Running Tasks Anywhere in Any language

Introduction

Vikram Koka
Chief Strategy Officer, Astronomer
Airflow Committer

Ash Berlin-Taylor
Airflow Committer & PMC Member
Engineering Leader @ Astronomer

Airflow’s usage evolution

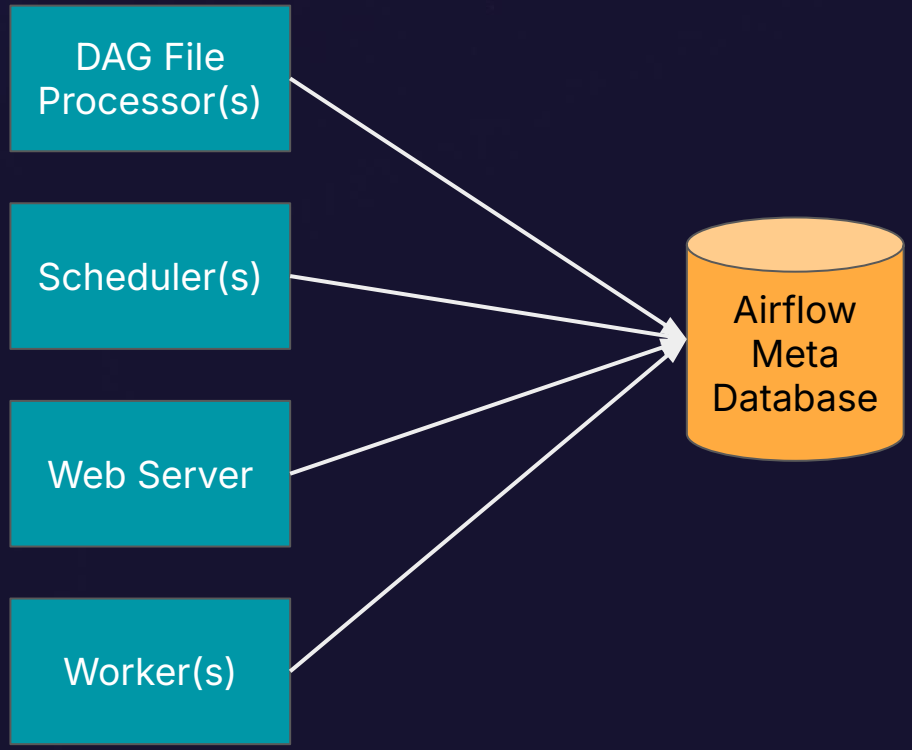
Operational challenges

1. Task Isolation

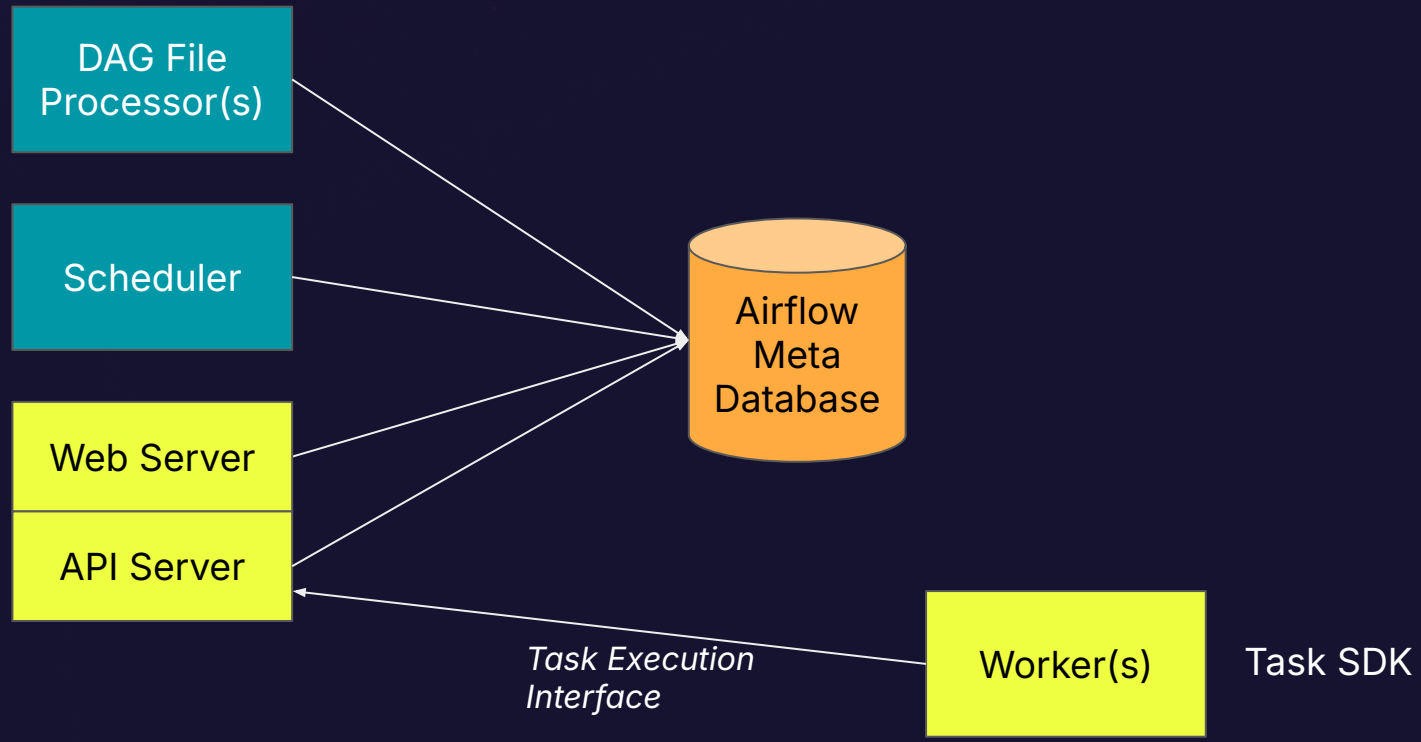
2. Dependency Management

3. Airflow Upgrades at scale

Current Airflow architecture



Architectural decoupling:
Task Execution Interface



Key Benefits

Task Isolation leading to an improved security posture
- Particularly important for shared deployments with varied data access
- Especially when connection information is directly stored in metaDB

Independent versioning of Python dependencies between Tasks
- Less need for KubePodOperator

Independent Python version upgrades from Airflow server
- Platform administrators vs. Data teams

Python based local execution

Airflow system components
- Web Server now includes an API Server

Airflow Python Task SDK
- Separate distribution
- Provider interfaces / dependencies

Airflow Task SDK

```
from airflow import DAG
```

becomes

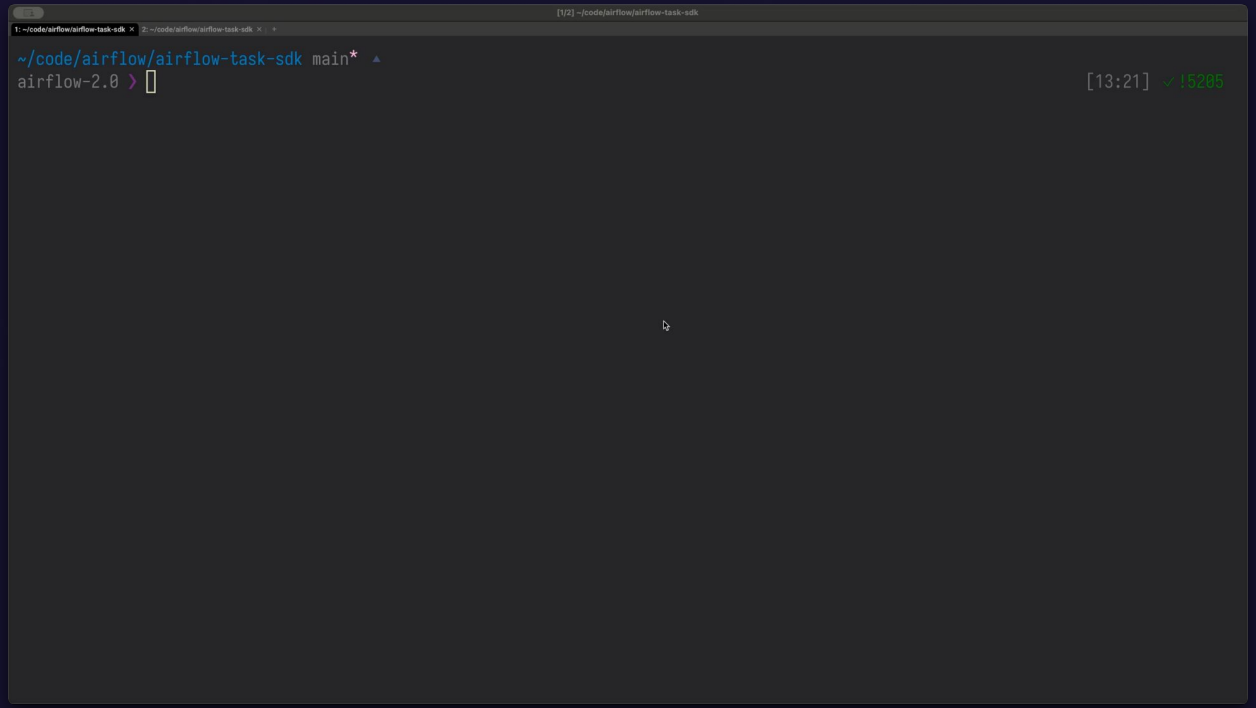
```
from airflow.sdk import DAG
```

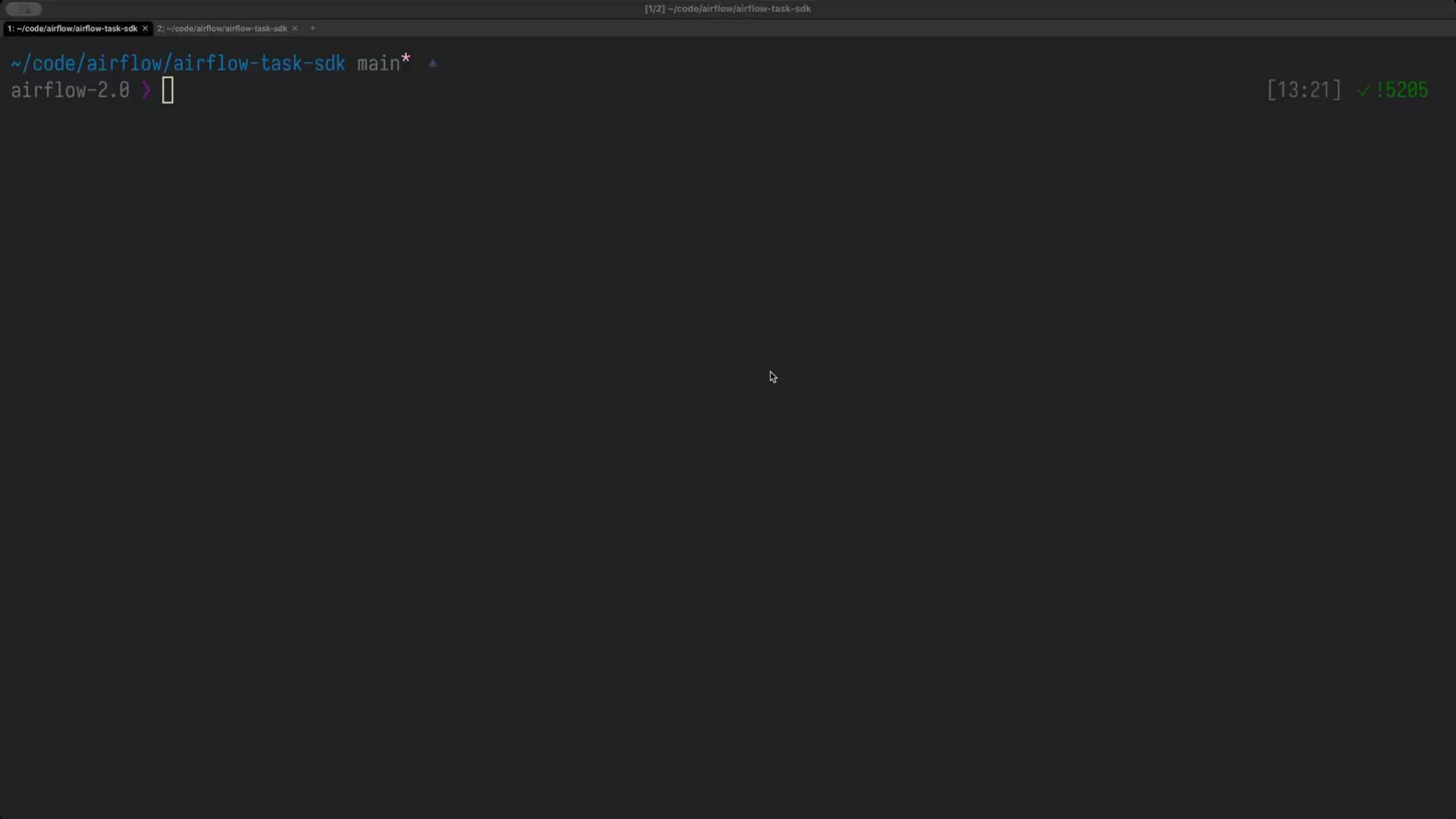
etc.

Wait, do I have to re-write every one of my DAGS?!

Of course not

Standard Execution Code + Demo




[1/2] ~/code/airflow/airflow-task-sdk
1: ~/code/airflow/airflow-task-sdk
2: ~/code/airflow/airflow-task-sdk
~/code/airflow/airflow-task-sdk main*
airflow-2.0 ❯
[13:21] ✓!5205

Task Execution Interface

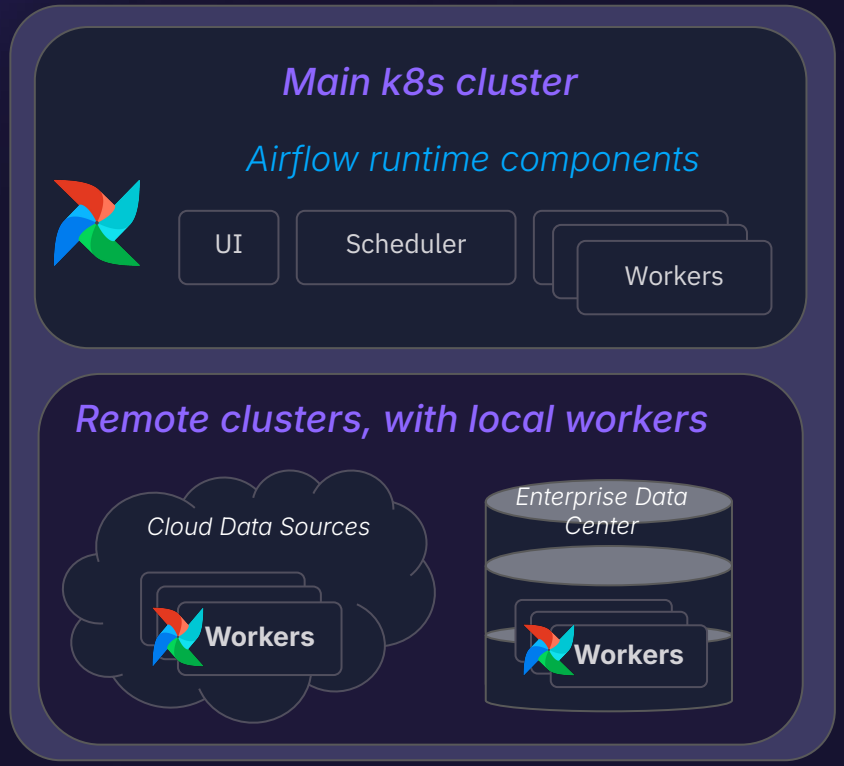
Includes Task Context:
- Connections
- Env variables / Secrets
- XCom input data

Includes Task Status:
- Task completion status
- Task Heartbeat
- Reading and writing XCom data
- Logs and Metrics

What’s not allowed:
- Direct access Airflow metadatabase

One more thing …

Run anywhere



Remote / Edge execution: run tasks on workers in remote clusters

Use cases

- Deployment flexibility with workers on public, hybrid, private cloud, on-prem, edge, GPU clouds
- Higher resilience and scalability
- Improved security isolation
- Easier upgrades, fewer dependencies
- Better meet data locality mandates

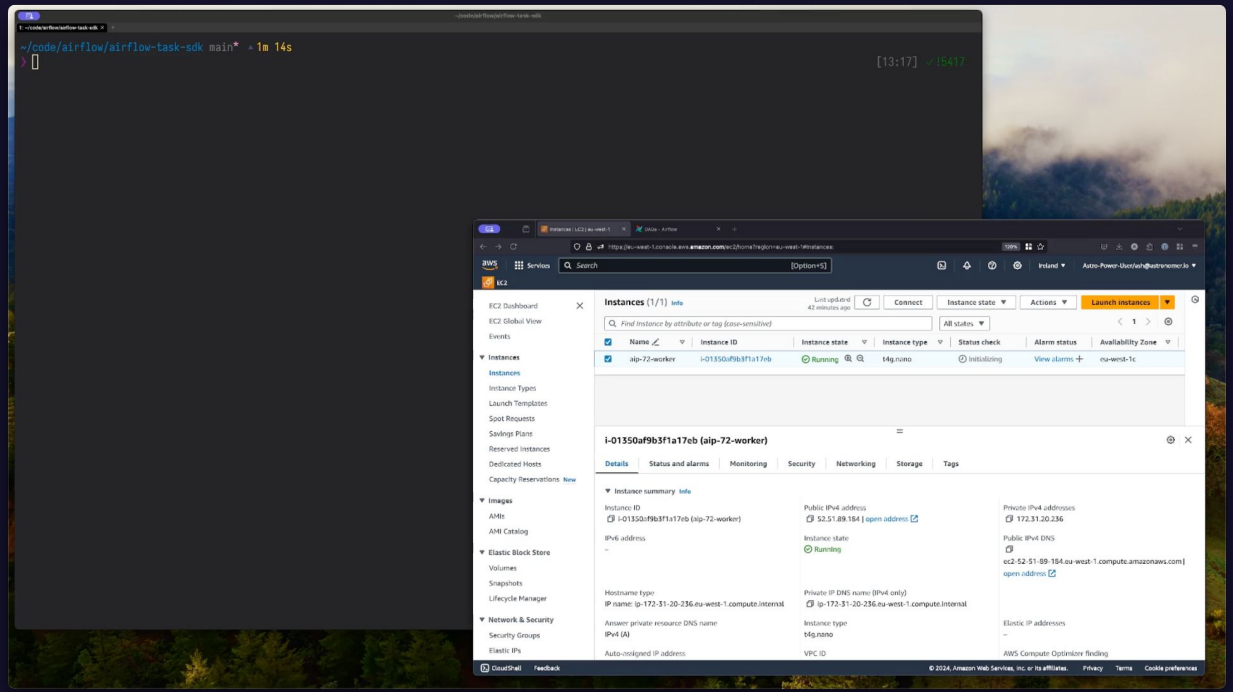
Python based remote execution

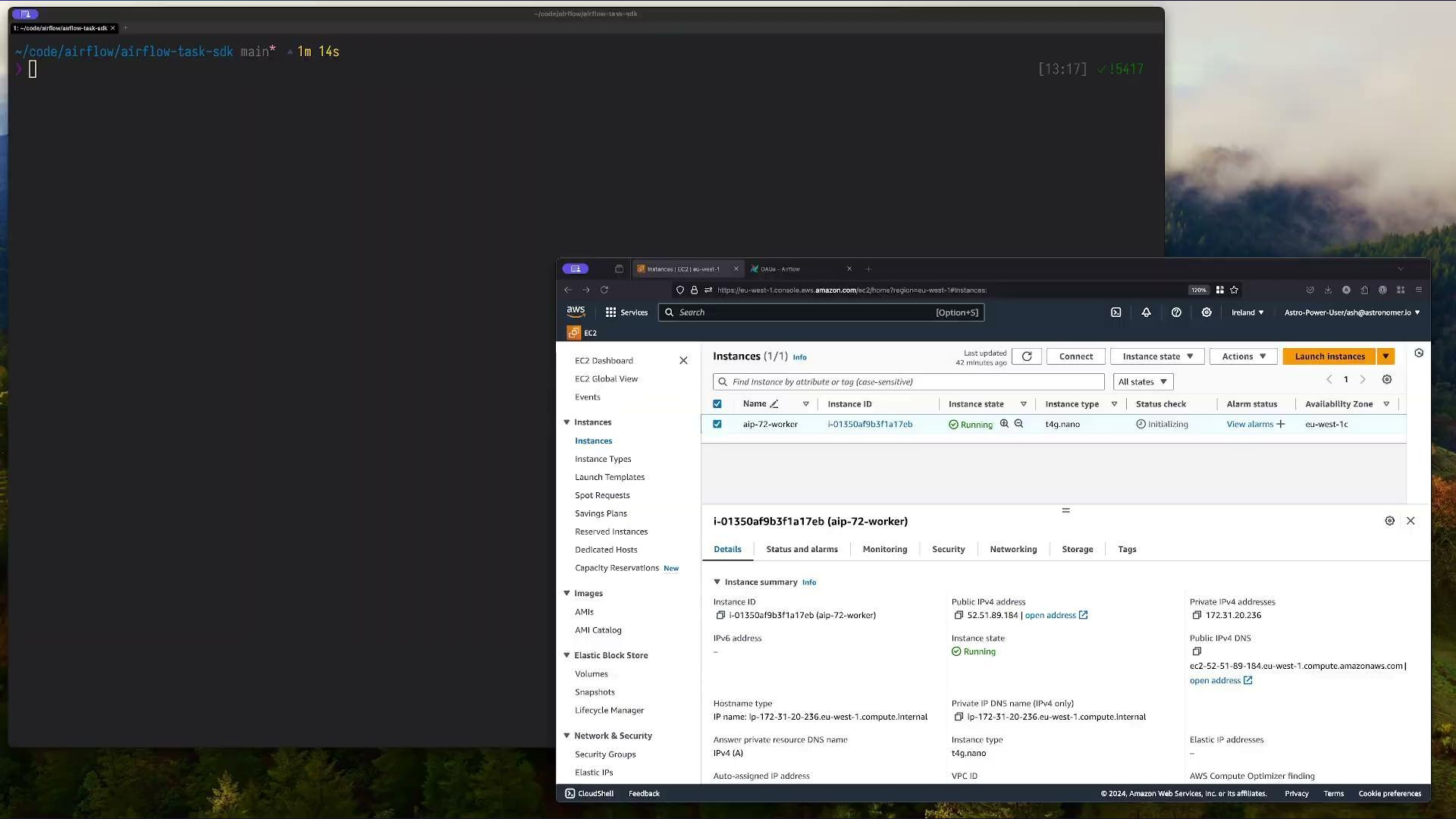
What needs to change in my DAG code?

What limitations exist?

Note for AIP-69 Remote / Edge Executor

Python Remote Worker




~/code/airflow/airflow-task-sdk main* 1m 14s
[13:17] !5417
Instances | EC2 | eu-west-1
DAGs - Airflow
https://eu-west-1.console.aws.amazon.com/ec2/home?region=eu-west-1#Instances:
Services
Search
[Option+S]
Ireland
Astro-Power-User/ash@astronomer.io
EC2
EC2 Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Instances (1/1) Info
Last updated 42 minutes ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
aip-72-worker
i-01350af9b3f1a17eb
Running
t4g.nano
Initializing
View alarms
eu-west-1c
i-01350af9b3f1a17eb (aip-72-worker)
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Instance summary Info
Instance ID
i-01350af9b3f1a17eb (aip-72-worker)
Public IPv4 address
52.51.89.184 | open address
Private IPv4 addresses
172.31.20.236
IPv6 address
–
Instance state
Running
Public IPv4 DNS
ec2-52-51-89-184.eu-west-1.compute.amazonaws.com | open address
Hostname type
IP name: ip-172-31-20-236.eu-west-1.compute.internal
Private IP DNS name (IPv4 only)
ip-172-31-20-236.eu-west-1.compute.internal
Answer private resource DNS name
IPv4 (A)
Instance type
t4g.nano
Elastic IP addresses
–
Auto-assigned IP address
VPC ID
AWS Compute Optimizer finding
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

And, one more …

Run in any language

- Airflow 3 is language agnostic
- Software teams building data apps
  - Eg: Typescript
- Airflow 3 is also multi-lingual
  - Extract in Java
  - Transform in Python, SQL
  - Analysis using Scala
  - Incoming data into an Go app

Current languages supported

python™

Language support in 3.0

python™

Golang Task SDK

What SDK do I use?

What does my Task code look like?

What does my DAG code look like?

DAG running go Tasks – Airflow 2

```
@dag
def my_dag_with_go():
    extract = BashOperator(task_id="extract", bash_command="go run my_workflow.go extract", do_xcom_push=True)
    transform = BashOperator(task_id="transform", bash_command="go run my_workflow.go transform",
                             env={"order_data": extract.output}, do_xcom_push=True)

    @task
    def load(total_order_value: float):
        print(f"Total order value is: {total_order_value:.2f}")

    load(transform.output["total_order_value"])

my_dag_with_go()
```

DAG running go Tasks – Airflow 3

```python
@dag
def my_dag_with_go():
    go = task.external(queue="my-go-queue")
    order_data = go.extract()
    order_summary = go.transform(order_date)

    @task()
    def load(total_order_value: float):
        print(f"Total order value is: {total_order_value:.2f}")

    load(order_summary["total_order_value"])

my_dag_with_go()
```

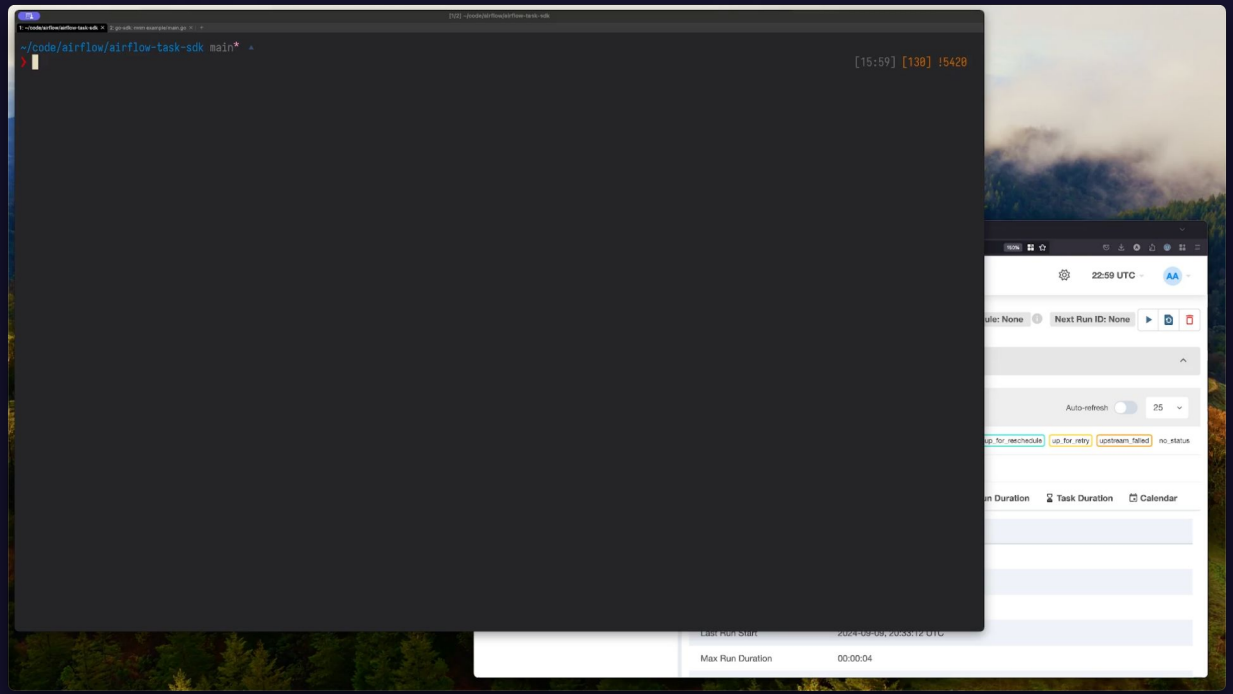
Tasks in Golang

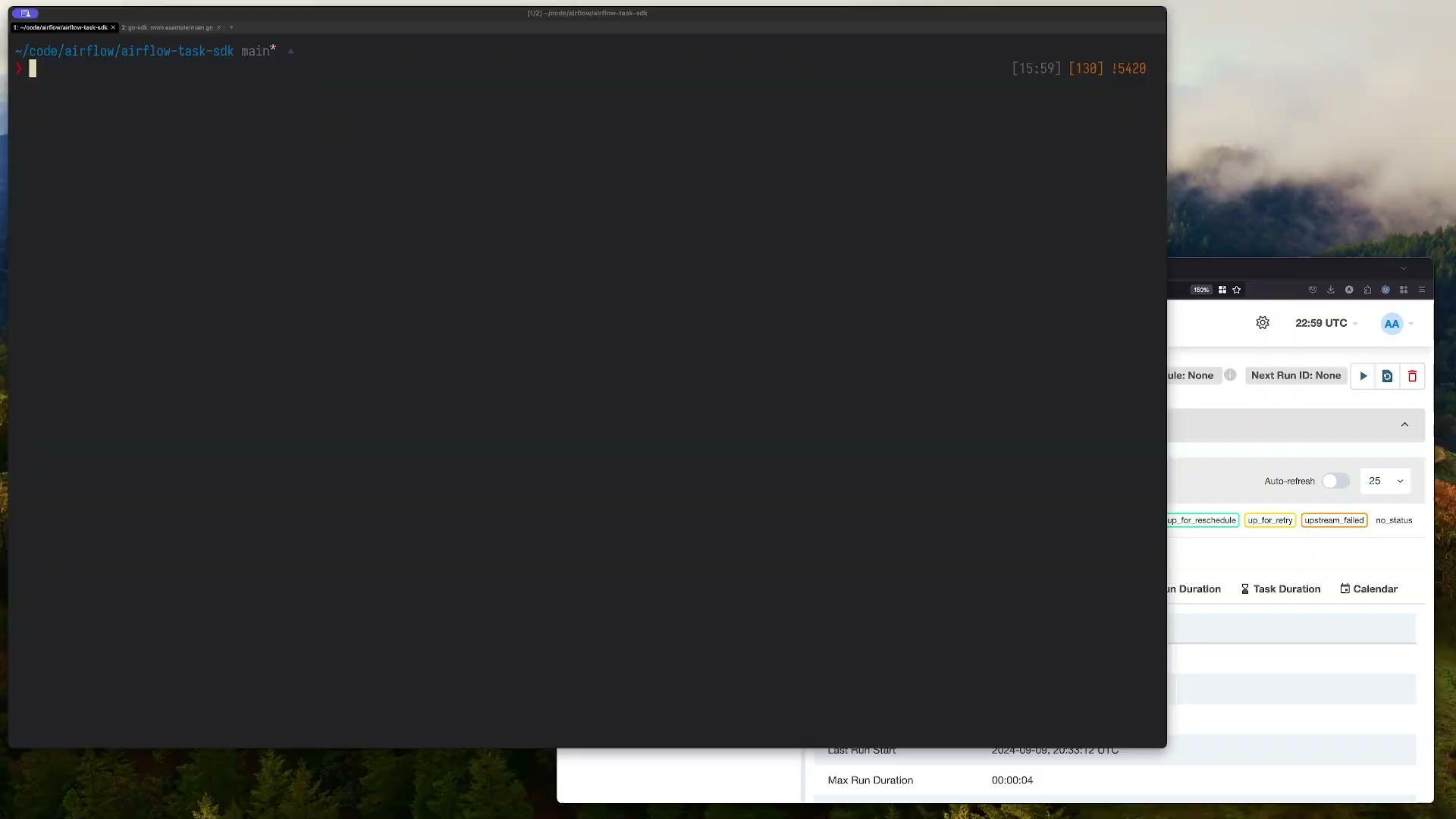
```go
import "github.com/apache/airflow/go-sdk/sdk"
func extract(ctx context.Context, log *slog.Logger) (map[string,any], error) {
   val, err := sdk.GetVariable(ctx, "my_variable")
   if err != nil { return err }
   order_data_dict, ok := val.(map[string]any)
   if !ok { return nil, fmt.Errorf("Invalid variable data type") }
   return order_data_dict, nil
}
type OrderSummary struct { Total float64 `json:"total_order_value"`}
func transform(ctx context.Context, log *slog.Logger, order_data map[string]float64) (OrderSummary, error) {
   total := 0
   for _, n := range maps.Values(order_data) { total += n }
   log.Infof("Summed %d orders", len(order_data))
   return OrderSummary{total}, nil
}
```

Running a Golang worker

```
import "github.com/apache/airflow/go-sdk/worker"

func registerTasks(worker worker.Worker) {
    worker.RegisterTask("tutorial_dag", extract)
}

func extract() error { … }

    // …
    worker := worker.New(logger)
    registerTasks(worker)
    worker.RunForever(context.TODO(), getServerUrl())
```

Golang Remote Worker




[1/2] ~/code/airflow/airflow-task-sdk
1: ~/code/airflow/airflow-task-sdk
2: go-sdk: nvim example/main.go
~/code/airflow/airflow-task-sdk main*
[15:59] [130] !5420
22:59 UTC
AA
ule: None
Next Run ID: None
Auto-refresh
25
up_for_reschedule
up_for_retry
upstream_failed
no_status
un Duration
Task Duration
Calendar
Last Run Start
2024-09-09, 20:33:12 UTC
Max Run Duration
00:00:04

In Summary

We need you!

Recruiting beta users:
- Deploying remote execution environments

Recruiting contributors for other languages:
- Add Providers for: Typescript, Scala, Kotlin, your language of choice!

Come speak at the next Airflow Summit about your use case on Airflow 3!