??????? ???????????: ?????????? ? ????? ??????

Эта статья разбирает архитектуру системы мониторинга — из чего она состоит и как через неё проходят данные. Настройка локаций, интервалы опроса, правила интерпретации SNMP-параметров и конструктор отчётов.

???? ? ???????

Мониторинг — ядро Printum. В отличие от ПринтМенеджера, его задача не обработка заданий, а сбор, обработка и хранение данных: об устройствах печати, о пользователях, статистике, событиях в системе. Мониторинг всегда существует в инфраструктуре в единственном экземпляре, даже если ПринтМенеджеров несколько (см. статьи про балансировку и филиальную сеть).

?????????? ?????

Архитектурный блок «Система мониторинга» связан со следующими логическими блоками:

- **Сервер с системой мониторинга и личный кабинет** — центральный компонент, разбирается ниже подробно.
- **Сетевой агент** — обычно устанавливается в одном экземпляре вместе с основными компонентами мониторинга на том же сервере, но при необходимости дополнительные агенты выносятся на отдельные серверы.
- **АРМы сотрудников** — на них устанавливается локальный агент мониторинга (при необходимости). Связан частично, поскольку сам АРМ не входит в состав системы мониторинга, но локальный агент входит.
- **Блок внешних интеграций** — с доменом (LDAP-каталогом), почтовым сервером, SIEM-системой и пр.

?????????? ??????? ???????????

Компонент	Тип	Роль

PostgreSQL	БД	Структурированные обработанные данные: пользователи, настройки системы, статистика (результат интерпретации “сырых” SNMP-данных).
ClickHouse	БД	Сырые данные, поступившие от сетевого агента (SNMP-опрос МФУ). Выбран вместо PostgreSQL, потому что может быстро сохранять большие объемы данных.
Redis	БД (используется как брокер)	Не хранит бизнес-данные — используется как брокер сообщений между Django, Celery и планировщиком, а также используется как кэш данных.
Django	Python-приложение	Бэкенд: принимает запросы через nginx, формирует ответы для личного кабинета и панели администратора, сетевого и локального агентов, кладёт данные в PostgreSQL/ClickHouse.
Celery	Python-приложение (фоновые задачи)	Обработка операций, которые занимают от 1 секунды: разбор “сырых” SNMP-данных после опроса, синхронизация с доменом, формирование тяжёлых отчётов.
Планировщик (scheduler)	Python-приложение (планировка фоновых задач)	Планирует периодические задачи: синхронизация с доменом, опрос устройств, отправка отчётов и уведомлений на почту. В нужный момент кладёт задачу в Redis, а тот передаёт её Celery.
nginx	Реверс-прокси	Маршрутизирует запросы от браузера и других компонентов, направляет их в Django.

?????? ?????? ??????: ????? ??? ? ?????? ???????

Это базовый сценарий, на котором стоит тренироваться читать схему.

1. Сетевой агент получает две периодические задачи: сканирование сети и опрос принтеров.
2. Агент отправляет SNMP-запрос МФУ (порт 161/UDP) с запросом на нужные OID.
3. МФУ отвечает сетевому агенту (порт 162/UDP) значениями запрошенных OID.
4. Сетевой агент передаёт полученные сырые данные через nginx в Django.

5. Django видит, что пришли сырые данные, и кладёт их в ClickHouse.
6. Scheduler по расписанию запускает задачу разбора сырых данных, ставя задачу в Redis. Celery забирает задачу из Redis.
7. Celery забирает сырые данные из ClickHouse, интерпретирует их и кладёт уже структурированные, готовые к отображению данные в PostgreSQL.
8. Пользователь через браузер запрашивает страницу устройств.
9. Запрос проходит через nginx в Django; Django идёт в PostgreSQL за данными.
10. Django готовит ответ, nginx собирает его вместе с шаблоном личного кабинета и отдаёт в браузер.

Если что-то в отчёте выглядит неактуальным или неполным — эта цепочка подсказывает, где искать: не дошли ли сырые данные до ClickHouse, выполнилась ли задача Celery, попали ли обработанные данные в PostgreSQL, или проблема на уровне отображения (nginx/Django).

????????? ????? ??????????? (?????? ? ???)

Отдельный поток — сбор статистики о заданиях печати, отправленных с АРМ на локальный (не сетевой) принтер:

1. Локальный агент на АРМ фиксирует данные о задании.
2. Агент отправляет накопленные данные через nginx в Django.
3. Django обрабатывает данные и кладёт их в PostgreSQL.
4. Администратор через личный кабинет запрашивает эти данные обратно тем же путём, что и в примере выше.

??????? ???????

- **LDAP-каталог (домен)** — планировщик инициирует задачу выгрузки данных о пользователях, Celery запрашивает домен через Django/nginx, ответ обрабатывается и попадает в PostgreSQL. Подробности логики синхронизации пользователей и групп — в статье модуля «Управление пользователями».
- **Почтовый сервер (SMTP)** — используется для отправки уведомлений и подписки на отчёты. Django формирует письмо (например, с PIN-кодом или отчётом), nginx отправляет его на SMTP-сервер.
- **SIEM-система (Syslog)** — планировщик инициирует выгрузку накопленных в PostgreSQL событий, Celery готовит пакет, Django/nginx отправляют его во внешнюю систему по протоколу Syslog.

?????????????? ? ??????????????

ПринтМенеджер и мониторинг обмениваются данными в обе стороны.

Каждая из систем выступает инициатором соединения для получения данных, которые нужны именно ей:

- Мониторинг запрашивает данные по напечатанным заданиям через nginx → Django ПринтМенеджер → PostgreSQL — именно поэтому статистика по заданиям печати видна в мониторинге отдельно от статистики по устройствам.
- В обратную сторону, тем же каналом, ПринтМенеджер запрашивает, а мониторинг передаёт данные о пользователях и устройствах.

Revision #1
Created 2026-07-12 14:56:40 UTC by DD
Updated 2026-07-12 15:01:25 UTC by DD