Read the Docs Template Documentation

Выпуск 0.1-alpha

Read the Docs

окт. 04, 2019

Оглавление

Глава 1

Установка и настройка мониторинга ARTA Synergy

1.1 Установка и настройка

Для мониторинга платформы ARTA Synergy используется пакет `arta-synergy-health-monitoring`, разработанный на основе `pmm-client`, в сочетании с PMM-Server. `arta-synergy-health-monitoring` предназначен для мониторинга основных сервисов и инструментов, используемых Synergy: MySQL, JBoss/Wildfly, nginx, Cassandra, Elasticsearch, а также операционной системы сервера.

Данный пакет содержит набор экспортеров метрик, характеризующих состояние вышеперечисленных сервисов, а также консольный инструмент pmm-admin, позволяющий настроить отправку метрик на сервер мониторинга. Собранные метрики сохраняются в Prometheus, затем при помощи Grafana строятся графики состояний, которые можно просматривать в браузере.

1.1.1 Установка PMM-Server

PMM-Server рекомендуется устанавливать на отдельном сервере, операционная система которого поддерживает Docker. Для каждого отслеживаемого узла требуется примерно 1 ГБ памяти на диске при сроке хранения данных, равном одной неделе. Согласно официальной документации, требуемый минимальный объем памяти составляет 2 ГБ для одного отслеживаемого узла, но, когда вы добавляете больше узлов, необходимый объём памяти возрастает нелинейно. Например, данные с 20 узлов должны легко обрабатываться с 16 ГБ памяти.

Установка Docker

Предварительно следует установить некоторые дополнительные пакеты:

```
aptitude install apt-transport-https ca-certificates curl software-properties-common
```

Далее добавить ключ для хранилища Docker:

```
wget https://download.docker.com/linux/debian/gpg
```

```
apt-key add gpg
```

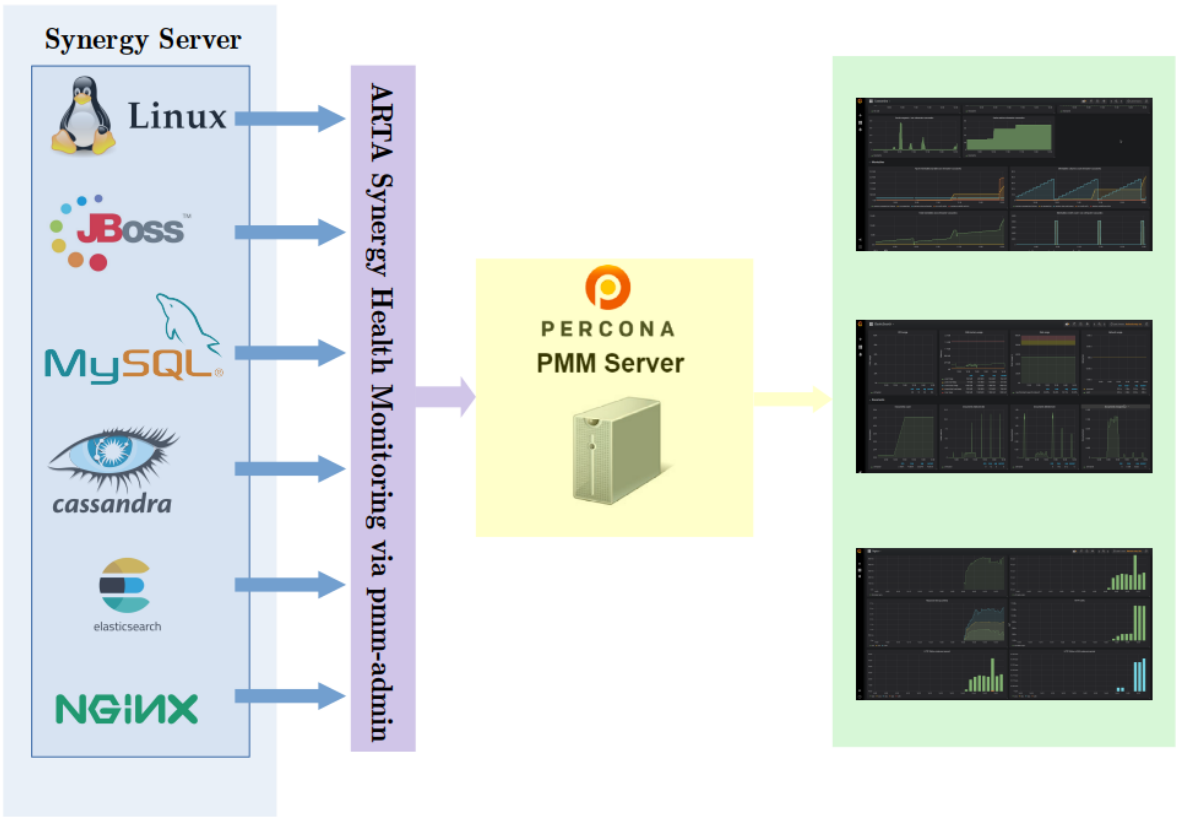


Рис. 1: Архитектура мониторинга Arta Synergy

Затем подключить репозиторий Docker:

```
echo "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs)
stable" | sudo tee -a /etc/apt/sources.list.d/docker.list
```

Обновить список пакетов и установить Docker:

```
aptitude update
```

```
aptitude install docker-ce
```

После установки запустить и включить Docker для запуска при загрузке:

```
systemctl start docker
```

```
systemctl enable docker
```

Создание контейнеров PMM-Server

Для установки собственно PMM-Server'а нужно получить его образ с Docker Hub:

```
docker pull percona/pmm-server:latest
```

Затем создать контейнер для обновляемых данных мониторинга:

```
docker create \
-v /opt/prometheus/data \
-v /opt/consul-data \
-v /var/lib/mysql \
-v /var/lib/grafana \
--name pmm-data \
percona/pmm-server:latest /bin/true
```

Данная команда делает следующее:

- `docker create` создаёт контейнер на основе указанного образа;
- опция -v инициализирует тома для хранения данных в контейнере;
- опция `--name` задаёт имя для контейнера, в данном случае `pmm-data`;
- `percona/pmm-server:latest` указывает название и версию образа, на основе которого создаётся контейнер.

Примечание: Этот контейнер запускать не нужно, он существует для сохранения данных мониторинга в случае, например, обновления образа PMM-Server. Не удаляйте и не пересоздавайте контейнер, если вы не намереваетесь начать мониторинг сначала, обнулив данные.

Следующая команда создаёт и запускает контейнер PMM-Server:

```
docker run -d \
-p 8080:80 \
--volumes-from pmm-data \
--name pmm-server \
--restart always \
percona/pmm-server:latest
```

- опция -d запускает контейнер в фоновом режиме;
- опция -p определяет порт для доступа к PMM-Server через браузер, в примере это порт 8080;

- опция `--volumes-from` примонтирует тома из ранее созданного контейнера `pmm-data`;
- опция `--name` задаёт имя для контейнера, в данном случае `pmm-server`;
- опция `--restart` определяет политику перезапуска контейнера; `always` означает, что Docker запустит контейнер при запуске и в случае отключения контейнера.
- `percona/pmm-server:latest` указывает название и версию образа, на основе которого создаётся контейнер.

После этого в браузере по адресу `http://host:8080` должна быть доступна стартовая страница Percona Monitoring and Management:

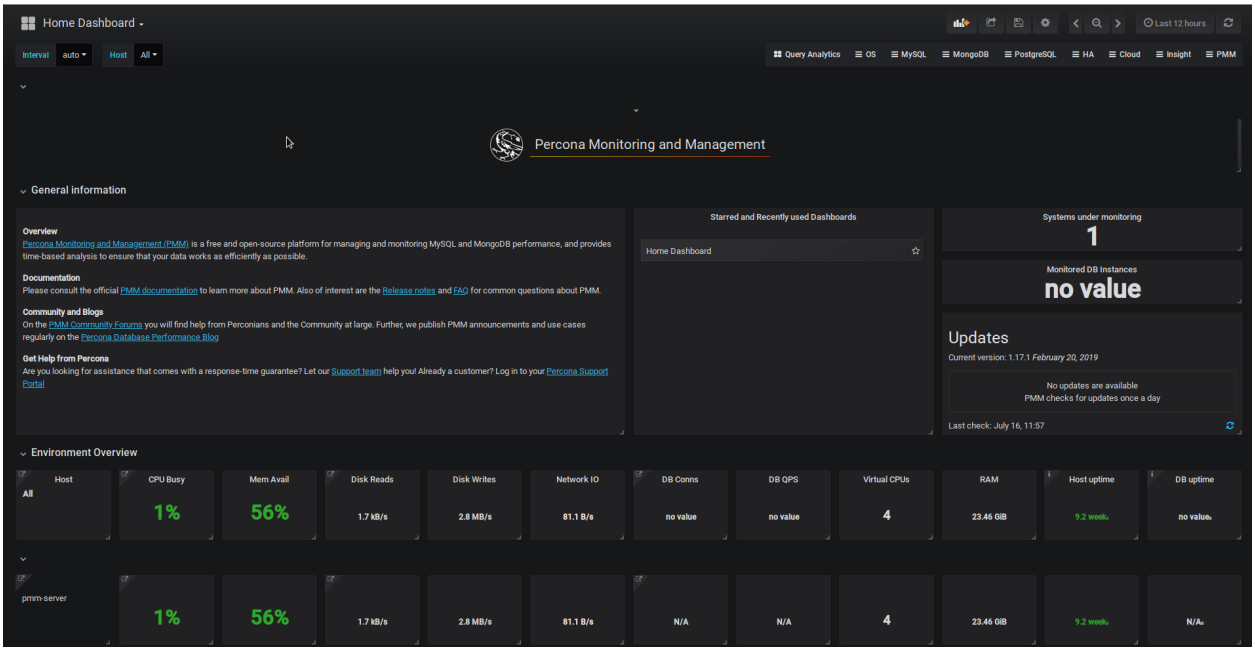


Рис. 2: Стартовая страница PMM

Более подробные инструкции по работе с PMM-Server можно найти на официальном сайте Percona.

В коробочную версию PMM-Server нужно импортировать необходимые дашборды для мониторинга: JBoss, nginx, Cassandra, Elasticsearch.

Для импорта нужно нажать название текущего дашборда в левом верхнем углу и выбрать пункт Import dashboard:

В открывшемся окне нажать Upload .json File, выбрать нужный дашборд и указать источником данных Prometheus, затем нажать `Import`:

1.1.2 Установка клиента на сервер Synergy

Для установки пакета arta-synergy-health-monitoring нужно добавить в `/etc/apt/sources.list` следующую строку:

```
deb [allow-insecure=yes] http://deb.arta.kz/tengri shm main contrib non-free
```

Затем обновить список пакетов и установить нужный пакет:

```
aptitude update
```

```
aptitude install arta-synergy-health-monitoring
```

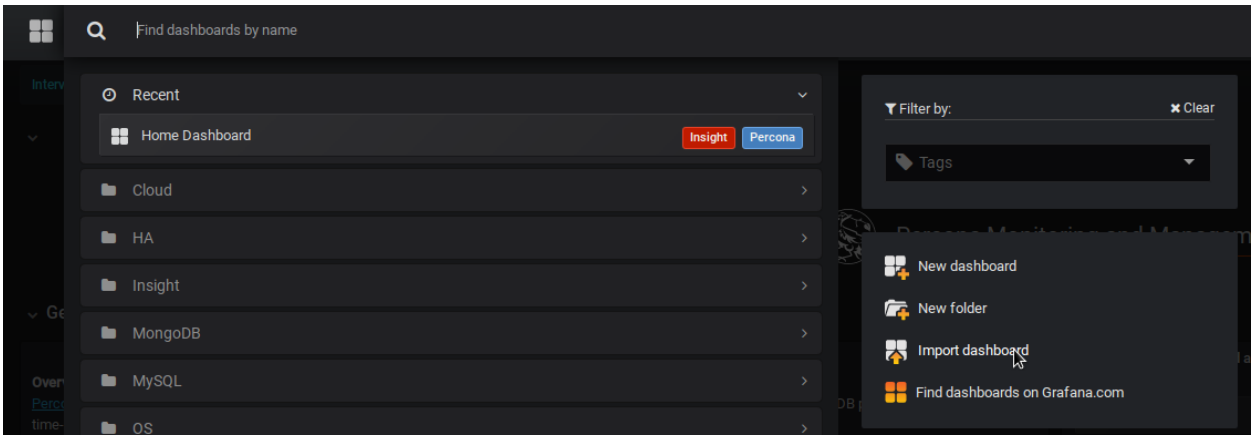


Рис. 3: Импорт дашборда

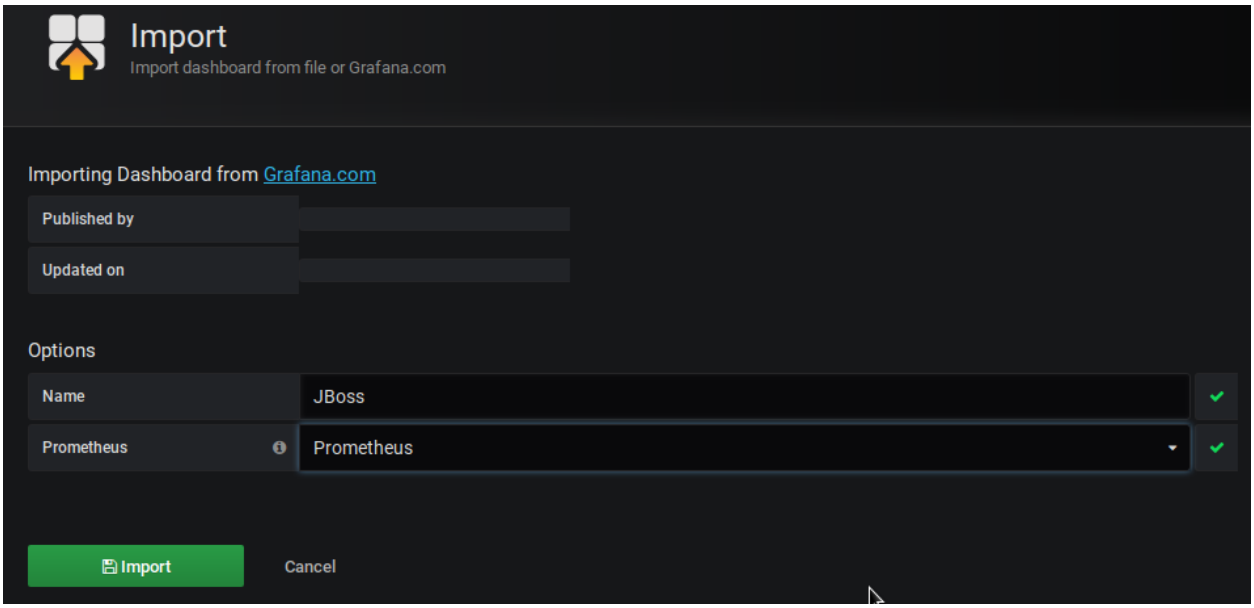


Рис. 4: Выбор json-файла

1.1.3 Настройка мониторинга на сервере Synergy

PMM-Server должен быть доступен с сервера Synergy, это можно проверить, например, командой `ping`.

В первую очередь нужно установить соединение между сервером Synergy и PMM-Server. Для этого используется консольная команда:

```
pmm-admin config --server=server[:port]
```

Если соединение успешно установилось, в консоли должен появиться подобный вывод:

```
root@client:~# pmm-admin config --server=192.168.2.234:8080
OK, PMM server is alive.

PMM Server      | 192.168.2.234:8080
Client Name     | client
Client Address  | 192.168.2.84
```

После этого можно подключать мониторинг требуемых сервисов.

Основные команды pmm-admin

Возможные команды, настройки и флаги pmm-admin можно посмотреть, выполнив в терминале `pmm-admin --help`.

Для просмотра списка и состояния наблюдаемых сервисов используется команда `pmm-admin list`.

В общем случае команда подключения мониторинга сервиса выглядит так:

`pmm-admin add external:services job_name [instance] --service-port=port [flags]`, где нужно указать имя задания для Prometheus, порт, используемый экспортером, и при необходимости название инстанса.

Инстанс нужно указывать, если настраивается мониторинг одновременно JBoss/ Wildfly и Cassandra, а также при добавлении нескольких нод одного сервиса, собранных в кластер.

Кроме того, можно добавить инстанс к уже существующему заданию Prometheus. Если, к примеру, позже была подключена ещё одна нода какого-либо сервиса и для неё также установлен пакет SHM, используется следующая команда:

```
pmm-admin add external:instances job_name [host1:port1[=instance1]]
[host2:port2[=instance2]] ... [flags]
```

Например, настроен мониторинг одного экземпляра Elasticsearch:

```
root@hamming:~# pmm-admin list
pmm-admin 1.17.1

PMM Server      | 192.168.2.234:8080
Client Name     | hamming
. . . . .
Job name  Scrape interval  Scrape timeout  Metrics path  Scheme  Target              Labels       ␣
↪      Health
elastic   1m0s             10s             /metrics      http    192.168.2.198:9114  instance=
↪"node1"    UP
```

Команда для добавления ноды может выглядеть так:

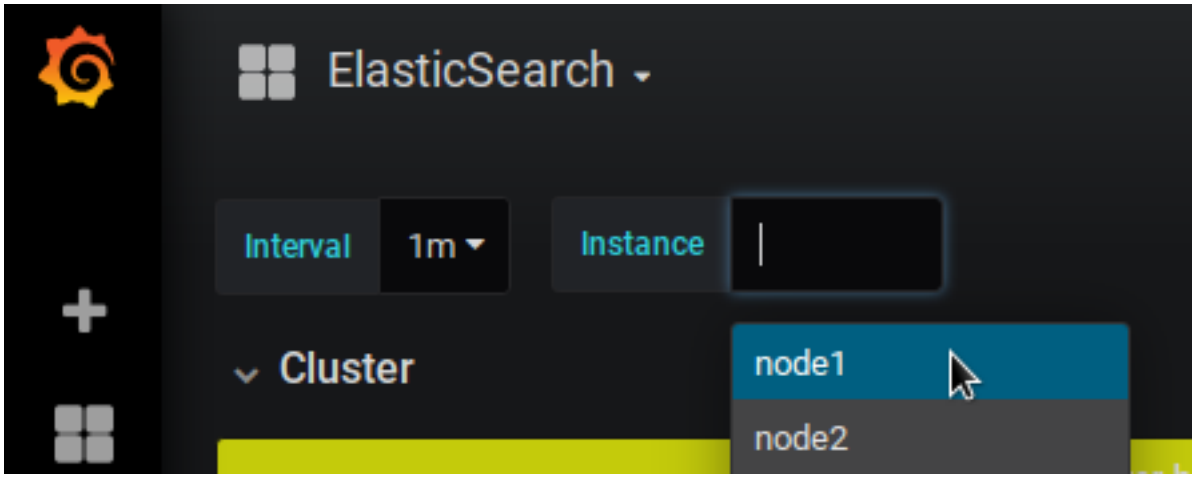
```
pmm-admin add external:instances elastic 192.168.2.181:9114=node2
```

Добавленная нода в списке:

```
root@hamming:~# pmm-admin list
pmm-admin 1.17.1

PMM Server      | 192.168.2.234:8080
Client Name     | hamming
. . . . .
Job name  Scrape interval  Scrape timeout  Metrics path  Scheme  Target              Labels
↪      Health
elastic   1m0s             10s             /metrics      http    192.168.2.198:9114  instance=
↪"node1"    UP
elastic   1m0s             10s             /metrics      http    192.168.2.181:9114  instance=
↪"node2"    UP
```

На соответствующем дашборде можно будет выбрать нужную ноду в параметре `Instance`:



Для удаления ненужных сервисов используется команда:

```
pmm-admin remove external:service job_name --service-port=port [flags]
```

Например, для Elasticsearch:

```
pmm-admin remove external:service elastic --service-port=9114
```

Для удаления отдельного инстанса:

```
pmm-admin remove external:instances job_name [host1:port1] [host2:port2] ... [flags]
```

Примечание: Удаляются только задания для инстанса, на котором выполнена команда. Чтобы удалить задания для других инстансов, нужно выполнять аналогичные команды на них.

Мониторинг Linux и MySQL

Для мониторинга операционной системы и MySQL в консоли нужно выполнить команду, используя логин и пароль пользователя для mysql:

```
pmm-admin add mysql --user root --password root
```

Если мониторинг этих сервисов успешно добавился, команда `pmm-admin list` должна показать подобный список:

```
root@client:~# pmm-admin list
pmm-admin 1.17.1

PMM Server      | 192.168.2.234:8080
Client Name     | client
Client Address  | 192.168.2.84
Service Manager | linux-systemd

-------------- ------- ----------- -------- ---------------------------------------- -----------
↪------------------------------------------------------------------------
SERVICE TYPE   NAME    LOCAL PORT  RUNNING  DATA SOURCE                              OPTIONS
-------------- ------- ----------- -------- ---------------------------------------- -----------
↪------------------------------------------------------------------------
mysql:queries  client  -           YES      root:***@unix(/var/run/mysqld/mysqld.sock)  query_
↪source=slowlog, query_examples=true, slow_log_rotation=true, retain_slow_logs=1
linux:metrics  client  42000       YES      -
mysql:metrics  client  42002       YES      root:***@unix(/var/run/mysqld/mysqld.sock)
```

Мониторинг JBoss/Wildfly

Для мониторинга JBoss/Wildfly нужно добавить следующие строки в `/opt/synergy/jboss/bin/standalone.conf`:

```
JAVA_OPTS="$JAVA_OPTS -Xbootclasspath/p:/opt/synergy/jboss/modules/system/layers/base/org/wildfly/
↪common/main/wildfly-common-1.4.0.Final.jar"
JAVA_OPTS="$JAVA_OPTS -Xbootclasspath/p:/opt/synergy/jboss/modules/system/layers/base/org/jboss/
↪logmanager/main/jboss-logmanager-2.1.4.Final.jar"
JAVA_OPTS="$JAVA_OPTS -Djboss.modules.system.pkgs=org.jboss.byteman,org.jboss.logmanager"
JAVA_OPTS="$JAVA_OPTS -Djava.util.logging.manager=org.jboss.logmanager.LogManager"
JAVA_OPTS="$JAVA_OPTS -javaagent:/opt/synergy/jboss/standalone/lib/ext/jmx_prometheus_javaagent.
↪jar=58080:/opt/synergy/jboss/standalone/configuration/jboss.yaml"
```

Затем сделать доступной статистику источников данных, добавив `statistics-enabled="true"` для SynergyDS и StorageDS в `/opt/synergy/jboss/standalone/configuration/standalone-onesynergy.xml`:

```
<xa-datasource jndi-name="java:/SynergyDS" pool-name="synergy_ds" enabled="true" use-ccm="false"␣
↪statistics-enabled="true">
    <xa-datasource-property name="URL">
        jdbc:mysql://127.0.0.1:3306/synergy?useUnicode=true&amp;characterEncoding=utf8
    . . . . . .
</xa-datasource>
```

...

```
<xa-datasource jndi-name="java:/StorageDS" pool-name="storage_ds" enabled="true" use-ccm="false"␣
↪statistics-enabled="true">
    <xa-datasource-property name="URL">
        jdbc:mysql://127.0.0.1:3306/storage?useUnicode=true&amp;characterEncoding=utf8
    . . . . . .
</xa-datasource>
```

После этого нужно перезапустить JBoss/Wildfly.

По умолчанию конфигурационный файл `jboss.yaml` и экспортер метрик `jmx_prometheus_javaagent.jar` находятся в вышеуказанном расположении. Если вы перемещаете их, укажите новый путь в `standalone.conf`. Здесь также можно указать другой порт для передачи метрик.

Собираемые метрики доступны для просмотра в браузере по адресу `http://server:58080/metrics`. Если в `standalone.conf` используется другой порт, в адресной строке нужно указывать его.

Далее следует добавить мониторинг JBoss/Wildfly в PMM-Server, в общем случае команда выглядит так:

`pmm-admin add external:service --service-port=58080 jboss testserver-jboss`, где

- jboss - имя задания для Prometheus;
- testserver-jboss - название инстанса. Так как аналогичный экспортер используется и для Cassandra, нужно указать название явно, чтобы метрики JBoss/Wildfly и Cassandra не смешивались на графиках. Кроме того, нужно указывать инстансы в случаях, когда используется несколько экземпляров.

Мониторинг nginx

Для мониторинга nginx нужно добавить следующие строки в `/etc/nginx/nginx.conf` в секцию `Logging settings`:

```
log_format shm '$remote_addr - $remote_user [$time_local] '
                        '"$request" $status $body_bytes_sent '
                        '"$http_referer" "$http_user_agent" '
                        '$request_time $hostname';
```

Далее в файле `/etc/nginx/sites-enabled/synergy-base` в разделе `location /Synergy` добавить в описание лога `synergy.access.log` определённый шагом ранее формат, в данном случае `shm`:

```
location /Synergy {
    proxy_pass              http://127.0.0.1:8080/Synergy;
    ...
    access_log /var/log/nginx/synergy.access.log shm;
}
```

Чтобы изменения вступили в силу, нужно перезапустить nginx:

`/etc/init.d/nginx restart`

В конфигурационном файле `/etc/prometheus-nginxlog-exporter.hcl` нужно указать параметры `app` и `instance`, также можно назначить другой порт (по умолчанию 4040):

```
listen {
  port = 4040
  address = "0.0.0.0"
}

namespace "synergy" {
  format = "$remote_addr - $remote_user [$time_local] \"$request\" $status $body_bytes_sent \"
↪$http_referer\" \"$http_user_agent\" $request_time $hostname"
  source_files = [
    "/var/log/nginx/synergy.access.log"
  ]
  labels {
    app = "nginx284"
    instance = "testserver-nginx"
    }

  histogram_buckets = [.005, .01, .025, .05, .1, .25, .5, 1, 2.5, 5, 10]
}
```

Запустить мониторинг запросов nginx:

```
systemctl start prometheus-nginxlog-exporter.service
```

Собираемые метрики доступны для просмотра в браузере по адресу `http://server:4040/metrics`.

Далее следует добавить мониторинг nginx в PMM-Server, в общем случае команда выглядит так:

```
pmm-admin add external:service --service-port=4040 nginx testserver-nginx --interval 10s
```

Параметр `--interval` указывает интервал сбора метрик в PMM-Server, для экспортера nginx лучше указать интервал меньше, чем интервал по умолчанию, который равен 1 минуте.

Мониторинг Cassandra

Для мониторинга Cassandra нужно добавить в `/etc/cassandra/cassandra-env.sh` строку:

```
JVM_OPTS="$JVM_OPTS -javaagent:/usr/share/cassandra/lib/jmx_prometheus_javaagent.
jar=7070:/etc/cassandra.yaml"
```

Затем закомментировать в `/etc/init.d/cassandra` строки:

```
# Read Cassandra environment file.
#. /etc/cassandra/cassandra-env.sh

#if [ -z "$JVM_OPTS" ]; then
#    echo "Initialization failed; \$JVM_OPTS not set!" >&2
#    exit 3
#fi
```

Выполнить обновление конфигов `systemd`:

```
systemctl daemon-reload
```

Перезапустить Cassandra:

```
/etc/init.d/cassandra restart
```

По умолчанию конфигурационный файл `cassandra.yaml` и экспортер метрик `jmx_prometheus_javaagent.jar` находятся в вышеуказанном расположении. Если вы перемещаете их, укажите новый путь в `cassandra-env.sh`. Здесь также можно указать другой порт для мониторинга.

Собираемые метрики доступны для просмотра в браузере по адресу `http://server:7070/metrics`. Если в `cassandra-env.sh` используется другой порт, в адресной строке нужно указывать его.

Далее следует добавить мониторинг Cassandra в PMM-Server, в общем случае команда выглядит так:

```
pmm-admin add external:service --service-port=7070 cassandra testserver-cassandra
```

Так как аналогичный экспортер используется и для JBoss/Wildfly, нужно указать явно название инстанса (в данном случае testserver-cassandra), чтобы метрики JBoss/Wildfly и Cassandra не смешивались на графиках. Кроме того, нужно указывать инстансы в случаях, когда используется кластер из нескольких нод Cassandra.

Мониторинг Elasticsearch

Запуск сбора метрик Elasticsearch осуществляется с помощью команды:

```
systemctl start prometheus-elasticsearch-exporter.service
```

Добавление мониторинга в PMM-Server:

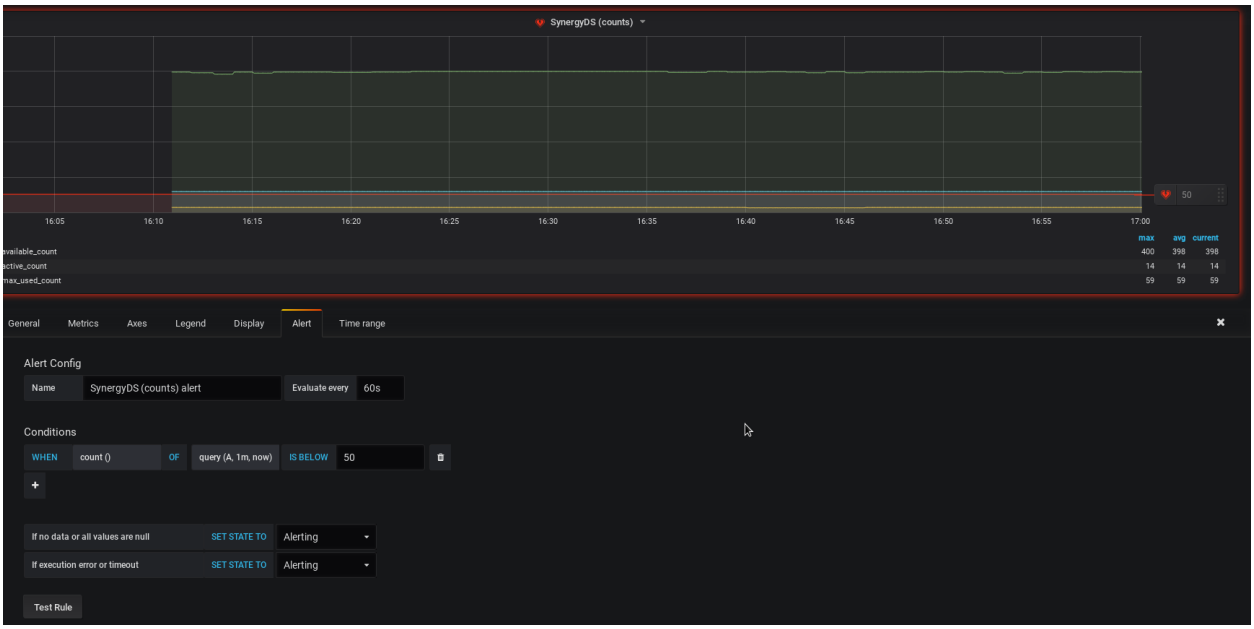
```
pmm-admin add external:service --service-port=9114 elastic
```

1.2 Дополнительные настройки SHM

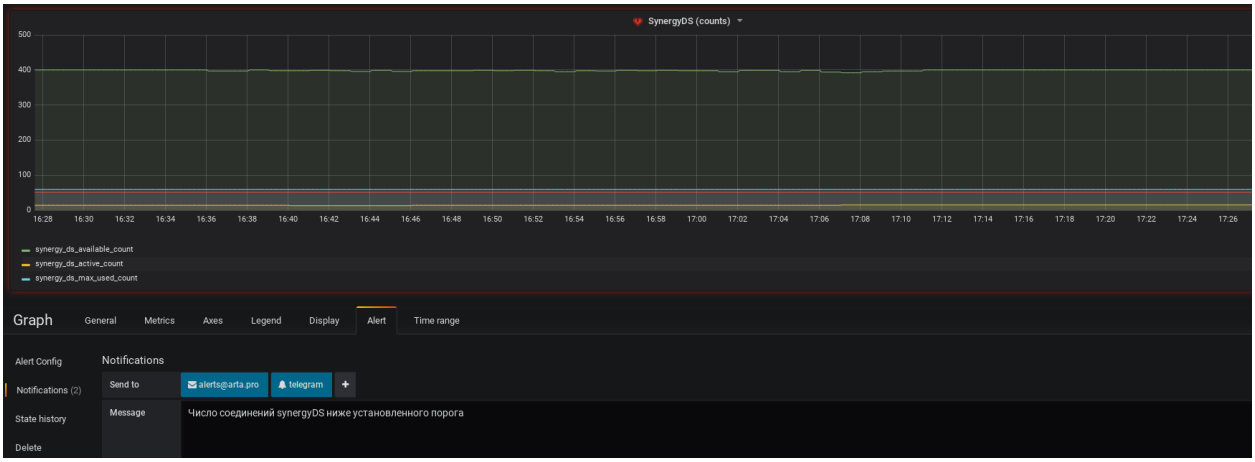
1.2.1 Настройки уведомлений

В данном разделе показан процесс настройки уведомлений при использовании мониторинга SHM.

Достижение параметрами критических значений фиксируется Grafana. Для указания критических значений нужно открыть на редактирование график, например, датасорсов JBoss, затем перейти на вкладку Alert.



Здесь нужно указать название и интервал пересчёта данных, а также условие, при котором будет срабатывать предупреждение. Условие на картинке означает следующее: уведомлять, если значение метрики в запросе A (available count) за последнюю минуту было ниже 50. Далее указать каналы, по которым будут отправляться уведомления, в данном случае e-mail и Telegram, и сохранить.



E-mail

Для настройки уведомлений по e-mail в первую очередь следует настроить smtp для Grafana на PMM-Server. Для этого на машине, где установлен PMM-Server, нужно:

1. получить ID контейнера

```
root@server:~# docker ps -a
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS
↪            PORTS                              NAMES
a4c99272241a        percona/pmm-server:latest      "/opt/entrypoint.sh"     2 months ago        Up 7
↪days           443/tcp, 0.0.0.0:8080->80/tcp      pmm-server
3e5ea7123736        percona/pmm-server:latest      "/bin/true"              2 months ago        Created
↪                                                  pmm-data
```

2. зайти в контейнер

```
docker exec -it a4c99272241a /bin/bash
```

3. открыть на редактирование файл

```
vi /etc/grafana/grafana.ini
```

4. в секции [smtp] указать настройки почты

```
[smtp]
enabled = true
host = smtp.yandex.com:465
user = alerts@arta.pro
# If the password contains # or ; you have to wrap it with trippel quotes. Ex """#password;"""
password = """qwerty!@#"""
;cert_file =
;key_file =
skip_verify = true
from_address = alerts@arta.pro
from_name = Grafana
# EHLO identity in SMTP dialog (defaults to instance_name)
;ehlo_identity = dashboard.example.com
```

5. выйти и перезапустить контейнер

```
docker restart pmm-server
```

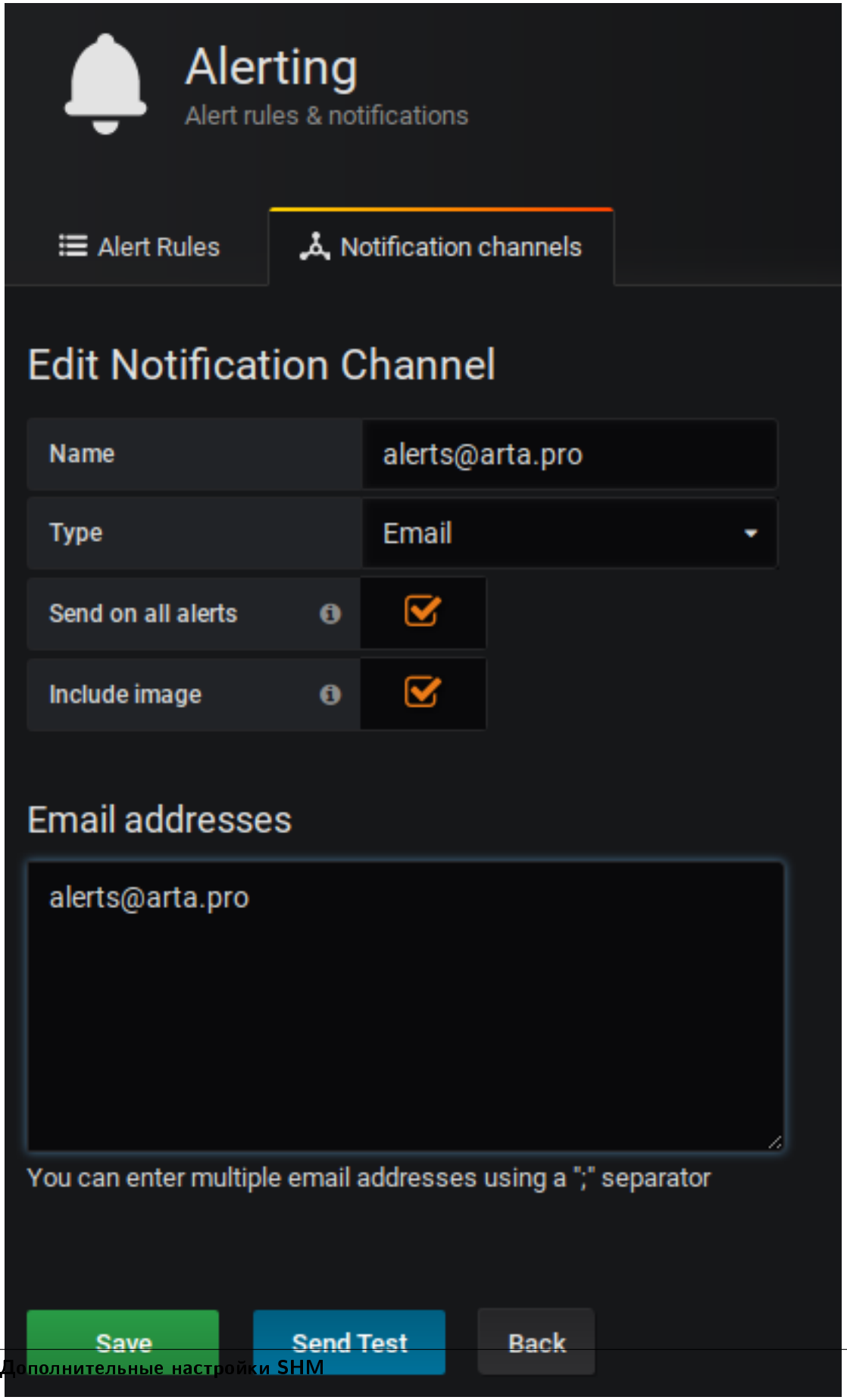
После этого нужно открыть в браузере меню Alerting и выбрать Notification Channels, указать название, тип Email и почтовый адрес, на который будут отправляться уведомления:

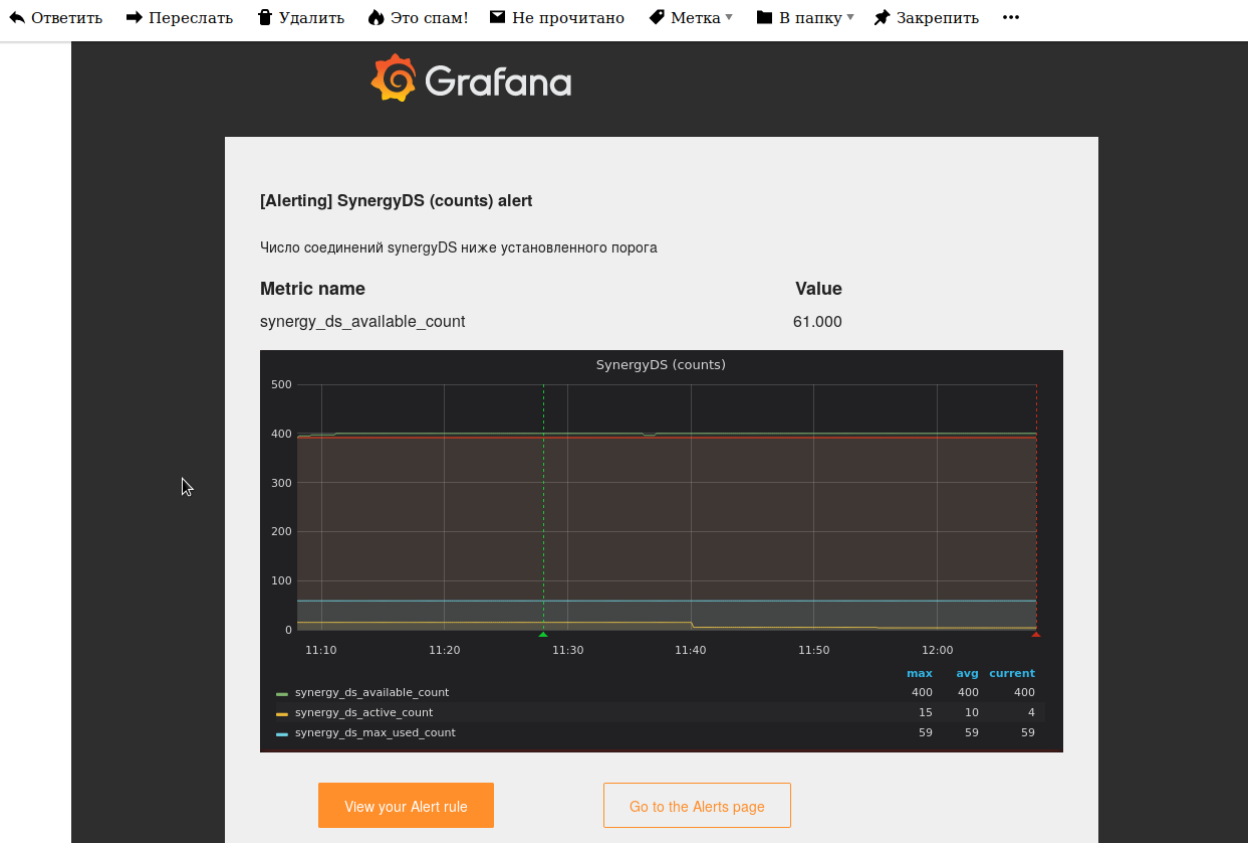
При достижении настроенным параметром установленного значения на почту будет отправлено подобное сообщение:

Telegram

Для уведомлений в Telegram нужно создать telegram-бот, например, через `@BotFather`. В Telegram нужно отправить этому пользователю следующие сообщения:

1. `/start`
2. `/newbot`
3. ввести имя бота, например, `shm_alerts`


Alerting
Alert rules & notifications
Alert Rules
Notification channels
Edit Notification Channel
Name
alerts@arta.pro
Type
Email
Send on all alerts
Include image
Email addresses
alerts@arta.pro
You can enter multiple email addresses using a ";" separator
Save
Send Test
Back



4. ввести юзернейм бота (должен быть уникальным и заканчиваться на _bot), например, `<your_project_name>_alerts_bot`

В ответ должно прийти сообщение с токеном. После этого нужно открыть меню Alerting и выбрать Notification Channels, указать название, тип Telegram и вставить полученный токен в `BOT API Token`.

В Telegram отправить боту хотя бы одно сообщение, например, `/my_id @shm_alerts`. Затем открыть в браузере `https://api.telegram.org/botYOUR_TOKEN/getUpdates`, подставив свой токен, и скопировать chatID в соответствующее поле.

При достижении настроенным параметром установленного значения в telegram будет отправлено подобное сообщение:

1.2.2 Отображение на графиках метрик более чем одной ноды

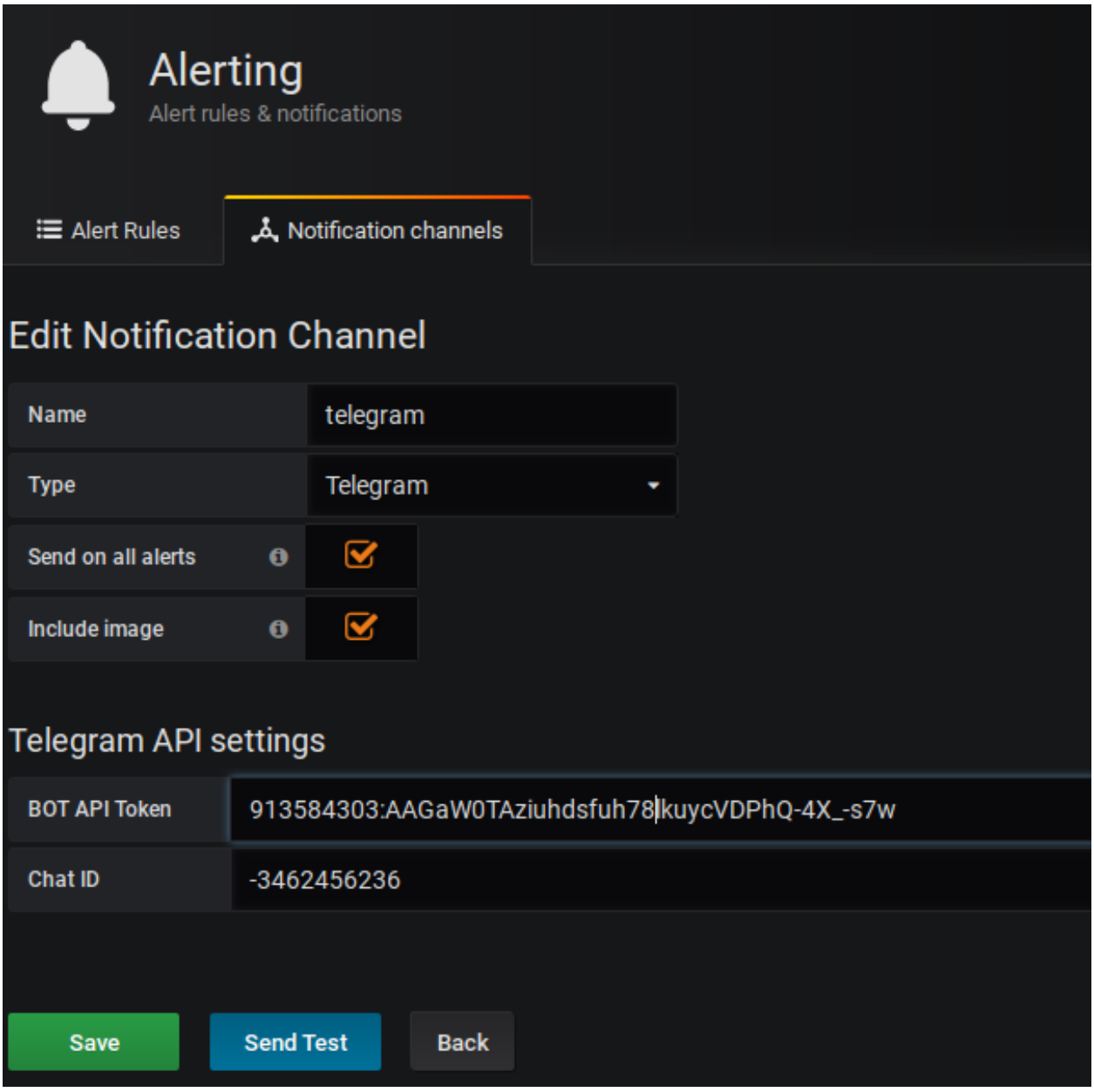
Большинство графиков в Grafana построены на запросах подобного вида (на примере nginx):

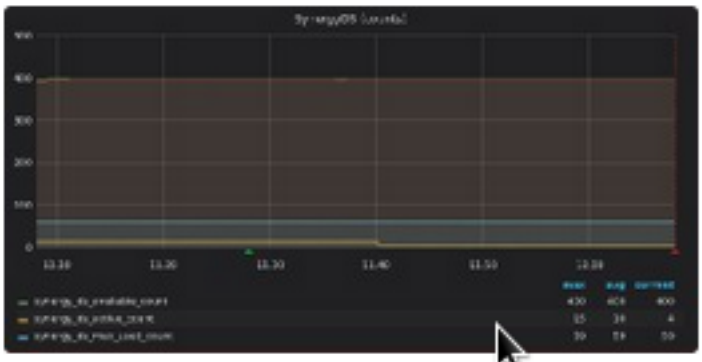
```
sum(rate(synergy_http_response_time_seconds_sum[5m])) by (instance) /
sum(rate(synergy_http_response_time_seconds_count[5m])) by (instance)
```

Суммирование по инстансам позволяет отображать графики нескольких экземпляров какого-либо сервиса одновременно (см. Average Response Times):

Соседний график Requests per Second отображает только данные по выбранному инстансу, так как запрос делается по параметрам `app` и `instance`:

```
sum(rate(synergy_http_response_time_seconds_count{app=~"$job",instance="$instance"}[1m]))
by (instance)
```


Alerting
Alert rules & notifications
Alert Rules
Notification channels
Edit Notification Channel
Name
telegram
Type
Telegram
Send on all alerts
Include image
Telegram API settings
BOT API Token
913584303:AAGaW0TAziuhdsfuh78kuycVDPhQ-4X_-s7w
Chat ID
-3462456236
Save
Send Test
Back



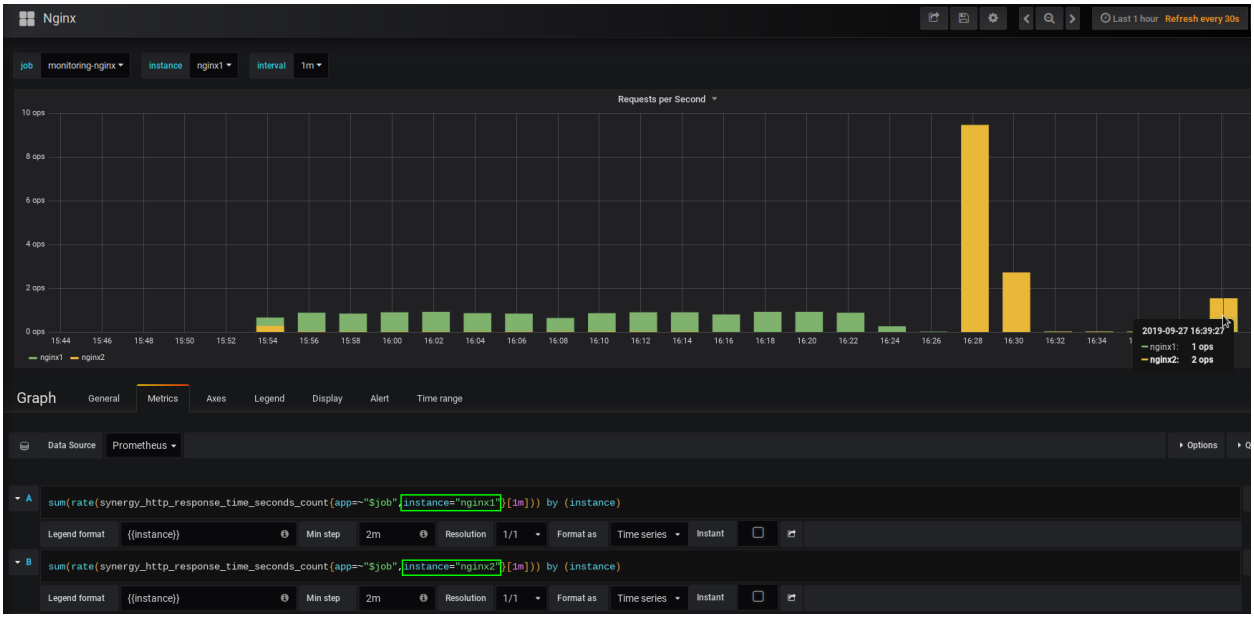
[Alerting] SynergyDS (counts) alert

Message: Число соединений synergyDS ниже установленного порога

Metrics:

synergy_ds_available_count: 61.000

Если нужно показывать на графике оба наблюдаемых сервиса nginx, независимо от выбранного на дашборде, вариантом решения будет дублирование запроса и явное указание инстансов:



1.3 Основные метрики

В разделе описаны основные метрики, используемые на графиках. Полный перечень возможных метрик можно посмотреть в браузере по указанным в предыдущем разделе ссылкам и добавить при необходимости в дополнительные графики.

1.3.1 JBoss

Источники данных

`datasource_pool_available_count` - количество доступных соединений в пуле.

`datasource_pool_active_count` - количество активных соединений; каждое соединение либо использовано приложением, либо доступно в пуле.

`datasource_pool_max_used_count` - максимальное количество использованных соединений.

`datasource_pool_max_creation_time` - максимальное время создания соединения, в миллисекундах.

`datasource_pool_average_creation_time` - среднее время создания соединения, в миллисекундах.

`datasource_pool_average_blocking_time` - среднее время блокирования при получении полной блокировки пула.

Эти метрики на графиках выведены отдельно для БД synergy, storage и jbpmdb.

`datasource_pool_total_creation_time` - общее время создания соединений в миллисекундах.

`datasource_pool_total_blocking_time` - общее время блокирования соединений в миллисекундах.

Транзакции

`transaction_aborted_transactions` - число прерванных транзакций.

`transaction_application_rollbacks` - число транзакций, откаченных назад запросом приложения. Включают в себя и те транзакции, для которых истекло время ожидания.

`transaction_timed_out_transactions` - чисто транзакций, откат которых произошёл из-за таймаута.

`transaction_committed_transactions` - число подтверждённых транзакций.

`transaction_inflight_transactions` - число транзакций, которые начались, но ещё не завершились.

`transaction_resource_rollbacks` - число транзакций, откаченных назад из-за сбоя ресурса.

JVM

`jvm_memory_pool_bytes_used` - использование пула памяти в байтах.

`jvm_memory_pool_bytes_max` - максимум пула памяти в байтах.

`jvm_memory_pool_bytes_committed` - выделенное количество пула памяти в байтах.

`jvm_memory_pool_bytes_init` - исходное количество пула памяти в байтах.

`jvm_memory_bytes_used` - использование выделенной области памяти в байтах.

На графиках показаны процент использованного Old Generation, количество использованной памяти JVM, использование памяти в зависимости от сегмента (heap и non-heap).

`jvm_gc_collection_seconds_count` - количество запущенных GC.

`jvm_gc_collection_seconds_sum` - время, которое GC выполнялись.

`jvm_threads_current` - текущее количество потоков в JVM.

1.3.2 nginx

`synergy_http_response_count_total` - общее количество завершённых HTTP-запросов/ ответов. Кроме графика среднего времени отклика используется для сводки кодов состояния HTTP.

`synergy_http_response_size_bytes` - общее количество переданного контента в байтах.

`synergy_http_response_time_seconds` - сводка всех времён отклика в секундах. На графиках Response Times (quantiles) линиями 0.5, 0.9, 0.99 отмечено время, за которое успевает выполниться соответственно 50, 90, 99 % запросов.

1.3.3 Cassandra

Метрики клиентов

`clientrequest_latency_count` - общее время задержек при выполнении запросов.

`clientrequest_latency_mean` - среднее время задержек при выполнении запросов.

`clientrequest_latency_95thpercentile` - 95-процентная доля задержек при выполнении запросов.

`columnfamily_rangelatency_mean` - задержка сканирования локального диапазона для этой таблицы.

`clientrequest_unavailables_count` - количество обнаруженных исключений из-за недоступности.

`clientrequest_timeouts_count` - количество обнаруженных таймаутов.

`clientrequest_timeouts_mean` - усреднённое количество обнаруженных таймаутов.

Хранилище

`storage_exceptions` - количество выявленных внутренних исключений. При стандартных исключениях значение должно равняться нулю.

`storage_load` - размер данных, которыми управляет данный узел, на диске в байтах.

`storage_totalhints` - количество сообщений с напоминаниями, записанных на этот узел с момента (ре)старта сервера.

`storage_totalhintsinprogress` - количество напоминаний, которое отправляется в данный момент.

Уплотнение (compaction)

Уплотнение - процесс освобождения места путём слияния больших файлов данных. В ходе операции уплотнения файлы SSTable сливаются: производится объединение ключей и соответствующих им столбцов, создание нового индекса. После уплотнения объединённые данные сортируются, над ними строится новый индекс, и только что объединённые, отсортированные и проиндексированные данные записываются в новый файл SSTable.

Ещё одна важная функция уплотнения - повышение производительности путём сокращения числа операций поиска. Для нахождения столбца данных с указанным ключом нужно просмотреть ограниченное количество файлов SSTable. Если этот столбец часто изменяется, то вполне может оказаться, что все версии находятся в сброшенных на диск файлах SSTable. Уплотнение позволяет базе данных не просматривать каждый файл SSTable в поисках указанного ключа и не выбирать из них самое последнее значение каждого столбца.

В процессе уплотнения наблюдается кратковременный всплеск интенсивности ввода-вывода и изменение занятого на диске места - это читаются старые и записываются новые файлы SSTable.

`compaction_pendingtasks` - расчётное количество уплотнений, оставшихся для выполнения.

`compaction_completedtasks` - количество завершённых уплотнений с момента (ре)старта сервера.

`compaction_bytescompacted` - общее число байтов, уплотнённых с момента (ре)старта сервера.

`compaction_totalcompactionscompleted` - пропускная способность выполненных уплотнений с момента (ре)старта сервера.

Фильтр Блума

Фильтры Блума служат для повышения производительности чтения. Это очень быстрый недетерминированный алгоритм, который проверяет, является ли некий объект элементом множества. Недетерминированность связана с тем, что фильтр Блума может давать ложноположительные ответы, но никогда не даёт ложноотрицательных. Принцип работы фильтра Блума заключается в отображении значений элементов множества в битовый массив и в сжатии большого количества данных в строку-дайджест с помощью хэш-функции. Дайджест, по определению, занимает гораздо меньше памяти, чем исходные данные. Фильтр сохраняется в памяти и позволяет повысить производительность, поскольку не каждая операция поиска ключа требует медленного доступа к диску. Таким образом, фильтр Блума является особым видом кэша.

`columnfamily_bloomfilterfalsepositives` - количество ложно-положительных результатов в фильтре таблицы.

`columnfamily_bloomfilterfalseratio` - пропорция ложно-положительных результатов в фильтре таблицы.

`columnfamily_bloomfilterdiskspaceused` - дисковое пространство, занятое фильтром Блума, в байтах.

`columnfamily_bloomfilteroffheapmemoryused` - память вне кучи, использованная фильтром Блума.

Пул потоков

Cassandra разбивает работу определённых типов на собственный пул потоков. Это обеспечивает асинхронность запросов на узле. Состояние потоков важно отслеживать, так как оно показывает, насколько насыщен узел.

`threadpools_completedtasks` - количество завершённых задач.

`threadpools_pendingtasks` - количество задач в очереди.

`threadpools_activetasks` - количество активных задач.

`threadpools_totalblockedtasks` - количество задач, блокированных из-за насыщения очереди.

`threadpools_currentlyblockedtasks` - количество задач, которые блокированы в настоящее время из-за насыщения очереди, но будут разблокированы при повторной попытке.

Кэш

`cache_hitrate` - коэффициент попадания в кэш за всё время.

`cache_size` - общий размер, занятый кэшем, в байтах.

`cache_hits` - общее количество попаданий в кэш.

`cache_requests` - общее количество запросов кэша.

`cache_entries` - общее количество записей в кэше.

Таблицы памяти

`columnfamily_memtablelivedatasize` - общий объем живых данных в таблице памяти, исключая любые заголовки структуры данных.

`columnfamily_memtablecolumnscount` - общее количество столбцов в таблице памяти.

`columnfamily_memtableonheapsize` - общий объем данных в таблице памяти, который находится в куче, включая относящийся к столбцам заголовок и перезаписанные разделы.

`columnfamily_memtableoffheapsize` - общий объем данных в таблице памяти, который находится вне кучи, включая относящийся к столбцам заголовок и перезаписанные разделы.

`columnfamily_memtableswitchcount` - сколько раз сброс данных приводил к выключению.

`columnfamily_livesstablecount` - количество SSTable на диске для данной таблицы.

CQL

`cql_regularstatementsexecuted` - количество выполненных неподготовленных операторов.

`cql_preparedstatementsexecuted` - количество выполненных подготовленных операторов.

1.3.4 Elasticsearch

Системные метрики

`process_cpu_percent` - процент использования CPU процессом Elasticsearch.

`jvm_memory_used_bytes` - текущее использование памяти JVM в байтах.

`jvm_memory_committed_bytes` - зафиксированная память JVM в байтах.

`jvm_memory_max_bytes` - максимальное использование памяти JVM в байтах.

`filesystem_data_available_bytes` - доступное пространство на диске в байтах.

`filesystem_data_size_bytes` - размер диска в байтах.

`transport_tx_size_bytes_total` - общее количество отправленных байтов.

`transport_rx_size_bytes_total` - общее количество полученных байтов.

Документы и операции

`indices_docs` - число документов на данном узле.

`indices_indexing_index_total` - общее число индексных вызовов.

`indices_docs_deleted` - число удалённых документов на данном узле.

`indices_merges_total` - общее число слияний.

`indices_search_query_total` - общее число поисковых запросов.

`indices_search_fetch_total` - общее число выборок.

`indices_refresh_total` - общее число обновлений.

`indices_flush_total` - общее число сбросов.

Время

`indices_search_query_time_seconds` - общее время выполнения поискового запроса в секундах.

`indices_indexing_index_time_seconds_total` - совокупное время индексирования в секундах.

`indices_merges_total_time_seconds_total` - общее время, потраченное на слияние, в секундах.

Кэш

`indices_fielddata_memory_size_bytes` - использование памяти для кэша данных полей в байтах.

`indices_fielddata_evictions` - вытеснение из поля данных.

`indices_query_cache_memory_size_bytes` - использование памяти для кэша запросов в байтах.

`indices_query_cache_evictions` - вытеснение из кэша запросов.

Пул потоков

`thread_pool_rejected_count` - число отклонённых операций.

`thread_pool_active_count` - число активных операций.

`thread_pool_queue_count` - число операций в очереди.

`thread_pool_completed_count` - число завершённых операций.

Garbage Collector

`jvm_gc_collection_seconds_count` - количество запущенных JVM GC.

`jvm_gc_collection_seconds_sum` - время выполнения GC в секундах.

Ссылки и использованные источники:

1. Транзакции JBoss
2. Источники данных JBoss
3. Полный список метрик Cassandra
4. Полный список метрик Elasticsearch
5. Дж. Карпентер, Э. Хьюитт - Cassandra. Полное руководство

1.4 Обработка проблемных ситуаций на основе данных мониторинга SHM

1. **Проблема:** в клиентской части не открываются записи реестра, сама Synergy открывается медленно, подолгу зависая на странице авторизации.

Данные мониторинга: на дашборде датасорсов Jboss значение available count=0,

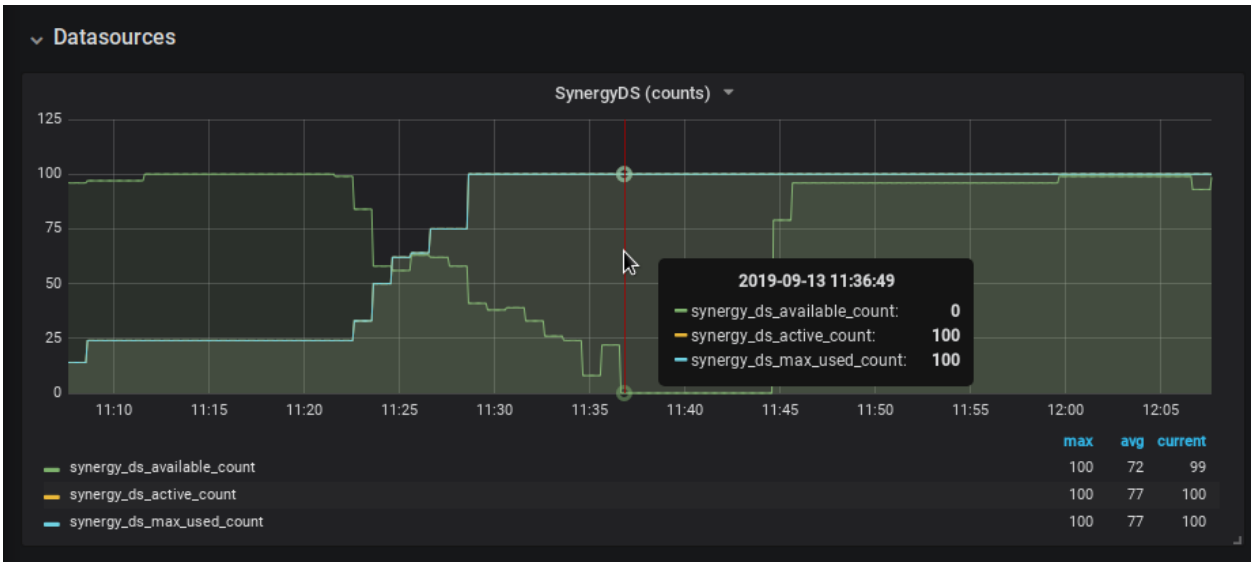


Рис. 5: Рис. 1. График соединений датасорса SynergyDS

на графике времён отклика в то же самое время возрастание времени.

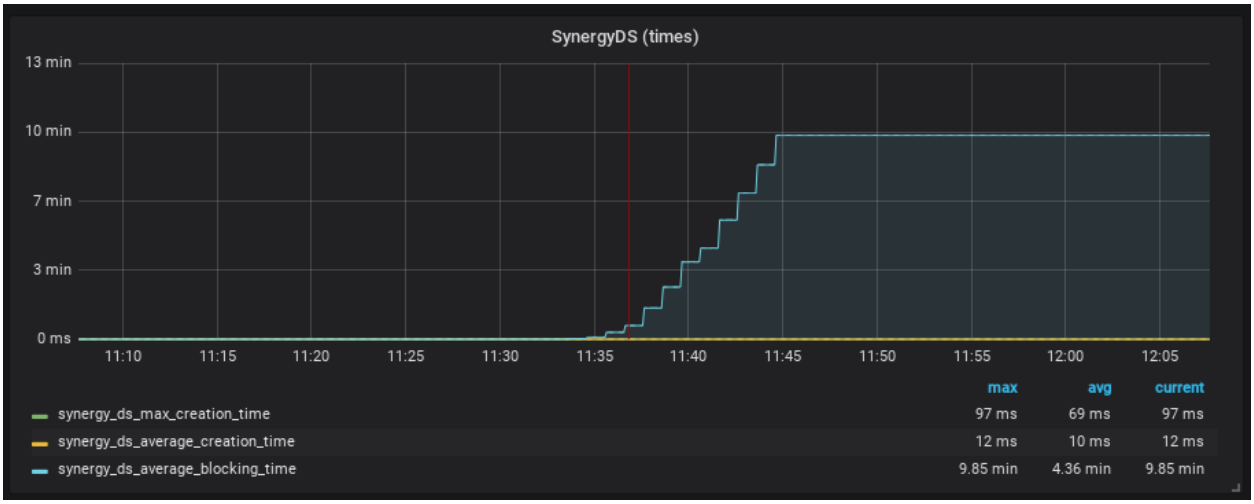


Рис. 6: Рис. 2. График времён отклика датасорса SynergyDS

Ошибки в логе:

```
ERROR [kz.arta.jcr.bd.mysql.SQLConnectionManager] (http-/0.0.0.0:8080-719) javax.resource.
↪ResourceException: IJ000453: Unable to get managed connection for java:/SynergyDS
```

Вероятная причина: закончились соединения в пуле SynergyDS. Так как нет доступных соединений (рис. 1), запросы не могут подключиться к базе, следовательно, время ожидания увеличивается (рис. 2).

Решение: увеличить количество соединений (параметр max-pool-size) для проблемного датасорса synergy в основном конфиге `/opt/synergy/jboss/standalone/configuration/standalone-onesynergy.xml`:

```
<xa-datasource jndi-name="java:/SynergyDS" pool-name="synergy_ds" enabled="true" use-ccm="fals$
     <xa-datasource-property name="URL">
         jdbc:mysql://127.0.0.1:3306/synergy?useUnicode=true&amp;characterEncoding=utf8
     </xa-datasource-property>
     <driver>com.mysql</driver>
     <xa-pool>
         <min-pool-size>20</min-pool-size>
         <max-pool-size></max-pool-size>
         <is-same-rm-override>false</is-same-rm-override>
         <interleaving>false</interleaving>
         <pad-xid>false</pad-xid>
         <wrap-xa-resource>false</wrap-xa-resource>
     </xa-pool>
     . . . . . .
 </xa-datasource>
```