Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Grafana: дашборды мониторинга серверов и сервисов

Как мы строим единый наблюдаемый контур офиса: стек, дашборды, алерты, доступ

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Инфраструктуру до 50 узлов часто мониторят по факту аварии: диск заполнился, служба упала, 1С встала — и об этом узнают от сотрудников, а не от системы. Мы разворачиваем Prometheus + Grafana + Loki с алертами в Telegram, чтобы деградацию (рост latency, утечка RAM, забитый канал, отвал экспортёра) видеть за минуты до остановки сервиса и чинить по графику, а не в аврал.

Почему это важно бизнесу

- Простой файлового сервера или 1С — это остановка отдела и прямые потери в человеко-часах
- Без истории метрик разбор пятничного зависания уходит в понедельник и в догадки
- Ручной просмотр логов не масштабируется: 30+ узлов глазами не проверить
- Раннее оповещение по метрикам ловит деградацию до отказа сервиса — чиним по графику, а не в аврал по жалобе пользователя
- Единый дашборд снимает зависимость от одного «знающего» админа

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

12.x

Ветка Grafana OSS из репозитория apt.grafana.com — встроенный unified alerting и provisioning

grafana.com docs

9100

Порт node_exporter на Linux; windows_exporter слушает 9182, Prometheus — 9090, Grafana — 3000

по докам Prometheus

15 с

scrape_interval и evaluation_interval в global prometheus.yml — баланс детализации и TSDB

наш стандарт

90% / 5m

Базовый порог алерта CPU: IS ABOVE 90, For 5m — отсекает всплески от реальной перегрузки

наш шаблон алертов

1860

ID дашборда Node Exporter Full из библиотеки grafana.com — импорт по ID вместо рисования с нуля

grafana.com/dashboards

2/4/30

vCPU / ГБ RAM / ГБ диска на ВМ мониторинга: хватает на офис до 50 серверов с запасом

наш стандарт

РЕАЛИЗАЦИЯ 1 ИЗ 3

Развёртывание ядра сбора на выделенной ВМ

Что настраиваем

Отдельная виртуалка Debian 12 в контуре заказчика, роль сервера мониторинга

Как мы это делаем

1. Ставим grafana из apt.grafana.com (signed-by keyring), systemctl enable --now grafana-server, порт 3000
2. Prometheus с global scrape_interval 15s, scrape_configs: job 'node' 9100, 'windows' 9182, 'blackbox' /probe http_2xx
3. На узлы раскатываем node_exporter (Linux) и windows_exporter (Windows), сетевое — через snmp_exporter
4. В Grafana Connections → Data sources добавляем Prometheus URL http://localhost:9090, Save & Test до зелёного статуса
5. Grafana прячем за Nginx с TLS Let's Encrypt и basic auth, дефолтный admin/admin меняем сразу

РЕЗУЛЬТАТ

За несколько часов появляется единый контур сбора: все хосты видны в одном месте, метрики CPU/RAM/диск/сеть пишутся в TSDB с историей, к которой можно вернуться при разборе инцидента.

КЛЮЧЕВОЙ НЮАНС

Grafana сама ничего не собирает — это фронт над источником. Держим порядок разделения: сбор в Prometheus/exporter, визуализация и алерты в Grafana.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Дашборды из библиотеки и переменные под парк

Что настраиваем

Слой визуализации Grafana поверх Prometheus, охват всех ролей узлов

Как мы это делаем

1. Импорт Dashboards → Import по ID: 1860 Node Exporter Full, 14510 Windows Exporter, 9628 PostgreSQL
2. Заводим переменную $host: Type=Query, label_values(node_uname_info, instance), Multi-value=on
3. Панели переводим на PromQL с $host: rate(node_cpu_seconds_total{instance="$host",mode!="idle"}[5m])
4. Диск: node_filesystem_avail_bytes / node_filesystem_size_bytes с fstype!~"tmpfs|overlay"
5. Datasource и провайдер дашбордов — в provisioning YAML, сами дашборды JSON под git: ничего не теряется при переезде

РЕЗУЛЬТАТ

Один шаблон обслуживает весь парк: инженер выбирает хост из выпадающего списка вместо десятков однотипных панелей. Новый узел появляется в дашборде автоматически по service discovery, без ручной правки.

КЛЮЧЕВОЙ НЮАНС

Provisioning под git — обязателен для продакшена: восстановление дашбордов и источников становится воспроизводимым, а не ручным кликаньем.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Алерты в Telegram и разграничение доступа

Что настраиваем

Unified alerting Grafana + интеграция AD, для техкоманды и отдельно под клиента

Как мы это делаем

1. Contact point типа Telegram: токен от @BotFather и chat_id через getUpdates, кнопка Test до сообщения в чат
2. Notification policy Default направляем на нужный contact point, разделяем чаты команды и клиента
3. Alert rule на CPU: 100 - avg by(instance)(rate(node_cpu_seconds_total{mode="idle"}[5m]))*100, IS ABOVE 90, For 5m
4. Аннотации с шаблоном: summary=CPU {{ $labels.instance }}, severity=warning в labels для маршрутизации
5. Доступ через auth.ldap на dc01: group_mappings CN=Grafana-Admins → org_role Admin, роли Admin/Editor/Viewer

РЕЗУЛЬТАТ

Деградация приходит пуш-уведомлением на телефон дежурного, а не всплывает жалобой пользователя. Управление ролями уходит в AD-группы: смена админов Grafana — это правка членства группы, а не ручные права в UI.

КЛЮЧЕВОЙ НЮАНС

For 5m критичен: без выдержки алерт срабатывает на секундных всплесках и превращается в шум, который перестают читать. Пороги подбираем под реальную нагрузку сервиса.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Grafana открыта в интернет на 3000**
  Дефолтный admin/admin ловят сканеры за минуты. Прячем за Nginx с TLS и basic auth, меняем пароль до первого выхода наружу.

- ✕ **SQLite как боевая БД Grafana**
  По умолчанию дашборды в SQLite — при росте и нагрузке это узкое место и риск потери. Для продакшена переводим на PostgreSQL с ежедневным бэкапом.

- ✕ **Нет бэкапа дашбордов**
  Кастомные панели живут только в БД инстанса. Экспортируем через provisioning YAML и Grafana HTTP API в git — переезд и восстановление воспроизводимы.

- ✕ **Алерты без выдержки For**
  Порог без For=5m срабатывает на каждом всплеске rate. Поток ложных сообщений в Telegram обесценивает алертинг — команда перестаёт реагировать.

- ✕ **Хардкод имён хостов в панелях**
  Дублирование дашборда под каждый сервер не масштабируется. Делаем переменную $host через label_values и один шаблон на весь парк.

- ✕ **Плагины отстают после мажора**
  После apt upgrade grafana панели с плагинами дают No data — плагин не догнал ядро. Проверяем совместимость на копии до обновления прода.

- ✕ **Нет мониторинга самих экспортёров**
  Упавший node_exporter выглядит как «тишина», а не как алерт. Добавляем правило на up==0, чтобы отвал таргета сам поднимал тревогу.

- ✕ **Один scrape_interval на всё**
  Слишком частый опрос раздувает TSDB, редкий — теряет пики. Разводим интервалы по job: критичное чаще, долгие тренды реже.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Grafana + Prometheus + node_exporter/windows_exporter на выделенной ВМ
- Импорт дашбордов 1860 и 14510 из библиотеки grafana.com
- Алерт CPU/диск/RAM в один Telegram-чат команды
- Смена admin-пароля и закрытие порта 3000 наружу

НОРМАЛЬНО

- Переменная $host и единый шаблон на весь парк узлов
- blackbox_exporter на доступность порталов и 1С (http_2xx)
- Provisioning datasource (YAML) и дашбордов (JSON) под git
- For=5m и severity-метки на всех alert rules

ХОРОШО

- Loki для централизованных логов рядом с метриками
- Grafana на PostgreSQL с ежедневным бэкапом БД
- SSO через LDAP/AD (или OIDC/Keycloak) с ролями по группам
- snmp_exporter на сетевое оборудование, up==0 на все таргеты

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Grafana закрыта TLS и basic auth, порт 3000 не торчит в интернет, дефолтный пароль сменён?
- [ ] Данные Grafana на PostgreSQL, а не на дефолтном SQLite, и настроен ежедневный бэкап?
- [ ] Дашборды и datasource описаны в provisioning YAML и лежат в git?
- [ ] На всех ролях узлов стоят экспортёры: node 9100, windows 9182, сетевое через snmp?
- [ ] Есть alert-правило up==0 на отвал самих экспортёров и таргетов?
- [ ] У всех alert rules выставлен For (например 5m) и метка severity для маршрутизации?
- [ ] Contact point Telegram прошёл кнопку Test и реально доставляет сообщения?
- [ ] Используется переменная $host — один дашборд на весь парк, без хардкода имён?
- [ ] Доступ разграничен ролями Admin/Editor/Viewer, для команды поднят LDAP/AD или OIDC?
- [ ] Совместимость дашбордов и плагинов проверена на копии до мажорного обновления Grafana?

Если хотя бы на два вопроса ответ «нет» или «не знаю» — тема требует внимания.

ЧТО МЫ ДЕЛАЕМ

Как поможет ITFresh

ITFresh — ИТ-аутсорсинг для юридических лиц до 50 рабочих мест в Москве и области. 15+ лет практики, собственная инфраструктура в дата-центре МТС (8 серверов Dell Xeon Platinum).

- Разворачиваем стек Grafana + Prometheus + Loki на выделенной ВМ в вашем контуре под ключ
- Ставим экспортёры под Linux/Windows/сетевое и настраиваем service discovery на весь парк
- Собираем кастомные дашборды под ваши бизнес-сервисы: 1С, БД, файловые, шлюзы
- Настраиваем алерты в Telegram с порогами под вашу нагрузку и разделением команда/клиент
- Заводим доступ через AD/LDAP, provisioning в git, бэкап и регламент обновлений

15+
лет в ИТ-поддержке

50
рабочих мест — наш профиль

МТС
дата-центр, Москва

КОНТАКТЫ

Обсудить вашу задачу

Сайт	**itfresh.ru**
Телефон	**+7 903 729-62-41**
Telegram	**@ITfresh_Boss**

Бесплатно посмотрим вашу инфраструктуру по этому чек-листу и скажем, где тонко — без обязательств.

itfresh.ru

ДОКУМЕНТАЦИЯ И НАШ СТЕК

Техническая база

01 Grafana OSS: установка из APT-репозитория Debian (grafana.com — 12.x)

02 Grafana Alerting: contact points, notification policies, rules (grafana.com — 12.x)

03 Grafana: provisioning dashboards and data sources (YAML) (grafana.com — 12.x)

04 Grafana: LDAP/AD authentication (auth.ldap, group_mappings) (grafana.com — 12.x)

05 Prometheus: configuration, scrape_configs, scrape_interval (prometheus.io — 3.x)

06 Prometheus: monitoring Linux host with Node Exporter (prometheus.io — 1.9.x)

07 Grafana Labs Dashboards: Node Exporter Full (ID 1860) (grafana.com — 2024)

08 Наш шаблон дашбордов, алертов и provisioning для парка (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.