

Airflow Beyond Python: New **Languages** for Tasks and Workflows

Jason (Zhe-You) Liu

Apache Airflow Committer

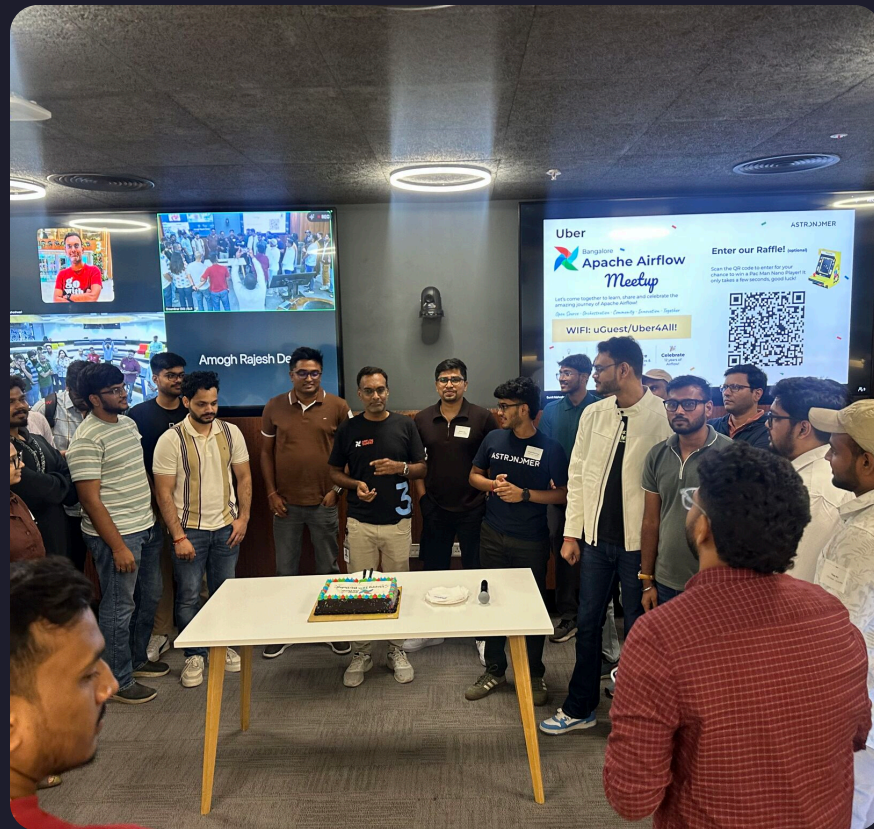
Wei Lee

Apache Airflow PMC Member

Community Over Code Glasgow 2026 · Airflow Contributor Days

Airflow *is* a language.

For 12 years, Airflow spoke only **Python**.



Airflow Meetup, India
(photo from Amogh's post)



Airflow Meetup, Korea
(photo from Yeonguk's post)



Airflow just turned 12.



Something big is coming in **Airflow 3.4.0.**

Expected around the end of October.

Author your entire Dags in **Go**, **TypeScript** or **Java**!

Without Python.



Go



TypeScript

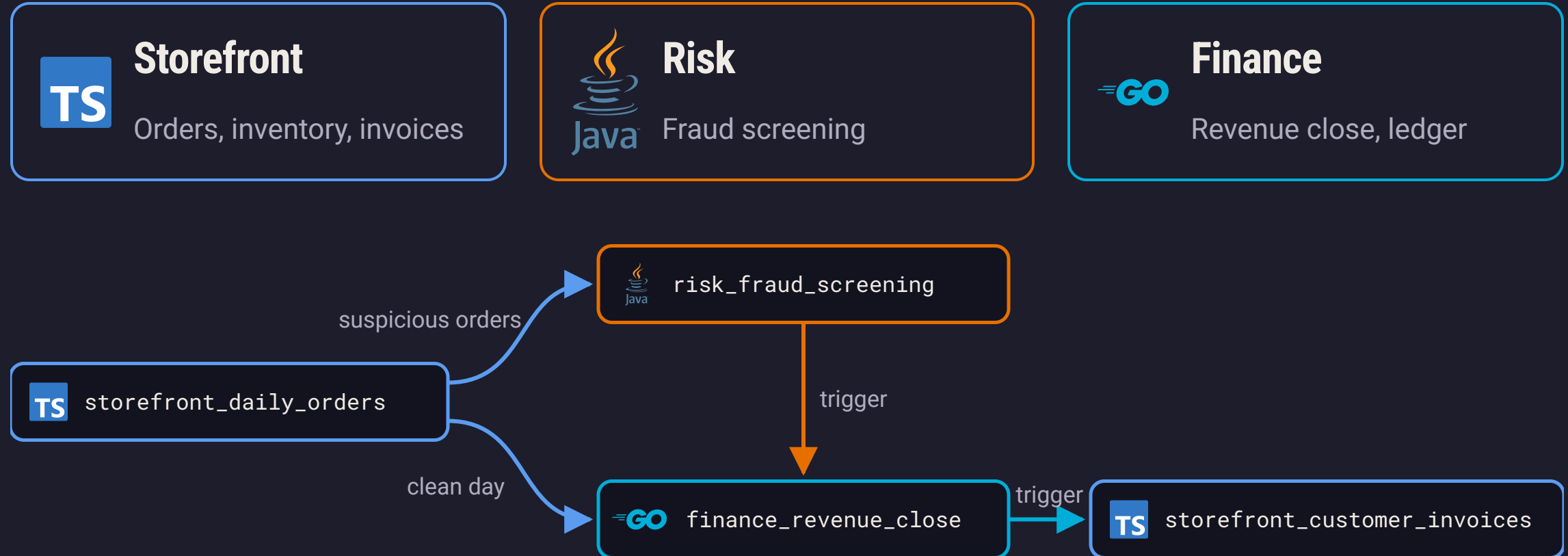


Java

What primitives do we support?

Primitive			
All Dag and task level arguments	✓	✓	✓
Wiring: <code>before</code> , <code>after</code>	✓	✓	✓
TaskFlow	✓	✓	✓
TaskGroup	✓	✓	✓
If / else	✓	✓	✓
Switch / case	✓	✓	✓
TriggerDagRun	✓	✓	✓
Dynamic Dag generation	✓	✓	✓
Variable, Connection, XCom, Task State Store	✓	✓	✓

Demo: CoC EU Retail



Each team ships its own Dag bundle in its own language.
Hand-offs only through `TriggerDagRun` and a data contract on a shared lake.

Live Demo

CoC EU Retail: four Dags, three languages, one Airflow.

Dag-level queue

 Go

```
dag := airflow.Dag(  
    "finance_revenue_close",  
    airflow.DagSpec{  
        Queue: "golang",  
        Tags:  []string{"finance"},  
    },  
)
```

 TypeScript

```
const dag = new Dag(  
    "storefront_daily_orders",  
    {  
        schedule: "@daily",  
        queue: "typescript",  
    },  
);
```

 Java

```
@Builder.Dag(  
    id = "risk_fraud_screening")  
public class FraudScreeningDag {  
  
    @Builder.Task(  
        id = "load_batch",  
        queue = "java")  
    public String loadBatch(...) {
```

No Dag-level queue yet: set it on each task.

A TaskFlow task with inputs

 Go

```
posted := dag.Task(  
    postJournalEntries,  
    airflow.TaskSpec{  
        TaskID: "post_journal_entries",  
    },  
    airflow.Inputs(sales, settled),  
)  
  
func postJournalEntries(  
    actx airflow.Context,  
    sales closing.SalesLoad,  
    settled closing.SettlementLoad,  
) (closing.Posted, error)
```

 TypeScript

```
const validated = dag.task(  
    "validate_orders",  
    validateOrders,  
)(({  
    orders: ordersExport,  
    refunds: refundsExport,  
}));  
  
async function validateOrders(  
    feeds: {  
        orders: ExportSummary;  
        refunds: ExportSummary;  
    },  
): Promise<SalesSummary>
```

 Java

```
@Builder.Task(  
    id = "apply_decisions",  
    queue = QUEUE)  
public String applyDecisions(  
    String scoresPath) { ... }  
  
// in the @Builder.Deps wiring  
var scores = scoreOrders(...);  
var decisions =  
    applyDecisions(scores);
```

If / else



```
dag.If(  
    ledgerBalanced,  
    airflow.TaskSpec{  
        TaskID: "ledger_balanced",  
    },  
    airflow.Inputs(posted),  
).  
    Then(published).  
    Else(incident)  
  
func ledgerBalanced(  
    actx airflow.Context,  
    posted closing.Posted,  
) (bool, error)
```



```
const gate = dag.if(  
    hasSuspiciousOrders,  
    { summary: validated },  
    { taskId: "has_suspicious_orders" }  
);  
gate  
    .then(requestRiskReview)  
    .else(closeRevenue);  
  
async function hasSuspiciousOrders({  
    summary,  
}): {  
    summary: SalesSummary;  
}): Promise<boolean>
```



```
@Builder.If(  
    id = "chargeback_exposure_high",  
    queue = QUEUE)  
public boolean  
    chargebackExposureHigh(  
        String decisionsPath,  
        Client client) {  
    ...  
    return exposure > threshold;  
}  
  
// in the @Builder.Deps wiring  
chargebackExposureHigh(decisions)  
    .Then(alerted)  
    .Else(tolerated);
```

Switch / case



```
dag.Switch(  
    closes.pick,  
    airflow.TaskSpec{  
        TaskID: "close_type",  
    },  
    airflow.Inputs(published),  
).  
    Case(closes.daily).  
    Case(closes.week).  
    Case(closes.month)  
  
func (c *closeTasks) pick(  
    actx airflow.Context,  
    published TrialBalance,  
) (*airflow.TaskRef, error) {  
    ...  
    return c.month, nil  
}
```



```
const strategy = dag.switch(  
    restockStrategy,  
    { inventory: inventoryExport },  
    { taskId: "restock_strategy" },  
);  
strategy  
    .case(cases.restock_standard)  
    .case(cases.restock_expedite)  
    .case(cases.pause_campaigns);  
  
async function restockStrategy({  
    inventory,  
}): {  
    inventory: ExportSummary;  
}): Promise<TaskRef> {  
    ...  
    return cases[choice];  
}
```



```
@Builder.Switch(  
    id = "route_by_worst_band",  
    queue = QUEUE)  
public Class<? extends Task>  
    routeByWorstBand(String path) {  
    ...  
    return FraudScreeningDagBuilder  
        .BlockAndRefund.class;  
}  
  
// in the @Builder.Deps wiring  
routeByWorstBand(decisions)  
    .Case(approved)  
    .Case(queued)  
    .Case(refunded);
```

Register the bundle



```
func main() {  
    b := airflow.Bundle()  
    b.Register(newRevenueCloseDag())  
  
    if err := b.Serve(); err != nil {  
        log.Fatal(err)  
    }  
}
```



```
import { Bundle } from  
    "apache-airflow-ts-sdk";  
  
const bundle = new Bundle(  
    dailyOrders,  
    customerInvoices,  
);  
await bundle.serve();
```



```
public class RiskBundleBuilder {  
    public static Bundle build() {  
        return new Bundle().register(  
            FraudScreeningDag.class);  
    }  
  
    public static void main(  
        String[] args) {  
        Server.create(args)  
            .serve(build());  
    }  
}
```

Pack the bundle



```
go tool airflow-go-pack \  
  --goos linux --goarch amd64 \  
  --output bundles/finance/finance \  
  .
```

Output: `finance`, no extension



```
pnpm exec airflow-ts-pack \  
  src/main.ts \  
  --outdir bundles/storefront
```

Output: `bundle.min.mjs`



```
plugins {  
    id("org.apache.airflow.sdk")  
}  
airflowBundle {  
    mainClass =  
        "com.acme.risk.RiskBundleBuilder"  
}
```

```
./gradlew bundle
```

Output: `build/bundle/*.jar`

Deployment config

[dag_processor]

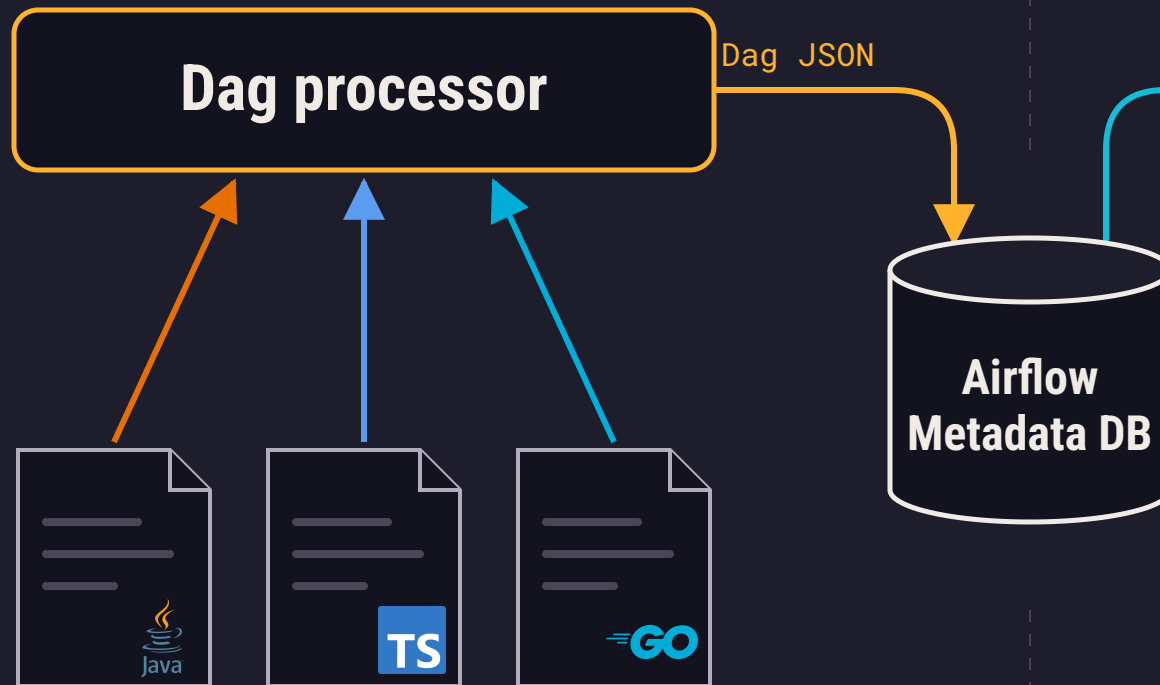
```
dag_bundle_config_list = [  
    {"name": "finance", "classpath": "airflow.dag_processing.bundles.local.LocalDagBundle",  
     "kwargs": {"path": "/files/bundles/finance"}},  
    {"name": "storefront", "classpath": "airflow.dag_processing.bundles.local.LocalDagBundle",  
     "kwargs": {"path": "/files/bundles/storefront"}},  
    {"name": "risk", "classpath": "airflow.dag_processing.bundles.local.LocalDagBundle",  
     "kwargs": {"path": "/files/bundles/risk"}}]
```

[sdk]

```
coordinators = {  
    "go": {"classpath": "airflow.sdk.coordinators.executable.ExecutableCoordinator"},  
    "ts": {"classpath": "airflow.sdk.coordinators.node.NodeCoordinator",  
          "kwargs": {"node_executable": "node"}},  
    "jdk": {"classpath": "airflow.sdk.coordinators.java.JavaCoordinator",  
            "kwargs": {"java_executable": "java"}}}  
queue_to_coordinator = {"ao1and": "ao". "tvdescript": "ts". "iava": "idk"}
```

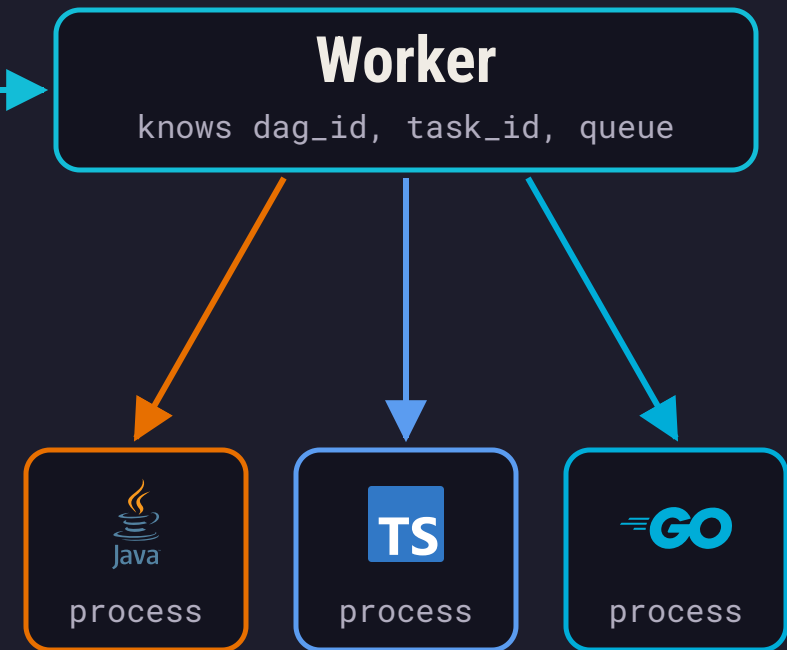
How does the SDK runtime work?

Dag processing time



Answer a compatible serialized Dag JSON

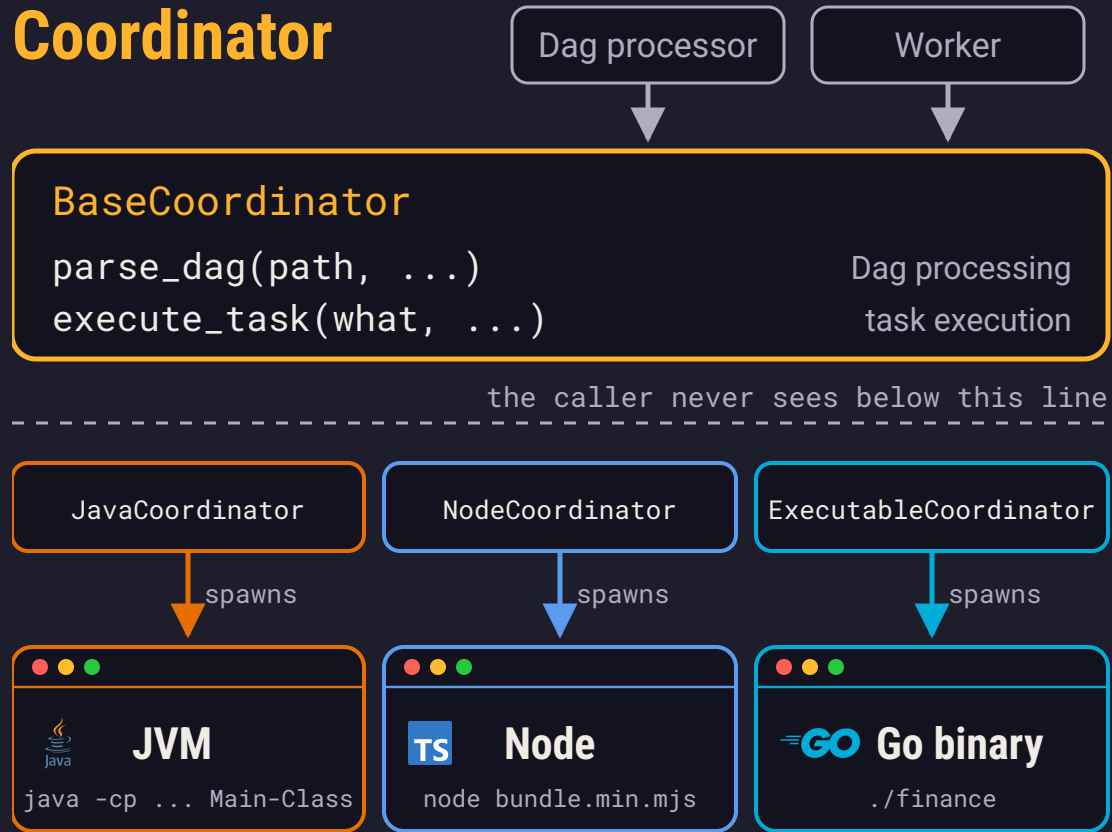
Task execution time



How do we spawn your task subprocess?

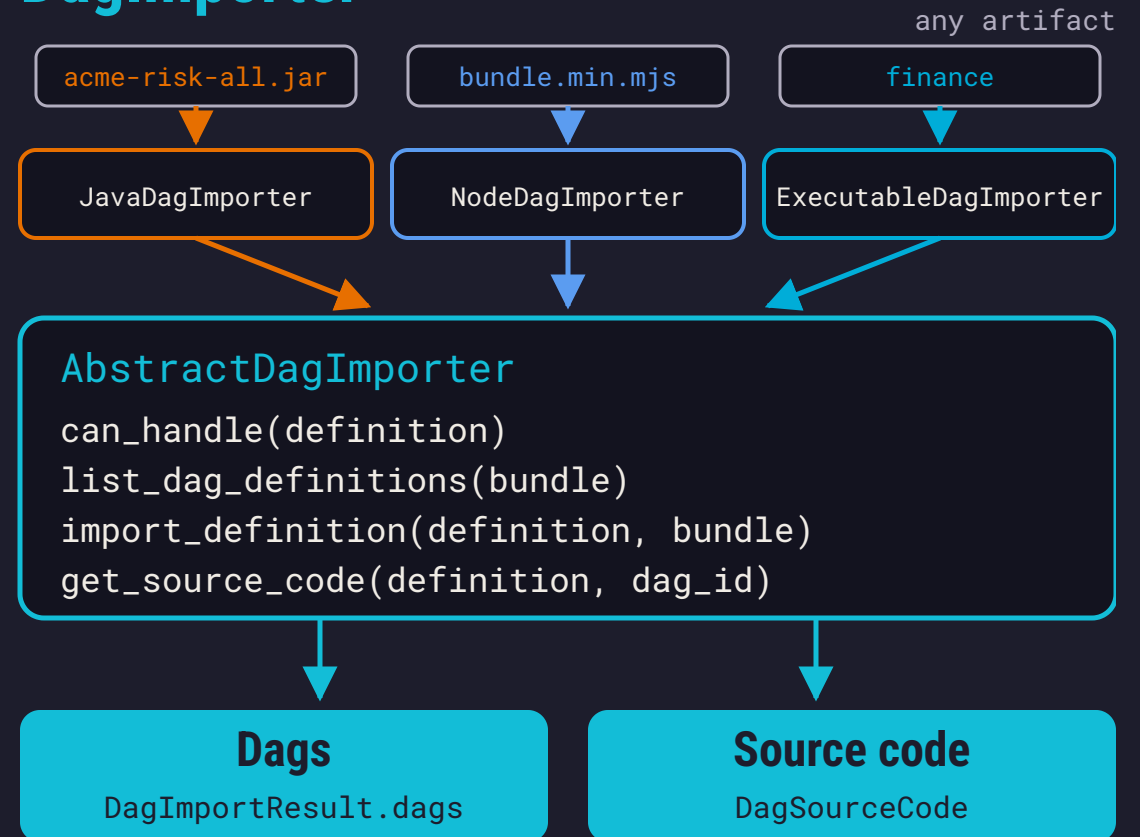
Coordinator and DagImporter

Coordinator



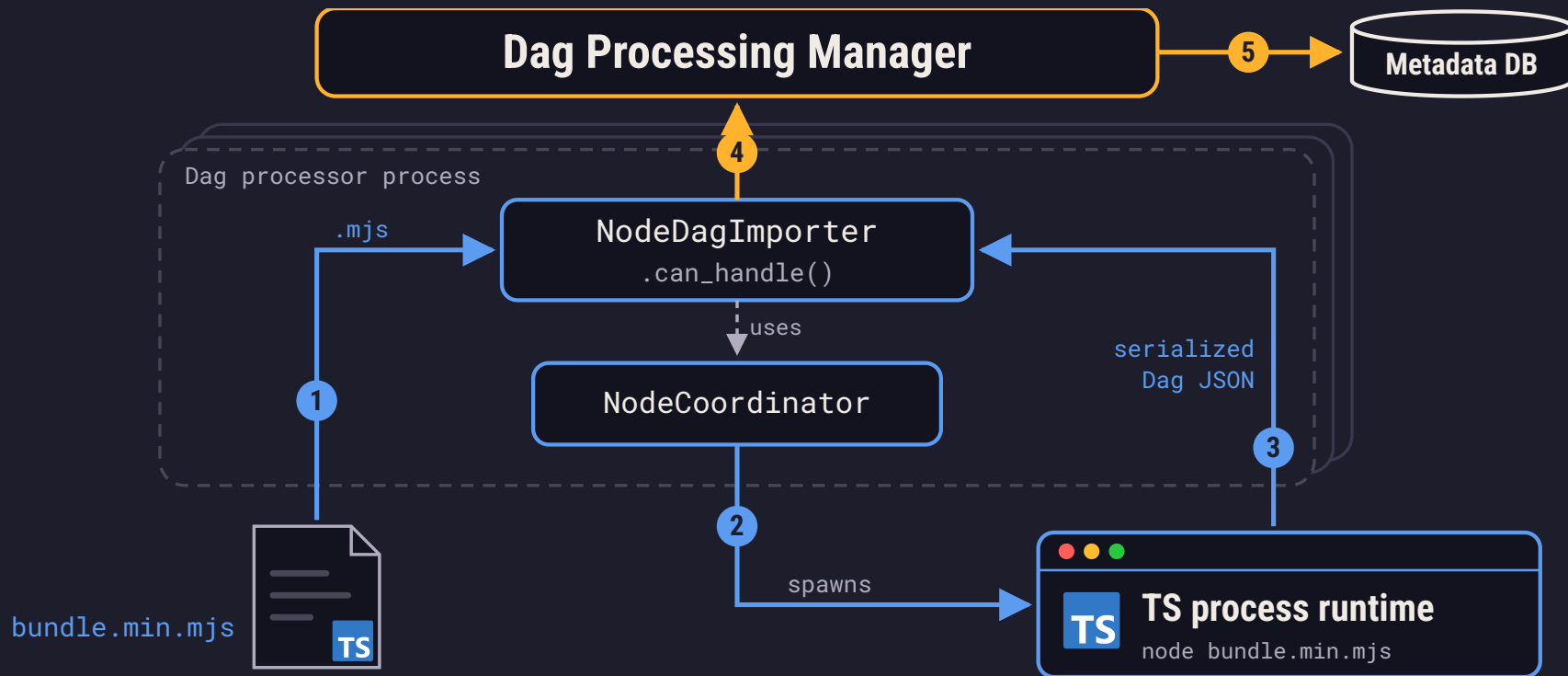
Each one spawns its process runtime.
Callers only use `parse_dag` and `execute_task`.

DagImporter



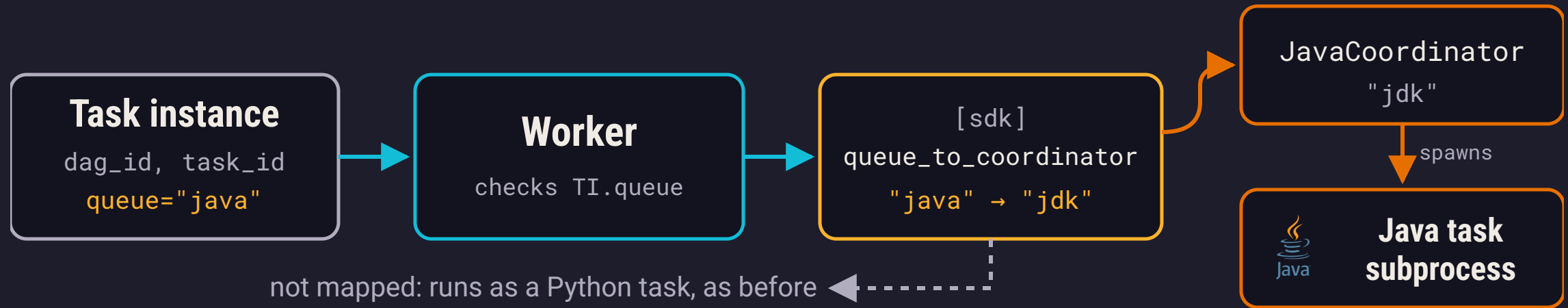
Whatever the artifact, you get Dags and their source.

Dag processing



- 1 `NodeDagImporter` claims the bundle by extension or `can_handle()`.
- 2 `NodeCoordinator` spawns the TS process runtime.
- 3 The runtime returns the serialized Dag JSON to the DagImporter.
- 4 The Dag processor process hands it to the manager.
- 5 The manager persists it to the Metadata DB.

Task execution



[sdk]

```
queue_to_coordinator = {"java": "jdk", "typescript": "ts", "golang": "go"}
```

One Airflow, many **dialects**.

The live demo: three teams, three languages, one Airflow.

Not to mention, once these come to every language:

Human-in-the-loop

Common AI features

Providers, e.g. operators in
TypeScript

Asset-native, event-driven

Every dialect talks to the others
in the **Airflow language**.

Jason (Zhe-You) Liu

Apache Airflow Committer

github.com/jason810496

linkedin.com/in/zhe-you-liu

Any feedback on the Airflow language SDKs?

Let's discuss after the talk or in `#sig-lang-sdks`.

So what's the
next language?

NEW

Introducing

`apache-airflow-sdk-english`

Write your Dags in plain English.

Dags, in English

```
from airflow.sdk.english import *
```

```
Every weekday at 9am,
```

```
    load yesterday's orders from Postgres.
```

```
    If anything looks off,
```

```
        page #data-oncall.
```

Retries, in English

Dear Airflow,
`if` it fails, `try` again,
unless the password `is` wrong. Then just tell me.
Thanks.

Why stop at English?

Taiwanese Mandarin

```
from airflow.sdk.taiwanese_mandarin import *
```

每天早上九點，
把昨天的訂單整理好，
寄給業務。

That's not how
the **SDK** works!

Airflow already speaks **both**.

Programming languages

**Python · Go ·
TypeScript · Java**

Natural language

**Prompts, agents, human
reviews**

The Dag stays code.

Natural language goes to the models, agents and reviewers inside it.

AI providers

`common.ai`

`anthropic`

`openai`

Plus AI operators in `google`, `microsoft.azure`, `snowflake`.

Growing fast — **much more capable**

AI providers

`common.ai`

`anthropic`

`openai`

Plus AI operators in `google`, `microsoft.azure`, `snowflake`.

Growing fast — **much more capable**

Making Airflow more **AI-native**

AI providers

`common.ai`

`anthropic`

`openai`

Plus AI operators in `google`, `microsoft.azure`, `snowflake`.

Growing fast — **much more capable**

Making Airflow more **AI-native**

Talk to Airflow in **natural language**

Start on 2.11.

Unlock more as you upgrade.

Airflow	2.11+	Operators · batch · failover · cost cap · toolsets
Airflow	3.1+	Human approval on answers
Airflow	3.3+	Retry policies · tool approval · durable replay

An LLM call is just a task.

Dag

Task · every attempt

prompt

@task.llm

branch

@task.llm_branch

tools

toolsets

human

approval

retries

attempts

@task.llm

```
class Entities(BaseModel):
    names: list[str]
    locations: list[str]

@task.llm(
    llm_conn_id="pydanticai_default",
    system_prompt="Extract named entities.",
    output_type=Entities,
)
def extract(text: str):
    return f"Extract entities from: {text}"
```

The answer lands in XCom

2 XCom Entries

 Add Filter

 Preset Filters

Key ↕	timestamp ↕	Value
decision	2026-10-06 17:50:52	model: gpt-oss:20b action: null +7 more
return_value	2026-10-06 17:50:52	example_llm.Entities@version=1(names=['Alice', 'Bob'], locations=['Paris', 'London']) 

@task.llm_batch

```
@task.llm_batch(  
    llm_conn_id="pydanticai_default",  
    model_id="openai:gpt-5",  
    result_path="s3://bucket/prefix/{{ run_id }}",  
    deferrable=True,  
)  
def build_prompts(rows: list[str]) -> list[str]:  
    return [f"Summarize: {row}" for row in rows]
```

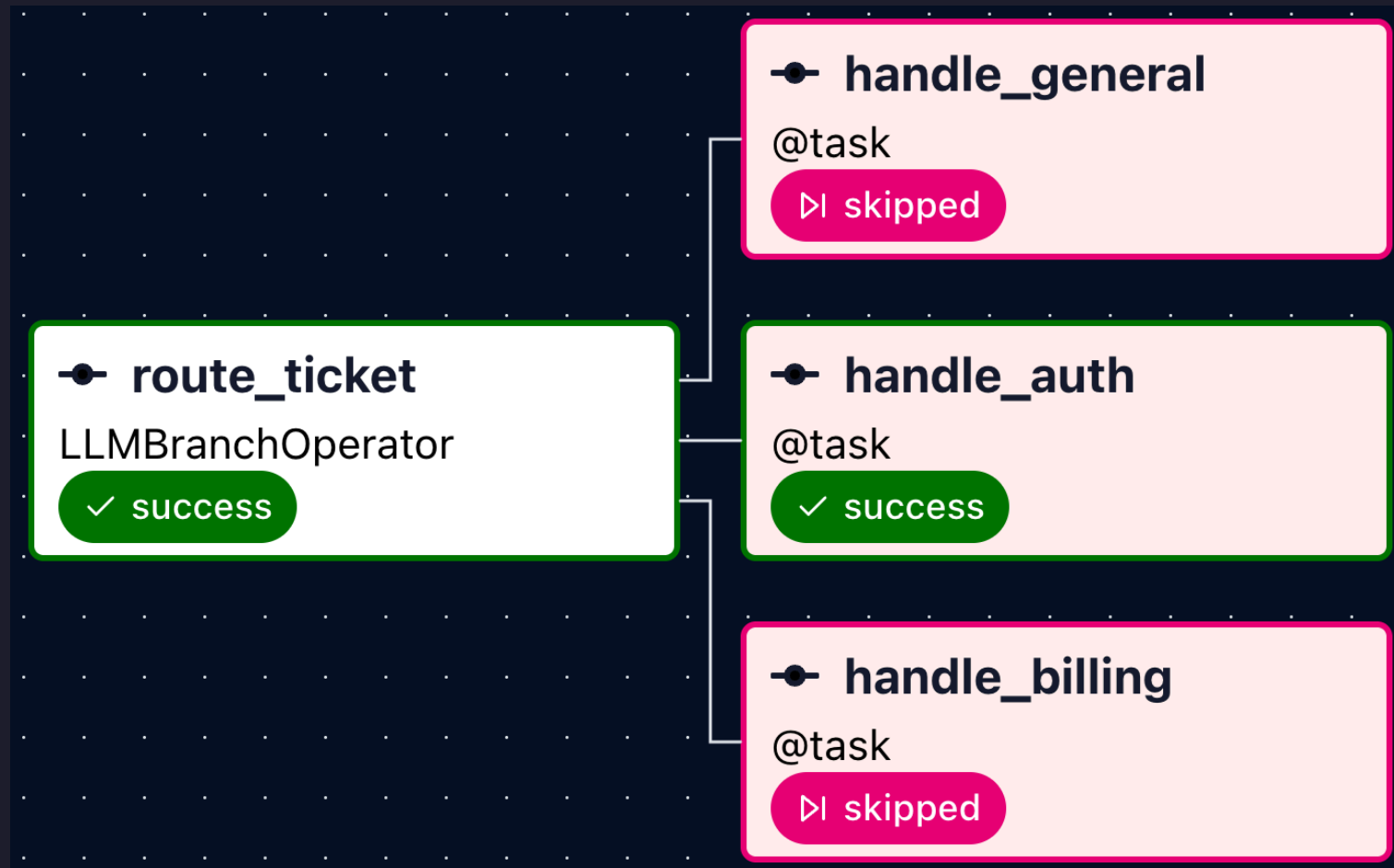
One batch job, about **half the price**.

OpenAI or Anthropic batch API · deferred, so no worker slot is held.

@task.llm_branch

```
@task.llm_branch(  
    llm_conn_id="pydanticai_default",  
    system_prompt="Route support tickets to the right team.",  
)  
def route_ticket(message: str):  
    return f"Route this support ticket: {message}"  
  
route_ticket("I was charged twice for my subscription.") >> [  
    handle_billing(), handle_auth(), handle_general(),  
]
```

1 branch runs, the rest skip



Branch on confidence

```
LLMBranchOperator(  
    ...,  
    branches={  
        "rerun": "Looks transient.",  
        "page_oncall": BranchOption("Fix it now.", min_confidence=0.9),  
        "ignore": "Harmless.",  
    },  
    decision_policy=DecisionPolicy(min_confidence=0.6),  
)
```

Set the bar, **per decision**

0.97 → rerun

Over the bar: it runs by itself

0.62 → page_oncall

Under its bar: a person decides

Not a guess: a number you chose, per branch.

Hand it Airflow's tools.

Dag

Task · every attempt

prompt

@task.llm

branch

@task.llm_branch

tools

toolsets

human

approval

retries

attempts

**Your connections
are the tool layer.**

AgentOperator + SQLToolset

```
AgentOperator(  
    task_id="analyst",  
    prompt="What are the top 5 customers by order count?",  
    llm_conn_id="pydanticai_default",  
    toolsets=[SQLToolset(  
        db_conn_id="postgres_default",  
        allowed_tables=["customers", "orders"],  
        max_rows=20,  
    )],  
)
```

What the agent **can't touch**

`allowed_tables` — other tables are refused

Read-only unless `allow_writes=True`

Every tool call, in the task log

```
► Log message source details
► Pre Execute
  4 [2026-10-06 17:50:19] INFO - Task instance is in running state
  5 [2026-10-06 17:50:19] INFO - Previous state of the Task instance: TaskInstanceState.QUEUED
  6 [2026-10-06 17:50:19] INFO - Current task name:analyst
  7 [2026-10-06 17:50:19] INFO - Dag name:example_agent_operator_sql
  8 [2026-10-06 17:50:20] INFO - Using explicit credentials for provider with model 'openai:gpt-oss:20b': ['api_key', 'base_url']
  9 [2026-10-06 17:51:21] INFO - HTTP Request: POST http://host.docker.internal:11434/v1/responses "HTTP/1.1 200 OK"
► Tool call: list_tables
 11 [2026-10-06 17:51:32] INFO - HTTP Request: POST http://host.docker.internal:11434/v1/responses "HTTP/1.1 200 OK"
► Tool call: get_schema
 13 [2026-10-06 17:52:01] INFO - HTTP Request: POST http://host.docker.internal:11434/v1/responses "HTTP/1.1 200 OK"
▼ Tool call: query
 14 [2026-10-06 17:52:01] INFO - Running statement: SELECT c.id AS customer_id, c.name AS customer_name, COUNT(*) AS order_count
    FROM orders o
    JOIN customers c ON o.customer_id = c.id
    GROUP BY c.id, c.name
    ORDER BY order_count DESC
    LIMIT 5, parameters: None
 15 [2026-10-06 17:52:01] WARNING - You are using a Python version (3.10.21) which Google will stop supporting in new releases
 16 [2026-10-06 17:52:02] WARNING - You are using a Python version (3.10.21) which Google will stop supporting in new releases
 17 [2026-10-06 17:52:02] INFO - Rows affected: 5
 18 [2026-10-06 17:52:02] INFO - Tool query returned in 0.67s
 19 [2026-10-06 17:52:10] INFO - HTTP Request: POST http://host.docker.internal:11434/v1/responses "HTTP/1.1 200 OK"
► LLM run complete: model=gpt-oss:20b, requests=4, tool_calls=3, input_tokens=2532, output_tokens=408, total_tokens=2940
► Post Execute
 23 [2026-10-06 17:52:10] INFO - Task instance in success state
 24 [2026-10-06 17:52:10] INFO - Previous state of the Task instance: TaskInstanceState.RUNNING
 25 [2026-10-06 17:52:10] INFO - Task operator:<Task(AgentOperator): analyst>
```

Any hook becomes a toolset

```
reports = HookToolset(  
    S3Hook(aws_conn_id="aws_default"),  
    allowed_methods=["list_keys", "read_key"],  
    pinned_arguments={"bucket_name": "acme-reports"},  
)
```

Airflow as an MCP client

```
AgentOperator(  
    task_id="mcp_agent",  
    prompt="What tools are available? Run the hello tool.",  
    llm_conn_id="pydanticai_default",  
    toolsets=[  
        MCPToolset(mcp_conn_id="my_mcp_server"),  
    ],  
)
```

SandboxToolset

```
AgentOperator(  
    task_id="analyst",  
    prompt="Find the outliers in this sample.",  
    llm_conn_id="pydanticai_default",  
    toolsets=[SandboxToolset(ModalSandboxBackend())],  
)
```

Why a **sandbox**?

The Airflow worker

**Your connections,
variables, environment**

The sandbox

**Model-written code. No
secrets. No network.**

The agent writes the code. It shouldn't run it next to your credentials.

More toolsets on the **Airflow Registry**

 **Airflow** | Registry

ProvidersStatsExploreAPI ExplorerDocs [↗](#)

Search... [🔍](#)

🌙

llamaindexOpenAI

All (36)

Operators (10)

U

Toolsets (8)

H

Hooks (7)

@

Decorators (7)

R

Retry Policies (2)

T

Triggers (1)

P

Plugins (1)

Categories

All36

Common AI

Pydantic AI

MCP Server

Modules

☐ Durable (includes deferrable) (4) ☐ Deferrable only (2)

Search modules...

U

HookToolset

Expose selected methods of an Airflow Hook as pydantic-ai tools.

`airflow.providers.common.ai.toolsets.hook.HookToolset`

📄

[View Docs](#) [↗](#) [Source](#) [↗](#)

Talk back to it.

Dag

Task · every attempt

prompt

@task.llm

branch

@task.llm_branch

tools

toolsets

human

approval

retries

attempts

1 • Approve the output

```
LLMOperator(  
    task_id="summarize_with_approval",  
    prompt="Summarize the quarterly financial report for stakeholders.",  
    llm_conn_id="pydanticai_default",  
    require_approval=True,  
    approval_timeout=timedelta(hours=24),  
    allow_modifications=True,  
)
```

Approve, reject, or edit

Logs Required Action (1) Rendered Templates Storage Asset Events Audit Log Code Details

Review output for task `summarize\_with\_approval`

text



```
1 Prompt: Summarize the quarterly financial report for stakeholders.  
2  
3 I'm happy to help—could you please share the key figures or the full quarterly report (e.g., revenue, expenses, net income)?
```

output *

I'm happy to help—could you please share the key figures or the full quarterly report (e.g., revenue, expenses, net income)?

Edit the output before approving (optional).

Approve

Reject

2 · Have a conversation

```
AgentOperator(  
    task_id="summarize_with_review",  
    prompt="Summarize the Q4 sales report in 3 bullet points.",  
    llm_conn_id="pydanticai_default",  
    enable_hitl_review=True,  
    max_hitl_iterations=5,  
    hitl_timeout=timedelta(minutes=30),  
)
```

A chat with a running task

AI ASSISTANT

I'm ready to summarize, but I'll need the Q4 sales data first. Could you share the key figures or a brief overview of the report?

Iteration 1

A chat with a running task

AI ASSISTANT

I'm ready to summarize, but I'll need the Q4 sales data first. Could you share the key figures or a brief overview of the report?

Iteration 1

YOU

There is no report attached. Write three bullet points from these numbers instead: revenue up 18%, churn down to 2.1%, EMEA now 40% of new deals.

Iteration 1

A chat with a running task

AI ASSISTANT

I'm ready to summarize, but I'll need the Q4 sales data first. Could you share the key figures or a brief overview of the report?

Iteration 1

YOU

There is no report attached. Write three bullet points from these numbers instead: revenue up 18%, churn down to 2.1%, EMEA now 40% of new deals.

Iteration 1

AI ASSISTANT

Q4 revenue grew by **18%**, underscoring strong overall demand.

Churn fell to 2.1%, indicating improved customer retention.

EMEA now represents 40% of new deals, highlighting a geographic shift in growth.

Iteration 2

All of it, on the **task page**

The screenshot displays a task page for a workflow named 'example_agent_operator_hitl_review'. The interface includes a sidebar with navigation options: Dashboard, Dags, Assets, Browse, Admin, Security, Docs, and User. The main content area shows the task instance 'summarize_with_review' in a 'Running' state. A table lists the operator as 'AgentOperator', the start date as '2026-10-06 17:50:19', the end date as '5m 25s', and the DAG version as 'v1'. Below this, a 'HITL Review' section shows a conversation between an AI Assistant and a user. The AI Assistant asks for Q4 sales data, and the user provides a summary of Q4 revenue, churn, and EMEA growth. The task is currently in a 'Pending Review' state, and the interface indicates it is 'Waiting for your review'.

Dag Run: example_agent_operator_hitl_review

Task Instance: summarize_with_review (Running)

OPERATOR	START DATE	END DATE	DURATION	DAG VERSION
AgentOperator	2026-10-06 17:50:19		5m 25s	v1

Add a note...

Logs | Rendered Templates | Storage | Asset Events | Audit Log | Code | Details | **HITL Review**

HITL Review

Task: summarize_with_review | DAG: example_agent_operator_hitl_review | Iteration: 2/5 | Pending Review

AI ASSISTANT

I'm ready to summarize, but I'll need the Q4 sales data first. Could you share the key figures or a brief overview of the report?

Iteration 1

YOU

There is no report attached. Write three bullet points from these numbers instead: revenue up 18%, churn down to 2.1%, EMEA now 40% of new deals.

Iteration 1

AI ASSISTANT

Q4 revenue grew by **18%**, underscoring strong overall demand.
Churn fell to 2.1%, indicating improved customer retention.
EMEA now represents 40% of new deals, highlighting a geographic shift in growth.

Iteration 2

Waiting for your review

3 · Approve each tool call

Airflow 3.3+

```
shop = FunctionToolset(tools=[lookup_order, refund_order]).approval_required(
    lambda ctx, tool_def, args: tool_def.name == "refund_order"
)
```

```
AgentOperator(
    task_id="handle_ticket",
    llm_conn_id="pydanticai_default",
    prompt="The customer says order 7 never arrived. Resolve it.",
    toolsets=[shop],
    tool_approval_timeout=timedelta(hours=4),
)
```

Waiting, without holding a worker

Approve tool call for task `handle\_ticket`

The agent wants to run:

refund_order

json



```
1 {  
2   "order_id": 7  
3 }
```

reason

Sent to the agent when you reject the call (optional).

Approve

Reject

1

Output

`require_approval`

“Is this right?”

2

Conversation

`enable_hitl_review`

“Not quite. Try again.”

3

Tool call

`approval_required`

“May I do this?”

Pick the level that matches the risk.

Why orchestrate this with Airflow?

Dag

Task · every attempt

prompt

@task.llm

branch

@task.llm_branch

tools

toolsets

human

approval

retries

attempts

**Why not just a Python
agent framework?**

durable=True

```
@dag(default_args={"retries": 3, "retry_delay": timedelta(seconds=30)})
def example_agent_durable_operator():
    AgentOperator(
        task_id="durable_analyst",
        prompt="What are the top 5 customers by order count?",
        llm_conn_id="pydanticai_default",
        durable=True,
        toolsets=[SQLToolset(db_conn_id="postgres_default", ...)],
    )
```

Retries replay completed model and tool steps from cache.

Replayed, not re-run

Task Tries

1



2



All Log Levels



All Sources



Q Search logs...

```
► Log message source details
► Pre Execute
  4 [2026-10-06 17:52:56] INFO - Task instance is in running state
  5 [2026-10-06 17:52:56] INFO - Previous state of the Task instance: TaskInstanceState.QUEUED
  6 [2026-10-06 17:52:56] INFO - Current task name:durable_analyst
  7 [2026-10-06 17:52:56] INFO - Dag name:talk_durable_failing_once
  8 [2026-10-06 17:52:57] INFO - Using explicit credentials for provider with model 'openai:gpt-oss:20b': ['api_key', 'base
► Tool call: list_tables
► Tool call: get_schema
► Tool call: get_schema
► Tool call: check_query
► Tool call: query
 18 [2026-10-06 17:52:58] WARNING - Durable: cached model response does not match the current request; re-running this step
 19 [2026-10-06 17:53:13] INFO - HTTP Request: POST http://host.docker.internal:11434/v1/responses "HTTP/1.1 200 OK"
 20 [2026-10-06 17:53:13] INFO - Durable: replayed 9 cached steps (5 model, 4 tool), cached 2 new steps (1 model, 1 tool)
► LLM run complete: model=gpt-oss:20b, requests=1, tool_calls=1, input_tokens=1029, output_tokens=134, total_tokens=1163
► Post Execute
 25 [2026-10-06 17:53:13] INFO - Previous state of the Task instance: TaskInstanceState.RUNNING
 26 [2026-10-06 17:53:13] INFO - Task operator:<Task(Agent0operator): durable_analyst>
```

What the retry **cost**

`durable=False`

4 model calls

2,773 tokens. The whole conversation, again.

`durable=True`

1 model call

1,163 tokens. 9 steps replayed from the cache.

Same Dag, same failure, 1 run each, local gpt-oss:20b.

A retry pays only for **what's new**

Replayed steps never reach the model

So they aren't billed

And they don't count against `usage_limits`

A dollar cap, across every attempt

```
AgentOperator(  
    ...,  
    usage_limits=UsageLimits(  
        request_limit=5,  
        input_tokens_limit=4_000,  
        output_tokens_limit=1_000,  
        cost_limit=Decimal("0.50"),  
    ),  
)
```

Fail over to another vendor

```
LLMOperator(  
    task_id="summarize_with_an_operator_owned_chain",  
    prompt="Summarize the key findings from the Q4 earnings report.",  
    llm_conn_id="llm_primary_down_no_chain",  
    fallback_conn_ids=["llm_fallback"],  
)
```

Same attempt, different model

```
► Log message source details
► Pre Execute
  4 [2026-10-06 17:50:20] INFO - Task instance is in running state
  5 [2026-10-06 17:50:20] INFO - Previous state of the Task instance: TaskInstanceState.QUEUED
  6 [2026-10-06 17:50:20] INFO - Current task name:summarize_through_the_chain
  7 [2026-10-06 17:50:20] INFO - Dag name:example_llm_fallback
  8 [2026-10-06 17:50:21] INFO - Using explicit credentials for provider with model 'openai:gpt-5-mini': ['api_key', 'base_url']
  9 [2026-10-06 17:50:21] INFO - Resolving LLM fallback chain: llm_primary_down -> llm_fallback
 10 [2026-10-06 17:50:21] INFO - Using explicit credentials for provider with model 'openai:gpt-oss:20b': ['api_key', 'base_url']
 11 [2026-10-06 17:50:21] INFO - Retrying request to /responses in 0.400175 seconds
 12 [2026-10-06 17:50:22] INFO - Retrying request to /responses in 0.950524 seconds
 13 [2026-10-06 17:51:10] INFO - HTTP Request: POST http://host.docker.internal:11434/v1/responses "HTTP/1.1 200 OK"
► LLM run complete: model=gpt-oss:20b, requests=1, tool_calls=0, input_tokens=97, output_tokens=497, total_tokens=594
► Post Execute
```

Let a model read the error

AIP-105 · Airflow 3.3+

```
llm_policy = LLMRetryPolicy(  
    llm_conn_id="pydanticai_default",  
    fallback_rules=[  
        RetryRule(exception=ConnectionError, action=RetryAction.RETRY,  
                   retry_delay=timedelta(seconds=10)),  
        RetryRule(exception=PermissionError, action=RetryAction.FAIL),  
    ],  
)  
  
@task(retries=3, retry_policy=llm_policy)  
def task_auth_error(): ...
```

Should this retry?

Stopped on try 1 of 4

ⓘ

data: The error reports a type mismatch between the expected INT type and an actual STRING in the input data. This is a schema/data validation issue that requires correcting the data or schema, not a transient or retryable error.

Task Instance

task_data_error

✖ Failed

↺

✓ / ✖

OPERATOR	START DATE	END DATE	DURATION	DAG VERSION
@task	2026-10-06 17:50:19	2026-10-06 17:51:38	1m 20s	v1

Rules → model → default

```
from airflow.sdk import ChainRetryPolicy

ChainRetryPolicy([
    ExceptionRetryPolicy(rules=[...]),          # rules
    LLMRetryPolicy(llm_conn_id="pydanticai_default"), # model
    ExceptionRetryPolicy(rules=[...]),          # floor
])
```

The first RETRY or FAIL wins. All DEFAULT: the task's own retries. Airflow 3.4+ (3.3 via common.compat).

Swap the model, not the Dag

```
"model": "openai:gpt-5-mini"
```

```
"model": "anthropic:claude-haiku-4-5-20251001"
```

openai:

anthropic:

google:

bedrock:

azure:

groq:

mistral:

snowflake:

Ollama, vLLM, any OpenAI-compatible server: openai:

Remember the **fake SDK**?

load yesterday's orders from Postgres

a plain Dag: a schedule and a SQL task

Remember the **fake SDK**?

load yesterday's orders from Postgres

a plain Dag: a schedule and a SQL task

If anything looks off

`@task.11m_branch`

Remember the **fake SDK**?

load yesterday's orders from Postgres

a plain Dag: a schedule and a SQL task

If anything looks off

`@task.llm_branch`

page #data-oncall

`HookToolset`

Remember the **fake SDK**?

load yesterday's orders from Postgres

a plain Dag: a schedule and a SQL task

If anything looks off

`@task.llm_branch`

page #data-oncall

`HookToolset`

if it fails, try again, unless the
password is wrong

`LLMRetryPolicy`

Remember the **fake SDK**?

load yesterday's orders from Postgres

a plain Dag: a schedule and a SQL task

If anything looks off

`@task.llm_branch`

page #data-oncall

`HookToolset`

if it fails, try again, unless the
password is wrong

`LLMRetryPolicy`

Then just tell me

it fails right away, with the model's reason
on the task

Instead of a new SDK: every line works today.

Natural language, across **all of Airflow**.

Dag

Task · every attempt

prompt

@task.llm

branch

@task.llm_branch

tools

toolsets

human

approval

retries

attempts

More to go: more **AI-native**

loop

AIP-111 Task Loops

An agent could keep trying until it converges.

Accepted

step

AIP-112 Task Steps

A retry could skip the steps that already finished.

Draft

plan

AIP-113 Dynamic Task Groups

The model could plan the graph at runtime.

Draft

gate

AIP-115 On-Demand Task Sections

A person could run optional parts later.

Draft

Wei Lee / 李唯

Apache Airflow PMC Member

wei-lee.me

How should Airflow support **your AI**?

Tell us now, or on dev@airflow.apache.org.