<u_container_id>:/# exit
```

Docker commit

- Создает image из контейнера
- Контейнер при этом остается запущенным

Docker commit

```
> docker commit <u_container_id> yourname/ubuntu-tmp-file  
sha256:c9b7e0f6b390a8c964bc4af7736e7f1015f4ce8da8648d95d1b88917742c8773
```

```
> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
yourname/ubuntu-tmp-file	latest	c9b7e0f6b390	3 seconds ago	122MB

- **Для сдачи домашнего задания**, необходимо сохранить вывод команды `docker images` в файл **docker-monolith/docker-1.log** и закоммитить в репозиторий
- **IMAGE ID** у всех будет разный, поэтому вместо **c9b7e0f6b390** далее будем писать **<u_image_id>**

Задание со *

Сравните вывод двух следующих команд

```
>docker inspect <u_container_id>
```

```
>docker inspect <u_image_id>
```

На основе вывода команд объясните чем отличается контейнер от образа.

Объяснение допишите в файл **docker-monolith/docker-1.log**

Docker kill & stop

- kill сразу посылает SIGKILL
- stop посылает SIGTERM, и через 10 секунд(настраивается) посылает SIGKILL
- SIGTERM - сигнал остановки приложения
- SIGKILL - безусловное завершение процесса
- Подробнее про сигналы в Linux

Docker kill

```
>docker ps -q
```

```
8d0234c50f77
```

```
>docker kill $(docker ps -q)
```

```
8d0234c50f77
```

docker system df

- Отображает сколько дискового пространства занято образами, контейнерами и volume'ами
- Отображает сколько из них не используется и возможно удалить

docker system df

```
> docker system df
```

TYPE	TOTAL	ACTIVE	SIZE	RECLAIMABLE
Images	7	5	6.481GB	347.3MB (5%)
Containers	5	1	176.3MB	176.3MB (99%)
Local Volumes	0	0	0B	0B

Docker rm & rmi

- rm удаляет **контейнер**, можно добавить флаг -f, чтобы удалялся работающий container(будет послан sigkill)
- rmi удаляет **image**, если от него не зависят запущенные контейнеры

Docker rm & rmi

>docker rm \$(docker ps -a -q) # удалит все незапущенные контейнеры

4e727649fb85

8d0234c50f77

7113ae217dc6

>docker rmi \$(docker images -q)

Untagged: <your-login>/docker-mk-01:first

Untagged: <your-login>/docker-

mk-01@sha256:f733255c83330f7d5897293736ab9b3c3edc7268a501ecc1c6ed8c33646e4b58

Deleted: sha256:c5a8113a357e7c62c42169b2758e42f0d2c48775c568da0202e9f366b120bbe1

mk-01@sha256:adabd21101c54f318d2539c8df746713114c84ee2cc9314492f472774ac31ac3

Deleted: sha256:a26e5bf03f515e83827e7f4c5b54f2644e06613a969395ee2b4d37555f5959b3

Deleted: sha256:47146b5be28e33939eda699625215d5c2e8c5a2510e6c7bf66c9b7a09fd4ba35

<skip>

Docker- контейнеры





- Создайте новый проект
<https://console.cloud.google.com/compute>
- Назовите его docker
- Запомните ID вашего проекта

gcloud

- Установите GCloud SDK
- <https://cloud.google.com/sdk/>

gcloud init

```
$ gcloud init
```

```
You must log in to continue. Would you like to log in (Y/n)? Y
```

```
Your browser has been opened to visit:
```

```
https://accounts.google.com/o/oauth2/....
```

У вас должен запуститься браузер с выбором учетных записей Google. Выберите ту, к которой привязан GCE и разрешите доступ приложению

Если браузер не запустился, то запустите его самостоятельно и перейдите по ссылке выведенной в консоли

gcloud init

...

Pick cloud project to use:

[1] docker-181710

[2] Create a new project

Please enter numeric choice or text value (must exactly match list item): 1

...

Do you want to configure Google Compute Engine
(<https://cloud.google.com/compute>) settings (Y/n)? n

...

Мы сконфигурировали gcloud

gcloud auth

```
$ gcloud auth application-default login
```

Your browser has been opened to visit:

<https://accounts.google.com/o/oauth2/....>

У вас должен запуститься браузер с выбором учетных записей Google. Выберите ту, к которой привязан GCE и разрешите доступ приложению

Если браузер не запустился, то запустите его самостоятельно и перейдите по ссылке выведенной в консоли

gcloud auth

В итоге мы получили файл с аутентификационными данными.

Он будет использоваться docker-machine для работы с облаком.

Docker machine


- Mac OS и Windows
идет в комплекте с docker (docker-machine -v)
- Linux
<https://docs.docker.com/machine/install-machine/>
- docker-machine - встроенный в докер инструмент для создания хостов и установки на них docker engine. Имеет поддержку облаков и систем виртуализации (Virtualbox, GCP и др.)
- Команда создания - docker-machine create <имя>. Имен может быть много, переключение между ними через eval \$(docker-machine env <имя>). Переключение на локальный докер - eval \$(docker-machine env --unset). Удаление - docker-machine rm <имя>.
- docker-machine создает хост для докер демона со указываемым образом в --google-machine-image, в ДЗ используется ubuntu-16.04. Образы которые используются для построения докер контейнеров к этому никак не относятся.
- Все докер команды, которые запускаются в той же консоли после eval \$(docker-machine env <имя>) работают с удаленным докер демоном в GCP.

Docker machine

```
$ export GOOGLE_PROJECT=_ваш-проект_
```

```
$ docker-machine create --driver google \  
  --google-machine-image https://www.googleapis.com/compute/v1/projects/ubuntu-  
os-cloud/global/images/family/ubuntu-1604-lts \  
  --google-machine-type n1-standard-1 \  
  --google-zone europe-west1-b \  
  docker-host
```

Регион можно поменять на
другой, если этот блокируется



...

Docker is up and running!

To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: `docker-machine env docker-host`

Docker machine

Проверяем, что наш Docker-хост успешно создан...

```
$ docker-machine ls
```

NAME	ACTIVE	DRIVER	STATE	URL	SWARM	DOCKER
docker-host	-	google	Running	tcp://35.195.54.250:2376		v17.09.0-ce

```
$ eval $(docker-machine env docker-host)
```

... и начинаем с ним работу

Повторение практики из демо на лекции

Теперь когда у вас запущен докер хост в GCP, можете самостоятельно повторить демо из лекции посвященные:

- PID namespace (изоляция процессов)
- net namespace (изоляция сети)
- user namespaces (изоляция пользователей)

P.S. Для реализации Docker-in-Docker можно использовать этот образ. Дока по user namespace.

Повторение практики из демо на лекции

И сравните сами вывод:

- `docker run --rm -ti tehbilly/htop`
- `docker run --rm --pid host -ti tehbilly/htop`

Можете добавить небольшое описание в
README

Структура репозитория

Для дальнейшей работы нам потребуются четыре файла, их содержание вы найдете на следующих слайдах

- Dockerfile - текстовое описание нашего образа
- mongod.conf - подготовленный конфиг для mongodb
- db_config - содержит переменную окружения со ссылкой на mongodb
- start.sh - скрипт запуска приложения

Вся работа происходит в папке **docker-monolith**



mongodb.conf

Where and how to store data.

[ссылка на gist](#)

storage:

dbPath: /var/lib/mongodb

journal:

enabled: true

where to write logging data.

systemLog:

destination: file

logAppend: true

path: /var/log/mongodb/mongod.log

network interfaces

net:

port: 27017

bindIp: 127.0.0.1

start.sh

[ссылка на gist](#)

```
#!/bin/bash
```

```
/usr/bin/mongod --fork --logpath /var/log/mongod.log --config /etc/  
mongodb.conf
```

```
source /reddit/db_config
```

```
cd /reddit && puma || exit
```

db_config

`DATABASE_URL=127.0.0.1`

Dockerfile

- Начнем создавать образ с приложением. За основу возьмем известный нам дистрибутив ubuntu версии 16.04
- Создадим файл "Dockerfile" и добавим в него строки:

```
$ vim Dockerfile  
FROM ubuntu:16.04
```
- Вместо vim можете использовать ваш редактор

Dockerfile

- Для работы приложения нам нужны mongo и ruby.
- Обновим кеш репозитория и установим нужные пакеты
- Добавим в "Dockerfile" строки:

FROM ubuntu:16.04

RUN apt-get update

RUN apt-get install -y mongodb-server ruby-full ruby-dev build-essential git

RUN gem install bundler

Dockerfile

Скачаем наше приложение в контейнер:

FROM ubuntu:16.04

RUN apt-get update

RUN apt-get install -y mongodb-server ruby-full ruby-dev build-essential git

RUN gem install bundler

RUN git clone -b monolith <https://github.com/express42/reddit.git>

Dockerfile

Скопируем файлы конфигурации в контейнер:

```
FROM ubuntu:16.04
```

```
RUN apt-get update
```

```
RUN apt-get install -y mongodb-server ruby-full ruby-dev build-essential git
```

```
RUN gem install bundler
```

```
RUN git clone -b monolith https://github.com/express42/reddit.git
```

```
COPY mongod.conf /etc/mongod.conf
```

```
COPY db_config /reddit/db_config
```

```
COPY start.sh /start.sh
```

Dockerfile

Теперь нам нужно установить зависимости приложения и произвести настройку:

```
FROM ubuntu:16.04
```

```
RUN apt-get update
```

```
RUN apt-get install -y mongodb-server ruby-full ruby-dev build-essential git
```

```
RUN gem install bundler
```

```
RUN git clone -b monolith https://github.com/express42/reddit.git
```

```
COPY mongod.conf /etc/mongod.conf
```

```
COPY db_config /reddit/db_config
```

```
COPY start.sh /start.sh
```

```
RUN cd /reddit && bundle install
```

```
RUN chmod 0777 /start.sh
```

Dockerfile

Добавляем старт сервиса при старте контейнера:

[ссылка на gist](#)

```
FROM ubuntu:16.04
```

```
RUN apt-get update
```

```
RUN apt-get install -y mongodb-server ruby-full ruby-dev build-essential git
```

```
RUN gem install bundler
```

```
RUN git clone -b monolith https://github.com/express42/reddit.git
```

```
COPY mongod.conf /etc/mongod.conf
```

```
COPY db_config /reddit/db_config
```

```
COPY start.sh /start.sh
```

```
RUN cd /reddit && bundle install
```

```
RUN chmod 0777 /start.sh
```

```
CMD ["/start.sh"]
```

Сборка образа

Теперь мы готовы собрать свой образ

- Выполните команду:

пример лога

```
$ docker build -t reddit:latest .  
Sending build context to Docker daemon 36.86kB  
Step 1/12 : FROM ubuntu:16.04  
16.04: Pulling from library/ubuntu  
...  
Successfully built ad21718d3eb5  
Successfully tagged reddit:latest
```

- Точка в конце обязательна, она указывает на путь до Docker-контекста
- Флаг -t задает тег для собранного образа

Сборка образа

Посмотрим на все образы (в том числе промежуточные):

```
$ docker images -a
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<none>	<none>	e0a3c979a0fb	2 minutes ago	687MB
<none>	<none>	0e07ff248072	2 minutes ago	687MB
reddit	latest	f7ff59db9594	2 minutes ago	687MB
<none>	<none>	96f7a69d48d2	2 minutes ago	687MB
<none>	<none>	a809efbf09a0	2 minutes ago	657MB
<none>	<none>	476d8d488857	2 minutes ago	657MB
<none>	<none>	b163309613e1	2 minutes ago	657MB
<none>	<none>	f7b02c42278f	2 minutes ago	657MB
<none>	<none>	746e7a1087d9	2 minutes ago	657MB
<none>	<none>	2d2469adc562	3 minutes ago	654MB
<none>	<none>	7ea520e2fdf1	6 minutes ago	161MB
ubuntu	16.04	2d696327ab2e	2 weeks ago	122MB

Запуск контейнера

- Отлично, теперь можно запустить наш контейнер командой:

```
$ docker run --name reddit -d --network=host reddit:latest
```

- Проверим результат:

```
$ docker-machine ls
```

NAME	ACTIVE DRIVER	STATE	URL	DOCKER
docker-host -	google	Running	tcp://_ваш_IP_:2376	v17.09.0-ce

- Откройте в браузере ссылку `http://_ваш_IP_адрес_:9292`

Ошибка



This site can't be reached

35.195.54.250 took too long to respond.

Search Google for [195 250 9292](#)

ERR_CONNECTION_TIMED_OUT

Настройка firewall'a

- Разрешим входящий TCP-трафик на порт 9292
выполнив команду:

```
$ gcloud compute firewall-rules create reddit-app \  
--allow tcp:9292 \  
--target-tags=docker-machine \  
--description="Allow PUMA connections" \  
--direction=INGRESS
```

- Попробуйте снова открыть ссылку (hint: должен открыться наш сервис) http://_ваш_IP_адрес_:9292

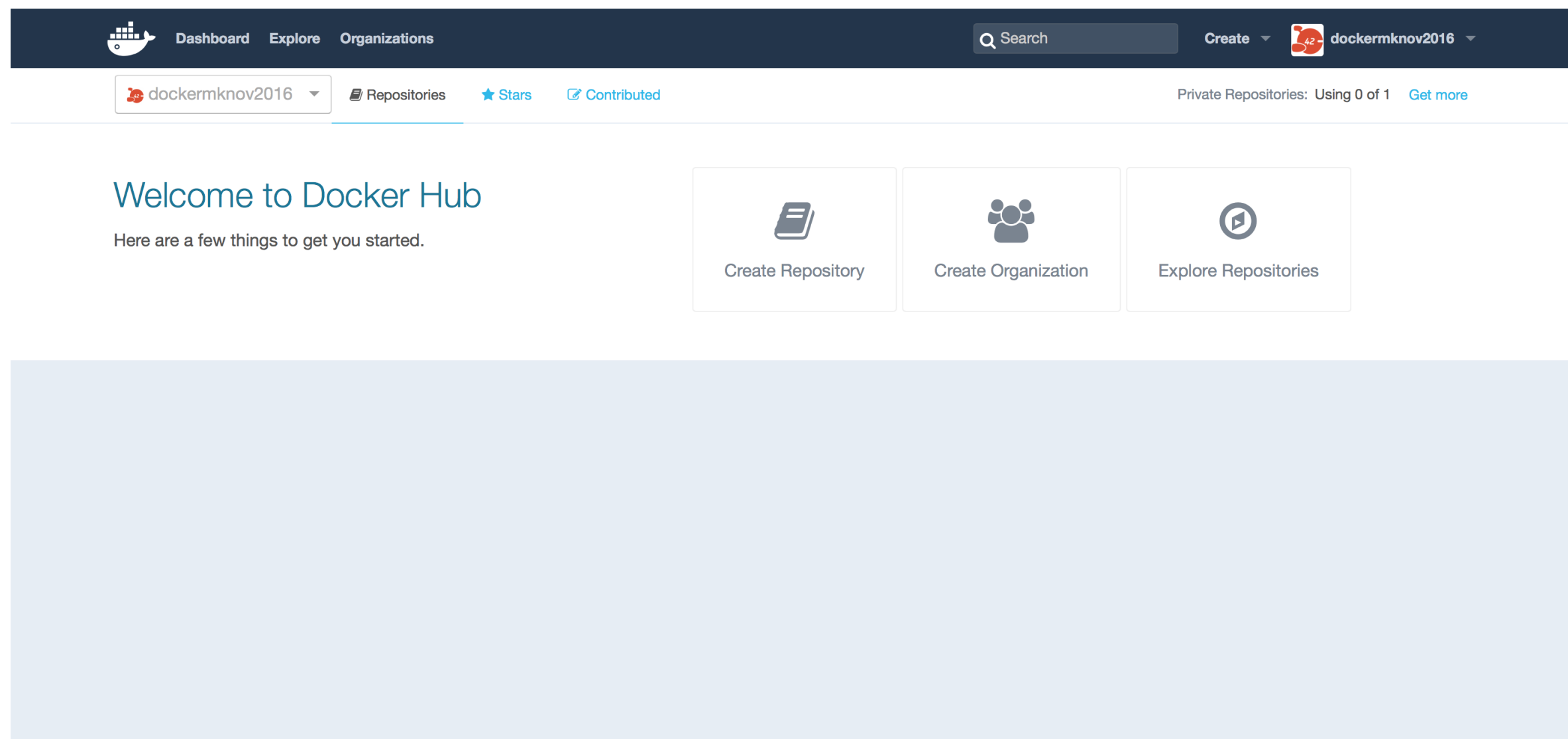
Docker hub: регистрация

Docker Hub - это облачный registry сервис от компании Docker. В него можно выгружать и загружать из него докер образы. Docker по умолчанию скачивает образы из докер хаба.

Зарегистрируемся в нем:

- Перейдите по ссылке <https://hub.docker.com/>
- Зарегистрируйте учетную запись, если у вас ее еще нет

Docker hub: регистрация



Docker hub: аутентификация

Аутентифицируемся на docker hub для продолжения работы:

```
$ docker login
```

Login with your Docker ID to push and pull images from Docker Hub.

If you don't have a Docker ID, head over to <https://hub.docker.com> to create one.

Username: **your-login**

Password:

Login Succeeded

Docker hub: push

Загрузим наш образ на docker hub для использования в будущем:

пример лога

```
$ docker tag reddit:latest <your-login>/otus-reddit:1.0
```

```
$ docker push <your-login>/otus-reddit:1.0
```

```
The push refers to a repository [docker.io/<your-login>/otus-reddit]
```

```
c6e5100de1e0: Pushed
```

```
...
```

```
a2022691bf95: Pushed
```

```
1.0: digest: sha256:77c6070400a5b04f8db3f7c129a2c16084c2fcf186aa6b436c8d6f57e0014378  
size: 3448
```

Проверка

Т.к. теперь наш образ есть в докер хабе, то мы можем запустить его не только в докер хосте в GCP, но и в вашем локальном докере или на другом хосте.

Выполним в другой консоли:

```
$ docker run --name reddit -d -p 9292:9292 <your-login>/otus-reddit:1.0
```

И проверим что в локальный докер скачался загруженный ранее образ и приложение работает.

Еще проверка

Дополнительно можете с помощью следующих команд изучить логи контейнера, зайти в выполняемый контейнер, посмотреть список процессов, вызвать остановку контейнера, запустить его повторно, остановить и удалить, запустить контейнер без запуска приложения и посмотреть процессы:

- `docker logs reddit -f`
- `docker exec -it reddit bash`
 - `ps aux`
 - `killall5 1`
- `docker start reddit`
- `docker stop reddit && docker rm reddit`
- `docker run --name reddit --rm -it <your-login>/otus-reddit:1.0 bash`
 - `ps aux`
 - `exit`

И еще проверка

И с помощью следующих команд можно посмотреть подробную информацию о образе, вывести только определенный фрагмент информации, запустить приложение и добавить/удалить папки и посмотреть дифф, проверить что после остановки и удаления контейнера никаких изменений не останется:

- `docker inspect <your-login>/otus-reddit:1.0`
- `docker inspect <your-login>/otus-reddit:1.0 -f '{{.ContainerConfig.Cmd}}'`
- `docker run --name reddit -d -p 9292:9292 <your-login>/otus-reddit:1.0`
- `docker exec -it reddit bash`
 - `mkdir /test1234`
 - `touch /test1234/testfile`
 - `rmdir /opt`
 - `exit`
- `docker diff reddit`
- `docker stop reddit && docker rm reddit`
- `docker run --name reddit --rm -it <your-login>/otus-reddit:1.0 bash`
 - `ls /`

Задание со *

Теперь, когда есть готовый образ с приложением, можно автоматизировать поднятие нескольких инстансов в GCP, установку на них докера и запуск там образа `<your-login>/otus-reddit:1.0`

Нужно реализовать в виде прототипа в директории **/docker-monolith/infra/**

- Поднятие инстансов с помощью Terraform, их количество задается переменной;
- Несколько плейбуков Ansible с использованием динамического инвентори для установки докера и запуска там образа приложения;
- Шаблон пакера, который делает образ с уже установленным Docker;

Проверка ДЗ

- Результаты вашей работы находятся в ветке **docker-2** вашего `microservices` репозитория.
- В README внесите описание того, что сделано.
- Создайте Pull Request к ветке `master` (описание PR нужно заполнять);
- В ревьюеры можно никого не добавлять;
- Добавьте "Labels" **docker** и **docker-2** к вашему Pull Request;
- После того, как один из преподавателей сделает approve пул реквеста, ветку с ДЗ можно смерджить и закрыть PR.