Создание собственного хранилища образов (container registry)

Что такое container registry?

Container registry — это каталогизированное хранилище, из которого можно извлекать образы контейнеров и куда их можно записывать. Образы группируются в репозитории — коллекции связанных образов с одинаковым названием. Например, в реестре от Docker Hub nginx представляет собой репозиторий, содержащий различные версии образов nginx.

Некоторые реестры являются публичными, то есть размещённые в них образы доступны любому пользователю интернета. Публичные реестры вроде Docker Hub отлично подходят для хостинга Open Source-проектов.

С другой стороны, приватные реестры обеспечивают безопасность и конфиденциальность для корпоративного хранилища образов вне зависимости от его расположения — в облаке или on-premise. Такие приватные реестры часто включают в себя расширенные функции безопасности и техническую поддержку. Выбор приватных реестров достаточно богат — Amazon ECR, GCP Artifact Registry, GitHub Container Registry. Docker Hub также поддерживает приватные репозитории.

Разработчики взаимодействуют с container registry с помощью команд `docker push` и `docker pull`:

```
docker push docker.io/pliutau/hello-world:v0


# В случае Docker Hub можно пропустить часть, относящуюся к registry.
docker push pliutau/hello-world:v0
```

Структура URL-адреса образа контейнера:

```
docker pull docker.io/pliutau/hello-world:v0@sha256:dc11b2...
                 ↓             ↓             ↓           ↓
```

```
            реестр      репозиторий     тег     дайджест
```

Зачем делать свой container registry?

Иногда вместо того, чтобы полагаться на готовое решение от провайдера (например, от AWS или GCP), разумно завести собственное хранилище образов. Плюсы такого подхода в том, что реестр остаётся полностью под вашим контролем — вы не зависите от сторонних вендоров. В некоторых регулируемых отраслях это обязательное требование.

Собственный реестр работает на ваших серверах. Вы можете контролировать его конфигурацию и место размещения образов контейнеров. В то же время это сопряжено с затратами на обслуживание и защиту.

Существуют различные решения с открытым исходным кодом для организации container registry. Наиболее популярное из них называется registry. Оно также упаковано в контейнер и позволяет хранить и распространять образы контейнеров и артефакты.

Чтобы запустить registry на сервере:

1. Установите Docker и Docker Compose на сервер.
2. Настройте и запустите контейнер с **registry**.
3. Запустите **nginx** для обработки TLS и перенаправления запросов в контейнер с **registry**.
4. Установите TLS-сертификаты и настройте домен.

Сервер

Можно воспользоваться любым сервером, который поддерживает Docker, например дроплетом DigitalOcean с Ubuntu. Для этой демонстрации я использовал виртуальную машину с Ubuntu в Google Compute Engine:

```
neofetch


# OS: Ubuntu 20.04.6 LTS x86_64
# CPU: Intel Xeon (2) @ 2.200GHz
# Memory: 3908MiB
```

Теперь в виртуальную машину нужно установить Docker и Docker Compose. Docker Compose необязателен, но упрощает управление мультиконтейнерными приложениями:

```
# Устанавливаем docker и docker-compose.
sudo snap install docker


# Проверяем правильность установки.
```

```
docker --version
docker-compose –version
```

Настройка container registry

Приведённый ниже файл **compose.yaml** создаёт контейнер реестра с томом для хранения образов и томом для хранения файла паролей.

```
services:
  registry:
    image: registry:latest
    environment:
      REGISTRY_AUTH: htpasswd
      REGISTRY_AUTH_HTPASSWD_REALM: Registry Realm
      REGISTRY_AUTH_HTPASSWD_PATH: /auth/registry.password
      REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY: /data
    volumes:
      # Монтируем файл паролей.
      - ./registry/registry.password:/auth/registry.password
      # Монтируем директорию с данными.
      - ./registry/data:/data
    ports:
      - 5000
```

Файл паролей, заданный в `REGISTRY_AUTH_HTPASSWD_PATH`, используется для аутентификации пользователей, когда те загружают образы в реестр или извлекают их из него. Создайте файл паролей с помощью команды `htpasswd`, а также директорию для хранения образов:

```
mkdir -p ./registry/data


# Устанавливаем htpasswd.
sudo apt install apache2-utils


# Создаём файл password. Имя пользователя: busy, пароль: bee.
htpasswd -Bbn busy bee > ./registry/registry.password
```

Теперь можно запустить контейнер **registry**. Если всё работает как надо, вы увидите следующее сообщение:

```
docker-compose up
```

```
# registry | level=info msg="listening on [::]:5000"
```

SSL-сертификаты и nginx

Как уже говорилось, можно использовать nginx для обработки TLS-трафика и перенаправления запросов к контейнеру **registry**. Docker registry для работы требуется действительный доверенный SSL-сертификат. Можно воспользоваться сервисом вроде Let's Encrypt или получить его вручную. Убедитесь, что доменное имя указывает на ваш сервер (в моём случае **registry.pliutau.com**). Для этой статьи я получил сертификаты с помощью certbot и поместил их в каталог `./nginx/certs`.

Поскольку Docker registry работает в контейнере, nginx тоже можно запустить в контейнере. Добавим соответствующий сервис в файл **compose.yaml**:

```
services:
  registry:
    # ...
  nginx:
    image: nginx:latest
    depends_on:
      - registry
    volumes:
      # Монтируем конфигурацию nginx.
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf
      # Монтируем сертификаты, полученные от Let's Encrypt.
      - ./nginx/certs:/etc/nginx/certs
    ports:
      - "443:443"
```

А вот пример файла конфигурации nginx.conf:

```
worker_processes auto;


events {
    worker_connections 1024;
}


http {
    upstream registry {
        server registry:5000;
    }
```

```
    server {
        server_name registry.pliutau.com;
        listen 443 ssl;

        ssl_certificate /etc/nginx/certs/fullchain.pem;
        ssl_certificate_key /etc/nginx/certs/privkey.pem;

        location / {
            # Важный параметр для больших образов.
            client_max_body_size                1000m;

            proxy_pass                          http://registry;
            proxy_set_header  Host              $http_host;
            proxy_set_header  X-Real-IP         $remote_addr;
            proxy_set_header  X-Forwarded-For   $proxy_add_x_forwarded_for;
            proxy_set_header  X-Forwarded-Proto $scheme;
            proxy_read_timeout                  900;
        }
    }
}
```

Всё готово

После этих шагов можно запускать контейнеры **registry** и **nginx**:

```
docker-compose up
```

Входим в реестр:

```
docker login registry.pliutau.com


# Username: busy
# Password: bee
# Login Succeeded
```

Всё! Теперь можно собирать образы и пушить их в собственный реестр:

```
docker build -t registry.pliutau.com/pliutau/hello-world:v0 .


docker push registry.pliutau.com/pliutau/hello-world:v0
```

```
# v0: digest: sha256:a56ea4... size: 738
```

На самом сервере загруженные образы хранятся в директории **data**:

```
ls -la ./registry/data/docker/registry/v2/repositories/
```

Другие варианты

Следуя приведённому выше примеру, собственный реестр также можно запустить в Kubernetes. Или можно воспользоваться managed-сервисом вроде Harbor — Open Source-реестром с продвинутыми возможностями в области безопасности, совместимым с Docker и Kubernetes.

Тем, кто предпочитает работать через пользовательский интерфейс, рекомендую обратить внимание на проекты вроде joxit/docker-registry-ui (его также можно запустить в отдельном контейнере).

Заключение

С собственным реестром вы получаете полный контроль над ним. В то же время возникают дополнительные издержки на обслуживание и защиту.

Какими бы ни были причины завести собственный реестр, теперь вы знаете, как это сделать. Сравнивайте различные варианты, выбирайте тот, который лучше всего соответствует вашим потребностям.

Исходники для этой статьи доступны на GitHub.

Revision #2
Created 2025-11-17 13:45:22 UTC by Эд Кукса
Updated 2025-11-27 18:38:06 UTC by Эд Кукса