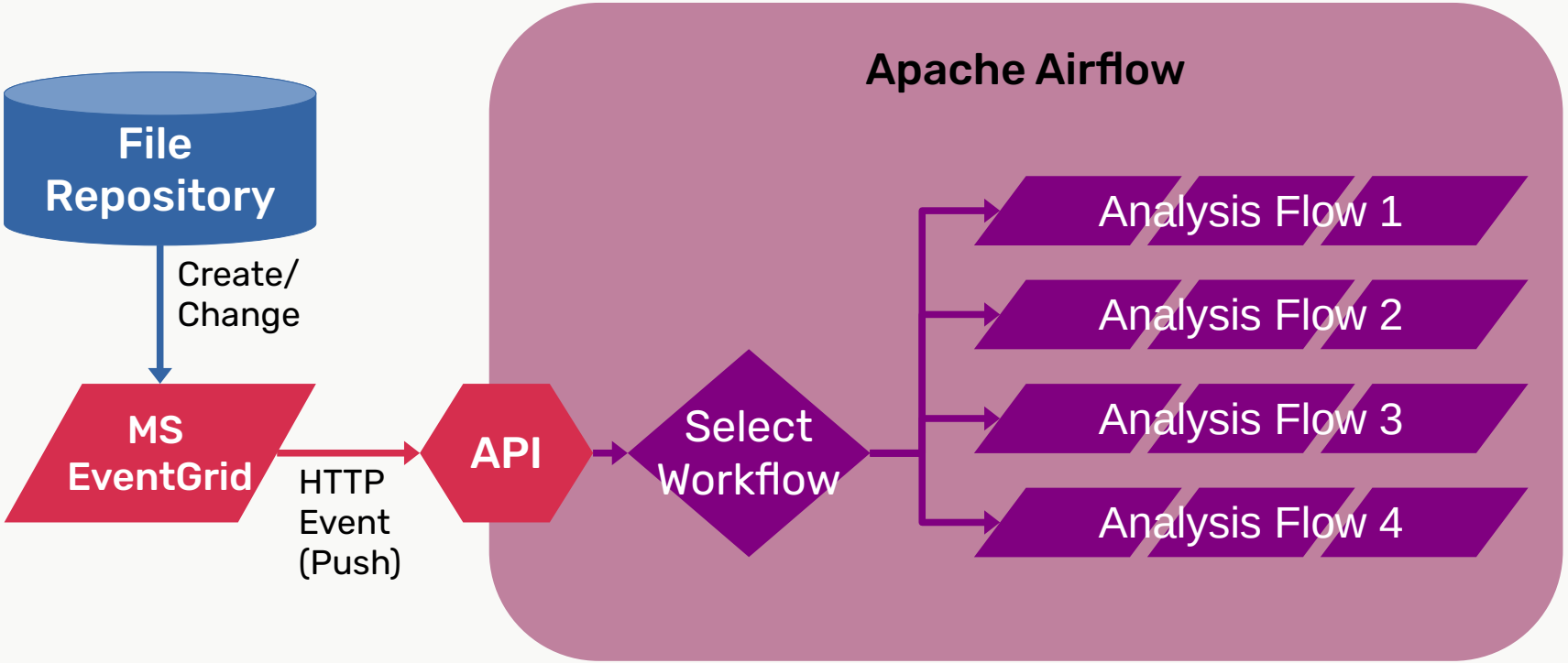
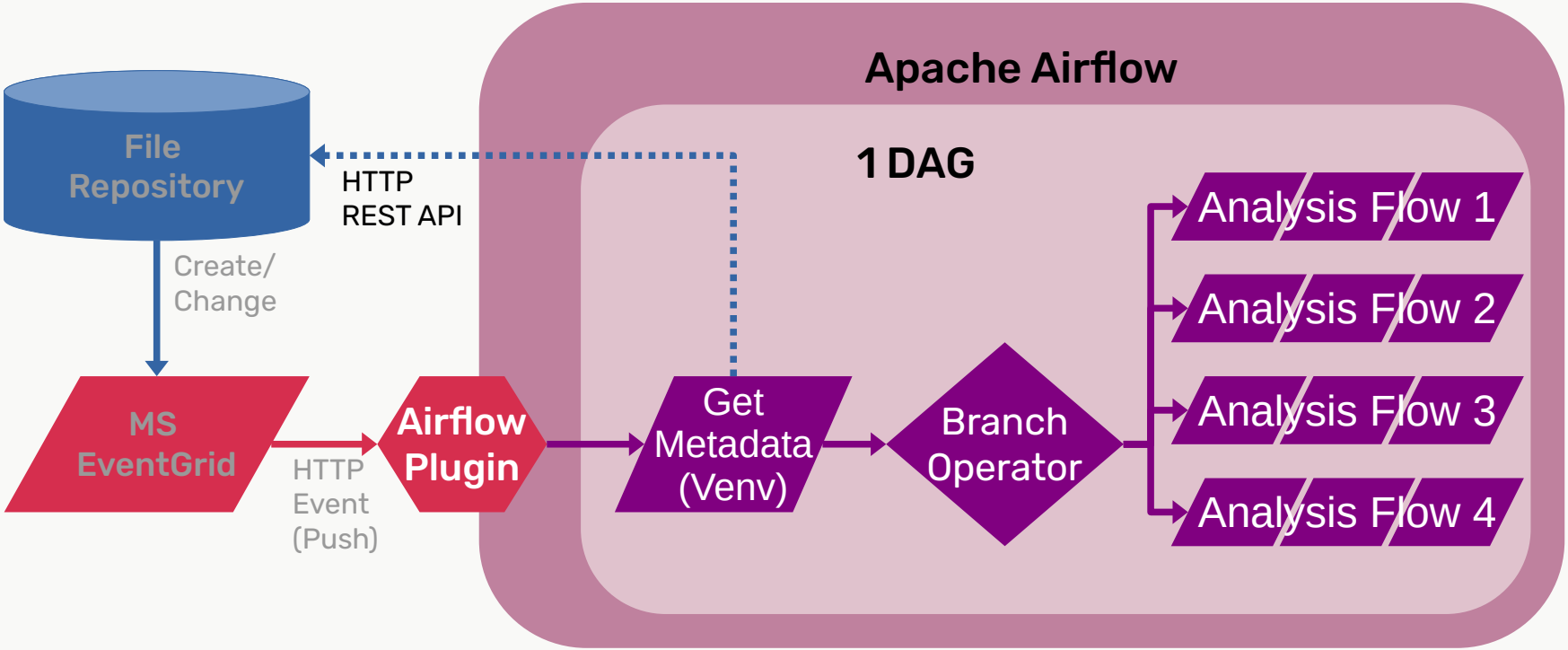
How we Tuned our Airflow to Make 1.2mio DAG runs - per Day!

Jens Scheffler, Robert Bosch GmbH

Some Context



The Implementation – 1 Event == 1 DAG Run



Initial Capacity: ~1 000 DAG runs/h

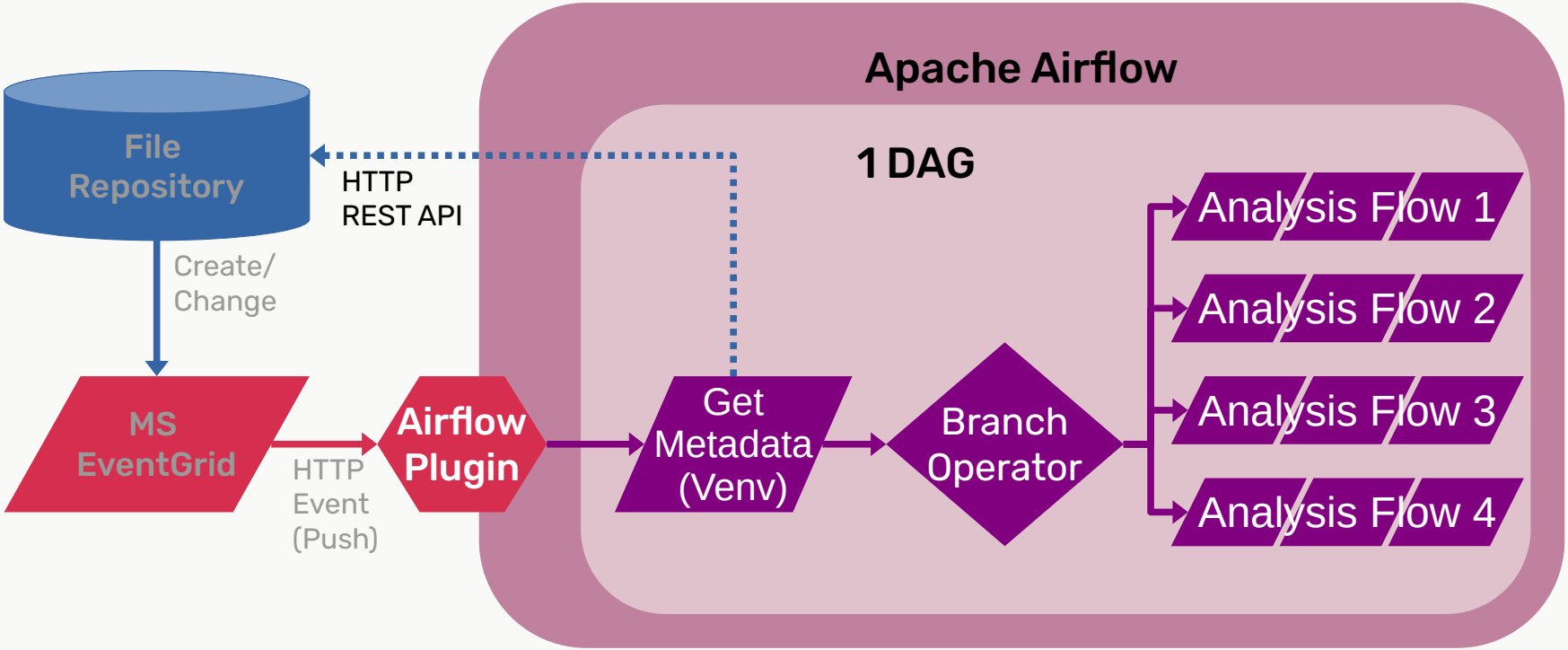
Deployment Tuning

- DAG
  - max_active_tasks 16 → 60 → 360
  - max_active_runs 16 → 60 → 360
  - Task:
    - priority_weight = -100
    - weight_rule = “absolute”
    - do_xcom_push = False
- Config Parameters
  - AIRFLOW__CORE__PARALLELISM = 500
  - AIRFLOW__SCHEDULER__MAX_DAGRUNS_PER_LOOP_TO_SCHEDULE = 200
- Scaling
  - 1 Scheduler → 3 Scheduler
  - 3 Celery Worker → 15 Celery Worker
  - Postgres 4 Cores → 16 Cores + IOPS++
  - PG Bouncer: max_client_conn = 1000
- Upgrade Postgres PaaS 12.x → Flexible Server 15.x → VACUUM FULL

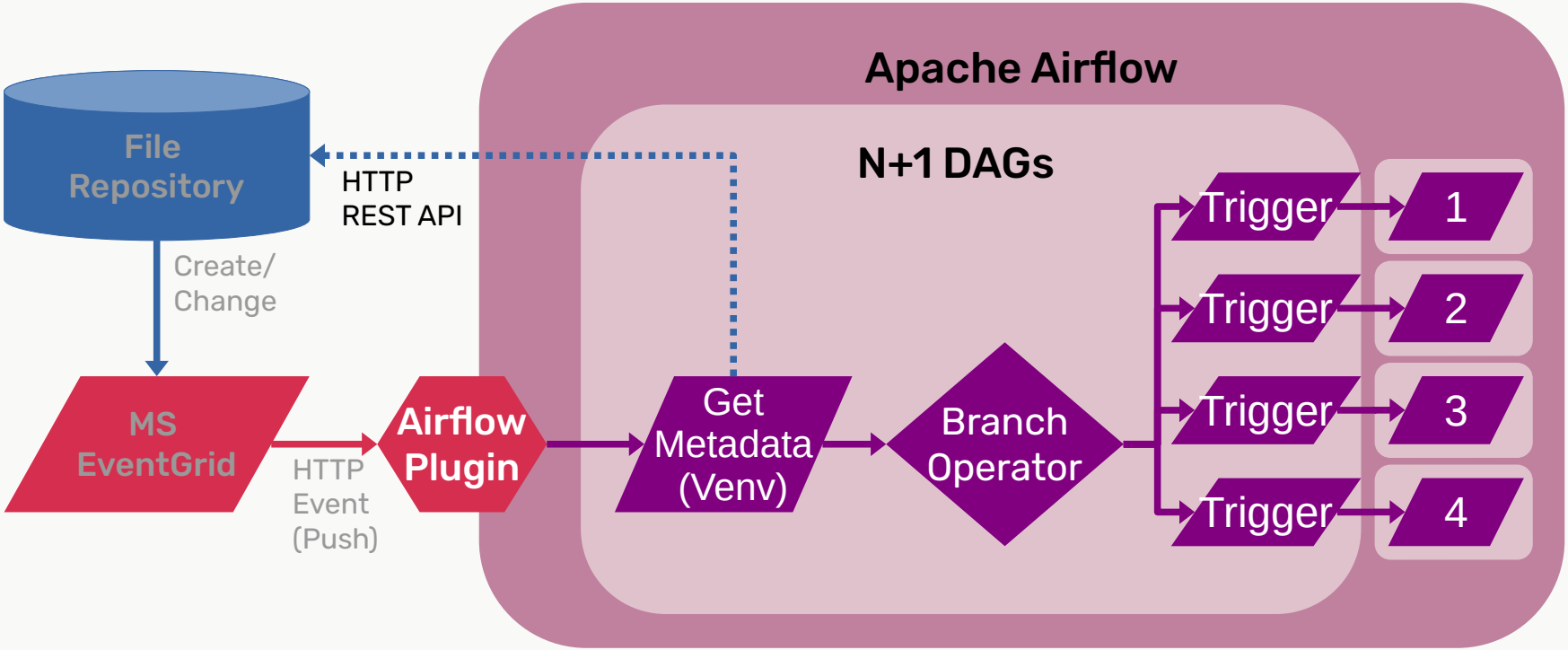
Airflow Optimizations

- Scheduler
  - Double query for DAGs to Schedule #30699 → Airflow 2.6.2
  - No next run scheduling if no interval #30706 → Airflow 2.6.2
  - Caching of DAG Bag #30704 → Airflow 2.7.0
  - Additional filtered DB index #30827 → Airflow 2.7.0
  - Optimize DAG parsing if no schedule #30911 → Airflow 2.7.0
- PythonVirtualenvOperator
  - Implement virtualenv caching #33355 → Airflow 2.8.0

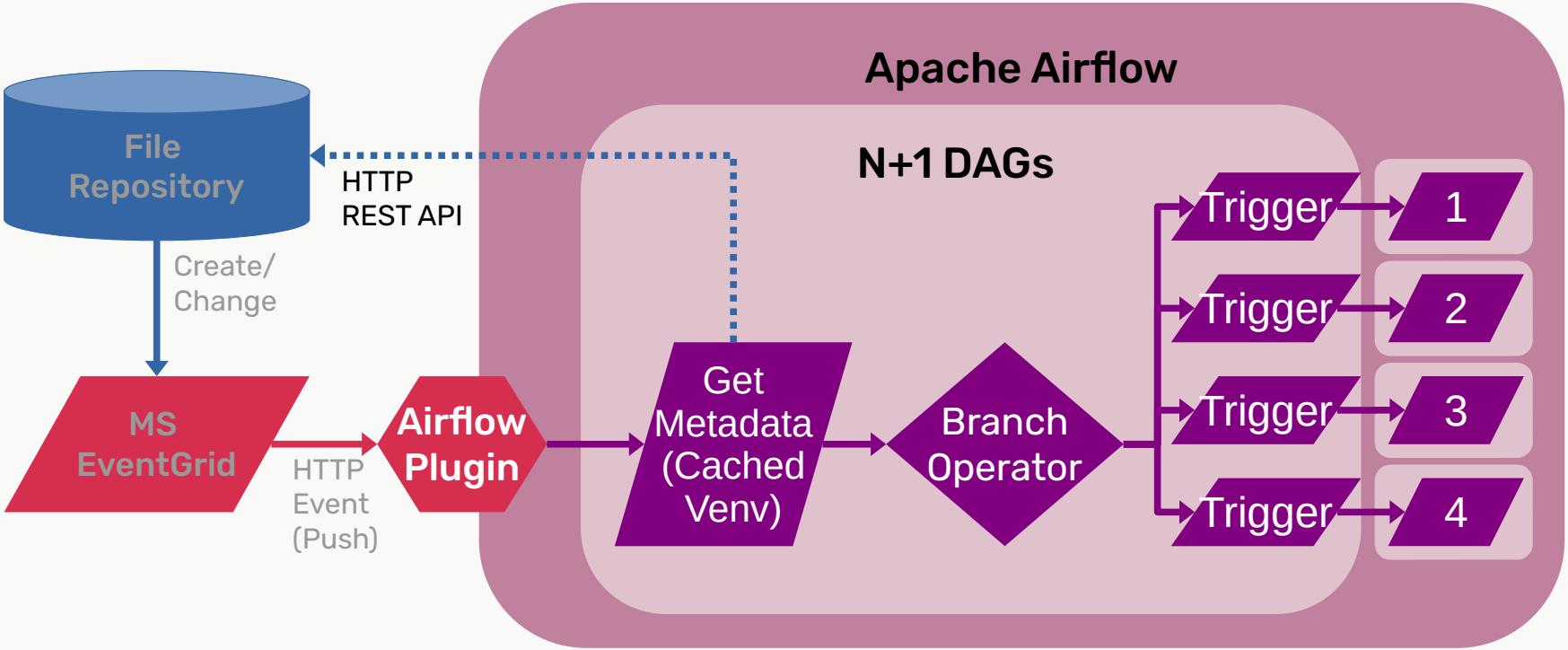
Incremental DAG optimizations (Start)



Incremental DAG optimizations (Trigger)

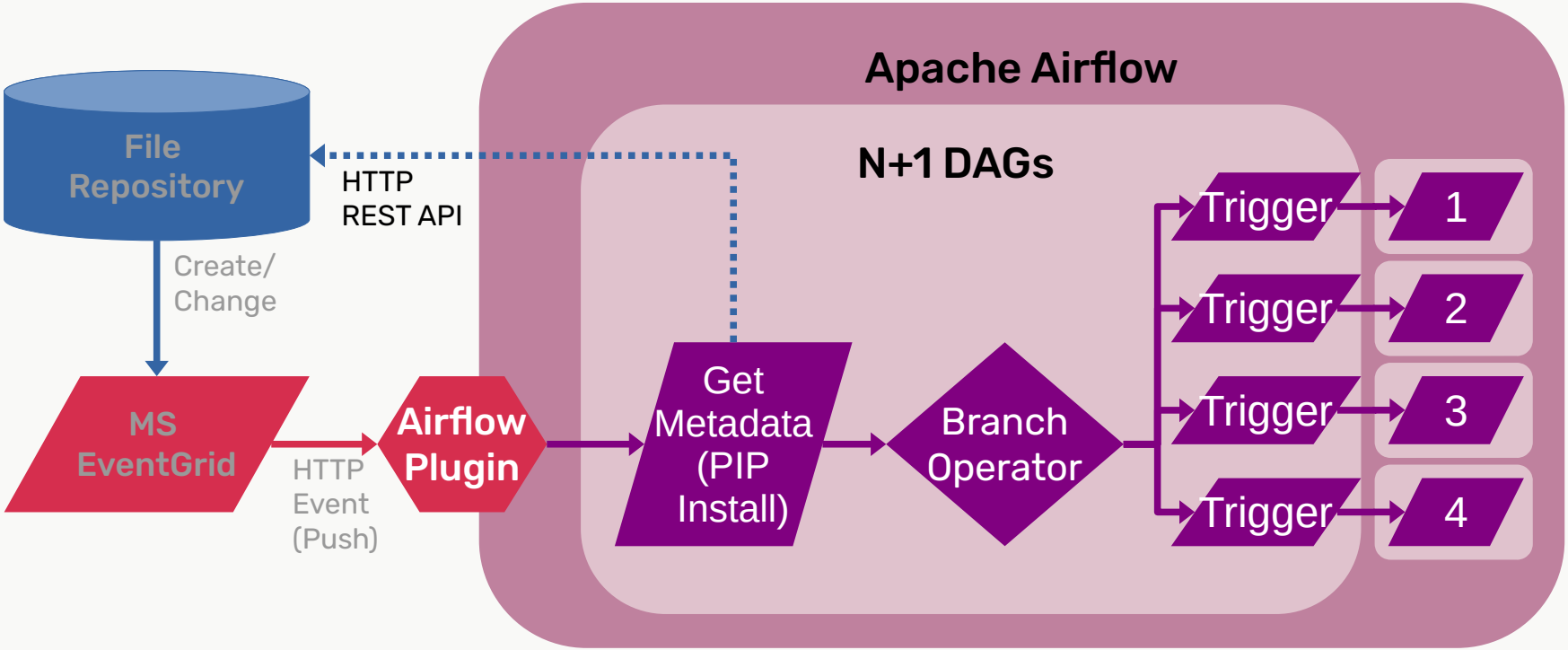


Incremental DAG optimizations (Venv Caching)



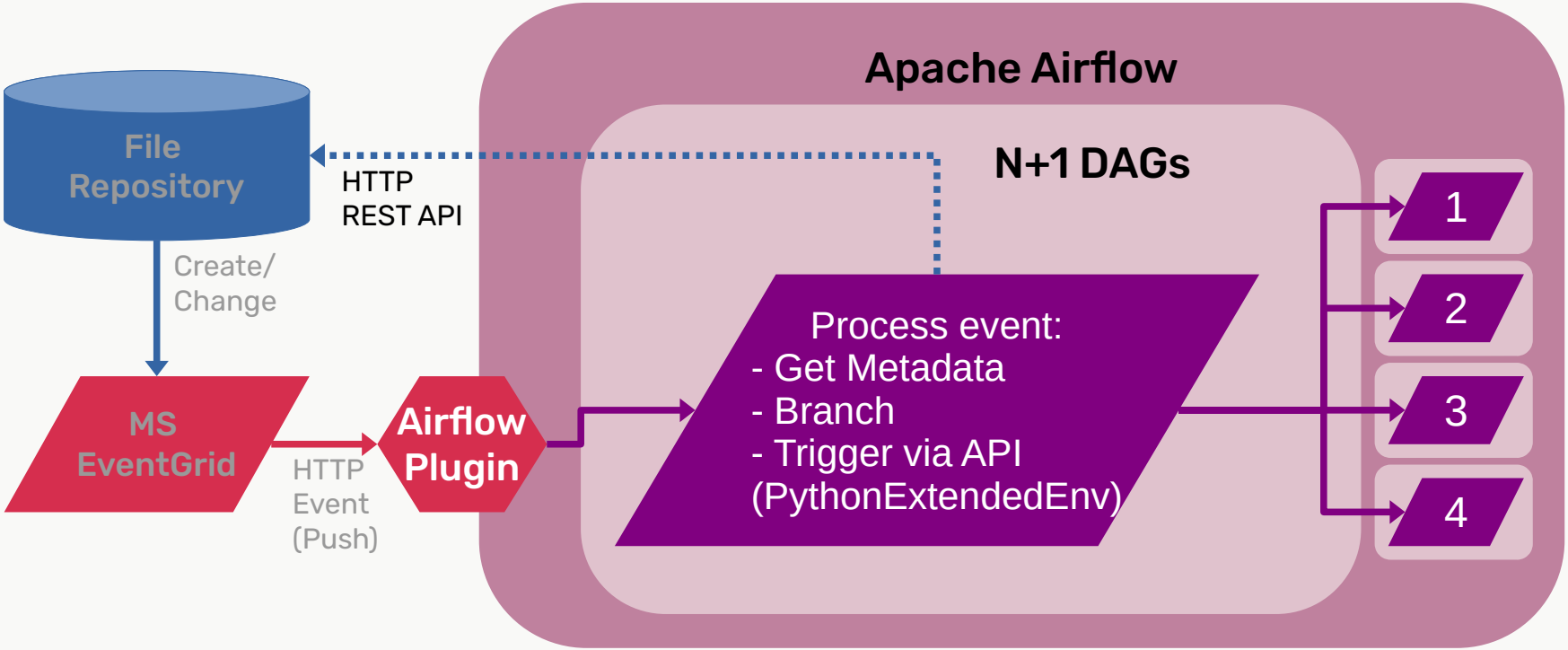
Virtualenv caching effect: ~40-50s → 3-15s

Incremental DAG optimizations (No Virtualenv)



No virtualenv: ~3-15s → 1-3s

Incremental DAG optimizations (1 Task)



Complete DAG execution: ~1s average

Environment Optimizations

- DAG Archiving
  - Problem
    - VACUUM FULL never completed
    - 40mio DAG runs in DB (+10 Task Instances each!)
    - “Other” DAGs needed to keep history (airflow db clean is global)
  - Solution: Custom archiving of events → 7 days retention, daily clean
- Early Event filtering
  - Increment 1: Negative Filtering
  - Increment 2: Positive Filtering
- Batched Event Support (up to 64 Events/call) → 1 DAG run == 64 Events
- Plugin HTTP Rate Limits (HTTP 429 Error)

Metrics and Outcome

Metrics today

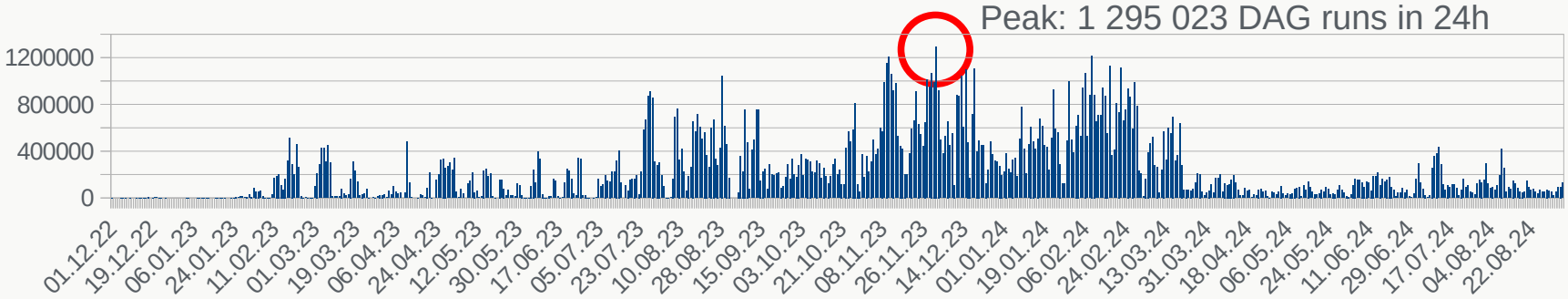
- Capacity: 50 000 runs/h, 70 000 peak
- Events: 1.7mio/day / 280k/h peak
- Utilization: 5 000 – 10 000 events/h
- 151 million events processed
- Average event latency: 1s

Outlook

- Move away from EventGrid → Kafka Pull
- File → Business Events with early filter

Deployment size today

- Postgres Flexible PaaS
  8 Cores, 32GB RAM, 512GB w/ 2300IOPS
- 3 Scheduler, 2 Cores, 2 GB RAM
  Separate DAG Processor
- 15 Celery Worker, Concurrency = 16
  4 Cores, 8 GB RAM
- 3 Webserver, 2 Cores, 2 GB RAM



Questions?

Kudos to all team mates making this possible

Jens Scheffler → @jscheffl

Marco → @AutomationDev85

Daniel Wolf → @wolfdn

BOSCH