



## Можешь запомнить это в контексте этой бесыды чтобы потом быстро восстановить контекст? Когда спрошу про метрики.

Пока хотел бы вернуться к ежедневным операциям.

Что скажешь если настроим это через pg\_cron ?

✓ **Запомнил!** Контекст по мониторингу сохранён — трёхуровневая система (Java-метрики, PostgreSQL views/functions, Grafana дашборды + алерты).<sup>[1]</sup> <sup>[2]</sup>

### □ pg\_cron для ежедневных операций — отличная идея!

Преимущества pg\_cron перед system cron:

✓ **Плюсы:**

- Всё внутри PostgreSQL — не нужен доступ к ОС
- Логирование встроенное (видно в cron.job\_run\_details)
- Проще управление в Kubernetes/облаке (нет зависимости от cron на хосте)
- Транзакционность — можно использовать все функции БД
- Централизованное управление через SQL
- Видно историю выполнения задач
- Работает в managed PostgreSQL (AWS RDS, Azure, GCP)

✗ **Минусы:**

- Требует установки расширения pg\_cron
- Использует системные ресурсы PostgreSQL (не критично для ваших задач)
- Нужны права superuser для установки

### □ Реализация через pg\_cron

## **1. Установка и настройка pg\_cron**

## Установка расширения:

```
-- От имени superuser  
CREATE EXTENSION IF NOT EXISTS pg_cron;
```

## **Настройка postgresql.conf:**

```
# Добавить в postgresql.conf
shared_preload_libraries = 'pg_cron'
cron.database_name = 'ksk_database' # ваша БД

# Опционально: настройка логирования
cron.log_run = on
cron.log_statement = on
```

## **После изменений:**

```
# Рестарт PostgreSQL  
systemctl restart postgresql
```

## 2. SQL скрипт настройки всех задач

```
-- НАСТРОЙКА ЕЖЕДНЕВНЫХ ЗАДАЧ ОБСЛУЖИВАНИЯ КСК ЧЕРЕЗ pg_cron
-- =====
-- Дата: 2025-10-28
-- Описание: Автоматизация всех maintenance задач через pg_cron
-- =====

-- =====
-- ОЧИСТКА СТАРЫХ ЗАДАЧ (опционально, если перенастраиваете)
-- =====

DO $$
DECLARE
    rec RECORD;
BEGIN
    FOR rec IN SELECT jobid FROM cron.job WHERE jobname LIKE 'ksk_%'
    LOOP
        PERFORM cron.unschedule(rec.jobid);
    END LOOP;
END $$;

-- =====
-- ЗАДАЧА #1: ANALYZE вчерашних партиций (00:30)
-- =====

SELECT cron.schedule(
    'ksk_analyze_yesterday_partitions', -- job name
    '30 0 * * *', -- cron schedule
```

```

$$
DO $job$
DECLARE
    v_date TEXT := TO_CHAR(CURRENT_DATE - 1, 'YYYYMMDD');
BEGIN
    EXECUTE 'ANALYZE upoa_ksk_reports.part_ksk_result_' || v_date;
    EXECUTE 'ANALYZE upoa_ksk_reports.part_ksk_figurant_' || v_date;
    EXECUTE 'ANALYZE upoa_ksk_reports.part_ksk_match_' || v_date;

    -- Логирование
    PERFORM upoa_ksk_reports.ksk_log_operation(
        'analyze_partitions',
        'system',
        'success',
        'Analyzed partitions for date: ' || v_date,
        NULL
    );
EXCEPTION
    WHEN OTHERS THEN
        PERFORM upoa_ksk_reports.ksk_log_operation(
            'analyze_partitions',
            'system',
            'error',
            'Failed to analyze partitions for date: ' || v_date,
            SQLERRM
        );
        RAISE;
END $job$;
$$
);

-- =====
-- ЗАДАЧА #2: Создание будущих партиций (01:00)
-- =====

SELECT cron.schedule(
    'ksk_create_future_partitions',
    '0 1 * * *',
    $$
    SELECT upoa_ksk_reports.ksk_create_partitions_for_all_tables(
        CURRENT_DATE,
        7
    );
    $$
);

-- =====
-- ЗАДАЧА #3: Генерация системных отчётов (01:30)
-- =====

SELECT cron.schedule(
    'ksk_generate_system_reports',
    '30 1 * * *',
    $$
DO $job$
DECLARE
    rec RECORD;
    v_report_id INTEGER;

```

```

BEGIN
    FOR rec IN
        SELECT report_code
        FROM upoa_ksk_reports.ksk_report_orchestrator
        ORDER BY report_code
    LOOP
        BEGIN
            -- Генерация отчёта
            v_report_id := upoa_ksk_reports.ksk_run_report(
                rec.report_code,
                'system'
            );

            -- Логирование успеха
            PERFORM upoa_ksk_reports.ksk_log_operation(
                'generate_report',
                rec.report_code,
                'success',
                'Report generated with ID: ' || v_report_id,
                NULL
            );
        EXCEPTION
            WHEN OTHERS THEN
                -- Логирование ошибки
                PERFORM upoa_ksk_reports.ksk_log_operation(
                    'generate_report',
                    rec.report_code,
                    'error',
                    'Failed to generate report',
                    SQLERRM
                );
        END;
    END LOOP;
END $job$;
$$
);

-- =====
-- ЗАДАЧА #4: Удаление прошлогодних партиций (02:00)
-- =====

SELECT cron.schedule(
    'ksk_drop_old_partitions',
    '0 2 * * *',
    $$
    SELECT upoa_ksk_reports.ksk_drop_old_partitions(365);
    $$
);

-- =====
-- ЗАДАЧА #5: Удаление empty записей (03:00)
-- =====

SELECT cron.schedule(
    'ksk_cleanup_empty_records',
    '0 3 * * *',
    $$
    SELECT upoa_ksk_reports.ksk_cleanup_empty_records(14);

```

```

$$
);

-- =====
-- ЗАДАЧА #6: Удаление пустыхパーティций (03:30)
-- =====

SELECT cron.schedule(
    'ksk_cleanup_empty_partitions',
    '30 3 * * *',
    $$
    SELECT upoa_ksk_reports.ksk_cleanup_empty_partitions('all', 14);
    $$
);

-- =====
-- ЗАДАЧА #7: Очистка старых отчётов (04:00)
-- =====

SELECT cron.schedule(
    'ksk_cleanup_old_reports',
    '0 4 * * *',
    $$
    SELECT upoa_ksk_reports.ksk_cleanup_old_reports();
    $$
);

-- =====
-- ЗАДАЧА #8: Очистка системных логов (04:30)
-- =====

SELECT cron.schedule(
    'ksk_cleanup_old_logs',
    '30 4 * * *',
    $$
    SELECT upoa_ksk_reports.ksk_cleanup_old_logs(365);
    $$
);

-- =====
-- ЗАДАЧА #9: VACUUM главных таблиц (05:00)
-- =====

SELECT cron.schedule(
    'ksk_vacuum_main_tables',
    '0 5 * * *',
    $$
    DO $job$
    BEGIN
        VACUUM ANALYZE upoa_ksk_reports.ksk_result;
        VACUUM ANALYZE upoa_ksk_reports.ksk_figurant;
        VACUUM ANALYZE upoa_ksk_reports.ksk_match;
        VACUUM ANALYZE upoa_ksk_reports.ksk_report_header;
        VACUUM ANALYZE upoa_ksk_reports.ksk_system_operations_log;

        -- Логирование
        PERFORM upoa_ksk_reports.ksk_log_operation(
            'vacuum_tables',
            'system',
            'success',

```

```

        'VACUUM ANALYZE completed for main tables',
        NULL
    );
EXCEPTION
    WHEN OTHERS THEN
        PERFORM upoa_ksk_reports.ksk_log_operation(
            'vacuum_tables',
            'system',
            'error',
            'VACUUM ANALYZE failed',
            SQLERRM
        );
END $job$;
$$
);

-- =====
-- ЗАДАЧА #10: Мониторинг bloat (воскресенье 04:00)
-- =====

SELECT cron.schedule(
    'ksk_monitor_bloat',
    '0 4 * * 0', -- 0 = воскресенье
    $$
    SELECT upoa_ksk_reports.ksk_monitor_table_bloat();
    $$
);

-- =====
-- ВЕРИФИКАЦИЯ: Проверка созданных задач
-- =====

SELECT
    jobid,
    schedule,
    command,
    nodename,
    nodeport,
    database,
    username,
    active,
    jobname
FROM cron.job
WHERE jobname LIKE 'ksk_%'
ORDER BY schedule;

COMMENT ON EXTENSION pg_cron IS
'PostgreSQL job scheduler for KSK maintenance tasks';

```

### 3. Мониторинг выполнения задач

**Просмотр статуса всех задач:**

```
-- Список активных задач
SELECT
    jobid,
```

```
jobname,  
schedule,  
active,  
database  
FROM cron.job  
WHERE jobname LIKE 'ksk_%'  
ORDER BY jobname;
```

#### История выполнения (последние 20):

```
-- История запусков задач  
SELECT  
    jr.jobid,  
    j.jobname,  
    jr.runid,  
    jr.job_pid,  
    jr.database,  
    jr.username,  
    jr.command,  
    jr.status,  
    jr.return_message,  
    jr.start_time,  
    jr.end_time,  
    jr.end_time - jr.start_time AS duration  
FROM cron.job_run_details jr  
JOIN cron.job j ON j.jobid = jr.jobid  
WHERE j.jobname LIKE 'ksk_%'  
ORDER BY jr.start_time DESC  
LIMIT 20;
```

#### Задачи с ошибками за последние 24 часа:

```
SELECT  
    j.jobname,  
    jr.status,  
    jr.return_message,  
    jr.start_time,  
    jr.end_time - jr.start_time AS duration  
FROM cron.job_run_details jr  
JOIN cron.job j ON j.jobid = jr.jobid  
WHERE j.jobname LIKE 'ksk_%'  
    AND jr.status = 'failed'  
    AND jr.start_time > NOW() - INTERVAL '24 hours'  
ORDER BY jr.start_time DESC;
```

#### Статистика выполнения по задачам:

```
SELECT  
    j.jobname,  
    COUNT(*) AS total_runs,  
    SUM(CASE WHEN jr.status = 'succeeded' THEN 1 ELSE 0 END) AS successful_runs,  
    SUM(CASE WHEN jr.status = 'failed' THEN 1 ELSE 0 END) AS failed_runs,
```

```
ROUND(AVG(EXTRACT(EPOCH FROM (jr.end_time - jr.start_time))), 2) AS avg_duration_sec,
MAX(jr.end_time) AS last_run
FROM cron.job j
LEFT JOIN cron.job_run_details jr ON j.jobid = jr.jobid
WHERE j.jobname LIKE 'ksk_%'
AND jr.start_time > NOW() - INTERVAL '7 days'
GROUP BY j.jobname
ORDER BY j.jobname;
```

## 4. Управление задачами

### Временное отключение задачи:

```
-- Отключить задачу
SELECT cron.unschedule(jobid)
FROM cron.job
WHERE jobname = 'ksk_vacuum_main_tables';

-- Включить обратно (пересоздать)
SELECT cron.schedule(
    'ksk_vacuum_main_tables',
    '0 5 * * *',
    $ <SQL код> $
);
```

### Изменение расписания:

```
-- Изменить время выполнения
UPDATE cron.job
SET schedule = '0 6 * * *' -- новое время
WHERE jobname = 'ksk_vacuum_main_tables';
```

### Ручной запуск задачи:

```
-- Запустить задачу вручную (выполняется немедленно)
SELECT cron.schedule(
    'ksk_manual_vacuum', -- времменное имя
    '* * * * *', -- каждую минуту
    $$           --
    VACUUM ANALYZE upoa_ksk_reports.ksk_result;
    $$           --
);

-- Через минуту удалить
SELECT cron.unschedule(jobid)
FROM cron.job
WHERE jobname = 'ksk_manual_vacuum';
```

## 5. View для мониторинга pg\_cron задач

```
-- =====
-- VIEW: ksk_cron_monitoring
-- Сводная информация о pg_cron задачах КСК
-- =====
CREATE OR REPLACE VIEW upoa_ksk_reports.ksk_cron_monitoring AS
SELECT
    j.jobid,
    j.jobname,
    j.schedule,
    j.active,
    COALESCE(last_run.start_time, '1970-01-01'::timestamp) AS last_run_time,
    COALESCE(last_run.status, 'never_run') AS last_status,
    COALESCE(last_run.return_message, 'N/A') AS last_message,
    COALESCE(EXTRACT(EPOCH FROM (last_run.end_time - last_run.start_time)), 0) AS last_duration

    -- Статистика за 7 дней
    stats.total_runs_7d,
    stats.failed_runs_7d,
    stats.avg_duration_7d_sec,

    -- Статус здоровья
    CASE
        WHEN NOT j.active THEN 'DISABLED'
        WHEN last_run.status = 'failed' THEN 'ERROR'
        WHEN last_run.start_time < NOW() - INTERVAL '25 hours' THEN 'STALE'
        WHEN last_run.status = 'succeeded' THEN 'OK'
        ELSE 'UNKNOWN'
    END AS health_status

FROM cron.job j
LEFT JOIN LATERAL (
    SELECT start_time, end_time, status, return_message
    FROM cron.job_run_details
    WHERE jobid = j.jobid
    ORDER BY start_time DESC
    LIMIT 1
) last_run ON TRUE
LEFT JOIN LATERAL (
    SELECT
        COUNT(*) AS total_runs_7d,
        SUM(CASE WHEN status = 'failed' THEN 1 ELSE 0 END) AS failed_runs_7d,
        ROUND(AVG(EXTRACT(EPOCH FROM (end_time - start_time))), 2) AS avg_duration_7d_sec
    FROM cron.job_run_details
    WHERE jobid = j.jobid
        AND start_time > NOW() - INTERVAL '7 days'
) stats ON TRUE
WHERE j.jobname LIKE 'ksk_%'
ORDER BY j.jobname;

COMMENT ON VIEW upoa_ksk_reports.ksk_cron_monitoring IS
'Mониторинг состояния pg_cron задач КСК';
```

**Использование view:**

```
-- Быстрая проверка статуса всех задач
SELECT jobname, last_run_time, health_status, last_message
FROM upoa_ksk_reports.ksk_cron_monitoring;

-- Задачи с проблемами
SELECT *
FROM upoa_ksk_reports.ksk_cron_monitoring
WHERE health_status IN ('ERROR', 'STALE', 'DISABLED');
```

## ✓ Преимущества этого подхода

1. Централизованное управление — всё в одном SQL файле
2. Встроенное логирование — ksk\_log\_operation() + cron.job\_run\_details
3. Легко мониторить — через SQL запросы и view
4. Работает в Kubernetes — не нужен доступ к cron на хосте
5. Идемпотентность — можно перезапускать скрипт настройки
6. Интеграция с Grafana — можно визуализировать данные из ksk\_cron\_monitoring

Что скажешь? Подходит такой вариант? 

\*\*

1. [https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/PostgreSQL\\_pg\\_cron.html](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/PostgreSQL_pg_cron.html)
2. <https://www.postgresql.org/docs/current/maintenance.html>