

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 75

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 – 1.8.0;
- Компонент «postgres_exporter». Версия компонента – 0.18.1;
- Компонент «sql_exporter». Версия компонента – 0.18.6.
- Система «Prometheus». Версия системы – 3.5.0.
- 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 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.....	6
3. Установка и настройка целевых СУБД.....	8
3.1. Установка СУБД.....	8
3.2. Настройка конфигурационных файлов.....	8
3.3. Установка расширения «pg_stat_statements».....	8
4. Установка экспортера «jatoba*_node_exporter».....	11
5. Установка экспортера «jatoba*_postgres_exporter».....	15
5.1. Установка утилиты и службы «jatoba*_postgres_exporter».....	15
5.2. Создание пользователя СУБД «postgres_exporter».....	16
5.3. Настройка переменных окружения.....	16
5.4. Запуск утилиты «postgres_exporter».....	18
6. Установка экспортера «jatoba*_sql_exporter».....	21
6.1. Установка утилиты и службы «sql_exporter».....	21
6.2. Настройка переменных окружения.....	22
6.3. Создание пользователя СУБД «sql_exporter».....	23
6.4. Настройка параметров экспортера и подключения к БД в файле «sql_exporter.yml».....	23
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».....	39
8.1. Установка утилиты и службы «alertmanager».....	39
8.2. Настройка параметров утилиты.....	40
8.2.1. SMTP.....	41
8.2.2. +-Telegram.....	42
8.2.3. Zulip.....	44
8.3. Запуск утилиты «alertmanager».....	49
9. Подключение к JDS.....	51
9.1. Настройка SSH-соединения.....	52
9.2. Конфигурирование JDS.....	53

9.3. Настройка связки системы «Prometheus» и утилиты «Alertmanager»	55
10. Настройка экспортёров для компонента ja_Hipe_Cluster	57
10.1. Параметры стенда	57
10.2. Конфигурирование системы «Prometheus»	60
11. Обновление компонентов экспортеров	63
11.1. Обновление компонента sql_exporter	63
11.1.1. Установка новой версии компонента sql_exporter.....	63
11.1.2. Настройка мониторинга кластера Citus после обновления	64
11.2. Обновление компонентов node_exporter, postgres_exporter и alertmanager	69
12. Обновление системы «Prometeus».....	71
Термины и определения	73
Перечень сокращений.....	74

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 сервер «Prometheus» 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

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.

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

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.

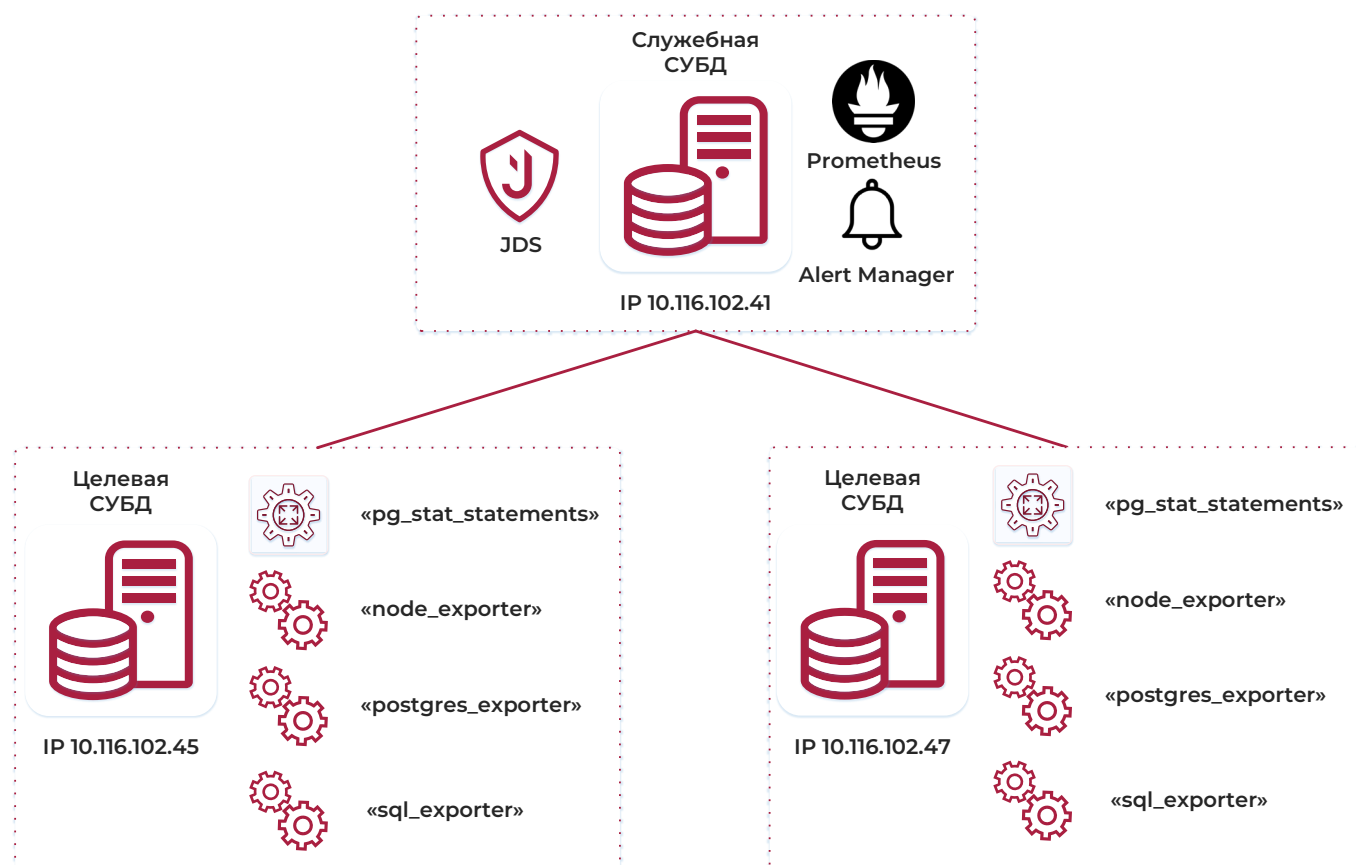


Рисунок 2.1 – Схема стенда

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 = '*'
```

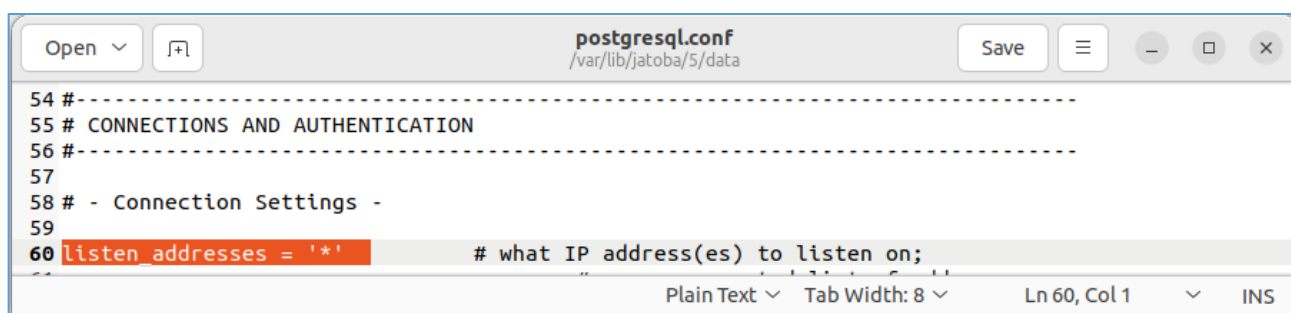


Рисунок 3.1 - Конфигурационный файл «postgresql.conf»

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

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

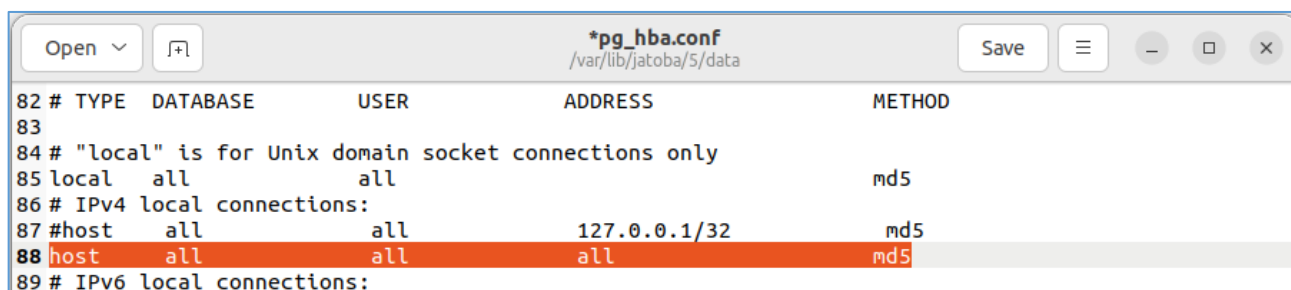


Рисунок 3.2 – Конфигурационный файл «pg_hba.conf»

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 случае, если утилитой `sql_exporter` опрашиваются метрики компонента SQL Firewall, в параметре `shared_preload_libraries` перед значением `sql_firewall` должно быть в обязательном порядке через запятую установлено значение `pg_stat_statements`:

```
shared_preload_libraries = 'pg_stat_statements,  
sql_firewall'
```

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

```
compute_query_id = on
```

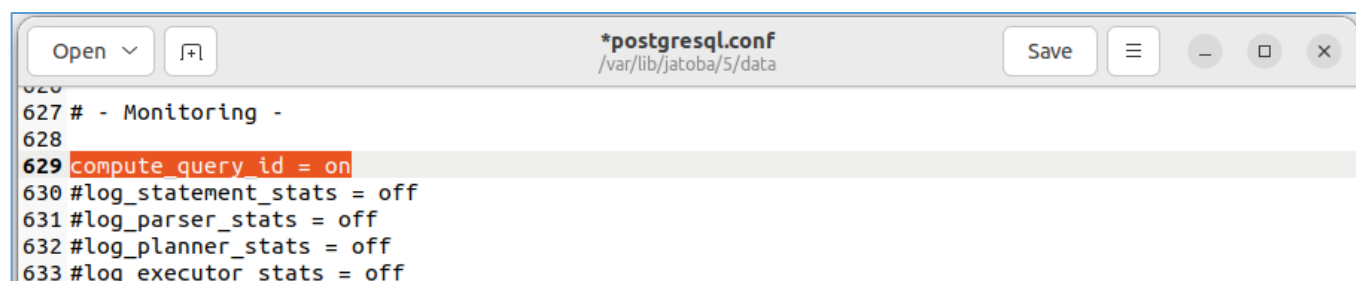


Рисунок 3.4 – Строка параметра «compute_query_id» в конфигурационном файле СУБД «postgresql.conf»

- 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;
```

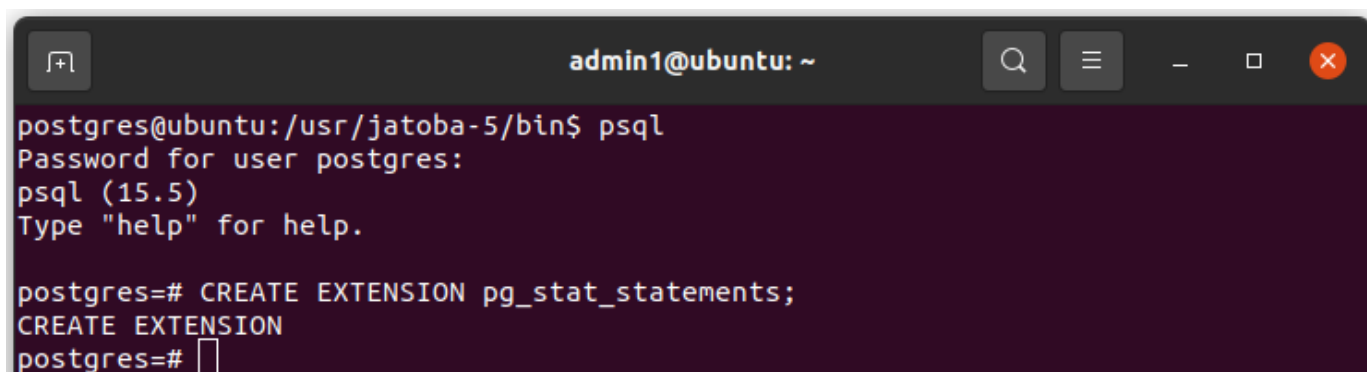


Рисунок 3.6 – Создание расширения

4. УСТАНОВКА ЭКСПОРТЕРА «JATOBA*_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
```

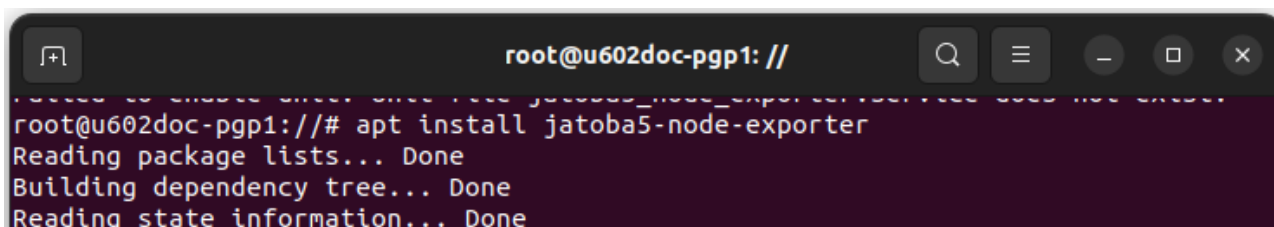


Рисунок 4.1 – Установка пакета «jatoba*_node_exporter»

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

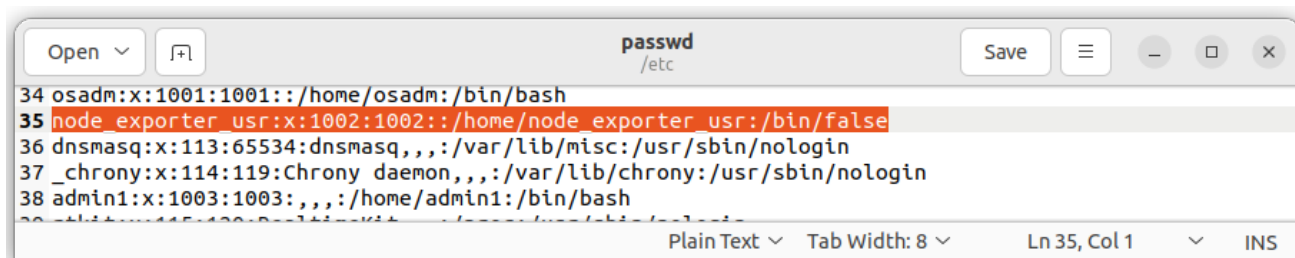


Рисунок 4.2 -пользователя «node_exporter_usr»

У данного пользователя нет интерактивной оболочки для входа.

Автоматически будет создан файл конфигурации сервиса по адресу:

```
/usr/lib/systemd/system/jatoba<ver>_node_exporter.service
```

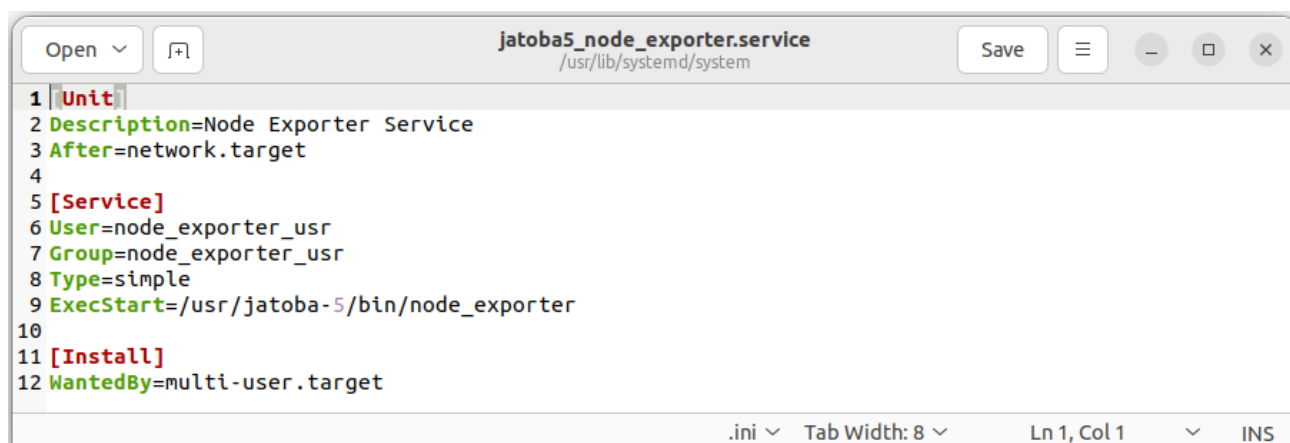


Рисунок 4.3 – Содержание конфигурационного файла

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
```

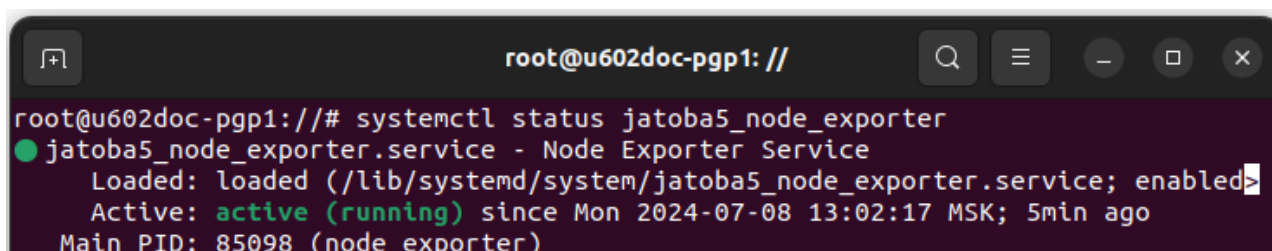


Рисунок 4.4 - Проверка сервиса «jatoba*_node_exporter»

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

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

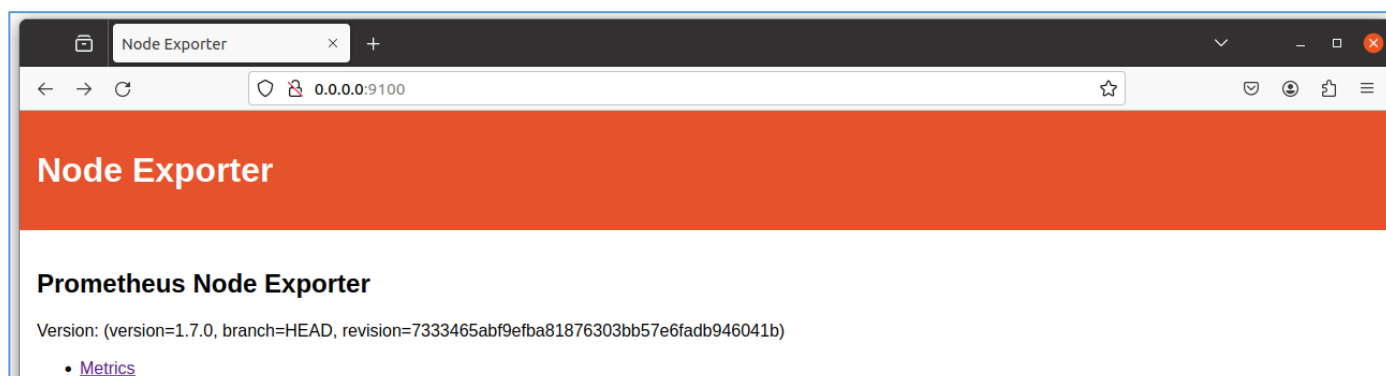


Рисунок 4.5 – Веб-интерфейс утилиты «node_exporter»

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
```

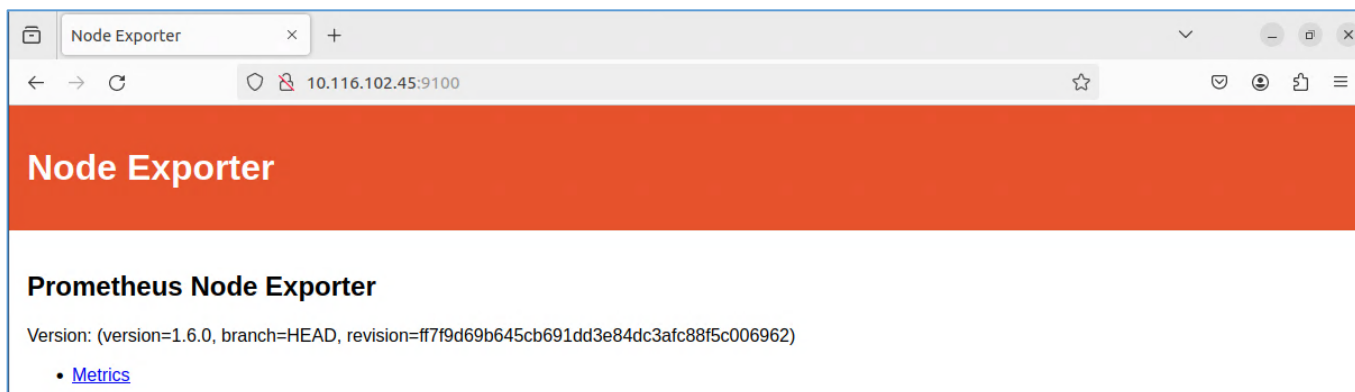


Рисунок 4.6 – Веб-интерфейс утилиты «node_exporter» на целевой СУБД u602doc-pgrp01 IP - 10.116.102.45

- 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
```

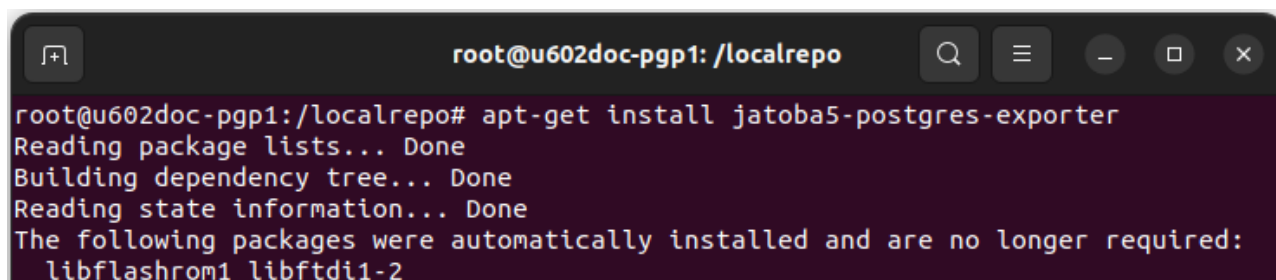


Рисунок 5.1 – Установка пакета jatoba*-postgres-exporter

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
```

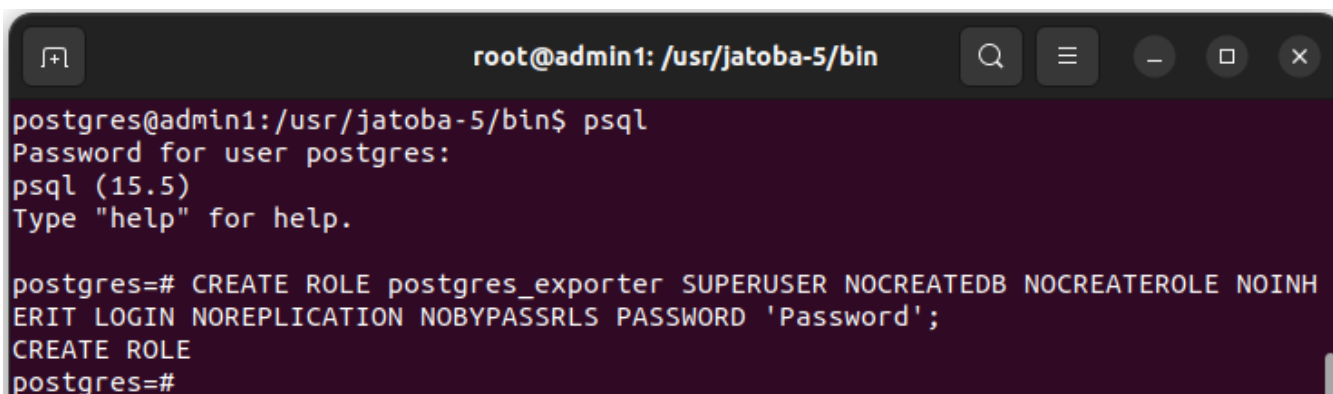
- пользователь ОС «postgres_exporter_usr», от которого будет производиться запуск сервиса.

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 (15.5)' prompt appears. The user enters the SQL command 'CREATE ROLE postgres_exporter SUPERUSER NOCREATEDB NOCREATEROLE NOINHERIT LOGIN NOREPLICATION NOBYPASSRLS PASSWORD 'Password';'. The command is executed successfully, and the prompt returns to '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-<ver>/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_параметра]
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

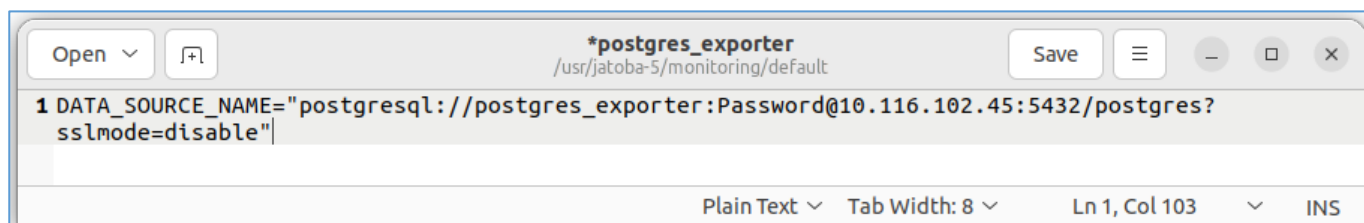


Рисунок 5.3 – Содержание файла «postgres_exporter.default» на целевой СУБД u602doc-pgr01 IP - 10.116.102.45

– u602doc-ldar01 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"
```

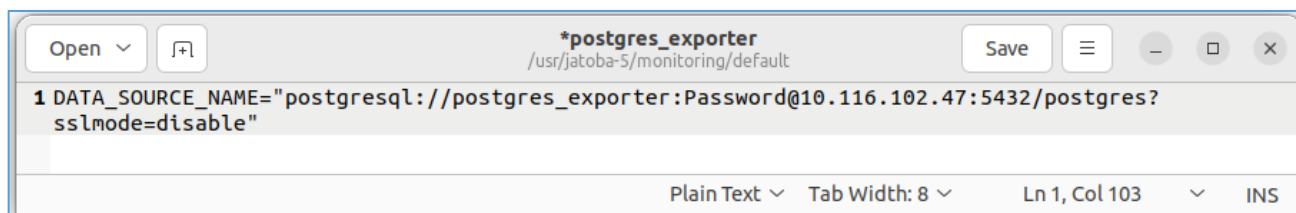


Рисунок 5.4– Содержание файла «postgres_exporter.default» на целевой СУБД u602doc-ldap01 IP-10.116.102.47

5.4. Запуск утилиты «postgres_exporter»

Обновить конфигурацию system командой:

```
# 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
```

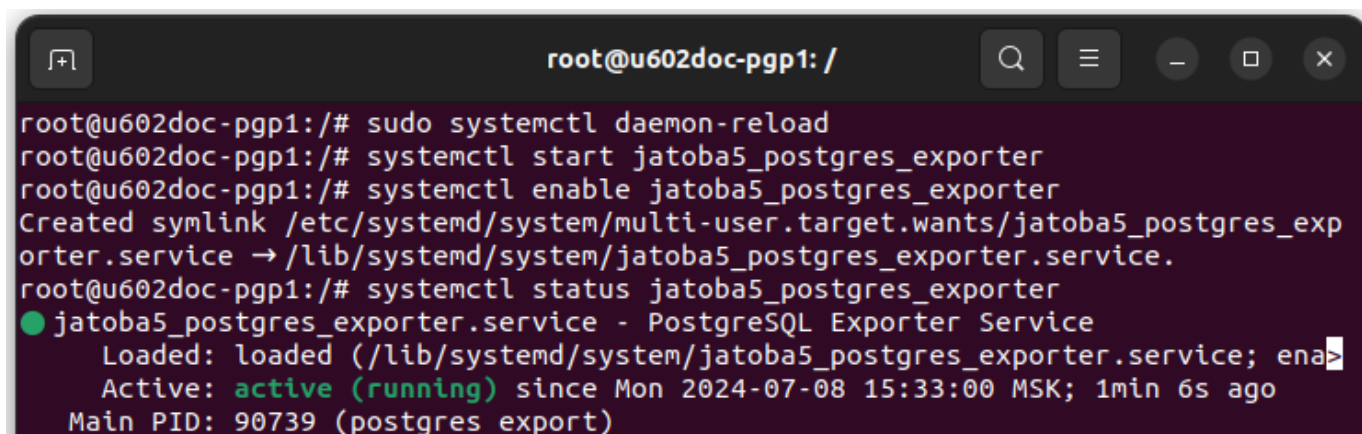


Рисунок 5.5 – Запуск и вывод статуса службы «postgres_exporter»

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

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

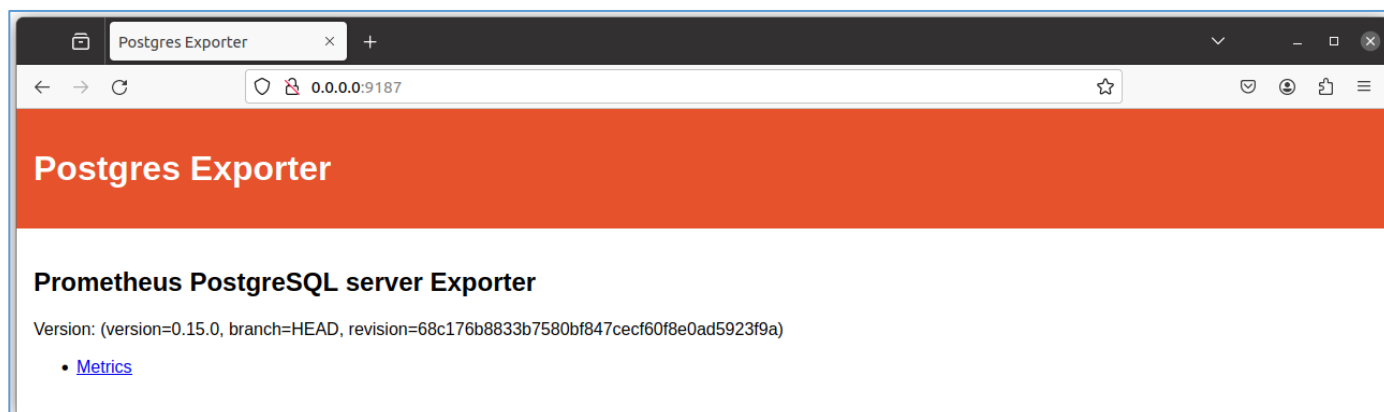


Рисунок 5.6 – Веб-интерфейс «postgres_exporter»

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

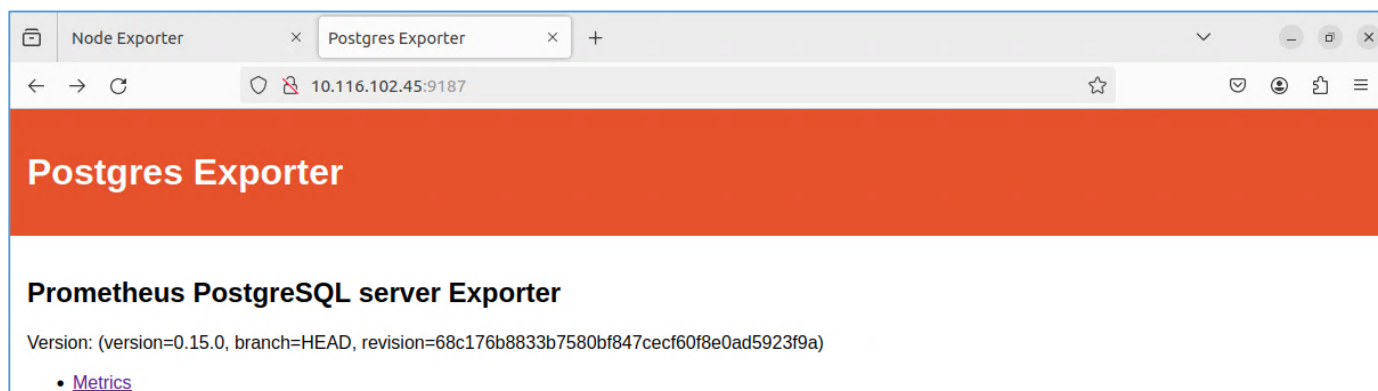


Рисунок 5.7 – Веб-интерфейс «postgres_exporter» на целевой СУБД u602doc-pgp01 IP - 10.116.102.45

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

http://10.116.102.47:9187

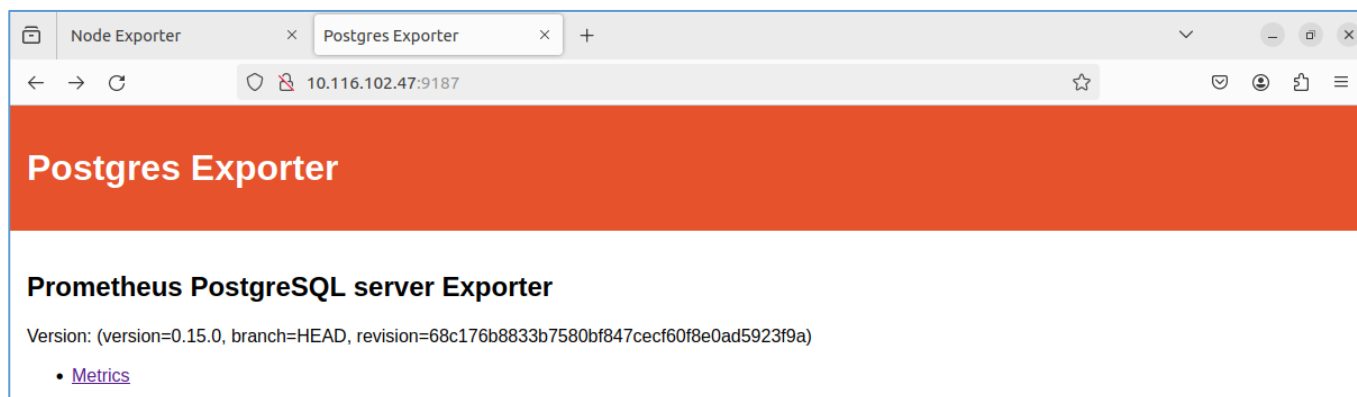


Рисунок 5.8 – Веб-интерфейс «postgres_exporter» на целевой СУБД u602doc-ldap01 IP-10.116.102.47

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.

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), а также для метрик компонента SQL_Firewall.

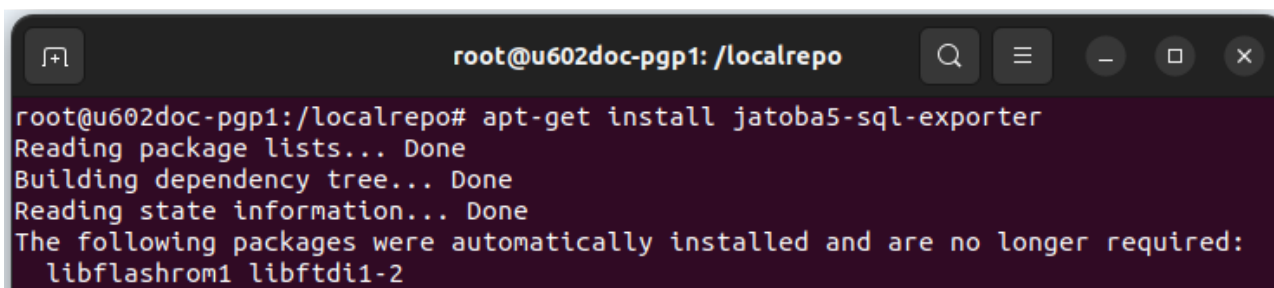
Это агент, также написанный на языке Golang, который подключается к заданному источнику данных (БД) и забирает с него метрики по pull-запросу сервера «Prometheus».

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 указатель 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
```



```
root@u602doc-pgp1: /localrepo
root@u602doc-pgp1:/localrepo# apt-get install jatoba5-sql-exporter
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 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-<ver>/monitoring/default/sql_exporter
```

- конфигурационный файл для сбора данных узлов кластера Citus по адресу:

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

- конфигурационный файл для сбора данных компонента SQL_Firewall по адресу:

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

- 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-<ver>/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.

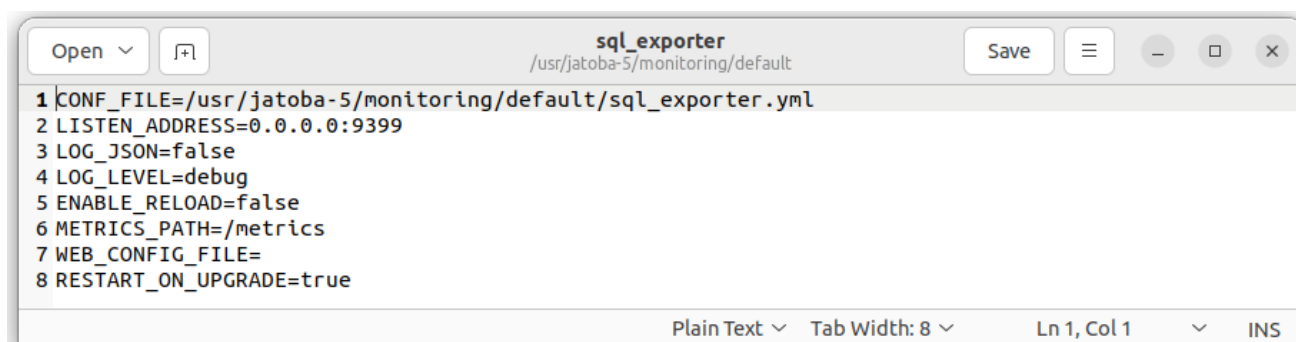


Рисунок 6.2 – Содержание файла переменных окружения «sql_exporter»

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';
```

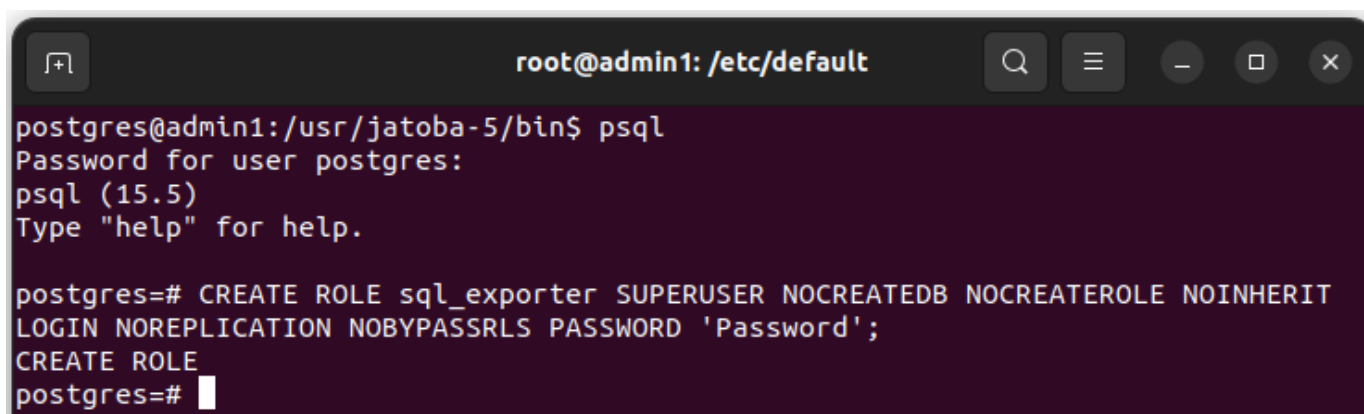


Рисунок 6.3 – Создание роли «sql_exporter»



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-<ver>/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-pgp01 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.

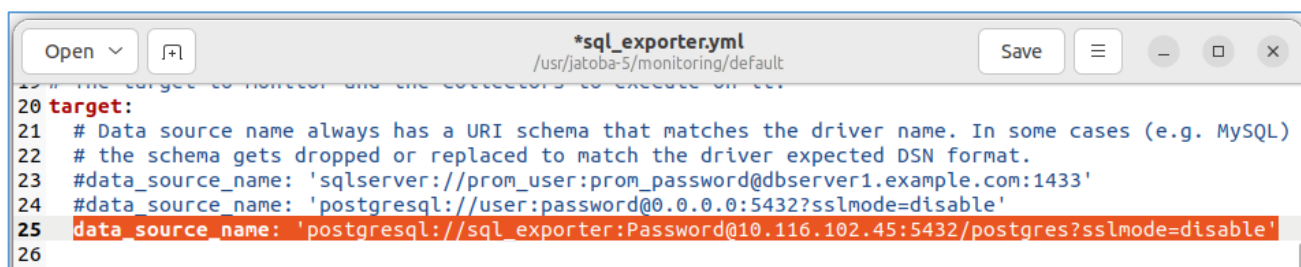


Рисунок 6.4 – Содержание файла «sql_exporter.yml», строка «data_source_name» на целевой СУБД u602doc-pgp01 IP - 10.116.102.45

– 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'
```

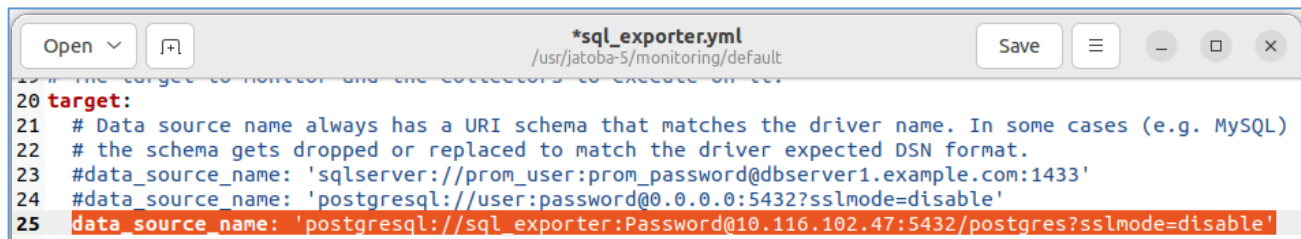


Рисунок 6.5 – Содержание файла «sql_exporter.yml», строка «data_source_name» на целевой СУБД u602doc-ldap01 IP-10.116.102.47



В случае необходимости мониторинга компонента SQL Firewall в конфигурационном файле `sql_exporter.yml` в поле «collectors» через запятую необходимо добавить значение `sql_firewall` (см. рисунок 6.6).

```

GNU nano 6.2 sql_exporter.yml *
max_connection_lifetime: 5m

# The target to monitor and the collectors to execute on it.
target:
  # Data source name always has a URI schema that matches the driver name. In s>
  # the schema gets dropped or replaced to match the driver expected DSN format.
  #data_source_name: 'sqlserver://prom_user:prom_password@dbserver1.example.com>
  #data_source_name: 'postgresql://user:password@0.0.0.0:5432?sslmode=disable'
  data_source_name: 'postgresql://sql_exporter:12345678@127.0.0.1:5432/postgres>

  # Collectors (referenced by name) to execute on the target.
  # Glob patterns are supported (see <https://pkg.go.dev/path/filepath#Match> f>
  collectors: [postgres,sql_firewall]

# Collector files specifies a list of globs. One collector definition is read f>
# Glob patterns are supported (see <https://pkg.go.dev/path/filepath#Match> for>
collector_files:
  - "*.collector.yml"

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line
  
```

Рисунок 6.6 – Добавление мониторинга компонента SQL Firewall в конфигурационном файле `sql_exporter.yml`

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 jatoba<ver>_sql_exporter
# systemctl enable jatoba<ver>_sql_exporter
# systemctl status jatoba<ver>_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 6.7 – 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
```

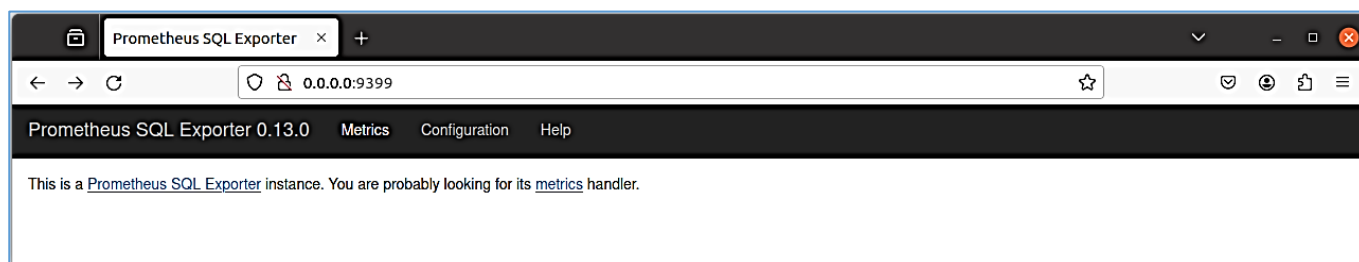


Рисунок 6.8 – Веб-интерфейс «sql_exporter»

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
```

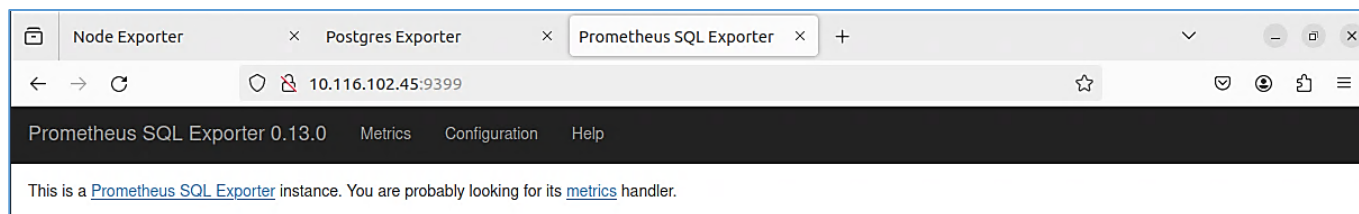


Рисунок 6.9 – Веб-интерфейс «sql_exporter» на целевой СУБД u602doc-pgp01 IP - 10.116.102.45

– 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

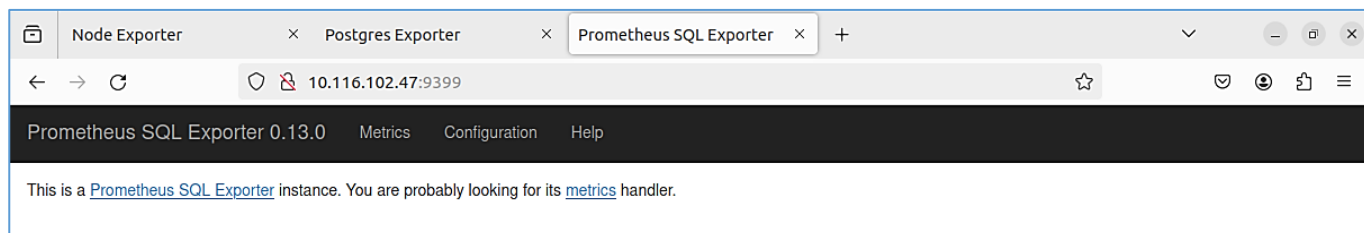


Рисунок 6.10 – Веб-интерфейс «sql_exporter» на целевой СУБД u602doc-ldap01 IP-10.116.102.47

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:

```
./jatoba<ver>_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.

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.

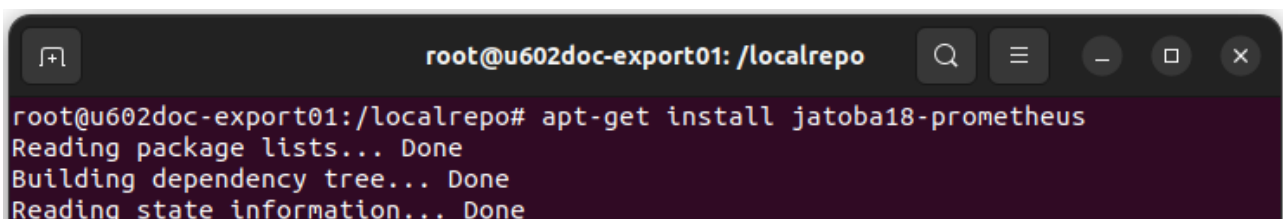
Документация по использованию системы находится на официальном сайте разработчика по адресу: <https://prometheus.io>.

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-export01: /localrepo
root@u602doc-export01:/localrepo# apt-get install jatoba18-prometheus
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

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

В результате установки пакета будет создан:

- файл переменных окружения сервиса по адресу:

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

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

- файл сервиса по адресу

```
usr/lib/system/system/jatoba<ver>_prometheus.service
```

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

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

- база данных по адресу:

```
/opt/prometheus
```

- служебные директории веб-консоли по адресу:

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

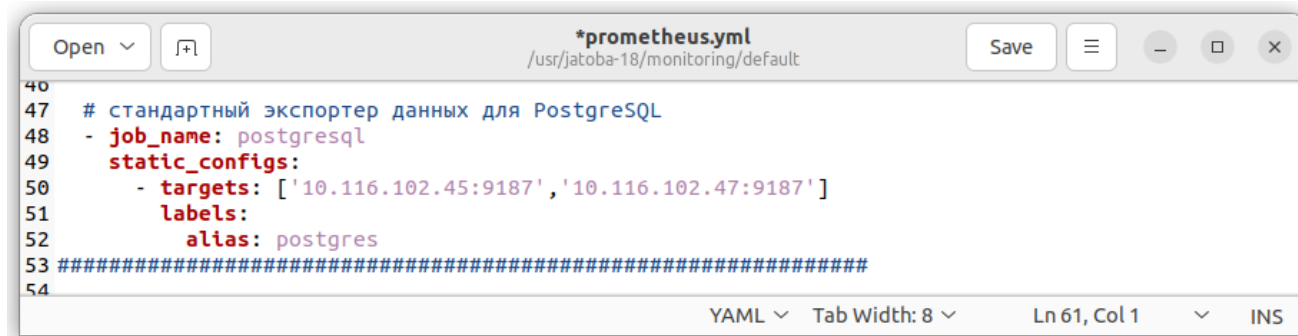


Рисунок 7.2 - Стандартный экспортер данных для PostgreSQL

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
```

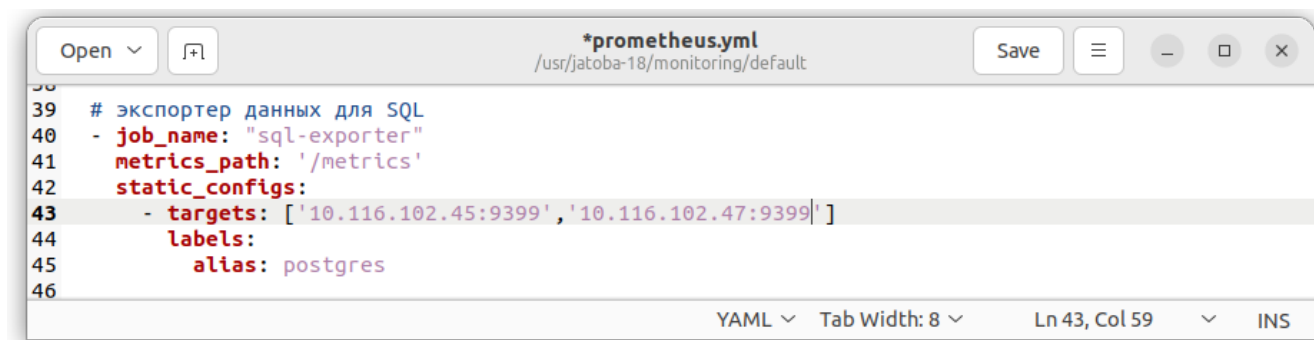


Рисунок 7.4 - «sql-exporter» с двумя целями

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/jatoba5_prometheus.service
# gedit usr/lib/systemd/system/jatoba18_prometheus.service
```

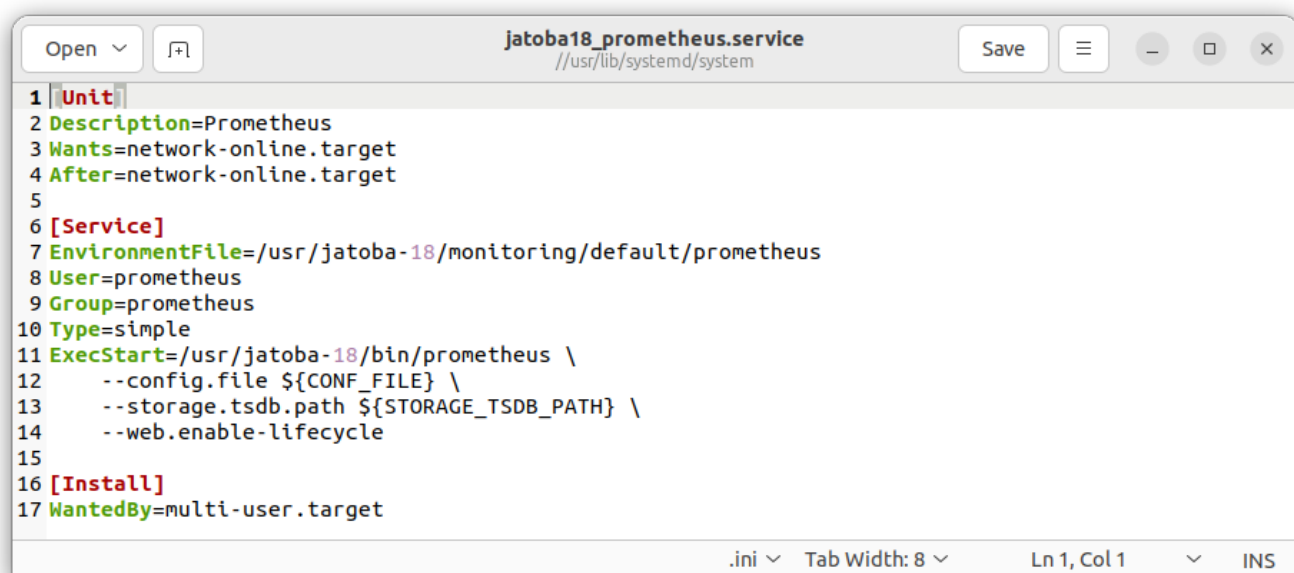


The screenshot shows a text editor window titled 'jatoba5_prometheus.service' with the path '/usr/lib/systemd/system'. The file content is as follows:

```
1 [Unit]
2 Description=Prometheus
3 Wants=network-online.target
4 After=network-online.target
5
6 [Service]
7 EnvironmentFile=/usr/jatoba-5/monitoring/default/prometheus
8 User=prometheus
9 Group=prometheus
10 Type=simple
11 ExecStart=/usr/jatoba-5/bin/prometheus \
12     --config.file ${CONF_FILE} \
13     --web.enable-lifecycle \
14     --storage.tsdb.path ${STORAGE_TSDB_PATH} \
15     --web.console.templates=${WEB_CONSOLE_TEMPLATES} \
16     --web.console.libraries=${WEB_CONSOLE_LIBRARIES}
17
18 [Install]
19 WantedBy=multi-user.target
```

The status bar at the bottom indicates '.ini', 'Tab Width: 8', 'Ln 13, Col 5', and 'INS'.

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



The screenshot shows a text editor window titled 'jatoba18_prometheus.service' with the path '//usr/lib/systemd/system'. The file content is as follows:

```
1 [Unit]
2 Description=Prometheus
3 Wants=network-online.target
4 After=network-online.target
5
6 [Service]
7 EnvironmentFile=/usr/jatoba-18/monitoring/default/prometheus
8 User=prometheus
9 Group=prometheus
10 Type=simple
11 ExecStart=/usr/jatoba-18/bin/prometheus \
12     --config.file ${CONF_FILE} \
13     --storage.tsdb.path ${STORAGE_TSDB_PATH} \
14     --web.enable-lifecycle
15
16 [Install]
17 WantedBy=multi-user.target
```

The status bar at the bottom indicates '.ini', 'Tab Width: 8', 'Ln 1, Col 1', and 'INS'.

Рисунок 7.9 – Содержание файла сервиса «jatoba18_prometheus.service»

Обновить конфигурацию systemd:

```
# systemctl daemon-reload
```

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

```
# systemctl start jatoba<ver>_prometheus
# systemctl enable jatoba<ver>_prometheus
# systemctl status jatoba<ver>_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 7.10 Установка и запуск службы системы «Prometheus»

```
root@u602doc-export01://# systemctl daemon-reload
root@u602doc-export01://# systemctl start jatoba18_prometheus
root@u602doc-export01://# systemctl enable jatoba18_prometheus
Created symlink /etc/systemd/system/multi-user.target.wants/jatoba18_prome
theus.service → /lib/systemd/system/jatoba18_prometheus.service.
root@u602doc-export01://# systemctl status jatoba18_prometheus
● jatoba18_prometheus.service - Prometheus
   Loaded: loaded (/lib/systemd/system/jatoba18_prometheus.service; ena>
   Active: active (running) since Mon 2025-12-22 13:58:48 MSK; 23s ago
   Main PID: 4962 (prometheus)
     Tasks: 8 (limit: 3468)
    Memory: 21.3M
       CPU: 104ms
    CGroup: /system.slice/jatoba18_prometheus.service
           └─4962 /usr/jatoba-18/bin/prometheus --config.file usr/jatob>
```

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

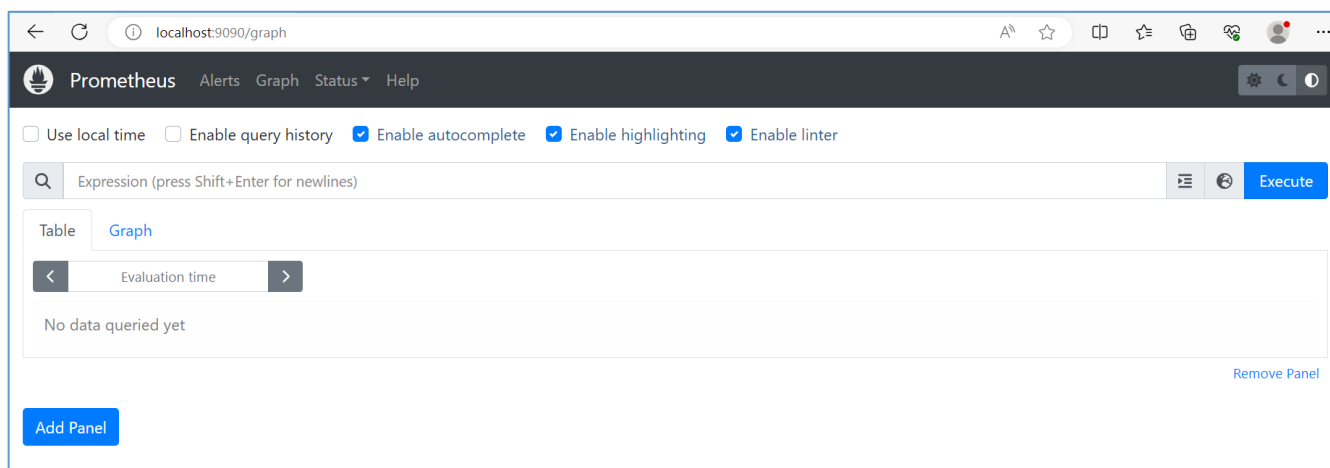


Рисунок 7.11 – Веб интерфейс системы «Prometheus»

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/>

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 web interface at 0.0.0.0:9090/targets?search=. The 'Targets' section displays three scrape pools, each with a table of targets. All targets are in an 'UP' state.

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 7.12 – 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.

The screenshot shows the Prometheus web interface at the 'Targets' page. The browser address bar shows '10.116.102.41:9090/targets?search='. The page has a navigation bar with 'Prometheus', 'Alerts', 'Graph', 'Status', and 'Help'. Below the navigation bar, the 'Targets' section is active. It includes a search bar, filter buttons for 'All', 'Unhealthy', and 'Collapse All', and a legend for 'Unknown', 'Unhealthy', and 'Healthy' states. Two scrape pools are listed: 'node-exporter (2/2 up)' and 'postgres-exporter (2/2 up)'. Each pool has a table of endpoints with columns for Endpoint, State, Labels, Last Scrape, Scrape Duration, and Error.

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	

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 7.13 – Страница целей системы «Prometheus»

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
```

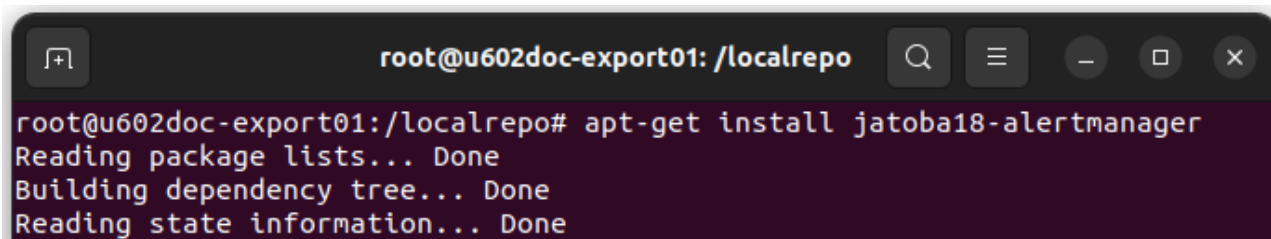


Рисунок 8.1 – Установка пакета «jatoba*-alertmanager»

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

Утилита «AlertManager» отправляет уведомления через каналы:

- SMTP;
- Telegram;
- Zulip.

Настройка каждого из каналов выполняется файле в конфигурационном файле «alertmanager.yml». После установки пакета в конфигурационном файле будут установлены параметры по умолчанию.



Рисунок 8.2 – Параметры по умолчанию

8.2.1. SMTP

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

Редактирование конфигурационного файла выполняется командой:

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

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 }}'
```

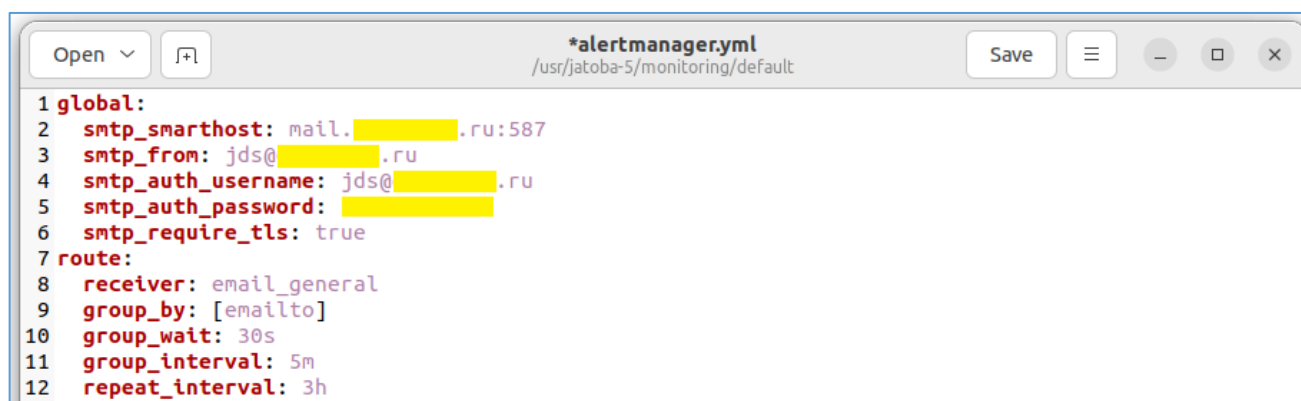


Рисунок 8.3 - Конфигурационный файл «alertmanager.yml»

8.2.2. +-Telegram

Настройка получения уведомлений (предупреждений) от утилиты «AlertManager» в Telegram состоит из нескольких этапов:

- Создать в Telegram бота;
- Создание канала в Telegram;
- открытие канала передачи предупреждений в утилите «AlertManager».

Создание в Telegram бота

Бот должен создаваться через @BotFather официальный сервис Telegram для создания, настройки и управления чат-ботами.

Сервис находится через поиск в Telegram.

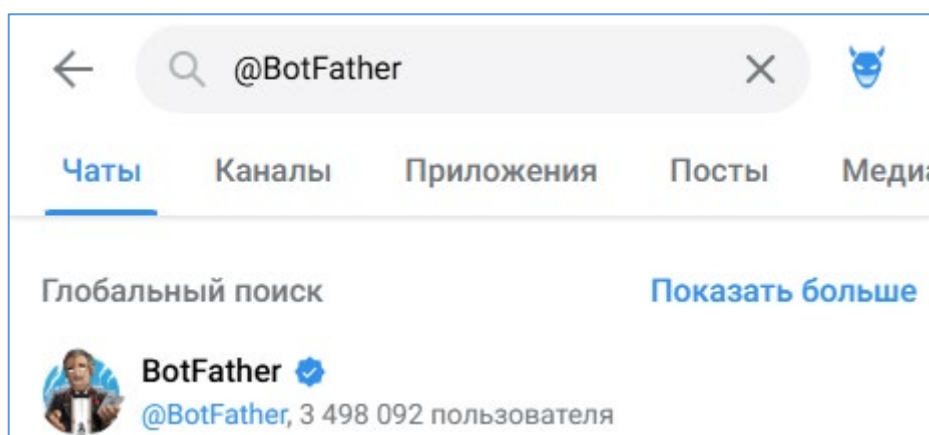
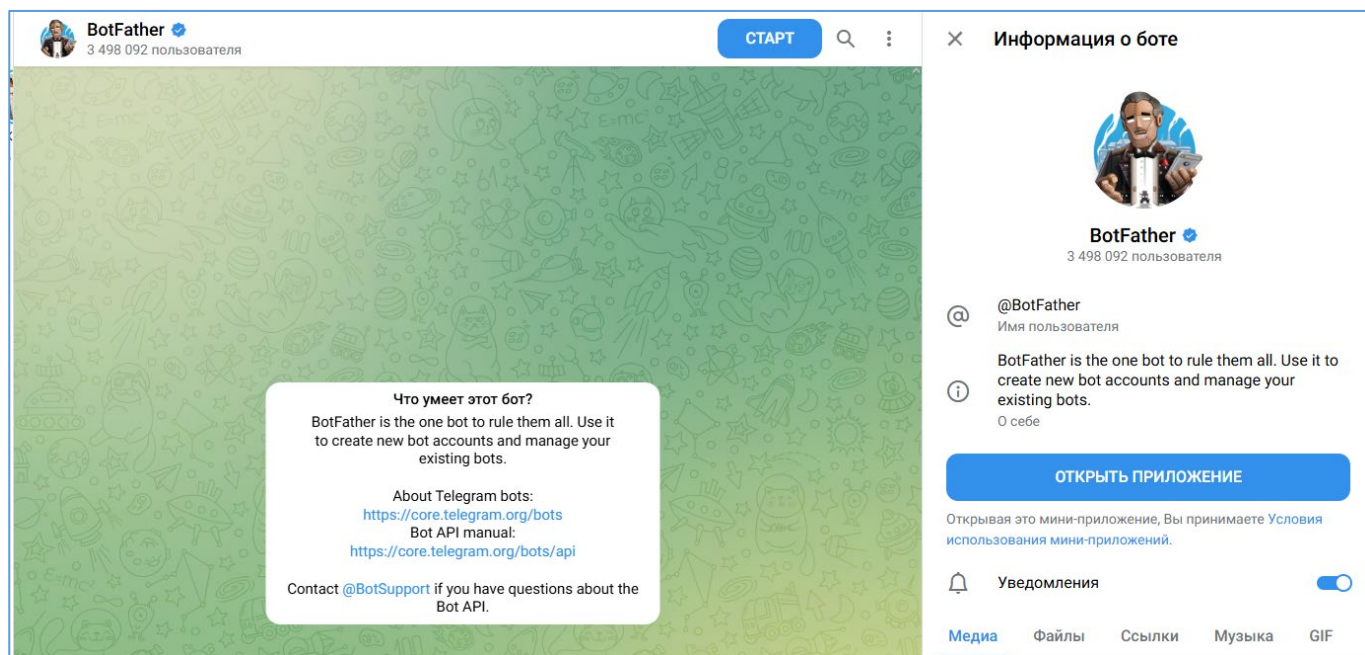


Рисунок 8.4 – Поиск сервиса @BotFather

Нажать кнопку «Старт».



Выполните команду создания бота и следуем инструкциям, задаем имя JDS alerts, а username – jds_alertmanager_bot.

Выполнить команду создания бота:

```
/newbot - create a new bot
```

Задайте уникальное имя бота:

```
jds_alertmanager_bot
my_jds_alertmanager_bot
```

фыва

Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris_bot.

ыав

Done! Congratulations on your new bot. You will find it at t.me/my_jds_alertmanager_bot. You can now add a description, about section and profile picture for your bot, see /help for a list of

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

commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

Use this token to access the HTTP API:

8533320067:AAGVc8RESkMDKbthGyN7tq9XDON1wjQTIoA

Keep your token secure and store it safely, it can be used by anyone to control your bot.

For a description of the Bot API, see this page: <https://core.telegram.org/bots/api>

8.2.3. Zulip

Конфигурирование утилиты «AlertManager» и приложения «Zulip» состоит из нескольких этапов. Создание канала и бота в приложении «Zulip» и открытие канала передачи предупреждений в утилите «AlertManager».

Создание канал и бота в приложении «Zulip» требует следующих действий:

- Открыть приложение Zulip и перейти в меню «Каналы»;

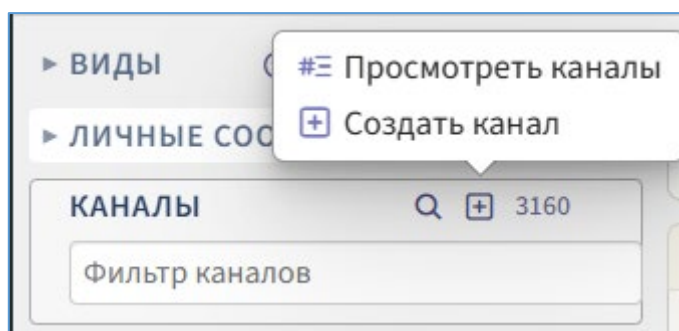


Рисунок 8.5 – Создание канала

- Создать канал;
- Установить настройки нового канала;
- Задать название канала и описание;

The screenshot shows a dialog box titled 'КАНАЛЫ' (Channels) with a close button in the top right corner. The main heading inside is 'Задать настройки нового канала' (Set up new channel). Below this, there are three input fields: 'Название канала' (Channel name) with the text 'alert', 'Описание канала' (Channel description) with the text 'Оповещения JDS', and 'У кого есть доступ к этому каналу' (Who has access to this channel). Under the access section, there are three radio button options: 'Открытый' (Open), 'Закрытый, открытая история переписки' (Closed, open message history), and 'Закрытый, защищенная история переписки' (Closed, protected message history). The third option is selected. At the bottom of the access section, there is a checkbox for 'Анонсировать новый канал в # Общий' (Announce new channel in # General). Below this is a section for 'Дополнительные настройки' (Additional settings). At the very bottom of the dialog are two buttons: 'Отмена' (Cancel) and 'Добавить еще подписчиков' (Add more subscribers).

Рисунок 8.6 – Настройки канала

- Нажать кнопку «Добавить еще подписчиков»

В качестве первого подписчика автоматически будет выбран создатель канала.

- Нажать кнопку «Создать»;
- Создать бота;

Через пиктограмму личного профиля, расположенную в правом верхнем углу приложения Zulip перейти в «Настройки». Выбрать меню «Боты» и нажать кнопку «Добавить нового бота».

В открывшемся окне «Добавить нового бота»:

- Выбрать тип бота – Входящий вебхук;
- Имя (бота) – уникальное имя (в рассматриваемом примере используется имя «alert»);
- Адрес электронной почты бота – допустимо использовать имя бота. Домен почты подключится автоматически;
- Аватар – не обязательно.

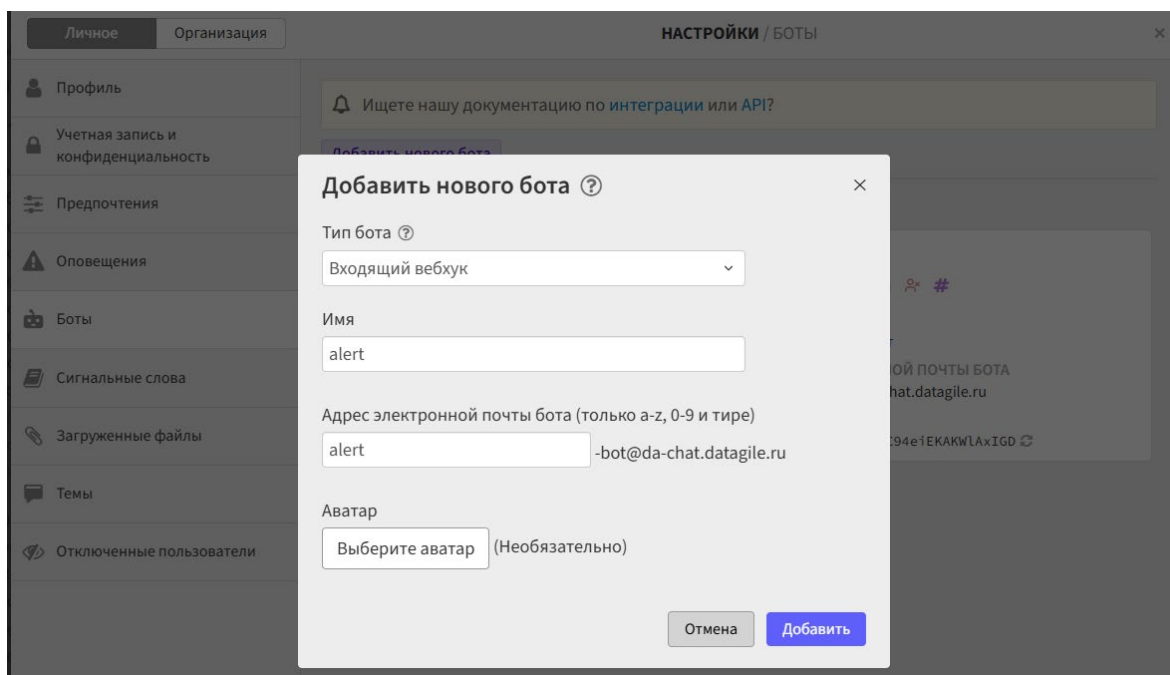


Рисунок 8.7 – Создание бота в Zilip

После установки параметров нажать кнопку «Добавить». Созданный бот отразится в списке «Активные боты».

— Создать «URL для интеграции»

Для интеграции утилиты «AlertManager» и приложения «Zulip» требуется «URL интеграции».

URL создается через пиктограмму в виде цепи, расположенную в блоке бота.

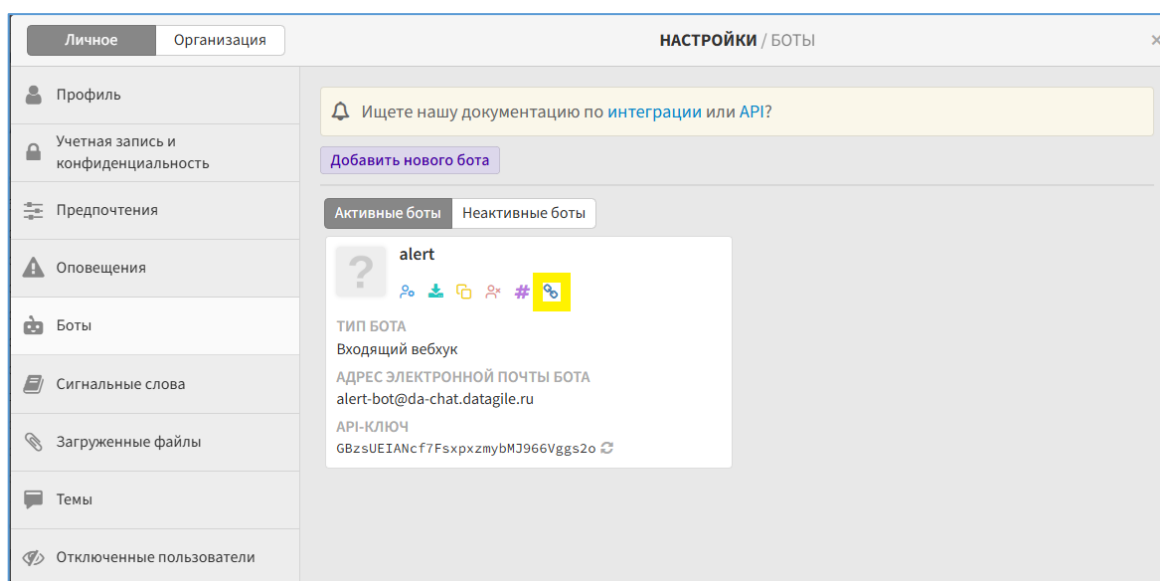


Рисунок 8.8 – Пиктограмма «Создать URL для интеграции»

Нажатие на пиктограмму вызовет окне «Создать URL для интеграции», в котором устанавливаются параметры:

- Интеграция – Prometheus Alertmanager;
- Куда отправлять оповещения – выбрать созданный канал (alert).

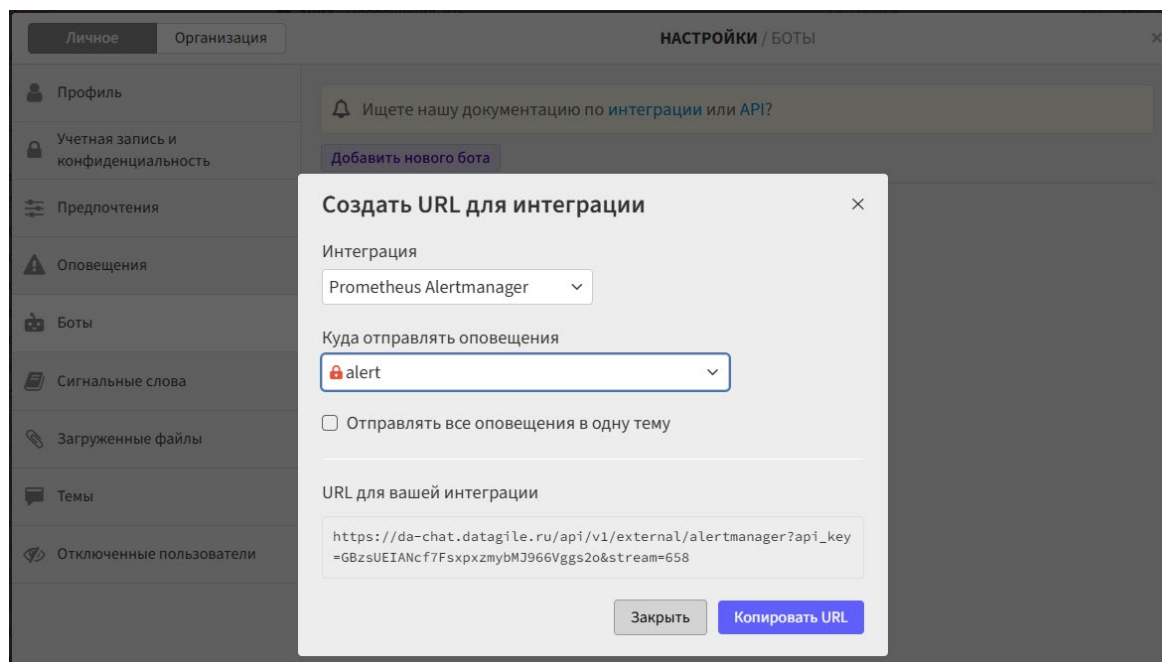


Рисунок 8.9 - Копирование URL для интеграции

Нажатие кнопки «Копировать URL» скопирует созданную ссылку в буфер обмена, после чего далее окно можно закрыть.

— Сконфигурировать «AlertManager»;

Создав бота в Zulip и сгенерировав «URL интеграции» сконфигурируйте «AlertManager» через конфигурационный файл «alertmanager.yml».

Редактирование конфигурационного файла выполняется командой:

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

Раскомментируйте строки:

- receiver: zulip_chat;
- continue: true.

Тем самым создаётся «маршрут» для отправки сообщений по каналу Zulip.

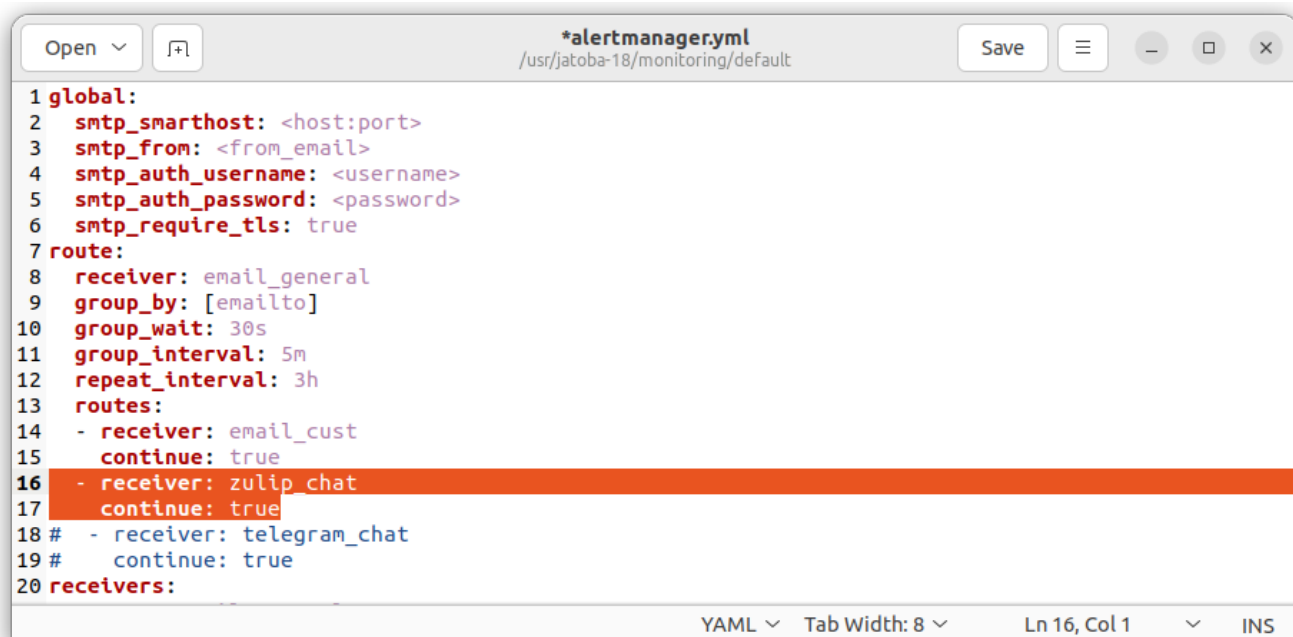


Рисунок 8.10 – Параметры «маршрута» для отправки предупреждений по каналу Zulip

Расскомментируйте строки:

- url;
- send_resolved.

В строку «URL» вставьте «URL для интеграции» скопированный из созданного бота в Zulip и в конце строки добавить значение:

&desc=summary

Данное значение служит для кастомизации сообщений.



Рисунок 8.11 – URL для отправки предупреждений в Zulip

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

На данном конфигурирование утилиты «AlertManager» для отправки сообщений по каналу Zulip закончено и модно переходить к запуску утилиты описанному в п.п. 8.3.

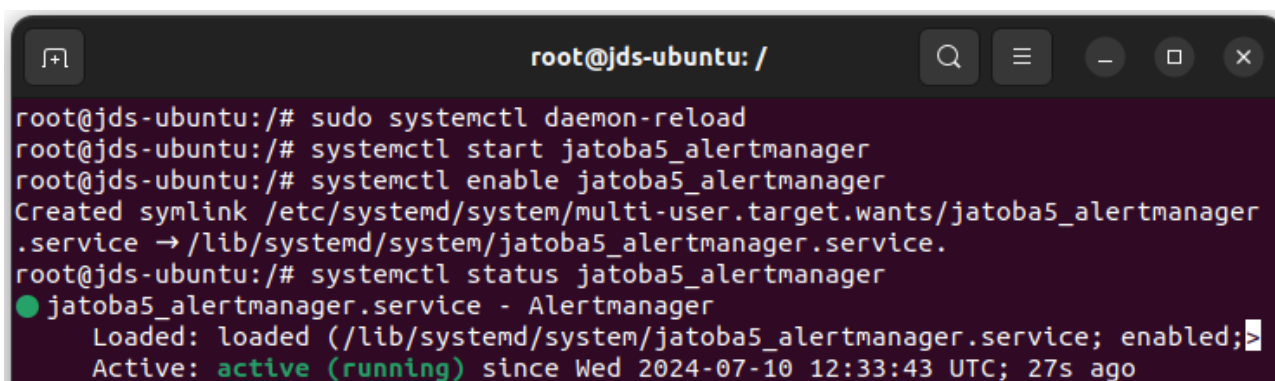
8.3. Запуск утилиты «alertmanager»

Обновить конфигурацию system командой:

```
# 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 8.12 - Запуск и вывод статуса службы «jatoba*_alertmanager»

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

```
localhost:9093
http://0.0.0.0:9093
http://<ip>:9093
```

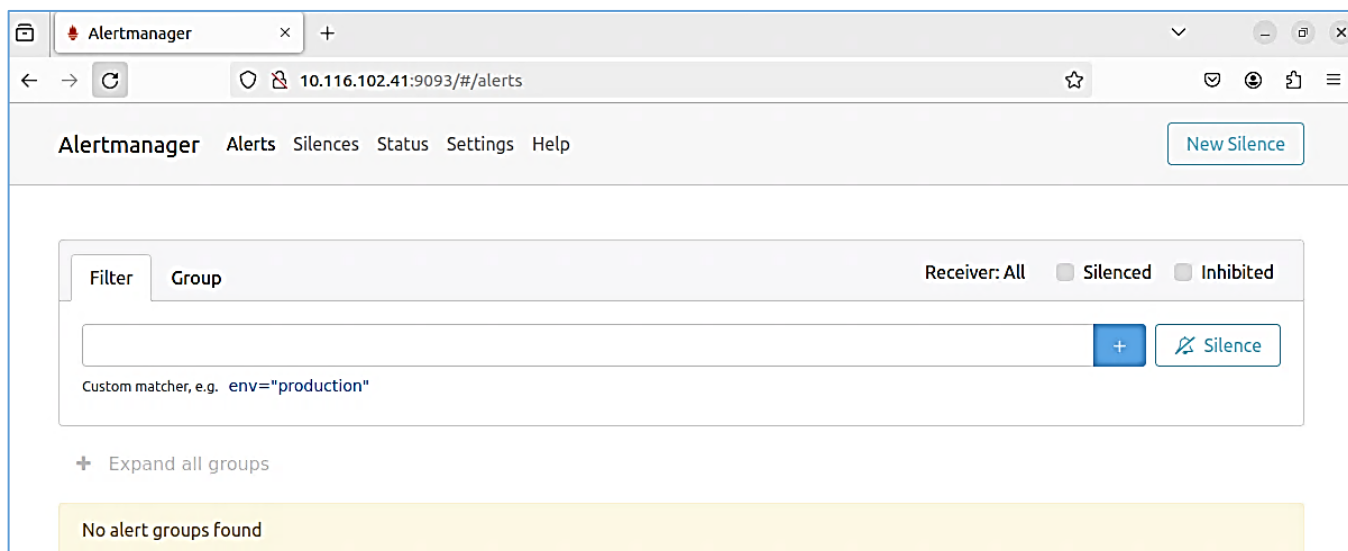


Рисунок 8.13 – Веб-интерфейс утилиты «alertmanager»

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-<ver>/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-<ver>/monitoring/default/alertrules.yml
```

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

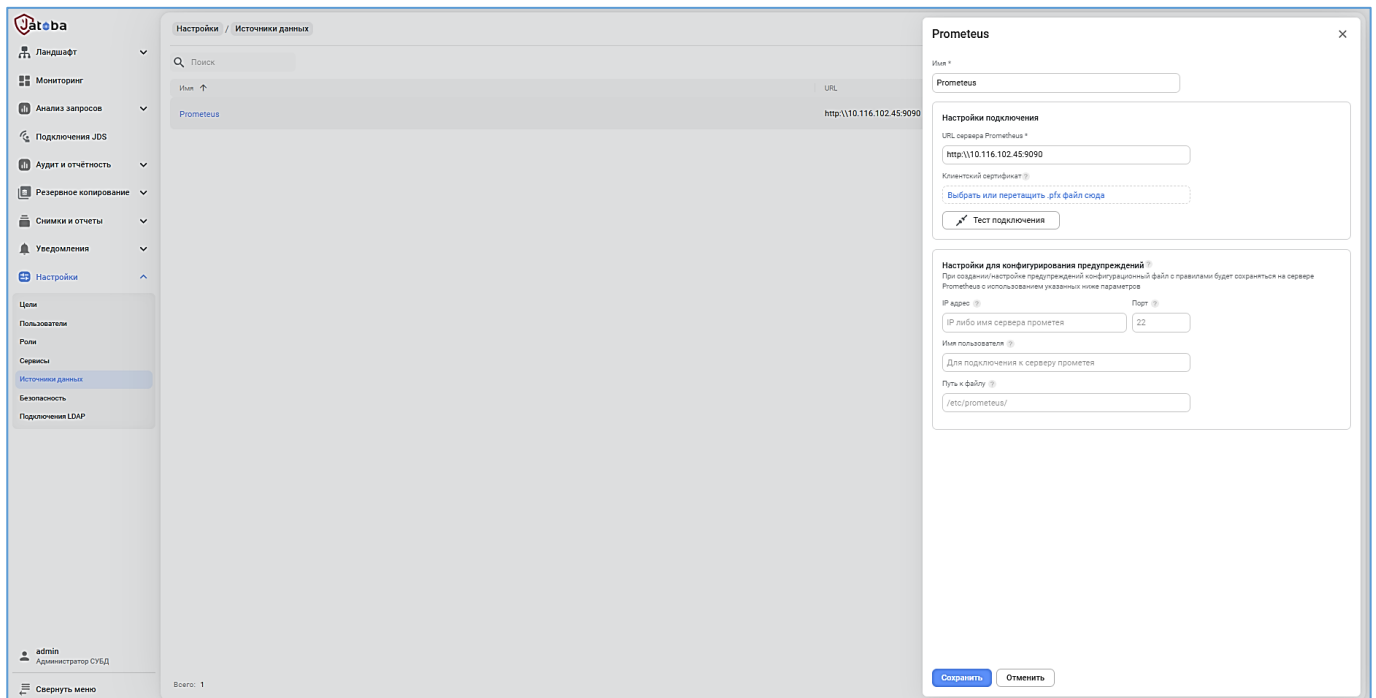


Рисунок 9.1 - «Настройки конфигурации предупреждений»

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-<ver>/monitoring/default/alertrules.yml
```

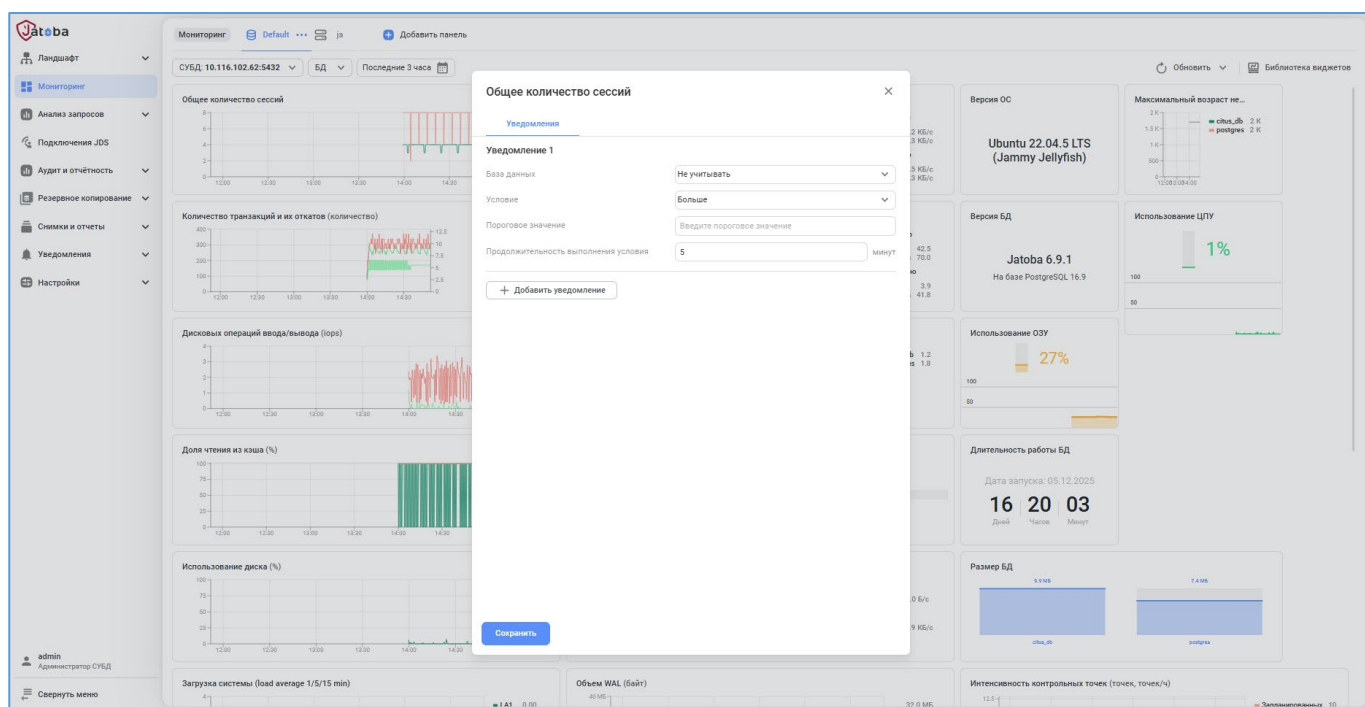


Рисунок 9.2 – Создание уведомления в компоненте JDS

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-<ver>/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"
```

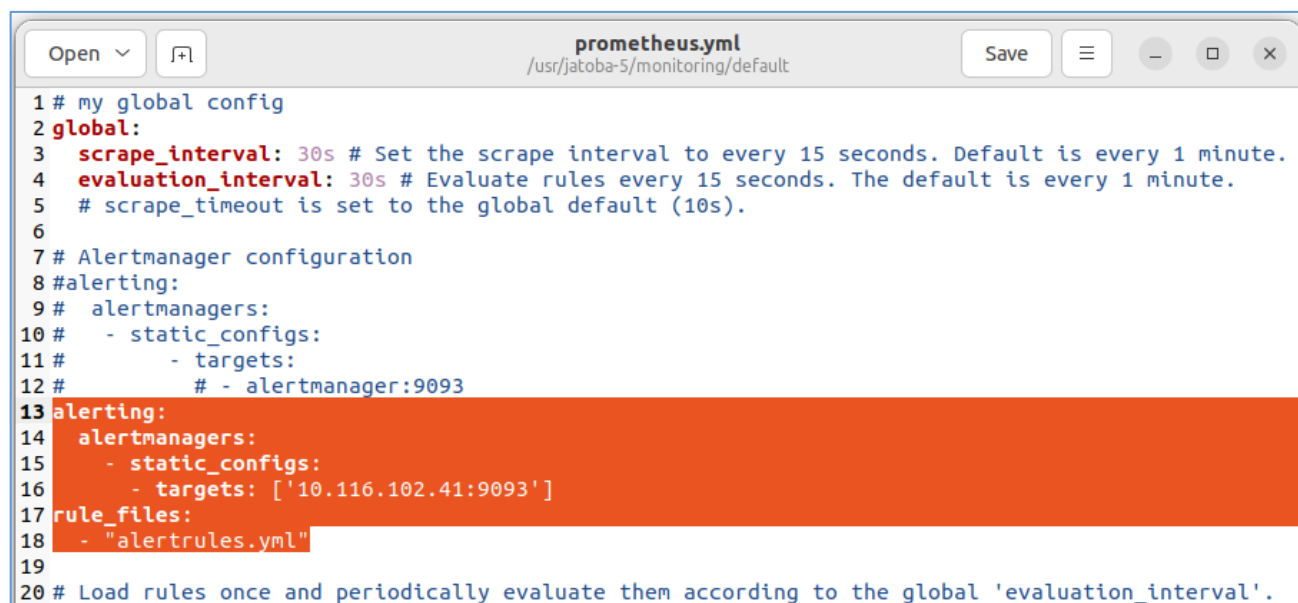


Рисунок 9.3 - Раздел «Alertmanager configuration»

Проверить корректность введенных параметров возможно командами:

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

Если параметры верны, перезапустить службу:

```
# systemctl restart jatoba<ver>_prometheus
```

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

10. НАСТРОЙКА ЭКСПОРТЁРОВ ДЛЯ КОМПОНЕНТА JA_HIPE_CLUSTER

Настройка системы мониторинга подразумевает конфигурирование целевых СУБД и кластера на основе компонента «ja_Hipe_Cluster» в экосистеме СУБД «Jatoba». Используется выделенный сервер мониторинга «Jatoba Data Safe» собирающий данные для виджетов с сервера «Prometheus». Сервера «Prometheus» аккумулирует данные о СУБД и ОС с кластера и СУБД.

10.1. Параметры стенда

Параметры стенда, приведенные в таблице 10.1.

Таблица 10.1 – Конфигурация стенда

№	Имя сервера	IP-адрес	ПО	Port	Роль
1	u602doc-jds01	10.116.102.41/24	JDS		Сервер мониторинга
2	u602doc-pgp01	10.116.102.45/24			Сервер Prometheus
			Prometheus	9090	
			Alert manager	9093, 22	
			pg_stat_statements		
			node_exporter	9100	
			postgres_exporter	9187	
			sql_exporter	9399	
3	u602doc-hipe01	10.116.102.61/24			Coordinator
			citus		
			pg_stat_statements		
			node_exporter	9100	
			postgres_exporter	9187	
			sql_exporter	9399	
4	u602doc-hipe02	10.116.102.62/24			Worker (Node1)
			citus		
			pg_stat_statements		
			node_exporter	9100	
			postgres_exporter	9187	
			sql_exporter	9399	
5	u602doc-hipe03	10.116.102.63/24			Worker (Node2)
			citus		
			pg_stat_statements		
			node_exporter	9100	
			postgres_exporter	9187	
			sql_exporter	9399	
6	u602doc-ldap01	10.116.102.47/24			Целевая СУБД
			pg_stat_statements		
			node_exporter	9100	
			postgres_exporter	9187	
			sql_exporter	9399	

Схема стенда представлена на рисунке 2.1.

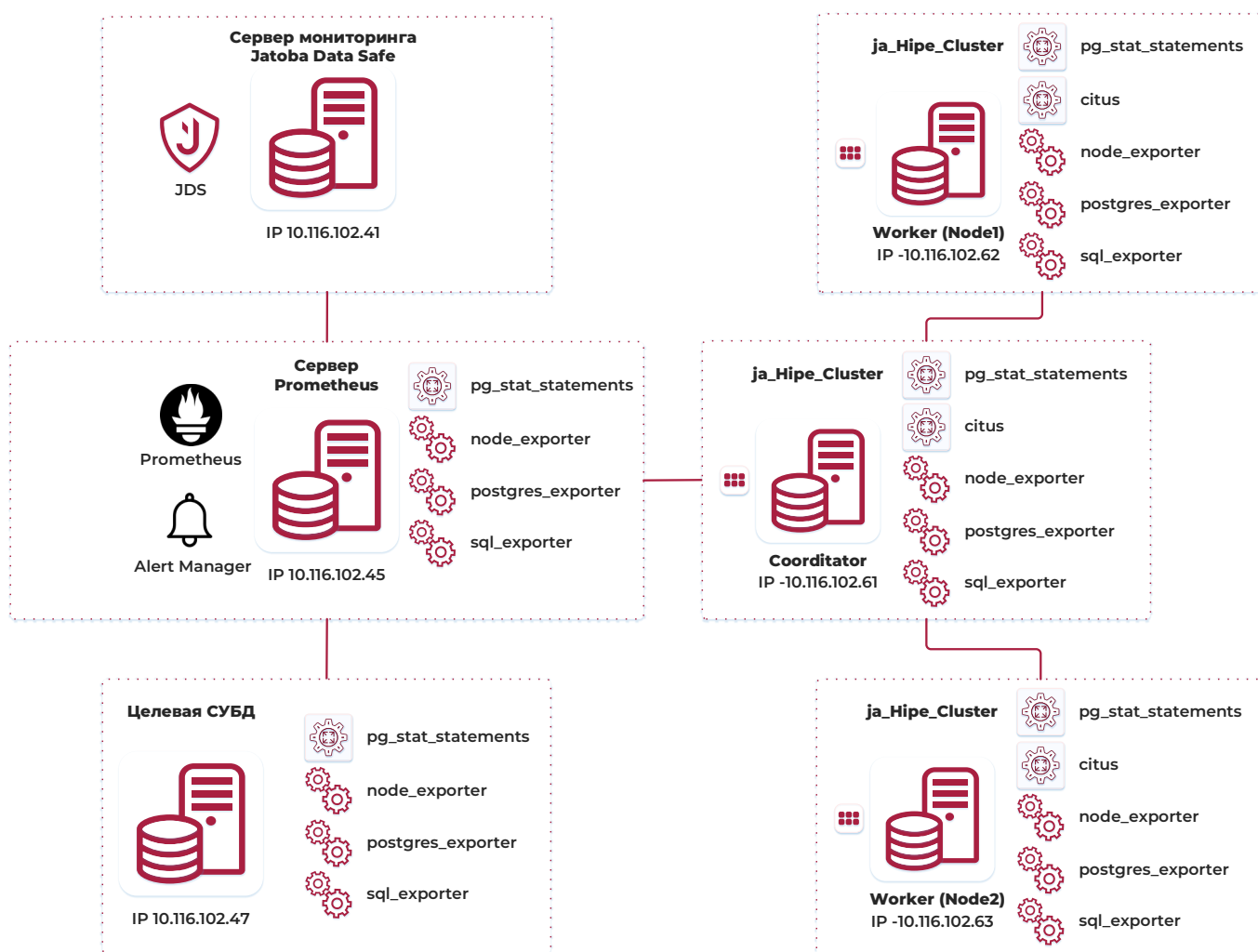


Рисунок 10.1 – Схема стенда мониторинга с кластером «ja_Hipe_Cluster»

Настройка экспортеров на целевой СУБД описан выше в настоящем документе и их параметры остаются неизменными в конфигурационном файле `/usr/jatoba-
<ver>/monitoring/default/prometheus.yml`.

Конфигурация экспортеров кластера добавляется отдельно.

Настройка экспортеров `ja_Hipe_Cluster` для принципиально не отличается от вышеописанного процесса, но имеет ряд особенностей.

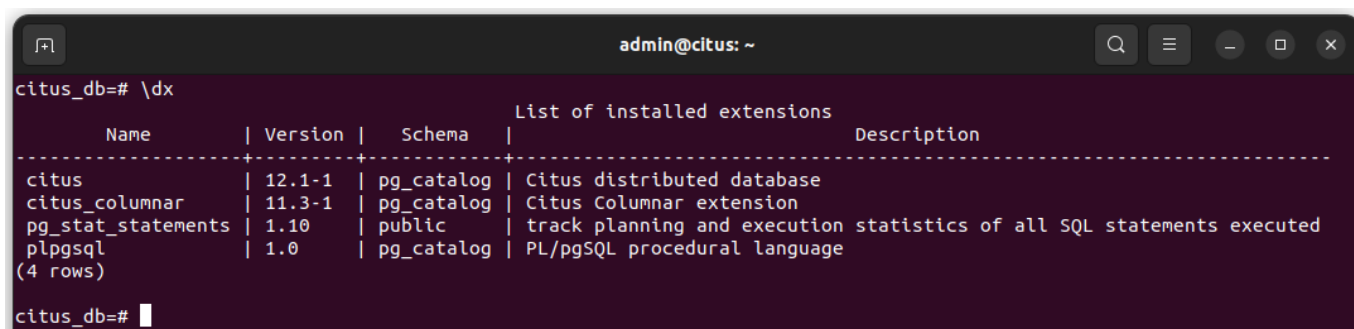
Настройка СУБД в кластере для мониторинга выполняется согласно раздела 3 «Установка и настройка целевых СУБД».

В результате:

- должна быть настроена аутентификация;
- в конфигурационный файл «postgresql.conf» добавлены параметры:

```
shared_preload_libraries = 'citus, pg_stat_statements'  
pg_stat_statements'compute_query_id = on  
pg_stat_statements.max = 10000  
pg_stat_statements.track = all
```

- должны быть установлены, как минимум, расширения pg_stat_statements, citus и citus_columnar.



Name	Version	Schema	Description
citus	12.1-1	pg_catalog	Citus distributed database
citus_columnar	11.3-1	pg_catalog	Citus Columnar extension
pg_stat_statements	1.10	public	track planning and execution statistics of all SQL statements executed
plpgsql	1.0	pg_catalog	PL/pgSQL procedural language

(4 rows)

citus_db=#

Рисунок 10.2 – Вывод установленных расширений

БД установки расширения pg_stat_statements не имеет принципиального значения.

Установка экспортёров выполняется согласно разделам настоящего документа:

- 4 «Установка экспортёра «jatoba*_node_exporter»;
- 5 «Установка экспортёра «jatoba*_postgres_exporter»;
- 6 «Установка экспортёра «jatoba*_sql_exporter».

В конфигурационных файлах «jatoba*_postgres_exporter» и «jatoba*_sql_exporter» в строке «DATA_SOURCE_NAME» должен быть указан IP-адрес узла.

10.2. Конфигурирование системы «Prometheus»

На сервере «Prometheus» (u602doc-pgp01, IP-10.116.102.45/24) в конфигурационный файл /usr/jatoba-<ver>/monitoring/default/prometheus.yml добавляется раздел с экспортёрами, установленными на узлах кластера.

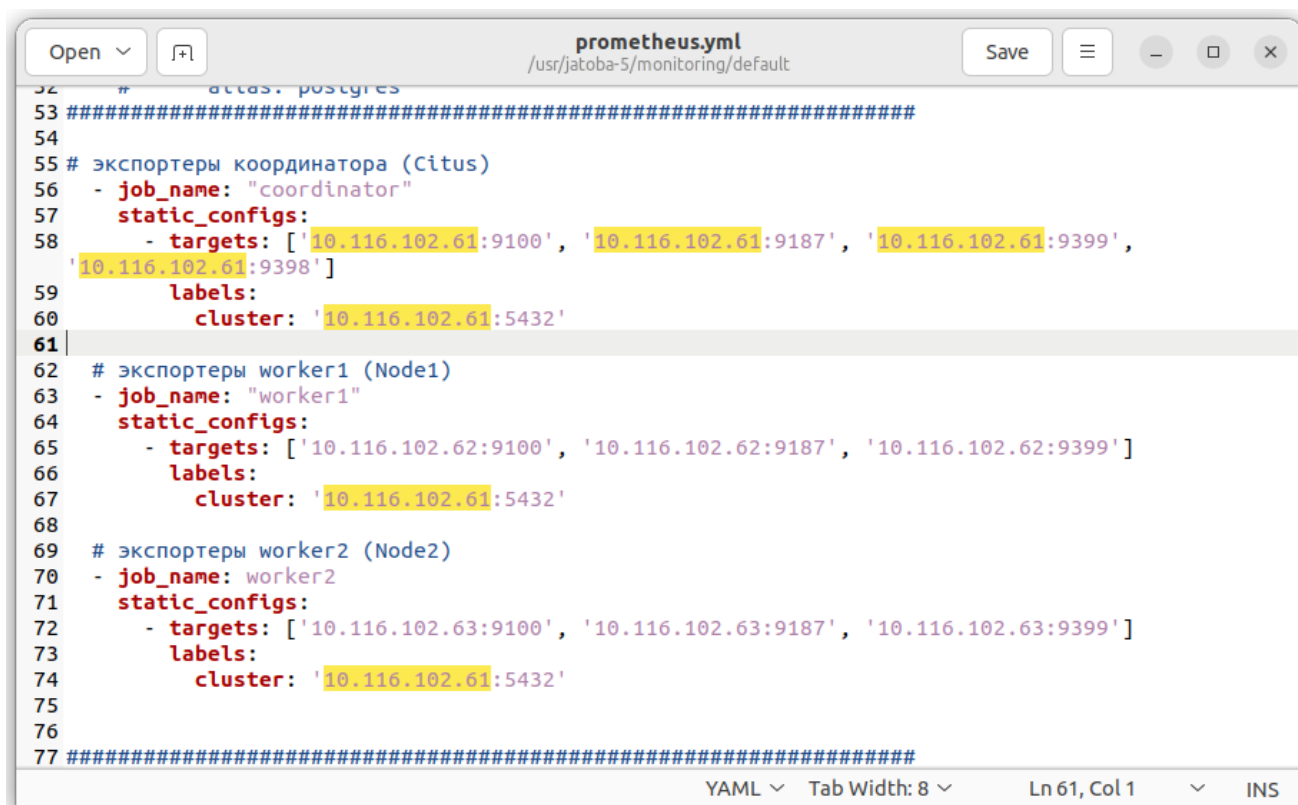
```
#####  
# экспортеры координатора (Citrus)  
- job_name: "coordinator"  
  static_configs:  
    - targets: ['10.116.102.61:9100', '10.116.102.61:9187',  
'10.116.102.61:9399', '10.116.102.61:9398']  
    labels:  
      cluster: 'ja_hipe_cluster'  
  
# экспортеры worker1 (Node1)  
- job_name: "worker1"  
  static_configs:  
    - targets: ['10.116.102.62:9100', '10.116.102.62:9187',  
'10.116.102.62:9399']  
    labels:  
      cluster: 'ja_hipe_cluster'  
  
# экспортеры worker2 (Node2)  
- job_name: worker2  
  static_configs:  
    - targets: ['10.116.102.63:9100', '10.116.102.63:9187',  
'10.116.102.63:9399']  
    labels:  
      cluster: 'ja_hipe_cluster'  
#####
```

В строке «job_name» целесообразнее указать роль узла в кластере.

В строке «targets» перечисляются IP-адрес и порт каждого из установленных экспортёров, установленных на узле.

Строка «labels: cluster:» является меткой для формирования выпадающего списка «Кластеры» по которой формируется кластерная панель виджетов.

Строка является текстовой. Допускается указание IP-адреса узла, как показано на рисунке 10.3.



```
52 # atlas.postgres
53 #####
54
55 # экспортеры координатора (Citrus)
56 - job_name: "coordinator"
57   static_configs:
58     - targets: ['10.116.102.61:9100', '10.116.102.61:9187', '10.116.102.61:9399',
59       '10.116.102.61:9398']
60     labels:
61       cluster: '10.116.102.61:5432'
62
63 # экспортеры worker1 (Node1)
64 - job_name: "worker1"
65   static_configs:
66     - targets: ['10.116.102.62:9100', '10.116.102.62:9187', '10.116.102.62:9399']
67     labels:
68       cluster: '10.116.102.61:5432'
69
70 # экспортеры worker2 (Node2)
71 - job_name: worker2
72   static_configs:
73     - targets: ['10.116.102.63:9100', '10.116.102.63:9187', '10.116.102.63:9399']
74     labels:
75       cluster: '10.116.102.61:5432'
76
77 #####
```

Рисунок 10.3 – Использование IP-адреса узла в строке Строка «labels: cluster:»

Целесообразнее указывать идентификационный признак подключенного к мониторингу кластера, как показано на рисунке 10.4.

```

53 #####
54
55 # экспортеры координатора (Citrus)
56 - job_name: "coordinator"
57   static_configs:
58     - targets: ['10.116.102.61:9100', '10.116.102.61:9187', '10.116.102.61:9399',
59       '10.116.102.61:9398']
60     labels:
61       cluster: 'ja_hipe_cluster'
62
63 # экспортеры worker1 (Node1)
64 - job_name: "worker1"
65   static_configs:
66     - targets: ['10.116.102.62:9100', '10.116.102.62:9187', '10.116.102.62:9399']
67     labels:
68       cluster: 'ja_hipe_cluster'
69
70 # экспортеры worker2 (Node2)
71 - job_name: "worker2"
72   static_configs:
73     - targets: ['10.116.102.63:9100', '10.116.102.63:9187', '10.116.102.63:9399']
74     labels:
75       cluster: 'ja_hipe_cluster'
76 #####
77

```

Рисунок 10.4 - Использование имени кластера в строке Строка «labels: cluster:»

В результате на панели виджетов для кластера, в выпадающем списке отразится имя кластера.

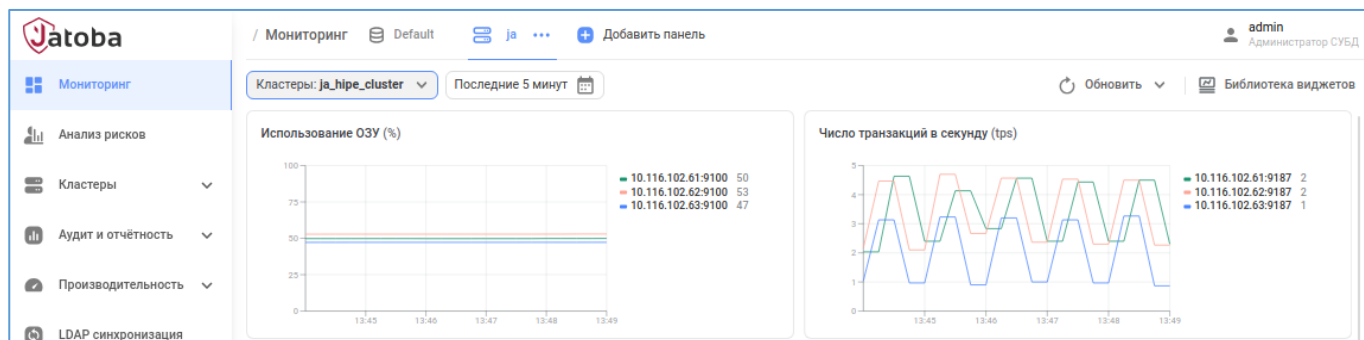


Рисунок 10.5 – Имя кластера на панели виджетов

На виджетах отразятся графы каждого узла кластера.

11. ОБНОВЛЕНИЕ КОМПОНЕНТОВ ЭКСПОРТЕРОВ

Предварительные условия: необходимо в каталоге /localrepo заменить файлы на ту версию дистрибутива, с которой в дальнейшем будет выполняться обновление компонентов. Если это не сделано, необходимо обратиться в раздел «1.2 Обновление СУБД с промежуточных версий» документа «Руководство по обновлению СУБД Jatoba» 643.72410666.00067-07 93 01.

11.1. Обновление компонента sql_exporter

11.1.1. Установка новой версии компонента sql_exporter

Обновление версии компонента экспортера sql_exporter требует сохранения конфигурационных файлов и выполняется на узлах в следующем порядке:

- 1) Скопировать новый пакет компонента sql_exporter в отдельную директорию. Например, в директорию /home/admin/Downloads.
- 2) Загрузить и установить новый пакет jatoba<ver>_sql_exporter от имени и с правами привилегированного пользователя ОС:

– Находясь в каталоге с пакетом:

```
# apt install ./jatoba<ver>-sql-exporter_* -y
```

– Из корневого каталога командой:

```
# apt install /home/admin/Downloads/jatoba<ver>-sql-exporter_*  
-y
```

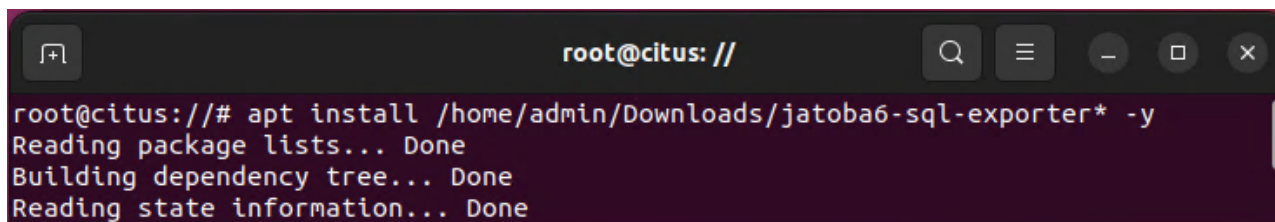


Рисунок 11.1 – Установка нового пакета

При установке новой версии конфигурационные файлы предыдущей версии будут сохранены в каталоге /usr/jatoba-<ver>/monitoring/default с расширением backup.

После установки новой версии `jatoba<ver>_sql_exporter` в каталоге `/usr/jatoba-<ver>/monitoring/default` будет создан дополнительный файл конфигурации `citus.collector.yml`, а также обновлен `postgres.collector.yml`.

3) Восстановить резервные копии конфигураций в каталог `/usr/jatoba-<ver>/monitoring/default`.

На данном шаге обновление компонента `sql_exporter` завершено.

11.1.2. Настройка мониторинга кластера Citus после обновления

Следующие шаги выполняются только на узле-координаторе кластера Citus.

1) Начиная с версии `jatoba6-sql-exporter_0.14.3-440` для поддержки мониторинга `ja_Hipe Citus` на узле-координаторе необходимо дополнительно создать второй экземпляр службы - `jatoba<ver>_sql_exporter.citus.service`. Для этого потребуется скопировать файл сервиса командой в консоли ОС:

```
# cp /lib/systemd/system/jatoba<ver>_sql_exporter.service  
/lib/systemd/system/jatoba<ver>_sql_exporter.citus.service
```

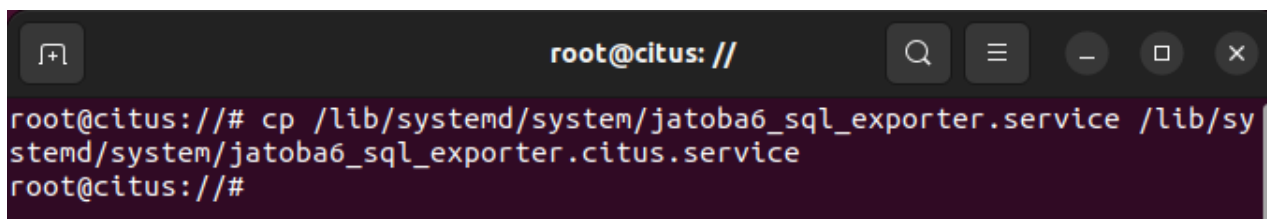


Рисунок 11.2 – Создание второго экземпляра службы

2) Настроить второй экземпляр `jatoba<ver>_sql_exporter.citus.service`. Для этого необходимо открыть файл

```
# nano  
/lib/systemd/system/jatoba<ver>_sql_exporter.citus.service
```

И внести следующие изменения:

```
[Service]  
EnvironmentFile=/usr/jatoba-<ver>/monitoring/default/sql_exporter.citus
```



```

root@citus: //
GNU nano 6.2 /lib/systemd/system/jatoba6_sql_exporter.citus.service *
[Unit]
Description=SQL Exporter for Prometheus
Documentation=https://github.com/burningalchemist/sql_exporter
Wants=network-online.target
After=network-online.target

[Service]
EnvironmentFile=/usr/jatoba-6/monitoring/default/sql_exporter.citus
User=sql_exporter_usr
Group=sql_exporter_usr
Type=simple

ExecStart=/usr/jatoba-6/bin/sql_exporter -config.file=${CONF_FILE} -web.listen-address=${LI
Restart=on-failure
TimeoutStopSec=20
RestartSec=10

[Install]
WantedBy=multi-user.target

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line
  
```

Рисунок 11.3 – Изменение настроек второго экземпляра

3) Скопировать файл командой в консоли ОС:

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

```

root@citus: //
root@citus:## cp /usr/jatoba-6/monitoring/default/sql_exporter.yml /usr/jatoba-6/monitoring
/default/sql_exporter.citus.yml
root@citus:##
  
```

Рисунок 11.4 – Копирование файла настроек sql_exporter.citus.yml

4) Открыть файл sql_exporter.citus.yml

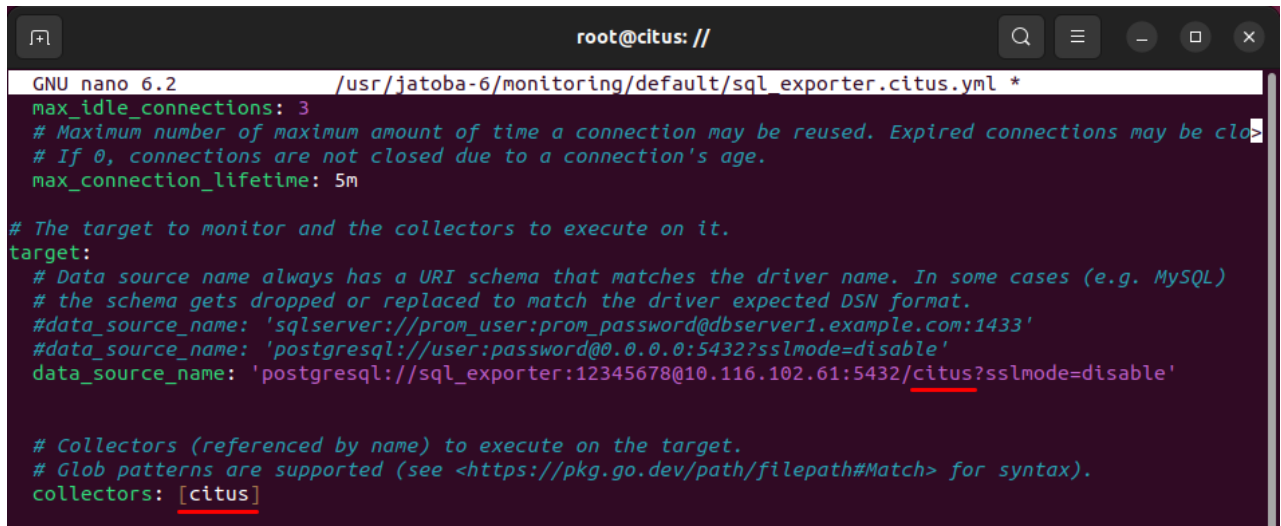
```
# nano /usr/jatoba-
<ver>/monitoring/default/sql_exporter.citus.yml
```

И внести следующие изменения:

```
data_source_name:
'postgresql://sql_exporter:[password]@10.116.102.61:5432/[citus
_db]?sslmode=disable'
collectors: [citus]
```

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

Где password – пароль пользователя СУБД, citus_db – название БД, в которой установлено расширение Citus.



```
root@citus: //
GNU nano 6.2 /usr/jatoba-6/monitoring/default/sql_exporter.citus.yml *
max_idle_connections: 3
# Maximum number of maximum amount of time a connection may be reused. Expired connections may be clo
# If 0, connections are not closed due to a connection's age.
max_connection_lifetime: 5m

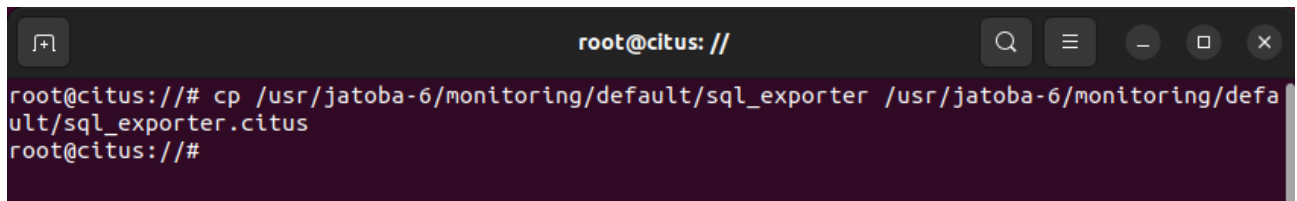
# The target to monitor and the collectors to execute on it.
target:
# Data source name always has a URI schema that matches the driver name. In some cases (e.g. MySQL)
# the schema gets dropped or replaced to match the driver expected DSN format.
#data_source_name: 'sqlserver://prom_user:prom_password@dbserver1.example.com:1433'
#data_source_name: 'postgresql://user:password@0.0.0.0:5432?sslmode=disable'
data_source_name: 'postgresql://sql_exporter:12345678@10.116.102.61:5432/citus?sslmode=disable'

# Collectors (referenced by name) to execute on the target.
# Glob patterns are supported (see <https://pkg.go.dev/path/filepath#Match> for syntax).
collectors: [citus]
```

Рисунок 11.5 – Изменение настроек второго экземпляра

5) Скопировать файл командой в консоли ОС:

```
# cp /usr/jatoba-<ver>/monitoring/default/sql_exporter
/usr/jatoba-<ver>/monitoring/default/sql_exporter.citus
```



```
root@citus: //
root@citus:## cp /usr/jatoba-6/monitoring/default/sql_exporter /usr/jatoba-6/monitoring/default/sql_exporter.citus
root@citus:##
```

Рисунок 11.6 – Копирование файла настроек sql_exporter.yml

6) Открыть файл sql_exporter.citus:

```
# nano /usr/jatoba-<ver>/monitoring/default/sql_exporter.citus
```

И внести следующие изменения:

```
CONF_FILE=/usr/jatoba-
<ver>/monitoring/default/sql_exporter.citus.yml
LISTEN_ADDRESS=0.0.0.0:9366
```

Здесь для доступа к метрикам компонента Citus будет использован сетевой порт 9366.

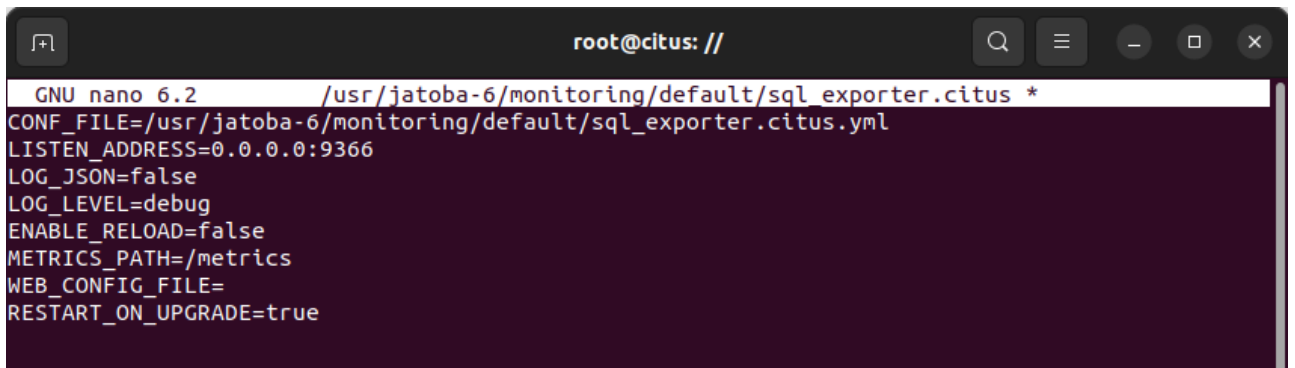


Рисунок 11.7 – Изменение настроек второго экземпляра

7) Обновить конфигурацию systemd:

```
# systemctl daemon-reload
```

8) Запустить второй экземпляр службы jatoba<ver>_sql_exporter.citus, включить ее в автозапуск и проверить статус работы:

```
# systemctl start jatoba<ver>_sql_exporter.citus
# systemctl enable jatoba<ver>_sql_exporter.citus
# systemctl status jatoba<ver>_sql_exporter.citus
```

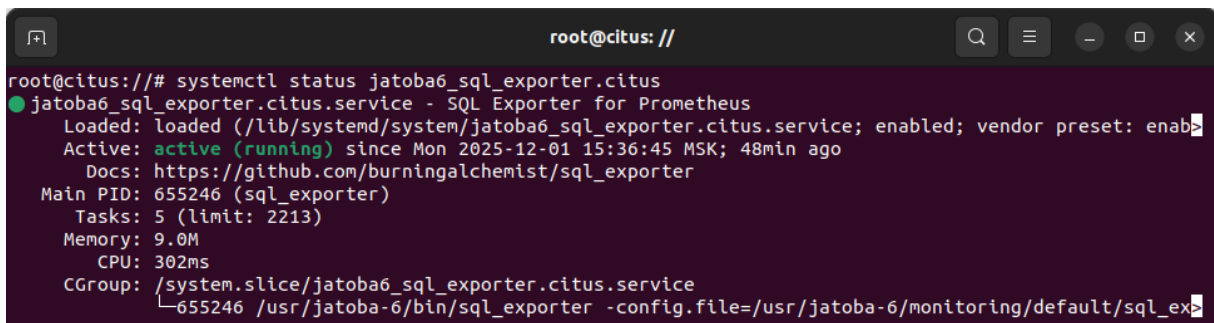


Рисунок 11.8 – Проверка статуса службы второго экземпляра

9) Запустить и проверить статус работы первого экземпляра:

```
# systemctl start jatoba<ver>_sql_exporter
# systemctl status jatoba<ver>_sql_exporter
```

10) Открыть в браузере веб-интерфейс sql_exporter и проверить наличие метрик компонента Citus для второго экземпляра службы:

```
http://localhost:9366/metrics
```

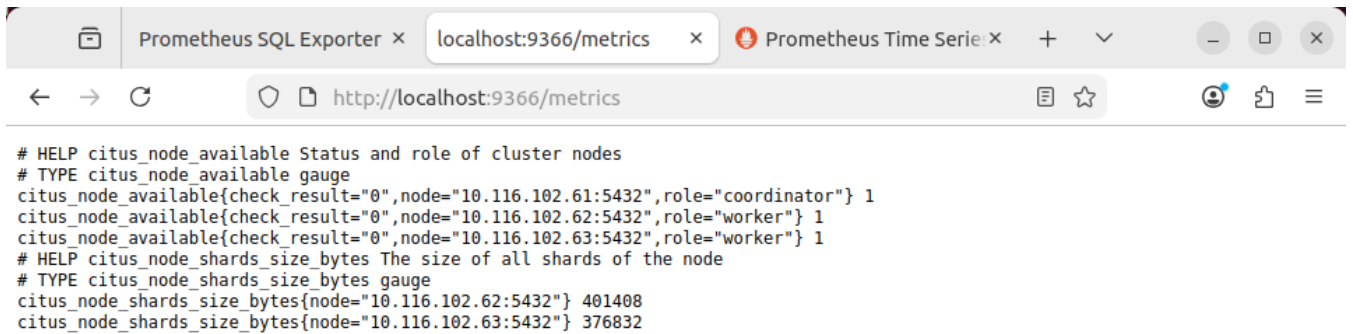


Рисунок 11.9 – Проверка метрик компонента sql_exporter

11) Переключиться на узел с запущенной системой «Prometheus» и открыть на редактирование файл:

```
# nano /usr/jatoba-<ver>/monitoring/default/prometheus.yml
```

В секции «# экспортеры координатора (Citus)» внести (добавить) следующие изменения:

```
# экспортеры координатора (Citus)
- job_name: "coordinator"
  static_configs:
    - targets: [..., 'IP-address:9366']
```

Где IP-address – сетевой адрес узла-координатора.

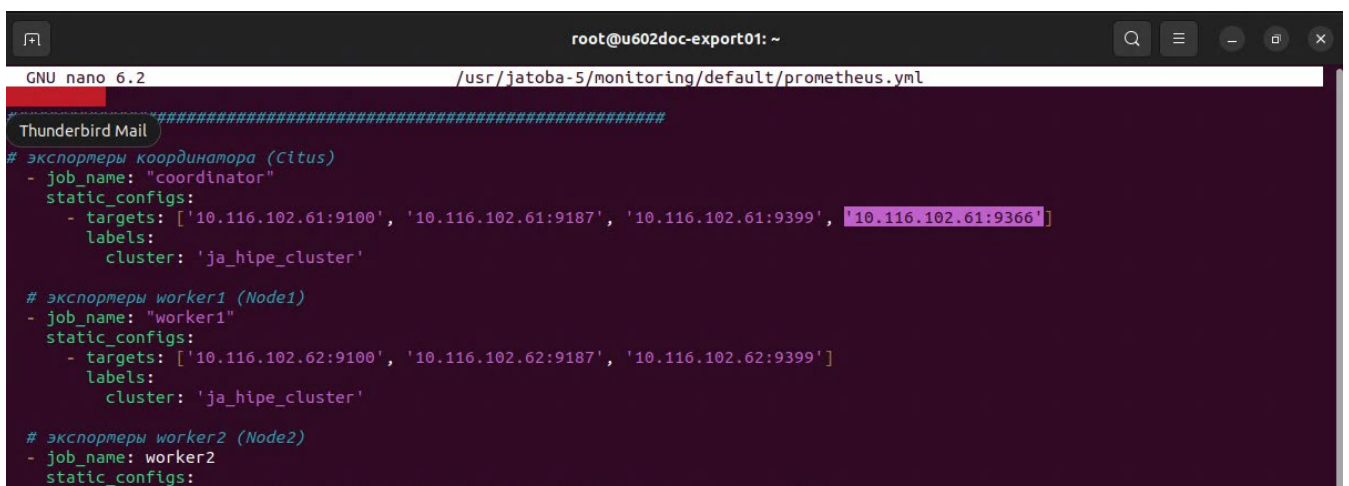


Рисунок 11.10 – Параметры в конфигурационном файле prometheus.yml

12) Перезагрузить службу «Prometheus»

```
# systemctl stop jatoba<ver>_prometheus
# systemctl start jatoba<ver>_prometheus
```

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

13) Открыть веб-интерфейс системы «Prometheus» по адресу `http://<IP-адрес>:9090/targets` и проверить наличие соединения с компонентом `sql_exporter`.

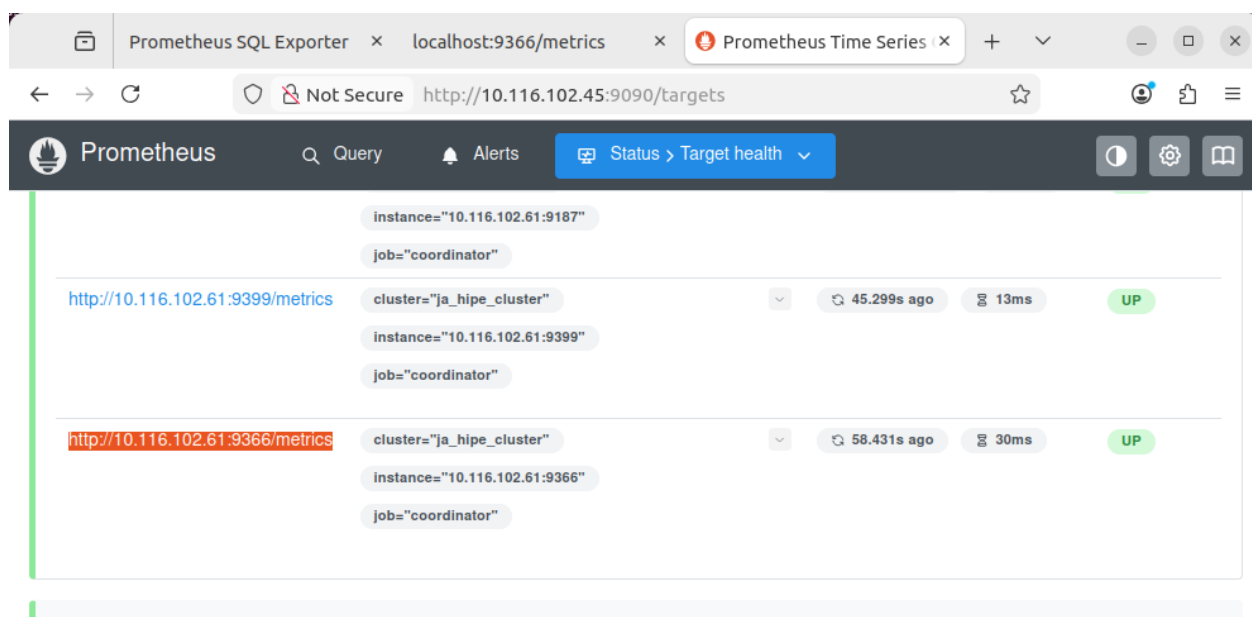


Рисунок 11.11 – Страница «Targets» в системе «Prometheus»

Открыть веб-интерфейс администратора Jatoba Data Safe (JDS) и проверить наличие метрик кластера Citus.

11.2. Обновление компонентов `node_exporter`, `postgres_exporter` и `alertmanager`

Обновление версии компонентов мониторинга СУБД `node_exporter`, `postgres_exporter` и `alertmanager` выполняется на узлах в следующем порядке:

1) Остановка и отключение служб обновляемых компонентов мониторинга СУБД:

```
# systemctl stop jatoba<ver>_node_exporter
jatoba<ver>_postgres_exporter jatoba<ver>_alertmanager
# systemctl disable jatoba<ver>_node_exporter
jatoba<ver>_postgres_exporter jatoba<ver>_alertmanager
```

2) Обновление версии компонентов мониторинга СУБД из локального репозитория:

```
# apt-get install jatoba<ver>-node-exporter jatoba<ver>-
postgres-exporter jatoba<ver>-alertmanager
```

3) После обновления открыть файлы конфигурации компонентов мониторинга СУБД и ознакомиться с новыми параметрами:

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

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

4) Отредактировать пользовательские файлы конфигурации компонентов мониторинга СУБД (с расширением .backup), созданные на этапе обновления с внесением в них параметров из новой версии:

```
# nano /usr/jatoba-
<ver>/monitoring/default/postgres_exporter.backup
# nano /usr/jatoba-
<ver>/monitoring/default/alertmanager.yml.backup
```

и переименовать обратно:

```
# cp /usr/jatoba-
<ver>/monitoring/default/postgres_exporter.backup /usr/jatoba-
<ver>/monitoring/default/postgres_exporter
# cp /usr/jatoba-
<ver>/monitoring/default/alertmanager.yml.backup /usr/jatoba-
<ver>/monitoring/default/alertmanager.yml
```

5) Активация, запуск и проверка статуса компонентов мониторинга СУБД после обновления:

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

6) Запустить веб-интерфейс и убедиться, что служба компонента работает:

- <http://localhost:9100/> - node_exporter;
- <http://localhost:9187/> - postgres_exporter;
- <http://localhost:9093/> - alertmanager.

На данном шаге обновление компонентов мониторинга СУБД завершено.

12. ОБНОВЛЕНИЕ СИСТЕМЫ «PROMETEUS»

Обновление версии системы «Prometheus» требует сохранения конфигурационных файлов системы и выполняется в следующем порядке:

- 1) Сохранить резервную копию каталога конфигураций /usr/jatoba-<ver>/monitoring/default
- 2) Скопировать новый пакет в отдельную директорию. Например, в директорию /home/admin/Downloads.
- 3) Удалить существующий пакет

```
# apt remove jatoba<ver>-prometheus -y
```

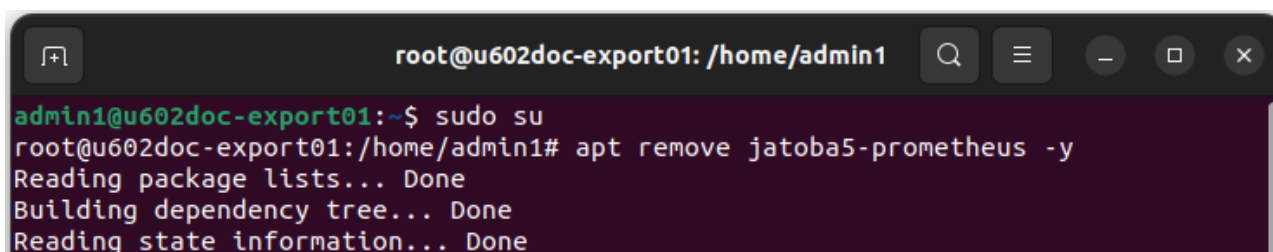


Рисунок 12.1 – Удаление существующего пакета

- 4) Установить новый пакет от имени и с правами привилегированного пользователя ОС:

– Находясь в каталоге с пакетом:

```
# apt install ./jatoba<ver>-prometheus_* -y
```

– Из корневого каталога командой:

```
# apt install /home/admin/Downloads/jatoba<ver>-prometheus* -y
```

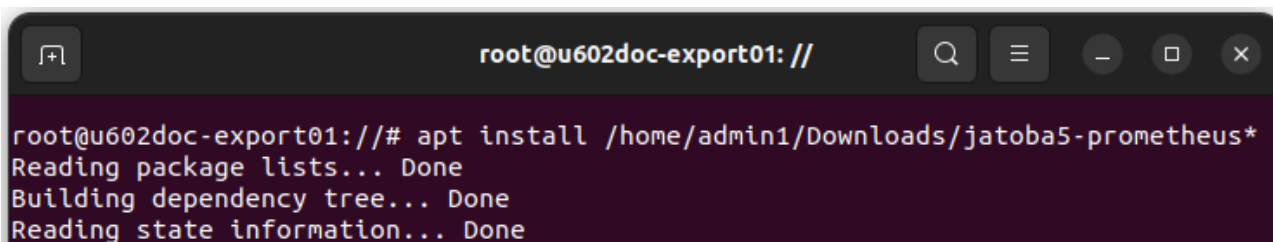


Рисунок 12.2 – Установка нового пакета

- 5) Восстановить резервные копии конфигураций в каталог /usr/jatoba-<ver>/monitoring/default

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

6) Проверить владельца и права на каталог приложения командой:

```
ls -la /opt/prometheus/
```

7) В случае отклонения настроек назначить корректные права командой:

```
chown prometheus:prometheus -R /opt/prometheus/
```

8) Обновить конфигурацию systemd:

```
# systemctl daemon-reload
```

9) Запустить службу системы, включить ее в автозапуск и проверить статус работы:

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

На данном шаге обновление системы «Prometheus» закончено.

В момент обновления компонента данные с экспортеров будут потеряны, т.к. экспортеры не хранят данные, а передают их постоянно.

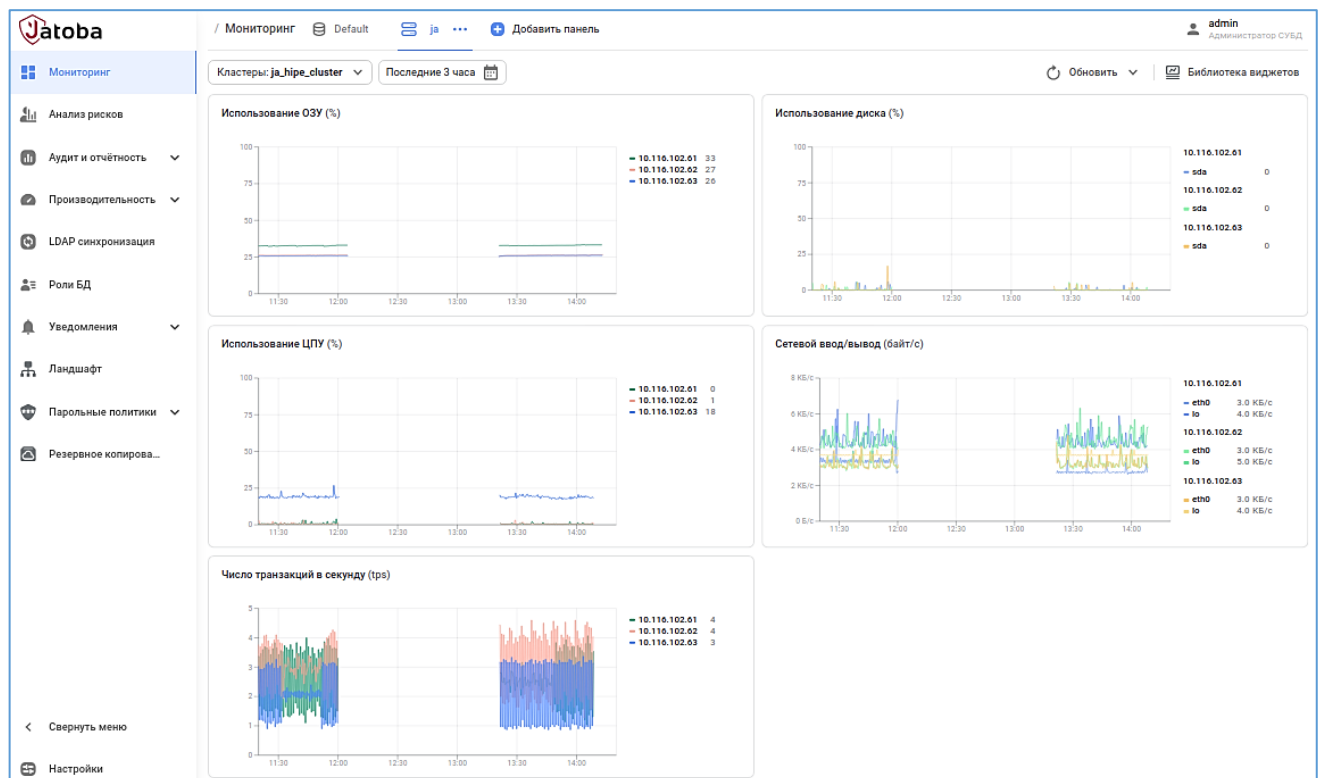


Рисунок 12.3 – Промежуток в данных собранных с экспортеров

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

[illegible]

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