

УТВЕРЖДЕН
643.72410666.00067-07 99 01-ЛУ

ЗАЩИЩЕННАЯ СИСТЕМА УПРАВЛЕНИЯ
БАЗАМИ ДАННЫХ «ЈАТОВА»

Инструкция по настройке и использованию контейнера

643.72410666.00067-07 99 01

Листов 47

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

АННОТАЦИЯ

В документе приведены сведения, описывающие работу контейнера СУБД «Jatoba».

Степени важности примечаний, применяемые в документе:



Важная информация – указания, требующие особого внимания



Дополнительная информация – указания, позволяющие упростить работу с изделием



Для СУБД «Jatoba» версии ядра 4 используется версия — 4.17.1

Для СУБД «Jatoba» версии ядра 5 используется версия — 5.12.1

Для СУБД «Jatoba» версии ядра 6 используется версия — 6.8.1



Для сертифицированной версии СУБД «Jatoba» поддерживается работа только на ОС, указанных в формуляре на поставку!

СОДЕРЖАНИЕ

1. Требования к ПО.....	4
2. Комплект контейнера.....	5
3. Установка контейнера.....	10
3.1. Обновление ОС.....	10
3.2. Установка платформы docker в ОС.....	11
3.3. Импорт образа	12
3.4. Переменные окружения	14
3.4.1. Запуск нового контейнера	16
3.4.2. Запуск контейнера	17
3.4.3. Запуск контейнера через docker compose.....	18
4. Основные операции с контейнером.....	21
4.1. Проверка статуса (docker ps -a).....	21
4.2. Проверка наличие созданных томов (volume list)	22
4.3. Остановка контейнера (stop.sh).....	22
4.4. Запуск остановленного контейнера (start.sh).....	23
4.5. Перезапуск контейнера (restart.sh).....	23
4.6. Логирование контейнера (log.sh)	24
4.7. Удаление контейнера (remove.sh)	24
4.8. Запуск контейнера через docker run.....	25
4.9. Сохранение данных контейнера	27
5. Изменение пользовательских настроек postgresql.conf.....	28
6. Примеры команд запуска контейнера	30
7. Использование утилиты psql	31
8. Сопоставление слоев контейнера с директориями	33
9. Поведение контейнера в особых случаях.....	35
9.1. Изменение настроек после запуска контейнера	35
10. SSL Аутентификация в СУБД «Jatoba» контейнере	36
Приложение 1	37
Установка компонента Oracle_FDW	37
Приложение 2	39
Установка компонента PLsPgSQL.....	39
Приложение 3	41
Пример настройки резервного копирования.....	41
Термины и определения	45
Перечень сокращений.....	46

1. ТРЕБОВАНИЯ К ПО

Контейнер устанавливается на ЭВМ под управлением ОС, указанных в таблице 1.1.

Таблица 1.1 – Перечень поддерживаемых ОС

№	Наименование ОС	Серверная часть	Клиентская часть	Docker (ver.)	Сертификат ФСТЭК	
					№ серт.	Дата выдачи
1	Windows 10	X	X	—	—	—
2	Windows 11	X	X	—	—	—
3	Windows Server 2016	X	X	—	—	—
4	Windows Server 2019	X	X	—	—	—
5	Windows Server 2022	X	X	—	—	—
6	Astra Linux 1.7 Special Edition Смоленск (x86-64)	X	X	25.0.5	2557	30.01.2012
7	Astra Linux 1.8 (x86-64)	X	X	25.0.5	—	—
8	Astra Linux 2.12 Common Edition Орел (x86-64)	X	X	—	—	—
9	Debian 10	X	X	24.0.2	—	—
10	Debian 11	X	X	24.0.2	—	—
11	Debian 12	X	X	24.0.2	—	—
12	АЛТ 8 СП	X	X	27.1.1	3866	10.08.2018
13	АЛТ 10 СП	X	X	27.1.1	3866	10.08.2018
14	АЛТ 9.1 Server	X	X	—	—	—
15	АЛТ 10 Server	X	X	23.0.1	—	—
16	Ubuntu 20.04	X	X	24.0.2	—	—
17	Ubuntu 22.04	X	X	24.0.2	—	—
18	Ubuntu 24.04	X	X	24.0.2	—	—
19	ОСНОВА2	X	X	20.10.5	4381	31.03.2021
20	РЕД ОС 7.3 Муром	X	X	25.0.7	4060	12.01.2019
21	РЕД ОС 8	X	X	—	—	—
22	РОСА 7.9	X	X	—	—	—
23	РОСА 12.4	X	X	—	—	—
24	RedHat Enterprise Linux 8	X	X	—	—	—
25	Oracle Linux 8.4	X	X	—	—	—

Для использования контейнера необходимо установить docker-се версии, указанной в таблице 1.1. Инструкцию по установке ПО для необходимой ОС можно найти на сайтах:

- для Astra Linux — <https://wiki.astralinux.ru/pages/viewpage.action?pageId=158601444>,
- для РЕД ОС — <https://redos.red-soft.ru/base/arm/arm-other/docker-install/>,
- для АЛТ — <https://www.altlinux.org/Docker>,
- для остальных ОС, представленных в таблице 1.1 — <https://docs.docker.com/engine/install/>.

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

2. КОМПЛЕКТ КОНТЕЙНЕРА

Контейнер СУБД «Jatoba» распространяется в виде архива, содержащего файлы, указанные в таблице 2.1.

Таблица 2.1 – Описание файлов комплекта контейнера

Название документа	Описание
standalone	Каталог содержащий инструмент предназначенный для определения и запуска многоконтейнерных приложений (docker-compose.yml)
jatobaX_X.X.X-XXXX-dfl.3.1-astralinux1.8.tar ¹	образ контейнера
.env	файл, содержащий переменные окружения для контейнера (см. подраздел 3.4)
changelog.md	текстовый файл, описывающий изменения ПО
postgresql.template.conf	конфигурация БД, использующаяся контейнером по умолчанию (см. раздел 5)
README.md	краткая инструкция к контейнеру
log.sh	скрипт, использующийся для вывода журнала событий контейнера (см. раздел 4.6)
remove.sh	скрипт, использующийся для удаления контейнера (см. подраздел 4.7)
restart.sh	скрипт, использующийся для перезапуска контейнера (см. подраздел 4.5)
run.sh	скрипт, использующийся для запуска нового контейнера (см. п. 3.4.2)
setup.sh	скрипт, использующийся для импорта образа контейнера (см. подраздел 3.3)
start.sh	скрипт, использующийся для запуска существующего контейнера (см. подраздел 4.4)
stop.sh	скрипт, использующийся для остановки контейнера (см. подраздел 4.3)

В образе контейнера содержится БД «Jatoba» и компоненты к ней (см. таблицу 2.2).
Настройка компонентов описана отдельно в соответствующих документах.

Установка дополнительных компонентов описаны в приложениях (Oracle_FWD – в Приложении 1 и PLsPgSQL – в Приложении 2).

¹ версия уточняется при поставке

Таблица 2.2 – Состав образа контейнера

№	Наименование	Описание	J4		J5		J6	
			Дист. ¹⁾	Обр.к. ²⁾	Дист. ¹⁾	Обр.к. ²⁾	Дист. ¹⁾	Обр.к. ²⁾
1	ядро СУБД		X	X	X	X	X	X
	pwgen	генератор паролей	—	—	X	X	X	X
	—	маскирование паролей	—	—	X	X	X	X
	KNN	поиск ближайших соседей	—	—	X	X	X	X
	xid64	компонент xid64	—	—	X	X	X	X
	ja_Compression	сжатие данных на уровне страниц	—	—	—	—	X	X
	WAL Recovery	восстановление поврежденных WAL записей	—	—	X	X	X	X
2	jaDog	компонент управления режимом работы узлов кластера	X	X	X	X	X	X
3	JDV (Jatoba data vault)	компонент контроля субъектов доступа	X	X	X	X	X	X
4	pgBadger	компонент формирования отчетов по журналам СУБД	X	X	X	X	X	X
5	pg_ProBackup	компонент расширенного резервного копирования	X	X	X	X	X	X
6	pg_Task	компонент планирования заданий СУБД	X	X	X	X	X	X
7	pg_Profile	компонент формирования отчетов производительности СУБД	X	X	X	X	X	X
8	JDS (Jatoba data safe)	компонент пользовательского веб-интерфейса для администраторов	X	—	X	—	X	—
9	ja_Sync Ldap	компонент синхронизации учетных записей со службами каталогов	X	X	X	X	X	X
10	PlsPgSQL	компонент обфускации кода PL/pgSQL	X	—	X	—	X	—
11	ja_Hipe_Cluster	компонент высокопроизводительного кластера	X	X	X	X	X	X
12	ja_Log	компонент централизованного сбора записей событий СУБД	X	—	X	—	X	—
13	lc_support	компонент поддержки платформы 1C	X	X	X	X	X	X
14	fasttrun	компонент совместимости с 1C	X	X	X	X	X	X
15	fulleq	компонент совместимости с 1C	X	X	X	X	X	X
16	mchar	компонент совместимости с 1C	X	X	X	X	X	X
17	online_analyze	компонент совместимости с 1C	X	X	X	X	X	X
18	plantuner	компонент совместимости с 1C	X	X	X	X	X	X
19	ja_CSum	компонент контроля целостности	X	X	X	X	X	X

№ изменения: _____ Подпись отв. лица: _____ Дата внесения изм: _____

№	Наименование	Описание	J4		J5		J6	
			Дист. ¹⁾	Обр.к. ²⁾	Дист. ¹⁾	Обр.к. ²⁾	Дист. ¹⁾	Обр.к. ²⁾
20	jaPooler	компонент балансировки подключений пользователей к СУБД	X	—	X	—	X	—
21	Oracle_FDW (Foreign data wrapper for oracle)	компонент доступа к данным СУБД Oracle	X	X	X	X	X	X
22	OraFCE (Oracle function compatibility extension)	компонент совместимости с СУБД Oracle	X	X	X	X	X	X
23	pg_Variables	компонент совместимости с системой глобальных переменных СУБД Oracle	X	X	X	X	X	X
24	SQL_Firewall	компонент выявления и предотвращения исполнения нетипичных SQL-запросов	X	X	X	X	X	X
25	JCS (Jatoba crypto access storage)	компонент сокрытия информации в файлах данных СУБД	X	X	X	X	X	X
26	pgSQL-HTTP	компонент формирования HTTP/HTTPS запросов из СУБД	X	X	X	X	X	X
27	TDS_FDW	компонент поддержки платформы Microsoft SQL Server	—	—	X	X	X	X
28	pgAudit	компонент расширенного журналирования событий СУБД	X	X	X	X	X	X
29	pgauditlogtofile	хранение событий безопасности в отдельном хранилище	—	—	X	X	X	X
30	PostGIS	компонент работы с географическими данными	X	X	X	X	X	X
31	PTrack	компонент расширенного резервного копирования	X	X	X	X	X	X
32	SecurityProfile	компонент управления парольными политиками пользователей СУБД	X	X	X	X	X	X
33	ja_Plan_Manager	компонент создания планов запросов в базах данных (БД), их оптимизации и экспорта в БД	X	X	X	X	X	X
34	pg_store_plans	контроль выполненных планов запросов	—	—	X	X	X	X
35	pg-hint-plan	компонент корректировки запросов	—	—	X	X	X	X

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

№	Наименование	Описание	J4		J5		J6	
			Дист. ¹⁾	Обр.к. ²⁾	Дист. ¹⁾	Обр.к. ²⁾	Дист. ¹⁾	Обр.к. ²⁾
36	ja_Container	СУБД «Jatoba» в контейнере	—	X	—	X	—	X
37	node_exporter	компонент сбора аппаратных и программных показателей работы GNU/Linux	—	—	X	—	X	—
38	postgres_exporter	компонент сбора метрик СУБД	—	—	X	—	X	—
39	sql_exporter	SQL экспортёр. Компонент сбора расширенных метрик СУБД	—	—	X	—	X	—
40	prometheus	компонент мониторинга различных программных систем и сервисов Prometheus	—	—	X	—	X	—
41	Alertmanager	компонент управления и обработки оповещений в системе мониторинга Prometheus	—	—	X	—	X	—
42	—	Работа СУБД «Jatoba» в режиме ЗПС в ОС Astra Linux	X	—	X	—	X	—
43	gis-cryptoplatform	библиотека «ГИС»	X	—	X	—	X	—
44	pg_ulid	компонент поддержки лексографического идентификатора	—	—	X	X	X	X
45	ja_SecEventLog	компонент записи событий информационной безопасности	—	—	X	X	X	X
46	rum	компонент поддерживающий обратный индекс с хранением позиционной информации и полнотекстовый поиск	—	—	X	X	X	X
47	pg_repack	компонент реорганизации таблицы с минимальными блокировками	—	—	X	X	X	X
48	osnova-digsig-key	работа СУБД Jatoba в режиме ЗПС в ОС ОСНОВА	X	—	X	—	X	—
49	tsvector2	компонент полнотекстового поиска в БД	—	—	X	X	X	X
50	ja_Similar	компонент для полнотекстового поиска и определения похожих текстов	—	—	X	X	X	X
51	ja_Inventory	компонент инвентаризации СУБД	X	—	X	—	X	—
52	ja_tune	Генератор конфигурационного файла	—	—	—	—	X	—

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

№	Наименование	Описание	J4		J5		J6	
			Дист. ¹⁾	Обр.к. ²⁾	Дист. ¹⁾	Обр.к. ²⁾	Дист. ¹⁾	Обр.к. ²⁾
53	hunspell	Свободная библиотека для проверки орфографии и морфологического анализа. Компонент «hunspell»	—	—	X	X	X	X
54	ja_anonymizer	Маскирование данных	—	—	—	—	X	X
55	wal-g	Архивация и восстановление данных. Компонент «wal-g»	—	—	—	—	X	X

3. УСТАНОВКА КОНТЕЙНЕРА

Установка контейнера состоит из следующих шагов:

- Обновление ОС (п. 3.1);
- Установка платформы docker в ОС (п. 3.2);
- Импорт образа (п. 3.3);
- Установка разрешений на исполнения всех исполняемых файлов контейнере;
- Настройка переменных окружения (п. 3.4)
- Запуск скрипта логирования контейнера log (п. 4.6);
- Проверка созданного контейнера (п. 4.2).

3.1. Обновление ОС

Обновление ОС выполняется синхронизацией пакетов и обновлением их командами, описанными ниже.

```
sudo apt update
```

```
root@ubuntu: /
root@ubuntu:/# sudo apt update
Hit:1 http://ppa.launchpad.net/serge-rider/dbeaver-ce/ubuntu focal InRelease
Hit:2 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu focal InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:5 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
4 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ubuntu:/#
```

Рисунок 3.1 – Синхронизация пакетов

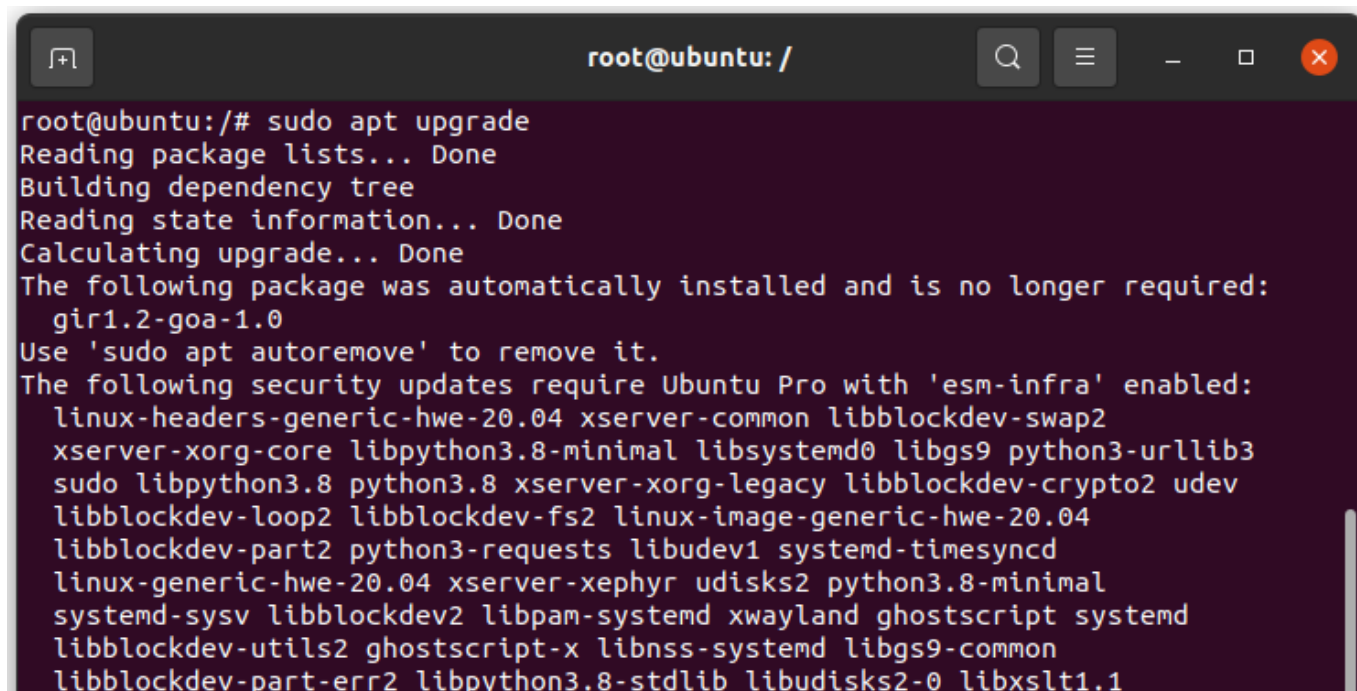
Команда «apt update» используется для синхронизации списков пакетов в вашей системе.

Она извлекает последние списки пакетов PPA и репозиториев в вашей системе и обеспечивает их актуальность.

Команда «apt upgrade» обновляет пакеты до последних версий и устанавливает новые пакеты, если они требуются в качестве зависимостей.

Он не удаляет никакие пакеты, а если какие-либо из них предназначены для удаления, он их пропускает.

```
sudo apt upgrade
```



```
root@ubuntu: /  
root@ubuntu:/# sudo apt upgrade  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
Calculating upgrade... Done  
The following package was automatically installed and is no longer required:  
  gir1.2-goa-1.0  
Use 'sudo apt autoremove' to remove it.  
The following security updates require Ubuntu Pro with 'esm-infra' enabled:  
linux-headers-generic-hwe-20.04 xserver-common libblockdev-swap2  
xserver-xorg-core libpython3.8-minimal libsystemd0 libgs9 python3-urllib3  
sudo libpython3.8 python3.8 xserver-xorg-legacy libblockdev-crypto2 udev  
libblockdev-loop2 libblockdev-fs2 linux-image-generic-hwe-20.04  
libblockdev-part2 python3-requests libudev1 systemd-timesyncd  
linux-generic-hwe-20.04 xserver-xephyr udisks2 python3.8-minimal  
systemd-sysv libblockdev2 libpam-systemd xwayland ghostscript systemd  
libblockdev-utils2 ghostscript-x libnss-systemd libgs9-common  
libblockdev-part-err2 libpython3.8-stdlib libudisks2-0 libxslt1.1
```

Рисунок 3.2 – Обновление пакетов ОС

3.2. Установка платформы docker в ОС

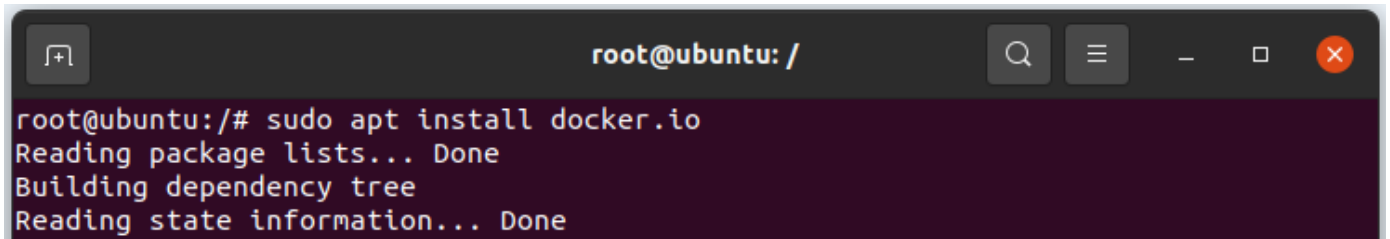
Платформа докера должна устанавливаться версии 20.10.12 или выше.



Подробная инструкция по установке находится на сайте разработчика по адресу: <https://docs.docker.com/engine/install/>

Альтернативный вариант (можно использовать для OS (DEB), которых нет в списке на сайте):

```
sudo apt install docker.io
```

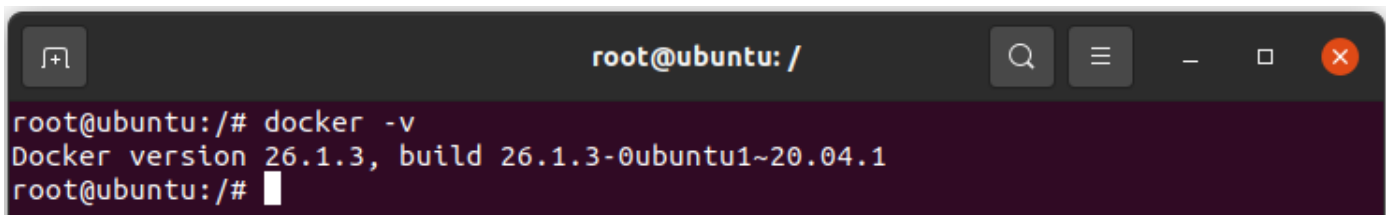


```
root@ubuntu: /
root@ubuntu:/# sudo apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Рисунок 3.3 - Установка платформы docker в ОС

Версия установленной платформы docker проверяется командой:

```
docker -v
```



```
root@ubuntu: /
root@ubuntu:/# docker -v
Docker version 26.1.3, build 26.1.3-0ubuntu1~20.04.1
root@ubuntu:/#
```

Рисунок 3.4 – Проверка версии установленной платформы docker

3.3. Импорт образа

Образ поставляется в виде архива, который необходимо импортировать в локальный набор образов. Для импорта образа надо исполнить скрипт `setup.sh`, который входит в комплект.

Необходимо дать скрипту право на выполнение командой:

```
chmod +x setup.sh
```

```

root@ubuntu: /home/admin1/Downloads/container
root@ubuntu: // # cd /home/admin1/Downloads/container/
root@ubuntu: /home/admin1/Downloads/container # chmod +x setup.sh
root@ubuntu: /home/admin1/Downloads/container # ls -la
total 433616
drwxrwxr-x 3 admin1 admin1      4096 Jul 10 03:21 .
drwxr-xr-x 3 admin1 admin1      4096 Jul 10 02:52 ..
-rwxrwx-rw- 1 admin1 admin1      546 Jul 10 02:26 .env
-rwxrwx-rw- 1 admin1 admin1 443907072 Jul 10 02:26 jatoba6_6.8.4-57423-df1.6.1-as
tralinux1.8.tar
-rwxrwx-rw- 1 admin1 admin1      582 Jul 10 02:26 log.sh
-rwxrwx-rw- 1 admin1 admin1    29275 Jul 10 02:26 postgresql.template.conf
-rwxrwx-rw- 1 admin1 admin1    31821 Jul 10 02:26 README.md
-rwxrwx-rw- 1 admin1 admin1      631 Jul 10 02:26 remove.sh
-rwxrwx-rw- 1 admin1 admin1      584 Jul 10 02:26 restart.sh
-rwxrwx-rw- 1 admin1 admin1    1985 Jul 10 02:26 run.sh
-rwxrwxrwx 1 admin1 admin1    2482 Jul 10 02:26 setup.sh
drwxrwxrwx 2 admin1 admin1      4096 Jul 10 02:27 standalone
-rwxrwx-rw- 1 admin1 admin1      574 Jul 10 02:26 start.sh
-rwxrwx-rw- 1 admin1 admin1      573 Jul 10 02:26 stop.sh
root@ubuntu: /home/admin1/Downloads/container #

```

Рисунок 3.5 – Выполнение команды chmod

Запустить установку образа:

```
./setup.sh
```

```

root@ubuntu: /home/admin1/Downloads/container
root@ubuntu: /home/admin1/Downloads/container # ./setup.sh
Settings enviroment
Setup Jatoba DB
Loading image jatoba6:6.8.4-57423-df1.6.1-astralinux1.8
b53346d3f176: Loading layer 165.1MB/165.1MB
3bb186834248: Loading layer 3.072kB/3.072kB
a08b9267a36e: Loading layer 8.192kB/8.192kB
810555838399: Loading layer 3.792MB/3.792MB
5b72097d9e67: Loading layer 13.82kB/13.82kB
ae654e2bdeef: Loading layer 25.98MB/25.98MB
8379f951702d: Loading layer 10.84MB/10.84MB
50c25ed17853: Loading layer 2.048kB/2.048kB
20a5731df9ec: Loading layer 1.246MB/1.246MB
dd41ad832f49: Loading layer 3.584kB/3.584kB
a29d32b25d6f: Loading layer 236.7MB/236.7MB
30bd8055d435: Loading layer 64kB/64kB
4aa1c3f2fc1d: Loading layer 2.048kB/2.048kB
561a3bb31e16: Loading layer 3.584kB/3.584kB
d8b40421b2d1: Loading layer 3.072kB/3.072kB
de081c6f3fa1: Loading layer 61.44kB/61.44kB
3b2072b1fb6b: Loading layer 61.44kB/61.44kB
Loaded image: jatoba6:6.8.4-57423-df1.6.1-astralinux1.8
root@ubuntu: /home/admin1/Downloads/container #

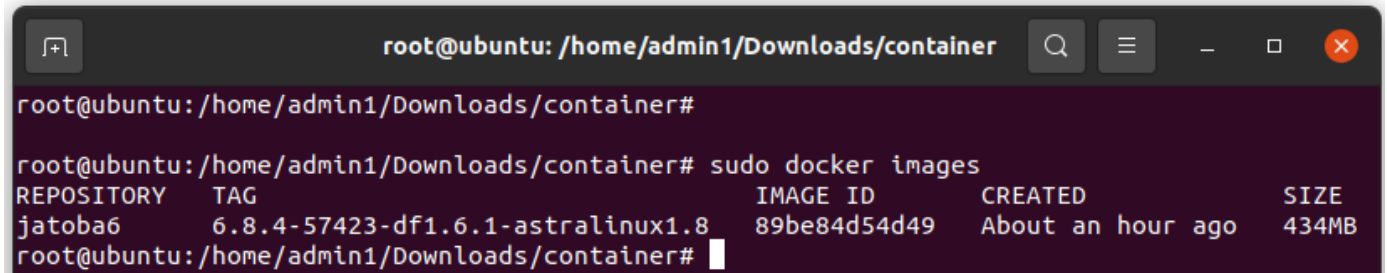
```

Рисунок 3.6 – Выполнение скрипта setup.sh

После выполнения скрипта образ будет загружен и готов для создания контейнеров.

Проверка наличия образа выполняется командой:

```
sudo docker images
```



```
root@ubuntu: /home/admin1/Downloads/container#  
root@ubuntu: /home/admin1/Downloads/container# sudo docker images  
REPOSITORY      TAG              IMAGE ID         CREATED          SIZE  
jatoba6          6.8.4-57423-df1.6.1-astralinux1.8  89be84d54d49    About an hour ago 434MB  
root@ubuntu: /home/admin1/Downloads/container#
```

Рисунок 3.7 – Вывод данных об образе

3.4. Переменные окружения

Переменные окружения, указанные в таблице 3.1, контейнер использует при инициализации БД.

Чтобы задать переменные окружения, необходимо указать значения параметров при запуске контейнера (`docker run`) в виде параметров (`-e <параметр>`) или в виде файла (`--env-file .env`). Шаблон файла `.env` есть в комплекте контейнера.

Таблица 3.1 – Переменные окружения

Название	Значение параметра	Необходимо указывать	Описание
# Common			
JATоба_VERSION	<ver.image>	—	Версия контейнера
PG_MAJOR	<ver>	—	Мажорная версия СУБД
IMAGE_NAME	jatoba	—	Имя образа
CONTAINER_NAME	jatobadb	—	Имя контейнера
# Need absolute path to file			
POSTGRESCONF_PATH	/path/to/postgresql.template.conf	—	Путь к шаблону конфигурационного файла
# For external postgres.conf			
PGCFGFILE_TIMEZONE	Europe/Moscow	X	Временная зона
# Volumes			
POSTGRESQL_VOLUME_DATA	postgresql-volume-data	—	Том с данными
POSTGRESQL_VOLUME_EXTENSIONS	postgresql-volume-extensions	—	Том с расширениям
# Account data			
POSTGRES_USER	postgres	—	Имя пользователя сервера БД. По умолчанию: postgres
POSTGRES_PASSWORD	mysecretpassword	X	Пользовательский пароль для подключения к системной БД
# setup.sh script (show additional docker layers information)			
SHOW_DOCKER_OVERLAY_LAYERS_TABLE	false	—	Вывод сопоставления слоёв контейне

3.4.1. Запуск нового контейнера

! Изменить версию СУБД при использовании текущей версии контейнера невозможно.

В данном разделе описаны действия, которые необходимо сделать администратору, чтобы контейнер корректно запустился и инициализировал БД.

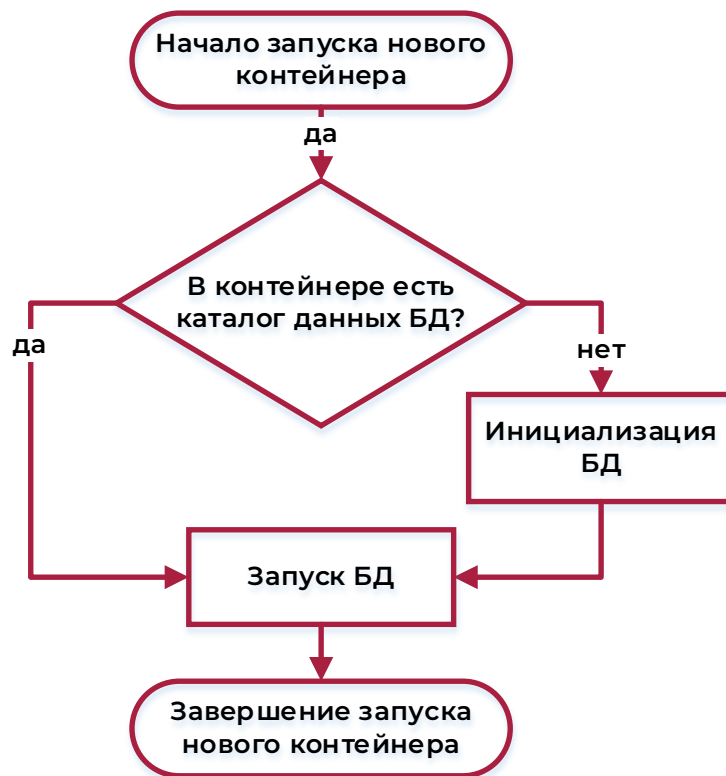


Рисунок 3.8 – Проверки, которые контейнер делает после запуска

На рисунке 3.8 отображен процесс запуска контейнера: какие проверки происходят после запуска, и в каких случаях могут происходить ошибки. На этой схеме видно, что при запуске новый контейнер может принимать часть данных от старого контейнера, такую как каталог данных БД (см. подраздел 4.9).

В качестве шаблона конфигурационного файла контейнер использует файл «postgresql.template.conf», содержащийся в комплекте. Однако пользователь может переопределить этот файл при запуске контейнера (см. подраздел 4.9).

Запустить контейнер можно с помощью команды «docker run» (см. подраздел 4.8) или с помощью скрипта run.sh (см. подраздел 3.4.2).

3.4.2. Запуск контейнера

Для запуска контейнера СУБД «Jatoba» необходимо выполнить последовательность действий, приведенную ниже.

1. Проверить параметры в файле .env, находящийся в текущем каталоге:

```
# cd /home/admin1/Downloads/container/  
# nano .env
```

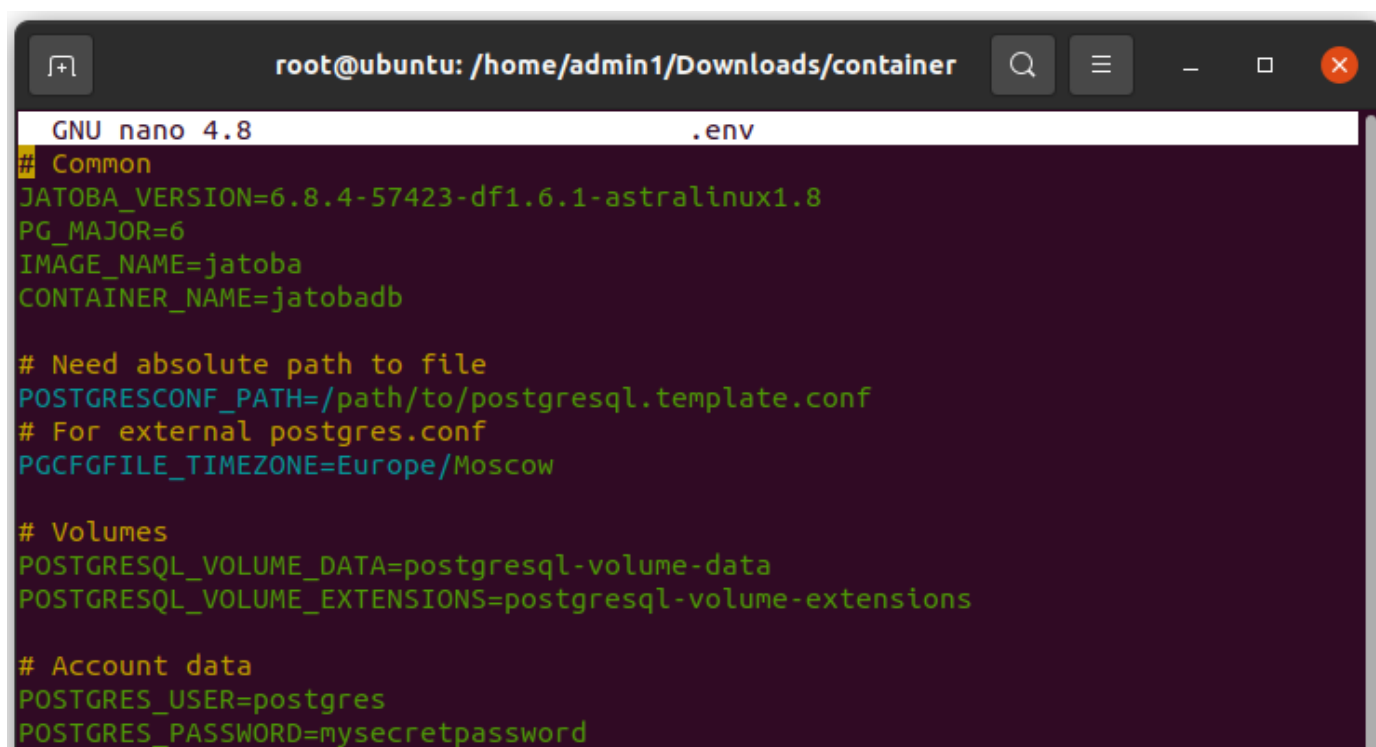


Рисунок 3.9 – Файл «.env»

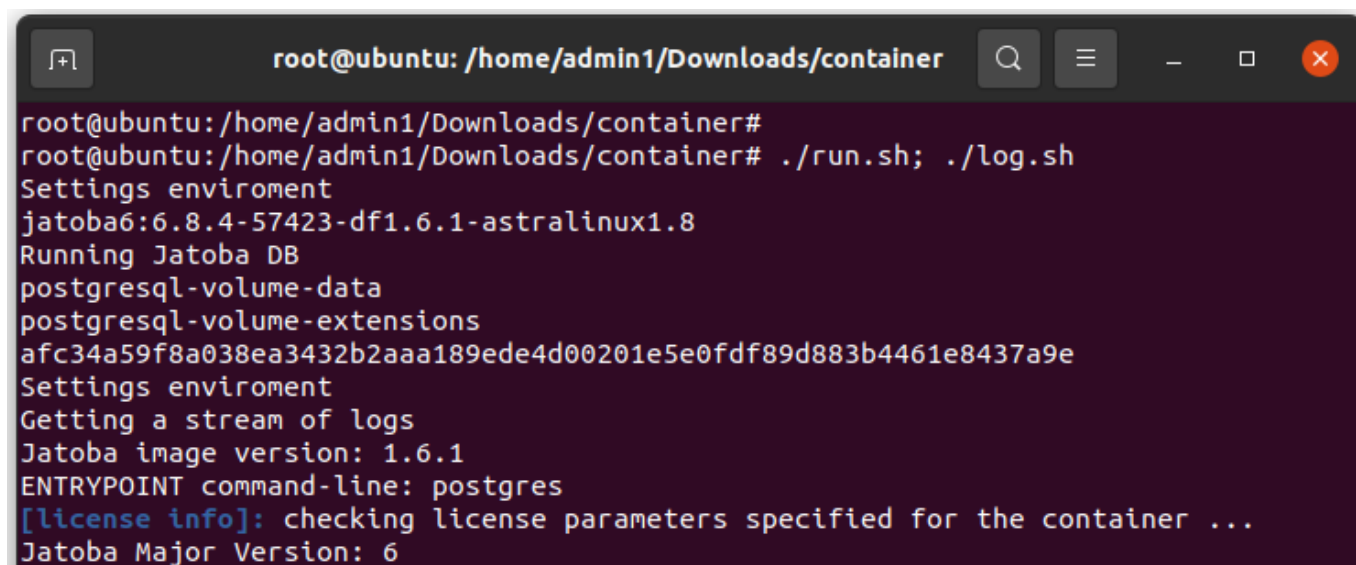
Сохранить внесенные изменения. Перемещать, переименовывать файл .env не требуется.

2. Перейти в каталог с системой. Если ранее скриптам не был дан доступ на выполнение, необходимо выполнить команду:

```
chmod +x *.sh
```

3. Далее необходимо запустить контейнер и одновременно вывести журнал событий контейнера:

```
./run.sh; ./log.sh
```



```
root@ubuntu: /home/admin1/Downloads/container
root@ubuntu: /home/admin1/Downloads/container# ./run.sh; ./log.sh
Settings enviroment
jatoba6:6.8.4-57423-df1.6.1-astralinux1.8
Running Jatoba DB
postgresql-volume-data
postgresql-volume-extensions
afc34a59f8a038ea3432b2aaa189ede4d00201e5e0fdf89d883b4461e8437a9e
Settings enviroment
Getting a stream of logs
Jatoba image version: 1.6.1
ENTRYPOINT command-line: postgres
[license info]: checking license parameters specified for the container ...
Jatoba Major Version: 6
```

Рисунок 3.10 – Журнал событий контейнера после запуска скриптов «./run.sh; ./log.sh»

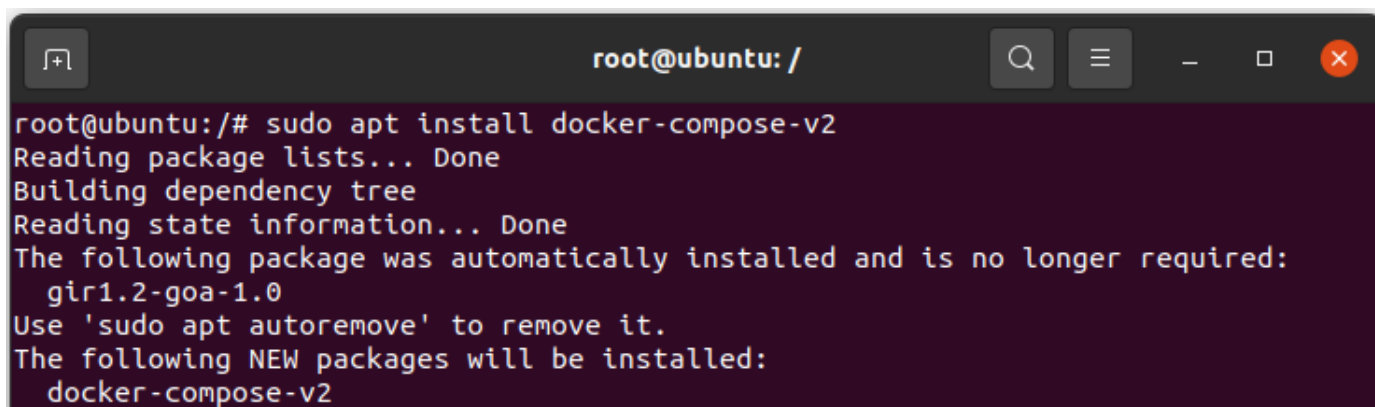
3.4.3. Запуск контейнера через docker compose

Запуск контейнера возможен после:

- импорта образа, как описано в п.п. 3.3, настоящего документа;
- Установленного docker compose plugin версии 2.20.2;

Либо установить docker compose plugin командами в терминале ОС:

```
sudo apt install docker-compose-v2
docker compose version
```



```
root@ubuntu: /
root@ubuntu: /# sudo apt install docker-compose-v2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  gir1.2-goa-1.0
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  docker-compose-v2
```

Рисунок 3.11 – Установка docker compose plugin

Далее выполняются следующие действия:

Перейти в директорию standalone.

В файле .env задать надёжный пароль для пользователя postgres в переменной

```
POSTGRES_PASSWORD
```

Собрать и запустить контейнер:

```
docker compose up -d
```

Проверить, что СУБД успешно запустилось:

```
docker ps
```

Проверить лог контейнера:

```
docker logs jatobadb
```

Проверить volume:

```
docker volume list
```

Проверить подключение к контейнеру:

```
docker exec -it jatobadb /bin/bash
```

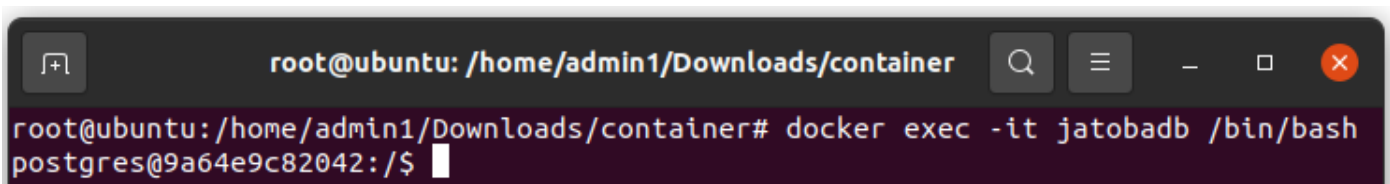


Рисунок 3.12 – Вывод команды проверки подключения к контейнеру

На данном этапе пользователь находится внутри встроенной операционной системы, от имени и с правами пользователя данной ОС «postgres», в которой доступен минимальный набор команд. Например, вывод списка каталогов.

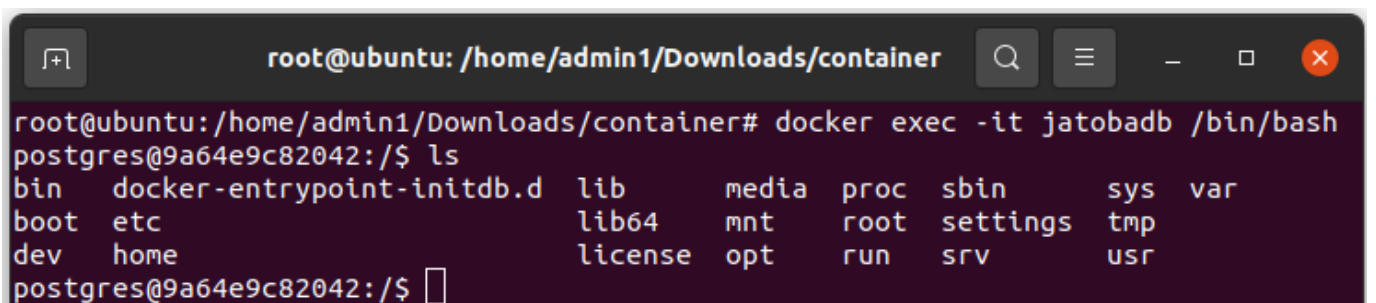
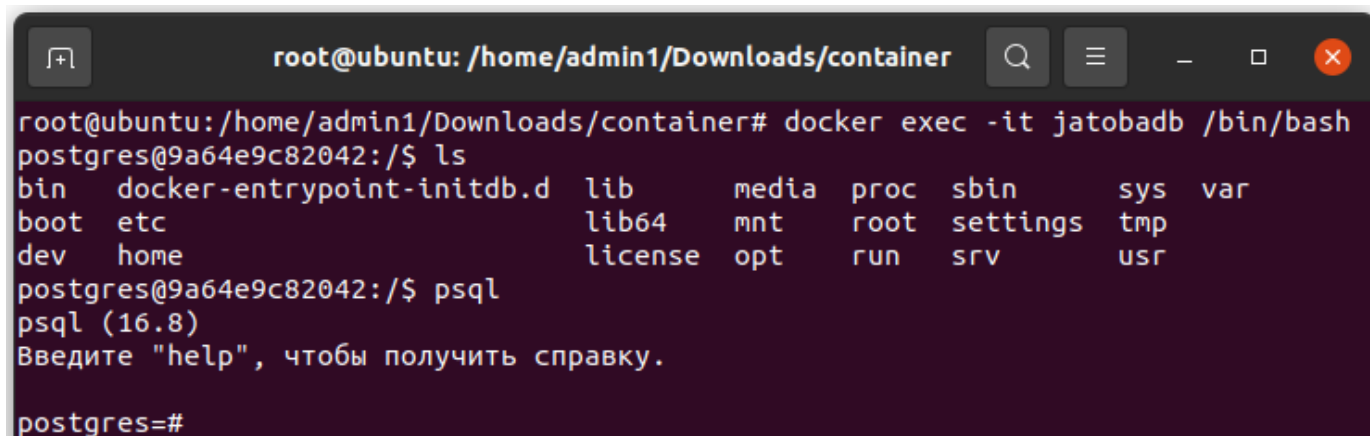


Рисунок 3.13 - Вывод списка каталогов

Проверить подключение к СУБД, не выходя из контейнера:

```
psql
```



```
root@ubuntu: /home/admin1/Downloads/container
root@ubuntu: /home/admin1/Downloads/container# docker exec -it jatobadb /bin/bash
postgres@9a64e9c82042:/$ ls
bin  docker-entrypoint-initdb.d  lib      media  proc  sbin      sys  var
boot etc                        lib64    mnt    root  settings tmp
dev  home                      license  opt    run   srv       usr
postgres@9a64e9c82042:/$ psql
psql (16.8)
Введите "help", чтобы получить справку.

postgres=#
```

Рисунок 3.14 – Запуск СУБД «Jatoba»

или, если уже отключились от контейнера на предыдущем шаге:

```
docker exec -it jatobadb psql -U postgres
```

4. ОСНОВНЫЕ ОПЕРАЦИИ С КОНТЕЙНЕРОМ

В комплекте контейнера расположены скрипты, упрощающие работу с контейнерами. Скрипты должны находиться в одном каталоге с файлом `.env`. Имя контейнера, с которым работают скрипты, указывается в `.env` параметром `CONTAINER_NAME`.

`.env` – файл, содержащий набор переменных окружения. Переменные окружения описаны в подразделе 3.4.

Описание скриптов:

- `remove.sh` – удаляет контейнер (см. подраздел 4.7);



Если каталог `data` не был примонтирован к хостовому серверу при инициализации контейнера, то при использовании скрипта `remove.sh` данные удалятся. Если каталог был примонтирован, то данные при запуске `remove.sh` сохранятся.

- `stop.sh` – останавливает контейнер (см. подраздел 4.3);
- `run.sh` – запускает новый контейнер (см. п. 3.4.2);
- `start.sh` – запускает остановленный (но не удаленный) контейнер (см. п. 4.4);
- `restart.sh` - используется для перезапуска контейнера (см. подраздел 4.5);
- `stop.sh` - используется для остановки контейнера (см. подраздел 4.3);
- `log.sh` – запускает просмотр журнала событий БД (см. подраздел 4.6);
- `remove.sh` – удаляет контейнер (см. подраздел 4.7);

Чтобы выполнить скрипт, надо дать ему право на выполнение:

```
chmod +x <название скрипта>.sh
```

И запустить:

```
./<название скрипта>.sh
```

4.1. Проверка статуса (`docker ps -a`)

Проверка статуса контейнера выполняется из терминала ОС командой:

```
# docker ps -a
```

```

root@ubuntu: /home/admin1/Downloads/container
root@ubuntu:/home/admin1/Downloads/container# docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS          NAMES
afc34a59f8a0   jatoba6:6.8.4-57423-df1.6.1-astralinux1.8  "docker-entrypoint.s..."  10 minutes ago  Exited (0) 3 minutes ago          jatobadb
root@ubuntu:/home/admin1/Downloads/container#

```

Рисунок 4.1 – Вывод статуса контейнера

В выводе статуса будут отражены столбцы:

- CONTAINER ID – идентификатор контейнера;
- IMAGE – имя образа, из которого был поднят контейнер;
- COMMAND - команда, которая выполнялась при старте контейнера
- CREATED – время с момента создания контейнера;
- STATUS – дни работы контейнера (текущее состояние);
- PORTS – используемые порты и адресация сети (проброс портов);
- NAMES – имя контейнера.

4.2. Проверка наличие созданных томов (volume list)

Проверка созданных томов выполняется из терминала ОС командой:

```
# docker volume list
```

```

root@ubuntu: /home/admin1/Downloads/container
root@ubuntu:/home/admin1/Downloads/container# docker volume list
DRIVER      VOLUME NAME
local       4facebb0ff8db729cdb28db706d93c0c98200b555746c4db5d8b1db91d639b57
local       e0e00109b5dfaff8aa23c60ef077a4095ab30f035c9befa8b1d575e1a8a9c44c
local       postgresql-volume-data
local       postgresql-volume-extensions
root@ubuntu:/home/admin1/Downloads/container#

```

Рисунок 4.2 – Вывод команды проверки созданных томов

4.3. Остановка контейнера (stop.sh)

Остановка контейнера выполняется скриптом «stop.sh», расположенным в домашней директории контейнера.

Скрипт «stop.sh» запускается командой из терминала ОС:

```
./stop.sh
```

```

root@ubuntu: /home/admin1/Downloads/container
root@ubuntu:/home/admin1/Downloads/container# ./stop.sh
Settings enviroment
Stopping Jatoba DB
jatobadb
root@ubuntu:/home/admin1/Downloads/container#

```

Рисунок 4.3 – Команда остановки контейнера

При проверке статуса контейнера в столбце «Status» значение «Exited» с указанием истекшего времени с момента остановки контейнера.

```
docker ps -a
```

```

root@ubuntu: /home/admin1/Downloads/container
root@ubuntu:/home/admin1/Downloads/container# ./stop.sh
Settings enviroment
Stopping Jatoba DB
jatobadb
root@ubuntu:/home/admin1/Downloads/container# docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS              PORTS          NAMES
781a26a63960   jatoba5:5.5.3-54782-df1.3.2-astralinux2.12  "docker-entrypoint.s..."  6 days ago    Exited (130) About a minute ago          jatobadb
root@ubuntu:/home/admin1/Downloads/container#

```

Рисунок 4.4 – Вывод статуса остановленного контейнера

4.4. Запуск остановленного контейнера (start.sh)

Запуск контейнера выполняется скриптом «start.sh», командой из корневого каталога контейнера в терминале ОС:

```
./start.sh
```

```

root@ubuntu: /home/admin1/Downloads/container
root@ubuntu:/home/admin1/Downloads/container# ./start.sh
Settings enviroment
Starting Jatoba DB
jatobadb
root@ubuntu:/home/admin1/Downloads/container#

```

Рисунок 4.5 – Команда запуска контейнера

4.5. Перезапуск контейнера (restart.sh)

Перезапуск контейнера выполняется скриптом «restart.sh», командой из корневого каталога контейнера в терминале ОС:

```
./restart.sh
```

```

root@ubuntu: /home/admin1/Downloads/container
root@ubuntu: /home/admin1/Downloads/container# ./restart.sh
Settings enviroment
Restarting Jatoba DB
jatobadb
root@ubuntu: /home/admin1/Downloads/container#

```

Рисунок 4.6 – Выполнение команды перезапуска контейнера

4.6. Логирование контейнера (log.sh)

Скрипт log.sh может запускаться самостоятельно, либо совместно со скриптами:

- run.sh (п. 3.4.2);

```
./run.sh; ./log.sh
```

- start.sh (п. 4.4).

```
./start.sh; ./log.sh
```

При запущенном контейнере скрипт «log.sh» выведет журнал событий контейнера:

```
./log.sh
```

```

root@ubuntu: /home/admin1/Downloads/container
root@ubuntu: /home/admin1/Downloads/container# ./log.sh
Settings enviroment
Getting a stream of logs
Jatoba image version: 1.6.1
ENTRYPOINT command-line: postgres
[license info]: checking license parameters specified for the container ...
Jatoba Major Version: 6
List available locales: C C.utf8 en_US.utf8 POSIX ru_RU.utf8
Initializing database ... initdb: предупреждение: включение метода аутентификации

```

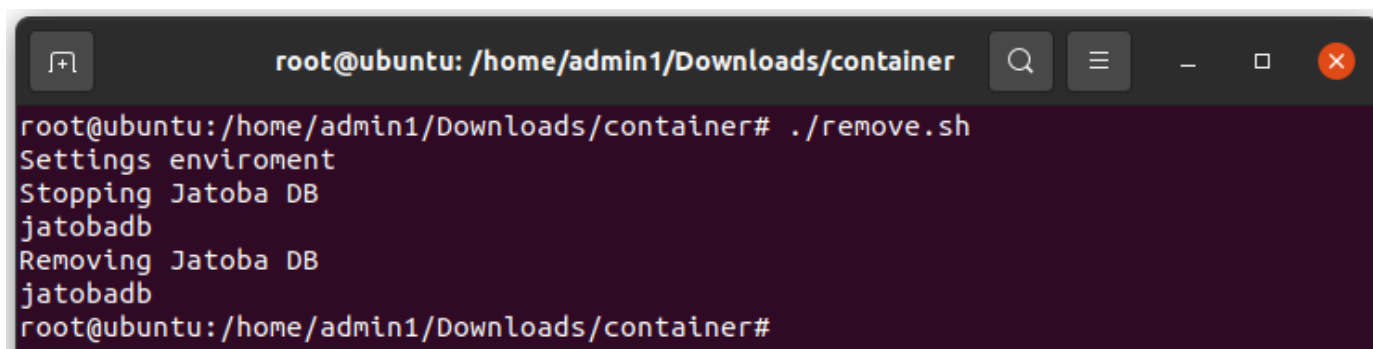
Рисунок 4.7 – Вывод скрипта логирования

4.7. Удаление контейнера (remove.sh)

Удаление контейнера должно выполняться по следующим шагам:

- Выполнить скрипт удаления контейнера:

```
./remove.sh
```

```
root@ubuntu: /home/admin1/Downloads/container
root@ubuntu:/home/admin1/Downloads/container# ./remove.sh
Settings enviroment
Stopping Jatoba DB
jatobadb
Removing Jatoba DB
jatobadb
root@ubuntu:/home/admin1/Downloads/container#
```

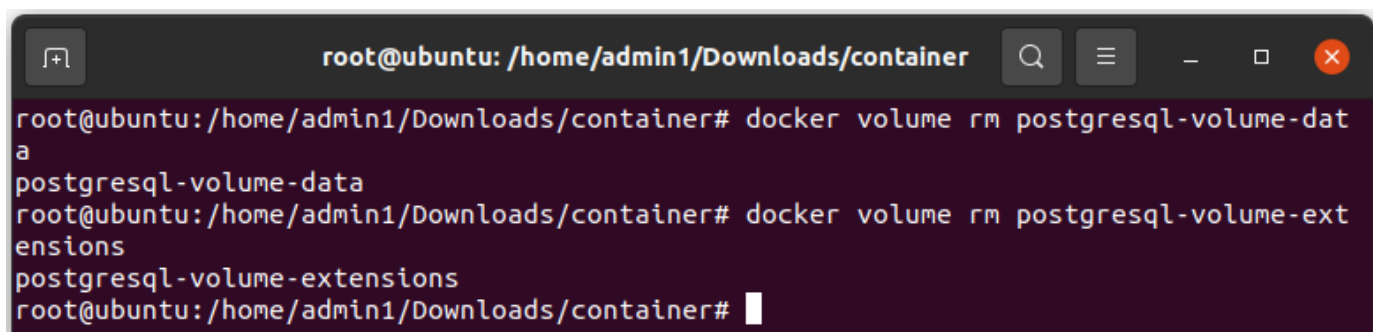
Рисунок 4.8 – Удаление контейнера

- Удалить том с данными:

```
docker volume rm postgresql-volume-data
```

- Удалить том с расширениями СУБД:

```
docker volume rm postgresql-volume-extensions
```

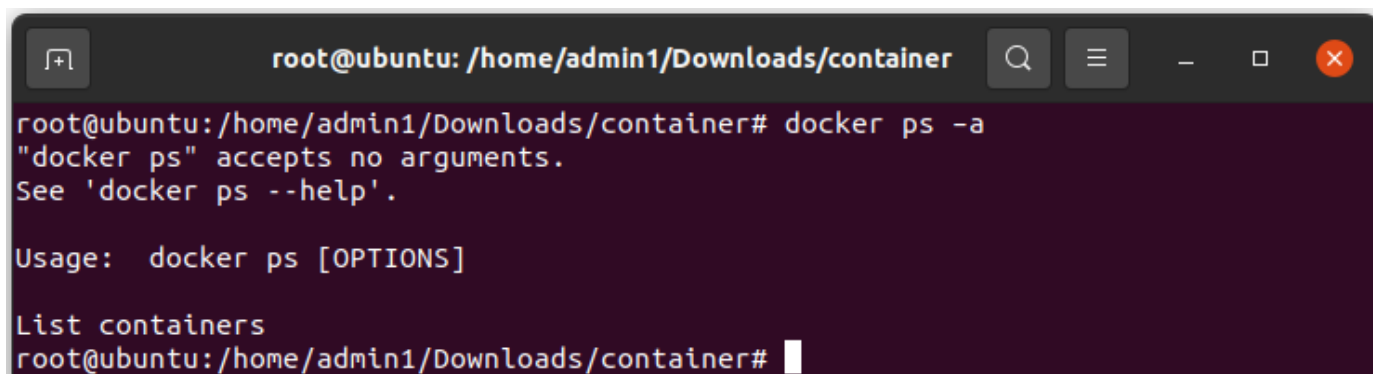


```
root@ubuntu: /home/admin1/Downloads/container
root@ubuntu:/home/admin1/Downloads/container# docker volume rm postgresql-volume-dat
a
postgresql-volume-data
root@ubuntu:/home/admin1/Downloads/container# docker volume rm postgresql-volume-ext
ensions
postgresql-volume-extensions
root@ubuntu:/home/admin1/Downloads/container#
```

Рисунок 4.9 – Команды удаления томов с данными и расширениями СУБД

- Проверить отсутствие контейнера в списке:

```
docker ps -a
```



```
root@ubuntu: /home/admin1/Downloads/container
root@ubuntu:/home/admin1/Downloads/container# docker ps -a
"docker ps" accepts no arguments.
See 'docker ps --help'.

Usage:  docker ps [OPTIONS]

List containers
root@ubuntu:/home/admin1/Downloads/container#
```

Рисунок 4.10 – Вывод статуса контейнера

4.8. Запуск контейнера через docker run

Для запуска контейнера СУБД «Jatoba» необходимо выполнить команду :

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

```
docker run <параметры> jatoba*:*.*. *-****-df1.*.*
```

Примеры команд запуска контейнера есть в разделе 6.

Параметры `docker run` для запуска контейнера СУБД «Jatoba»:

- `--user` – запуск контейнера от имени пользователя



Параметр `--user` активирует режим «non-root», т.е. запуск контейнера не от привилегированного пользователя.

Данный параметр прописан в скрипте «run.sh», однако при игнорировании его контейнер запустится от пользователя «root».

- `-i` – параметр, который делает возможным взаимодействие с терминалом внутри контейнера;

- `-t` – параметр, который предоставляет доступ к терминалу внутри контейнера;



Параметры `-t` и `-i` обычно используются в паре


- `-d` – параметр для запуска контейнера в фоновом режиме. При запуске контейнера с использованием этого параметра контейнер не остановится при закрытии консоли;

- `-v <название созданного docker-volume>:<путь до каталога внутри контейнера>` – параметр, позволяющий сохранять каталог контейнера на хостовом сервере. При использовании этого параметра данные контейнера сохраняются при его удалении (см. подраздел 4.9);

- `-e <параметр>=<значение параметра>` – параметр, который задает переменную окружения для контейнера;

- `--env-file <путь к файлу>` – параметр, который задает путь к файлу, содержащему переменные окружения для контейнера;

- `-p <внешний порт>:5432` – параметр, который сопоставляет порт внутри контейнера с портом на хостовом сервере — таким образом, к БД в контейнере можно обратиться через указанный порт на хостовом сервере;

 Если на хостовом сервере уже запущен процесс, использующий порт 5432, можно поменять порт для контейнера, например, на 54321 (-p 54321:5432)

- --name – имя запускаемого контейнера.

При успешном создании контейнера появится его идентификатор.


По имени или идентификатору контейнера можно получить журнал событий и убедиться, что сервер БД корректно запустился. Команда для просмотра журнала событий:

```
docker logs <имя контейнера>
```

Либо для отслеживания потока вывода журнала событий:

```
docker logs -f <имя контейнера>
```

4.9. Сохранение данных контейнера

 При запуске контейнера с использованием скрипта run.sh сохранять данные не требуется, это делает скрипт.

Перед запуском нового контейнера рекомендуется создать docker-volume для каталога данных сервера БД. Иначе в случае удаления контейнера эти данные будут потеряны.

Чтобы создать docker-volume для сохранения каталога данных сервера БД (каталога /var/lib/jatoba/<ver>/data), выполнить команду:

```
docker volume create postgresql-volume-data
```

Для использования этого docker-volume в контейнере, при запуске контейнера с помощью команды docker run (см. подраздел 4.8) необходимо добавить параметр -v postgresql-volume-data:/var/lib/jatoba/<ver>/data.

5. ИЗМЕНЕНИЕ ПОЛЬЗОВАТЕЛЬСКИХ НАСТРОЕК POSTGRESQL.CONF

В комплекте контейнера есть файл конфигурации БД (postgresql.template.conf), который используется в качестве шаблона при настройке БД. Но его можно переопределить через монтирование нового файла конфигурации на хостовой сервер или через перенос файла в контейнер:

- Монтирование файла. При использовании этого способа файл конфигурации сохранится на хостовом сервере в случае удаления контейнера.

Для этого при запуске контейнера (docker run) указать параметр:

```
-v <путь до postgresql.conf на хостовом сервере>:/settings/postgresql.conf
```

- Перенос файла. При использовании этого способа файл конфигурации не сохранится при удалении контейнера.

В каталог /settings, находящийся внутри контейнера, поместить свой файл конфигурации с именем postgresql.conf и перезапустить контейнер. Это можно сделать с помощью команды:

```
docker cp <путь до postgresql.conf на хостовом сервере>  
<имя контейнера>:/settings
```

Затем выполнить команду:

```
docker restart <имя контейнера>
```

Шаблонизация параметров

В пользовательском файле настроек поддерживается шаблонизация параметров. Например, если при создании контейнера задать переменную окружения PGCFGFILE_TIMEZONE, то ее можно указать в пользовательском файле настроек: log_timezone= {{PGCFGFILE_TIMEZONE}}.

Если в файле конфигурации было шаблонизировано значение, но соответствующей переменной окружения не существует, контейнер будет выдавать ошибку. В таком случае необходимо или указать значение этого параметра в качестве переменной окружения при

запуске, или изменить файл конфигурации, чтобы в нем было явно указано значение параметра.

6. ПРИМЕРЫ КОМАНД ЗАПУСКА КОНТЕЙНЕРА

1) С созданием docker-volume.

Выполнить команду создания docker-volume для сохранения каталога данных сервера БД:

```
docker volume create postgresql-volume-data
```

Выполнить команду запуска контейнера:

```
docker run -it -d \  
-v postgresql-volume-data:/var/lib/jatoba/4/data \  
-e POSTGRES_PASSWORD=mysecretpassword \  
--user postgres \  
--name jatobadb jatoba4:4.8.2-1124-df1.2.1
```

В данном примере в интерактивном режиме создается контейнер с примонтированными docker-volume для каталога данных БД. Доступ к БД есть с хостового сервера по порту 5432. Имя контейнера – jatobadb.

2) Без создания docker-volume.

Для запуска контейнера выполнить команду:

```
docker run -it -d \  
-e POSTGRES_HOST_AUTH_METHOD= trust -e POSTGRES_USER=admin \  
--user postgres \  
-p 5432:5432 --name jatobadb jatoba4:4.8.2-1124-df1.2.1
```

В данном примере создается контейнер jatobadb. Доступ к БД у пользователя admin, пароль для подключения не используется. После удаления контейнера данные не сохраняются, потому что docker-volume не были примонтированы (см. подраздел 4.9).

7. ИСПОЛЬЗОВАНИЕ УТИЛИТЫ PSQL


psql – это терминальный интерфейс, использующийся для работы с СУБД «Jatoba». Он позволяет вводить запросы в интерактивном режиме, отправлять их в СУБД и просматривать результат выполнения запросов.

Для подключения к СУБД «Jatoba» с сервера, на котором развернут и запущен контейнер, можно использовать команду:

```
docker exec -it <название контейнера> psql -U <имя пользователя БД>
```

Здесь:

- `docker exec -it` – команда запуска процесса в контейнере, `-it` обозначает интерактивный режим выполнения (позволяет использовать оболочку командной строки внутри контейнера и выполнять из-под нее команды);
- `<название контейнера>` – имя контейнера СУБД «Jatoba», указанное при запуске;
- `psql` – утилита, запускаемая внутри контейнера;
- `-U <имя пользователя БД>` – параметр утилиты `psql`, который обозначает, что `psql` будет подключаться к СУБД «Jatoba» под указанным пользователем БД. Необходимо использовать имя, указанное в параметре `POSTGRES_USER` файла `.env`.

 По умолчанию, если в файле `.env` не был указан `POSTGRES_USER`, при инициализации создается пользователь `postgres`.

Пример использования команды:

```
docker exec -it jatobadb psql -U postgres
```

В данном примере подключение идет к контейнеру с названием `jatobadb`, в нем в интерактивном режиме запускается утилита `psql` под пользователем `postgres`.

```
root@ubuntu: /home/admin1/Downloads/container
root@ubuntu:/home/admin1/Downloads/container# docker exec -it jatobadb psql -U postgres
psql (16.8)
Введите "help", чтобы получить справку.

postgres=# dx
postgres=# \dx
                Список установленных расширений
  Имя   | Версия | Схема   | Описание
-----+-----+-----+-----
plpgsql | 1.0    | pg_catalog | PL/pgSQL procedural language
(1 строка)

postgres=#
```

Рисунок 7.1 - Запуск утилиты psql в контейнере

8. СОПОСТАВЛЕНИЕ СЛОЕВ КОНТЕЙНЕРА С ДИРЕКТОРИЯМИ

При импорте в локальное хранилище образа контейнера докер именует директории содержащие слои распакованных образов случайным образом. В силу данной причины для сопоставления слоёв образа директориям на хосте содержащим распакованные слои используется скрипт setup.sh.

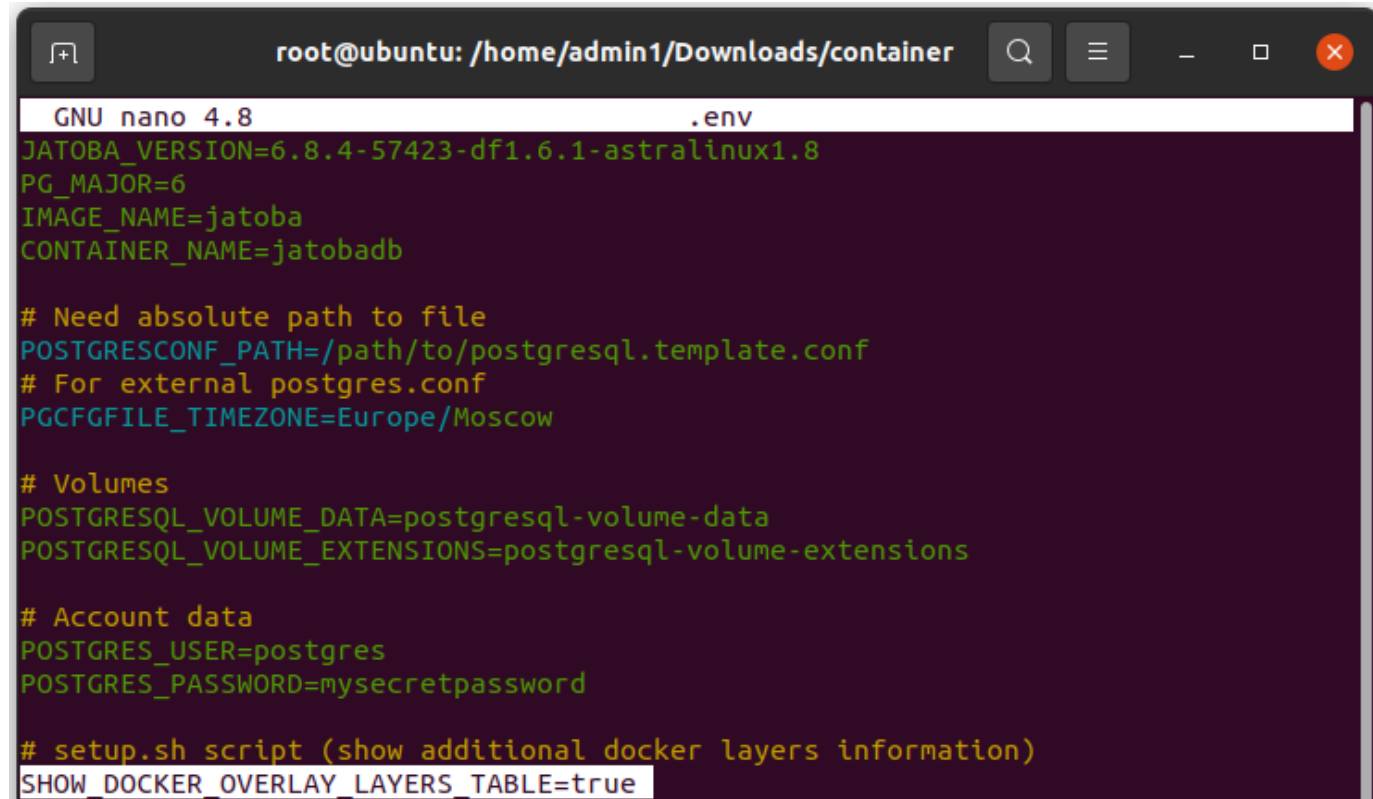
Перед использованием скрипта требуется

- установить утилиту командной строки для работы с JSON-данными jq командой:

```
apt install jq
```

- чтобы была установлена платформа docker в ОС (см. п.п. 3.2);
- образ, залитый в локальное хранилище образов;
- в файле переменных окружения установлен параметр:

```
# setup.sh script (show additional docker layers information)
SHOW_DOCKER_OVERLAY_LAYERS_TABLE=true
```



The screenshot shows a terminal window titled 'root@ubuntu: /home/admin1/Downloads/container'. Inside, the nano 4.8 editor is open to a file named '.env'. The file contains the following configuration:

```
GNU nano 4.8 .env
JATOBA_VERSION=6.8.4-57423-df1.6.1-astralinux1.8
PG_MAJOR=6
IMAGE_NAME=jatoba
CONTAINER_NAME=jatobadb

# Need absolute path to file
PGRESCONF_PATH=/path/to/postgresql.template.conf
# For external postgres.conf
PGCFGFILE_TIMEZONE=Europe/Moscow

# Volumes
PGRESQL_VOLUME_DATA=postgresql-volume-data
PGRESQL_VOLUME_EXTENSIONS=postgresql-volume-extensions

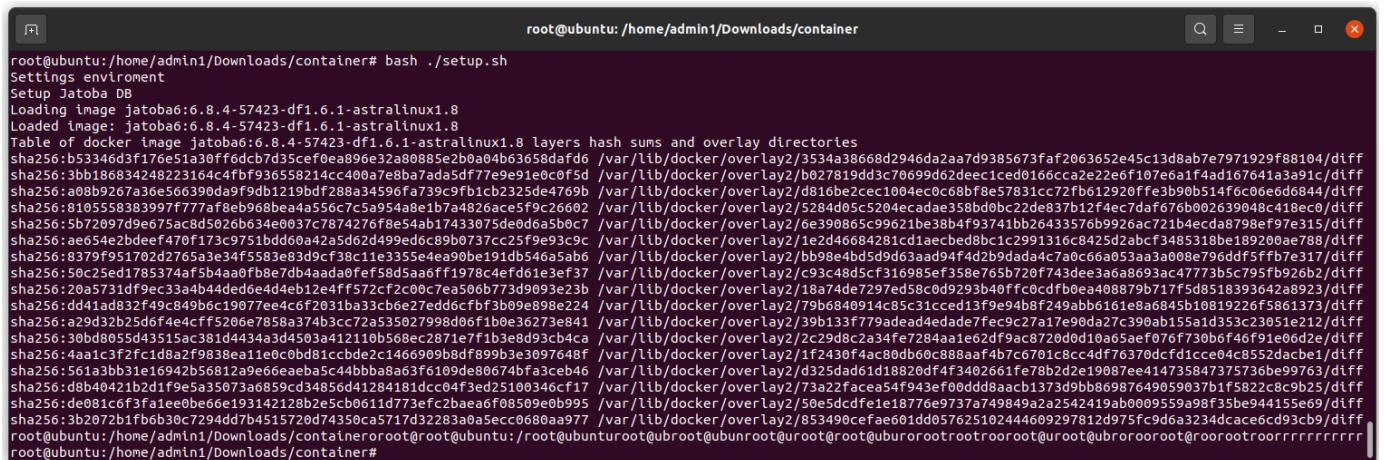
# Account data
PGRES_USER=postgres
PGRES_PASSWORD=mysecretpassword

# setup.sh script (show additional docker layers information)
SHOW_DOCKER_OVERLAY_LAYERS_TABLE=true
```

Рисунок 8.1 – Параметр «SHOW_DOCKER_OVERLAY_LAYERS_TABLE» в файле переменных окружения

Скрипт запускается командой:

```
bash ./setup.sh
```



```
root@ubuntu: /home/admin1/Downloads/container# bash ./setup.sh
Settings environment
Setup Jatoba DB
Loading image jatoba6:6.8.4-57423-df1.6.1-astralinux1.8
Loaded image: jatoba6:6.8.4-57423-df1.6.1-astralinux1.8
Table of docker image jatoba6:6.8.4-57423-df1.6.1-astralinux1.8 layers hash sums and overlay directories
sha256:b53346d3f176e51a30ff6dcdb7d35cef0ea896e32a80885e2b0a04b63658dafd6 /var/lib/docker/overlay2/3534a38668d2946da2aa7d9385673faf2063652e45c13d8ab7e7971929f88104/diff
sha256:3bb186834248223164c4fbf936558214cc400a7e8ba7ada5df77e9e91e0c0f5d /var/lib/docker/overlay2/b027819dd3c70699d62deec1ced0166cca2e22e6f107e6a1f4ad167641a3a91c/diff
sha256:a08b9267a36e566390da9f9db1219bdf288a34596fa739c9fb1cb2325de4769b /var/lib/docker/overlay2/d816be2cec1004ec0c68bf8e57831cc72fb612920ffe3b90b514f6c06e6d6844/diff
sha256:8105558383997f777af8eb968bea4a556c7c5a954a8e1b7a4826ace5f9c26602 /var/lib/docker/overlay2/5284d05c5204ecadae358bd0bc22de837b12f4ec7daf676b002639048c418ec0/diff
sha256:5b72097d9e675ac8d5026b634e0037c7874276f8e54ab17433075de0d6a5b0c7 /var/lib/docker/overlay2/6e390865c99621be38b4f93741bb26433576b9926ac721b4ecda8798ef97e315/diff
sha256:ae654e2bdeef470f173c9751bdd60a42a5d62d499ed6c89b0737cc25f9e93c9c /var/lib/docker/overlay2/1e2d46684281cd1aecbed8bc1c2991316c8425d2abc3485318be189200ae788/diff
sha256:8379f951702d765a3e34f5583e83d9c38c11e3355e4ea90be191db546a5ab6 /var/lib/docker/overlay2/bb98e4bd5d9d63aad94f4d2b9dada4c7a0c66a053aa3a008e796ddf5ff7e317/diff
sha256:50c25ed1785374af5b4aa0fb8e7db4aada0fef58d5aa6ff1978c4ef6d1e3ef37 /var/lib/docker/overlay2/c93c48d5cf316985ef358e765b720f743dee3a6a8693ac47773b5c795fb926b2/diff
sha256:20a5731df9ec33a4b44ded6e44eb12e4ff572cf2c00c7ea506b773d9093e23b /var/lib/docker/overlay2/18a74de7297ed58c0d9293b40ffc0cdfb0ea408879b717f5d8518393642a8923/diff
sha256:dd41ad832f49c849b6c19077ee4c6f2031ba33cb6e27edd6cfb3b09e898e224 /var/lib/docker/overlay2/79b6840914c85c31cced13f9e94b8f249abb6161e8a6845b10819226f5861373/diff
sha256:a29d32b25d6f4e4c5f5206e7858a374b3cc72a535027998d06f1b0e36273e841 /var/lib/docker/overlay2/39b133f779adead4edade7fec9c27a17e90da27c390ab155a1d353c23051e212/diff
sha256:30bd8055d43515ac381d4434a3d4503a412110b568ec2871e7f1b3e8d93cb4ca /var/lib/docker/overlay2/2c29d8c2a34fe7284a1e62df9ac8720d0d10a65aef076f730b6f46f91e06d2e/diff
sha256:4aa1c3f2fcd8a2f9838ea11e0c0bd81ccbd2c1466909b8df899b3c3097648f /var/lib/docker/overlay2/1f2430f4ac80db60c888aaf4b7c6701c8cc4df76370dcfd1cce04c8552dacbe1/diff
sha256:561a3bb31e16942b56812a9e66eae5c44bba8a63f6109de80674bfa3ceb46 /var/lib/docker/overlay2/d325dad61d18820df4f3402661fe78b2d2e19087ee41473584737536be99763/diff
sha256:d8b40421b2d1f9e5a35073a6859cd34856d41284181dc04f3ed25100346cf17 /var/lib/docker/overlay2/73a22facea54f943ef00dd88aac1373d9bb86987649059037b1f5822c8c9b25/diff
sha256:de081cf3fa1ee0be66e193142128b2e5cb0611d773efc2baae6f08509e0b995 /var/lib/docker/overlay2/50e5dcdfe1e18776e9737a749849a2a2542419ab0009559a98f35be944155e69/diff
sha256:3b2072b1fb6b30c7294dd7b4515720d74350ca5717d32283a0a5ecc0680aa977 /var/lib/docker/overlay2/853490cefae601dd05762510244609297812d975fc9d6a3234dcace6cd93cb9/diff
root@ubuntu: /home/admin1/Downloads/container#
```

Рисунок 8.2 – Вывод сопоставленных слоёв

9. ПОВЕДЕНИЕ КОНТЕЙНЕРА В ОСОБЫХ СЛУЧАЯХ

9.1. Изменение настроек после запуска контейнера

Настройки подключения к БД, такие как пароль пользователя и метод аутентификации, указываются переменными окружения при первом запуске БД (см. подраздел 3.4). Они используются контейнером при инициализации БД, после чего эти настройки можно изменить двумя способами:

- Удалить контейнер и создать его заново с использованием новых переменных окружения. При этом необходимо использовать новый docker-volume для каталога данных, потому что при использовании старого docker-volume БД не будет проинициализирована заново;

- Вручную изменить настройки в запущенном контейнере, например:

- Для смены пароля пользователя БД выполнить команду:

```
docker exec -it <имя контейнера> psql -U <имя пользователя>  
> -c "ALTER USER <имя пользователя> PASSWORD '<новый пароль>';"
```

- Для смены метода аутентификации необходимо отредактировать файл pg_hba.conf. Если для контейнера был создан и примонтирован docker-volume для каталога данных, отредактировать файл можно таким образом:

1. На хостовом сервере выполнить команду:

```
docker inspect postgresql-volume-data
```

2. Открыть файл pg_hba.conf: он находится в каталоге, указанном в графе «Mountpoint».

3. Изменить настройки в файле pg_hba.conf.

4. Перезапустить контейнер:

```
docker restart <имя контейнера>
```

10. SSL АУТЕНТИФИКАЦИЯ В СУБД «JATOBА» КОНТЕЙНЕРЕ

SSL аутентификация в СУБД «Jatoba» контейнере описана в документе «Руководство по безопасности СУБД «Jatoba».

ПРИЛОЖЕНИЕ 1

Установка компонента Oracle_FDW

Чтобы установить компонент Oracle_FDW, необходимо:

1. Скачать внешний rpm-пакет Oracle Instance Client версии 11.1. Скачать его возможно по адресу: <https://www.oracle.com/database/technologies/instant-client/linux-x86-64-downloads.html>

2. Скопировать файл пакета в контейнер:

```
docker cp <oracle instance client>.rpm <имя контейнера>:/<oracle instance client>.rpm
```

3. Перейти в каталог с дистрибутивом astralinux17smolensk.

4. Скопировать установщик компонента на контейнер:

```
docker cp jatoba4-oracle-fdw_4.8.2-1124_amd64.deb <имя контейнера>:/jatoba4-oracle-fdw_4.8.2-1124_amd64file.deb
```

5. Зайти в bash оболочку контейнера:

```
docker exec -it <имя контейнера> /bin/bash
```

6. Прописать репозиторий:

```
echo "deb https://dl.astralinux.ru/astra/stable/1.7_x86-64/repository-main/ 1.7_x86-64 main contrib non-free" > /etc/apt/source.list
```

7. Обновить списки пакетов:

```
apt update
```

8. Установить дополнительные библиотеки:

```
apt install alien libaiol
```

9. Установить Oracle Instance Client:

```
alien -i <oracle instance client>.rpm
```

10. Выполнить конфигурирование установленной библиотеки:

```
echo /usr/lib/oracle/11.2/client64/lib > /etc/ld.so.conf.d/oracle-instantclient.conf  
ldconfig
```

11. Установить пакет Oracle_FDW:

```
dpkg -i jatoba4-oracle-fdw_4.8.2-1124_amd64file.deb
```

Далее необходимо настроить компонент в соответствии с инструкцией к компоненту.

ПРИЛОЖЕНИЕ 2

Установка компонента PLsPgSQL

Чтобы установить компонент PLsPgSQL, необходимо:

1. Скачать библиотеку lsb-cprosp.

lsb-cprosp – криптопровайдер «КриптоПро CSP» версии 5.0. Данный пакет распространяется в виде самостоятельного DEB/RPM дистрибутива. Для установки необходимо обратиться к документации на официальном сайте разработчика КриптоПро – для доступа требуется зарегистрированный пользователь по правилам сайта <https://cryptopro.ru/user>);

2. Создать каталог cryptopro в контейнере:

```
sudo docker exec -it <имя контейнера> mkdir cryptopro
```

3. Скопировать файлы библиотеки в каталог cryptopro в контейнере:

```
docker cp <имя каталога с файлами КриптоПро>/ . <имя контейнера>:/cryptopro
```

4. Дать скрипту install.sh право на исполнение:

```
sudo docker exec -it <имя контейнера> chmod +x cryptopro/install.sh
```

5. Установить библиотеку КриптоПро:

```
sudo docker exec -it <имя контейнера> cryptopro/install.sh
```

6. Перейти в каталог с дистрибутивом astralinux17smolensk.

7. Скопировать пакет gis-cryptoplatform17 на контейнер:

```
docker cp gis-cryptoplatform17_1.7.3-4_amd64.deb <имя контейнера>:/gis-cryptoplatform17_1.7.3-4_amd64.deb
```

8. Скопировать пакет PLsPgSQL на контейнер:

```
docker cp jatoba4-plspgsql_4.8.2-1124_amd64.deb <имя контейнера>:/jatoba4-plspgsql_4.8.2-1124_amd64.deb
```

9. Установить библиотеку gis-cryptoplatform17:

```
docker exec -it <имя контейнера> dpkg -i gis-cryptoplatform17_1.7.3-4_amd64.deb
```

10. Установить компонент PLsPgSQL:

```
docker exec -it <имя контейнера> dpkg -i jatoba4-plspgsql_4.8.2-1124_amd64.deb
```

Далее необходимо настроить компонент в соответствии с инструкцией к компоненту.

ПРИЛОЖЕНИЕ 3

Пример настройки резервного копирования

Настройка резервного копирования СУБД в контейнерном исполнении требует выполнения следующих шагов:

1) Предварительные действия:

Создать

BM1 - файловый сервер.

BM2 - сервер Jatoba.

2) 2 Подготовить BM1 с файловым сервером:

обновить ОС и установить nfs-сервер командой в терминале ОС:

```
apt-get update -y  
apt-get install nfs-kernel-server
```

3) Открыть файл настройки сервера:

```
nano /etc/exports
```

и добавить строку:

```
/path/to/share/dir <IP-адрес BM2>(rw,sync,no_subtree_check)
```

Например:

```
/home/user/share 192.168.223.130(rw,sync,no_subtree_check)
```

4) Создать каталог для файлового сервера:

```
mkdir /home/user/share
```

--здесь и далее будет использован каталог /home/user/share для примера из шага 3

5) Настроить права:

```
chmod 777 /home/user/share
```

6) Применить настройки и перезапустить nfs-сервер:

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

```
exportfs -a  
systemctl restart nfs-server
```

7) Подготовить VM2 с сервером Jatoba:

Обновить ОС и установить nfs-клиент и docker:

```
apt-get update -y  
apt-get install nfs-common docker.io
```

8) Создать и примонтировать сетевой каталог:

```
mkdir /mnt/<share каталог>  
mount -t nfs <IP-адрес VM1>:/path/to/share/dir /mnt/<share  
каталог>
```

9) Скопировать в домашний каталог на VM2 папку, содержащую архив с образом и скрипты для работы с ним.

10) Сценарий:

Выдать права на выполнение скриптов

```
chmod a+x *.sh
```

11) Проверить права:

```
ls -l
```

12) Выполнить скрипт:

```
sudo ./setup.sh
```

13) Проверить наличие образа:

```
sudo docker images
```

14) Отредактировать скрипт run.sh, добавить выделенную ниже строку:

```
docker run -it -d \  
-e PGDATA=/var/lib/jatoba/${PG_MAJOR}/data/pgdata \  
-v /mnt/nfs_share:/var/lib/jatoba/${PG_MAJOR}/backups \  

```

```
-v  
${POSTGRESQL_VOLUME_DATA}:/var/lib/jatoba/${PG_MAJOR}/data \  
-v  
${POSTGRESQL_VOLUME_EXTENSIONS}:/var/lib/jatoba/${PG_MAJOR} \  
--user root \  
--env-file .env \  
-p 54321:5432 --name ${CONTAINER_NAME} $_IMAGE_TAG
```

15) Запустить контейнер:

```
./run.sh
```

16) Проверить примонтированные в контейнер каталоги:

```
docker inspect jatobadb | grep -A 50 Mounts
```

17) Выполнить подключение к контейнеру:

```
docker exec -it jatobadb /bin/bash
```

18) Инициализировать каталог резервных копий (директория backups):

```
/usr/jatoba-*/bin/pg_probackup init -B  
/var/lib/jatoba/*/backups
```

19) Определить копируемый экземпляр БД (-B каталог_резервных_копий -D директория_данных):

```
/usr/jatoba-*/bin/pg_probackup add-instance -B  
/var/lib/jatoba/*/backups -D /var/lib/jatoba/*/data/pgdata --  
instance localdb
```

20) Указать параметры подключения:

```
/usr/jatoba-*/bin/pg_probackup set-config -B  
/var/lib/jatoba/*/backups --instance localdb -d postgres -h  
127.0.0.1 -U postgres --archive-timeout=1min
```

21) Выполнить команду создания резервной копии:

```
/usr/jatoba-*/bin/pg_probackup backup -B  
/var/lib/jatoba/*/backups --instance localdb -b FULL --stream
```

22) Убедиться, что полная резервная копия в режиме STREAM была создана:

```
/usr/jatoba-*/bin/pg_probackup show -B  
/var/lib/jatoba/*/backups --instance localdb
```

23) Убедиться, что каталог backups не пустой:

```
ls -la /var/lib/jatoba/6/backups/
```

24) Выйти из контейнера и проверить сетевую папку:

```
ls -la /mnt/nfs_share
```

25) Проверить сетевую папку на VM1:

```
ls -la /home/user/share
```

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

docker-volume – это механизм сохранения данных, сгенерированных и используемых контейнерами Docker, которые остаются после работы и удаления контейнеров, если были использованы в команде запуска контейнера через аргументы -v или --volume.

Хостовой сервер — сервер, на который устанавливается образ и запускается контейнер СУБД «Jatoba».

Docker - это платформа для автоматизации развертывания и управления приложениями в средах с различными операционными системами. Он позволяет упаковывать приложения со всем их зависимостями в легкие, повторно используемые модули-контейнеры. Docker ускоряет процесс разработки и доставки приложений, а также упрощает управление инфраструктурой и контейнеризацию приложений.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

ОС	–	Операционная система
БД	–	База данных
ПО	–	Программное обеспечение
СУБД	–	Система управления базами данных
ЭВМ	–	Электронно-вычислительная машина

[illegible]