

# Первичный Траблшут в KUMA (Troubleshoot)

## Проверка статуса основных КОМПОНЕНТОВ

Основные службы KUMA:

```
systemctl status kuma-collector-ID_СЕРВИСА.service
systemctl status kuma-correlator-ID_СЕРВИСА.service
systemctl status kuma-storage-ID_СЕРВИСА.service
systemctl status kuma-core.service
systemctl status kuma-clickhouse.service
systemctl status kuma-mongodb.service
systemctl status kuma-victoria-metrics.service
systemctl status kuma-grafana.service
```

В версии KUMA 3.2 сервис ядра изменил название на **kuma-core-00000000-0000-0000-0000-000000000000.service**

ID\_СЕРВИСА можно скопировать в веб-интерфейсе системы, в разделе Активные сервисы, выделив в checkbox нужный сервис, затем нажав кнопку скопировать ID.

[Resources and services](#) > **Services** Add service Refresh

Reload Restart Copy ID Go to events Go to active lists Go to partitions

<input type="checkbox"/>	Status	Kind ↑	Service	Version	Tenant
<input checked="" type="checkbox"/>	●	Collector	<a href="#">[Example] CEF</a>	1.5.0.318	Main
<input type="checkbox"/>	●	Collector	<a href="#">[Example] KSC</a>	1.5.0.318	Main

# Проверка журнала ошибок КУМА

```
/opt/kaspersky/kuma/<Компонент>/<ID_компонента>/log/<Компонент>
```

Пример:

```
/opt/kaspersky/kuma/storage/<ID хранилища>/log/storage
```

Для неактуальных версий (2.0 и ниже):

- `journalctl -xe`
- Журнал ошибок Click-House: `/opt/kaspersky/kuma/clickhouse/logs/clickhouse-server.err.log`

## Вывод ошибок сервиса в консоль

В случае отсутствия информации в журналах целесообразно вывести информацию в консоль. Проверяем статус сервиса и копируем его параметры запуска:

```
[root@test-kuma ~]# systemctl status kuma-collector-cef0527c-25ad-4490-a8ce-bf9ab2af71ee.service
● kuma-collector-cef0527c-25ad-4490-a8ce-bf9ab2af71ee.service - KUMA collector
   Loaded: loaded (/usr/lib/systemd/system/kuma-collector-cef0527c-25ad-4490-a8ce-bf9ab2af71ee.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-10-21 11:04:32 MSK; 4s ago
     Process: 177169 ExecStartPre=/usr/bin/chown -R kuma:kuma /opt/kaspersky/kuma/collector/cef0527c-25ad-4490-a8ce-bf9ab2af71ee (code=exited, status=0/SUCCESS)
     Process: 177167 ExecStartPre=/usr/bin/chown kuma:kuma /opt/kaspersky/kuma/collector (code=exited, status=0/SUCCESS)
    Main PID: 177172 (kuma)
      Memory: 617.3M
   CGroup: /system.slice/kuma-collector-cef0527c-25ad-4490-a8ce-bf9ab2af71ee.service
           └─177172 /opt/kaspersky/kuma/kuma collector --id cef0527c-25ad-4490-a8ce-bf9ab2af71ee --core https://test-kuma.sales.lab:7210 --api.port 7276

Oct 21 11:04:27 test-kuma.sales.lab systemd[1]: kuma-collector-cef0527c-25ad-4490-a8ce-bf9ab2af71ee.service: Service RestartSec=5s expired, scheduling restart.
Oct 21 11:04:27 test-kuma.sales.lab systemd[1]: kuma-collector-cef0527c-25ad-4490-a8ce-bf9ab2af71ee.service: Scheduled restart job, restart counter is at 4.
Oct 21 11:04:27 test-kuma.sales.lab systemd[1]: Stopped KUMA collector.
Oct 21 11:04:27 test-kuma.sales.lab systemd[1]: Starting KUMA collector ...
Oct 21 11:04:30 test-kuma.sales.lab kuma[177172]: [destination '[Example] Correlator'] 2022/10/21 08:04:30 /opt/tfs-agent/_work/1/s/sdk/destinations/base.go:65: started
Oct 21 11:04:31 test-kuma.sales.lab kuma[177172]: [destination '[Example] Storage'] 2022/10/21 08:04:31 /opt/tfs-agent/_work/1/s/sdk/destinations/base.go:65: started
Oct 21 11:04:32 test-kuma.sales.lab kuma[177172]: [connector 'db137535-97eb-468a-9c0a-350600d4aa1f'] 2022/10/21 08:04:32 /opt/tfs-agent/_work/1/s/sdk/connectors/base.go:36: started
Oct 21 11:04:32 test-kuma.sales.lab kuma[177172]: [collector cef0527c-25ad-4490-a8ce-bf9ab2af71ee] 2022/10/21 08:04:32 /opt/tfs-agent/_work/1/s/collector/processor.go:85: started
Oct 21 11:04:32 test-kuma.sales.lab systemd[1]: Started KUMA collector.
```

**Останавливаем сервис**, переходим в папку `cd /opt/kaspersky/kuma/` и запускаем следующим образом, пример команды ниже:

```
sudo -u kuma /opt/kaspersky/kuma/kuma collector --id cef0527c-25ad-4490-a8ce-bf9ab2af71ee --core
https://test-kuma.sales.lab:7210 --api.port 7276
```

## Пустые метрики

В случае если раздел метрик пустой (отсутствуют значения на дашбордах), то проверьте указан ли IP и hostname сервера КУМА в файле `/etc/hosts`, если нет то добавьте.

Перезапустите службы:

```
systemctl restart kuma-victoria-metrics.service  
systemctl restart kuma-grafana.service
```

Либо присутствует конфликт со службой Cockpit на Oracle Linux.

Она слушает тот же порт 9090, что и Victoria Metrics. Останови Cockpit и посмотри, поднимутся ли метрики.

Либо проблема из-за наличия прокси между ядром и АРМом, с которого к вебке подключались.

---

## Проверка прослушивания порта, например, 5144

```
netstat -antplu | grep 5144
```

---

## Работа с портами на МЭ (firewall-cmd)

Проверка открытых портов на МЭ:

```
firewall-cmd --list-ports
```

Добавление порта 7210 на МЭ:

```
firewall-cmd --add-port=7210/tcp --permanent
```

Применение настроек:

```
firewall-cmd --reload
```

---

## Проверка получения событий на порту 5144 в ASCII

```
tcpdump -i any port 5144 -A
```

---

# Отправка тестового события на порт 5144, для проверки работы коллектора

Для TCP:

```
nc 127.0.0.1 5144 <<< 'CEF:0|TESTVENDOR|TESTPRODUCT|1.1.1.1|TEST_EVENT|This is a test event|Low|message="just a test coming through"'
```

Для UDP:

```
nc -u 127.0.0.1 5144 <<< 'CEF:0|TESTVENDOR|TESTPRODUCT|1.1.1.1|TEST_EVENT|This is a test event|Low|message="just a test coming through"'
```

В случае наличия в сыром событии двух типов кавычек ' и ", то целесообразно отправлять событие из файла (для этого тестовое событие запишите в файл 1 строку).

```
cat test.txt | nc -u 127.0.0.1 5144
```

Для HTTP:

```
curl -X POST -d "Your message here" http://localhost:<port>/path
```

Если многострочное событие в файле с разделителем \0:

```
truncate -s +1 sample  
curl http://<ip/host>:<port>/input --data-binary "@/root/sample"
```

---

## Статус службы красный / Ошибка на компоненте

Переписать / перепроверить учетные данные (если используются) используемые в коннекторе на коллекторе. Проверить владельца папки службы, должно быть **kuma:kuma**

Посмотреть статус сервиса через консоль ssh. В случае если он запущен остановить:

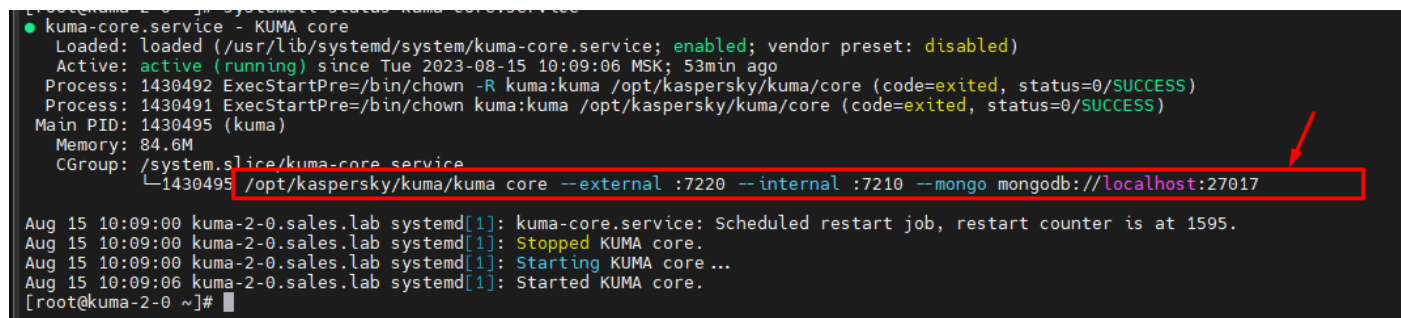
```
systemctl stop kuma-collector-ID_CEPBICA.service
```

запустить вручную (`--api.port` выбирайте любой свободный), и посмотреть есть ли ошибки при запуске:

```
/opt/kaspersky/kuma/kuma collector --id ID_CEPBICA --core https://FQDN_KUMA:7210 --api.port 7225
```

Если будут ошибки, они явно отразятся в консоли.

Для сервисов без ID, сначала смотрим параметры запуска службы, например:



```
● kuma-core.service - KUMA core
   Loaded: loaded (/usr/lib/systemd/system/kuma-core.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2023-08-15 10:09:06 MSK; 53min ago
     Process: 1430492 ExecStartPre=/bin/chown -R kuma:kuma /opt/kaspersky/kuma/core (code=exited, status=0/SUCCESS)
     Process: 1430491 ExecStartPre=/bin/chown kuma:kuma /opt/kaspersky/kuma/core (code=exited, status=0/SUCCESS)
    Main PID: 1430495 (kuma)
      Memory: 84.6M
    CGroup: /system.slice/kuma-core.service
            └─1430495 /opt/kaspersky/kuma/kuma core --external :7220 --internal :7210 --mongo mongodb://localhost:27017

Aug 15 10:09:00 kuma-2-0.sales.lab systemd[1]: kuma-core.service: Scheduled restart job, restart counter is at 1595.
Aug 15 10:09:00 kuma-2-0.sales.lab systemd[1]: Stopped KUMA core.
Aug 15 10:09:00 kuma-2-0.sales.lab systemd[1]: Starting KUMA core ...
Aug 15 10:09:06 kuma-2-0.sales.lab systemd[1]: Started KUMA core.
[root@kuma-2-0 ~]#
```

Затем делаем стоп службы и запускаем строку запуска от пользователя KUMA:

```
sudo -u kuma /opt/kaspersky/kuma/kuma core --external :7220 --internal :7210 --mongo
mongodb://localhost:27017
```

## Ручная очистка пространства хранилища

Если место на сервере хранения заканчивается, но еще не закончилось.

Выберите в активных сервисах - сервис хранилища (storage) и нажмите на кнопку смотреть разделы. Удаляйте наиболее старые партиции.

Далее (чтобы в будущем не заполнялось):

1. Resources/Storage - уменьшаем ретеншен период
2. после этого рестарт сервисов KUMA Storage. Изменения применятся примерно в течение часа и место освободится.

## Закончилось дисковое пространство

Если место уже закончилось.

- При all-in-one инсталляции в web консоль KUMA уже не пустит. Для all-in-one последовательность действий следующая:
  - Остановить все сервисы `systemctl stop kuma-*`
  - Удалить буферы коллекторов и коррелятора: `rm -rf /opt/kaspersky/kuma/collector/*/buffers/*` и `rm -rf /opt/kaspersky/kuma/correlator/*/buffers/`
  - Удалить кеш обогащения (если есть) коллекторов и коррелятора: `rm -rf /opt/kaspersky/kuma/collector/*/cache/enrichment/*` и `rm -rf /opt/kaspersky/kuma/correlator/*/cache/enrichment/`
  - Удалить лог файлы mongo и clickhouse: `rm -rf /opt/kaspersky/kuma/mongodb/log/*` и `rm -rf /opt/kaspersky/kuma/storage/*/logs/`
  - запустить сервис clickhouse: `systemctl start kuma-storage*`
  - запустить сервисы mongo и core: `systemctl start kuma-mongodb` и `systemctl start kuma-core` ( `systemctl start kuma-core-000000000-0000-0000-0000-000000000000` в версии 3.0+ )
  - залогиниться в web консоль KUMA, удалить лишние партиции (см. предыдущий пункт траблшута)
  - Стартуем оставшиеся сервисы KUMA ( `kuma-victoria-metrics.service` , `kuma-vmaalert.service` , `kuma-grafana.service` , `kuma-collector-*` , `kuma-correlator-<id>` )
- Если серверы хранения находятся отдельно:
  - Остановить все коллекторы и корреляторы. Никакие новые события не должны отправляться в Storage.
  - На серверах хранения остановить сервис kuma-storage
  - очистить файлы журналов `rm -rf /opt/kaspersky/kuma/storage/*/logs/`
  - очистить файлы буфера `rm -rf /opt/kaspersky/kuma/storage/*/buffers/`
  - Запустить сервис `systemctl start kuma-storage`
  - Залогиниться в web консоль KUMA, удалить лишние партиции (см. предыдущий пункт траблшута)

Далее (чтобы в будущем не заполнялось) в KUMA с версии 3.2: В Параметры - Мониторинг сервисов - Хранилище, выставьте нужные значения по оповещениям и ротации.

Для полной очистки данных хранилища (выполнять на хранилищах и киперах):

- `systemctl stop kuma-storage-*.service`
  - `rm -rf /opt/kaspersky/kuma/clickhouse/data`
  - `rm -rf /opt/kaspersky/kuma/clickhouse/coordination`
  - `rm -rf /opt/kaspersky/kuma/clickhouse/logs/`
  - Иногда полезно чистить
    - `rm -rf /opt/kaspersky/kuma/clickhouse/tmp/`
    - `rm -rf /opt/kaspersky/kuma/storage/*/buffers/`
  - `systemctl start kuma-storage-*.service`
-

# Создание сервисов в случае их отсутствия

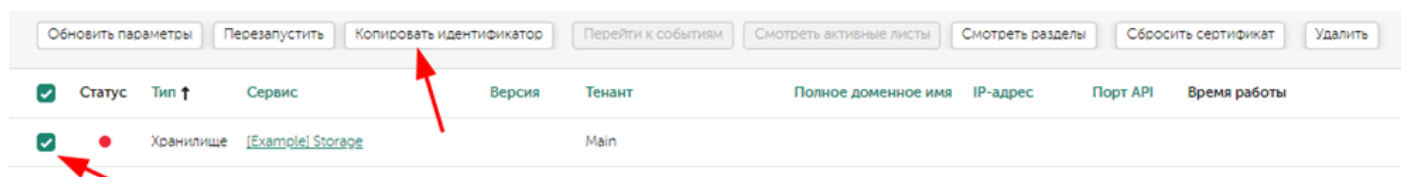
Если в разделе **Ресурсы - Активные сервисы** отсутствуют что-либо, то необходимо создать необходимые службы.

## Создаем сервис хранилища

Перейдите в **Ресурсы - Хранилища** затем нажать на кнопку **Добавить хранилище** придумайте название и затем укажите количество дней хранения событий и событий аудита (от 365 дней срок хранения аудита), затем нажмите **Сохранить**.

Публикуем созданный сервис **Ресурсы - Активные сервисы** затем выбрать созданный ранее сервис и нажать на кнопку Создать сервис.

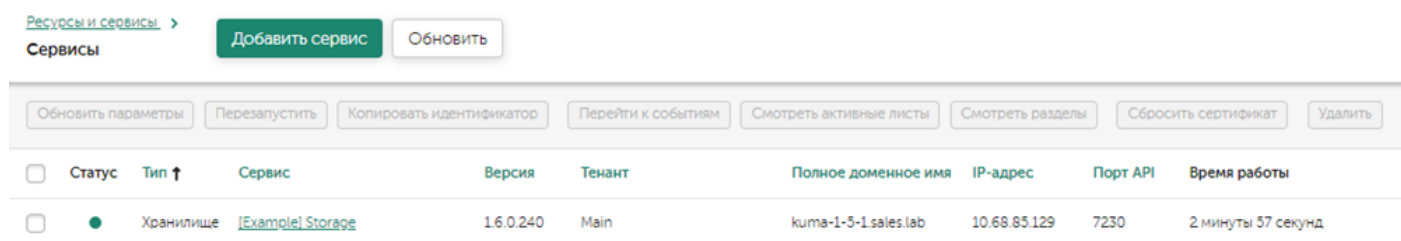
Скопируйте идентификатор сервиса:



Выполните команду в консоли:

```
/opt/kaspersky/kuma/kuma storage --id <ВАШ_ИДЕНТИФИКАТОР> --core  
https://<FQDN/ИМЯ_ХОСТА_СЕРВЕРА_ЯДРА>:7210 --api.port 7230 --install
```

В разделе **Ресурсы - Активные сервисы** убедитесь, что служба работает более 30 секунд с «зеленым» статусом индикатора:



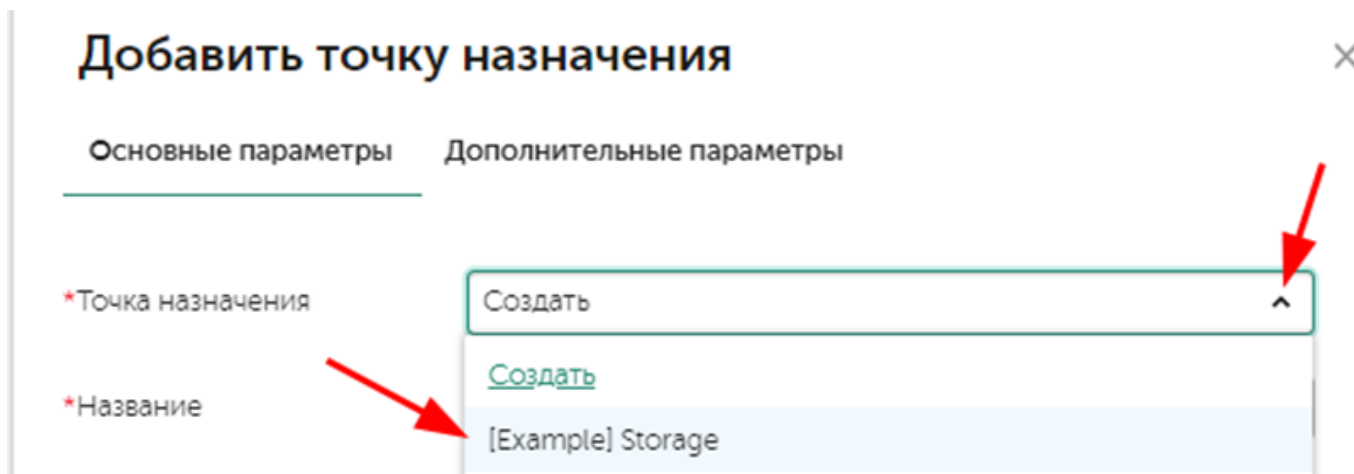
Далее создадим точку назначения, которая используется в маршрутизации событий, перейдите в **Ресурсы - Точки назначения**, затем нажмите на кнопку **Добавить точку назначения**. Придумайте название и затем в поле URL укажите FQDN и порт службы хранилища, например: `kuma-1-5-1.sales.lab:7230`, затем нажмите **Сохранить**.

Аналогичные действия понадобятся для установки остальных компонентов, только в интерфейсе будет доступна команда, которую необходимо будет выполнить для установки

службы.

## Создаем сервис коррелятора

Развернем коррелятор, перейдите в **Ресурсы - Корреляторы**, нажмите на кнопку Добавить коррелятор и затем пройдите по мастеру настроек, на шаге добавления маршрутизации добавьте точку назначения ранее созданного хранилища:



На последнем шаге нажмите кнопку **Сохранить и создать сервис**, в случае отсутствия ошибок появится командная строка для установки службы, скопируйте ее и выполните по ssh.

Рекомендуемая команда для установки коррелятора

```
/opt/kaspersky/kuma/kuma correlator --core https://kuma-1-5-1.sales.lab:7210 --id 73772985-95cf-4a3c-bc53-bc0ac5d8d131 --api.port 7231 --install
```

Копировать

Аналогично по мастеру создаются коллекторы для приема или получения событий с источников.

## Ошибка скрипта при установке - ipaddr

Если при установке возникает следующая ошибка (пишется в конце исполнения скрипта):

```
TASK [Ensure there is no localhost]
*****
skipping: [localhost] => (item=kuma.homelab.lan)

TASK [Ensure there is no IP as hostname]
*****
fatal: [localhost]: FAILED! => {}

MSG:
The conditional check 'item | ansible.netcommon.ipaddr' failed. The error was: The ipaddr filter requires python's netaddr be installed on the ansible controller
The error appears to be in '/tmp/kuma-ansible-installer/tasks/validate-inventory.yml': line 48, column 3, but may
be elsewhere in the file depending on the exact syntax problem.

The offending line appears to be:

- name: Ensure there is no IP as hostname
  ^ here

PLAY RECAP
```

То попробуйте установить нужную библиотеку для python, это было в требованиях <https://support.kaspersky.com/KUMA/1.6/ru-RU/231034.htm>, в случае отсутствия возможности это сделать то прокомментируйте (вставьте # перед строками) следующие строки в файле `kuma-ansible-installer/tasks/validate-inventory.yml`:

```
#- name: Ensure there is no IP as hostname
# loop: "{{ groups.all }}"
# when: item | ansible.netcommon.ipaddr
# fail:
# msg: |
# host: {{ item }}
# error: Hostname expected
```

## Слушать на 514 порту (порты < 1024)

По умолчанию это невозможно, так работает Linux. Для обхода этого в описании сервиса:

```
/usr/lib/systemd/system/kuma-collector-<ID>
```

Добавьте значение ниже тега [Service]

```
AmbientCapabilities=CAP_NET_BIND_SERVICE
```

Затем запустите команду ниже для обновления параметров сервиса

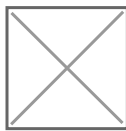
```
systemctl daemon-reload
```

# Не запускается хранилище после перезагрузки в версии от 2.1

Прописать следующую команду в SSH консоли:

```
sudo -u kuma touch /opt/kaspersky/kuma/clickhouse/data/flags/force_restore_data
```

## Ошибка БД ClickHouse Table read only replicas



Ошибка в логах хранилища выглядит так:

Зайдите в клиент ClickHouse:

```
/opt/kaspersky/kuma/clickhouse/bin/client.sh
```

Выполните команду:

```
system restart replica on cluster kuma kuma.events_local_v2
```

Для выхода из клиента нажмите **Ctrl+D**.

## Ошибка ZooKeeper - KEEPER\_EXCEPTION

Убедиться, что ipv6 включен:

```
ping ::1
```

Убедиться, что у вас корректное содержимое в /etc/hosts, смотрите этап **подготовки** п. 8

Очистить Coordination Zookeeper и наработки хранилища (удаляет все события):

```
systemctl stop kuma-storage-<ID>.service  
rm -rf /opt/kaspersky/kuma/storage/<ID>/data/*
```

```
rm -rf /opt/kaspersky/kuma/storage/<ID>/tmp/*
rm -rf /opt/kaspersky/kuma/clickhouse/coordination/*
systemctl start kuma-storage-<ID>.service
```

Иногда требуется (в случае если ошибка остаётся):

```
sudo -u kuma touch /opt/kaspersky/kuma/clickhouse/data/flags/force_restore_data
```

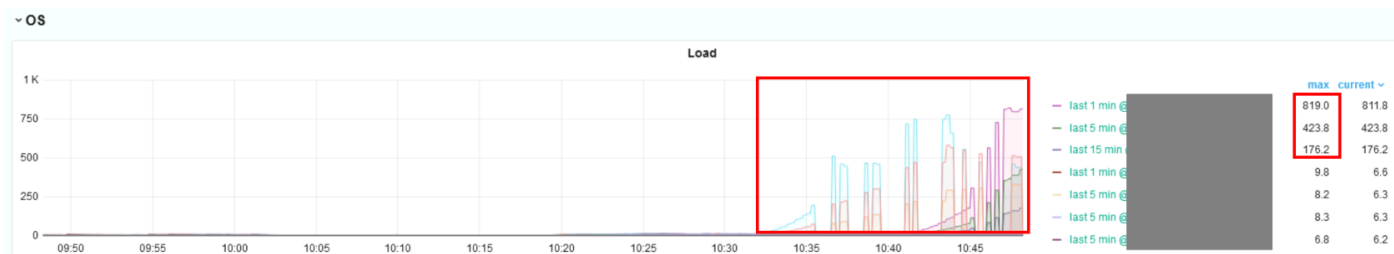
Либо:

```
/opt/kaspersky/kuma/clickhouse/bin/client.sh
system restore replica on cluster kuma kuma.events_local_v2
```

## Признаки нехватки выделенных ресурсов

Описание метрик доступно тут: <https://kb.kuma-community.ru/books/kuma-how-to/page/opisanie-metrik-v-kuma>

### Универсальный показатель Load

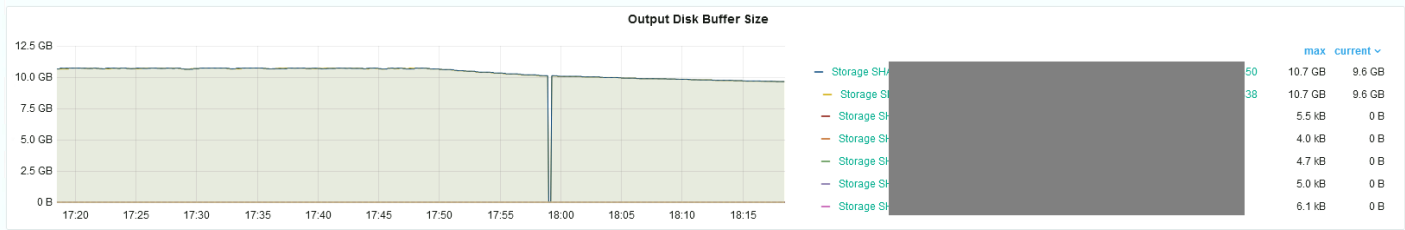


Мощностей не хватает, если на нодах высокий LA (Load average), утилизация CPU доходит до 100%, нагрузка зависит не только от потока (EPS) зависит, но и от профиля использования (количество обогащений, интеграций, используемых правил и т.д.). Средние значения нагрузки (Load averages):

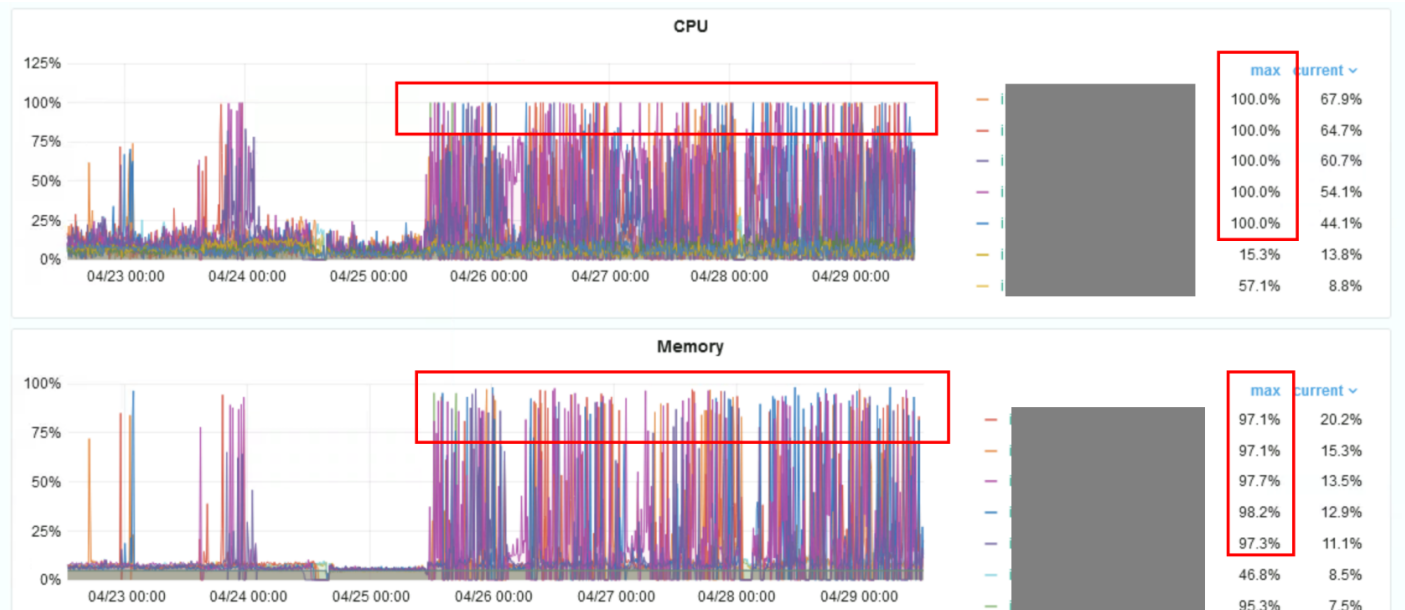
- Если значения равны 0.0, то система в состоянии простоя
- Если среднее значение для 1 минуты выше, чем для 5 или 15, то нагрузка растёт
- Если среднее значение для 1 минуты ниже, чем для 5 или 15, то нагрузка снижается
- Если значения нагрузки выше, чем количество (виртуальных) процессоров, то могут быть проблемы с производительностью (в зависимости от ситуации)

# Проблема с производительностью диска на хранилище по метрикам

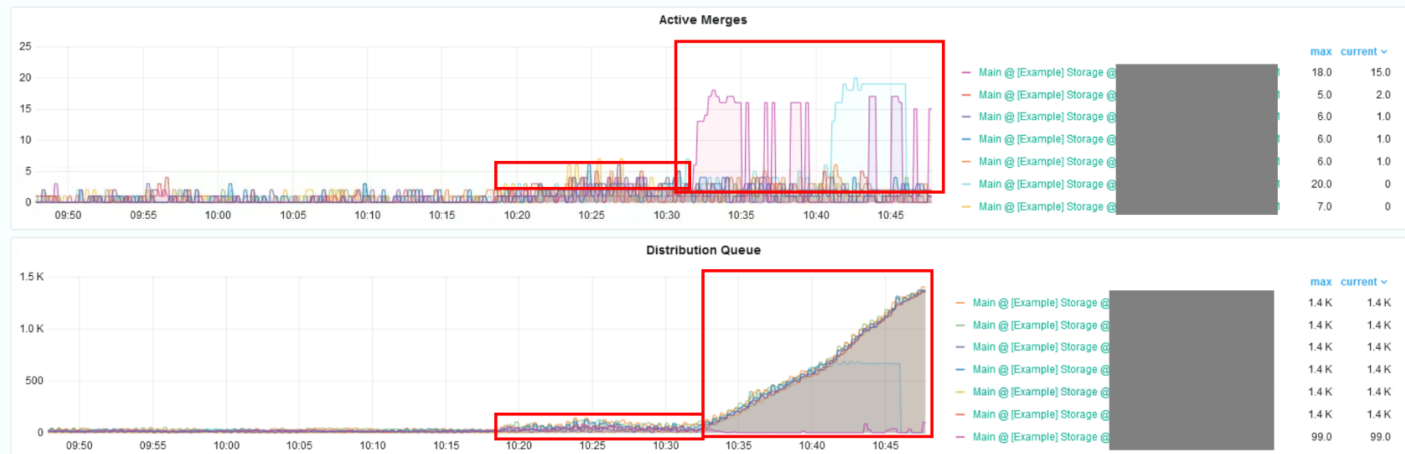
Накапливается буфер в точках назначения, на рисунке ниже хранилище не справляется с потоком, в норме буфер не накапливается при доступности хранилища:



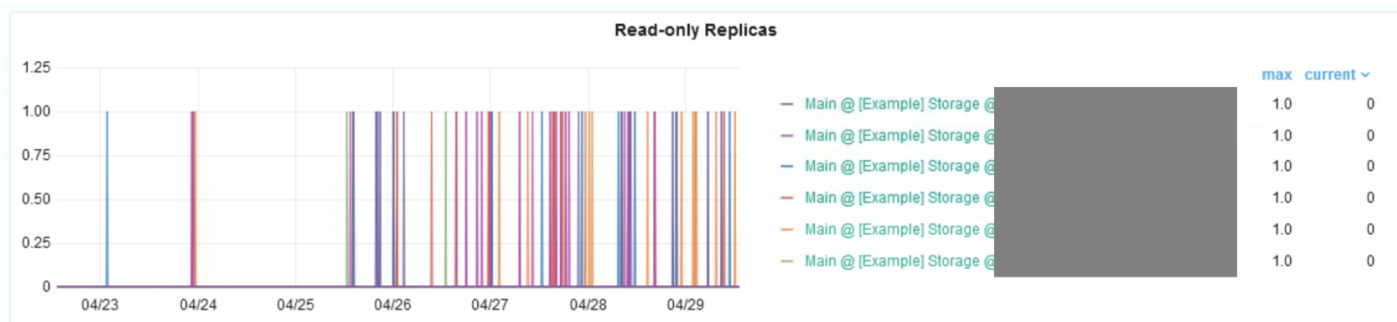
Загрузка CPU и RAM доходит до 100% или близко к этому значению:



В случае использования кластера хранилищ растут очереди распределенных запросов и мерджи:



Хранилища выпадают в режим только для чтения:



Может быть полезно для хранилища при нагрузках:

- Вынесение и использование отдельных машин с ролью keeper
- Использование более производительного RAID массива
- Использование более производительного дискового носителя
- Работа с объемом буфера для записи в хранилище и временем ожидания
- Использование дополнительного шарда для повышения производительности

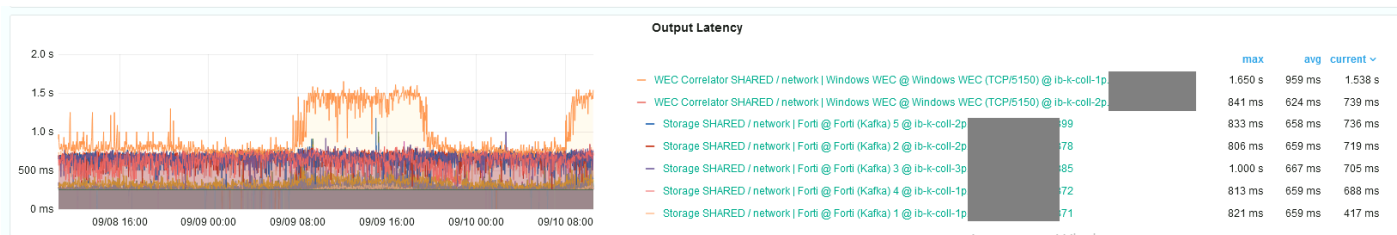
Также оценить нагрузку на диск можно командой `top` в ОС Linux, отслеживая параметр `wa` (wa: io wait cpu time (or) % CPU time spent in wait on disk) — показывает процент операций, готовых быть выполненными процессором, но находящихся в состоянии ожидания от диска. Допустимое значение  $< 0.5$ , в идеале 0.

Требования к хранилищу - [тут](#)

Устройство кластера хранилищ - [тут](#)

## Проблема с производительностью сервисов (особенно коррелятор), большой Output Latency

Если CPU заняты не полностью (не на 100%), а Latency существенный (более 1-2 сек) — проблема вероятней всего в сети. Ниже пример допустимого Output Latency.



Для подтверждения, что проблема именно в сети, установите коррелятор локально (что бы события не через сетевой интерфейс летели, а через localhost) на сервер с коллекторами, и подайте туда поток событий с аналогичной нагрузкой. Также полезно ознакомиться с разделом ниже.

# Проблема с производительностью сервисов (особенно коррелятор), большая нагрузка на CPU, памяти и периодические перезапуски службы

Большое кол-во коллекторов (> 100 шт.) перенаправляют события на коррелятор. Высокая загрузка CPU на серверах и большое количество отправляемых пакетов с коллекторов приводит к общей деградации серверов со службой коррелятора по CPU и памяти (корреляторы перезапускаются каждые 5 минут). Пример поведения по метрикам:



Для решения проблемы были внесены изменения в точку назначения коррелятора, увеличив время ожидания соединения до 300-400 сек (задача найти такой баланс между размером пачки и скоростью обработки/времени задержки). Множество мелких вставок данных от множества коллекторов в коррелятор хуже, лучше редкие и пачки объемней. Утилизация памяти на серверах со службой коррелятора выше 50% в течение дня не поднималась.

## Редактирование точки назначения

Основные параметры

Дополнительные параметры

Размер буфера ⓘ

0



Время ожидания ⓘ

300



Размер дискового буфера ⓘ

0



Если CPU потребляются не на 100%, но при этом увеличивается время задержки обработки или нестабильное потребление ОЗУ, то можно:

- попробовать подобрать оптимальное (больше не значит производительней - используйте метод научного тыка) число рабочих процессов / workers для службы, количество воркеров должно быть идентичным для совместно работающих служб, например, на коллекторе и его точке назначения - корреляторе.

- в случае когда соединений к коллектору становится очень много, то стандартное значение буфера (нуль = 1Мб) приводит к увеличению потребления оперативной памяти, так как память выделяется под каждое соединение. Рекомендуемое значение = размер события + небольшой запас в байтах (можно посмотреть в метриках в разделе нормализация - Raw & Normalized event size). Настройка выполняется на коллекторе "Транспорт - Дополнительные - Размер буфера")

Также полезным будет события от множества коллекторов отправлять в маршрутизатор событий (отдельный служба в KUMA) для реализации отдачи потока событий от одной точки.

Также это один из поводов дополнительно увеличить доступные коррелятору вычислительные ресурсы, либо разделить имеющиеся правила на N независимых групп, разместив каждую группу правил на отдельной группе корреляторов, если решение выше не сильно помогло.

---

## Профилирование нагрузки служб

Профилирование необходимо для более глубокого изучения проблем связанных с нагрузкой и ресурсами.

1. Включить профилирование для сервиса, отредактируйте файл сервиса

```
/usr/lib/systemd/system/kuma-<serviceKind>-<ID>.service
```

В файле необходимо найти параметр `ExecStart=` и дописать в конце `--pprof` после этого выполнить `systemctl daemon-reload` и перезапустить (restart) сервис.

2. Команда для сбора профиля данных по памяти

```
curl http://127.0.0.1:7211/debug/pprof/heap > /tmp/heap.out
```

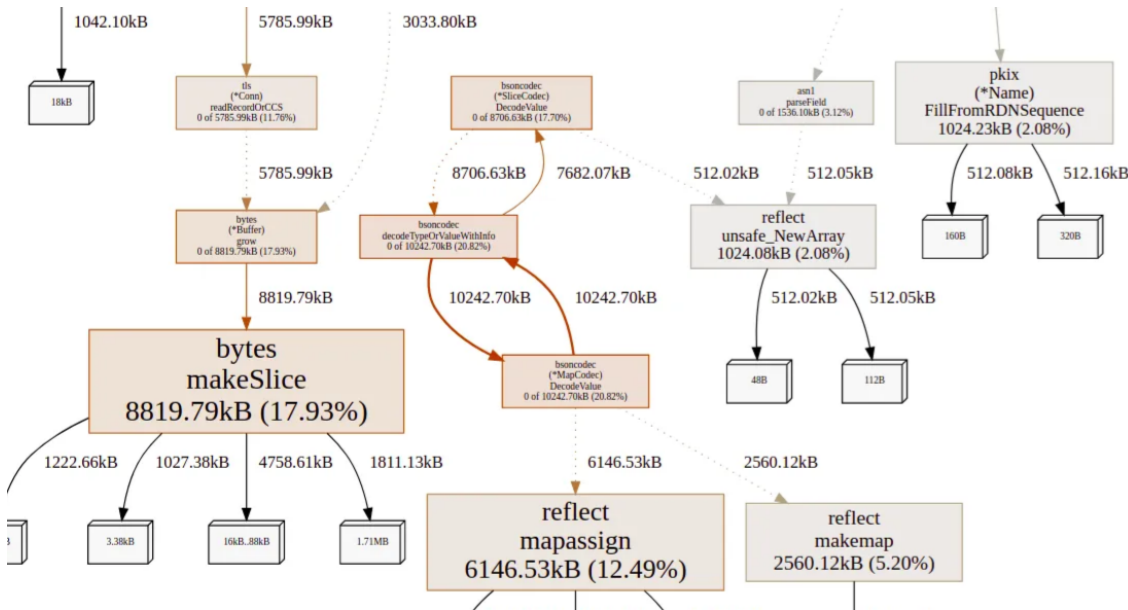
3. Команда для сбора профиля данных по процессору

```
curl http://127.0.0.1:7211/debug/pprof/profile?seconds=30 > /tmp/cpuprof.out
```

4. Расшифровка профиля. Потребуется установленный Go из папки с бинарем go (загружаете для своей ОС <https://go.dev/dl/>) запускается команда (в результате будет сгенерирована картинка с профайлером):

```
go tool pprof -png /tmp/heap.out
```

Пример резултата:



Другие профили:

```
curl -o allocs.out http://127.0.0.1:7211/debug/pprof/allocs
curl -o block.out http://127.0.0.1:7211/debug/pprof/block
curl -o cmdline.out http://127.0.0.1:7211/debug/pprof/cmdline
curl -o goroutine.out http://127.0.0.1:7211/debug/pprof/goroutine
curl -o mutex.out http://127.0.0.1:7211/debug/pprof/mutex
curl -o threadcreate.out http://127.0.0.1:7211/debug/pprof/threadcreate
curl -o trace.out http://127.0.0.1:7211/debug/pprof/trace?seconds=30
```

# Мониторинг работы ядра в кластере Kubernetes

Посмотреть логи сервисов KUMA:

```
k0s kubectl logs -f -l app=core --all-containers -n kuma
```

На хосте контроллера можно посмотреть какие сервисы запущены:

```
k0s kubectl get pods --all-namespaces
```

По команде выше, в столбце `NAMESPACE` нам интересна строка `kuma`, в этой строке берем значение столбца `NAME` для просмотра конфигурации используем команду:

```
k0s kubectl get pod core-deployment-984fd44df-5cfk5 -n kuma -o yaml | less
```

Получить список рабочих узлов:

```
k0s kubectl get nodes
```

Получить расширенную инфу по рабочему узлу, в т.ч. потребление:

```
k0s kubectl describe nodes kuma-4.local
```

Зайти в командную строку пода core-\*:

```
k0s kubectl exec --stdin --tty core-deployment-984fd44df-gqzlx -n kuma -- /bin/sh
```

Потребление ресурсов контейнеров внутри пода:

```
k0s kubectl top pod core-deployment-984fd44df-gqzlx -n kuma --containers
```

Для более удобного управления кластером используйте утилиту k9s - <https://github.com/derailed/k9s>

После установки ядра в кластере в папке установки прописывается конфиг, например, `/root/kuma-ansible-installer/artifacts/k0s-kubeconfig.yml`. Его надо “скормить” k9s командой: `export KUBECONFIG=/root/kuma-ansible-installer/artifacts/k0s-kubeconfig.yml`

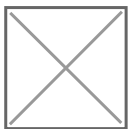
Для персистентности переменной KUBECONFIG, необходимо выполнить команду:

```
echo "KUBECONFIG=/root/kuma-ansible-installer/artifacts/k0s-kubeconfig.yml" >> /etc/environment
```

Проверить переменные окружения можно командой:

```
export -p
```

Далее можно запустить утилиту (ее можно загрузить **отсюда**) просто указав на исполняемый файл `./k9s`

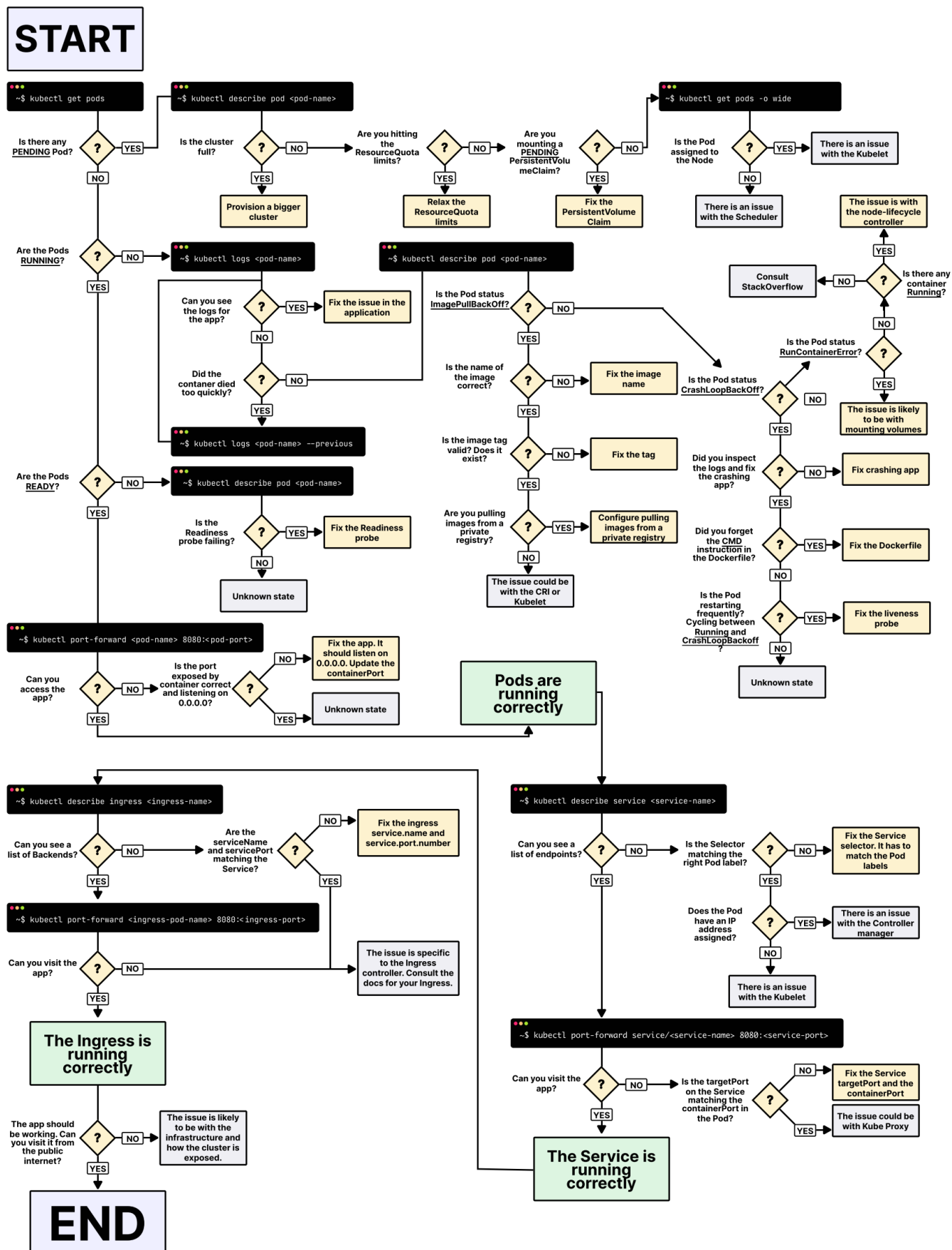


Небольшие подсказки по горячим клавишам k9s:

- `?` — Посмотреть хелп по командам
- `Shift+F` — Настроить порт форвардинг (например, для longhorn-ui)
- `:` — для ручного ввода команд, ESC для выхода
- `:events` — Смотреть события кубера

- `:pods` — Смотреть поды кубера (стандартное отображение)

Общий подход по траблшутингу k8s:



Revision #35

Created 11 August 2023 14:05:15 by Boris RZR

Updated 7 April 2025 11:20:07 by Boris RZR