Мессенджер и ВКС

Примеры проблем и их решение

Портал с документацией: https://biz.mail.ru/docs/on-premises/

Оглавление

Назначение документа

В документе рассмотрены потенциальные проблемы, которые не являются специфичными для конкретного сервиса, но могут возникнуть в процессе эксплуатации Мессенджер и ВКС.

Все описанные ниже проблемы можно определить и устранить заранее, пока они не стали мешать функционированию Мессенджер и ВКС. Для этого достаточно ввести мониторинг этих параметров и устранять проблему при возникновении статуса Warning (то есть не доводить до статуса Critical).

Документ предназначен для использования администраторами организации.

Дополнительная документация

Авторизация в системе — в документе рассмотрены проблемы, специфичные для сервисов авторизации (синхронизация пользователей, авторизация, доставка OTP).

Мониторинг Мессенджер и ВКС— в документе приведен перечень параметров инсталляции, которые необходимо контролировать.

Оценить состояние сервисов инсталляции

Все сервисы инсталляции взаимодействуют по протоколу IPROS — бинарный протокол обмена сообщениями между сервисами.

Чтобы найти друг друга, сервисы регистрируются в контроллере (сервис Ctrl). Контроллер хранит информацию о всех сервисах, зарегистрированных в нем, и уведомляет об изменениях в инсталляции.

Чтобы запросить у контроллера информацию о состоянии сервисов инсталляции, используется скрипт `ic srvc`.

Скрипт отображает состояние инстансов всех сервисов, которые зарегистрированы в контроллере в данный момент. Отображает IP-адрес и порт, на котором зарегистрирован сервис, а также его состояние. Возможны три состояния сервиса:

- alive — сервис функционирует. Если сервис не в статусе alive, посмотрите состояние инстансов сервиса при помощи команды `system status <наименование сервиса>`.
- busy — сервис какое-то время не связывался с контроллером (например, из-за большой нагрузки на виртуальную машину), или очередь IPROS-сообщений переполнена.
- dead — сервис не функционирует. Для такого состояния отображается последнее время регистрации сервиса в контроллере.

Информация об инструментах сбора логов клиентских приложений и серверных логов, расположение логов, а также примеры логов клиентских приложений описаны в документации по логам.

Оценить состояние инстансов БД

Для проверки состояния инстансов БД вы можете воспользоваться утилитой `tarantoolctl`.

Для просмотра всех доступных команд утилиты `tarantoolctl` на сервере, где установлено приложение, в командной строке выполните команду `tarantoolctl --help`.

С помощью данной утилиты выможете выполнить:

- запустить инстанс.
- остановить/перезапустить инстанс.
- управлять конфигурацией.

Основные команды управления инстансами:

```
tarantoolctl start name_instance # Запускает инстанс
tarantoolctl stop name_instance # Останавливает работу инстанса
tarantoolctl status name_instance # Показывает текущий статус инстанса
tarantoolctl check name_instance # Проверка файла конфигурации на ошибки
tarantoolctl restart name_instance # Перезапуск инстанса
```

Пример использования утилиты:

1. При входе в Мессенджер выдает ошибку о неправильно введенном email.
2. Проверьте состояние работы инстансов выполнив команду `tarantoolctl status`.
3. В списке статусов у инстанса **nomail-1** статус `stopped` (из-за большой нагрузки на виртуальную машину).
4. Выполните команды:

```
tarantoolctl stop nomail-1
tarantoolctl start nomail-1
```

Блокировка целлов БД KUST

Целл — единица хранения информации, например чат. Блокировка целлов может происходить в следующих случаях:

- При применении мода (единицы изменения для целла).
- При выгрузке целла из диска в оперативную память.

Как понять, что целл заблокирован

В логах записи вида:

```
cell error
```

или

```
lock
```

```
A COMPOT: alert lock
```

Чтобы посмотреть количество заблокированных целлов, подключитесь к командному порту сервиса и выполните команду:

```
stat kust_cell_locked
```

Если во время блокировки целла придет запрос для этого целла, то в логах будут записи:

```
GET fail: cell locked
```

или

```
TXN fail: cell locked
```

Если целл заблокировался при применении мода, в логах будут записи вида:

```
apply fail
```

При ошибке выгрузки целла с мастер-ноды:

```
apply_repair_rcb: Could not restore cell from master
```

При ошибке сравнения версии целла и версии запроса и если запрещено восстановление целла путем выгрузки из мастер-ноды:

```
repair cell with reload from master is turned off. Reload tries will be missed.
```

При выгрузке целла в логах будут записи вида:

```
restore fail
```

Как разблокировать целл

Способ 1. Пометить целлы разблокированными

Разблокировать целл можно только через командный порт сервиса руками. Чтобы подключиться к порту сервиса:

Для сервисов в Kubernetes:

1. Получите IP-адрес экземпляра сервиса:

   ```
   kubectl -n vkteams describe pod $pod_name | grep IP
   ```

   Пример команды для сервиса Gbld-mchat:

   ```
   kubectl -n vkteams describe pod gbld-mchat-a-1-1 | grep IP
   ```

2. Подключитесь по IP-адресу и порту (дефолтный порт — 4020). Пример:

   ```
   rlwrap nc 10.10.10.10 4020
   ```

Если подключение не произошло, проверьте конфигурационный файл сервиса. Чтобы получить конфигурационный файл сервиса, выполните команды:

```
kubectl -n vkteams exec -it gbld-mchat-a-1-1 -- /bin/bash
```

Далее:

```
ps aux | grep gbld-mchat
```

В выводе команды будет конфигурационный файл, порт указан в параметре compot_bind.

Для сервисов, развернутых не в Kubernetes:

1. Найдите процесс нужного экземпляра:

   ```
   ps aux | grep $service_name
   ```

   Пример команды для сервиса Prof-st:

   ```
   ps aux | grep prof-st
   ```

2. Найдите процесс вида:

   ```
   /usr/local/bin/prof-st -c /usr/local/etc/prof-st-1.conf -l 2 -o /oap/icq/logs/prof-st-1.log
   ```

3. Получите адрес командного порта из конфигурационного файла сервиса:

   ```
   grep compot_bind /usr/local/etc/prof-st-1.conf
   ```

4. Пример вывода:

   ```
   compot_bind 127.0.0.1:4281
   ```

5. Подключитесь к командному порту:

   ```
   rlwrap nc 127.0.0.1 4281
   ```

После подключения к командному порту сервиса разблокируйте целлы:

```
kust_cell_unlock ignore
```

Команда применяется сразу ко всем заблокированным целлам и помечает их как разблокированные, отдельно к одному целлу применить нельзя.

Если способ 1 не помог, нужно понять, на какой ноде заблокированы целлы. Если на реплике — перейдите в к способу 2, если на мастере — к способу 3.

Способ 2. Если целлы заблокированы на реплике

Этот способ удаляет все данные на реплике и перезаписывает данными из мастер-ноды.

Для сервисов в Kubernetes

1. Удалите PVC сервиса командой:

   ```
   kubectl delete pvc $pvc_name -n vkteams
   ```

После этого реплика будет перезаписана с мастера.

2. Выполните команду удаления пода, иначе удаление PVC будет висеть бесконечно, так как под его держит:

```
kubectl delete pod $pod_name -n vkteams
```

Для сервисов из Systemctl

Переключитесь на супер-пользователя и выполните инструкцию из раздела У сервиса типа KUST реплика не получает данных из мастера

Способ 3. Если целлы заблокированы на мастер-ноде

1. В командном порте реплики проверьте, что на реплике нет заблокированных целлов:

```
stat kust_cell_locked
```

2. Подключитесь к командному порту сервиса, как описано в способе 1, и посмотрите статистику:

```
stat
```

Если в выводе записи вида:

```
kust_1_stat:        1241 (1 lsn/min, 0.00 MB/s) [realtime]
```

это означает, что можно переключить роли между мастером и репликой.

3. Подключитесь к командному порту контроллера и получите карту необходимого сервиса. Пример команды для сервиса Gbld-mchat:

```
map gbld-mchat
```

4. Найдите необходимую пару матера и реплики проблемного инстанса.
5. Переключитесь на реплику, при этом передайте имя инстанса, которое было указано в карте:

```
flip $slave_node
```

6. Если переключение прошло успешно, перезапишите данные с мастер-ноды на реплику, как описано в способе 2.

Недостаточно места на диске

Частота ротации лог-файлов и время хранения подобраны так, чтобы места на диске всегда хватало. Тем не менее возможны ситуации, при которых место на диске может заканчиваться.

Примеры:

- Инсталляция используется для большего числа пользователей, чем предполагалось.
- Один из сервисов слишком активно создает сообщения об ошибках. Например, Keycloak создает большие лог-файлы в случае дубликатов пользователей.

Как решить проблему

Найдите источник проблемы, освободите место и сообщите о проблеме разработчику:

Шаг 1. Проверьте место на диске, выполнив команду `df -h.`

Шаг 2. Выясните, чем занято место:

```
du -smx /* 2>/dev/null| sort -rn | head
```

Ключ `-x` обязателен.

Если обнаружен каталог, который сильно превышает допустимые размеры, то выполните команду `du` в этом каталоге и найдите проблемное место. В дистрибутив включена утилита ncdu, которой удобно осуществлять поиск занятого места:

```
>ncdu / // искать по всем fs
>ncdu -x / // искать только по корневой fs и не тратить время на расчет занятого места на
data-дисках
```

Шаг 3. Соберите информацию о проблеме и передайте ее разработчикам:

```
ls -alh /<путь к каталогу с большим лог-файлом>/
head -n 1000 <путь к большому лог-файлу>
tail -n 1000 <путь к большому лог-файлу>
```

Шаг 4. Удалите старые лог-файлы и выполните команду `truncate` по текущему, если он большого размера.

Возможна ситуация, когда команды `df` и `du` выдают разные результаты. Например, команда `df` показывает, что занято 50 Гб, а команда `du` показывает только 10 Гб. Это значит, что в файловой системе есть удаленные и незакрытые файлы. Найти такие файлы можно, выполнив:

```
lsof +L1 | sort -rn -k 7 | head
```

Особенности удаленных файлов

В активной системе такие файлы есть практически всегда, и нужно различать, когда это является проблемой, а когда нет. В большинстве случаев проблему с нехваткой места создают файлы журналов, которые не были переоткрыты сервисом. При этом важно понимать, насколько старый лог остался незакрытым.

Пример

Логи не были переоткрыты сервисом Nginx-im:

```
nginx 729949 quantum 115w REG 252,1 17601 608228
/mnt/log/oap/icq/domains/local_proxy.icq.com/logs/old_logs/u-error.log-20200410-1800.rot
(deleted)

nginx 729949 quantum 150w REG 252,1 398647142 608229
/mnt/log/oap/icq/domains/local_proxy.icq.com/logs/old_logs/u-access.log-20200410-1800.rot
(deleted)
```

Если с 18:00 (время ротации исходя из имени файла) прошло менее двух часов, то никаких действия не требуется. Nginx обрабатывает длинные запросы (long polling), поэтому после отправки ему команды `reopen` еще долгое время остаются старые дочерние процессы, которые продолжают держать этот файл. Если же с 18:00 прошло много времени, нужно отправить команду `reopen`:

```
/usr/local/nginx-im/sbin/nginx -c /usr/local/nginx-im/conf/nginx.conf -s reopen
```

Если лог-файлы остались в статусе `deleted`, то необходимо проверить правильность конфигурационных файлов Nginx. Вероятно, в них есть синтаксическая ошибка:

```
/usr/local/nginx-im/sbin/nginx -c /usr/local/nginx-im/conf/nginx.conf -t
```

Заканчивается память

Излишнее потребление памяти может происходить по разным причинам.

Примеры:

- Утечка в ПО.
- Некорректные пропорции выделенного RAM и выставленных лимитов в приложениях (наиболее вероятный вариант). Разные клиенты используют разное количество аккаунтов, и профиль нагрузки заранее неизвестен, поэтому невозможно предугадать, насколько корректны выставленные настройки и лимиты.
- Непрогнозируемое повышение нагрузки. Обычно повышение нагрузки связано с потреблением CPU, но расход памяти в этом случае может тоже увеличиваться. Также может не хватать ресурсов CPU для работы сборщиков мусора.

Как решить проблему

Шаг 1. Выполните `free -m`.

Шаг 2. Изучите параметры free, available, buff, cache.

Free — это неиспользуемая память.

Available — это память, которую система может освободить при необходимости, в том числе туда входят кэш и буфер. Если же память available составляет менее 20-30% от общей, то это может быть проблемой, так как системе для быстрой работы необходимы буфер и кэш.

Шаг 3. Изучите список процессов, которые потребляют память больше всего. Можно использовать любой из способов:

- `top -o RES`
- `smem -s rss -p`
- `ps -ax -o pid,rss,command --sort rss`

Шаг 4. Для временного устранения проблемы перезапустите процессы, которые занимают наибольшее количество памяти. Однако нельзя гарантировать, что проблема не повторится, поэтому обязательно пришлите разработчику следующую информацию:

- вывод `smem -s rss -p`
- вывод `top -o RES -n 1`
- вывод `free -m`

После изучения этих данных мы предложим варианты для предотвращения подобной ситуации в дальнейшем.

Пример работы с журналами

В качестве примера рассматривается отправка сообщения. Для отправки сообщения выполняется запрос `POST /wim/im/sendIM.`

Пример

```
14/151406 COMPAT0: sajp_wrapper.c:1539 SAJP_SendResponse >>10.10.10.10 POST /wim/im/sendIM
"aimsid=001.3656109597.XXXXXXXXXX%3Atester%40example.com&f=json&message=XXXXXXX&notifyDelivery
=true&offlineIM=1&r=5b3e43e5-19ac-4f69-93b8-
f56d4de3dfd9-15868662101408592&t=exampler%40example.com&updateMsgId=6815539145092366384" 200
"200[0] Ok" 0.009s [onpremise:50:38975693:1586866446.257] "myteam Desktop tester@example.com
on2fah4R-win 10.0.0(2132) Windows_10 PC"
```

В примере приведена отправка сообщения от пользователя user1 к пользователю user2. `sendIM` означает, что пользователь A отправил сообщение пользователю B, безотносительно факта получения сообщения. Для полноценной проверки потребуются дополнительные журналы, например журнал сервиса bos_srv, который отправляет сообщения от пользователя к пользователю.

Сервис bos_srv имеет несколько экземпляров приложения (инстансов), у каждого из них свой набор журналов, поэтому ниже используются подстановочные знаки (wildcards):

- **/oap/icq/logs/bos_srv-*.err.log** — лог ошибок. Кроме ошибок в этот лог попадает большое количество дополнительной информации, поэтому его неудобно использовать для поиска отправлений и получений.
- **/oap/icq/logs/bos_srv-*.access.log** — лог обращений клиента (аналог лога доступа в Nginx).
- **/oap/icq/logs/bos_srv-*.fss.log** — файл отражает весь путь сообщений. Он позволяет понять, было ли отправлено и получено сообщение.

Проверяем факт отправки сообщения:

```
grep '|IM|' /oap/icq/logs/bos_srv-*.fss.log
2020-04-14 00:05:38|10.10.10.10|user1@vkteams.example.com|user2@vkteams.example.com|->|myteam
Desktop user1@vkteams.example.com on2fah4R-mac 5.0.0(2134) MacOSX_10.15 PC|IM|-|
aimsid=001.0805324701.XXXXXXXXXX:user1@vkteams.example.com,devId=on2fah4R-
mac,mmb=50;0;2134;1999,device=XXXXXXXXXX,msgId=86a7345a-7dca-11ea-952d-52540098da02,pin=0,tele
chat=0,ohtype=,friendship=1;1,histId=6815305378612379774,bot_detected=0,is_pymk=0,updMsgId=0,p
hone=00000000000,suspicious=1;0,emoji_txt=0,white=0,eq_num_phone=0,bot_ignored=0,mrasd_relaxed
=0
```

Для одного сообщения таких записей будет две, так как запись происходит как на сервисе bos_srv отправителя, так и на сервисе bos_srv получателя. Направление отправки зависит от указателя после второго адреса почты. В данном примере `->` означает отправку от user1 к user2. Отправка от user2 к user1 будет обозначена как `<-`.

Проверяем, что приложение клиента забрало сообщение (тип записи `|HIST|`):

```
2020-04-14 00:00:36|46.73.143.144|user1@vkteams.example.com|user2@vkteams.example.com|<-|
myteam Desktop user1@vkteams.example.comon2fah4R-mac 5.0.0(2796) MacOSX_10.15 PC|HIST|-|
aimsid=001.1947658530.XXXXXXXXXX:aaa@vkteams.example.com,devId=on2fah4R-
mac,mmb=50;0;2796;1999,device=XXXXXXXXX,histId=6815180970589684273,phone=00000000000,stranger=
0
```

Из примера видно, что приложение клиента user1 забрало сообщение, которое было отправлено от user2@vkteams.example.com (указатель направления в данной строчке `<-`).

Не отправляются сообщения в чаты/группы

Чтобы узнать о проблемах, используйте систему мониторинга согласно документации — она будет мгновенно присылать алерты в случае обнаружения проблем. Ниже описан пример алерта при недостатке памяти для сохранения сообщений и шаги для восстановления хранилища.

Алерт:

```
1 snmpexec_mon_gbld_mchat_1_status - - - close_to_overmem (first: 2025-09-05/04:27:11, last:
2025-09-05/04:27:11, total: 1)
```

Причина: у сервисов типа KUST для хранения сообщений заканчивается память. Для чатов и каналов используется сервис Gbld-mchat. Для личных сообщений используется сервис Gbld-st.

Ниже рассмотрим, как устранить проблему на примере сервиса Gbld-mchat. Для сервиса Gbld-st все абсолютно идентично, нужно заметить имя сервиса на Gbld-st и повторить аналогичные действия.

Шаг 1. Посмотрите карту сервиса

Выполните команду:

```
$ ic map gbld-mchat
```

Шаг 2. Повысьте лимит у работающего сервиса

Есть два варианта работы сервиса — сервис как служба или в Kubernetes. В зависимости от этого по-разному определяются командный порт сервиса (command_port) и нужный IP-адрес (ip_addr).

Если команда

```
sudo systemctl status gbld-mchat-1
```

даст вывод, как на примере ниже, то сервис поднят как служба:

```
  gbld-mchat-1.service - IPROS service gbld-mchat-1
   Loaded: loaded (/etc/systemd/system/gbld-mchat-1.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2025-08-22 16:59:35 MSK; 2 weeks 6 days ago
  Process: 243651 ExecStopPost=/bin/bash -c /bin/echo STOP $CMD >> $NOTIFY_TO_LOG
(code=exited, status=0/SUCCESS)
  Process: 243656 ExecStartPost=/bin/bash -c /bin/echo START $CMD >> $NOTIFY_TO_LOG
(code=exited, status=0/SUCCESS)
 Main PID: 243655 (im-start.sh)
    Tasks: 14
   Memory: 229.1M
   CGroup: /system.slice/gbld-mchat-1.service
           243655 /bin/bash /usr/local/bin/im-start.sh gbld-mchat-1 /tmp/service-
notify.txt /usr/local/bin/gbld-mchat -c /usr/local/etc/gbld-mchat-1.con...
           243659 /usr/local/bin/gbld-mchat -c /usr/local/etc/gbld-mchat-1.conf -l 2 -o /
oap/icq/logs/gbld-mchat-1.log
```

Если команда

```
sudo kubectl get pods -n vkteams | grep gbld-mchat
```

даст вывод, как на примере ниже, то сервис поднят в Kubernetes:

```
gbld-mchat-a-1-1                              2/2     Running   2 (4d ago)      4d
```

1. Получите командный порт сервиса (command_port) и нужный IP-адрес (ip_addr).

   Если сервис поднят как служба

   IP-адрес всегда — localhost (127.0.0.1)

   Чтобы определить список командных портов сервиса (command_port) для всех инстансов, выполните команду ниже для каждого инстанса поднятой службы:

   ```
   $ sudo grep compot_ /usr/local/etc/gbld-mchat-<номер инстанса>.conf
   ```

   Пример:

   ```
   $ sudo grep compot_ /usr/local/etc/gbld-mchat-1.conf
   compot_bind 127.0.0.1:4061
   ```

   Обычно для сервиса Gbld-mchat используется порт 4061, а для Gbld-st порт 4021. Тем не менее, выбирайте порт, полученный именно из команды выше.

   Если сервис поднят в Kubernetes

command_port чаще всего — 4020. Вы можете проверить командный порт при помощи команды:

```
$ sudo grep compot_bind /usr/local/etc/k8s/helmwave/projects/gbld-mchat/values/gbld-mchat/
application.yml
    compot_bind: 0.0.0.0:4020
$ sudo grep compot_bind /usr/local/etc/k8s/helmwave/projects/gbld-mchat/values/gbld-mchat/
application.yml
    compot_bind: 0.0.0.0:4020
```

Чтобы определить список IP-адресов сервиса (ip_addr) для всех инстансов, выполните команду ниже для каждого пода поднятой службы:

```
$ sudo kubectl get pods -n vkteams -l app=gbld-mchat -o jsonpath='{range .items[*]}
{.status.podIP}{"\n"}{end}'
```

2. Получите актуальное значение kust_mem_limit, используя command_port и ip_addr из пункта выше. Обозначим его как old_limit.

   Если сервис поднят как служба, выполните команду ниже для всех командных портов, полученных на шаге 1.

   Если в Kubernetes — для всех IP-адресов, полученных на шаге 1.

   ```
   $ echo get kust_mem_limit | nc <ip_addr> <command_port>
   ```

   Пример применения для сервиса Gbld-mchat в Kubernetes и демонстрационного IP-адреса (у вас он будет другой, нужно подставить полученный IP-адрес из пункта 1):

   ```
   $ echo get kust_mem_limit | nc 10.32.128.192 4020
   2147483648
   ```

3. Полученное ранее значение old_value, умножьте на 2. Обозначим это как new_limit. Далее выполните команду:

   ```
   $ echo set kust_mem_limit <new_limit> | nc <ip_addr> <command_port>
   ```

   Пример применения для сервиса Gbld-mchat, поднятого как служба:

   ```
   $ echo get kust_mem_limit | nc localhost 4061
   ```

4. Проверьте, что выставленное значение применилось:

   ```
   $ echo get kust_mem_limit | nc <ip_addr> <command_port>
   ```

Шаг 3. Выставите повышенный лимит на будущее

Новый лимит будет подхвачен при рестарте сервисов. Для этого укажите новый лимит в конфигурационном файле.

Если сервис работает как служба

1. Сделайте копию текущей конфигурации:

```
$ cp -v /usr/local/etc/gbld-mchat-<номер инстанса>.conf /usr/local/etc/gbld-mchat-<номер инстанса>.conf.$(date +%s)
```

   Пример:

```
$ cp -v /usr/local/etc/gbld-mchat-1.conf /usr/local/etc/gbld-mchat-1.conf.$(date +%s)
```

2. Замените старое значение old_limit, полученное в шаге 2 (пункт 2), на new_limit из шага 2 (пункт 3):

```
$ sed -i 's/kust_mem_limit <old_limit>/kust_mem_limit <new_limit>/1' /usr/local/etc/gbld-mchat-<номер инстанса>.conf
```

   Пример, где значение лимита было 2147483648 и выставлен такой же лимит 2147483648:

```
$ sed -i 's/kust_mem_limit 2147483648/kust_mem_limit 2147483648/1' /usr/local/etc/gbld-mchat-1.conf
```

3. Проверьте, что изменения применились:

```
$ grep kust_mem_limit /usr/local/etc/gbld-mchat-1.conf
```

   В выводе команды должно быть новое значение лимита.

Если сервис работает в Kubernetes

1. Сделайте копию текущей конфигурации:

```
$ cp -v /usr/local/etc/k8s/helmwave/store/kust.yml /usr/local/etc/k8s/helmwave/store/kust.yml.$(date +%s)
```

2. Замените старое значение old_limit, полученное в шаге 2 (пункт 2), на new_limit также из шага 2(пункт 3).

   Для сервиса Gbld-mchat:

```
$ sed -i 's/gbld_mchat: <old_limit>/gbld_mchat: <new_limit>/1' /usr/local/etc/k8s/helmwave/store/kust.yml
```

   Для сервиса Gbld-st:

```
$ sed -i 's/gbld_st: <old_limit>/gbld_st: <new_limit>/1' /usr/local/etc/k8s/helmwave/store/kust.yml
```

3. Примените настройки.

   Для инсталляции на одну виртуальную машину:

```
$ sudo su
$ hwup
$ exit
```

Далее сделайте рестарт подов сервиса:

```
$ sudo kubectl delete pod -n vkteams -l app=gbld-mchat
```

Для распределенной инсталляции выполните команду на второй виртуальной машине (пара 1, сторона b):

```
$ im_deployer --helmwave --update --hw-once --hw-project gbld-mchat
```

Посмотрите текущую топологию сервиса:

```
$ ic map gbld-mchat
```

Пример вывода команды:

```
node   gbld-mchat-1     (2147483648   buckets)
    srv   gbld-mchat.vkt-4vm-1618-standart-01.gbld-mchat-1      10.32.0.64:2500   alive
main
    srv   gbld-mchat.vkt-4vm-1618-standart-02.gbld-mchat-1     10.32.48.77:2500   alive
slave
node   gbld-mchat-2     (2147483648   buckets)
    srv   gbld-mchat.vkt-4vm-1618-standart-03.gbld-mchat-1     10.32.64.76:2500   alive
main
    srv   gbld-mchat.vkt-4vm-1618-standart-04.gbld-mchat-1     10.32.96.71:2500   alive
slave
```

В данном примере мастер-ноды на стороне a, реплики на стороне b. Перезапустите реплики:

```
$ im_pod_cleaner delete -n vkteams --pod-label app=gbld-mchat --node-label im/topology-
side=b
```

Переключите нагрузку на реплику:

```
$ ic flip gbld-mchat.vkt-4vm-1618-standart-02.gbld-mchat-1
$ ic flip gbld-mchat.vkt-4vm-1618-standart-04.gbld-mchat-1
```

Проверьте топологию сервиса:

```
$ ic map gbld-mchat
```

Пример вывода команды:

```
node   gbld-mchat-1     (2147483648   buckets)
    srv   gbld-mchat.vkt-4vm-1618-standart-01.gbld-mchat-1      10.32.0.64:2500   alive
slave
    srv   gbld-mchat.vkt-4vm-1618-standart-02.gbld-mchat-1     10.32.48.24:2500   alive
main
node   gbld-mchat-2     (2147483648   buckets)
    srv   gbld-mchat.vkt-4vm-1618-standart-03.gbld-mchat-1     10.32.64.76:2500   alive
```

```
slave
    srv   gbld-mchat.vkt-4vm-1618-standart-04.gbld-mchat-1     10.32.96.55:2501   alive
main
```

Перезапустите бывшие мастер-ноды:

```
$ im_pod_cleaner delete -n vkteams --pod-label app=gbld-mchat --node-label im/topology-
side=a
```

Несоответствие логина пользователя и адреса сервера (API endpoints)

Когда пользователь вводит логин вида username@vkteams.EXAMPLE.com, приложение обращается по следующим адресам для получения конечных точек (эндпоинтов) API:

- https://u.vkteams.EXAMPLE.com/myteam-config.json
- https://vkteams.EXAMPLE.com/myteam-config.json

Файл **myteam-config.json** генерируется автоматически скриптом конфигурации сервера Мессенджер и ВКС и не требует ручного редактирования.

Если сервер Мессенджер и ВКС установлен, например, по адресу vkteams.EXAMPLE.com, а адрес пользователя — username@EXAMPLE.com, приложение не сможет найти конфигурационный файл по адресам:

- https://u.EXAMPLE.com/myteam-config.json
- https://EXAMPLE.com/myteam-config.json

В этом случае приложение отобразит пользователю диалоговое окно с предложением ввести адрес сервера, на котором установлен Мессенджер и ВКС. Чтобы пользователям не пришлось выполнять дополнительные действия, настройте редирект:

```
https://EXAMPLE.com/myteam-config.json -> https://u.vkteams.EXAMPLE.com/myteam-config.json
```

Для корректной работы браузерной версии (WebIM) необходимо к редиректу добавить CORS:

```
access-control-allow-credentials: true
access-control-allow-origin: https://webim.vkteams.EXAMPLE.com
```

Если настройка редиректа невозможна (например, для гостевых учетных записей с внешним доменом), пользователь может:

- указать в диалоговом окне адрес сервера (vkteams.EXAMPLE.com для примера выше);
- сразу ввести логин с адресом сервера через знак #:

```
username@example.com#vkteams.EXAMPLE.com
```

Тогда приложение будет сразу обращаться по адресам:

- https://u.vkteams.EXAMPLE.com/myteam-config.json
- https://vkteams.EXAMPLE.com/myteam-config.json

Массовые проблемы с клиентским приложением

Протокол HTTPS (443/TCP) — основа коммуникации «клиент — сервер», поэтому при любых массовых проблемах с приложением необходимо проверять доступность по HTTPS. Все запросы от клиента (за исключением звонков) принимает сервис Nginx-im (сборка Nginx с дополнительными модулями, необходимыми для функционирования проекта).

Управление:

```
sudo /usr/local/bin/nginx.sh
```

Расположение конфигурационных файлов:

- /usr/local/nginx-im/confv/2nginx.conf — основной файл;
- /usr/local/nginx-im/confv2/conf.d/ — настройки отдельных виртуальных хостов.

Расположение журналов /oap/icq/domains/local_proxy.icq.com/logs/:

- *error.log — журналы ошибок отдельных виртуальных хостов;
- *access.log — журналы доступа отдельных виртуальных хостов.

В лог-файлах проверяйте наличие запросов и статусы HTTP-ответов. Основной точкой входа для запросов пользователя является `u.<user_domain>`, поэтому наиболее важными журналами являются:

- /oap/icq/domains/local_proxy.icq.com/logs/u-access.log
- /oap/icq/domains/local_proxy.icq.com/logs/u-error.log

Ротация лог-файлов происходит каждый час. Предыдущие журналы доступны в каталоге **/oap/icq/domains/local_proxy.icq.com/logs/old_logs/** и сжаты с использованием gzip (по умолчанию), lz4 или zstd.

Примечание

Файлы с расширением .zst необходимо открывать при помощи команды `zstdcat` и искать в них при помощи `zstdgrep`. Например: zstdgrep user@vkteams.example.com/oap/icq/domains/local_proxy.icq.com/logs/old_logs/u-access.log-20200410-1900.rot.zst

Проверка конфигурационных файлов на ошибки:

```
/usr/local/nginx-im/sbin/nginx -tc /usr/local/nginx-im/conf/nginx.conf
```

SSL-сертификаты:

- /usr/local/etc/im_ssl/default_ssl.cert
- /usr/local/etc/im_ssl/default_ssl.key

Не доходят пуш-сообщения

При получении сообщения на устройство пользователя отправляется пуш-уведомление. В зависимости от платформы уведомление отправляется либо на серверы Apple, либо на серверы Google. Журнал пуш-сервиса: **/var/log/mru-пуш-me/gosender.log.** Ошибки логируются с пометкой ERROR (или пометкой более высокого уровня), стандартные сообщения — с пометкой INFO.

Пример 1

```
2020-04-14 16:56:08:INFO:5852:+ios:icq aimsid=XXX.XXXXXXXX.topsecret:user@vkteams.example.com;
bundle_id=ru.mail.myteam-onpremise; hist_msg_id=0; token=XXXXXXXXXXXXXXXXXX
```

Отправлено пуш-сообщение для пользователя user, платформа iOS.

Пример 2

```
2020-03-22 10:45:27:FATAL:1098524:error create queueCluster nodes: [пушer.intim]; queues:
[127.0.0.1:3321]
```

Недоступна БД Tarantool с очередью на отправку.

2 марта 2026 г.