Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Prometheus и Grafana с нуля: мониторинг серверов офиса

Как мы разворачиваем стек метрик и дашборд нагрузки для компаний до 50 рабочих мест

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

В компаниях до 50 РМ серверы обычно живут без метрик: о нехватке места на диске узнают по звонку бухгалтера, а о деградации сети — постфактум из логов. Мы разворачиваем Prometheus + node_exporter + Grafana + Alertmanager как единый стек: сбор метрик каждые 15 секунд, хранение 15-30 дней, один дашборд CPU/память/диск/сеть и алерты в Telegram до сбоя сервиса.

Почему это важно бизнесу

- Диск заполняется тихо — 1С, почта и бэкапы падают без предупреждения администратора
- Без истории нагрузки невозможно доказать клиенту, нужен ли апгрейд сервера или дело в приложении
- Ручной обход серверов раз в неделю не ловит краткие всплески CPU/сети перед сбоем
- Инциденты без метрик расследуются по логам часами вместо графика за 5 минут
- Заказчик хочет видеть отчёт о нагрузке, а не верить на слово

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

15s

scrape_interval для node_exporter в prometheus.yml

prometheus.io/docs, наш стандарт

9100

порт node_exporter по умолчанию, путь /metrics

github.com/prometheus/node_exporter

15-30d

storage.tsdb.retention.time для локального хранения на VPS/сервере

prometheus.io/docs/storage

15/10%

порог warning/critical свободного места (node_filesystem_avail_bytes)

наш стандарт алертов

5m

окно rate() для node_cpu_seconds_total и network-метрик

prometheus.io PromQL best practices

30s/5m/4h

group_wait/group_interval/repeat_interval в Alertmanager

prometheus.io/docs/alerting

РЕАЛИЗАЦИЯ 1 ИЗ 3

Базовый стек на одном сервере 1С и файловой шары

Что настраиваем

Торговая компания, 35 РМ, один физический сервер Windows + Linux-VM под 1С и файлы

Как мы это делаем

1. Ставим node_exporter на Linux-хосты (systemd unit) и windows_exporter на Windows-сервер
2. Поднимаем Prometheus в Docker с prometheus.yml: job node, scrape_interval 15s, retention.time 30d
3. Grafana: data source Prometheus по proxy-доступу, импортируем дашборд Node Exporter Full
4. Настраиваем алерты через Alertmanager: диск <15%, память <10%, недоступность target > 2m
5. Подключаем receiver webhook в Telegram-бота компании для мгновенных уведомлений

РЕЗУЛЬТАТ

За первый месяц дашборд показал рост /var на 4% в сутки из-за логов 1С — почистили и настроили logrotate до переполнения диска, которое раньше случалось раз в квартал внепланово.

КЛЮЧЕВОЙ НЮАНС

Без node_filesystem_avail_bytes и алерта на 15% свободного места инцидент с диском всегда узнают по звонку пользователя, а не по графику.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Дашборд сети и памяти для терминального сервера RDS

Что настраиваем

Юридическая фирма, 22 PM, терминальный сервер с 15+ одновременными сессиями

Как мы это делаем

1. Собираем node_network_receive_bytes_total и node_network_transmit_bytes_total по интерфейсам
2. Строим панель rate() за 5m с group by instance и device, разносим по интерфейсам LAN/WAN
3. Добавляем node_memory_MemAvailable_bytes и node_load1/5/15 на одну строку дашборда
4. Задаём recording rule для агрегата нагрузки по часам, чтобы не пересчитывать тяжёлый PromQL на каждый рендер
5. Прогоняем 2 недели без алертов, чтобы откалибровать пороги под реальный трафик офиса

РЕЗУЛЬТАТ

Обнаружили, что пиковая утилизация исходящего канала совпадает с резервным копированием днём, а не ночью как считал клиент — перенесли backup-окно и убрали жалобы на "тормозит RDP".

КЛЮЧЕВОЙ НЮАНС

Пороги алертов нельзя ставить по умолчанию из шаблона — две недели наблюдения дают реальную базовую линию нагрузки конкретного офиса.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Мульти-сервер мониторинг для клиента с 3 площадками

Что настраиваем

Производственная компания, 48 РМ, 3 сервера на разных физических площадках

Как мы это делаем

1. Один центральный Prometheus с тремя job в scrape_configs, static_configs с labels site=офис/склад/цех
2. Federation не потребовался — трафик метрик небольшой, тянем напрямую по VPN до каждого узла
3. В Grafana делаем dashboard variable $site и $instance, чтобы переключать площадку одним кликом
4. Настраиваем inhibit_rules в Alertmanager: если недоступен весь узел, не шлём отдельные алерты по CPU/диску этого узла
5. Документируем runbook: что означает каждый алерт и что делать инженеру на дежурстве

РЕЗУЛЬТАТ

После внедрения inhibit_rules число уведомлений в Telegram при обрыве VPN упало с 12 дублирующих алертов до одного понятного — инженер сразу видит первопричину, а не список симптомов.

КЛЮЧЕВОЙ НЮАНС

Без inhibit_rules любой сетевой сбой на площадке превращается в лавину алертов, в которой теряется реальная причина.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Scrape_interval 5-10 секунд без нужды**
  Слишком частый опрос раздувает TSDB и retention съедает диск быстрее — для офисного сервера хватает 15s.

- ✕ **Prometheus без storage.tsdb.retention.time**
  По умолчанию 15d, но без явного флага в докер-конфиге его легко потерять при пересборке контейнера.

- ✕ **Дашборд без node_filesystem_readonly**
  Файловая система, ушедшая в read-only при сбое диска, не покажет падение свободного места — нужен отдельный алерт.

- ✕ **Алерты без for в rules**
  Без параметра for единичный всплеск CPU на 10 секунд шлёт ложное уведомление вместо устойчивой проблемы.

- ✕ **Один receiver на все severity**
  Critical и warning в одном канале Telegram тонут — критичные алерты нужно выделять route и отдельной группой.

- ✕ **Grafana data source без Access: Server**
  Proxy-доступ через сервер Grafana надёжнее прямого browser-access, который ломается за NAT и в закрытых сетях клиента.

- ✕ **rate() на слишком коротком окне**
  Окно rate() короче 4x scrape_interval даёт рваный график — для 15s-опроса берём минимум 5m.

- ✕ **Нет node_time_seconds для дрейфа часов**
  Рассинхронизация времени между сервером и Prometheus искажает все rate()-графики и алерты по задержке.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Prometheus + node_exporter на одном сервере, retention.time 15d
- Один дашборд Grafana: CPU, память, диск, сеть по node_exporter
- Алерт на диск <15% и недоступность сервера через Alertmanager

НОРМАЛЬНО

- Retention 30d, Alertmanager с группировкой по severity и Telegram-каналом
- Recording rules для тяжёлых агрегатов, чтобы не грузить дашборд на каждый рендер
- Dashboard variables $instance для мульти-сервер клиентов
- Runbook с описанием каждого алерта для дежурного инженера

ХОРОШО

- Inhibit_rules и maintenance-окна для плановых работ без ложных алертов
- Federation или remote_write при росте числа серверов свыше 10-15
- Разделение критичных и информационных receiver с эскалацией по времени
- Регулярный пересмотр порогов на основе накопленной истории метрик

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] node_exporter установлен как systemd unit и стартует при перезагрузке
- [ ] prometheus.yml содержит job для каждого сервера с корректным static_configs
- [ ] storage.tsdb.retention.time явно задан в command-line флагах Prometheus
- [ ] Grafana data source Prometheus настроен через Access: Server (proxy)
- [ ] Импортирован и адаптирован дашборд с панелями CPU/память/диск/сеть
- [ ] Alerting rules покрывают диск, память, недоступность target и load average
- [ ] Alertmanager route разделяет severity и ведёт в рабочий канал уведомлений
- [ ] Пороги алертов откалиброваны минимум по 2 неделям реальной нагрузки
- [ ] Есть runbook: что означает алерт и какие действия предпринимает инженер
- [ ] Backup конфигурации Prometheus/Grafana (YAML, дашборды) хранится в git

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 Prometheus + node_exporter + Grafana + Alertmanager под ключ за 1-2 дня
- Настраиваем дашборд под реальную инфраструктуру клиента, а не типовой шаблон
- Калибруем пороги алертов по факту наблюдений, а не по умолчанию из интернета
- Подключаем уведомления в Telegram компании с разделением critical/warning
- Обслуживаем стек на регулярной основе: обновления, retention, ротация дашбордов

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 Configuration (prometheus.yml, scrape_configs) (prometheus.io — 2026)

02 Command-line flags: storage.tsdb.retention (prometheus.io — 2026)

03 Storage (prometheus.io — 2026)

04 node_exporter README и CHANGELOG (github.com — 2026)

05 Configure the Prometheus data source (grafana.com — 2026)

06 Alertmanager configuration (prometheus.io — 2026)

07 Grafana support for Prometheus (prometheus.io — 2026)

08 Наш шаблон prometheus.yml + alert.rules.yml (внутренний стандарт ITfresh — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.