

# DI-212 docker - Шенгелай Всеволод

▶ На текущий момент идёт работа над этим заданием ▶

Ресурс	URL
Статический веб-сайт	<a href="https://109.71.246.234/">https://109.71.246.234/</a>
Список образов в нашем registry	<a href="http://109.71.246.234:5000/v2/_catalog">http://109.71.246.234:5000/v2/_catalog</a>

## Этап 1. Развёртывание NGINX в докере

Будем использовать alpine-образ nginx в качестве базового.

Стягиваем с dockerHub необходимый образ

```
docker pull nginx:stable-alpine3.19-perl
```

Все необходимые файлы для сборки образа будем хранить в следующей структуре. Содержимое файла конфигурации веб-сервера можно посмотреть здесь [DI-192 nginx + сети \(Шенгелай Всеволод\)](#)

```
root@3087515-ge73243:~/nginx-files# tree
.
├── sites-configs
│   └── my_website
├── ssl
│   ├── nginx-selfsigned.crt
│   └── nginx-selfsigned.key
└── static-websites
    ├── simplestyle_horizon_with_404_page
    │   ├── 404.html
    │   ├── another_page.html
    │   ├── contact.html
    │   ├── examples.html
    │   ├── index.html
    │   ├── page.html
    │   ├── READ_ME
    │   │   ├── HTML5WebTemplates.co.uk.url
    │   │   ├── PLEASE_READ.txt
    │   │   └── Remove the footer link.URL
    │   └── style
    │       ├── 404.webp
    │       ├── banner.jpg
    │       ├── bullet.png
    │       ├── footer.png
    │       ├── graphic.png
    │       ├── link.png
    │       ├── search.png
    │       ├── side_back.png
    │       ├── side_base.png
    │       ├── side_top.png
    │       └── style.css
```

Структура файлов нашего веб-сайта

### Напишем Dockerfile

```
# Nginx
FROM nginx:stable-alpine3.19-perl

#
COPY static-websites/simplestyle_horizon_with_404_page/ /var/www/html/simplestyle_horizon_with_404_page/

#
COPY sites-configs/my_website /etc/nginx/conf.d/my_site.conf

# SSL
COPY ssl/nginx-selfsigned.crt /etc/nginx/ssl/
COPY ssl/nginx-selfsigned.key /etc/nginx/ssl/

#
EXPOSE 80
EXPOSE 443

# nginx " "
CMD ["nginx", "-g", "daemon off;"]
```

```
docker build -t pre-configured-nginx .
```

### Создаём Docker Container

```
docker run --rm -d -p 80:80 -p 443:443 --name my-nginx-container pre-configured-nginx
```

### Запуск интерактивной оболочки sh, работающей в контейнере (использовалась для отладочных целей)

```
docker exec -it my-nginx-container /bin/sh
```

## Этап 2. Проверка контейнера на наличие уязвимостей

Скачаем образ trivy, и будем запускать контейнеры, сканирующие образ

### Команда запуска сканирования образа

```
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock -v $HOME/Library/Caches:/root/.cache/ aquasec/trivy:latest image pre-configured-nginx
```

```
root@3087515-ge73243:~# docker run --rm -v /var/run/docker.sock:/var/run/docker.sock -v $HOME/Library/Caches:/root/.cache/ aquasec/trivy:latest image pre-configured-nginx
2024-08-02T10:37:38Z INFO [vuln] Vulnerability scanning is enabled
2024-08-02T10:37:38Z INFO [secret] Secret scanning is enabled
2024-08-02T10:37:38Z INFO [secret] If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2024-08-02T10:37:38Z INFO [secret] Please see also https://aquasecurity.github.io/trivy/v0.54/docs/scanner/secret#recommendation for faster secret detection
2024-08-02T10:37:38Z INFO Detected OS family="alpine" version="3.19.3"
2024-08-02T10:37:38Z INFO [alpine] Detecting vulnerabilities... os_version="3.19" repository="3.19" pkg_num=68
2024-08-02T10:37:38Z INFO Number of language-specific files num=0
2024-08-02T10:37:38Z WARN Using severities from other vendors for some vulnerabilities. Read https://aquasecurity.github.io/trivy/v0.54/docs/scanner/vulnerability#severity-selection for details.

pre-configured-nginx (alpine 3.19.3)
=====
Total: 14 (UNKNOWN: 0, LOW: 0, MEDIUM: 12, HIGH: 2, CRITICAL: 0)
```

Library	Vulnerability	Severity	Status	Installed Version	Fixed Version	Title
curl	CVE-2024-2398	HIGH	fixed	8.5.0-r0	8.7.1-r0	curl: HTTP/2 push headers memory-leak <a href="https://avd.aquasec.com/nvd/cve-2024-2398">https://avd.aquasec.com/nvd/cve-2024-2398</a>
	CVE-2024-0853	MEDIUM			8.6.0-r0	curl: OCSP verification bypass with TLS session reuse <a href="https://avd.aquasec.com/nvd/cve-2024-0853">https://avd.aquasec.com/nvd/cve-2024-0853</a>
	CVE-2024-2004				8.7.1-r0	curl: Usage of disabled protocol <a href="https://avd.aquasec.com/nvd/cve-2024-2004">https://avd.aquasec.com/nvd/cve-2024-2004</a>
	CVE-2024-2379					curl: QUIC certificate check bypass with wolfSSL <a href="https://avd.aquasec.com/nvd/cve-2024-2379">https://avd.aquasec.com/nvd/cve-2024-2379</a>
	CVE-2024-2466					curl: TLS certificate check bypass with mbedTLS <a href="https://avd.aquasec.com/nvd/cve-2024-2466">https://avd.aquasec.com/nvd/cve-2024-2466</a>
	CVE-2024-6197				8.9.0-r0	curl: freeing stack buffer in utf8asn1str <a href="https://avd.aquasec.com/nvd/cve-2024-6197">https://avd.aquasec.com/nvd/cve-2024-6197</a>
	CVE-2024-6874					curl: macidn punycode buffer overread <a href="https://avd.aquasec.com/nvd/cve-2024-6874">https://avd.aquasec.com/nvd/cve-2024-6874</a>
libcurl	CVE-2024-2398	HIGH			8.7.1-r0	curl: HTTP/2 push headers memory-leak <a href="https://avd.aquasec.com/nvd/cve-2024-2398">https://avd.aquasec.com/nvd/cve-2024-2398</a>
	CVE-2024-0853	MEDIUM			8.6.0-r0	curl: OCSP verification bypass with TLS session reuse <a href="https://avd.aquasec.com/nvd/cve-2024-0853">https://avd.aquasec.com/nvd/cve-2024-0853</a>
	CVE-2024-2004				8.7.1-r0	curl: Usage of disabled protocol <a href="https://avd.aquasec.com/nvd/cve-2024-2004">https://avd.aquasec.com/nvd/cve-2024-2004</a>
	CVE-2024-2379					curl: QUIC certificate check bypass with wolfSSL <a href="https://avd.aquasec.com/nvd/cve-2024-2379">https://avd.aquasec.com/nvd/cve-2024-2379</a>
	CVE-2024-2466					curl: TLS certificate check bypass with mbedTLS <a href="https://avd.aquasec.com/nvd/cve-2024-2466">https://avd.aquasec.com/nvd/cve-2024-2466</a>
	CVE-2024-6197				8.9.0-r0	curl: freeing stack buffer in utf8asn1str <a href="https://avd.aquasec.com/nvd/cve-2024-6197">https://avd.aquasec.com/nvd/cve-2024-6197</a>
	CVE-2024-6874					curl: macidn punycode buffer overread <a href="https://avd.aquasec.com/nvd/cve-2024-6874">https://avd.aquasec.com/nvd/cve-2024-6874</a>

Список найденных уязвимостей. Все они являются fixed, то есть в новых версиях пакетов

```
/etc/nginx/ssl/nginx-selfsigned.key (secrets)
Total: 1 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 1, CRITICAL: 0)
HIGH: AsymmetricPrivateKey (private-key)
Asymmetric Private Key
/etc/nginx/ssl/nginx-selfsigned.key:1 (added by 'COPY ssl/nginx-selfsigned.key /etc/nginx')
1 [ -----BEGIN PRIVATE KEY-----
-----END PRIVATE KEY
2
```

В данном блоке trivy показывает нам обнаруженные ключи. Хранение ключей в образе создаёт угрозу безопасности. Тот, кто получит образ, может получить к ним доступ

После анализа образа, проведённого trivy, внесём изменения в Dockerfile.

```
# Используем официальный образ Nginx в качестве базового
FROM nginx:stable-alpine3.19-perl

# Копируем статические вебсайты
COPY static-websites/simplestyle_horizon_with_404_page/ /var/www/html/simplestyle_horizon_with_404_page/

# Копируем конфигурационные файлы сайтов
COPY sites-configs/my_website /etc/nginx/conf.d/my_site.conf

# Обновляем пакетный менеджер и устанавливаем последние версии curl и libcurl
RUN apk update && \
    apk upgrade && \
    apk add --no-cache \
    curl=8.9.0-r0 \
    libcurl=8.9.0-r0

# Открываем порты
EXPOSE 80
EXPOSE 443

# Запуск nginx "на переднем плане"
CMD ["nginx", "-g", "daemon off;"]
```

Dockerfile с фиксом уязвимостей

Что мы сделали:

1. Установили последние версии пакетов, в которых уже устранены обнаруженные уязвимости
2. Убрали копирование ключей в образ, чтобы никто их не мог из него вытащить. Ключи теперь получаем через volume

Новая команда для запуска контейнера, с монтированием тома

```
docker run --rm -d -p 80:80 -p 443:443 -v $(pwd)/ssl:/etc/nginx/ssl --name my-nginx-container pre-configured-nginx
```

## Этап 3. Проведение линта докерфайла

Раз уж у нас неделя докера, то hadolint тоже будем использовать в докеризованном варианте

```
root@3087515-ge73243:~/nginx-files# docker run --rm -i ghcr.io/hadolint/hadolint < Dockerfile
root@3087515-ge73243:~/nginx-files#
```

Hadolint ничего не написал, значит ошибок нет. Мы преднамеренно написали синтаксически неверные выражения в Dockerfile, чтобы проверить, ч

Веб-версия, правда, указала на недопустимость использования `apk upgrade`. Это может сделать образ менее предсказуемым и привести к конфликтам в зависимостях. Следует обновлять только необходимые пакеты.

Do not use `apk upgrade`

```
11 RUN apk update && \
12     apk upgrade && \
13     apk add --no-cache \
14     curl=8.9.0-r0 \
15     libcurl=8.9.0-r0
```

Изменим команду RUN:

```

11 RUN apk --no-cache add \
12     curl=8.9.0-r0 \
13     libcurl=8.9.0-r0

```

Версия RUN с обновлением только конкретных пакетов

## Этап 4. Уменьшение размера образа

Мы постарались сделать итоговый образ как можно меньше. Наши шаги в достижении показателей:

1. Использовали alpine версию базового образа;
2. Обновляли только необходимые пакеты, не сохраняя кеш.

В Dockerfile мы копируем конфигурацию веб-сервера и статический веб-сайт в Image, поэтому он "подрастает" на размер этих файлов. Такова была наша задача - мы хотим, чтобы любой мог развернуть именно наш веб-сайт, без дополнительных конфигураций.

```

root@3087515-ge73243:~/nginx-files# docker image ls

```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
pre-configured-nginx	0.0.1	f7b6dff5ba34	8 minutes ago	82MB
<none>	<none>	046d3e81591b	2 hours ago	84.2MB
aquasec/trivy	latest	2275ced91cdc	45 hours ago	147MB
nginx	stable-alpine3.19-perl	2c94260d23a7	6 weeks ago	80.8MB
hadolint/hadolint	latest	da13a5ec2e84	20 months ago	2.43MB
ghcr.io/hadolint/hadolint	latest	da13a5ec2e84	20 months ago	2.43MB
weejewel/wg-easy	latest	c90b263d8a38	2 years ago	142MB
aquasec/trivy	0.18.3	9e888dd7d7a9	3 years ago	53.6MB

Размер итогового образа

Но в исследовательских целях мы будем пробрасывать данные через volume.

```

root@3087515-ge73243:~/nginx-files# docker image ls

```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
pre-configured-nginx	0.0.1	25685b9c5482	3 seconds ago	81.9MB
<none>	<none>	f7b6dff5ba34	16 minutes ago	82MB
<none>	<none>	046d3e81591b	2 hours ago	84.2MB
aquasec/trivy	latest	2275ced91cdc	45 hours ago	147MB
nginx	stable-alpine3.19-perl	2c94260d23a7	6 weeks ago	80.8MB
hadolint/hadolint	latest	da13a5ec2e84	20 months ago	2.43MB
ghcr.io/hadolint/hadolint	latest	da13a5ec2e84	20 months ago	2.43MB
weejewel/wg-easy	latest	c90b263d8a38	2 years ago	142MB
aquasec/trivy	0.18.3	9e888dd7d7a9	3 years ago	53.6MB

Размер образа с монтированием тома

Это не сильно помогло. Всего 100Кб...

Давайте откажемся от обновления пакетов, посмотрим, насколько "похудеет" образ

```
root@3087515-ge/3243:~/nginx-files# docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<none>	<none>	25685b9c5482	2 minutes ago	81.9MB
<none>	<none>	f7b6dff5ba34	18 minutes ago	82MB
<none>	<none>	046d3e81591b	2 hours ago	84.2MB
aquasec/trivy	latest	2275ced91cdc	45 hours ago	147MB
pre-configured-nginx	0.0.1	42db3f8def0d	6 weeks ago	80.8MB
nginx	stable-alpine3.19-perl	2c94260d23a7	6 weeks ago	80.8MB
hadolint/hadolint	latest	da13a5ec2e84	20 months ago	2.43MB
ghcr.io/hadolint/hadolint	latest	da13a5ec2e84	20 months ago	2.43MB
weejewel/wg-easy	latest	c90b263d8a38	2 years ago	142MB
aquasec/trivy	0.18.3	9e888dd7d7a9	3 years ago	53.6MB

Основную часть образа содержали обновлённые пакеты

## Этап 5.1. Поднимаем регистры (docker pull registry), и пушим туда свой образ

```
docker run -d -p 5000:5000 --restart always --name my-local-registry registry:2
```

### Тестируем образ для нашего реестра

```
docker tag pre-configured-nginx:0.0.1 localhost:5000/pre-configured-nginx:0.0.1
```

### Пушим образ в registry

```
docker push localhost:5000/pre-configured-nginx:latest
```

## Этап 5.1. Поднимаем nexus, и пушим туда свой образ

Nexus имеет большой аппетит к процессорному времени и ОЗУ. Пока я не собрал мощный сервер с Рохтох и не настроил ReverseProху для доступа к нему, придётся поднять Nexus локально.

Соберём свой образ Nexus, где увеличим размер памяти для кучи JVM, согласно [рекомендациям разработчиков Nexus](#).

## Dockerfile для кастомного образа nexus

```
# Nexus
FROM sonatype/nexus3

# JVM-
ENV INSTALL4J_ADD_VM_PARAMS="-Xms5G -Xmx5G -Djava.util.prefs.userRoot=/nexus-data/javaprefs"

# ( "/" )
ENV NEXUS_CONTEXT="/"

# Nexus
EXPOSE 8081

# Nexus
VOLUME /nexus-data

# Nexus
RUN mkdir -p /nexus-data && chown -R 200 /nexus-data

# Nexus
CMD ["sh", "-c", "/opt/sonatype/nexus/bin/nexus run"]
```

Nexus способен работать с двумя БД - H2 и PostgreSQL. Разработчики Nexus рекомендуют использовать PostgreSQL, причём ставить её вне контейнера, на специализированном сервере для БД (у H2 есть проблемы с ACID, а именно с D-Durability). Но работа с внешним сервером БД является лицензионной функцией (Nexus уже придумали как взламывать, но оставим это на будущее).

Можем посмотреть логи контейнера и увидеть, что нексус by-default использует именно H2.

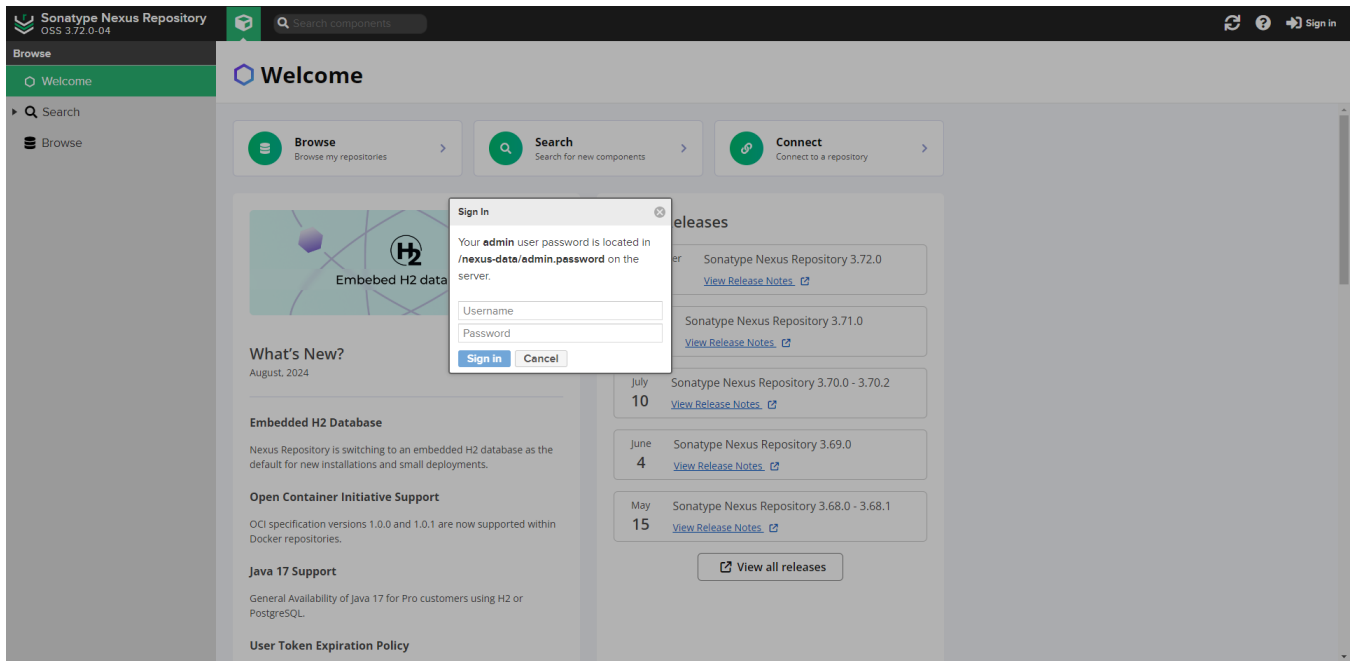
```
docker logs nexus
```

```
2024-09-06 12:11:54,536+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.datastore.DataStoreConfigurationDefaultSource - Loaded 'nexus' data store configuration defaults (Embedded H2)
2024-09-06 12:11:55,132+0000 INFO [FelixStartLevel] *SYSTEM com.zaxxer.hikari.HikariDataSource - nexus - Starting...
2024-09-06 12:11:55,347+0000 INFO [FelixStartLevel] *SYSTEM com.zaxxer.hikari.HikariDataSource - nexus - Start completed.
2024-09-06 12:11:55,351+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.datastore.mybatis.MyBatisDataStore - nexus - Loading MyBatis configuration from /opt/sonatype/nexus/etc/fabric/mybatis.xml
2024-09-06 12:11:55,598+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.datastore.mybatis.MyBatisDataStore - nexus - MyBatis databaseId: H2
2024-09-06 12:11:55,943+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.datastore.mybatis.MyBatisDataStore - nexus - Creating schema for UpgradeTasksDAO
2024-09-06 12:11:55,957+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.datastore.mybatis.MyBatisDataStore - nexus - Creating schema for JwtSecretDAO
2024-09-06 12:11:55,972+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.datastore.mybatis.MyBatisDataStore - nexus - Creating schema for LoggingOverridesDAO
2024-09-06 12:11:55,983+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.datastore.mybatis.MyBatisDataStore - nexus - Creating schema for DeploymentIdDAO
2024-09-06 12:11:55,993+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.datastore.mybatis.MyBatisDataStore - nexus - Creating schema for NodeIdDAO
2024-09-06 12:11:56,003+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.datastore.mybatis.MyBatisDataStore - nexus - Creating schema for AnonymousConfigurationDAO
2024-09-06 12:11:56,015+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.datastore.mybatis.MyBatisDataStore - nexus - Creating schema for CPrivilegeDAO
2024-09-06 12:11:56,026+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.datastore.mybatis.MyBatisDataStore - nexus - Creating schema for RoleDAO
2024-09-06 12:11:56,034+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.datastore.mybatis.MyBatisDataStore - nexus - Creating schema for UserDAO
2024-09-06 12:11:56,046+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.datastore.mybatis.MyBatisDataStore - nexus - Creating schema for UserRoleMappingDAO
2024-09-06 12:11:56,060+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.datastore.mybatis.MyBatisDataStore - nexus - Creating schema for RealmConfigurationDAO
2024-09-06 12:11:56,080+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.datastore.mybatis.MyBatisDataStore - nexus - Creating schema for SecretDAO
2024-09-06 12:11:56,094+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.datastore.mybatis.MyBatisDataStore - nexus - Creating schema for QuartzDAO
2024-09-06 12:11:56,146+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.datastore.mybatis.MyBatisDataStore - nexus - Creating schema for KeyStoreDAO
2024-09-06 12:11:56,163+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.datastore.mybatis.MyBatisDataStore - nexus - Creating schema for BlobStoreMetricsDAO
```

Билдим и запускаем контейнер

```
docker build --rm=true --tag=nexus-custom -f custom-nexus-dockerfile .
docker run -d -p 8081:8081 -p 8082:8082 -p 8083:8083 --name nexus -v nexus-data:/nexus-data nexus-custom
docker logs nexus
```

Перейдём к веб-интерфейсу. Nexus подсказывает, где можно посмотреть пароль.



Получаем пароль админа

```
sudo cat /var/lib/docker/volumes/nexus-data/_data/admin.password
```

При первом входе Nexus нужно будет настроить. Нам сразу же предложат включить анонимный доступ. Потренируемся работать с credentials - не будем его включать

### Configure Anonymous Access 3 of 4

**Enable anonymous access** means that by default, users can search, browse and download components from repositories without credentials. Please **consider the security implications for your organization**.

**Disable anonymous access** should be chosen with care, as it **will require credentials for all users and/or build tools**.

[More information](#)

☐ Enable anonymous access

☒ Disable anonymous access

Back Next

Перед тем как пушить образы в Nexus, его сначала нужно настроить как DockerRegistry.



**Name:** A unique identifier for this repository

**Online:** ☒ If checked, the repository accepts incoming requests

**Repository Connectors**

Connectors allow Docker clients to connect directly to hosted registries, but are not always required. Consult our [documentation](#) for which connector is appropriate for your use case. For information on scaling the repositories see our [scaling documentation](#).

**HTTP:**  
 Create an HTTP connector at specified port. Normally used if the server is behind a secure proxy.  
☒ 8082

**HTTPS:**  
 Create an HTTPS connector at specified port. Normally used if the server is configured for https.  
☐

**Allow anonymous docker pull:**  
☐ Allow anonymous docker pull ( Docker Bearer Token Realm required )

**Docker Registry API Support**

**Enable Docker V1 API:**  
☒ Allow clients to use the V1 API to interact with this repository

**Storage**

**Blob store:**  
 Blob store used to store repository contents

**Strict Content Type Validation:**  
☒ Validate that all content uploaded to this repository is of a MIME type appropriate for the repository format

Осуществляем логин с аккаунта админа. не забываем, что docker требует наличия SSL-сертификата у registry. поэтому игнорируем быдло добавляем эндпоинт с репозиторием в список insecure-registries в конфигурации службы docker

Отредактируем файл `/etc/docker/daemon.json`, вставив в него следующую запись:

```
vsevolod@192.168.0.100:22
GNU nano 6.2
{
    "insecure-registries" : [ "localhost:8082", "192.168.0.100:8082" ]
}
```

Перезапустим службу docker:

```
sudo systemctl restart docker.service
```

Залогинимся в registry:

```
vsevolod@vm-pc:~$ docker login localhost:8082
Username: admin
Password:
WARNING! Your password will be stored unencrypted in /home/vsevolod/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores

Login Succeeded
```

Тестируем образ для нашего реестра

```
docker tag nexus-custom:latest localhost:8082/nexus-custom:latest
```

### Пушим образ

```
docker push localhost:8082/pre-configured-nginx:latest
```

```
vsevolod@vm-pc:~$ docker push localhost:8082/nexus-custom
Using default tag: latest
The push refers to repository [localhost:8082/nexus-custom]
ffb95ea3c777: Pushed
27ecfccc1043: Pushed
123e3ccb343a: Pushed
39d5ca9e3e8a: Pushed
112d2a44dc1d: Pushed
9bca9f7811f4: Pushed
8ee14fa8d376: Pushed
59a5c510c4c9: Pushed
latest: digest: sha256:f7ecedbbe356b1e9e3bf79e1504bb662a13b0561751c6047ce1d9e29b5fc1814 size: 1995
```

Образ улетел в registry

Видим наш образ в Docker Registry через Nexus UI:

The screenshot shows the Sonatype Nexus Repository OSS 3.72.0-04 interface. The left sidebar has a 'Browse' menu. The main content area shows the 'my\_docker\_repo' repository with a tree view of 'v2' containing 'blobs', 'nexus-custom', 'manifests', 'tags', and 'latest'. The 'manifests' section is expanded, showing a list of manifests with their SHA256 hashes. The right sidebar shows the 'Delete asset' button and a 'Summary' section with 'Attributes'.

Attributes	
<b>Checksum</b>	
sha256	f7ecedbbe356b1e9e3bf79e1504bb662a13b0561751c6047ce1d9e29b5fc1814
sha1	ac4b82c05dfbdf0368865147767c41dac452404a1
<b>Docker</b>	
content_digest	sha256:f7ecedbbe356b1e9e3bf79e1504bb662a13b0561751c6047ce1d9e29b5fc1814
asset_kind	MANIFEST

## Этап 6. Docker inside Docker и Docker outside Docker

### Запуск контейнера на основе образа dind

```
docker run --privileged --name my-dind -d docker:27.2.0-dind
```

```
vsevolod@vm-pc:~$ docker run --privileged --name my-dind -d docker:27.2.0-dind
79439d7790286e297a7834045a53f1cae5616ef097fd0a98f6ac23bd21683835
vsevolod@vm-pc:~$ docker exec -it my-dind /bin/sh
/ # docker version
Client:
Version:      27.2.0
API version:  1.47
Go version:   go1.21.13
Git commit:   3ab4256
Built:        Tue Aug 27 14:14:20 2024
OS/Arch:      linux/amd64
Context:      default

Server: Docker Engine - Community
Engine:
Version:      27.2.0
API version:  1.47 (minimum version 1.24)
Go version:   go1.21.13
Git commit:   3ab5c7d0
Built:        Tue Aug 27 14:15:44 2024
OS/Arch:      linux/amd64
Experimental: false
containerd:
Version:      v1.7.21
GitCommit:    472731909fa34bd7bc9c087e4c27943f9835f111
runc:
Version:      1.1.13
GitCommit:    v1.1.13-0-g58aa920
docker-init:
Version:      0.19.0
GitCommit:    de40ad0

/ # docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS        NAMES
```

Запускаем контейнер с установленным внутри докер демоном

Здесь я решил браво пробросить сокет docker В контейнер с debian, не учтя при этом что сокет docker-сервера мы пробросили, но нам нужен ещё клиент для общения с сервером. Можно доставить его сюда, но проще запустить контейнер на основе образа docker и уже туда пробросить сокет.

```
latest: Pulling from library/debian
8cd46d290033: Pull complete
Digest: sha256:b8084b1a576c5504a031936e1132574f4ce1d6cc7130bbcc25a28f074539ae6b
Status: Downloaded newer image for debian:latest
docker.io/library/debian:latest
vsevolod@vm-pc:~$ docker run -it -v /var/run/docker.sock:/var/run/docker.sock debian /bin/bash
root@7c55cf95a3c7:/# exit
exit
vsevolod@vm-pc:~$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS        NAMES
79439d779028   docker:27.2.0-dind                 "dockerd-entrypoint..." 7 minutes ago  Up 7 minutes  2375-2376/tcp  my-dind
57781e1c0898   sonatype/nexus3                    "/opt/sonatype/nexus..." 2 hours ago    Up 2 hours    0.0.0.0:8081-8083->8081-8083/tcp, :::8081-8083->8081-8083/tcp, 0.0.0.0:8123->8123/tcp, :::8123->8123/tcp  nexus
vsevolod@vm-pc:~$ docker run -it -v /var/run/docker.sock:/var/run/docker.sock debian /bin/bash
root@228b786158f7:/# docker run -d -it --name another-one-debian debian
bash: docker: command not found
root@228b786158f7:/#
```

Заработало!

## Запуск контейнера на основе образа dind с пробросом сокета

```
docker run -it -v /var/run/docker.sock:/var/run/docker.sock docker:27.2.0-dind /bin/sh
```

```
vsevolod@vm-pc:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
57781e1c0a98   sonatype/nexus3 "/opt/sonatype/nexus..." 2 hours ago    Up 2 hours    0.0.0.0:8081-8083->8081-8083/tcp, :::8081-8083->8081-8083/tcp, 0.0.0.0:8123->8123/tcp, :::8123->8123/tcp    nexus

vsevolod@vm-pc:~$ docker run -it -v /var/run/docker.sock:/var/run/docker.sock docker:27.2.0-dind /bin/sh
/ # docker run -it -d --name another-one-debian-machine debian
0b8f935452dbc755e5c6fb672369d242ee4fd7e8546ef6edb12c724bd87892e
/ # docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
0b8f935452db   debian        "bash"                  6 seconds ago  Up 6 seconds    2375-2376/tcp                another-one-debian-machine
186600d96440   docker:27.2.0-dind "dockerd-entrypoint..." 46 seconds ago  Up 45 seconds    2375-2376/tcp                romantic_mirzakhani
57781e1c0a98   sonatype/nexus3 "/opt/sonatype/nexus..." 2 hours ago    Up 2 hours    0.0.0.0:8081-8083->8081-8083/tcp, :::8081-8083->8081-8083/tcp, 0.0.0.0:8123->8123/tcp, :::8123->8123/tcp    nexus
/ # exit
vsevolod@vm-pc:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
0b8f935452db   debian        "bash"                  23 seconds ago  Up 22 seconds    2375-2376/tcp                another-one-debian-machine
57781e1c0a98   sonatype/nexus3 "/opt/sonatype/nexus..." 2 hours ago    Up 2 hours    0.0.0.0:8081-8083->8081-8083/tcp, :::8081-8083->8081-8083/tcp, 0.0.0.0:8123->8123/tcp, :::8123->8123/tcp    nexus
vsevolod@vm-pc:~$
```

## Выводы:

### Docker Outside Docker (DoD)

**Преимущества:** Экономит ресурсы, позволяет контейнеру напрямую управлять Docker демоном хоста через проброс сокета.

**Недостатки:** Небезопасен, так как контейнер получает полный доступ к Docker хоста, и может эксплуатировать его уязвимости (например, создавать неограниченное кол-во контейнеров, истощая ресурсы хост-системы).

### Docker in Docker (DiD)

**Преимущества:** Полная изоляция, контейнер автономен и запускает свои контейнеры без влияния на хост.

**Недостатки:** Требуется больше ресурсов и привилегий, сложнее в настройке и занимает дополнительные ресурсы хост-системы.