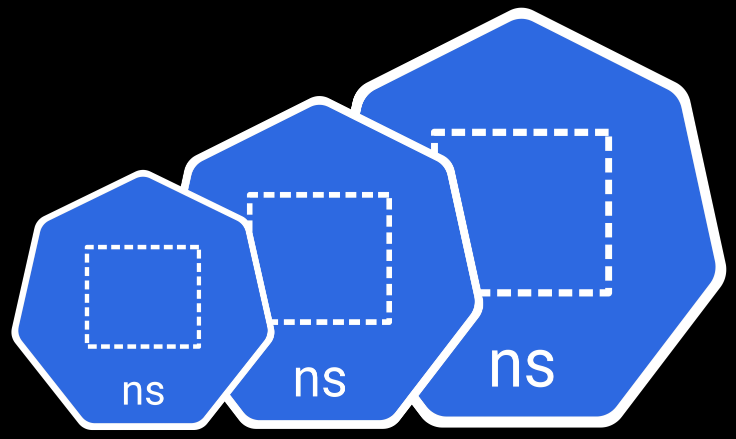
Building in Resource Awareness and Event Dependency into Airflow

Roberto Santamaria, Apple
Anandhi Murali, Apple
Xiaodong Deng, Apple

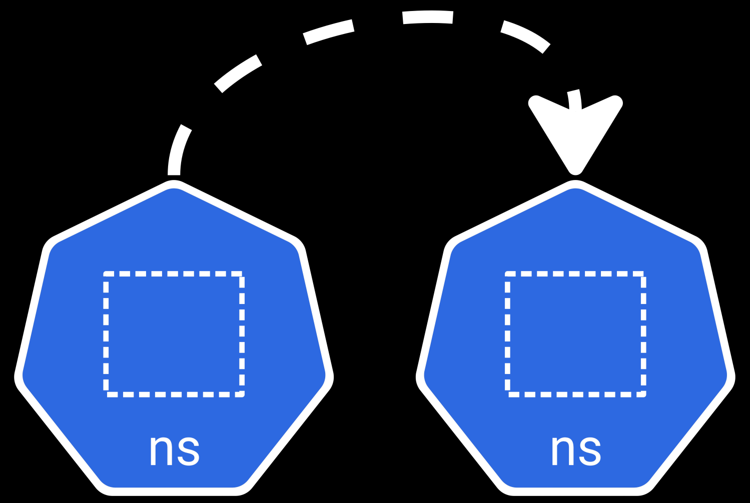
Event-driven, Resource Awareness and SLO Orchestration

Problems



Required awareness of compute resource constraints

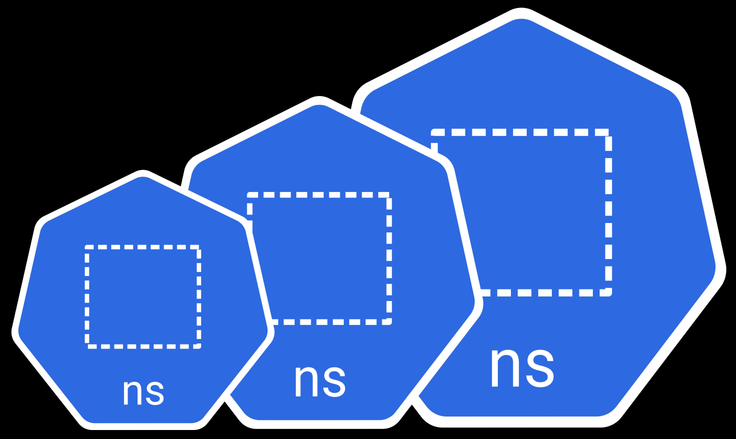
Coordinating workloads takes time and is human error prone.

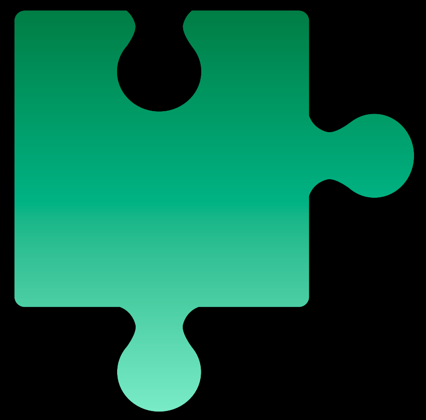


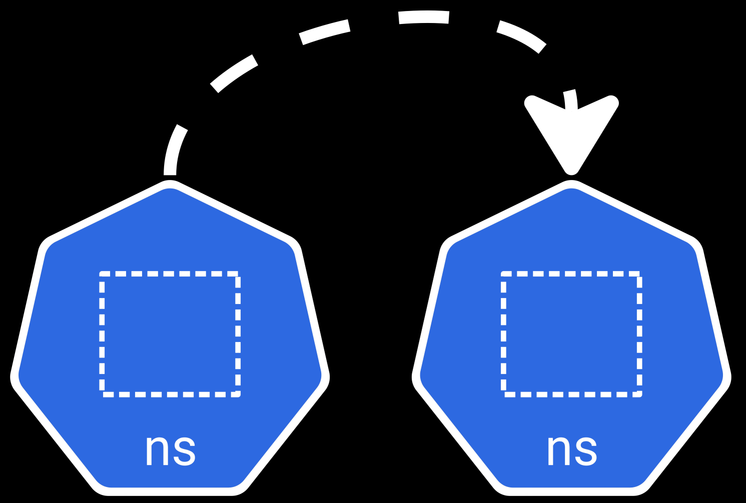
Unable to take advantage of multiple compute options and flexibility

Forced to describe DAGs in terms of start time and cadence, but sometimes need in terms of deadline.

Solution: Event-Driven, SLO-based orchestration







Users with multiple compute options can now schedule across them seamlessly

Offloaded the decision making regarding compute resources and scheduling to the orchestration system

Users provide scheduling windows and deadlines

Event Service

Why

🤯 Multiple hops between various services

- 10s of services: Spark, Flink, Trino, Airflow etc.
- Services * Jobs * Runs * States

😇 Centralized hub for system events

- Collects, stores and distributes state to interested parties
- Decouples systemic dependencies with push mechanism
- Realtime notifications and dashboards

Why

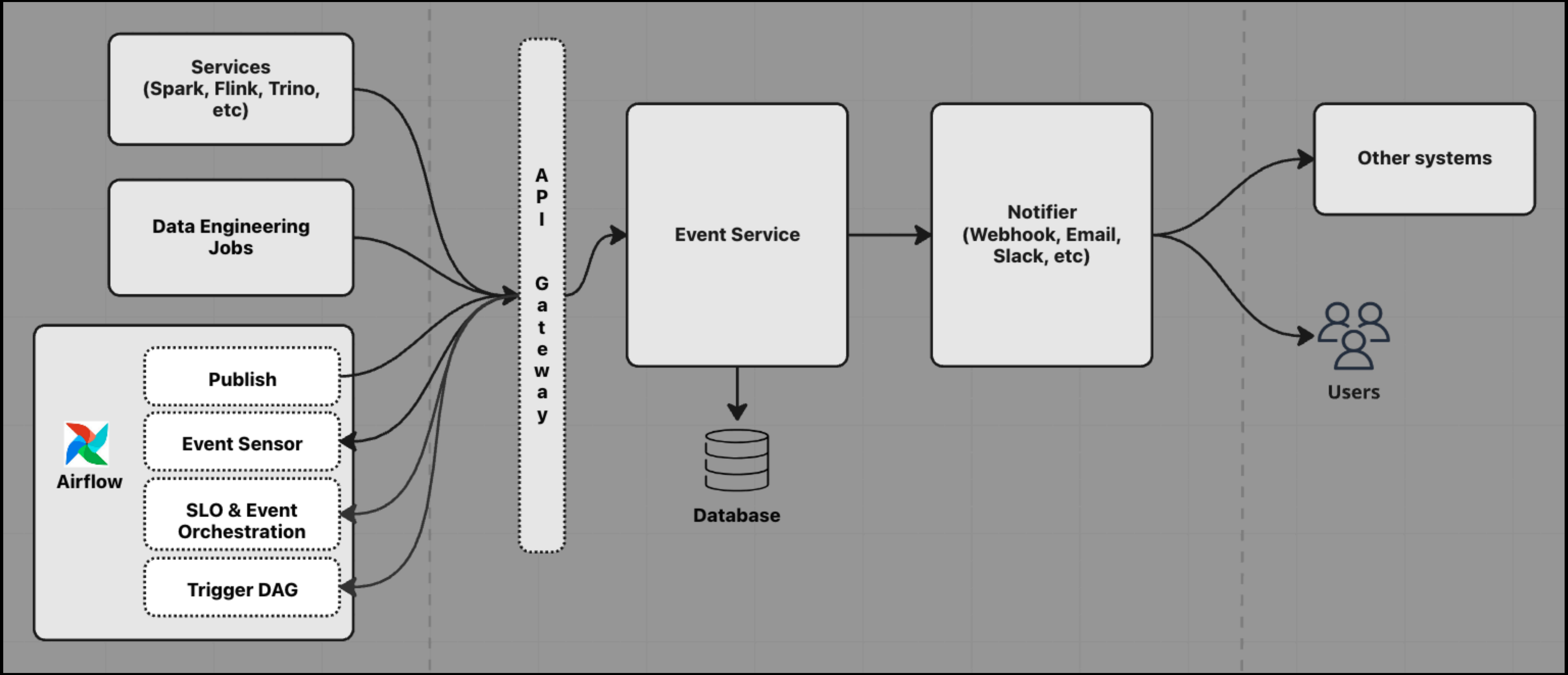
🤯 Dependencies

- ETLs depend on upstream data availability
- Several data generation jobs, several data sources / tables

😇 Event Based Workflow Orchestration

- Efficient state based triggers
- Lower latency / just in time scheduling
- Avoid resource wastage

Architecture



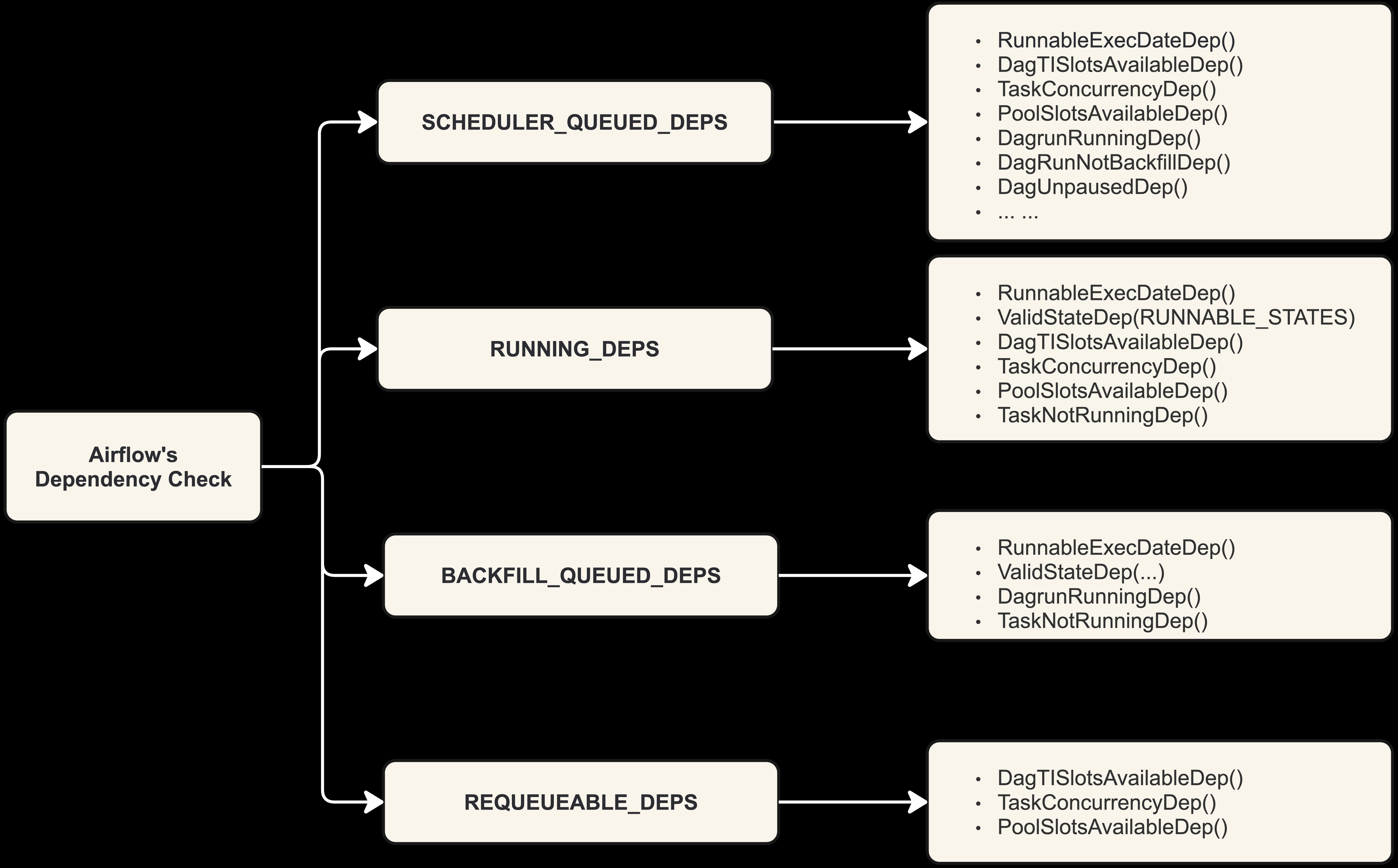
vs Data Aware Scheduling

	Data Aware Scheduling	Event Service
Handle dependency	✅ (only datasets)	✅ (multi purpose)
Standalone	❌ (tightly coupled to Airflow)	✅ (supports external events)
Isolation	❌ (reside on same Airflow instance)	✅ (centralized)
Scalability	❓ (limited by Airflow cluster's capacity)	✅ (designed for scale)
Traceability	✅	… work in progress

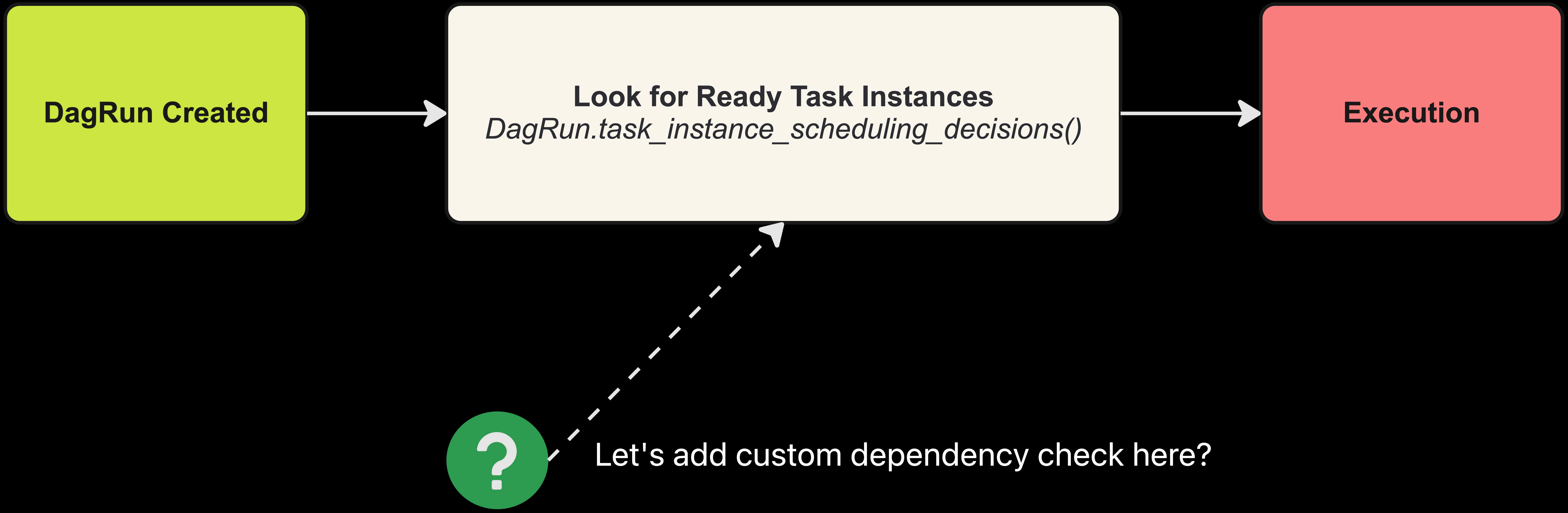
Resource Awareness and SLO Orchestration

“Stop and check”

Airflow’s Built-in Dep Checks



Add Custom Dep Check?


DagRun Created
Look for Ready Task Instances
DagRun.task_instance_scheduling_decisions()
Execution
?
Let's add custom dependency check here?

Add Custom Dep Check?

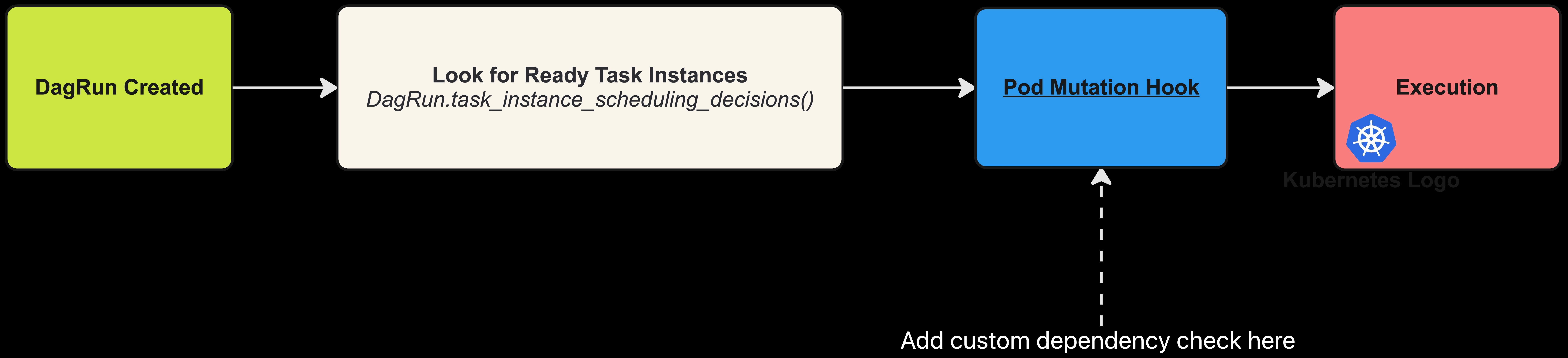
Pros ✓

- More flexible and customizable scheduling

Cons ✗

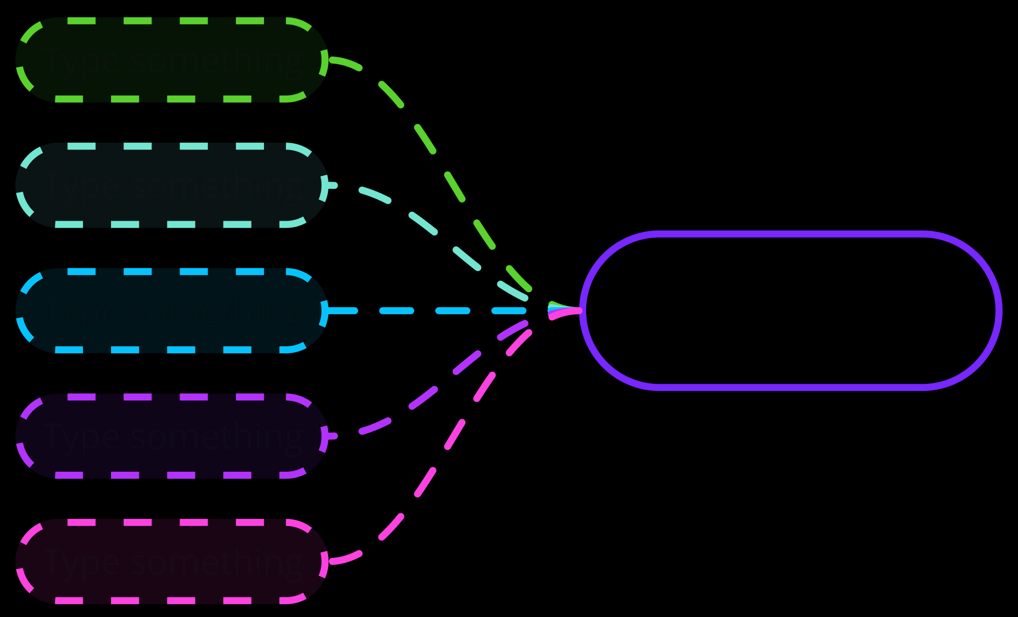
- High risk: allowing adding user code in the very centre of the hot path for the scheduler

The Solution We Adopted Eventually


DagRun Created
Look for Ready Task Instances
DagRun.task_instance_scheduling_decisions()
Pod Mutation Hook
Execution
Kubernetes Logo
Add custom dependency check here

The Solution We Adopted Eventually

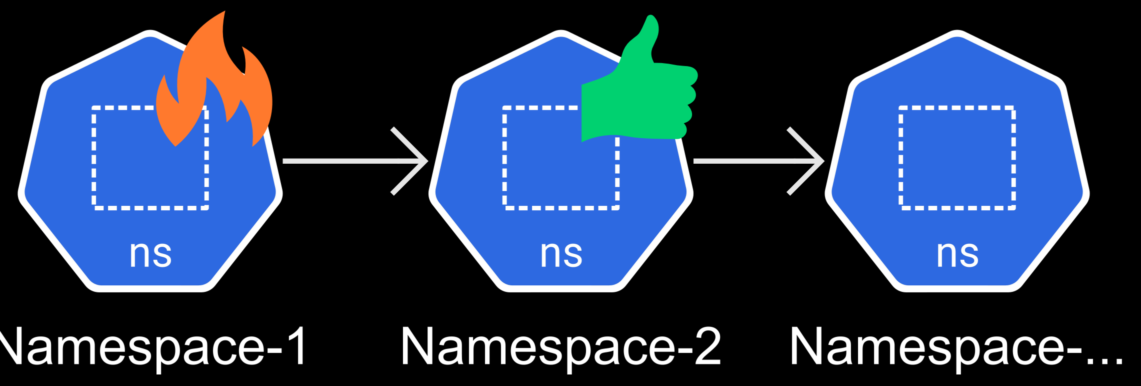
Features we deliver with this solution



Integrate with the Event Service

So a certain TI will only be executed

when the event dependencies are met

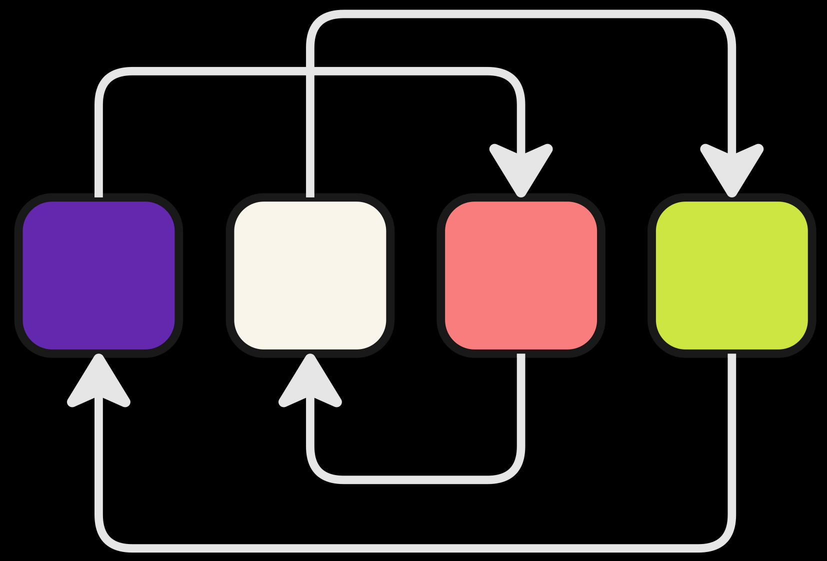


Resource Availability Check

If the namespace lacks enough resource,

automatically switch to another namespace

to execute the job



Smarter Scheduling

e.g. shuffle the execution order of TIs

for better global scheduling performance

Thanks!