AIRFLOW® SUMMIT

DAG Upgrade Agent

3.0

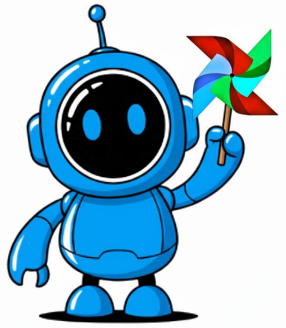
Christian Yarros

Strategic Cloud Engineer, Google Cloud Professional Services

Agenda

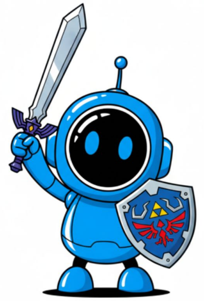
01

Airflow Popularity



Google Cloud

A Double-Edged Sword

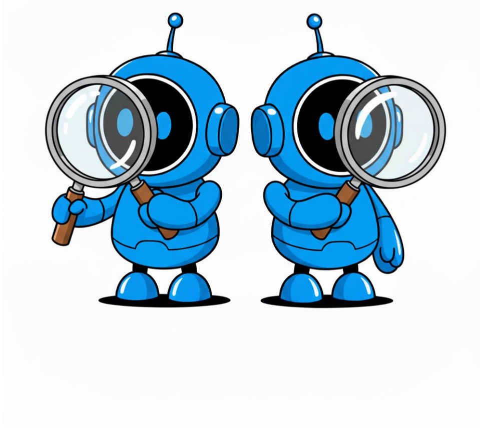


- **In the last month (GitHub Pulse):**
    - Excluding merges, 165 authors have pushed 635 commits to main and 970 commits to all branches.
    - On main, 2209 files have changed and there have been 54,909 additions and 24,446 deletions
- **FANTASTIC** that we have such an active community contributing to the project.
- **BUT:** One of the biggest complaints among our customers is how often they need to upgrade their Apache Airflow and provider packages to remain on a supported version.

Google Cloud

02

What does a DAG upgrade look like?

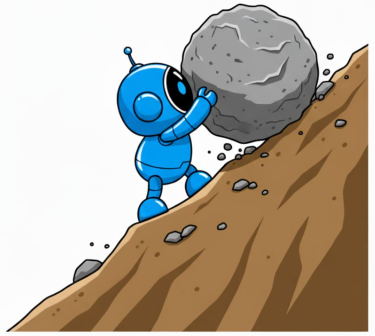


Google Cloud

The Upgrade Process

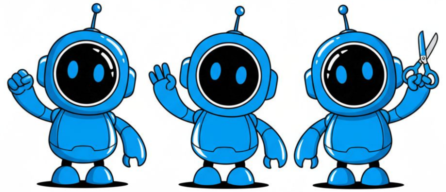
1. Understand your code
2. Look at the Airflow Release Notes
3. Identify the Provider Packages you use, and check their Changelogs.
4. Mentally filter what's relevant to DAG code and what isn't.
5. Figure out what's been deprecated (imports, operators, parameters, etc.)
6. Rewrite the DAG Code
7. Ensure it parses successfully
8. Ensure business logic is maintained
9. Deploy latest version

→ **Repeat for every single DAG in your inventory**



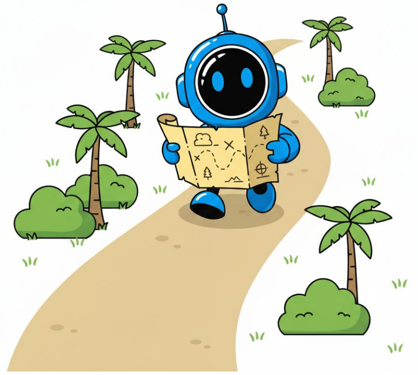
Google Cloud

03



Reality Check:
What are our options?

Choose your journey



1. **Manual**: Tedious, redundant, takes forever.
2. **Programmatic Tools**: Static, brittle, complex, and need to be maintained. Fine if you want to cover EVERY single case.
3. **Foundational LLMs**: Full of Hallucinations, old data, search tools go anywhere
4. Or...

04

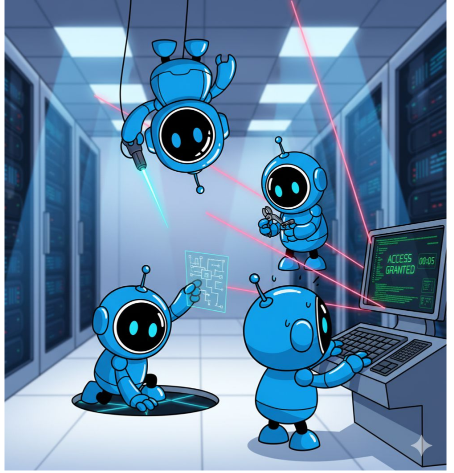
The DAG Upgrade Agent

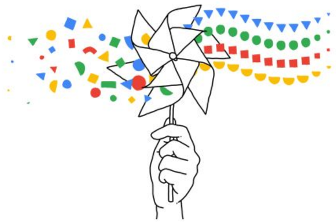
Google Cloud

The Challenge Ahead

Apache Airflow is **notoriously difficult** for foundational LLMs...

- Hundreds of similar sounding Operators, Sensors, Hooks, and Parameters across dozens of Providers
- Subtle changes and deprecations across many versions of code
- Training cutoffs that prevent awareness of recent Airflow code
- Foundational models are trained on vast amounts of data, evaluating the specifics of Airflow code is like finding a needle in a haystack
- Inclination to always say “Yes” - LLMs will attempt to generate code even if it lacks confidence in the results.



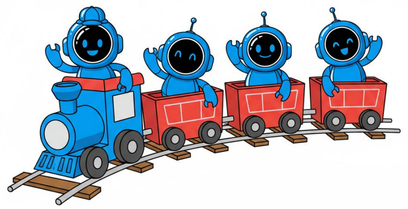


Google Cloud

Goals and Guardrails

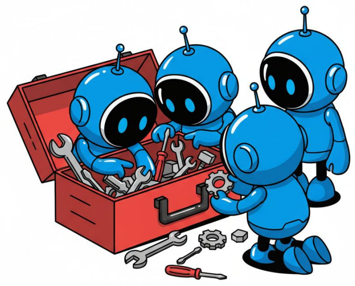
1. **Every change must be transparent and identifiable.**
   a. We must know what the DAG Upgrade Agent did.
2. **Every change must be referenced.**
   a. The DAG Upgrade Agent cannot introduce a code change unless referenced by release notes.
3. **Every change must be explained.**
   a. The DAG Upgrade Agent must provide reasoning for its changes.
4. **Business Logic must not be altered**
   a. Variable, function, object naming conventions must be followed, and business logic must be preserved.

The goal is to **accelerate** DAG upgrades by at least 50%.



Google Cloud

Techniques of the AI Toolbox

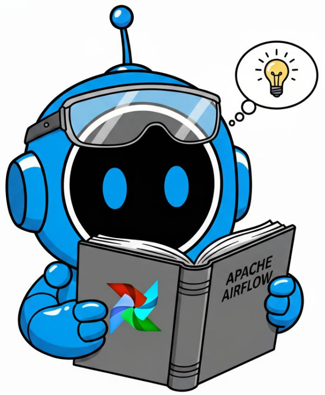


1. **Prompt Engineering:** Crafting the perfect question to get the best answer from an AI.
2. **Context Engineering:** Managing all the information an AI sees to keep it focused and accurate.
3. **Retrieval Augmented Generation (RAG):** Giving an AI an "open book" to look up fresh, external facts before it answers.
4. **Supervised Fine-Tuning:** Training a general AI model on custom data to make it a specialist for a specific task.
5. **Thinking Models:** AI that "shows its work" step-by-step to improve complex reasoning.
6. **Agents:** AI that can take actions and use tools to autonomously complete a goal.

What we'll use

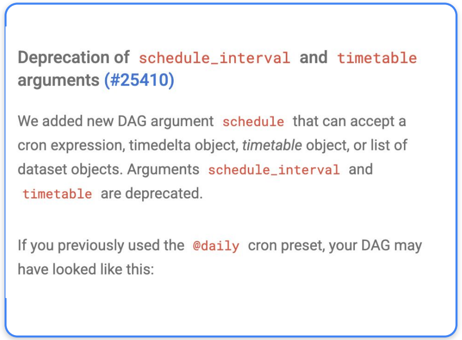
Agents + System Instructions + Tools / Context Engineering + RAG Knowledge base.

1. **Google ADK Agents** are superior to classical LLMs because they can understand context, intent, and nuance. Thereby remaining more flexible than programmatic tools. We can divide responsibilities across multiple agents as well.
2. **System Instructions:** a form of "meta-prompting." telling the model *how* to answer all questions.
3. **Vertex AI RAG Engine** confines responses to a specific corpus of knowledge. This knowledge is **curated** to include latest information, reduce noise, and therefore reduce hallucinations.
4. **Context Engineering** through tooling immediately aligns the user's request with referential information. This context **derives from** the RAG knowledgebase, allowing the Agent to quickly retrieve relevant information.



A Peek into Knowledge Setup

Source Release Notes



Deprecation of `schedule_interval` and `timetable` arguments (#25410)

We added new DAG argument `schedule` that can accept a cron expression, timedelta object, *timetable* object, or list of dataset objects. Arguments `schedule_interval` and `timetable` are deprecated.

If you previously used the `@daily` cron preset, your DAG may have looked like this:

RAG Imported, Enriched Data

```
package: "airflow-core"
version: "2.4.0"
release_date: "2022-09-19"
rules:
- component: "DAG"
   summary: "schedule_interval parameter
deprecated in favor of schedule"
   change_pattern: |
     BEFORE: DAG(dag_id="my_example",
start_date=datetime(2021, 1, 1),
schedule_interval="@daily")
     AFTER:  DAG(dag_id="my_example",
start_date=datetime(2021, 1, 1),
schedule="@daily")  # schedule replaces
schedule_interval
   notes: "schedule_interval is deprecated.
schedule accepts cron expressions, timedelta
objects, timetable objects, or list of dataset
objects."
```

Shorthand Agent Context

```
package: "airflow-core"
version: "2.4.0"
release_date: "2022-09-19"
parameters:
  - component: "DAG"
    old: "schedule_interval"
    new: "schedule"
    note: "accepts cron, timedelta,
timetable"
```

Google Cloud

But wait!

Release notes are not enough.

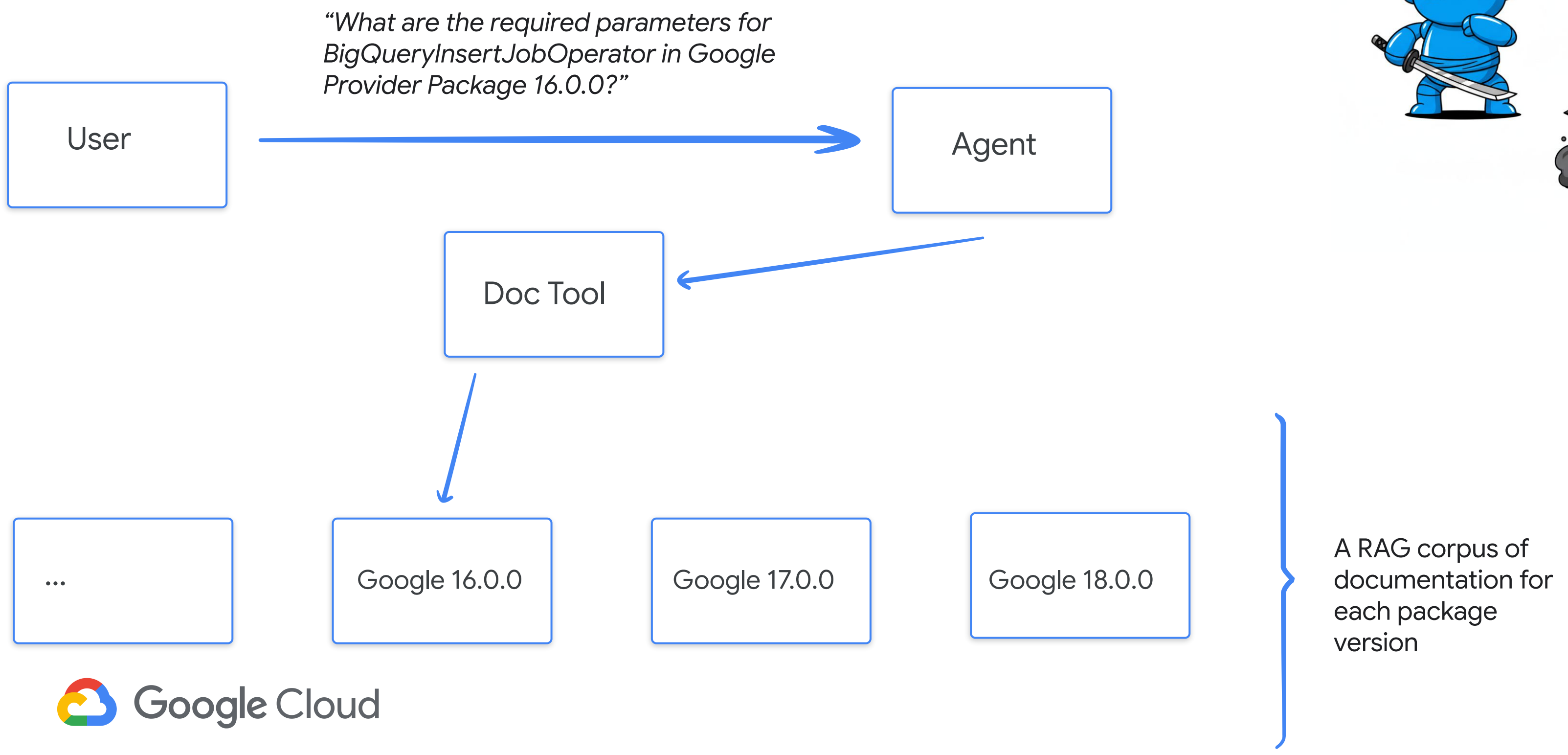
What about all the Airflow and Provider Package information not included in the release notes?

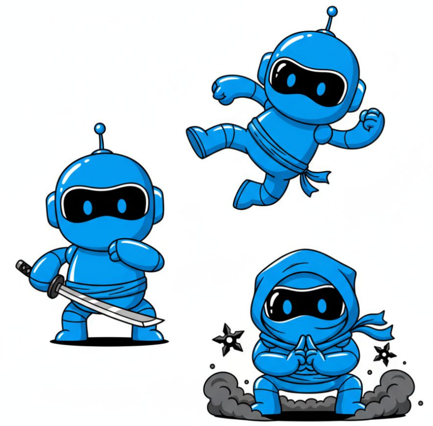
We'll need a RAG corpus containing the full documentation to ensure that new code contains the correct usage.

But not just **one** RAG corpus...

Google Cloud

Dynamic RAG Usage!

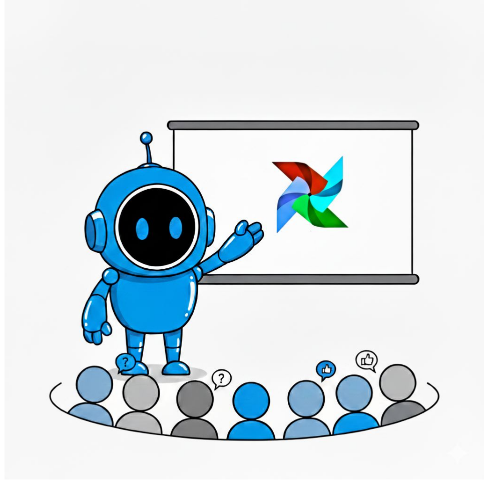

User
"What are the required parameters for BigQueryInsertJobOperator in Google Provider Package 16.0.0?"
Agent
Doc Tool
...
Google 16.0.0
Google 17.0.0
Google 18.0.0
A RAG corpus of documentation for each package version



Google Cloud

05

Demo



Prompt

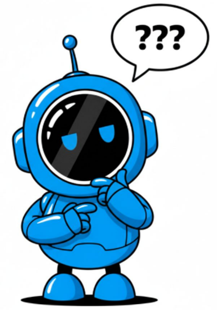
Please retrieve the following dag file: deprecated_google_dag.py

The file exists in:

- ***Composer Environment:*** *composer-summit-demo-2-7-3*
- ***Project:*** *af-summit-test-7*
- ***Location:*** *us-east4.*

Then perform an upgrade on this dag file. The desired target versions are:

- ***Airflow:*** *2.10.5*
- ***Google:*** *18.0.0*

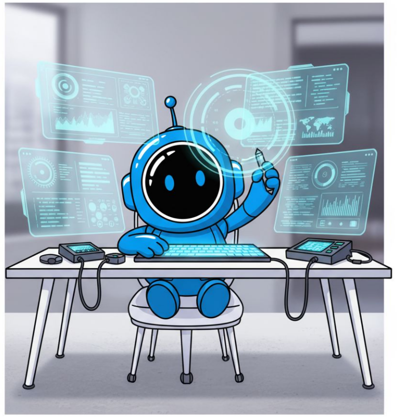


http://screencast/cast/NTQ5MDMyNDU4MjQzMjc2OHwzNzc2OTI0ZS1iMg

Google Cloud

06

Results



Google Cloud

Estimated Time Savings

#	Activity	Manual	Specialized DAG Upgrader Agent	Time Savings %
1	Understand your code	~30 min	~10 seconds (Agent)	~99%
2	Look at Airflow Release Notes	~30 min	~10 seconds (RAG)	>99%
3	Check Provider Changelogs	~30 min	~10 seconds (RAG)	>99%
4	Filter relevant changes	~20 min	~10 second (Context)	>99%
5	Find deprecated code	~10 min	~10 seconds (Context)	>99%
6	Rewrite the DAG	~30 min	~30 seconds (Agent)	>99%
7	Ensure it parses successfully	~30 min	~10 seconds (Agent)	>99%
8	Ensure business logic is maintained	~60 min	~60 min	0%
9	Deploy latest version	~10 min	~10 seconds (Agent)	>99%
	TOTAL ESTIMATED TIME	**~4 hours**	**~60 minutes**	**75%**

And 99% of the accelerated time frame is focused on business logic!

Google Cloud

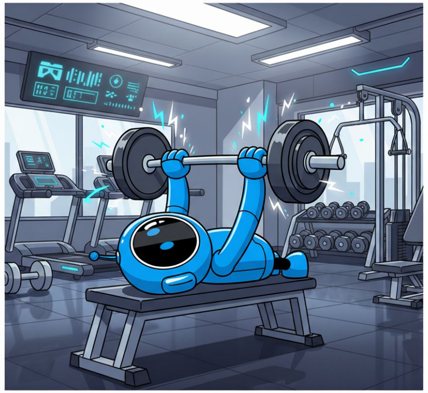
07

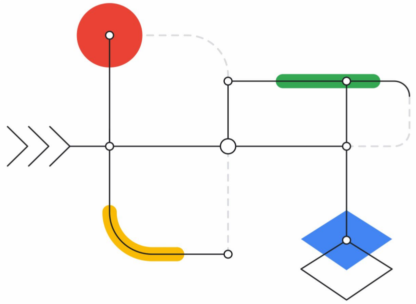
Next Steps

Google Cloud

Always room for improvement

- Continuously enhancing RAG data quality, scale, and freshness
- Add support for more providers, frameworks for custom operator, or business logic integration
- Agent tools for edge use cases
- More Feedback loops
- Evaluation datasets
- Integration into a larger suite of Airflow DAG Development agents. DAG Generations, Conversions, Optimizations.

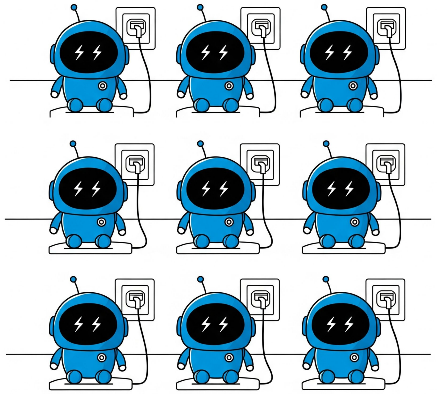




Google Cloud

Thank you!

Questions?



Christian Yarros

linkedin.com/in/cyarros