

УТВЕРЖДЕН
643.72410666.00067-07 98 01-ЛУ

ЗАЩИЩЕННАЯ СИСТЕМА УПРАВЛЕНИЯ
БАЗАМИ ДАННЫХ «ЈАТОВА»

Руководство по настройке. Часть 28.
Поддержка мониторинга СУБД

643.72410666.00067-07 98 01-28

Листов 60

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

АННОТАЦИЯ

В документе приведены сведения, необходимые для установки и эксплуатации компонентов, предназначенных для мониторинга СУБД:

- Компонент «node_exporter». Версия компонента – 1.7.0;
- Компонент «postgres_exporter». Версия компонента – 0.15.0;
- Компонент «sql_exporter». Версия компонента – 0.13.0.
- Система «Prometheus». Версия системы – 0.52.0.
- Утилита «Alertmanager». Версия компонента – 0.27.0.

Настоящее руководство предназначено для администраторов СУБД.



Все примеры в данном документе приведены для СУБД «Jatoba» версии ядра 5.x, для других версий все шаги выполняются аналогично, разница состоит в именах директорий.

Например, СУБД «Jatoba» версии 6.x по умолчанию устанавливается в директорию:

- ОС Windows – «C:\Program Files\GIS\Jatoba\6\bin»;
- ОС Linux – «/usr/jatoba-6/bin».



Важная информация

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

Степени важности примечаний, применяемые в документе:



Важная информация – указания, требующие особого внимания



Дополнительная информация – указания, позволяющие упростить работу с изделием

СОДЕРЖАНИЕ

1. Назначение компонентов.....	5
1.1. Условия применения.....	5
1.2. Ограничения по эксплуатации.....	5
2. Архитектура системы мониторинга.....	7
3. Установка и настройка целевых СУБД.....	9
3.1. Установка СУБД.....	9
3.2. Настройка конфигурационных файлов.....	9
3.3. Установка расширения «pg_stat_statements».....	9
4. Установка экспортера «jatoba*_node_exporter».....	12
5. Установка экспортера «jatoba*_postgres_exporter».....	16
5.1. Установка утилиты и службы «jatoba*_postgres_exporter».....	16
5.2. Создание пользователя СУБД «postgres_exporter».....	17
5.3. Настройка переменных окружения.....	17
5.4. Запуск утилиты «postgres_exporter».....	19
6. Установка экспортера «jatoba*_sql_exporter».....	22
6.1. Установка утилиты и службы «sql_exporter».....	22
6.2. Настройка переменных окружения.....	23
6.3. Создание пользователя СУБД «sql_exporter».....	23
6.4. Настройка параметров экспортера и подключения к БД в файле «sql_exporter.yml».....	24
6.5. Запуск утилиты «jatoba*_sql_exporter».....	25
7. Система «Prometheus».....	28
7.1. Установка системы «Prometheus».....	28
7.2. Конфигурация системы «Prometheus».....	29
7.2.1. Примеры блока «postgres-exporter».....	30
7.2.2. Примеры блока «sql-exporter».....	31
7.2.3. Примеры блока «node-exporter».....	32
7.3. Запуск системы «Prometheus».....	34
8. Утилита «Alertmanager».....	38
8.1. Установка утилиты и службы «alertmanager».....	38
8.2. Настройка параметров утилиты файле «alertmanager.yml».....	39
8.3. Запуск утилиты «alertmanager».....	40
9. Подключение к JDS.....	42
9.1. Настройка SSH-соединения.....	43
9.2. Конфигурирование JDS.....	44
9.3. Настройка связки системы «Prometheus» и утилиты «Alertmanager».....	46
10. Система «Grafana».....	48
10.1. Установка системы «Grafana».....	48

10.2. Запуск системы «Grafana».....	48
10.3. Создание дашбордов	51
10.4. Импорт дашбордов	51
11. Дашборды.....	54
Приложение 1	56
Термины и определения	58
Перечень сокращений.....	59

1. НАЗНАЧЕНИЕ КОМПОНЕНТОВ

Компонент «node_exporter» – программный инструмент, предназначенный для мониторинга и сбора метрик с различных компонентов в Linux-подобных ОС.

Компонент собирает и экспортирует различные метрики, такие как загрузка процессора, использование памяти, статистика сети, системные вызовы и т.д. Собранные данные могут быть отправлены на серверы «Prometheus» или «Grafana» для визуализации и анализа.

Компонент «postgres_exporter» – инструмент для сбора и экспорта метрик PostgreSQL, таких как статистика по базе данных, нагрузка на сервер, количество запросов и т.д. Он разработан для работы с PostgreSQL и предоставляет данные в формате, удобном для системы «Prometheus». С помощью «postgres_exporter» можно отслеживать производительность PostgreSQL, выявлять проблемы и оптимизировать настройки базы данных.

Компонент «sql_exporter» – инструмент для экспорта данных из SQL-запросов в формат, удобный для анализа и визуализации. Он позволяет получать информацию о структуре таблиц, данных, индексах, статистике и других параметрах базы данных. Полезен для анализа производительности системы, выявления проблем и оптимизации запросов.

1.1. Условия применения

Компоненты могут использоваться с СУБД «Jatoba» версий 5.x и выше, под управлением операционных систем GNU/Linux.

1.2. Ограничения по эксплуатации

В комплект поставки не входит система «Grafana».

К системам «Prometheus» и «Grafana» поставляются конфигурационные файлы, обеспечивающие снятие метрик с СУБД «Jatoba».

При установке системы «Grafana» следует руководствоваться документацией с официального сайта разработчика систем.

Приведенные в документе действия по вышеуказанной системе носят рекомендательный характер.

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

Для подключения целевой СУБД к компоненту «Jatoba data safe» требуется указывать IP-адрес в строке подключения утилит к СУБД и не использовать параметр «localhost».

Символы «коммерческое эт» «@», «амперсанд» «&», «равно» «=», «вопросительный знак» «?» и «двоеточие» «:», не рекомендуется использовать в именах пользователей и в паролях, для исключения ошибки в строке подключения.

Эти символы используются для разделения параметров строки подключения.

Ограничений по совместимости с другими компонентами нет.

2. АРХИТЕКТУРА СИСТЕМЫ МОНИТОРИНГА

Архитектура системы мониторинга основана на том, что:

- на серверах целевых СУБД устанавливается экспортер «node_exporter» (см. р. 4);
- на целевых СУБД с предустановленным расширением «pg_stat_statements» устанавливаются утилиты сбора метрик, такие как:
 - экспортер «postgres_exporter» (см. р. 5);
 - экспортер «sql_exporter» (см. р. 6);
- система «Prometheus» собирает их в своём хранилище (см. р. 7);
- компонент «Jatoba data safe» использует данные хранилища «Prometheus» для отображения их в разделе «Мониторинг»;
- утилита «Alertmanager» обеспечивает контроль над пороговыми значениями и рассылку уведомлений (см. р. 8).

В зависимости от количества СУБД, подключенных к мониторингу и общей нагрузки, система «Prometheus» может быть установлена на отдельном сервере. В этом случае «JDS» будет получать данные по сети, что увеличит нагрузку на неё.

Целесообразнее компонент JDS и систему «Prometheus» устанавливать на одном сервере. Такая конфигурация сделает данный сервер полноценным сервером мониторинга и безопасности.

Для каждой наблюдаемой СУБД должны быть настроены все экспортёры.

В рассматриваемом примере на ОС Ubuntu 22.04 используются параметры сети и программного обеспечения, приведенные в таблице 2.1.

Таблица 2.1 – Конфигурация стенда

№	Имя сервера	IP-адрес	ПО	Port	Роль
1	u602doc-jds01	10.116.102.41/24			Сервер мониторинга
1.1			JDS		
1.2			Prometheus	9090	
			Alert manager	9093 22	
2	u602doc-pgp01	10.116.102.45/24			Целевая СУБД
2.1			pg_stat_statements		
2.2			node_exporter	9100	
2.3			postgres_exporter	9187	
2.4			sql_exporter	9399	

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

№	Имя сервера	IP-адрес	ПО	Port	Роль
3	u602doc-ldap01	10.116.102.47/24			Целевая СУБД
3.1			pg_stat_statements		
3.2			node_exporter	9100	
3.3			postgres_exporter	9187	
3.4			sql_exporter	9399	

Схема стенда представлена на рисунке 2.1.

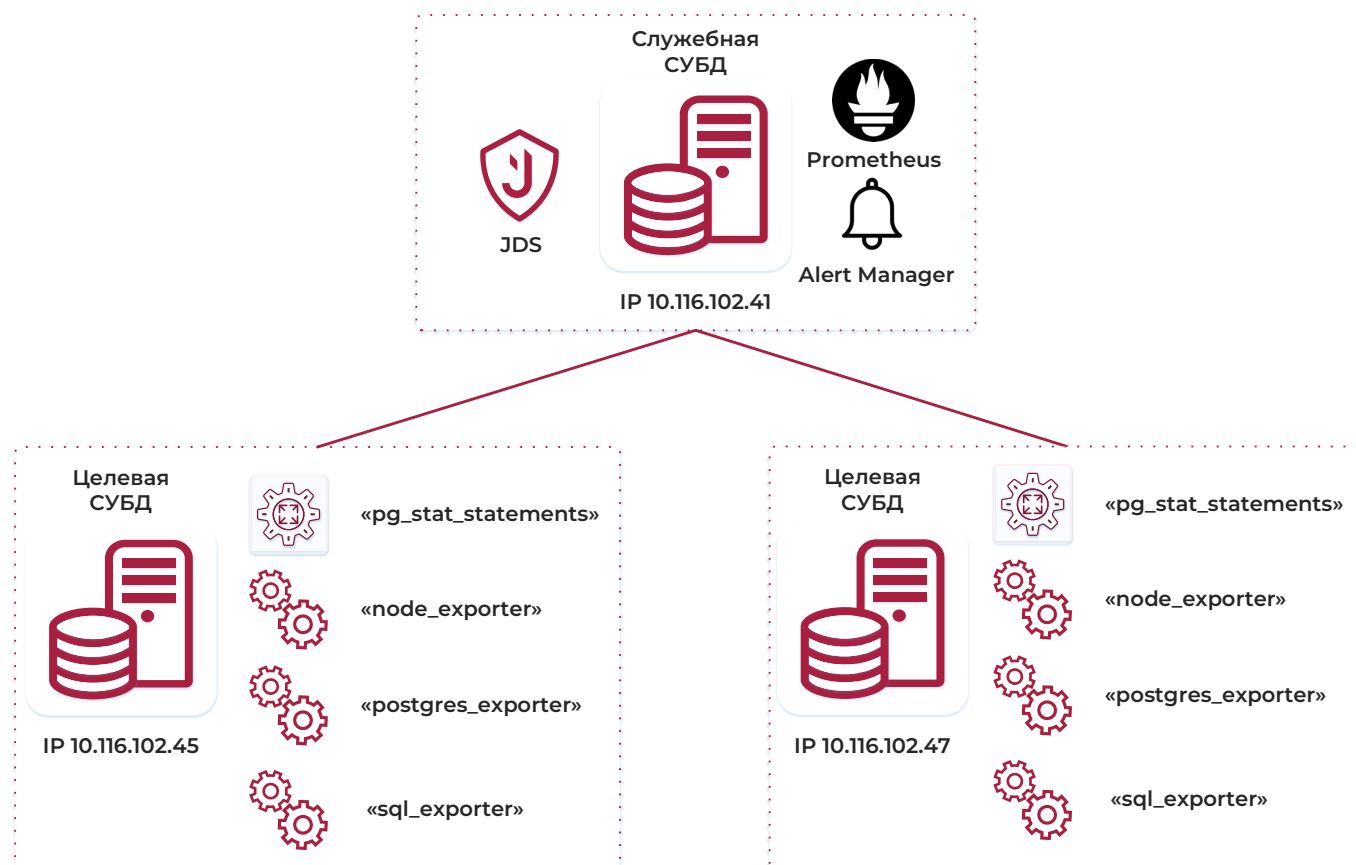


Рисунок 2.1 – Схема стенда

3. УСТАНОВКА И НАСТРОЙКА ЦЕЛЕВЫХ СУБД

3.1. Установка СУБД

Установка СУБД «Jatoba» выполняется от имени пользователя, обладающего административными привилегиями в системе, в соответствии с документом «Защищенная система управления базами данных «Jatoba». Руководство по установке».

3.2. Настройка конфигурационных файлов

Целевые СУБД должны быть настроены на приём подключений

В конфигурационном файле «postgresql.conf», в разделе «CONNECTIONS AND AUTHENTICATION» раскомментирован и установлен параметр:

```
listen_addresses = '*'
```

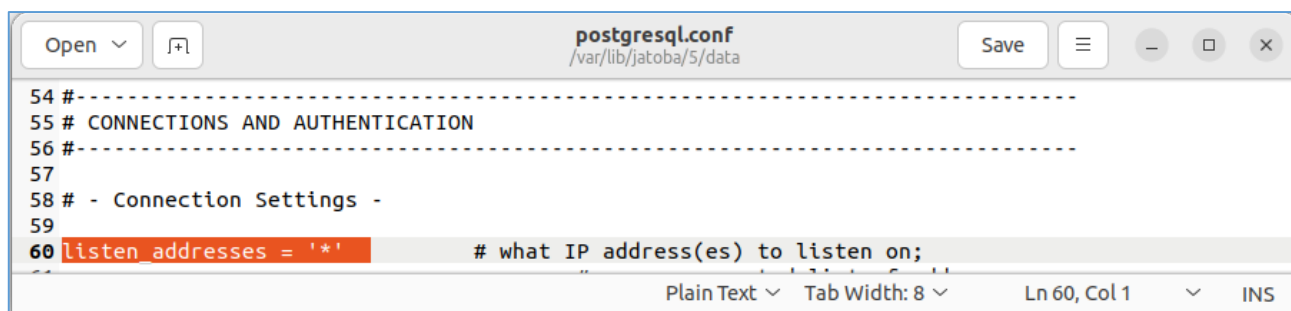


Рисунок 3.1 - Конфигурационный файл «postgresql.conf»

В конфигурационном файле «pg_hba.conf» разрешены подключения к СУБД в параметре:

host	all	all	all	md5
------	-----	-----	-----	-----

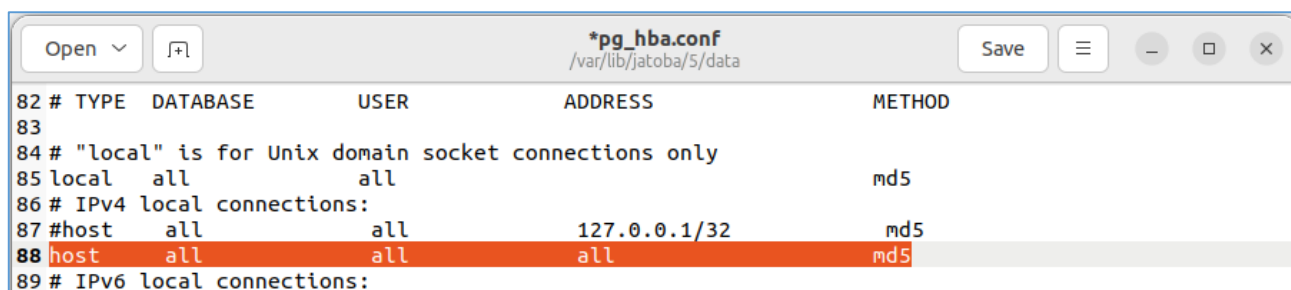


Рисунок 3.2 – Конфигурационный файл «pg_hba.conf»

3.3. Установка расширения «pg_stat_statements»

На каждой целевой СУБД должно быть установлено расширение «pg_stat_statements».

Для установки расширения «pg_stat_statements» потребуется:

- В конфигурационном файле «postgresql.conf», в разделе «Shared Library Preloading» для последующей загрузки расширения установить параметр:

```
shared_preload_libraries = 'pg_stat_statements'
```



Рисунок 3.3 – Строка загрузки расширения в конфигурационном файле СУБД «postgresql.conf»

- В разделе «STATISTICS» – «Monitoring» раскомментировать строку и добавить параметры:

```
compute_query_id = on
```

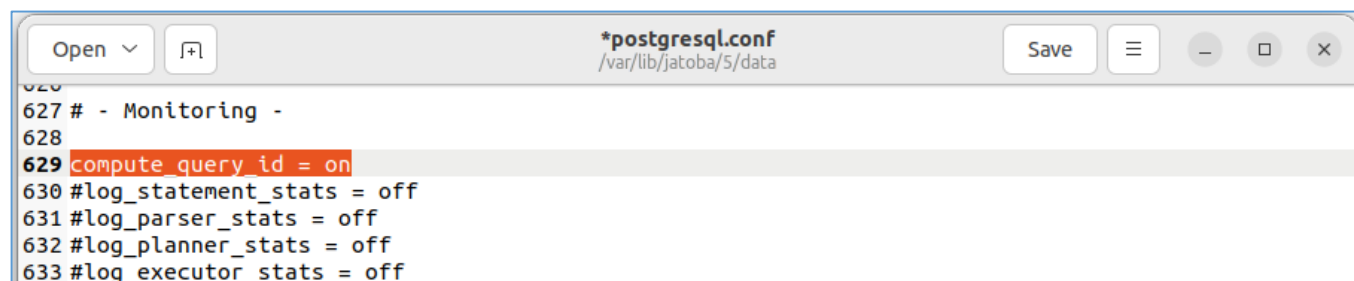


Рисунок 3.4 – Строка параметра «compute_query_id» в конфигурационном файле СУБД «postgresql.conf»

- В разделе «CUSTOMIZED OPTIONS» добавить параметры:

```
pg_stat_statements.max = 10000  
pg_stat_statements.track = all
```



Рисунок 3.5 – Параметры статистики в конфигурационном файле СУБД «postgresql.conf»

Сохранить конфигурационный файл «postgresql.conf» и перезагрузить СУБД.

Расширение «pg_stat_statements» устанавливается при помощи SQL-команды:

```
CREATE EXTENSION pg_stat_statements;
```

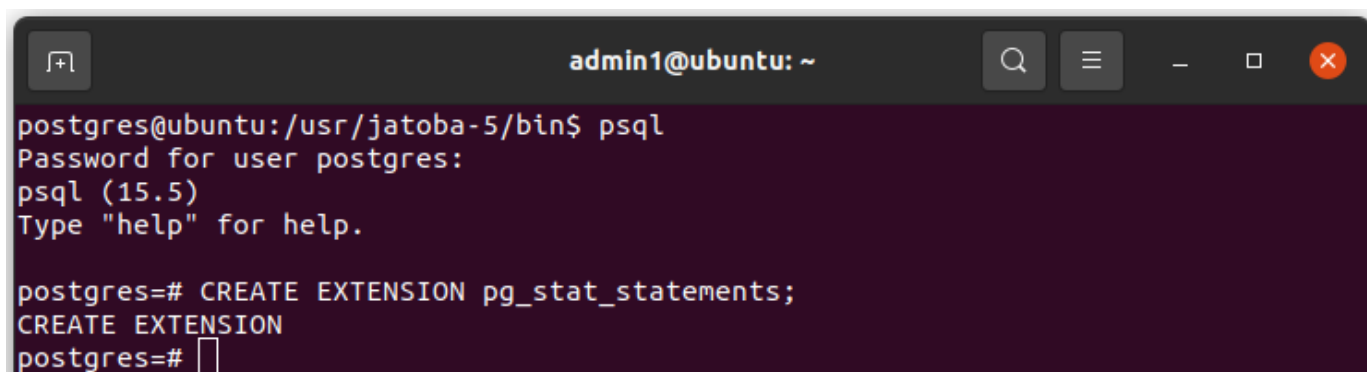


Рисунок 3.6 – Создание расширения

4. УСТАНОВКА ЭКСПОРТЕРА «JATOBА*_NODE_EXPORTER»

Экспортер «jatoba*_node_exporter» должен быть установлен на всех целевых СУБД.

Экспортер позволяет снимать различные метрики с Linux-подобных операционных систем. Это агент, который передает серверу «Prometheus» аппаратные и программные показатели работы GNU/Linux.

Установка пакета выполняется в соответствии с Руководством по установке, из локального репозитория командой:

```
apt-get install jatoba<ver>-node-exporter
```

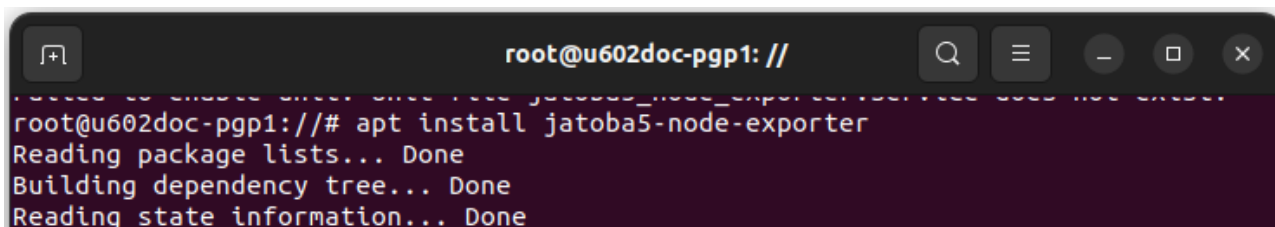


Рисунок 4.1 – Установка пакета «jatoba*_node_exporter»

В результате установки пакета будет создан пользователь ОС «node_exporter_usr», от которого будет производиться запуск утилиты.

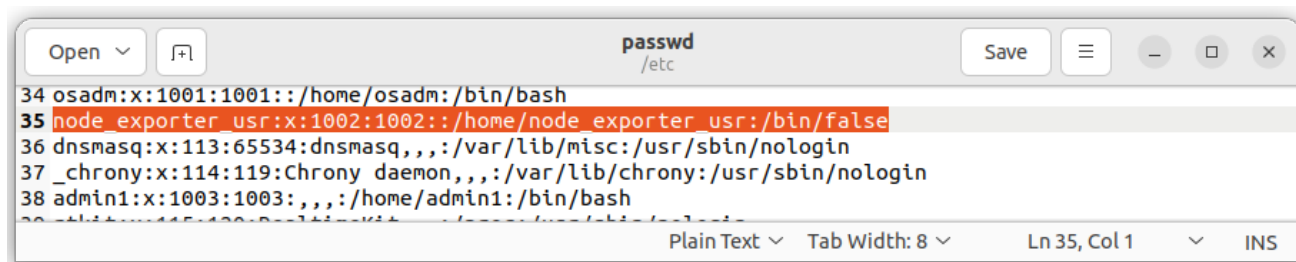


Рисунок 4.2 -пользователя «node_exporter_usr»

У данного пользователя нет интерактивной оболочки для входа.

Автоматически будет создан файл конфигурации сервиса по адресу:

```
/usr/lib/systemd/system/jatoba5_node_exporter.service
```



Рисунок 4.3 – Содержание конфигурационного файла

Далее требуется запустить службу экспортера, включить ее в автозапуск и проверить статус работы:

```
# systemctl enable jatoba<ver>_node_exporter
# systemctl start jatoba<ver>_node_exporter
# systemctl status jatoba<ver>_node_exporter
```

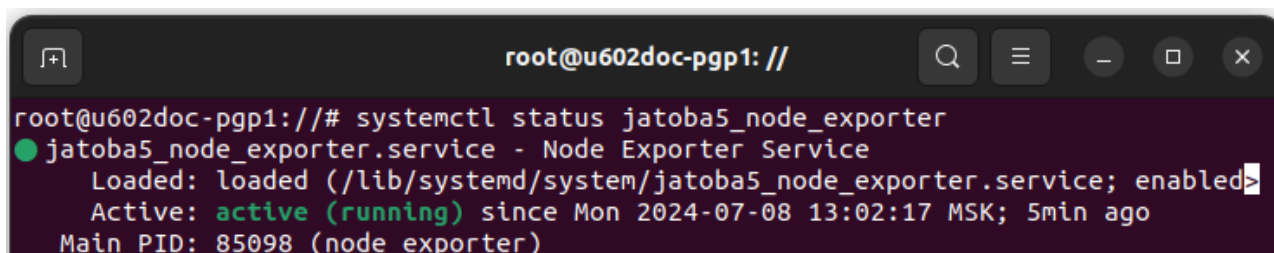


Рисунок 4.4 - Проверка сервиса «jatoba*_node_exporter»

Чтобы проверить статус работы экспортера нужно в браузере открыть веб-интерфейс экспортера:

```
# localhost:9100
# http://0.0.0.0:9100
```

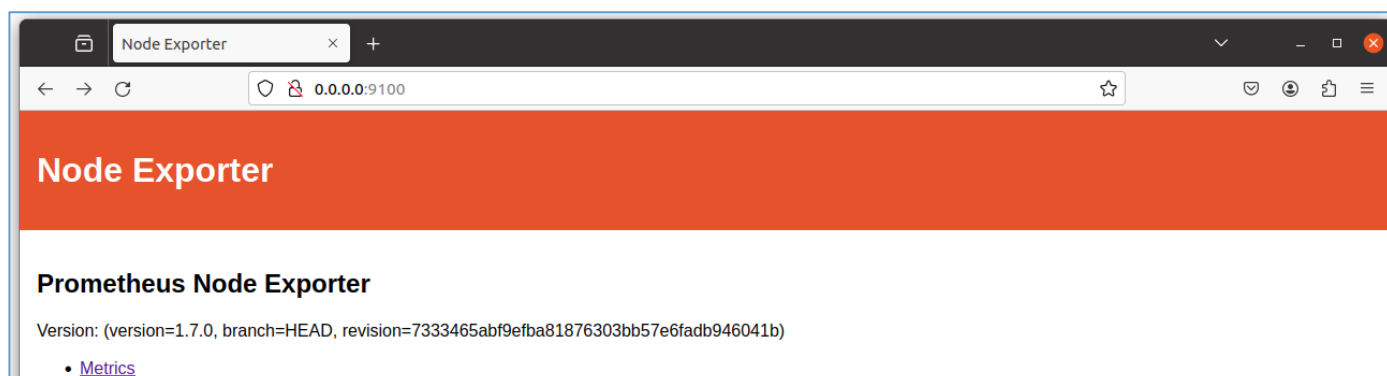


Рисунок 4.5 – Веб-интерфейс утилиты «node_exporter»

В рассматриваемом примере на целевой СУБД:

– u602doc-pgrp01 IP - 10.116.102.45 веб-интерфейс утилиты «node_exporter» проверяется по URL:

```
# http://10.116.102.45:9100
```

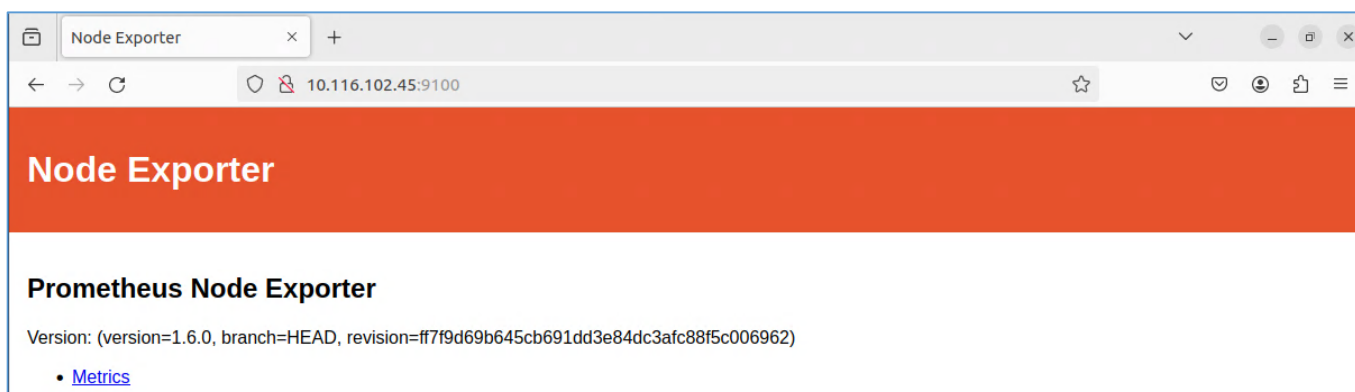


Рисунок 4.6 – Веб-интерфейс утилиты «node_exporter» на целевой СУБД u602doc-pgrp01 IP - 10.116.102.45

– u602doc-ldap01 IP-10.116.102.47 веб-интерфейс утилиты «node_exporter» проверяется по URL:

```
# http://10.116.102.47:9100
```



Рисунок 4.7 – Веб-интерфейс утилиты «node_exporter» на целевой СУБД u602doc-ldap01 IP-10.116.102.47

По умолчанию экспортер использует все доступные коллекторы метрик.

Состав снимаемых метрик отображается на странице:

```
localhost:9100/metrics
```

При необходимости может быть изменен состав используемых коллекторов с помощью опций командной строки:

```
./jatoba<ver>_node_exporter --[no-]collector.netdev --[no-]  
collector.netstat
```

Если необходимо изменить значения адреса веб-интерфейса (:9100), node_exporter запускается с опцией --web.listen-address:

```
./jatoba<ver>_node_exporter --web.listen-address=:9101
```



Изменение состава метрик либо адреса веб-интерфейса целесообразнее сохранить в файле сервиса «node_exporter.service». Иначе при перезагрузке ОС настройки компонента вернутся к изначальным, хранящимся в файле сервиса

Ручной запуск утилиты производится командой:

```
./jatoba<ver>_node_exporter
```

Никакой конфигурации экспортера не требуется.

5. УСТАНОВКА ЭКСПОРТЕРА «JATOBA*_POSTGRES_EXPORTER»

Экспортер «jatoba*postgres_exporter» должен быть установлен на всех целевых СУБД, и в той же БД в которой установлено расширение pg_stat_statements.

С помощью данного экспортера снимаются метрики с сервера PostgreSQL (Jatoba). Это агент, написанный на языке Golang, подключающийся к заданному источнику данных (БД) и по запросу сервера «Prometheus» возвращающий ему значения метрик. Состав метрик заранее предопределен и их значения вычисляются с помощью фиксированных SQL-запросов.

5.1. Установка утилиты и службы «jatoba*_postgres_exporter»

Установка пакета выполняется в соответствии с Руководством по установке, из локального репозитория командой:

```
apt-get install jatoba<ver>-postgres-exporter
```

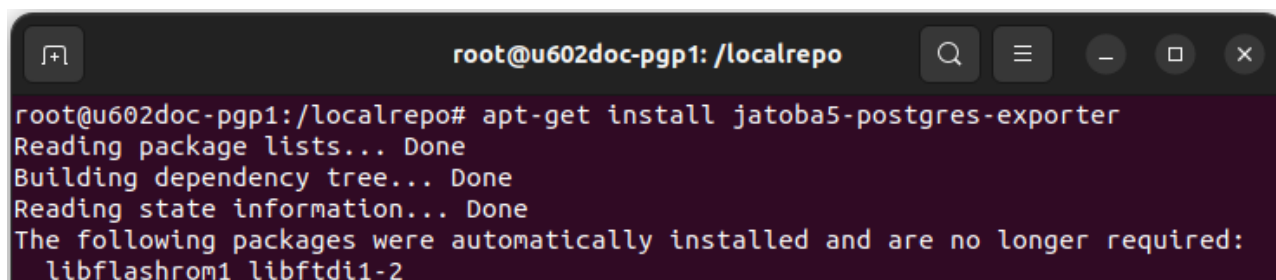


Рисунок 5.1 – Установка пакета jatoba*-postgres-exporter

В результате установки пакета будет создан:

- файл запуска по адресу:

```
/usr/jatoba-<ver>/bin/postgres_exporter
```

- конфигурационный файл по адресу:

```
/usr/jatoba-<ver>/monitoring/default/postgres_exporter
```

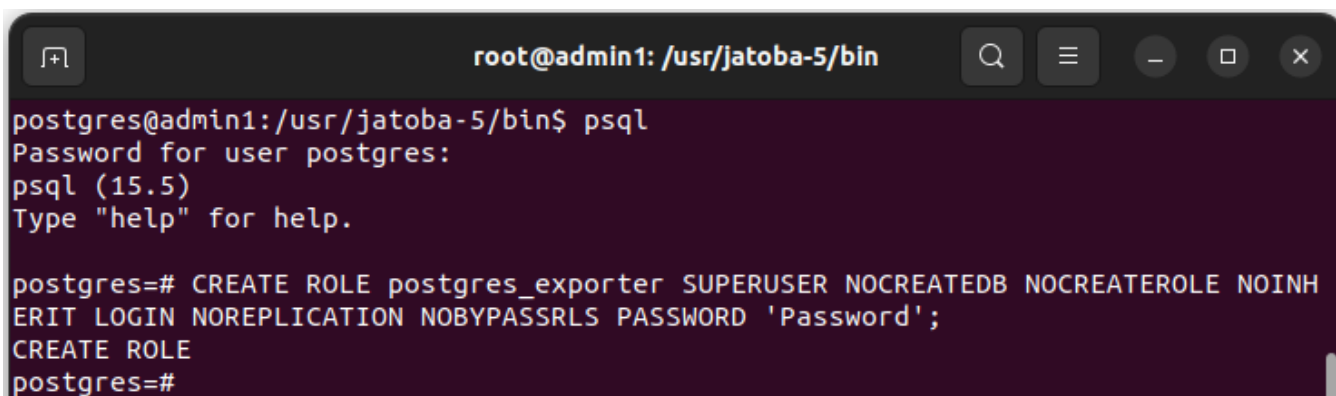
- пользователь ОС «postgres_exporter_usr», от которого будет производиться запуск сервиса.

У данного пользователя нет интерактивной оболочки для входа.

5.2. Создание пользователя СУБД «postgres_exporter»

Для соединения утилиты с СУБД создать пользователя СУБД «postgres_exporter» SQL-командой:

```
CREATE ROLE postgres_exporter SUPERUSER NOCREATEDB NOCREATEROLE  
NOINHERIT LOGIN NOREPLICATION NOBYPASSRLS PASSWORD 'Password';
```



The screenshot shows a terminal window with the title bar 'root@admin1: /usr/jatoba-5/bin'. The user 'postgres' is at the prompt 'postgres@admin1: /usr/jatoba-5/bin\$'. They enter 'psql', which prompts for a password. After entering the password, the 'psql' prompt appears. The user then enters the SQL command to create the role. The command is echoed back, and the prompt returns to 'postgres=#'.

```
postgres@admin1: /usr/jatoba-5/bin$ psql  
Password for user postgres:  
psql (15.5)  
Type "help" for help.  
  
postgres=# CREATE ROLE postgres_exporter SUPERUSER NOCREATEDB NOCREATEROLE NOINH  
ERIT LOGIN NOREPLICATION NOBYPASSRLS PASSWORD 'Password';  
CREATE ROLE  
postgres=#
```

Рисунок 5.2 – Создание роли «postgres_exporter»



В рассматриваемом примере пользователь СУБД «postgres_exporter» является привилегированным пользователем

5.3. Настройка переменных окружения

Дальнейшая настройка утилиты требует внесения параметров подключения в файле переменных окружения «postgres_exporter», командой:

```
gedit /usr/jatoba-5/monitoring/default/postgres_exporter
```

Необходимо настроить имя пользователя, пароль и параметры SSL-подключения в файле переменных окружения «postgres_exporter».

Строка подключения выполнена в формате схемы URL. Основная форма URI подключения имеет синтаксис:

```
postgresql://[пользователь@][сервер] [/база_данных] [?:указание_па  
раметра]  
где пользователь:  
имя_пользователя[:пароль]  
и сервер:  
[узел] [:порт] [, ...]
```

и указание параметра:
имя=значение [&...]

В качестве обозначения схемы URI может использоваться postgresql:// или postgres://. Остальные части URI являются необязательными. В следующих примерах показан допустимый синтаксис URI:

```
postgresql://  
postgresql://localhost  
postgresql://localhost:5433  
postgresql://localhost/mydb  
postgresql://user@localhost  
postgresql://user:secret@localhost  
postgresql://other@localhost/otherdb?connect_timeout=10&application_name=myapp  
postgresql://host1:123,host2:456/somedb?target_session_attrs=any&application_name=myapp
```

В рассматриваемом примере на целевой СУБД:

– u602doc-pgr01 IP - 10.116.102.45 строка подключения утилиты к СУБД имеет следующий вид:

```
DATA_SOURCE_NAME="postgresql://postgres_exporter:Password@10.116.102.45:5432/postgres?sslmode=disable"
```



Обратите внимание, что необходимо прописывать общий, а не локальный адрес сетевого интерфейса

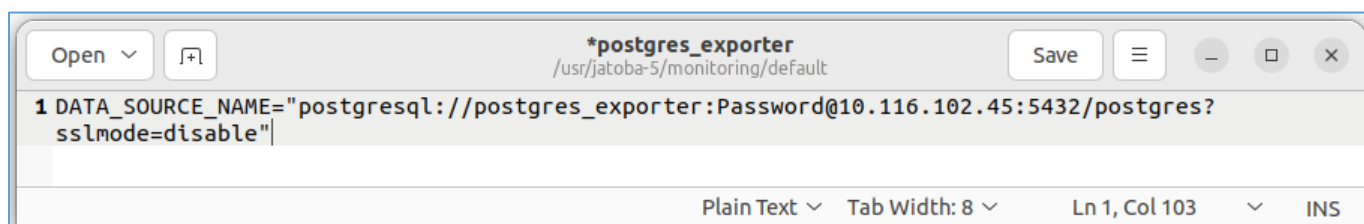


Рисунок 5.3 – Содержание файла «postgres_exporter.default» на целевой СУБД u602doc-pgr01 IP - 10.116.102.45

– u602doc-ldap01 IP-10.116.102.47 строка подключения утилиты к СУБД имеет следующий вид:

```
DATA_SOURCE_NAME="postgresql://postgres_exporter:Password@10.116.102.47:5432/postgres?sslmode=disable"
```

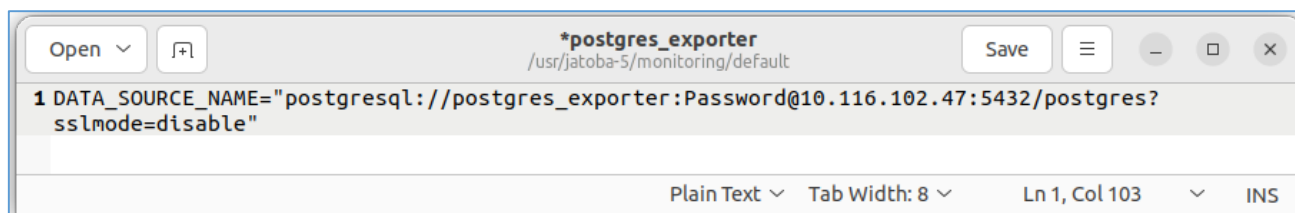


Рисунок 5.4— Содержание файла «postgres_exporter.default» на целевой СУБД u602doc-ldap01 IP-10.116.102.47

5.4. Запуск утилиты «postgres_exporter»

Обновить конфигурацию system командой:

```
# sudo systemctl daemon-reload
```

Запустить службу экспортера, включить ее автозапуск и проверить статус работы:

```
# systemctl start jatoba<ver>_postgres_exporter
# systemctl enable jatoba<ver>_postgres_exporter
# systemctl status jatoba<ver>_postgres_exporter
```

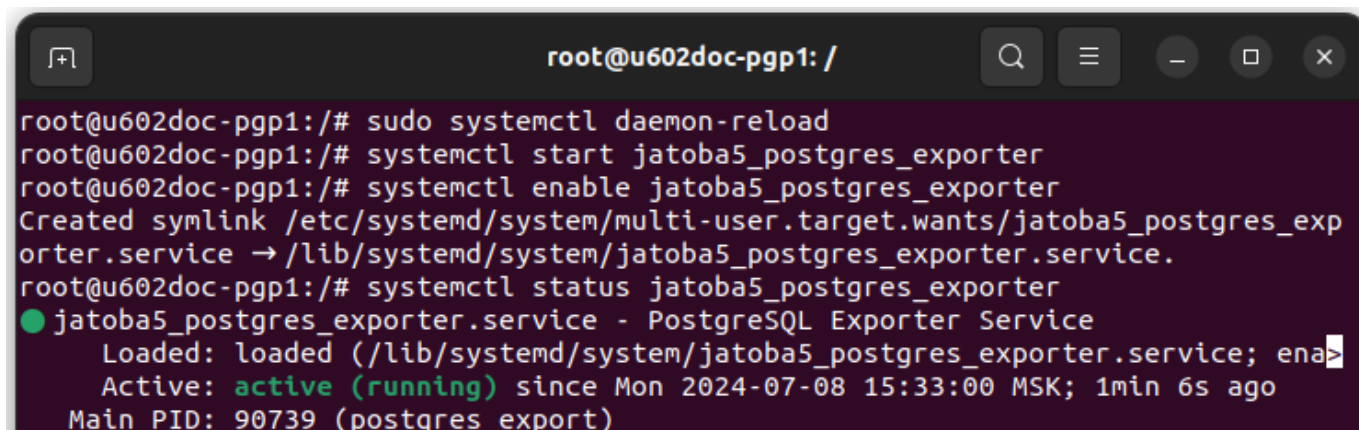


Рисунок 5.5 – Запуск и вывод статуса службы «postgres_exporter»

Чтобы проверить статус работы экспортера нужно в браузере открыть веб-интерфейс экспортера:

```
# localhost:9187
# http://0.0.0.0:9187
```

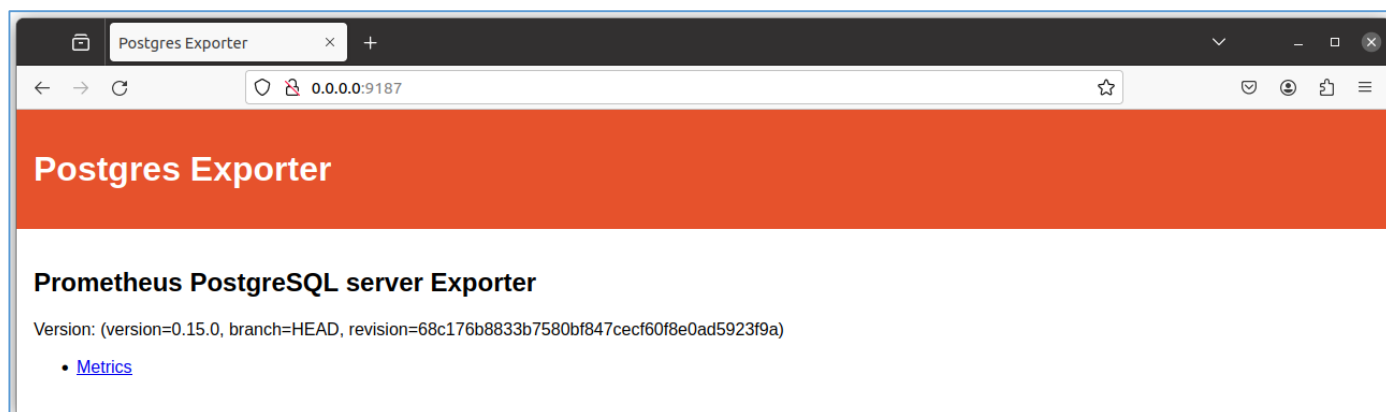


Рисунок 5.6 – Веб-интерфейс «postgres_exporter»

В рассматриваемом примере на целевой СУБД:

– u602doc-pgp01 IP - 10.116.102.45 веб-интерфейс утилиты «node_exporter» проверяется по URL:

```
# http://10.116.102.45:9187
```

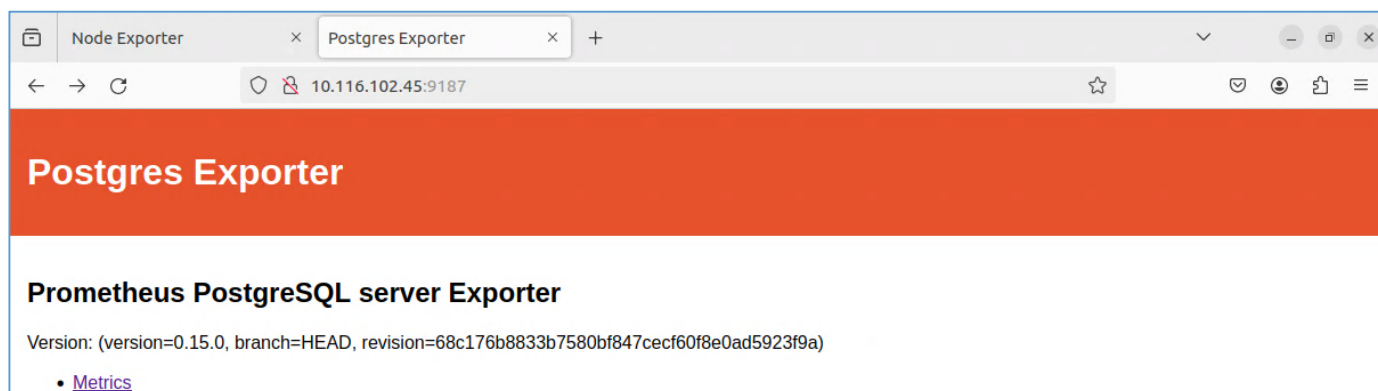


Рисунок 5.7 – Веб-интерфейс «postgres_exporter» на целевой СУБД u602doc-pgp01 IP - 10.116.102.45

– u602doc-ldap01 IP-10.116.102.47 веб-интерфейс утилиты «node_exporter» проверяется по URL:

```
# http://10.116.102.47:9187
```

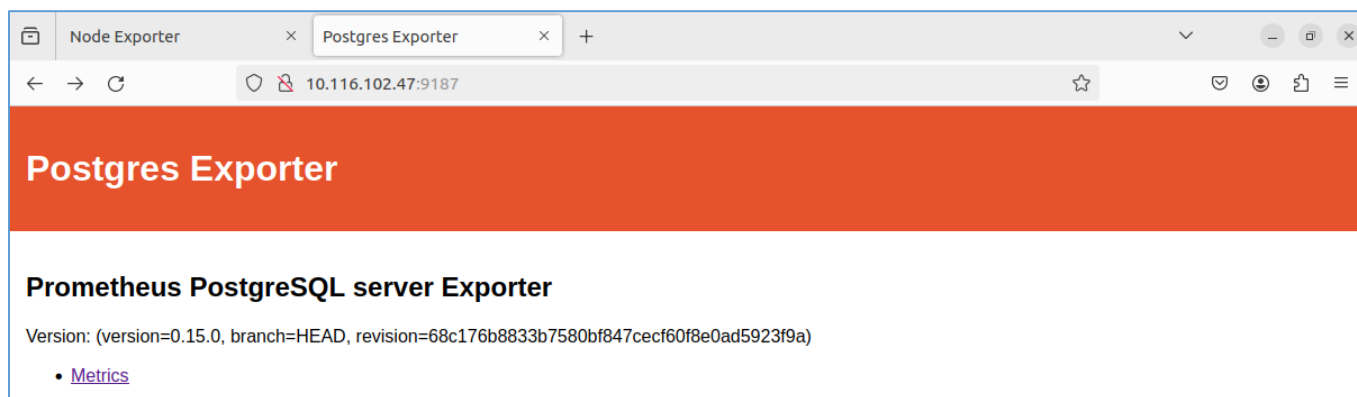


Рисунок 5.8 – Веб-интерфейс «postgres_exporter» на целевой СУБД u602doc-ldap01 IP-10.116.102.47

При успешном подключении к БД на странице localhost:9187/metrics будет показан список значений метрик с префиксом «pg_» в имени.

Если необходимо изменить значение адреса веб-интерфейса (по умолчанию :9187), postgres_exporter запускается с опцией --web.listen-address, например:

```
export DATA_SOURCE_NAME=postgresql://postgres:secret@127.0.0.1
./jatoba*_postgres_exporter --web.listen-address=:9188
```



Изменение адреса веб-интерфейса целесообразнее сохранить в файле сервиса «postgres_exporter». Иначе при перезагрузке ОС настройки компонента вернуться к изначальным, хранящимся в файле сервиса.

Полный список опций командной строки postgres_exporter можно вывести, если запустить его с опцией --help.

Компонент «postgres_exporter» можно запускать из Docker, если использовать готовые образы из репозитория <https://quay.io/repository/prometheuscommunity/postgres-exporter>.

Например:

```
docker pull quay.io/prometheuscommunity/postgres-
exporter:latest
docker run -p 9187:9187 -e
DATA_SOURCE_NAME="postgresql://postgres:sql@IP/" --name
postgres-exporter quay.io/prometheuscommunity/postgres-exporter
```

6. УСТАНОВКА ЭКСПОРТЕРА «JATOBA*_SQL_EXPORTER»

Экспортер «jatoba*_SQL_exporter» должен быть установлен на всех целевых СУБД и в той же БД в которой установлено расширение pg_stat_statements.

Данный экспортер можно использовать для расширения состава метрик, снимаемых с сервера PostgreSQL стандартным экспортером «jatoba*_postgres_exporter» (см. п. 5).

Это агент, также написанный на языке Golang, который подключается к заданному источнику данных (БД) и забирает с него метрики по pull-запросу сервера «Prometheus».

Состав собираемых метрик и SQL-запросов, которые их возвращают, полностью конфигурируемы пользователем. Используемые SQL-запросы группируются в так называемые коллекторы, состав которых легко может быть расширен. Также в коллекторе для каждого возвращаемого запросом поля задается мэппинг на соответствующую метрику.

6.1. Установка утилиты и службы «sql_exporter»

Установка пакета выполняется в соответствии с Руководством по установке, из локального репозитория командой:

```
apt-get install jatoba<ver>-sql-exporter
```

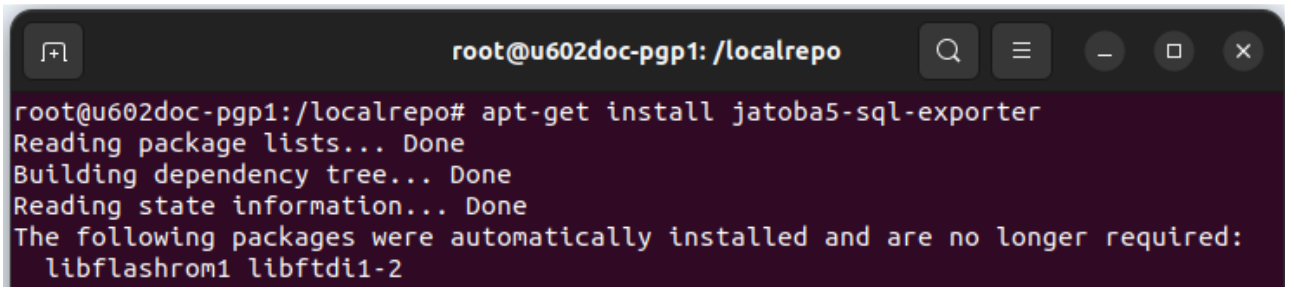


Рисунок 6.1 – Установка пакета «jatoba*-sql-exporter»

В результате установки пакета будет создан:

- файл запуска по адресу:

```
/usr/jatoba-<ver>/bin/sql_exporter
```

- конфигурационный файл по адресу:

```
/usr/jatoba-<ver>/monitoring/default/sql_exporter.yml
```

- конфигурационный файл переменных окружения по адресу

```
/usr/jatoba-5/monitoring/default/sql_exporter
```

– пользователь ОС «sql_exporter_usr», от которого будет производиться запуск сервиса.

У данного пользователя нет интерактивной оболочки для входа.

6.2. Настройка переменных окружения

Проверить параметры экспортера в файле переменных окружения «sql_exporter», выполнив команду редактирования:

```
gedit /usr/jatoba-5/monitoring/default/sql_exporter
```

Основным из параметров является путь к конфигурационному файлу «sql_exporter.yml» в строке параметра CONF_FILE.

```
CONF_FILE=/etc/sql_exporter/sql_exporter.yml
```

Настройка и расположение файла «sql_exporter.yml» приведены в п. 6.4 настоящего документа.

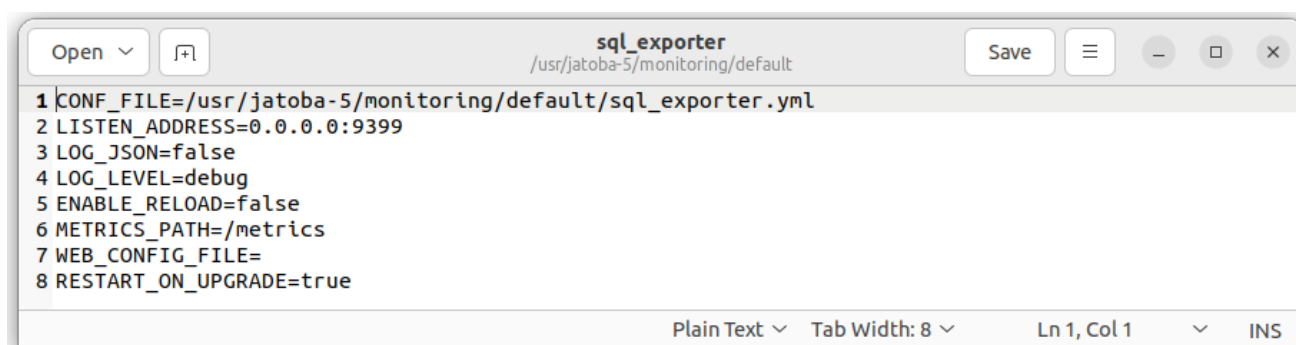
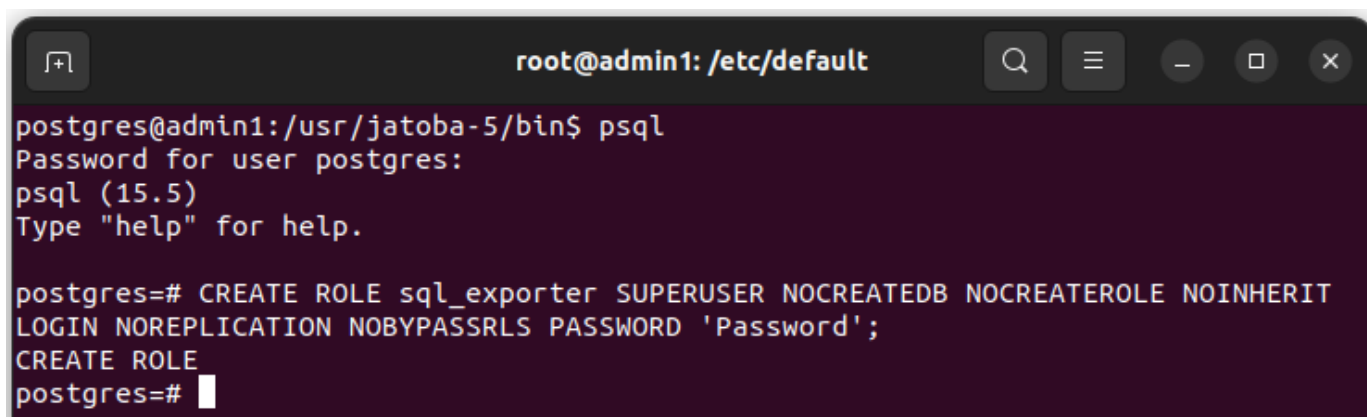


Рисунок 6.2 – Содержание файла переменных окружения «sql_exporter»

6.3. Создание пользователя СУБД «sql_exporter»

Для соединения утилиты с СУБД необходимо создать пользователя СУБД «sql_exporter» SQL-командой:

```
CREATE ROLE sql_exporter SUPERUSER NOCREATEDB NOCREATEROLE
NOINHERIT LOGIN NOREPLICATION NOBYPASSRLS PASSWORD 'Password';
```



```
root@admin1: /etc/default
postgres@admin1:/usr/jatoba-5/bin$ psql
Password for user postgres:
psql (15.5)
Type "help" for help.

postgres=# CREATE ROLE sql_exporter SUPERUSER NOCREATEDB NOCREATEROLE NOINHERIT
LOGIN NOREPLICATION NOBYPASSRLS PASSWORD 'Password';
CREATE ROLE
postgres=#
```

Рисунок 6.3 – Создание роли «sql_exporter»



В рассматриваемом примере пользователь СУБД «sql_exporter» является привилегированным пользователем

6.4. Настройка параметров экспортера и подключения к БД в файле «sql_exporter.yml»

Основным параметром для настройки параметров экспортера и подключения к БД в файле «sql_exporter.yml» является параметр «data_source_name».

Требуется открыть файл для редактирования командами:

```
gedit /usr/jatoba-5/monitoring/default/sql_exporter.yml
```

Строка подключения выполнена в формате схемы URL. Синтаксис строки описан в п. 5.3 настоящего документа.

В рассматриваемом примере на целевой СУБД:

– u602doc-pgr01 IP - 10.116.102.45 строка подключения утилиты к СУБД имеет следующий вид:

```
data_source_name:
'postgresql://sql_exporter:Password@10.116.102.45:5432/postgres
?sslmode=disable'
```



Обратите внимание, что необходимо прописывать общий, а не локальный адрес сетевого интерфейса

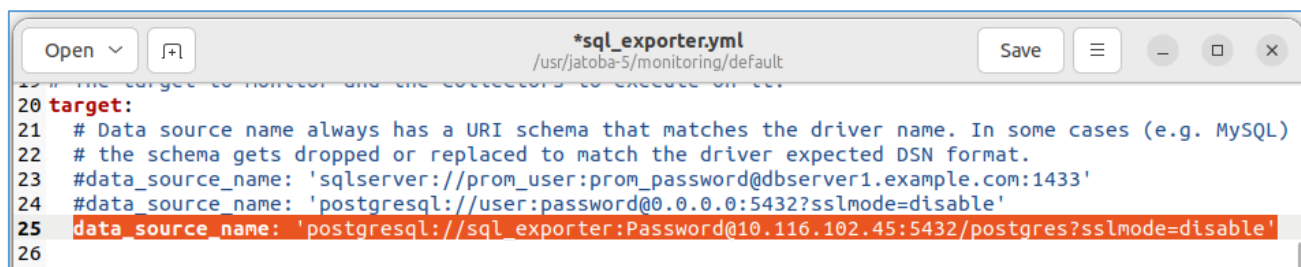


Рисунок 6.4 – Содержание файла «sql_exporter.yml», строка «data_source_name» на целевой СУБД u602doc-pgp01 IP - 10.116.102.45

– u602doc-ldap01 IP-10.116.102.47 строка подключения утилиты к СУБД имеет следующий вид:

```
data_source_name:
'postgresql://sql_exporter:Password@10.116.102.47:5432/postgres
?sslmode=disable'
```

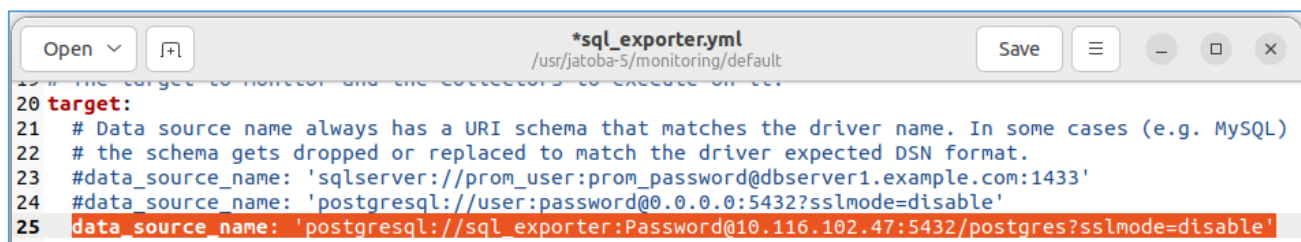


Рисунок 6.5 – Содержание файла «sql_exporter.yml», строка «data_source_name» на целевой СУБД u602doc-ldap01 IP-10.116.102.47

Сохранить внесенные изменения.

В дистрибутиве содержится файл с подготовленными метриками для мониторинга СУБД «Jatoba» «postgres.collector.yml», который по умолчанию использует «jatoba*_SQL_exporter».

6.5. Запуск утилиты «jatoba*_sql_exporter»

Обновить конфигурацию systemd:

```
# systemctl daemon-reload
```

Запустить службу экспортера, включить ее в автозапуск и проверить статус работы:

```
# systemctl start jatoba5_sql_exporter
# systemctl enable jatoba5_sql_exporter
# systemctl status jatoba5_sql_exporter
```

```
root@u602doc-pgp1: /
root@u602doc-pgp1:/# systemctl daemon-reload
root@u602doc-pgp1:/# systemctl start jatoba5_sql_exporter
root@u602doc-pgp1:/# systemctl enable jatoba5_sql_exporter
Created symlink /etc/systemd/system/multi-user.target.wants/jatoba5_sql_exporter.service → /lib/systemd/system/jatoba5_sql_exporter.service.
root@u602doc-pgp1:/# systemctl status jatoba5_sql_exporter
● jatoba5_sql_exporter.service - SQL Exporter for Prometheus
   Loaded: loaded (/lib/systemd/system/jatoba5_sql_exporter.service; enabled;
   Active: active (running) since Tue 2024-07-09 08:38:59 MSK; 17s ago
```

Рисунок 6.6 – Установка и запуск службы «sql_exporter»

Чтобы проверить статус работы экспортера, нужно в браузере открыть веб-интерфейс экспортера:

```
# localhost:9399
# http://0.0.0.0:9399
```

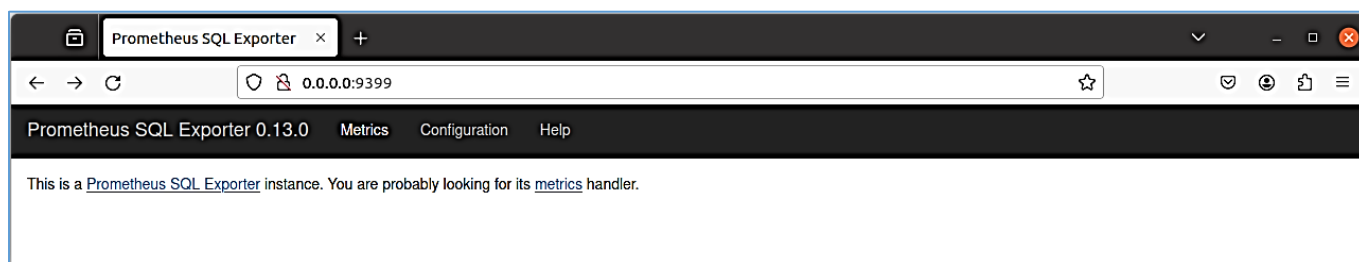


Рисунок 6.7 – Веб-интерфейс «sql_exporter»

В рассматриваемом примере на целевой СУБД:

– u602doc-pgp01 IP - 10.116.102.45 веб-интерфейс утилиты «sql_exporter» проверяется по URL:

```
# http://10.116.102.45:9399
```

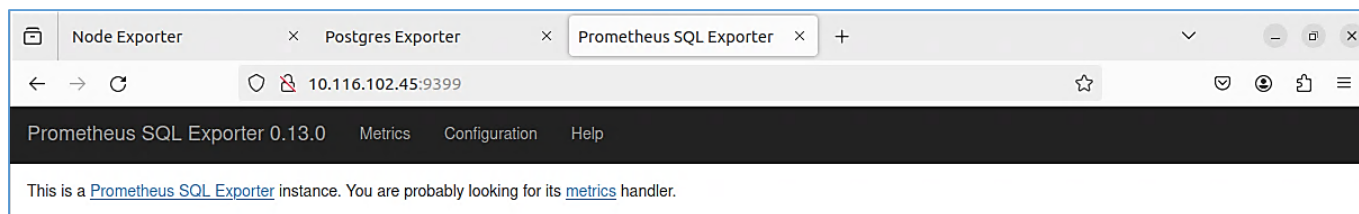


Рисунок 6.8 – Веб-интерфейс «sql_exporter» на целевой СУБД u602doc-pgp01 IP - 10.116.102.45

– u602doc-ldap01 IP-10.116.102.47 веб-интерфейс утилиты «node_exporter» проверяется по URL:

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

```
# http://10.116.102.47:9399
```

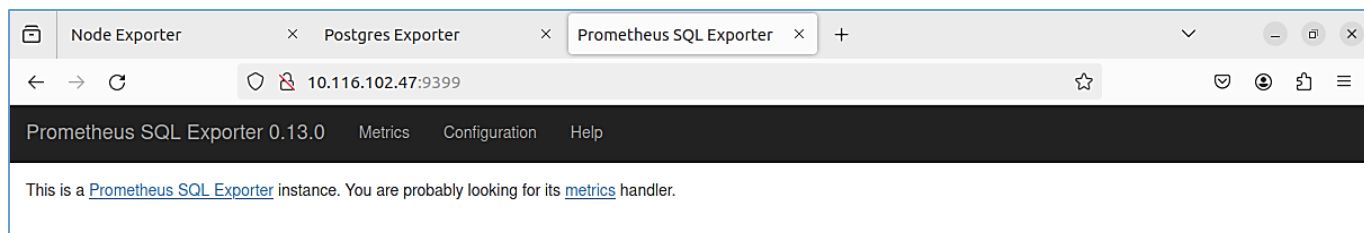


Рисунок 6.9 – Веб-интерфейс «sql_exporter» на целевой СУБД u602doc-ldap01 IP-10.116.102.47

При успешном подключении к БД и отсутствии ошибок в конфигурации на странице localhost:9399/metrics будет показан список значений снятых метрик.

Если необходимо изменить значения адреса веб-интерфейса (:9399), jatoba*_sql_exporter запускается с опцией -web.listen-address, например:

```
./jatoba5_sql_exporter -web.listen-address :9398
```



Изменение адреса веб-интерфейса целесообразнее сохранить в файле сервиса «sql_exporter». Иначе при перезагрузке ОС настройки компонента вернуться к изначальным, хранящимся в файле сервиса.

Полный список опций командной строки sql_exporter можно вывести, если запустить его с опцией -help.

sql_exporter можно запускать из Docker, если использовать готовый образ из репозитория https://hub.docker.com/r/burningalchemist/sql_exporter. Например

```
docker pull burningalchemist/jatoba5_sql_exporter
docker run -p 9399:9399 --name
jatoba5_sql_exporter burningalchemist/jatoba5_sql_exporter
```

7. СИСТЕМА «PROMETHEUS»

«Prometheus» – система мониторинга различных программных систем и сервисов. «Prometheus» собирает и сохраняет метрики в виде временных рядов данных. Информация о каждой метрике хранится вместе с отметкой времени, когда она была записана, и опциональным набором меток (labels), представляющих пары «ключ: значение». Сами метрики являются числовыми измерениями, которые по типу могут быть монотонно возрастающими значениями счетчиков (counter) или произвольно изменяющимися значениями датчиков (gauge).

Основными компонентами системы «Prometheus» являются:

- Сервер «Prometheus», который собирает и сохраняет метрики в своей встроенной базе данных TSDB;
- Экспортеры данных, которые по запросу сервера снимают метрики с заданных сервисов (targets) и возвращают их серверу;
- Web UI, с помощью которого можно исследовать собранные метрики с помощью языка запросов promQL.

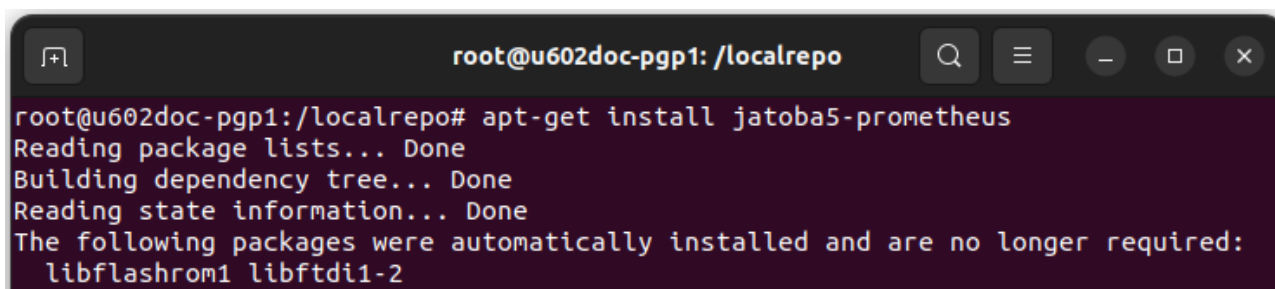
Документация по использованию системы находится на официальном сайте разработчика по адресу: <https://prometheus.io>.

7.1. Установка системы «Prometheus»

Документация по использованию системы находится на официальном сайте разработчика по адресу: <https://prometheus.io>.

Установка пакета выполняется в соответствии с Руководством по установке, из локального репозитория командой:

```
apt-get install jatoba<ver>-prometheus
```



```
root@u602doc-pgp1: /localrepo
root@u602doc-pgp1:/localrepo# apt-get install jatoba5-prometheus
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libflashrom1 libftdi1-2
```

Рисунок 7.1 – Установка пакета «jatoba<ver>-prometheus»

В результате установки пакета будет создан:

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

- файл переменных окружения сервиса по адресу:

```
usr/jatoba-5/monitoring/default/prometheus
```

- файл сервиса по адресу

```
usr/lib/systemd/system/jatoba-5_prometheus.service
```

- файл конфигурации, адаптированный под использование с СУБД «Jatoba» по адресу:

```
usr/jatoba-5/monitoring/default/prometheus.yml
```

- база данных по адресу:

```
/opt/prometheus
```

- служебные директории веб-консоли по адресу:

```
usr/jatoba-5/monitoring/prometheus
```

- пользователь ОС «prometheus», от которого будет производиться запуск сервиса.

У данного пользователя нет интерактивной оболочки для входа и нет домашней директории.

7.2. Конфигурация системы «Prometheus»

Необходимо задать конфигурацию сервера в формате YAML выполнив команду редактирования:

```
gedit /usr/jatoba-5/monitoring/default/prometheus.yml
```

В конфигурации важными параметрами являются:

- частота опроса метрик (scrape_interval);
- время ожидания ответа (scrape_timeout);
- HTTP, IP адреса (targets).

Для параметра «targets» возможно указать одну или несколько целей, для получения метрик с экспортера, при этом параметр будет иметь синтаксис, с одной целью:

```
- targets: ['X.X.X.X:port']
```

и с несколькими целями

```
- targets: ['X.X.X.X:port', 'X.X.X.X:port']
```

В рассматриваемом примере, в конфигурационном файле `prometheus.yml` устанавливаются IP-адреса серверов, находящихся под наблюдением.



Обратите внимание, что необходимо прописывать общий, а не локальный адрес сетевого интерфейса

Ниже приведены примеры таких «job-name».

7.2.1. Примеры блока «postgres-exporter»

Пример стандартного «postgres-exporter» экспортера с двумя целями:

```
# стандартный экспортер данных для PostgreSQL
- job_name: "postgres-exporter"
  metrics_path: '/metrics'
  static_configs:
    - targets: ['10.116.102.45:9187', '10.116.102.47:9187']
      labels:
        alias: postgres
```

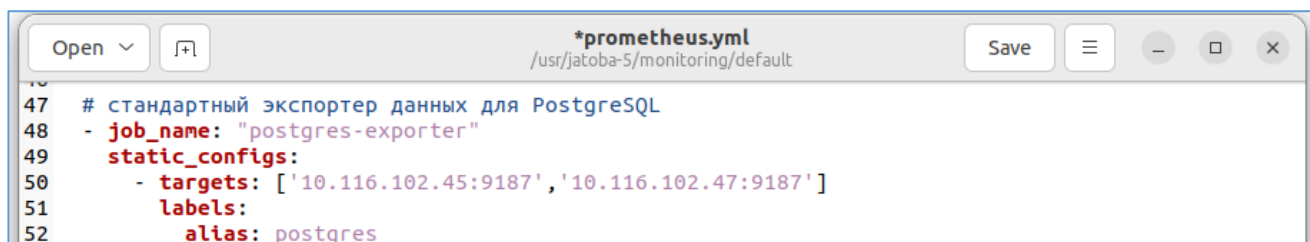


Рисунок 7.2 - Стандартный экспортер данных для PostgreSQL

Пример стандартного «postgres-exporter» экспортера для двух «job-name» с одной и двумя целями:

```
# Экспортер данных для PostgreSQL сервера srv1
- job_name: "srv1-postgres-exporter"
  metrics_path: '/metrics'
  static_configs:
    - targets: ['10.116.103.45:9187']
      labels:
```

```
alias: postgres

# Экспортер данных для PostgreSQL сервера srv2 и srv3
- job_name: "srv2-srv3-postgres-exporter"
  metrics_path: '/metrics'
  static_configs:
    - targets: ['10.116.103.46:9187', '10.116.103.47:9187']
      labels:
        alias: postgres
```

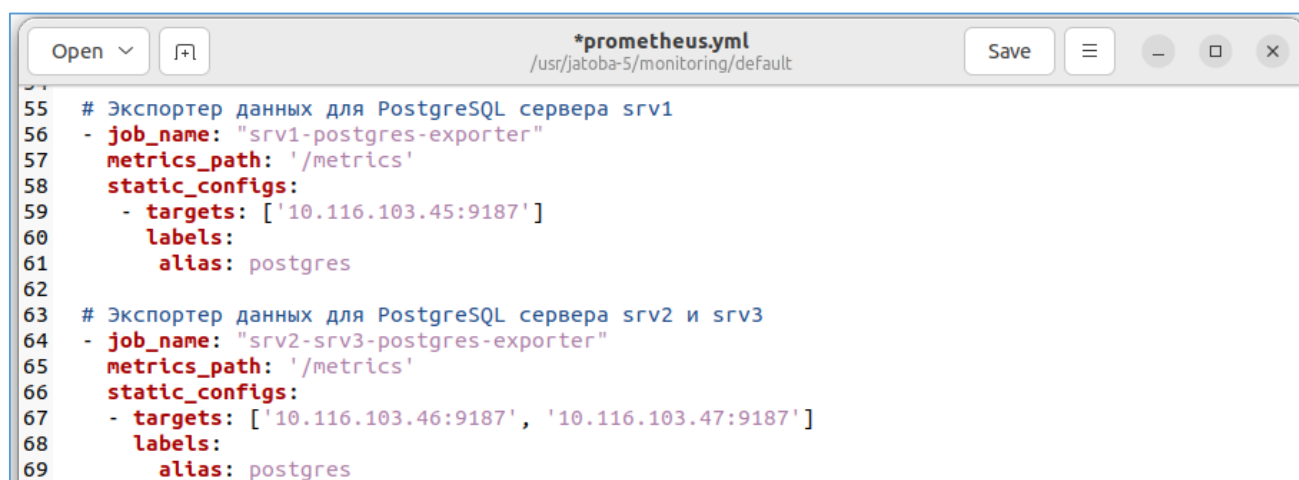


Рисунок 7.3 - «postgres-exporter» экспортера для двух «job-name» с одной и двумя целями

7.2.2. Примеры блока «sql-exporter»

Пример экспортера данных для SQL с двумя целями:

```
# экспортер данных для SQL
- job_name: "sql-exporter"
  metrics_path: '/metrics'
  static_configs:
    - targets: ['10.116.102.45:9399', '10.116.102.47:9399']
      labels:
        alias: postgres
```

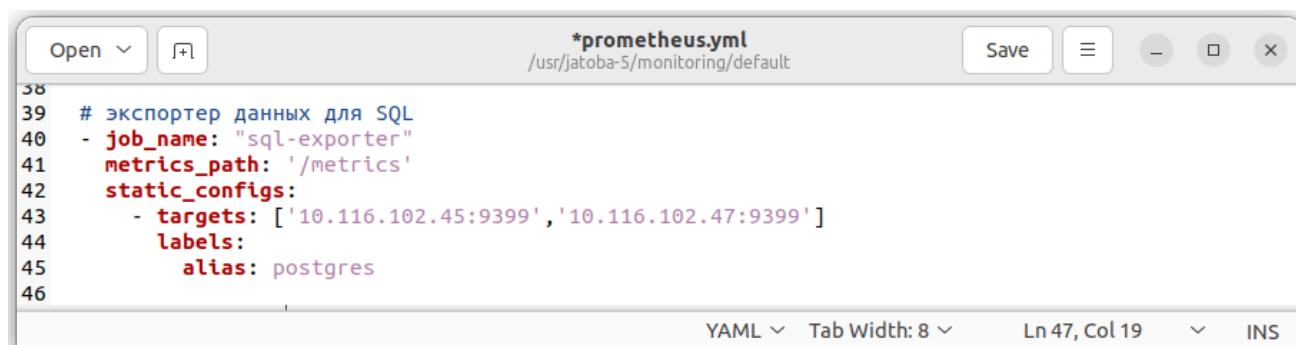


Рисунок 7.4 - «sql-exporter» с двумя целями

Пример экспортера SQL для двух «job-name» с одной и двумя целями:

```
# Экспортер данных для SQL сервера srv1
- job_name: "srv1-sql-exporter"
  metrics_path: '/metrics'
  static_configs:
    - targets: ['10.116.103.45:9399']
      labels:
        alias: postgres

# Экспортер данных для SQL сервера srv2 и srv3
- job_name: "srv2-srv3-sql-exporter"
  metrics_path: '/metrics'
  static_configs:
    - targets: ['10.116.103.46:9399', '10.116.103.47:9399']
      labels:
        alias: postgres
```



Рисунок 7.5 – «sql-exporter» для двух «job-name» с одной и двумя целями

7.2.3. Примеры блока «node-exporter»

Пример экспортера данных "node-exporter" для GNU/ Linux с двумя целями:

```
# экспортер данных для Linux
- job_name: "node-exporter"
  metrics_path: '/metrics'
  static_configs:
    - targets: ['10.116.102.45:9100', '10.116.102.47:9100']
      labels:
        alias: os
```

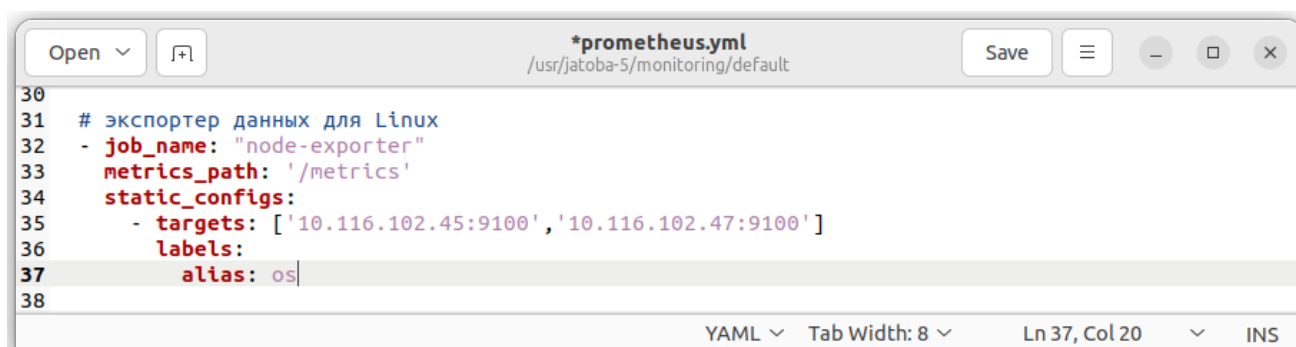



Рисунок 7.6 – «node-exporter» для GNU/ Linux с двумя целями

Пример экспортера данных «node-exporter» для GNU/Linux для двух «job-name» с одной и двумя целями:

```

# Экспортер данных для Linux сервера srv1
- job_name: "srv1-node-exporter"
  metrics_path: '/metrics'
  static_configs:
    - targets: ['10.116.103.46:9100']
      labels:
        alias: os

# Экспортер данных для Linux сервера srv2 и srv3
- job_name: "srv2-srv3-node-exporter"
  metrics_path: '/metrics'
  static_configs:
    - targets: ['10.116.103.46:9100', '10.116.103.47:9100']
      labels:
        alias: os

```

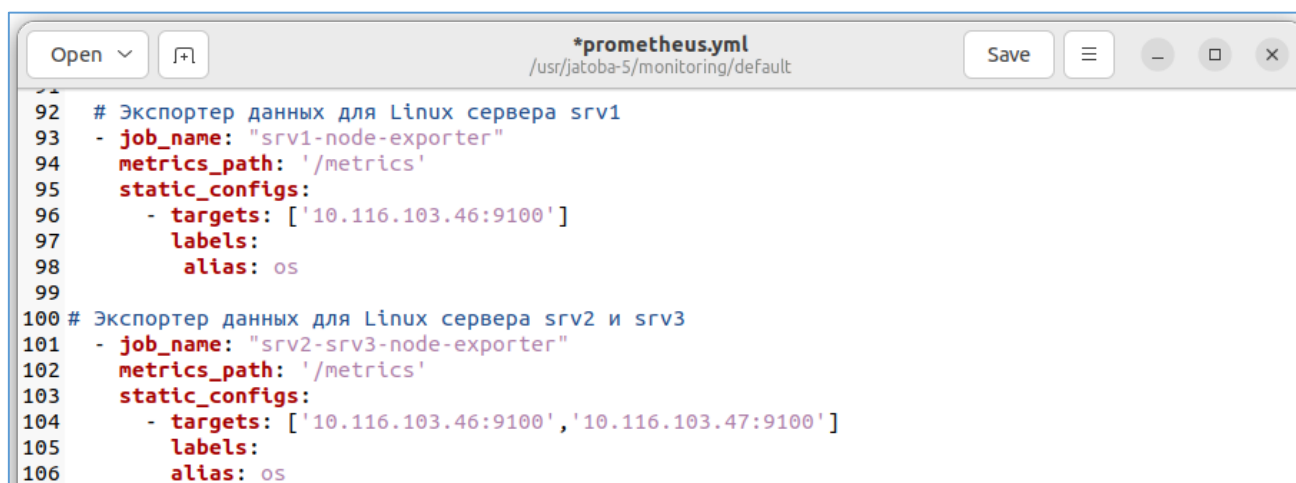


Рисунок 7.7 – «node-exporter» для GNU/Linux для двух «job-name» с одной и двумя целями

В конфигурационном файле `prometheus.yml`, в строке «targets», допустимо указывать любое количество адресов экспортеров, относящихся к одной или разным СУБД.

Обработка критических событий и вычисление rules не заданы, хотя соответствующие блоки присутствуют в конфигурации.

Полное описание параметров конфигурации сервера приведено в документации <https://prometheus.io/docs/prometheus/latest/configuration/configuration/>

7.3. Запуск системы «Prometheus»

Перед запуском сервиса требуется удостовериться в корректности содержания файла сервиса.

Просмотр файла осуществляется командой в терминале ОС:

```
gedit usr/lib/systemd/system/jatoba-5_prometheus.service
```

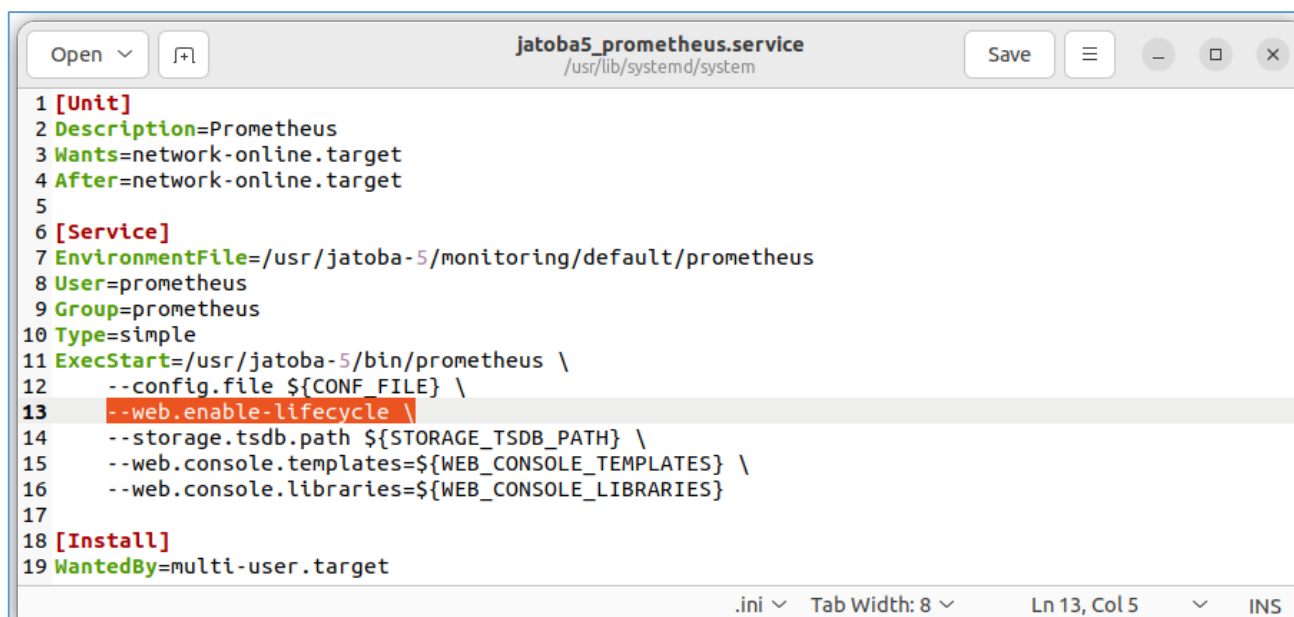


Рисунок 7.8 – Содержание файла сервиса «jatoba-5_prometheus.service»

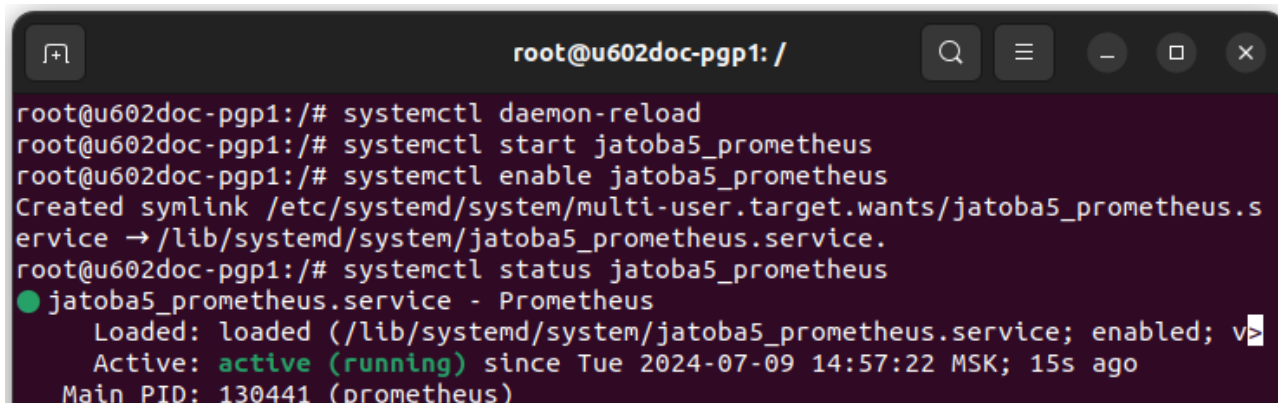
Обновить конфигурацию systemd:

```
# systemctl daemon-reload
```

Запустить службу система, включить ее в автозапуск и проверить статус работы:

```
# systemctl start jatoba5_prometheus
# systemctl enable jatoba5_prometheus
```

```
# systemctl status jatoba5_prometheus
```



```
root@u602doc-pgp1: /
root@u602doc-pgp1:/# systemctl daemon-reload
root@u602doc-pgp1:/# systemctl start jatoba5_prometheus
root@u602doc-pgp1:/# systemctl enable jatoba5_prometheus
Created symlink /etc/systemd/system/multi-user.target.wants/jatoba5_prometheus.s
ervice → /lib/systemd/system/jatoba5_prometheus.service.
root@u602doc-pgp1:/# systemctl status jatoba5_prometheus
● jatoba5_prometheus.service - Prometheus
   Loaded: loaded (/lib/systemd/system/jatoba5_prometheus.service; enabled; v
   Active: active (running) since Tue 2024-07-09 14:57:22 MSK; 15s ago
   Main PID: 130441 (prometheus)
```

Рисунок 7.9 Установка и запуск службы системы «Prometheus»

Статус запущенного сервера «Prometheus» можно проверить с помощью web UI, открыв в браузере страницу с адресом <http://localhost:9090>.

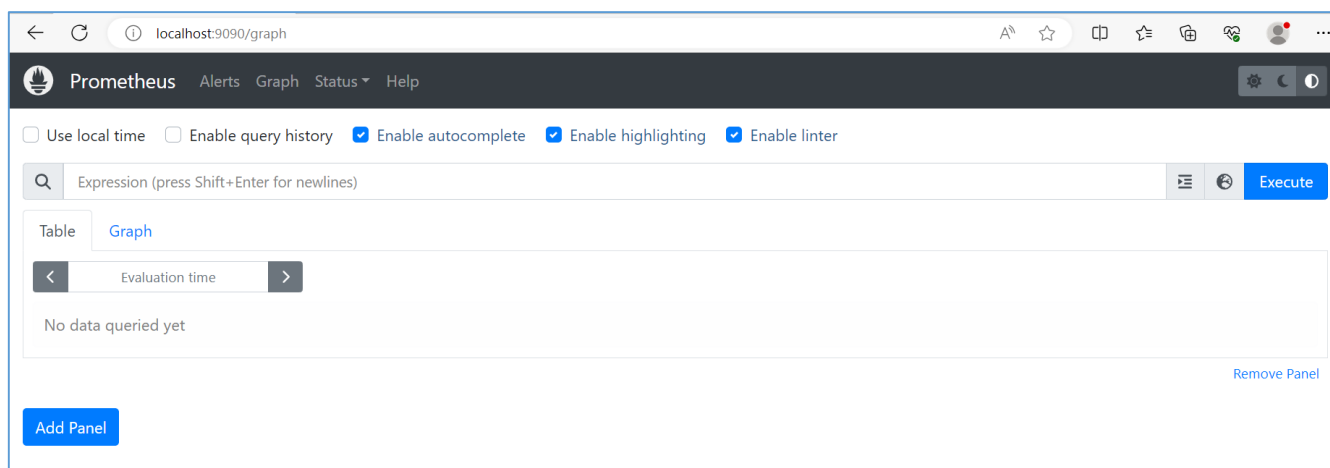


Рисунок 7.10 – Веб интерфейс системы «Prometheus»

На вкладке «Status» можно посмотреть текущую конфигурацию и опции запуска сервера, статус встроенной базы данных tsdb и заданные цели (targets).

В окне «Expression» можно ввести название метрики или выражение на языке promQL и, нажатием на кнопку «Execute», отобразить результаты в виде таблицы или графика.

Руководство по языку запросов promQL располагается по адресу: <https://prometheus.io/docs/prometheus/latest/querying/basics/>

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

Полный список собираемых метрик можно открыть при нажатии на кнопку «Open Metrics Explorer» слева от кнопки «Execute» или отобразить в окне «Expressions» при вводе первых символов наименования метрики, если включена опция автодополнения.

Список значений собираемых метрик для каждой цели можно отобразить на странице веб-интерфейса соответствующего экспортера данных, например, localhost:9090/metrics.

The screenshot shows the Prometheus 'Targets' page. It lists three scrape targets, all of which are 'UP'. Each target has a table of labels, a 'Last Scrape' time, a 'Scrape Duration', and an 'Error' field.

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
node-exporter (1/1 up) show less					
http://localhost:9100/metrics	UP	alias="os" instance="localhost:9100" job="node-exporter"	26.282s ago	12.156ms	
postgres-exporter (1/1 up) show less					
http://localhost:9187/metrics	UP	alias="postgres" instance="localhost:9187" job="postgres-exporter"	6.890s ago	1.9s	
sql-exporter (1/1 up) show less					
http://localhost:9399/metrics	UP	alias="postgres" instance="localhost:9399" job="sql-exporter"	1.496s ago	7.645ms	

Рисунок 7.11 – Список значений собираемых метрик

В рассматриваемом примере подключение к системе «Prometheus» используется адрес:

http://10.116.102.41:9090/

Перейдя в меню «Target» отразятся статистические данные наблюдаемых СУБД.

Targets

All scrape pools ▾ All Unhealthy Collapse All 🔍 Filter by endpoint or labels

Unknown Unhealthy Healthy

node-exporter (2/2 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.116.102.45:9100/metrics	UP	alias="os" instance="10.116.102.45:9100" job="node-exporter" ▾	5.404s ago	11.536ms	
http://10.116.102.47:9100/metrics	UP	alias="os" instance="10.116.102.47:9100" job="node-exporter" ▾	18.15s ago	10.563ms	

postgres-exporter (2/2 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.116.102.45:9187/metrics	UP	alias="postgres" instance="10.116.102.45:9187" job="postgres-exporter" ▾	18.958s ago	1.2s	
http://10.116.102.47:9187/metrics	UP	alias="postgres" instance="10.116.102.47:9187" job="postgres-exporter" ▾	6.189s ago	30.167ms	

Рисунок 7.12 – Страница целей системы «Prometheus»

Система «Prometheus» может быть запущен в Docker с использованием готового образа из репозитория, например:

```
docker pull prom/prometheus-linux-amd64
docker run -p 9090:9090 --name prometheus prom/prometheus-
linux-amd64
```

8. УТИЛИТА «ALERTMANAGER»

Alertmanager — это инструмент для управления и обработки оповещений в системе мониторинга Prometheus. Он выполняет следующие функции:

- группировка оповещений: группирует похожие оповещения для снижения шума и предотвращения дублирования.
- удаление дубликатов: гарантирует отправку уникальных оповещений без повторений.
- маршрутизация и приглушение оповещений: позволяет определять правила и конфигурации для маршрутизации оповещений нужным получателям на основе их важности или других критериев. Также можно временно приглушить оповещения во время обслуживания или определённых периодов.
- уведомление о тревоге: интегрируется с различными каналами связи, такими как электронная почта, Slack, PagerDuty и другие, позволяя отправлять уведомления о тревогах нужным людям или командам.

8.1. Установка утилиты и службы «alertmanager»

Установка пакета выполняется в соответствии с Руководством по установке, из локального репозитория командой:

```
apt-get install jatoba<ver>-alertmanager
```

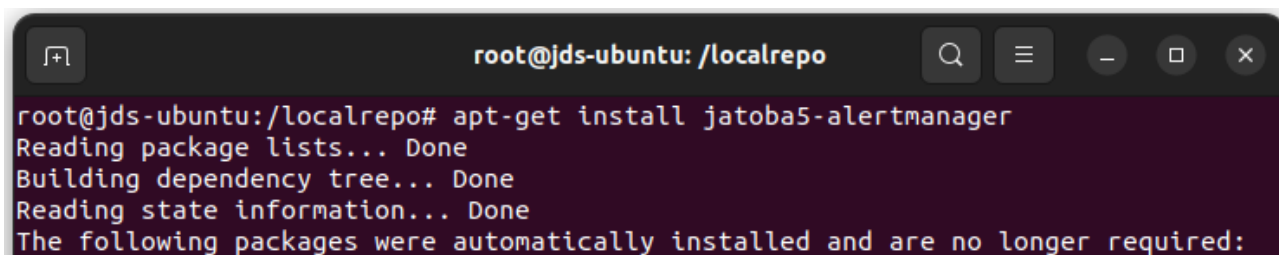


Рисунок 8.1 – Установка пакета «jatoba*-alertmanager»

В результате установки пакета будет создан:

- файл запуска по адресу:

```
/usr/jatoba-<ver>/bin/alertmanager
```

- конфигурационный файл по адресу:

```
/usr/jatoba-<ver>/monitoring/default/alertmanager.yml
```

- служба по адресу:

```
/usr/lib/systemd/jatoba<ver>_alertmanager.service
```

- пользователь ОС «alertmanager», от которого будет производиться запуск сервиса.

У данного пользователя нет интерактивной оболочки для входа и нет домашней директории.

8.2. Настройка параметров утилиты файле «alertmanager.yml»

После установки пакета в конфигурационном файле будут установлены параметры по умолчанию.

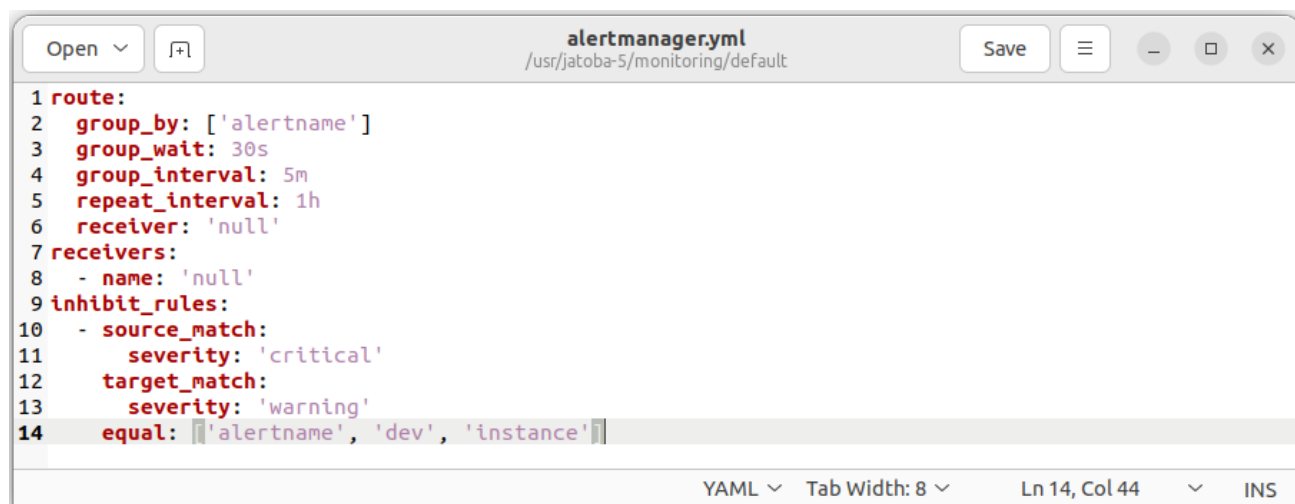


Рисунок 8.2 – Параметры по умолчанию

Редактирование конфигурационного файла выполняется командой:

```
gedit /usr/jatoba-<ver>/monitoring/default/alertmanager.yml
```

В узле «global» необходимо указать данные для подключения к почтовому серверу. Целесообразно использовать специальную, неперсонофицированную, техническую учетную запись почты, от имени которой будет рассылаться предупреждения.

В узле route указываются настройки агрегирования предупреждений.

В узле «receivers» в узле «email_general» указываются настройки получателя. Используется общий узел для всех получателей, email получателя подставляется из метки «emailto» с помощью шаблона.

```
global:
  smtp_smarthost: mail.domain.ru:587
  smtp_from: domain_name@domain.ru
  smtp_auth_user_name: user_name@domain.ru
  smtp_auth_password: password
  smtp_require_tls: true
route:
  receiver: email_general
  group_by: [emailto]
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 3h
receivers:
  - name: email_general
    email_configs:
      - send_resolved: true
        to: '{{ .CommonLabels.emailto }}'
```

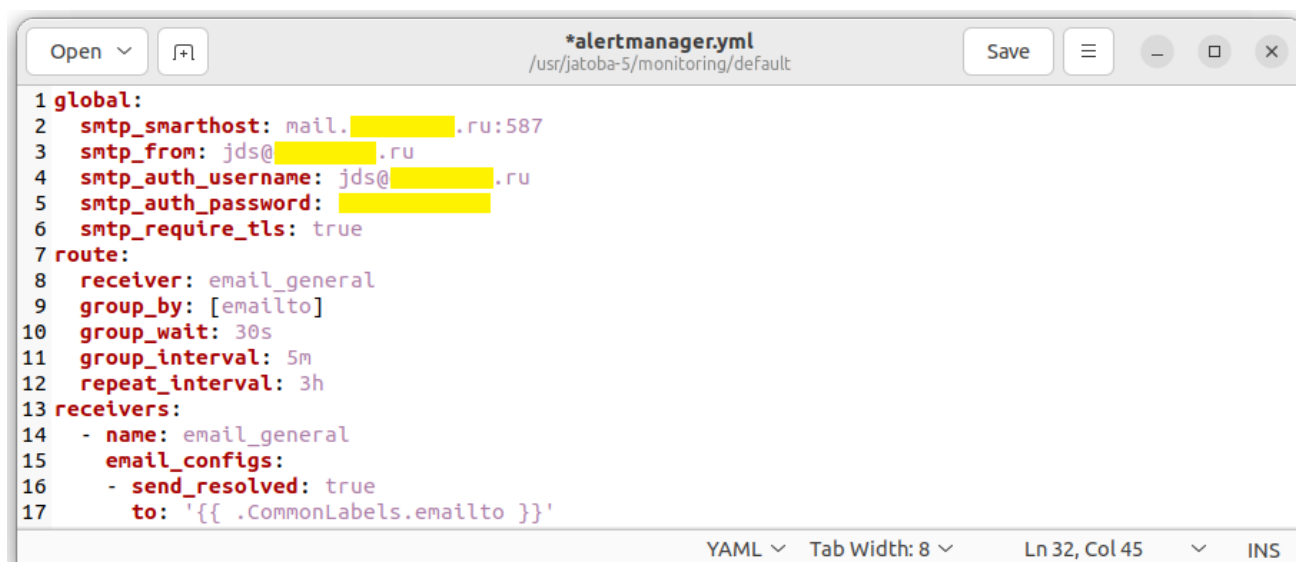


Рисунок 8.3 - Конфигурационного файла «alertmanager.yml»

8.3. Запуск утилиты «alertmanager»

Обновить конфигурацию system командой:

```
# sudo systemctl daemon-reload
```

Запустить службу утилиты, включить ее автозапуск и проверить статус работы:

```
# systemctl start jatoba<ver>_alertmanager
# systemctl enable jatoba<ver>_alertmanager
# systemctl status jatoba<ver>_alertmanager
```



```
root@jds-ubuntu: /
root@jds-ubuntu:/# sudo systemctl daemon-reload
root@jds-ubuntu:/# systemctl start jatoba5_alertmanager
root@jds-ubuntu:/# systemctl enable jatoba5_alertmanager
Created symlink /etc/systemd/system/multi-user.target.wants/jatoba5_alertmanager.service → /lib/systemd/system/jatoba5_alertmanager.service.
root@jds-ubuntu:/# systemctl status jatoba5_alertmanager
● jatoba5_alertmanager.service - Alertmanager
   Loaded: loaded (/lib/systemd/system/jatoba5_alertmanager.service; enabled;
   Active: active (running) since Wed 2024-07-10 12:33:43 UTC; 27s ago
```

Рисунок 8.4 - Запуск и вывод статуса службы «jatoba*_alertmanager»

Чтобы проверить статус работы утилиты, нужно в браузере открыть веб-интерфейс утилиты «Alertmanager»:

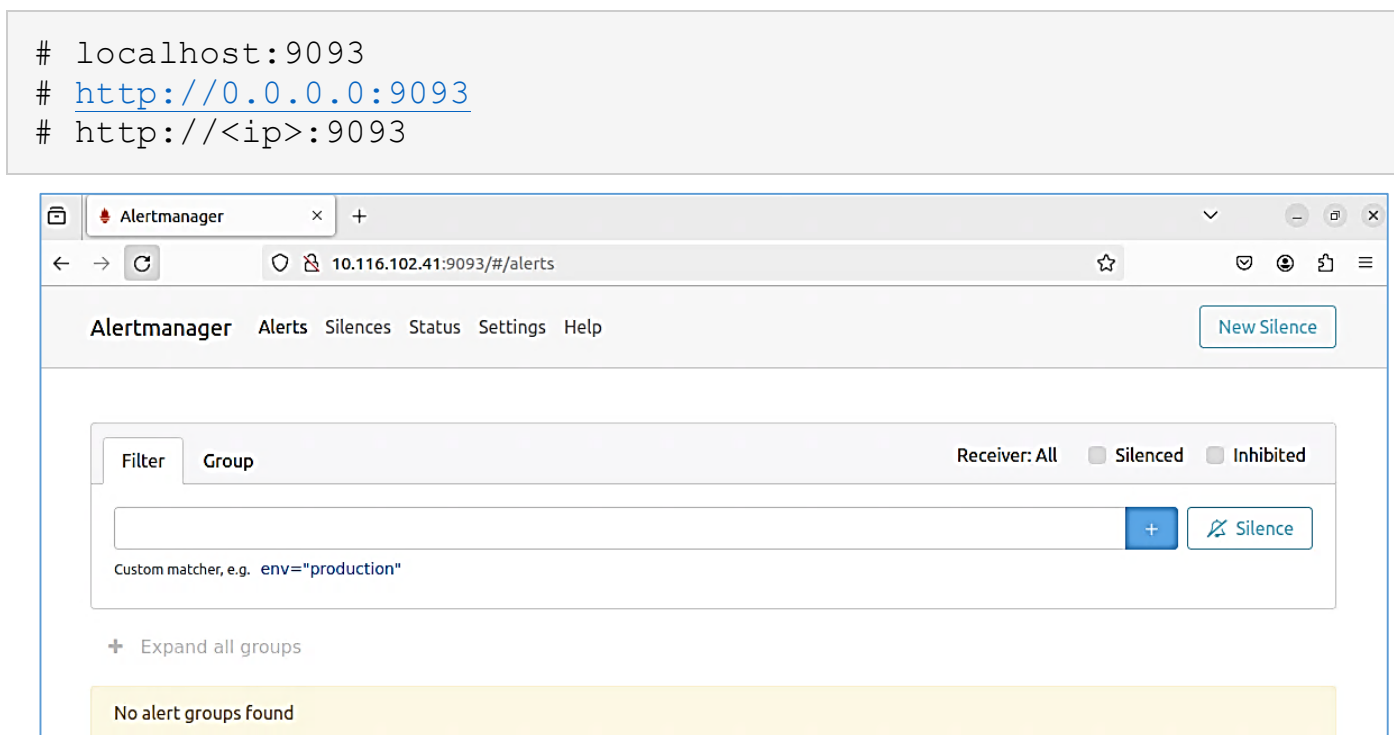


Рисунок 8.5 – Веб-интерфейс утилиты «alertmanager»

На данном шаге конфигурирование утилиты не закончено. Проверена, только работоспособность. Интеграция с другими компонентами описана в разделе 9 «Подключение к JDS».

9. ПОДКЛЮЧЕНИЕ К JDS

Подключение хранилища системы «Prometheus» к компоненту «Jatoba data safe» для отображения в разделе «Мониторинг» описано в документе «Руководство по настройке. Часть 7. Пользовательский веб-интерфейс для администраторов. Компонент «Jatoba data safe», в пункте «Источники данных».

Для настройки «Уведомлений» о контролируемых значениях СУБД требуется сконфигурировать 3 компонента, такие как система «Prometheus», утилита «Alertmanager» и компоненты пользовательского веб-интерфейса для администраторов «Jatoba data safe» (JDS).

Последовательность действий будет следующая.

Настраивается SSH-соединение на хосте и/или с хостом с установленной системой «Prometheus» (см. п. 9.1).

В разделе «Настройки» компонента JDS, во вкладке «Источник данных» созданное подключение к системе «Prometheus» дополняется параметрами «Настройки конфигурации предупреждений» (уведомлений).

В этой настройке указывается, IP адрес системы «Prometheus», порт подключения, пользователь и путь к файлу с правилами уведомлений. В последствии это имя файла будет использовано в конфигурировании системы «Prometheus».

Файл с правилами уведомлений предварительно не создаётся и появляется по вышеуказанному пути. Поэтому для его создания требуется создать уведомление в разделе «Мониторинг» в любом из дашбордов.

На хосте с системой «Prometheus» в конфигурационном файле

```
usr/jatoba-5/monitoring/default/prometheus.yml
```

связать систему «Prometheus» и утилиту «Alertmanager».

9.1. Настройка SSH-соединения

Настройка SSH-соединения производится в обязательном порядке для любой архитектуры компонентов. В том числе, если утилита «Alertmanager», система «Prometheus» и JDS установлены на одном хосте.

Необходимо настроить SSH-соединение с хоста компонента JDS на сервер с развернутой системой «Prometheus». Соединение будет использоваться компонентом JDS для копирования конфигурационного файла с правилами предупреждений.

В настройках SSH-сервера должны быть разрешены локальные подключения и подключения от имени и с правами пользователя «root».

Следует выполнить следующие действия:

- создать папку пользователя, под которым работает JDS:

```
sudo -s  
mkdir /home/jds  
chown jds /home/jds  
exit
```

- сгенерировать ключи под пользователем JDS, скопировать на хост с системой «Prometheus»:

```
sudo -u jds /usr/bin/bash  
ssh-keygen  
(задать пустой пароль)  
ssh-copy-id root@IP  
(yes)
```

- проверить соединение (должно соединиться без запроса пароля):

```
ssh root@IP  
exit  
exit
```

9.2. Конфигурирование JDS

Вкладка «Источник данных»

На хосте с установленным компонентом JDS перейти в раздел «Настройки». Созданной подключение к системе «Prometheus» изменить, дополнив параметрами «Настройки конфигурации предупреждений» (уведомлений).

В этой настройке указывается:

- IP адрес системы «Prometheus»;
- порт подключения – 22, соответствующий SSH-подключению;
- пользователь – root;
- путь к файлу с правилами уведомлений:

```
/usr/jatoba-5/monitoring/default/alertrules.yml
```

В последствии это имя файла будет использовано в конфигурировании системы «Prometheus».

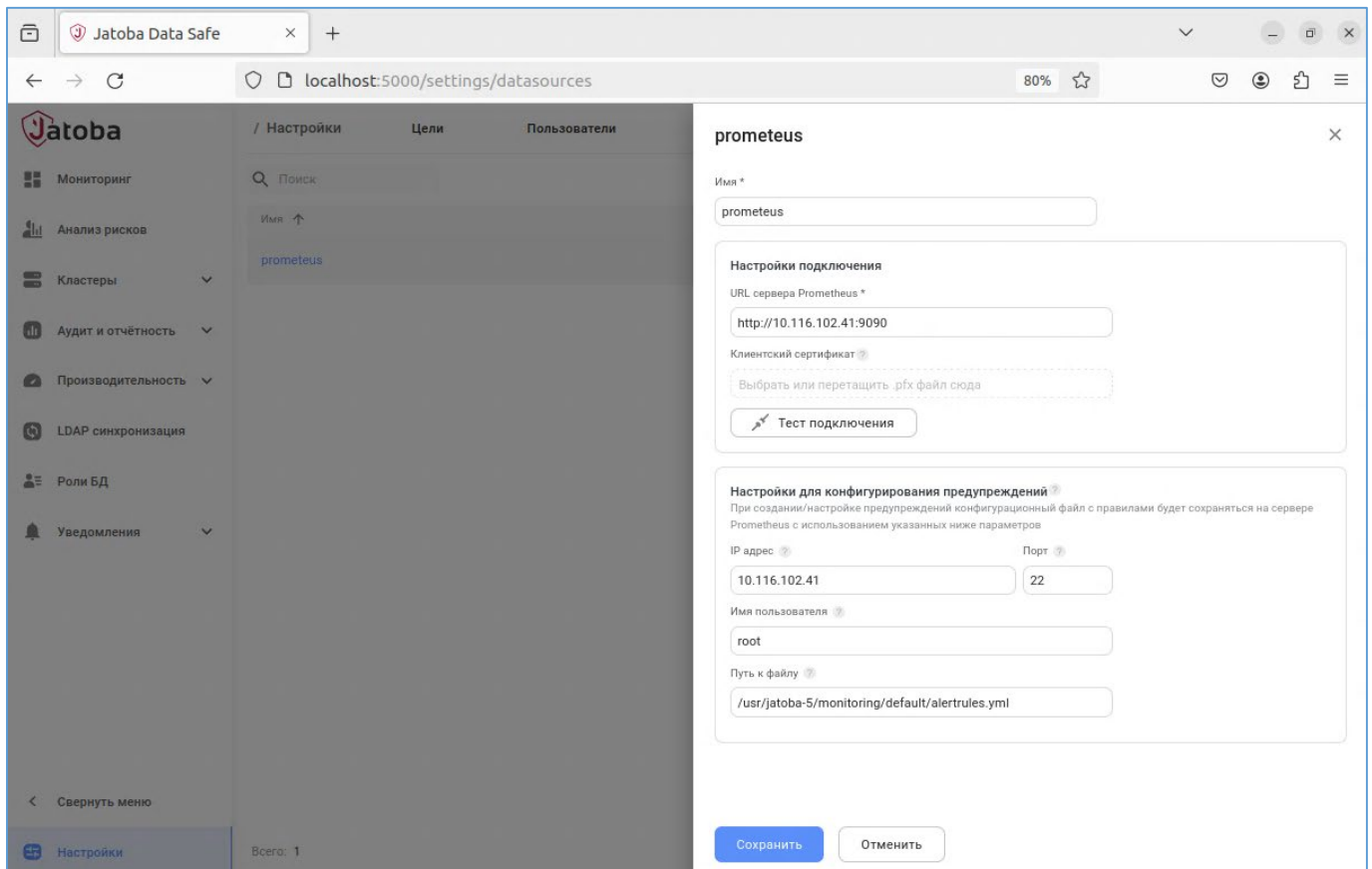


Рисунок 9.1 - «Настройки конфигурации предупреждений»

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

Раздел «Мониторинг»

В разделе «Мониторинг» создав уведомление в любом из дашбордов с динамическими данными будет сформирован файл с правилами уведомлений по пути указанному в настройках «Источника данных» в сформированном подключении к системе «Prometheus»:

```
/usr/jatoba-5/monitoring/default/alertrules.yml
```

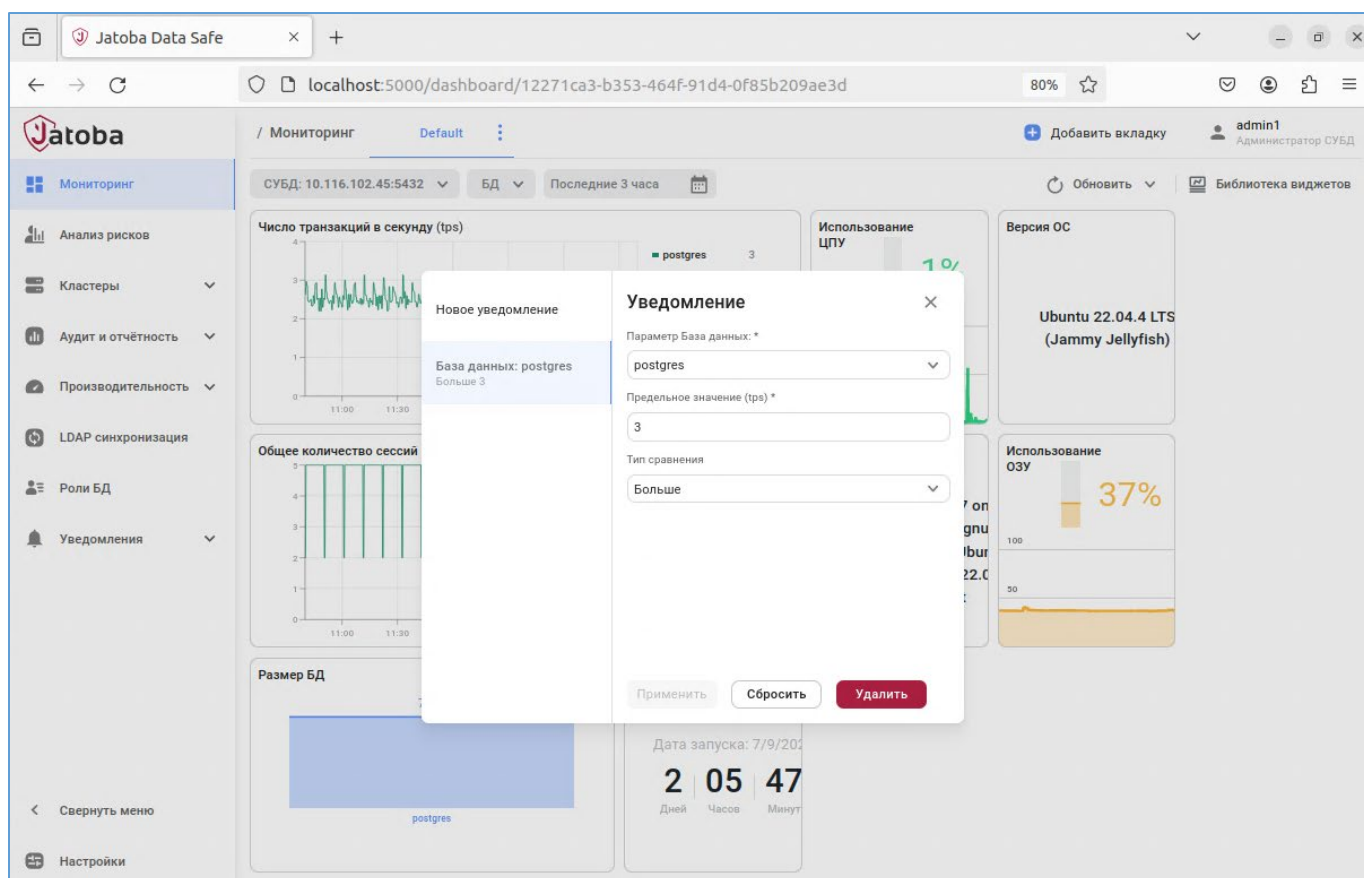


Рисунок 9.2 – Создание уведомления в компоненте JDS

9.3. Настройка связки системы «Prometheus» и утилиты «Alertmanager»

Имея данные конфигурации и конфигурационный файл уведомлений можно приступить к связке системы «Prometheus» и утилиты «Alertmanager», для чего надо выполнить команду редактирования конфигурационного файла системы «Prometheus»:

```
gedit usr/jatoba-5/monitoring/default/prometheus.yml
```

Соответствующий раздел «Alertmanager configuration» находится в начале файла и параметры надо внести именно в него. Вставка параметров в конец файла может привести к ошибке.

В узле «targets» указывается хост или хосты с установленной утилитой «Alertmanager»

В узле rule_files необходимо указать имя конфигурационного файла уведомлений

```
alerting:
  alertmanagers:
    - static_configs:
      - targets:
        - IP**.*.*.*.*:9093
rule_files:
  - "alertrules.yml"
```

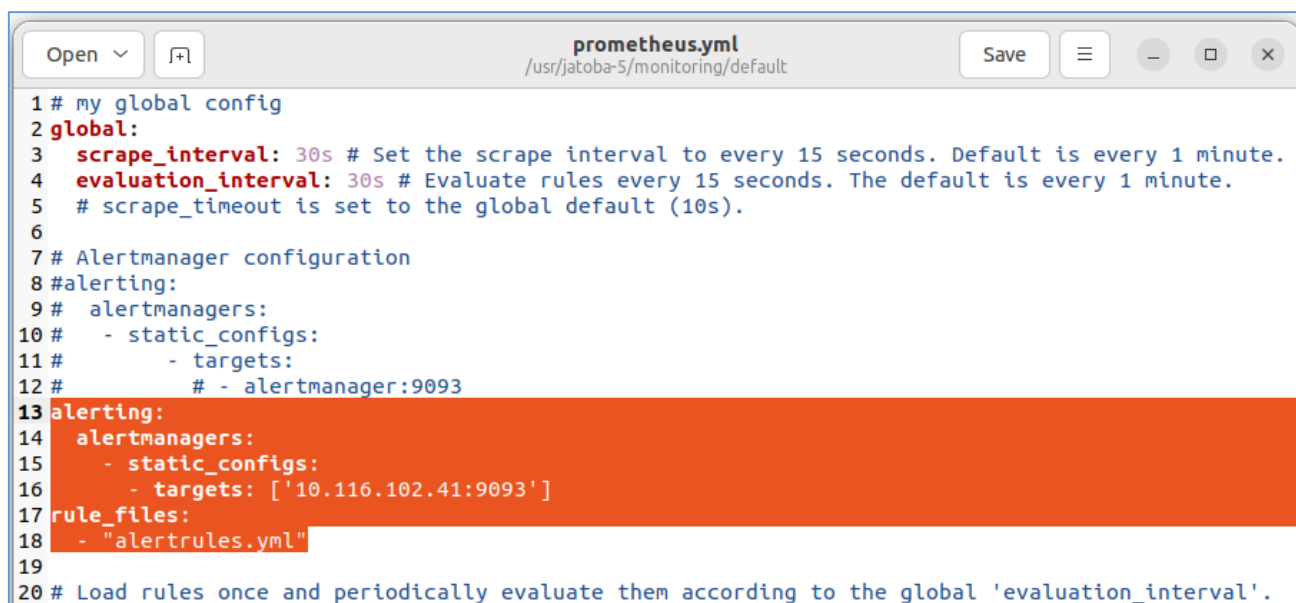


Рисунок 9.3 - Раздел «Alertmanager configuration»

Проверить корректность введенных параметров возможно командами:

```
# cd /usr/jatoba-5/bin#  
# ./promtool check config /usr/jatoba-  
5/monitoring/default/prometheus.yml
```

Если параметры верны, перезапустить службу:

```
# systemctl restart jatoba5_prometheus
```

На данном шаге конфигурирование раздела «Мониторинг» компонента JDS закончено.

10. СИСТЕМА «GRAFANA»



Установка системы «Grafana» для работы компонента «Jatoba data safe» не требуется

«Grafana» – графическая система визуализации данных, интегрируемая с системой «Prometheus». Сервер «Grafana» может быть установлен на любом компьютере, с которого есть доступ к серверу «Prometheus».

10.1. Установка системы «Grafana»

На официальном сайте <https://grafana.com/grafana/download> доступны сборки «Grafana» для разных операционных систем, включая Windows и образы Docker.

Документация по установке системы находится на официальном сайте разработчика по адресу: <https://grafana.com/docs/grafana/latest/setup-grafana/installation/debian/>.

Для каждого варианта на сайте приведены подробные инструкции по установке.

Например, для запуска образа «Grafana» в Docker-контейнере требуется выполнить:

```
docker run -d --name=grafana -p 3000:3000 grafana/grafana-enterprise
```

Пример установки системы приведен в Приложении 1 настоящего документа.

10.2. Запуск системы «Grafana»

По умолчанию веб-интерфейс «Grafana» имеет адрес <http://localhost:3000>. Открыв в браузере эту страницу, при первом входе, потребуется ввести имя пользователя «admin» и пароль «admin».

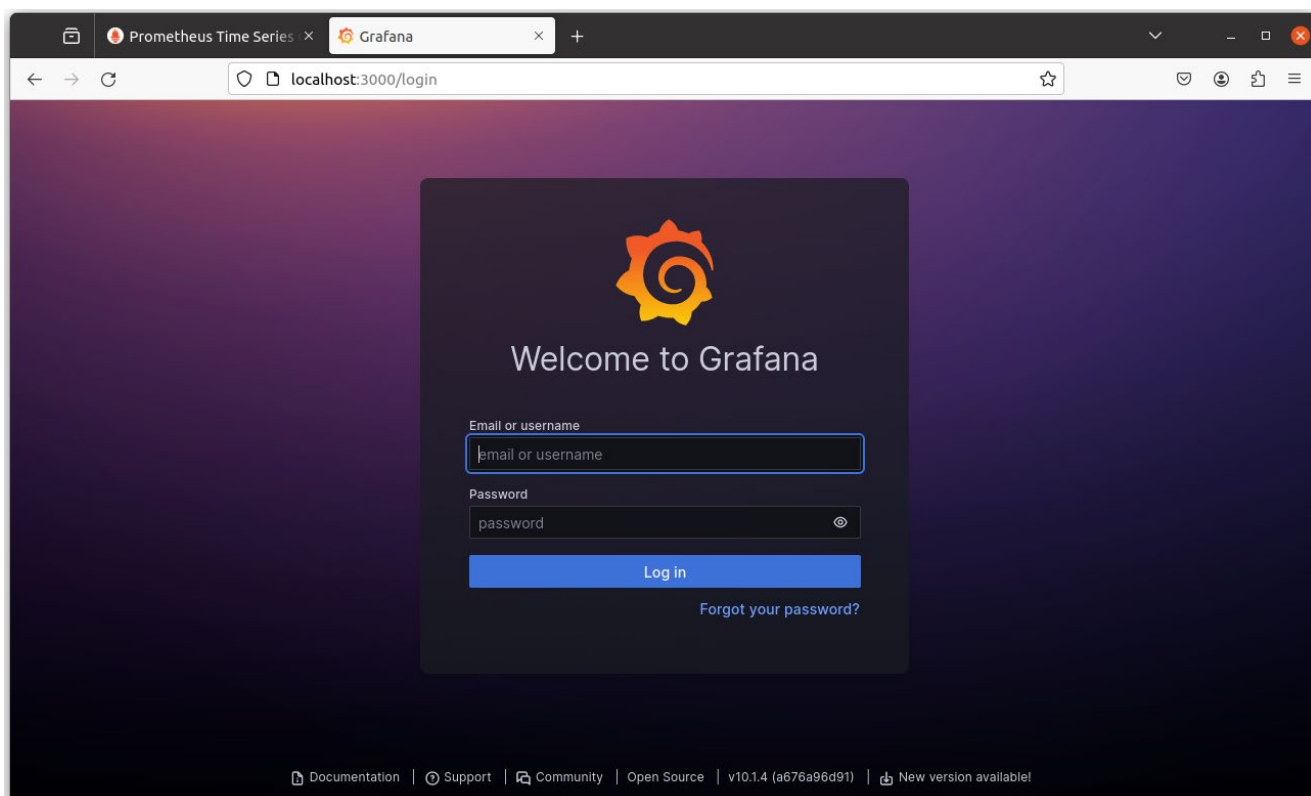


Рисунок 10.1 – Стартовое окно

В следующем окне система предложит сменить пароль, заданный по умолчанию.

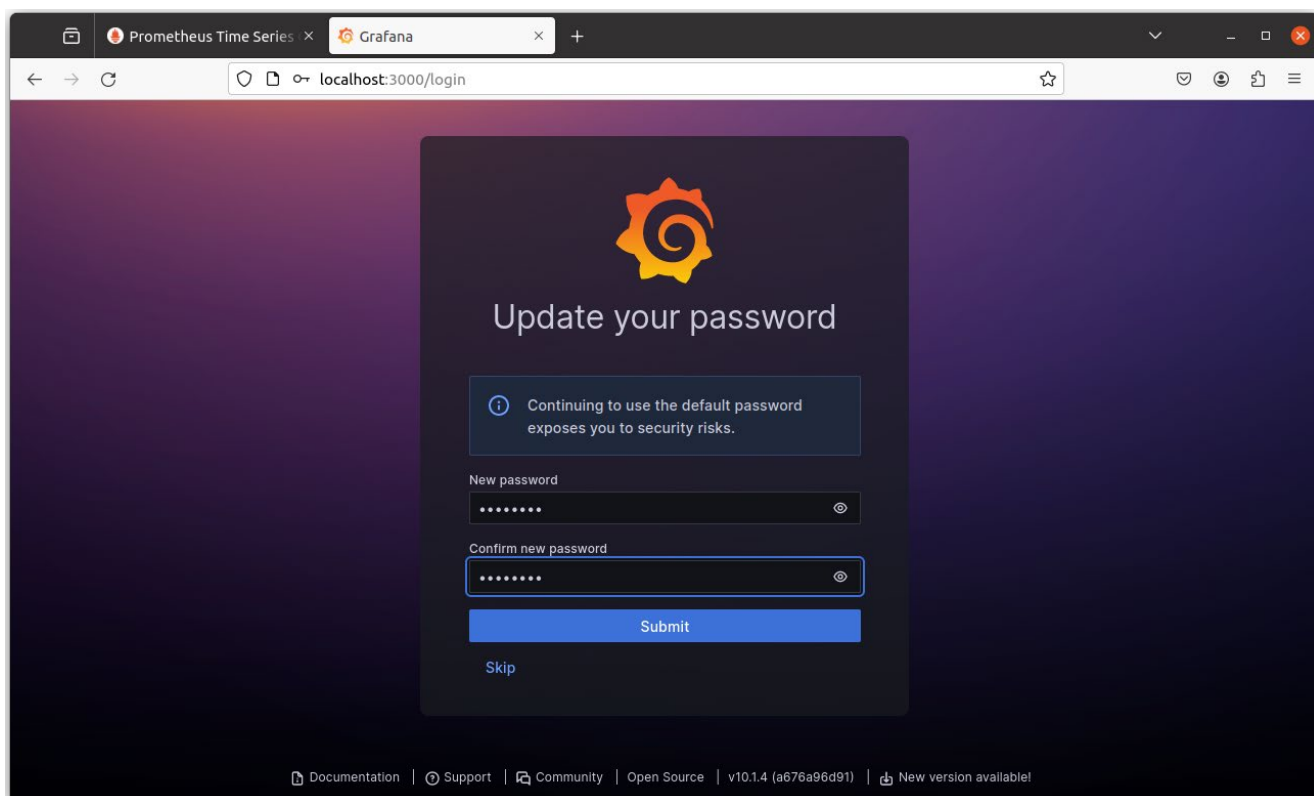


Рисунок 10.2 – Окно смены пароля

Открыть боковое меню, нажав по значку главного меню в левом верхнем углу.

Нажать в открывшемся меню «Connections» → «Data Sources».

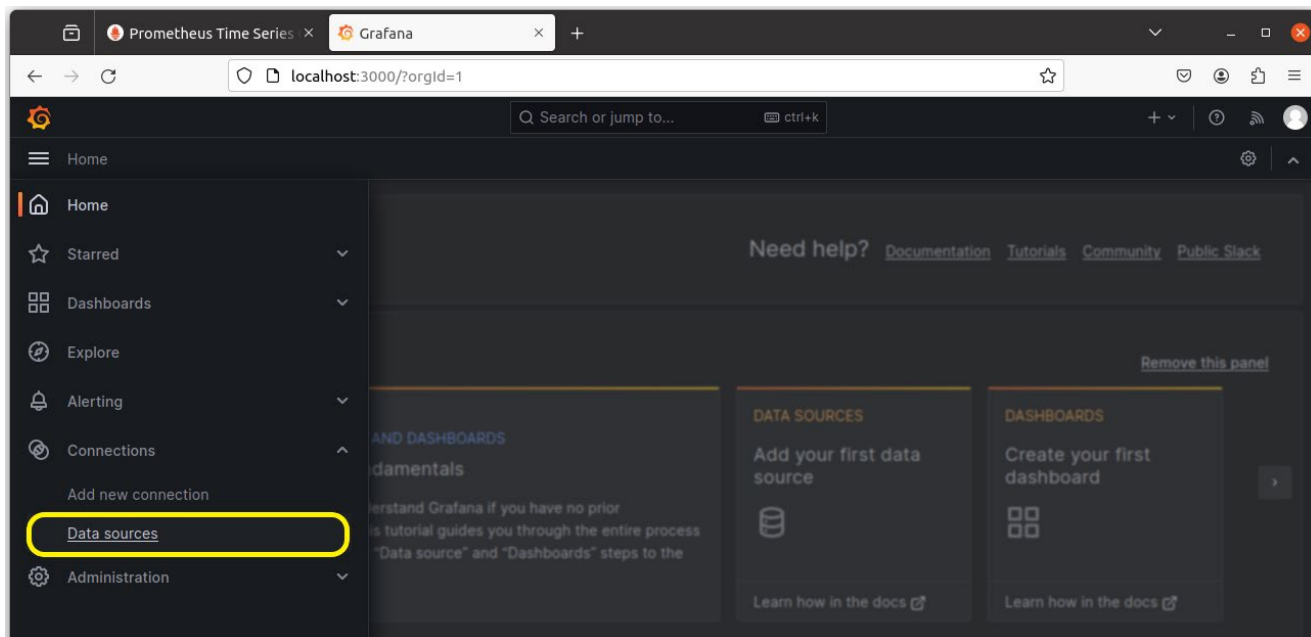


Рисунок 10.3 – Расположение меню «Data Sources»

Нажать кнопку «Add new data source» в верхнем правом углу и выбрать в списке сервер «Prometheus».

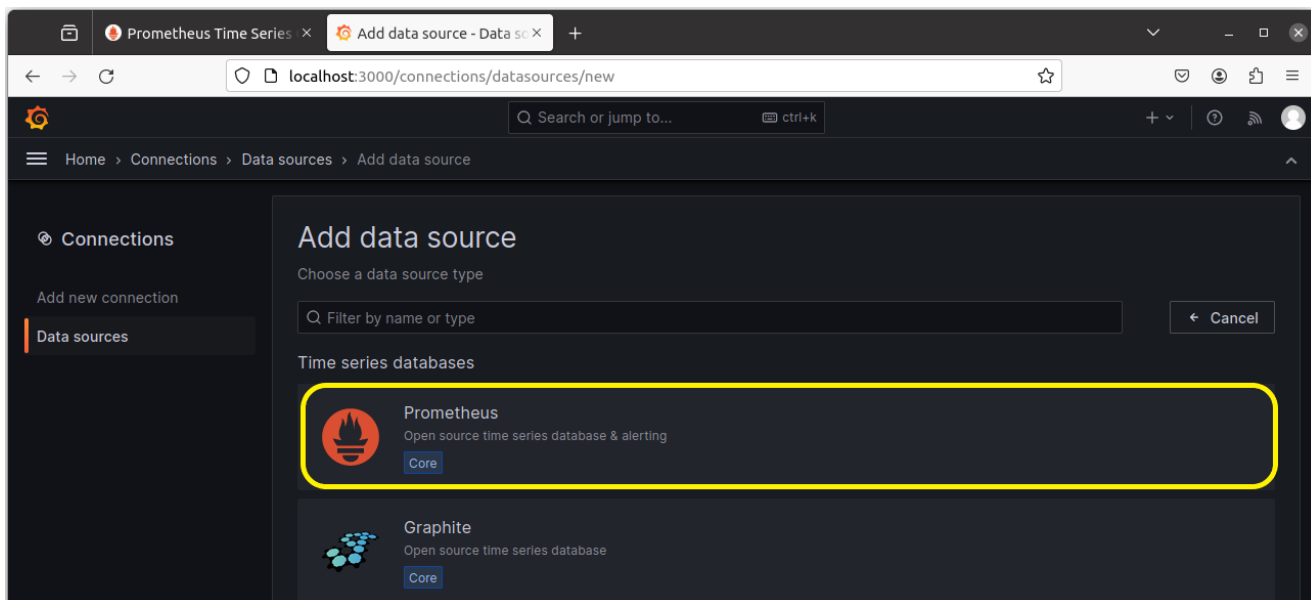


Рисунок 10.4 – Выбор сервера «Prometheus»

В следующем окне, задайте URL сервера «Prometheus» (например, <http://0.0.0.0:9090/>).



При запуске системы «Grafana» в Docker необходимо указывать реальный IP-адрес.

При необходимости задаются прочие параметры.

По окончании настройки, нажать кнопку «Save and test», расположенную внизу окна, чтобы сохранить изменения и проверить доступ к источнику данных.

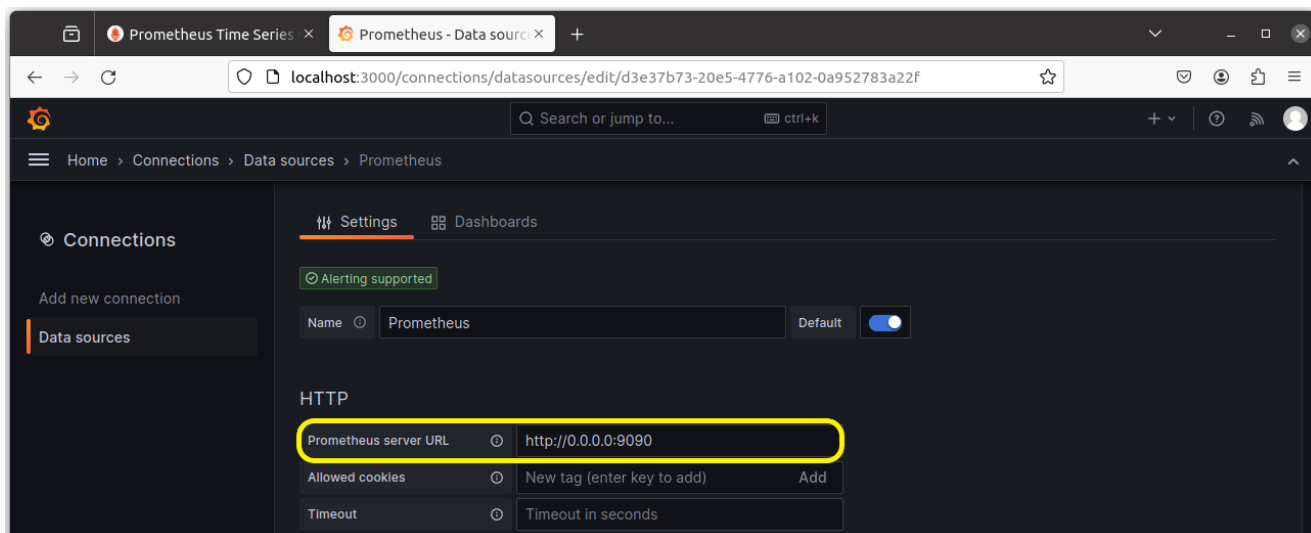


Рисунок 10.5 – Задание URL сервера «Prometheus»

При успешной проверке источник данных будет сохранен в системе и можно будет перейти к созданию дашбордов.

10.3. Создание дашбордов

Описание дашбордов и действий с ними приведено в официальной документации на сайте разработчика по адресу: <https://grafana.com/docs/grafana/latest/dashboards/>

10.4. Импорт дашбордов

На дистрибутивном диске содержится архив `jatoba_dashboard.tar`, который требуется распаковать в доступный каталог. В архиве хранится файл «Мониторинг основных показателей работы сервера БД-1703516982482.json» с подготовленными дашбордами.

Импорт дашбордов выполняется следующими шагами:

- Открыть меню «Dashboards», расположенном слева вверху.
- Нажать кнопку «New» и выбрать опцию «Import» в выпадающем меню.

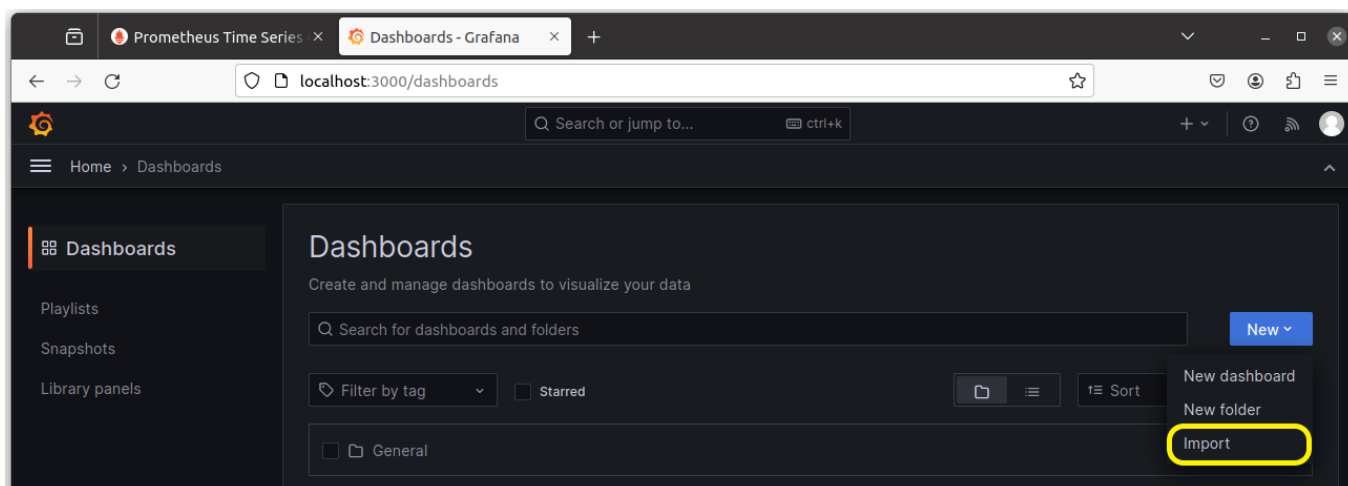


Рисунок 10.6 – Выбор опции импорт

Выполнить загрузку из файла «Мониторинг основных показателей работы сервера БД-1703516982482.json» и сервер «Prometheus».

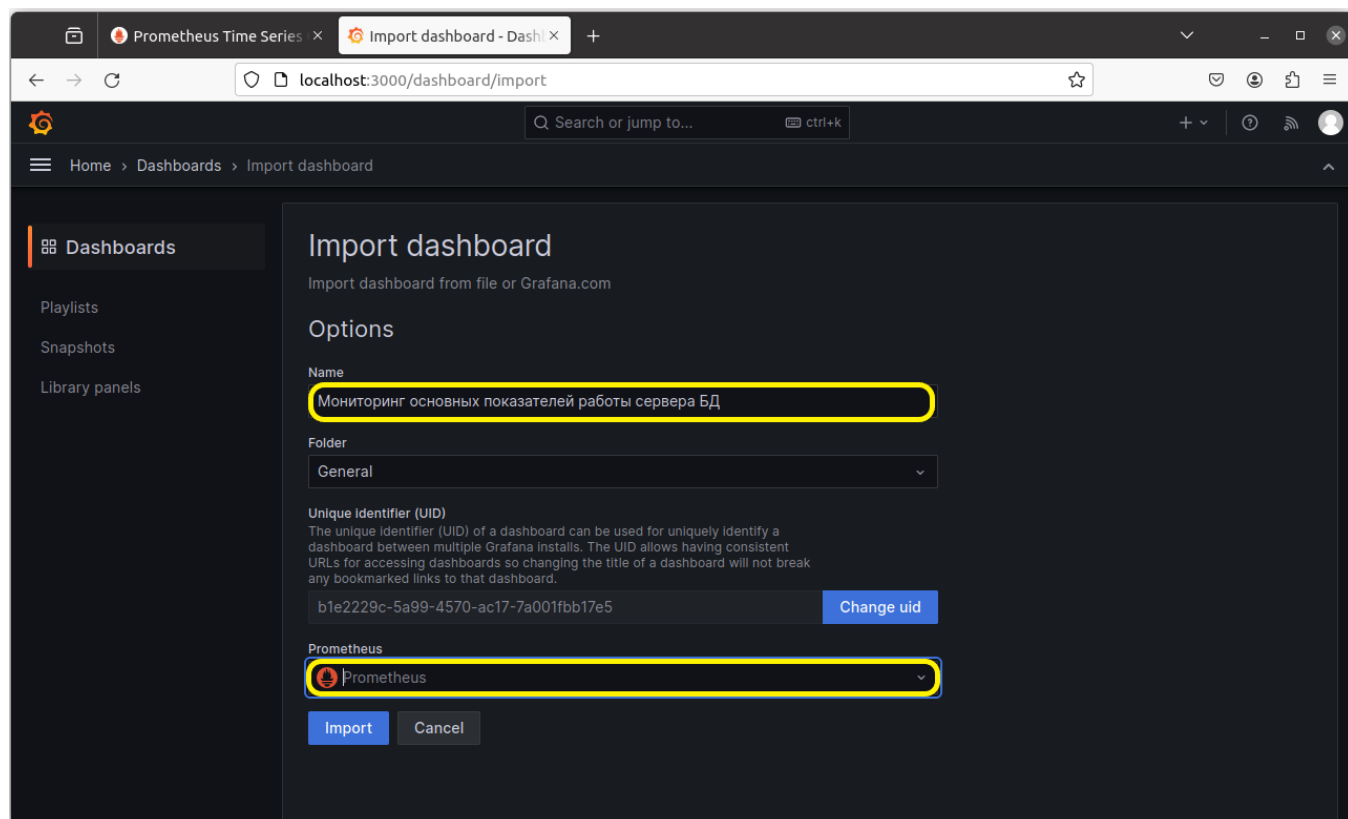


Рисунок 10.7 – Импорт файла дашбордов

В результате будет выведен группированный список подготовленных дашбордов.

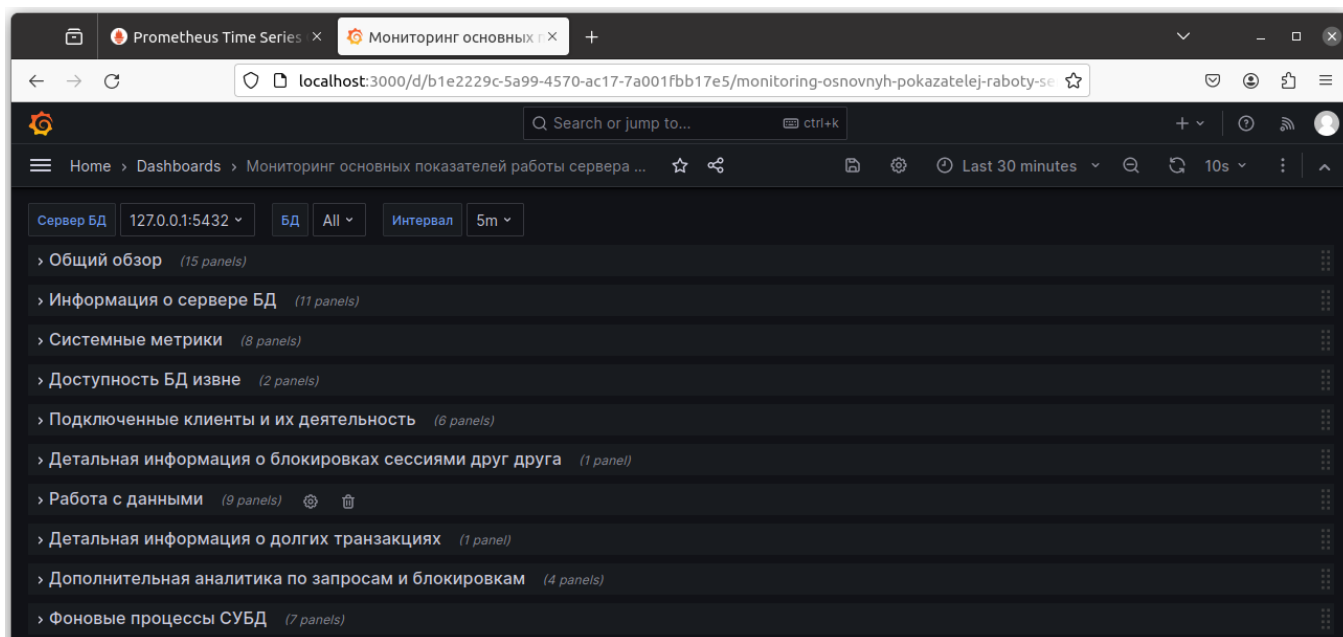


Рисунок 10.8 – Список дашбордов

На данном шаге формирование дашбордов закончено.

11. ДАШБОРДЫ

В результате установки экспортеров и импорта дашбордов получен готовый, преднастроенный механизм мониторинга СУБД. Перечень импортированных дашбордов приведен в таблице 11.1.

Таблица 11.1 – Перечень импортированных дашбордов

Раздел	Метрика
Общий обзор	
	Наименование хоста
	Длительность работы сервера
	Длительность работы СУБД
	Количество сессий
	Среднее время выполнения запросов
	Утилизация CPU
	Использование ОЗУ
	Утилизация I/O (производительность)
	Количество транзакций и запросов /сек (tps.qps)
	Дисковый ввод/вывод, байт/сек
	Количество дисковых iops/сек
	Сетевой ввод/вывод, байт/сек
	Размер БД, байт
	Доля чтения из кэша (% попадания в кэш)
	Возраст незамороженных транзакций
Информация о сервере БД	
	Hostname
	OS version (Версия ОС)
	Server uptime
	CPU cores
	Total memory (Размер оперативной памяти)
	Used memory
	Disk volume (Размер файловой системы)
	Free space (Свободное место на дисках)
	Network speed
	DB version
	DB uptime
Системные метрики	
	Утилизация CPU
	Загрузка CPU процессами (system load 1/5/15 min)
	Использование ОЗУ экземпляром СУБД, %
	Использование ОЗУ экземпляром СУБД, байт
	Дисковый ввод/вывод, байт/сек
	Сетевой ввод/вывод, байт/сек
	Количество дисковых iops/сек
	Утилизация I/O (произв-ть), %
Доступность БД извне	
	Среднее время выполнения запросов суммарно по всем БД, мс
№ изменения: _____ Подпись отв. лица: _____ Дата внесения изм: _____	

Раздел	Метрика
	Среднее время выполнения запросов по указанным БД, мс
Подключенные клиенты и их деятельность	
	Количество сессий
	Количество сессий по статусам
	Максимальная продолжительность запросов / сессий / ожиданий в сек.
	Количество заблокированных и ожидающих сессий
	Наибольшая длительность сессии в "idle in transaction", сек
	Суммарная продолжительность ожидания заблокированными сессиями, сек.
Детальная информация о блокировках сессиями друг друга	
	Блокирование сессиями друг друга
Работа с данными	
	Размеры БД, байт
	Изменение размеров БД, байт/сек
	Количество транзакций и запросов / сек (tps, qps)
	Количество транзакций и их откатов
	Работа со строками данных, tuples / сек.
	Блокировки / сек.
	Доля чтения из кэша (% попадания в кэш)
	Использование счетчика транзакций - age(datfrozenxid)
	Наибольшая длительность транзакции (без автовакуумов), сек.
Детальная информация о долгих транзакциях	
	Информация о 10-ти самых долгих транзакциях
Детальная информация о долгих транзакциях	
	Информация о 10-ти самых долгих транзакциях
Дополнительная аналитика по запросам и блокировкам	
	ТОР-10 запросов по частоте выполнения (calls)
	ТОР-10 запросов по времени выполнения (total_time), мс
	Среднее время ожидания назначения блокировки, сек.
	Назначенные блокировки в зависимости от режима
	Наибольшая длительность и кол-во автовакуумов, сек
	Интенсивность записи буферов на диск, буф./сек.
	Количество контрольных точек / сек.
	Длительность обработки контрольных точек, мс
	Запись во временные файлы
	Deadlocks и конфликты
	Объем WAL, байт

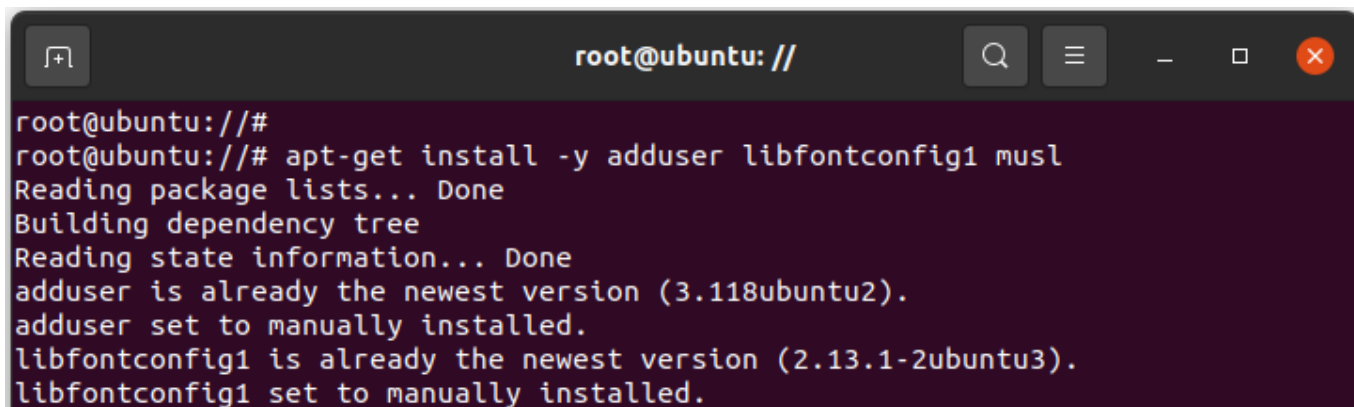
ПРИЛОЖЕНИЕ 1

Пример установки системы «Grafana»

Для установки системы «Grafana» требуется выполнить следующие шаги:

1. Установить пакет «musl»:

```
apt-get install -y adduser libfontconfig1 musl
```

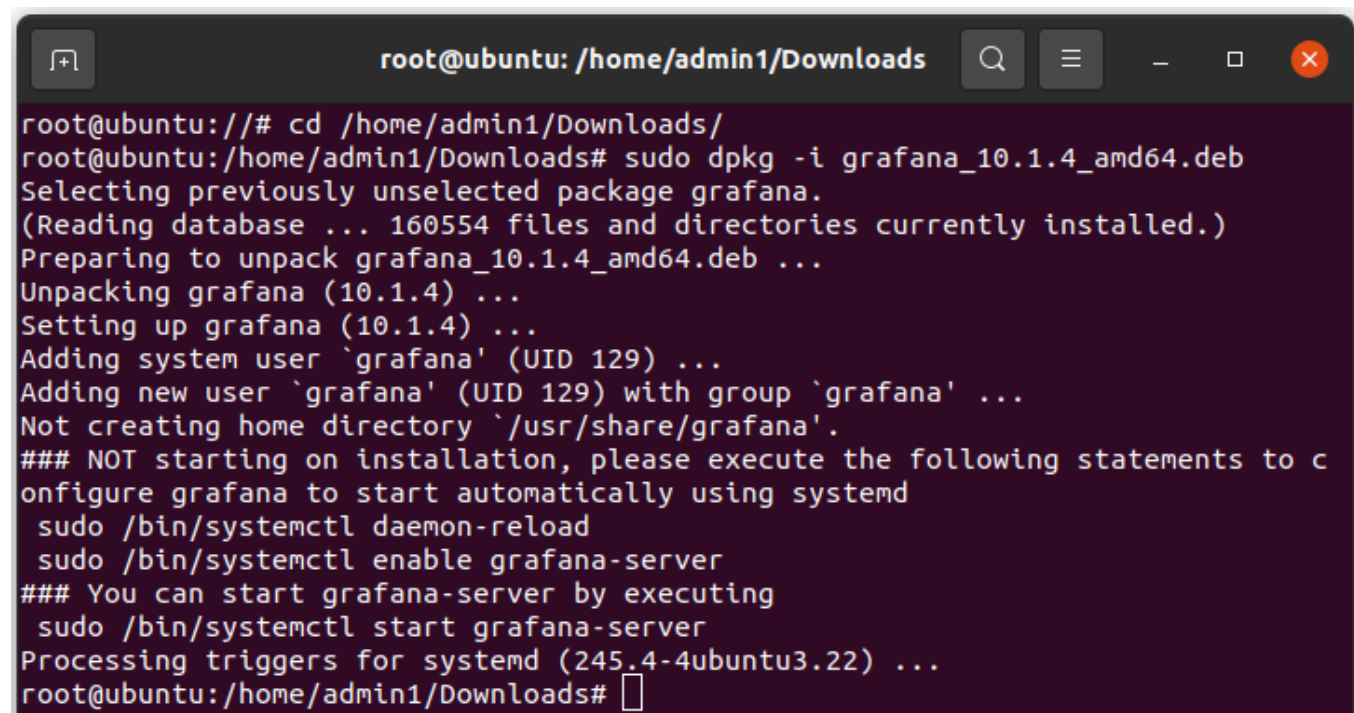


```
root@ubuntu: //
root@ubuntu: // # apt-get install -y adduser libfontconfig1 musl
Reading package lists... Done
Building dependency tree
Reading state information... Done
adduser is already the newest version (3.118ubuntu2).
adduser set to manually installed.
libfontconfig1 is already the newest version (2.13.1-2ubuntu3).
libfontconfig1 set to manually installed.
```

Рисунок 1.1 – Установка пакета «musl»

2. Установить систему «grafana» из пакета командой:

```
sudo dpkg -i grafana_10.1.4_amd64.deb
```



```
root@ubuntu: /home/admin1/Downloads
root@ubuntu: /home/admin1/Downloads # cd /home/admin1/Downloads/
root@ubuntu: /home/admin1/Downloads # sudo dpkg -i grafana_10.1.4_amd64.deb
Selecting previously unselected package grafana.
(Reading database ... 160554 files and directories currently installed.)
Preparing to unpack grafana_10.1.4_amd64.deb ...
Unpacking grafana (10.1.4) ...
Setting up grafana (10.1.4) ...
Adding system user `grafana' (UID 129) ...
Adding new user `grafana' (UID 129) with group `grafana' ...
Not creating home directory `/usr/share/grafana'.
### NOT starting on installation, please execute the following statements to c
onfigure grafana to start automatically using systemd
  sudo /bin/systemctl daemon-reload
  sudo /bin/systemctl enable grafana-server
### You can start grafana-server by executing
  sudo /bin/systemctl start grafana-server
Processing triggers for systemd (245.4-4ubuntu3.22) ...
root@ubuntu: /home/admin1/Downloads #
```

Рисунок 1.2 – Установка системы «Grafana»

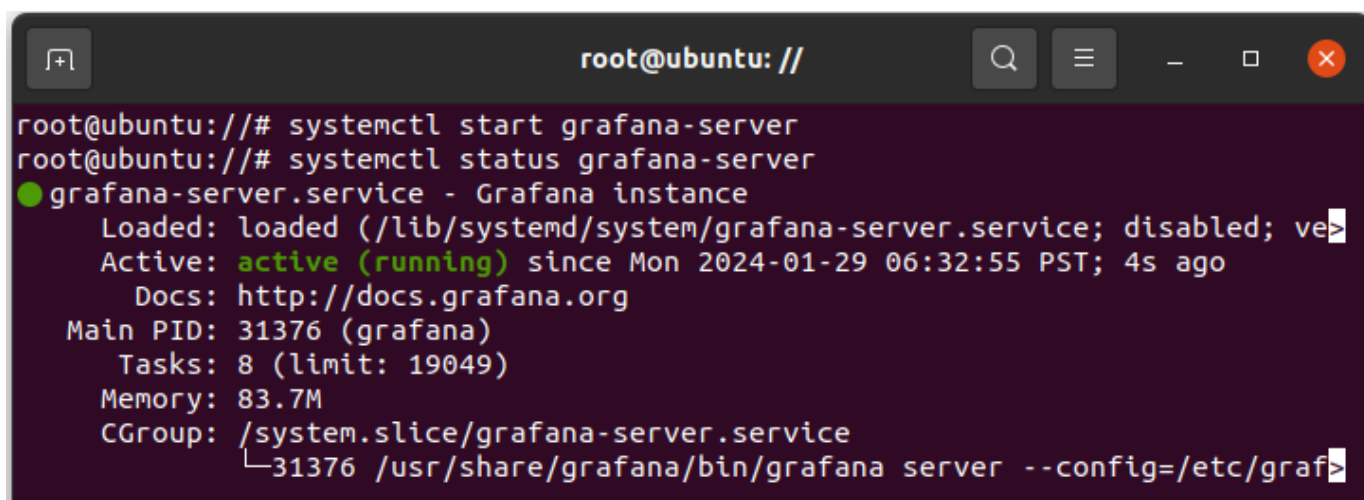
В процессе установки будет:

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

- создан пользователь «grafana»;
- служба «grafana-server» установит себя в автозагрузку.

3. Запустить службу и проверить ее статус:

```
systemctl start grafana-server  
systemctl status grafana-server
```



```
root@ubuntu: //  
root@ubuntu://# systemctl start grafana-server  
root@ubuntu://# systemctl status grafana-server  
● grafana-server.service - Grafana instance  
   Loaded: loaded (/lib/systemd/system/grafana-server.service; disabled; ve  
   Active: active (running) since Mon 2024-01-29 06:32:55 PST; 4s ago  
     Docs: http://docs.grafana.org  
   Main PID: 31376 (grafana)  
     Tasks: 8 (limit: 19049)  
    Memory: 83.7M  
    CGroup: /system.slice/grafana-server.service  
            └─31376 /usr/share/grafana/bin/grafana server --config=/etc/graf
```

Рисунок 1.3 – Запуск и проверка статуса службы «grafana-server»

4. Для того, чтобы убедиться в работоспособности системы «Grafana» работает, необходимо зайти на порт 3000 по адресу сервера, на котором установлена система.

Например, можно использовать любой текстовый браузер на сервере:

```
# lynx http://0.0.0.0:3000  
# w3m http://0.0.0.0:3000
```

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

@ - ГОСТ 34.302.2-91 (ИСО 8859/2-87) «Наборы 8 битных однобайтовых кодированных графических символов. латинский алфавит № 2» определяет символ как, «коммерческое эт» (англ. «commercial at»).

& - ГОСТ 34.302.2-91 (ИСО 8859/2-87) «Наборы 8 битных однобайтовых кодированных графических символов. латинский алфавит № 2» определяет символ как, «коммерческое и» (амперсанд) (англ. ampersand)

= - ГОСТ 34.302.2-91 (ИСО 8859/2-87) «Наборы 8 битных однобайтовых кодированных графических символов. латинский алфавит № 2» определяет символ как, «равно» (англ. equals SIGN).

? - ГОСТ 34.302.2-91 (ИСО 8859/2-87) «Наборы 8 битных однобайтовых кодированных графических символов. латинский алфавит № 2» определяет символ как, «вопросительный знак» (англ. question mark).

: - ГОСТ 34.302.2-91 (ИСО 8859/2-87) «Наборы 8 битных однобайтовых кодированных графических символов. латинский алфавит № 2» определяет символ как, «двоеточие» (англ. colon).

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

SQL	–	Structured Query Language
БД	–	База данных
ОС	–	Операционная система
СУБД	–	Система управления базами данных

Лист регистрации изменений

Дата внесения изм: