



САМАРСКИЙ УНИВЕРСИТЕТ  
SAMARA UNIVERSITY

# Безопасность систем баз данных

## Лекция 10 Мониторинг

Агафонов Антон Александрович  
д.т.н., профессор кафедры ГИИБ

Самара



- Мониторинг инфраструктуры
- Мониторинг базы данных
- Возможности мониторинга в PostgreSQL
- Системы мониторинга: Zabbix, Prometheus
- Пример настройки системы мониторинга





Мониторинг инфраструктуры — это автоматизированная система контроля состояния элементов, включающая получение сведений о рабочих характеристиках системы путем регулярного измерения параметров функционирования различных компонентах инфраструктуры с целью оперативного выявления и устранения сбоев в работе системы.

Зачем нужен мониторинг:

- Исправления и изменения в связи с неисправностями
- Анализ производительности и поведения
- Планирование мощностей
- Отладка и послеаварийный анализ
- Бизнес-анализ
- Причинно-следственные связи





### Уровень приложения:

- *Мониторинг бизнес-логики приложения* – проверка факта доступности приложения и оцениваем бизнес-показатели.
- *Мониторинг health-метрик сервисов* – проверка жизнеспособности сервисов и всех ресурсов, требуемых для их стабильной работы.
- *Интеграционный мониторинг* – проверка синхронной и асинхронной коммуникации между критическими для бизнеса системами.

### Уровень приложения как продукта:

- *Сбор и наблюдение журналов приложения.*
- *Мониторинг производительности приложения* (Application Performance Monitoring, APM) – мониторинг приложения с целью анализа ошибок в коде и их влияния на производительность. Проверка состояние физического оборудования и виртуальной машины, контейнера и самого приложения, вспомогательной инфраструктуры, кэша.
- *Трассировка* – отслеживание всех вызовов для контроля выполнения в распределенных системах.





### Уровень инфраструктуры

- *Мониторинг уровня оркестрации* — проверка, насколько работоспособны контейнеры, кластеры и системы управления.
- *Мониторинг системного ПО* — проверка, насколько хорошо функционирует операционная система и ее компоненты.
- *Мониторинг уровня аппаратного обеспечения* — диагностика аппаратной части: процессора, материнской платы, жестких дисков и оптических накопителей.

### Оповещение:

- *Организация единой системы рассылки оповещения* с контролем доставки оповещений.
- *Организация системы дежурств* – необходимо определить зоны ответственности с указанием кто, когда и за что отвечает.
- *Организация «базы знаний» обработки инцидентов* – по каждому серьезному инциденту или ошибке в приложении фиксируются действия, которые решают проблемы.





Мониторинг баз данных — это комплексная задача, которая включает в себя отслеживание показателей производительности баз данных на различных уровнях: SQL (оптимальность запросов), экземпляр БД, инфраструктура, пользователи.

Виды мониторинга:

- Мониторинг доступности БД
- Мониторинг подключенных клиентов и их активности
- Мониторинг работы с данными
- Мониторинг запросов
- Мониторинг фоновых процессов
- Мониторинг системных метрик





1. Мониторинг доступности баз данных.
2. Мониторинг общих показателей задержек/ошибок и сквозной мониторинг работоспособности.
3. Инструментарий прикладного уровня для измерения задержек/ошибок для каждого вызова базы данных.
4. Сбор показателей, относящихся к уровням системы, хранилища, базы данных и приложений, независимо от того, будут они полезны или нет. Для большинства операционных систем, сервисов и БД это обеспечивается подключаемыми модулями.
5. Специальные проверки для отдельных известных проблем.





Параметры мониторинга хранилища данных:

- уровень соединения с хранилищем данных;
- контроль процессов внутри базы данных;
- объекты базы данных;
- вызовы/запросы к базе данных.







- Контроль времени, которое требуется для подключения к внутреннему хранилищу данных в рамках транзакции.
- Контроль максимального и фактического количества соединений с БД.
- Загруженность:
  - журнал TCP-соединений,
  - превышение времени ожидания соединения,
  - ожидание потоков в пулах соединений,
  - подкачка страниц памяти,
  - блокировки процессов в базе данных и т.д.
- Ошибки:
  - журналы базы данных;
  - журналы приложений и прокси-серверов;
  - журналы хоста.





- Показатели пропускной способности и задержки:
  - операции чтения / записи;
  - фиксация (подтверждение) / откат изменений;
  - использование DDL-операторов.
- Состояние репликации:
  - задержка репликации;
  - прерванная репликация;
  - несогласованность данных.
- Коллизии параллельного доступа и блокировки.



- Мониторинг, сколько памяти занимают каждый объект базы данных и связанные с ним ключи и индексы.
- Контроль характера распределения критически важных данных. Например, знать верхние и нижние границы, средние значения и область значений для элементов данных полезно для понимания производительности операций индексации и поиска.
- Если набор данных сегментирован с использованием диапазонов значений ключа или списков, то понимание того, как происходит распределение между сегментами, поможет обеспечить максимальную производительность для каждого узла. Мониторинг позволит сформулировать рекомендации о необходимости изменения балансировки или пересмотра моделей сегментирования данных.

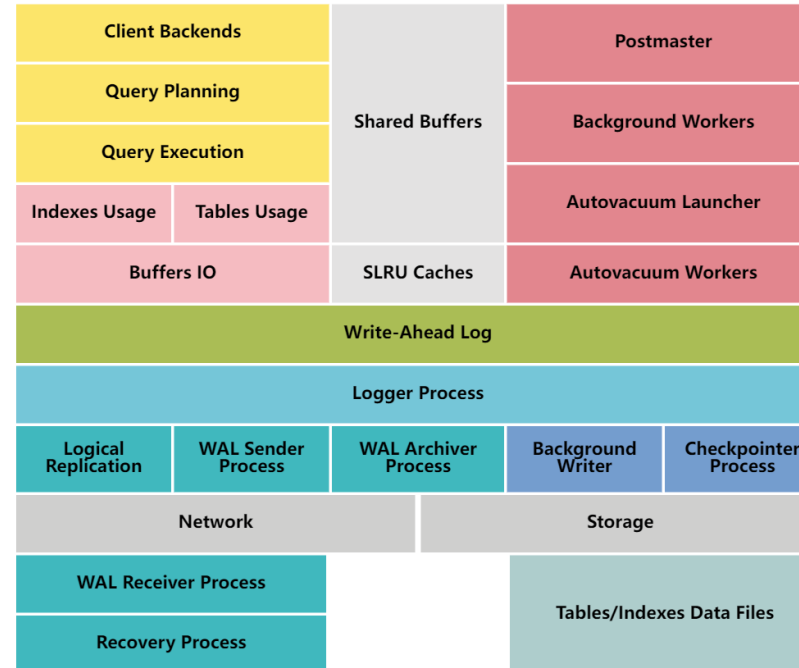


- Потребление ресурсов процессора и ввода-вывода.
- Количество прочитанных и записанных строк.
- Детализированное время выполнения и время ожидания запроса.
- Счетчик выполненных операций.

Решающее значение для оптимизации имеет понимание способов оптимизации, применяемых индексов и статистики по объединению, сортировке и агрегированию.



## Postgres Observability



nicstat

iostat

`pg_stat_ssl`

`_log_backend_memory_contexts()`

`pg_stat_monitor`  
`pg_stat_kcache`  
`pg_wait_sampling`  
`pg_blocking_pids()`  
`pg_stat_progress_cluster`  
`pg_stat_progress_copy`

`pg_stat_activity`  
`pg_backend_memory_contexts`  
`EXPLAIN`  
`pg_stat_statements`  
`pg_stat_user_functions`  
`pg_prepared_xacts`  
`pg_locks`  
`pg_stat_progress_create_index`  
`pg_stat_all_indexes`  
`pg_stat_all_tables`  
`pg_statio_all_indexes`  
`pg_statio_all_tables`  
`pg_statio_all_sequences`

`pg_get_wal_replay_pause_state()`

`pg_is_wal_replay_paused()`  
`pg_current_wal_lsn()`  
`pg_wal_lsn_diff()`  
`pg_ls_logdir()`  
`pg_current_logfile()`  
`pg_replication_slots`  
`pg_stat_replication_slots`  
`pg_stat_replication`  
`pg_stat_subscription`  
`pg_stat_wal_receiver`

`pg_ls_logicalmapdir()`  
`pg_ls_logicalsnapdir()`  
`pg_ls_replslotdir()`  
`pg_stat_subscription_stats`

`pg_stat_archiver`  
`pg_ls_archive_statusdir()`

`pg_stat_database_conflicts`  
`pg_stat_recovery_prefetch`

`pg_buffercache`  
`pg_shmem_allocations`  
`pg_stat_slru`  
`pg_stat_activity`

`pg_stat_database`  
`pg_stat_progress_vacuum`

`pg_stat_progress_analyze`  
`pg_stat_all_tables`

`pg_stat_wal`  
`pg_ls_waldir()`  
`pg_walfile_name()`  
`pg_current_wal_insert_lsn()`

`pg_last_wal_receive_lsn()`  
`pg_last_wal_replay_lsn()`

`pg_walfile_name_offset()`  
`pg_current_wal_flush_lsn()`

`pg_last_xact_replay_timestamp()`

`pg_stat_bgwriter`

`pg_stat_progress_basebackup`  
`pgstattuple`

`pg_tablespace_size()`  
`pg_database_size()`  
`pg_total_relation_size()`  
`pg_relation_size()`  
`pg_table_size()`  
`pg_indexes_size()`  
`pg_ls_tmpdir()`  
`pg_ls_dir()`

`pg_relation_filenode()`  
`pg_relation_filepath()`  
`pg_filenode_relation()`

<https://pgstats.dev/>





Спецификация	Описание
<code>pg_stat_activity</code>	Одна строка для каждого серверного процесса с информацией о текущей активности процесса, включая его состояние и текущий запрос.
<code>pg_stat_replication</code>	По одной строке для каждого процесса-передатчика WAL со статистикой по репликации на ведомом сервере, к которому подключён этот процесс.
<code>pg_stat_wal_receiver</code>	Одна строка со статистикой приёмника WAL, полученной с сервера, на котором работает приёмник.
<code>pg_stat_subscription</code>	Одна строка для подписки, сообщающая о рабочих процессах подписки.
<code>pg_stat_gssapi</code>	Одна строка для каждого подключения, в которой показывается информация об использовании аутентификации и шифровании GSSAPI для данного подключения.
<code>pg_stat_progress_analyze</code>	По одной строке с текущим состоянием для каждого обслуживающего процесса, в котором работает ANALYZE.
<code>pg_stat_progress_create_index</code>	По одной строке с текущим состоянием для каждого обслуживающего процесса, в котором выполняется CREATE INDEX или REINDEX.





Спецификация	Описание
<code>pg_stat_archiver</code>	Одна строка со статистикой работы процесса архивации WAL.
<code>pg_stat_database</code>	Одна строка для каждой базы данных со статистикой на уровне базы.
<code>pg_stat_all_tables</code>	По одной строке на каждую таблицу в текущей базе данных со статистикой по обращениям к этой таблице.
<code>pg_stat_all_indexes</code>	По одной строке для каждого индекса в текущей базе данных со статистикой по обращениям к этому индексу.
<code>pg_statio_all_tables</code>	По одной строке для каждой таблицы в текущей базе данных со статистикой по операциям ввода/вывода с этой таблицей.
<code>pg_statio_all_indexes</code>	По одной строке для каждого индекса в текущей базе данных со статистикой по операциям ввода/вывода для этого индекса.





Просмотр информации о текущих подключениях

```
SELECT * FROM pg_stat_activity;
```

	datid	datname	username	application_name	state	query
	5	postgres	postgres	pgAdmin 4 - DB:postgres	idle	...
	16908	pagila	postgres	pgAdmin 4 - DB:pagila	idle	...
	16908	pagila	postgres	pgAdmin 4 - CONN:729358	active	SELECT * FROM pg_stat_activity;

Просмотр статистики использования пользовательских таблиц

```
SELECT * FROM pg_stat_user_tables;
```

	relid	schemaname	relname	seq_scan	n_tup_ins	n_tup_upd	n_tup_upd
	17004	public	country	1	100	0	0
	16935	public	customer	6	599	0	0
	16947	public	actor	34	205	19	0





Модуль `pg_stat_statements` предоставляет возможность отслеживать статистику выполнения сервером всех операторов SQL.

Этот модуль нужно загружать, добавив `pg_stat_statements` в `shared_preload_libraries` в файле `postgresql.conf`, так как ему требуется дополнительная разделяемая память.

Когда модуль `pg_stat_statements` загружается, он отслеживает статистику по всем базам данных на сервере. Для получения и обработки этой статистики этот модуль предоставляет представление `pg_stat_statements` и вспомогательные функции.

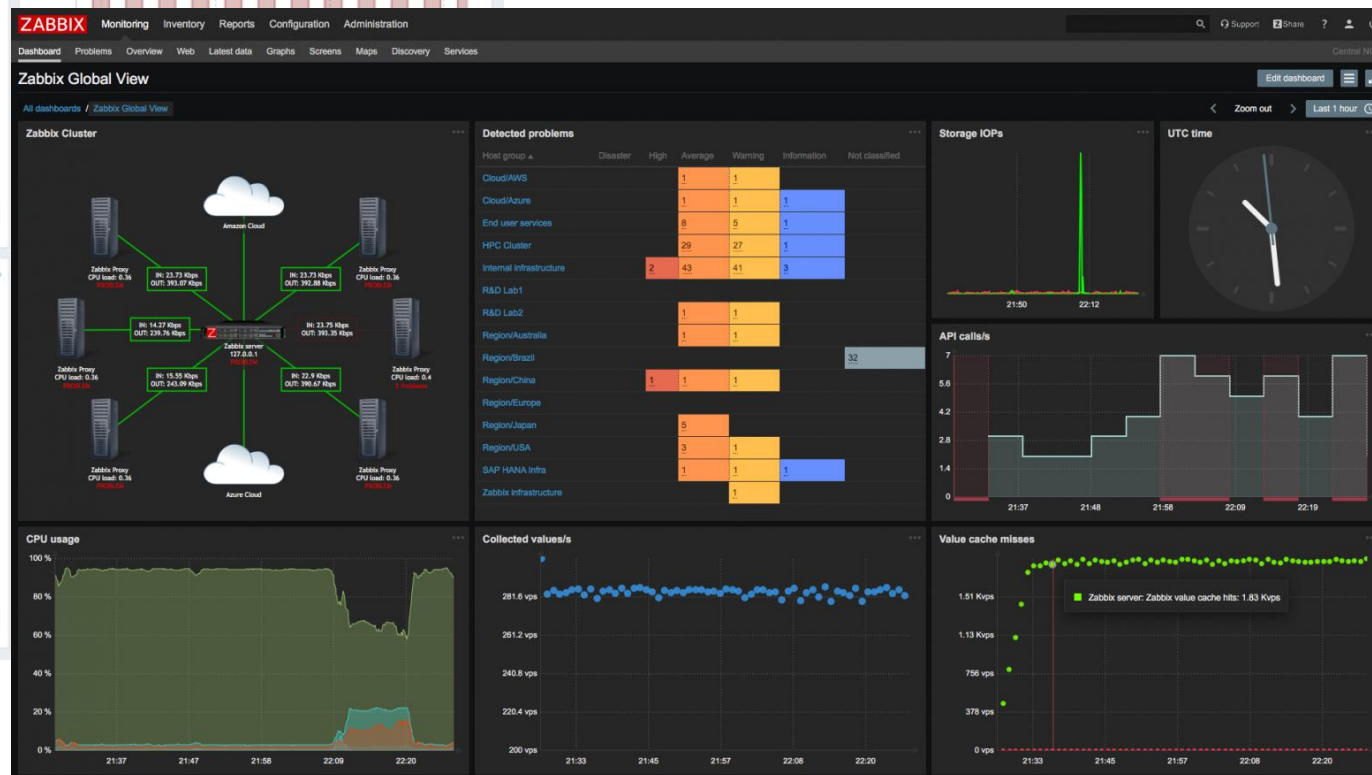
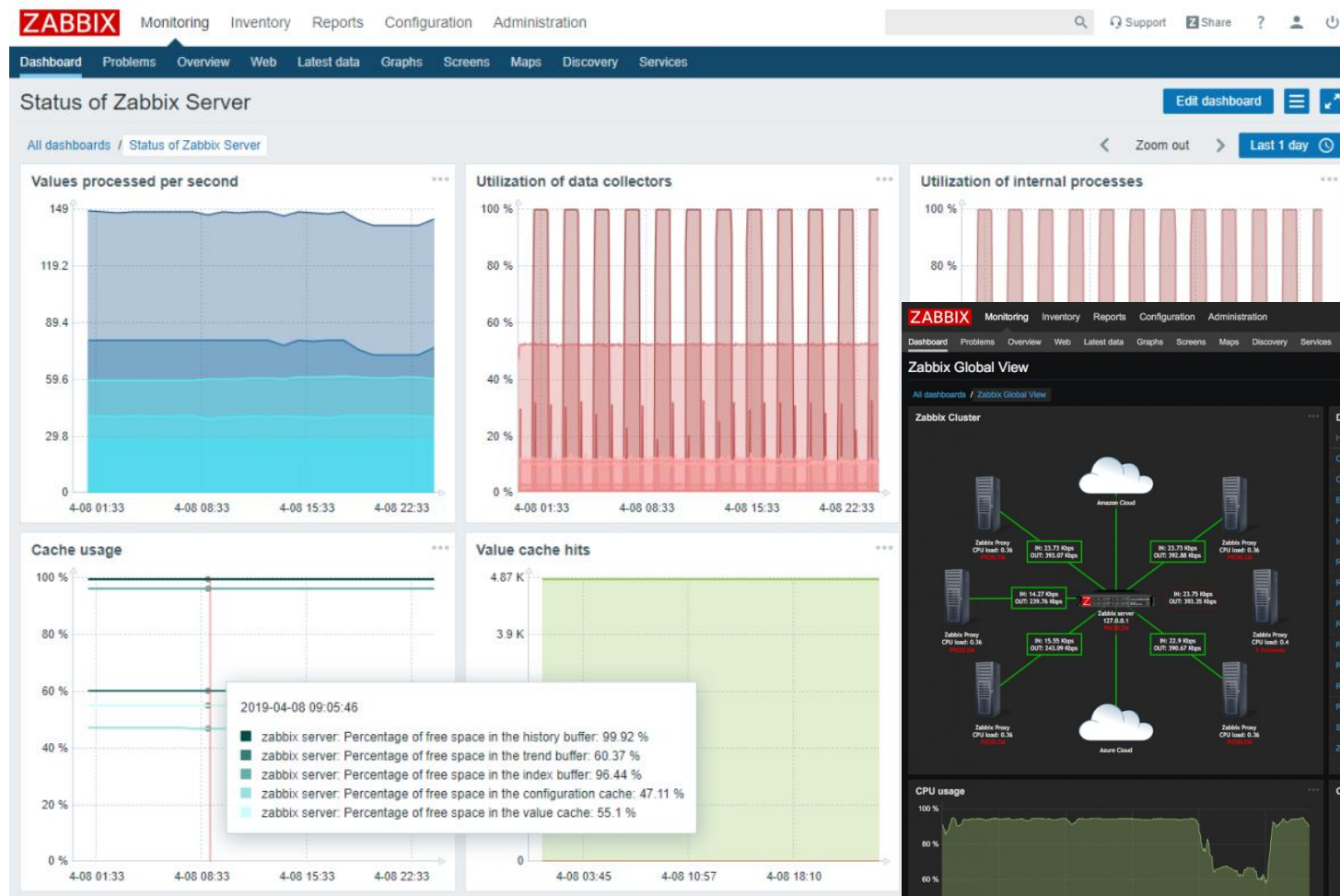
Можно мониторить:

- наиболее часто выполняемые запросы;
- самые долгие запросы;
- самые «тяжелые» запросы в плане использования ресурсов;
- запросы, которые возвращают большое число строк и т.д.





# Системы мониторинга. Zabbix





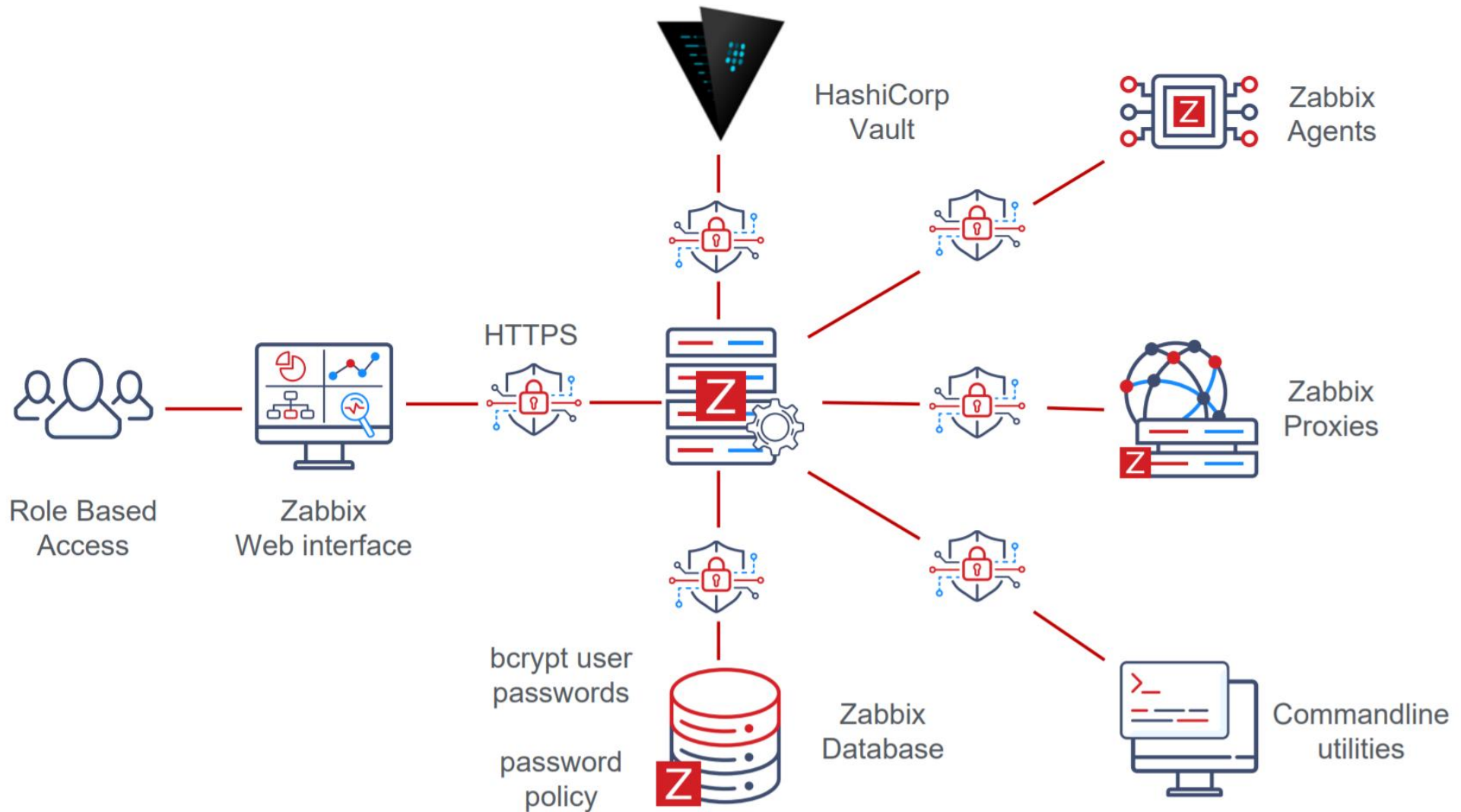
**Zabbix-сервер** — ядро системы, которое дистанционно контролирует сетевые сервисы и является хранилищем, в котором содержатся все конфигурационные, статистические и оперативные данные. Для хранения данных используется MySQL, PostgreSQL, SQLite или Oracle Database.

**Zabbix-агент** — программа контроля локальных ресурсов и приложений на сетевых системах, эти системы должны работать с запущенным Zabbix-агентом.

**Zabbix-прокси** собирает данные о производительности и доступности от имени Zabbix-сервера. Все собранные данные заносятся в буфер на локальном уровне и передаются Zabbix-серверу, к которому принадлежит прокси-сервер. Он может быть также использован для распределения нагрузки одного Zabbix-сервера.



# Zabbix. Архитектура системы





### **Хранение данных**

Для хранения данных используется внешняя БД MySQL, PostgreSQL, SQLite или Oracle Database.

### **Запросы**

Для работы с БД используется язык SQL.

### **Оповещение**

Встроена функция оповещения, что позволяет управлять событиями различными способами: отправка сообщений, выполнение удаленных команд и т.д. Также можно настраивать сообщения в зависимости от роли получателя, выбирая, какую информацию включать, например, дату, время, имя хоста, значение элементов, значения триггеров, профиль хоста и т.д.





- ▲ Единая точка доступа для всех пользователей.
- ▲ Доступ к данным и к конфигурации разграничен.
- ▲ Поддерживает все основные платформы (Linux, MacOS, Windows).
- ▲ В качестве базы данных для хранения метрик можно использовать MySQL, PostgreSQL, SQLite или Oracle.
- ▲ Широкие возможности мониторинга.
- ▼ Сложная настройка.
- ▼ Сложно и долго масштабируется.
- ▼ Все данные мониторинга хранятся в базе (нужны дополнительные вычислительные мощности).
- ▼ Мониторинг происходит через агента, который работает постоянно и устанавливается на каждом сервере.



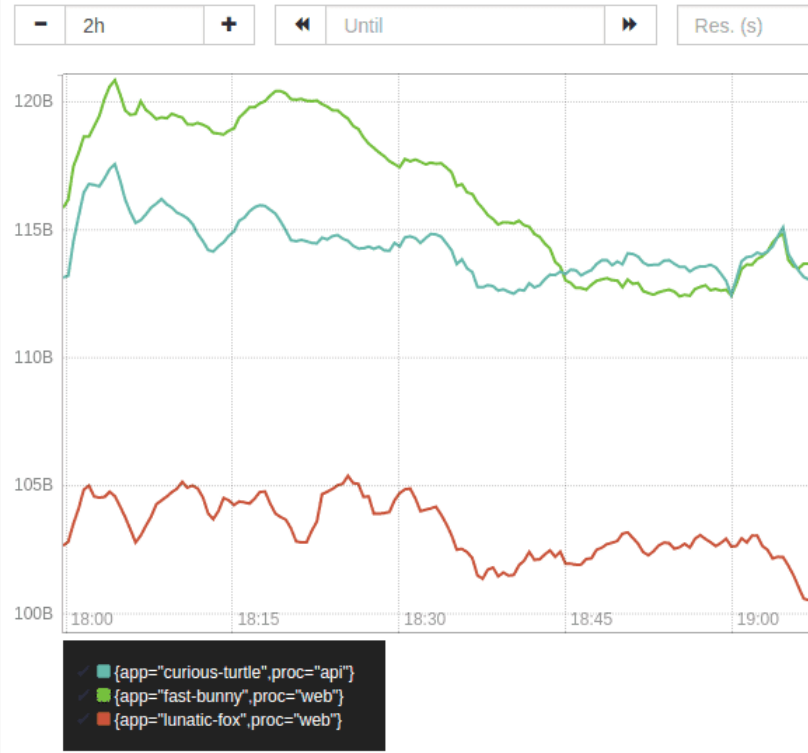
# Системы мониторинга. Prometheus

Prometheus Alerts Graph Status Help

topk(3, sum(rate(bazooka\_instance\_cpu\_time\_ns[5m])) by (app, proc))

Execute

Graph Console







**Сервер Prometheus** (Prometheus Server) – центральный компонент системы мониторинга, чья основная задача — хранить и мониторить определенные объекты. В терминах Prometheus объекты мониторинга называются **целевыми объектами** (Prometheus targets). Через конечную точку HTTP-сервер Prometheus проверяет статус приложения.

Каждый элемент целевого объекта, который необходимо мониторить, называется **метрикой**. Prometheus собирает метрики на основе методов push и pull. В первом методе отслеживаемое приложение отвечает за отправку метрик в систему мониторинга с помощью **Push шлюза** (Pushgateway).

В методе pull приложение подготавливает метрики, а сервер Prometheus считывает целевые объекты с интервалом, который определяет сбор метрик, и сохраняет их в **базе данных временных рядов** (Time-Series database, TSDB). Целевые объекты и временной интервал считывания метрик задаются в конфигурационном файле `prometheus.yml`.





Компонент управления оповещениями **AlertManager** служит для запуска оповещений через Email, Slack или другие клиентские уведомления.

Для визуализации данных можно использовать как пользовательский **веб-интерфейс Prometheus** (web UI), так и сторонние клиенты (наиболее популярный – **Grafana**).

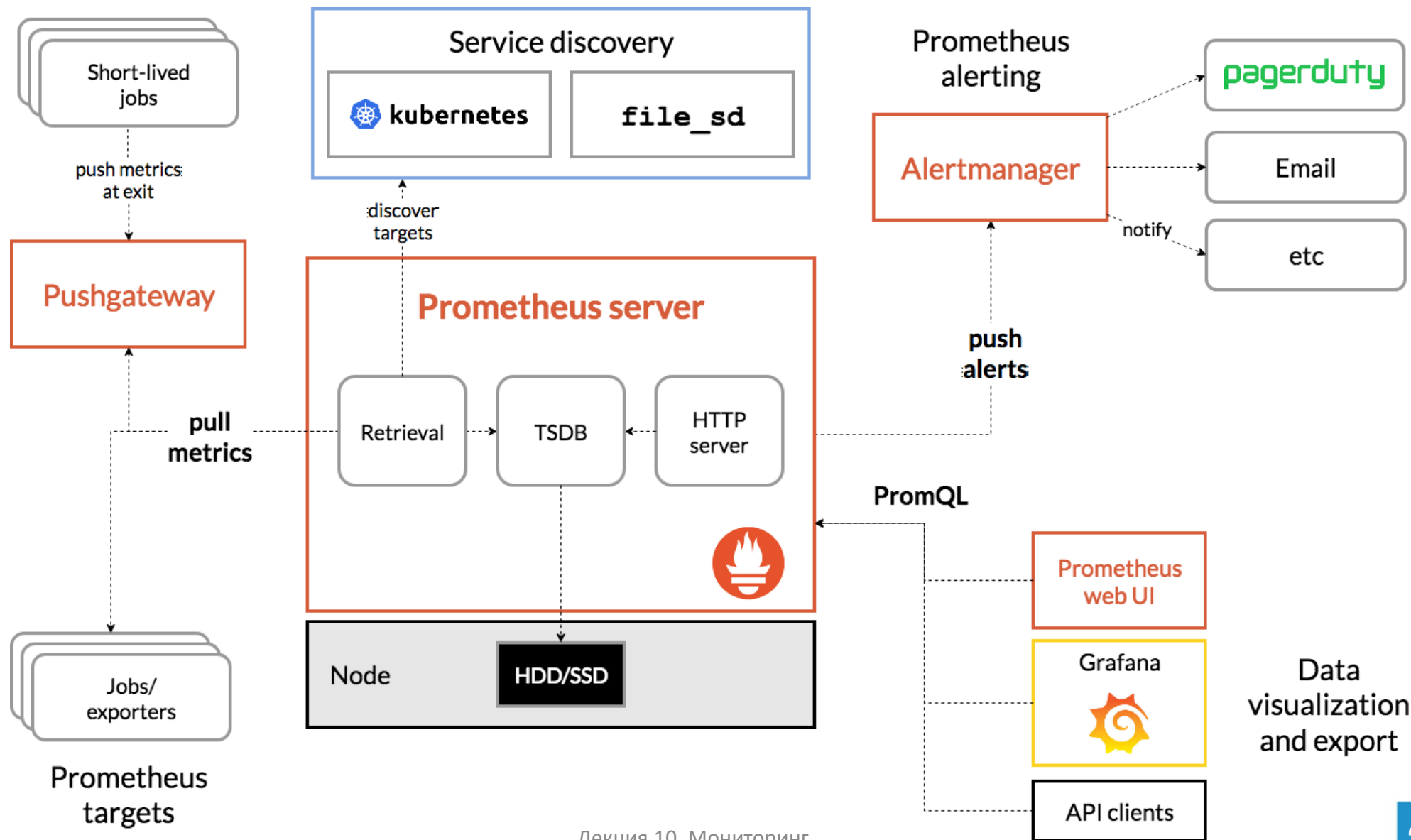
Для мониторинга сторонних систем (таких как сервер Linux, сервер MySQL и т.д.) можно использовать **экспортеры**. Экспортер — часть программного обеспечения, которое получает существующие метрики от сторонней системы и экспортирует их в формат, понятный серверу Prometheus.

Примеры экспортеров:

- `node_exporter` – выдача метрик файловой системы.
- `postgres_exporter` – выдача метрик базы данных PostgreSQL.
- `mongodb_exporter` – выдача метрик базы данных MongoDB.



# Prometheus. Архитектура системы





## Хранение данных

Prometheus хранит данные в виде БД временных рядов (time-series database, TSDB). Prometheus не подходит для хранения текста, логов или журналов событий.

## Запросы

Prometheus предоставляет собственный язык для запросов PromQL. Язык запросов может применять функции и операторы к запросам метрик, фильтровать, группировать по меткам и использовать регулярные выражения для улучшения сопоставления и фильтрации. Результат выражения можно отобразить в виде графика, просмотреть в виде табличных данных в браузере или использовать во внешних системах через HTTP API.

## Оповещение

В Prometheus необходимо установить Alertmanager. Правила оповещения пересылают оповещения менеджеру. Alertmanager заботится о дедупликации оповещений, группировке и отправке получателям. Уведомления можно отправлять по электронной почте, через системы уведомлений по телефону и через чаты.





- ▲ Хорошо подходит для динамических систем, например, Kubernetes. Позволяет автоматически находить необходимые целевые объекты для мониторинга.
- ▲ Формат данных Prometheus поддерживает большое количество приложений. Они сразу выдают все метрики для Prometheus, остается только указать их адреса в настройках.
- ▲ Свой язык запросов (PromQL, Prometheus query language), с помощью которого можно удобно и быстро построить запрос на выборку данных.
- ▲ Отличается высокой производительностью за счет использования TSDB: позволяет собирать данные с тысячи серверов с интервалом в десять секунд.
- ▲ Предусмотрена возможность масштабирования.
- ▲ Простота настройки.
- ▼ В первую очередь рассчитан на краткосрочное хранение метрик (14 дней).
- ▼ Хранит только значения временных рядов и не подходит для хранения текста, логов, журналов событий.
- ▼ Отсутствие аутентификации и авторизации пользователей.
- ▼ Необходимость использования сторонних приложений для визуализации данных.





Система мониторинга включает в себя

- СУБД PostgreSQL;
- Экспортер метрик в Prometheus из PostgreSQL [postgres\\_exporter](#);
- СУБД MongoDB с настроенным набор из трех реплик;
- Экспортер метрик в Prometheus из MongoDB [mongodb\\_exporter](#) (v0.30);
- Prometheus;
- Grafana.



services:

**prometheus:**

image: prom/prometheus

volumes:

- ./prometheus:/etc/prometheus/

command:

- '--config.file=/etc/prometheus/prometheus.yml'

ports:

- 9090:9090 ...

**grafana:**

image: grafana/grafana

ports:

- 3000:3000 ...

**postgres-exporter:**

image: prometheuscommunity/postgres-exporter

ports:

- 9187:9187

environment:

DATA\_SOURCE\_NAME: "CONNECTION\_STRING" ...

**mongo\_exporter:**

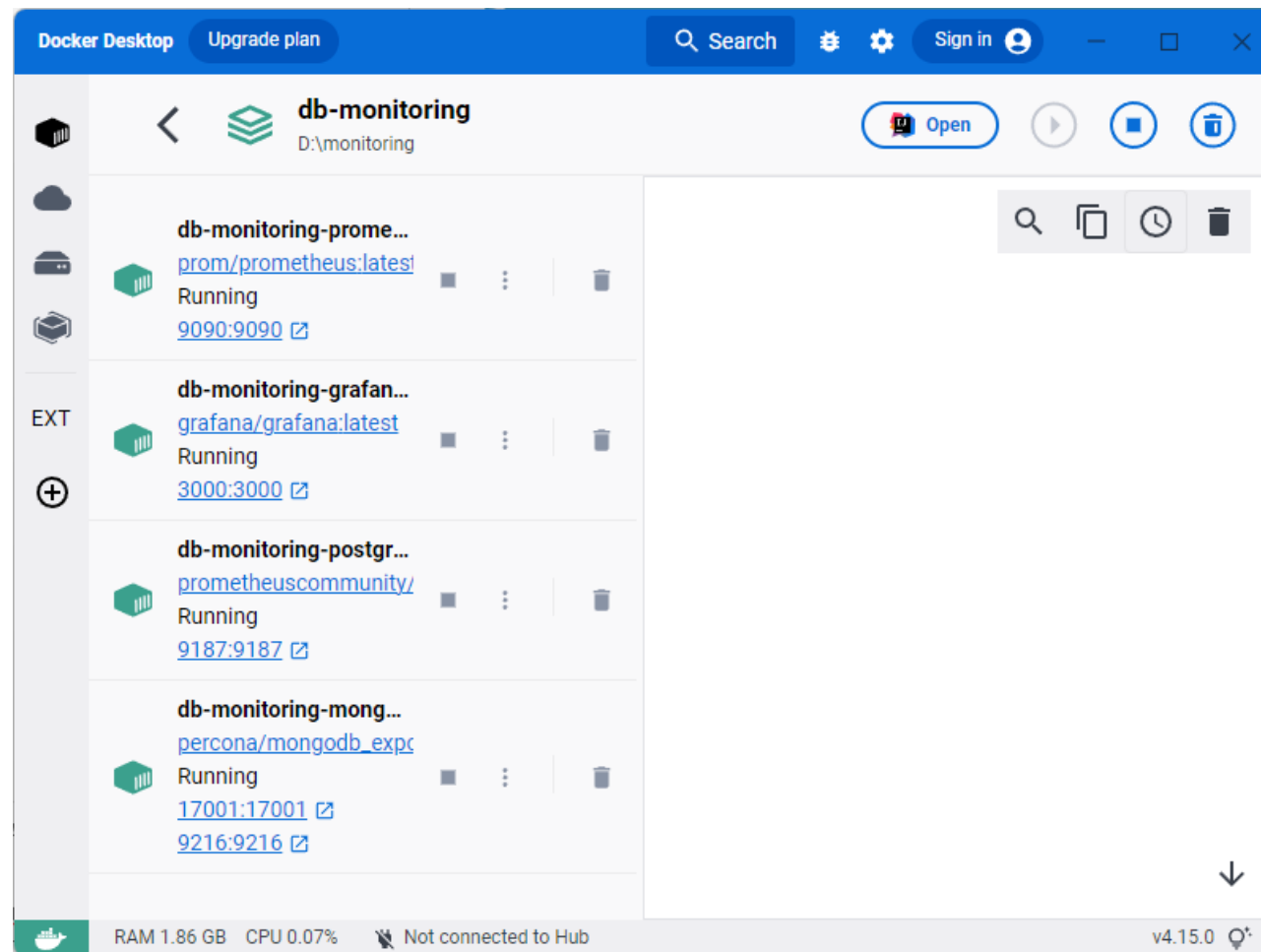
image: percona/mongodb\_exporter:0.30

ports:

- 9216:9216

command:

- "--mongodb.uri=mongodb://CONNECTION\_STRING" ...





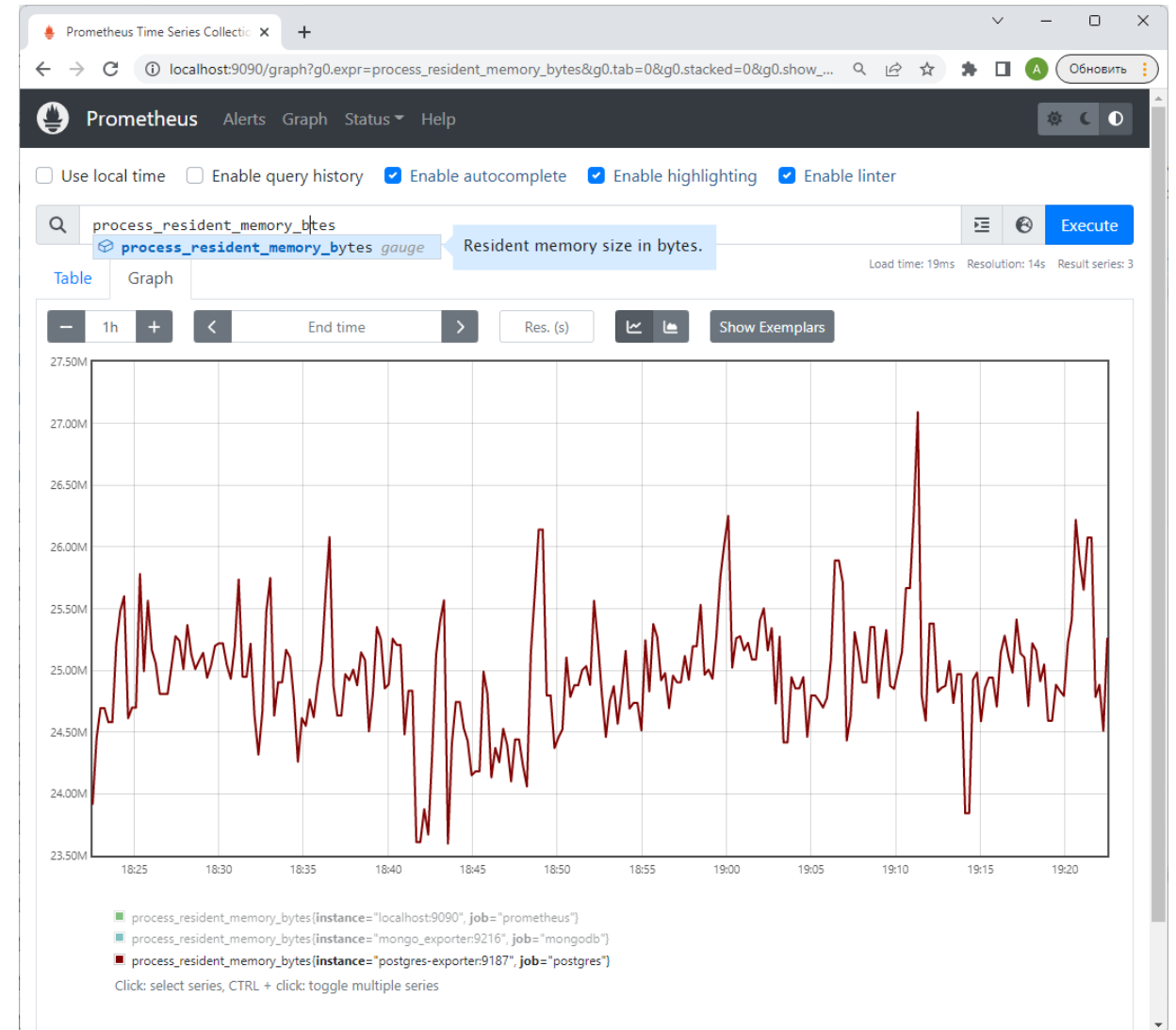
# Конфигурационный файл Prometheus

```
global:
  scrape_interval:      15s
  evaluation_interval: 15s
  external_labels:
    monitor: 'db-monitoring'

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']

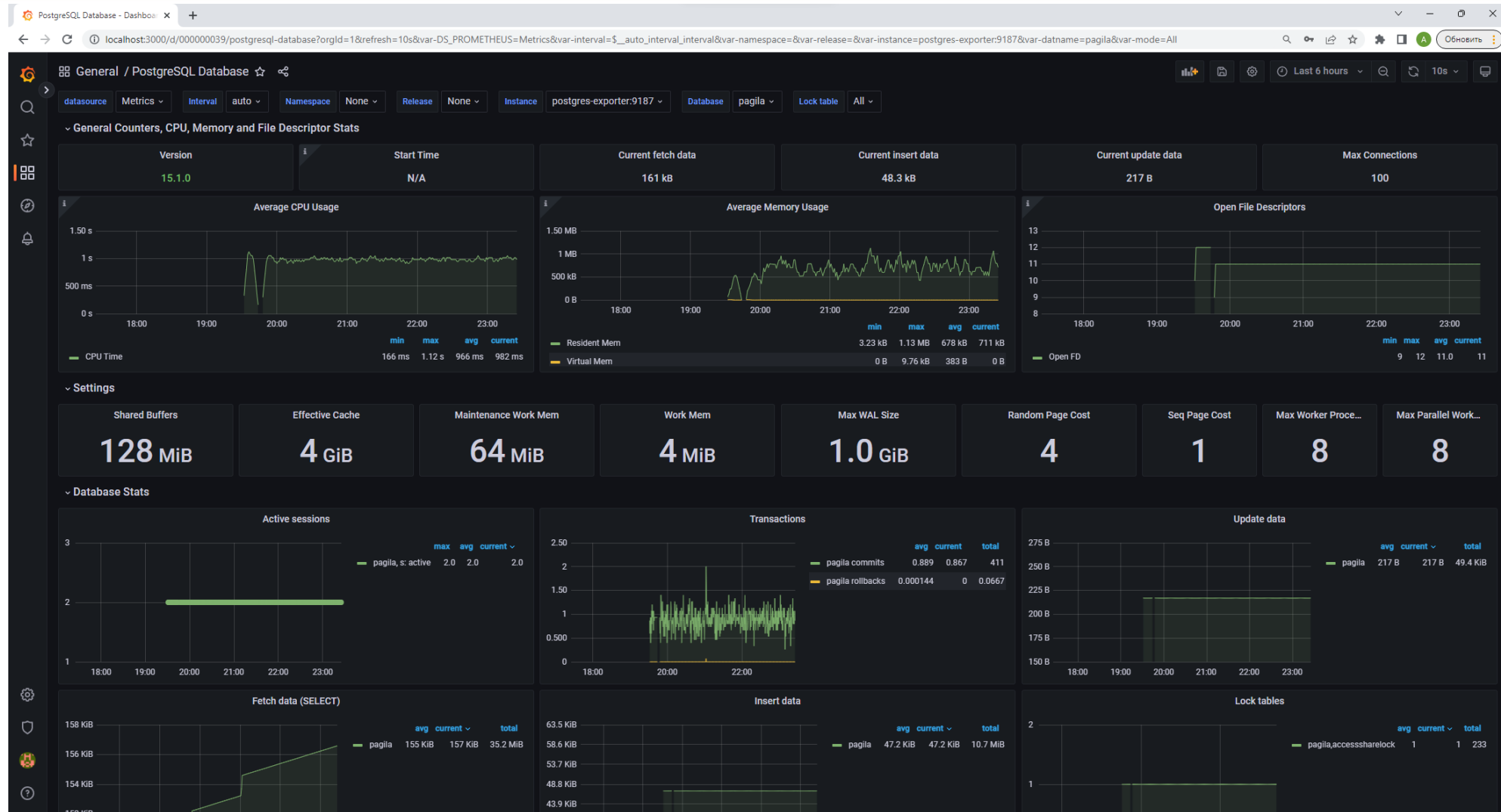
  - job_name: 'postgres'
    static_configs:
      - targets: ['postgres-exporter:9187']

  - job_name: 'mongodb'
    static_configs:
      - targets: ['mongo_exporter:9216']
```





# Визуализация данных в Grafana

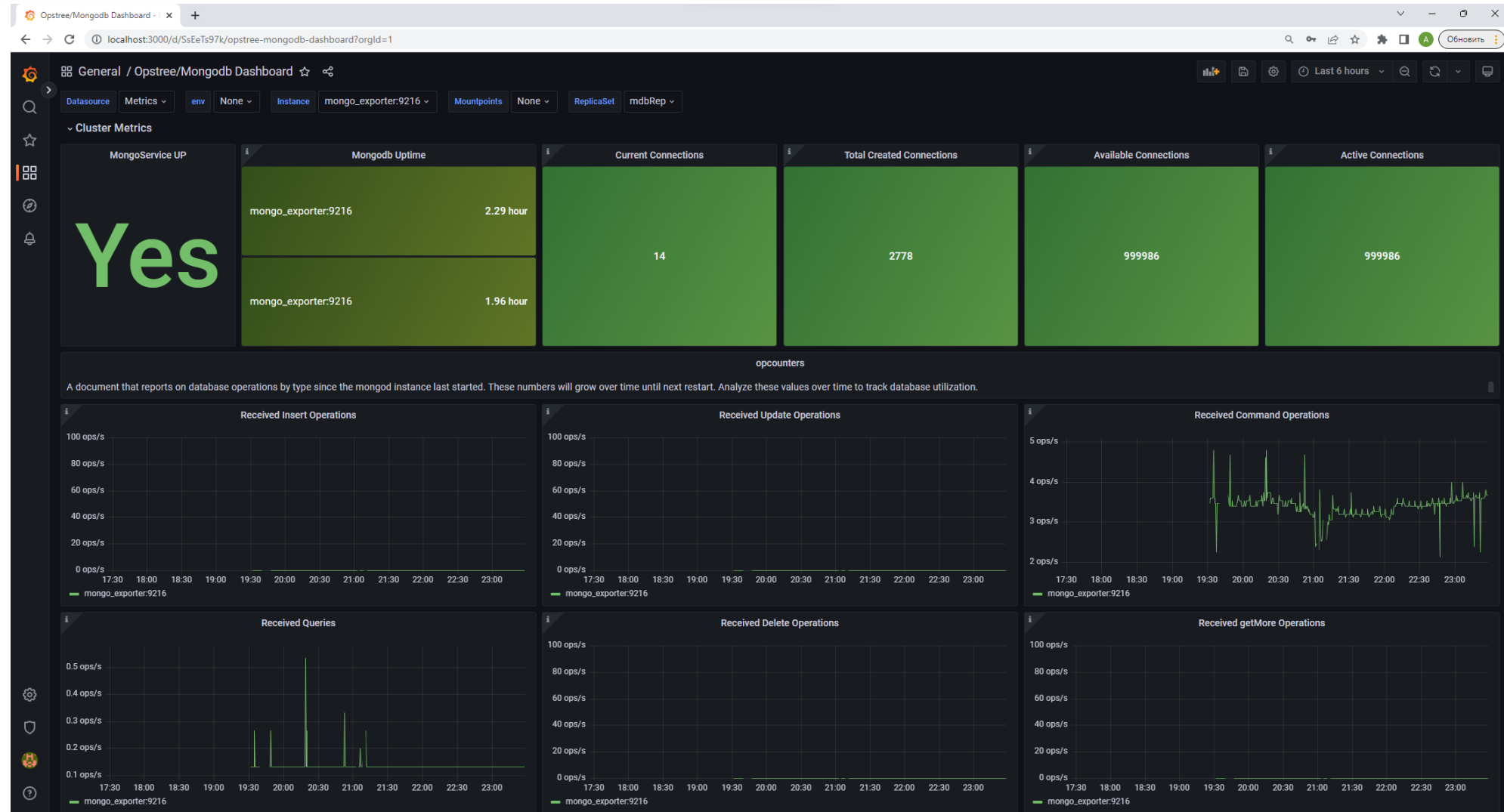


<https://grafana.com/grafana/dashboards/9628-postgresql-database/>





# Визуализация данных в Grafana



<https://grafana.com/grafana/dashboards/16490-opstree-mongodb-dashboard/>



**САМАРСКИЙ** УНИВЕРСИТЕТ  
SAMARA UNIVERSITY

**БЛАГОДАРЮ  
ЗА ВНИМАНИЕ**

Агафонов А.А.  
д.т.н., профессор кафедры ГИИБ