

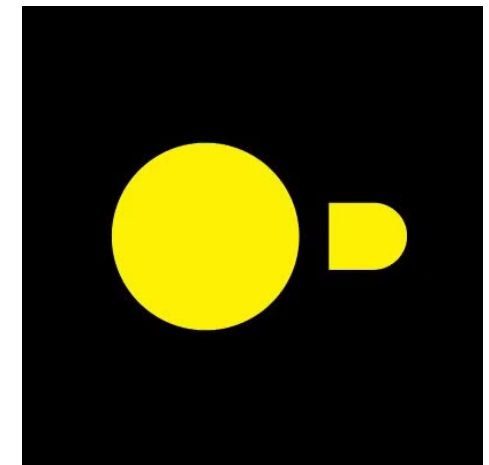
A quack at building Scalable Data Pipelines with DuckDB

Junaid Rahim

@junaidrahxm | junaid@atlan.com

Software Engineer, Atlan

atlan



A short Intro to **Argo Workflows**

The screenshot displays the Argo Workflows interface. At the top, the breadcrumb navigation shows 'Workflows / default / loops-m82mb-tk27h'. Below this is a toolbar with buttons for '+ RESUBMIT', 'DELETE', 'LOGS', 'SHARE', 'PREVIOUS RUNS', and 'OPEN WORKFLOW TEMPLATE'. A search bar is located below the toolbar. The main area shows a workflow graph with the following nodes:

- loops-m82mb-tk27h (Root node, successful)
- loops-m82mb-tk27h(0) (Loop node, successful)
- print-message(0:hello world) (Task node, successful)
- print-message(0:goodbye world) (Task node, successful)
- print-message(0:good morning) (Task node, successful)
- print-message(0:good night) (Task node, successful)
- print-message(0:hello world)(0) (Task node, successful)
- print-message(0:goodbye world)(0) (Task node, successful)
- print-message(0:good morning)(0) (Task node, successful)
- print-message(0:good night)(0) (Task node, successful)

The workflow graph shows a root node 'loops-m82mb-tk27h' which leads to a loop node 'loops-m82mb-tk27h(0)'. This loop node branches into four parallel task nodes: 'print-message(0:hello world)', 'print-message(0:goodbye world)', 'print-message(0:good morning)', and 'print-message(0:good night)'. Each of these task nodes has a corresponding looped task node below it, such as 'print-message(0:hello world)(0)'. All nodes in the graph are marked with a green checkmark, indicating they have completed successfully.

YAML based DSL to define K8s Native DAGs

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: loops-
spec:
  entrypoint: loop-example
  templates:
  - name: loop-example
    steps:
    - - name: print-message
      template: whalesay
      arguments:
        parameters:
        - name: message
          value: "{{item}}"
      withItems: ["hello world", "goodbye world", "good morning", "good night"]

  - name: whalesay
    inputs:
      parameters:
      - name: message
    container:
      image: docker/whalesay:latest
      command: [cowsay]
      args: ["{{inputs.parameters.message}}"]
```

Workflow Artifacts on S3

- Argo workflows can use major cloud object stores as artifactories
- Artifacts can be inputs to or outputs from workflow steps
- Artifacts on S3 are very versatile
 - Storing large datasets
 - Sharing data between workflow steps
 - Persisting results for later analysis
- DuckDB + Parquet + S3 is very ergonomic

```
- name: process
  outputs:
    artifacts:
      - name: orders_summary
        path: /tmp/orders_summary.parquet
        s3:
          key: "some/s3/prefix/orders_summary.parquet"
  container:
    image: junaidrahim/duckdb-with-argo:latest
    ...
```

atlan **Assets** All assets ▾

+ New ▾ ? 5 5 Andrew Ermogenous ▾

Snowflake ▾ Q Search all assets 🔍

dbt-food-beve... ▾ All Databases ▾ All Schemas ▾

All 2.3K Table 90 View 11 Column 2.2K Query 10 Process 21 Connection 1 Database 1

V_OSHA_INSPECTION ✓

View > DBT_FOOD_BEVERAGE > POSTGRES_RDS_OSHA_INS...

The dataset consists of inspection case detail for approximately 100,000 OSHA inspections conducted annually. The dataset includes information regarding the impetus for conducting the inspection, and details on citations and penalty assessments resulting from...

37 columns 1 query by 1 user

Preview

OSHA_INSPECTION ✓

Table > DBT_FOOD_BEVERAGE > POSTGRES_RDS_OSHA_INS...

The dataset consists of inspection case detail for approximately 100,000 OSHA inspections conducted annually. The dataset includes information regarding the impetus for conducting the inspection, and details on citations and penalty assessments resulting from...

1.9M rows 40 columns 2 queries by 1 user

PERMIT ✓

Table > DBT_FOOD_BEVERAGE > PUBLIC

Permit information detailing the date(s) and type(s) of work done on the property

~ rows 8 columns 0 queries

INSTACART_BEVERAGES_ORDER_CUSTOMER ✓

Table > DBT_FOOD_BEVERAGE > PUBLIC | instacart_beverages_order_c...

Dataset containing order level data for beverages departments of Instacart with customer detail

51K rows 15 columns 372 queries by 2 users

STG_SALESFORCE_ACCOUNT ✓

V_OSHA_INSPECTION ✓

View > DBT_FOOD_BEVERAGE > POSTGRES_RDS_OSHA_INSPECTI...

Open Query

Description

The dataset consists of inspection case detail for approximately 100,000 OSHA inspections conducted annually. The dataset includes information regarding the impetus for conducting the inspection, and details on citations and...

See more

Usage (last 30 days)

1 query by 1 user

Last queried

4 weeks ago

Definition

SQL

Columns

37

Terms

Perfect Order Rate

Number of Inspections

Owners

ravi

Classification

OSHA (2) Restricted

Certificate

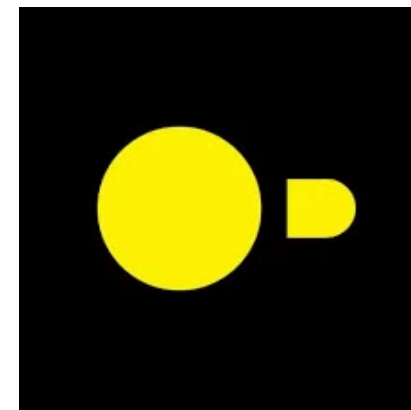
Verified

ravi 3 months ago

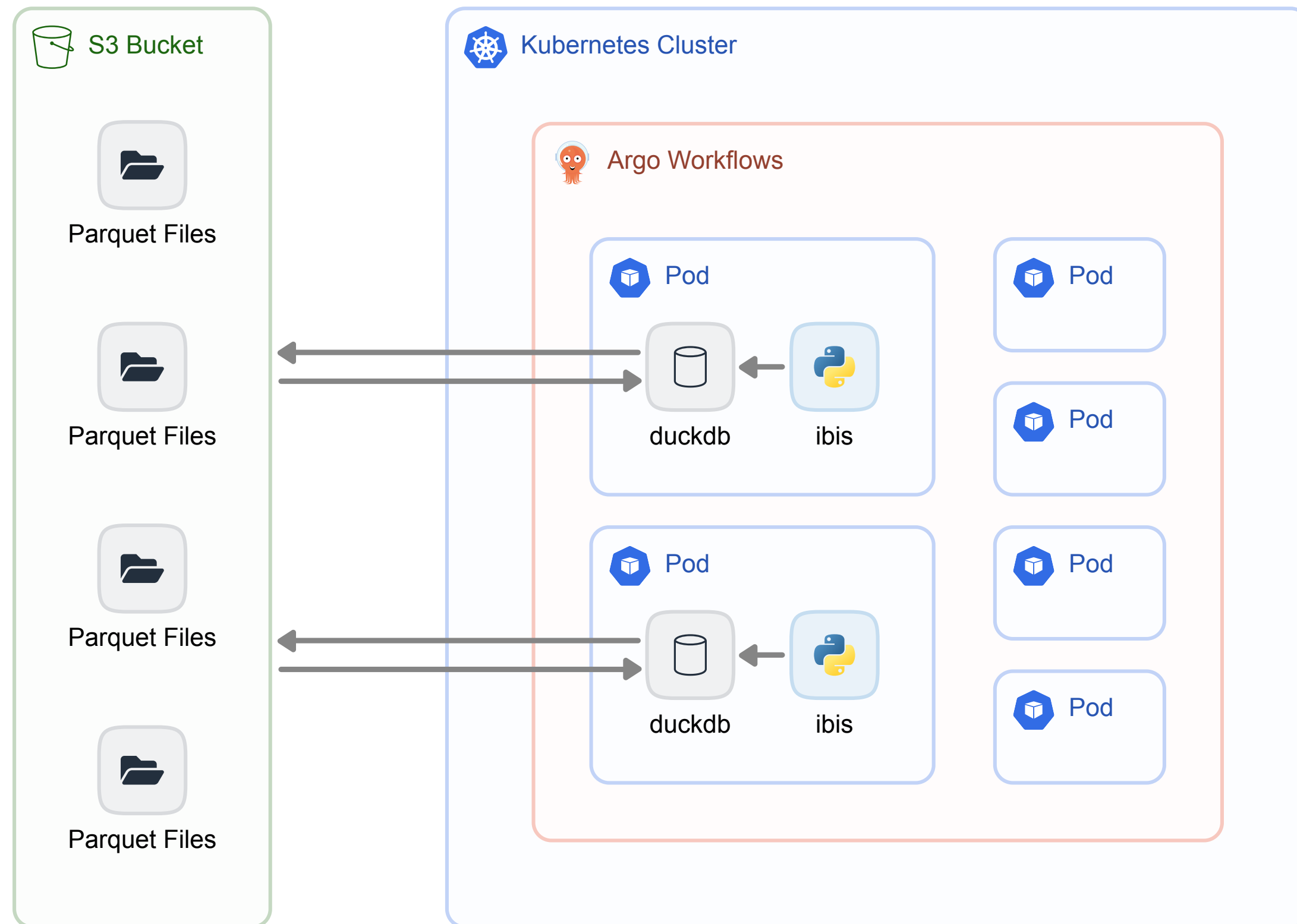
- Filters
- Certificate
- Owners
- Classifications
- Terms
- Properties
- dbt
- Usage
- Ownership
- DQ Table Summary
- Data Stewards
- Airflow
- Workflows
- Governance
- Admin
- Reporting
- Support

- Overview
- Columns
- Usage
- Lineage
- Activity
- Resources
- Queries
- Requests
- Properti...
- Slack
- Teams
- Jira
- Owners...

atlan



DuckDB Pipelines with Argo Workflows



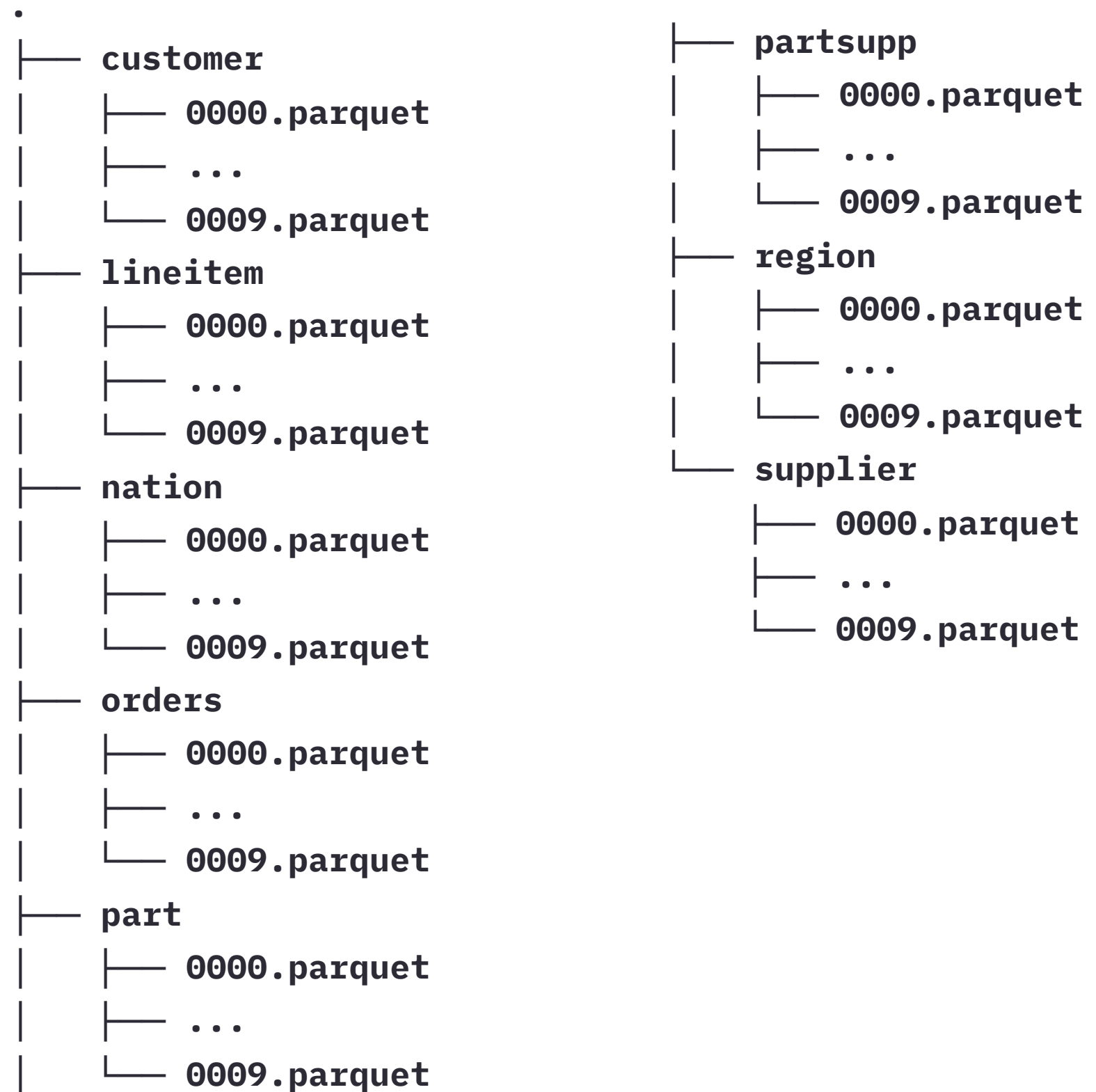
Example

Code available at github.com/junaidrahim/duckdb-with-argo-wf

TPC-DS

Sample Data

- Generated using [ibis-bench](#)
- 100 GB in memory
- 39 GB on disk



Workflow with Aggregate Queries

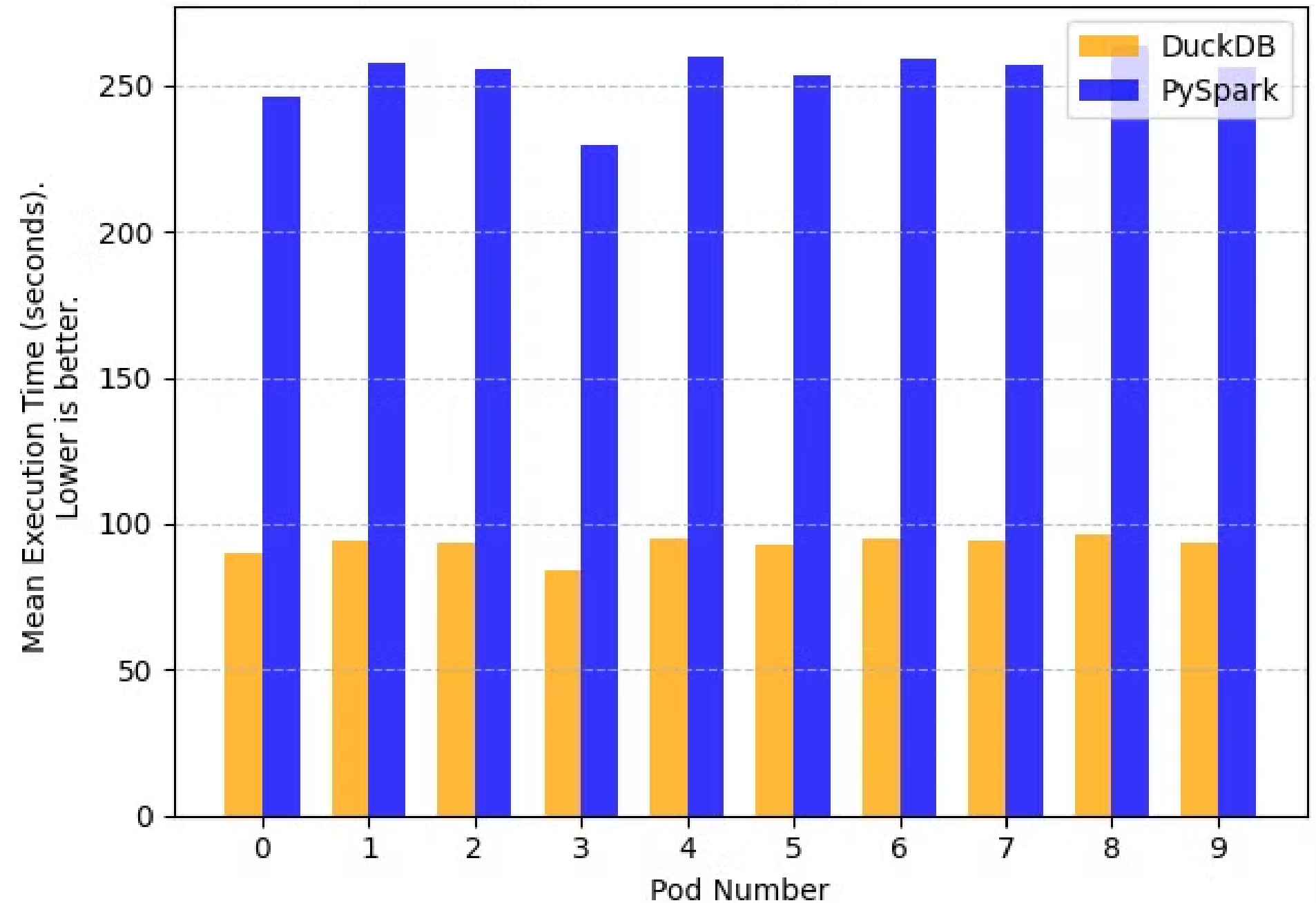
- A pod processing a chunk each
 - ~3.3 GB compressed parquet
- DuckDB
 - `memory_limit="2GB"`
 - `threads="2"`
- 2 queries to aggregate on **lineitem** and **orders** table



Performance

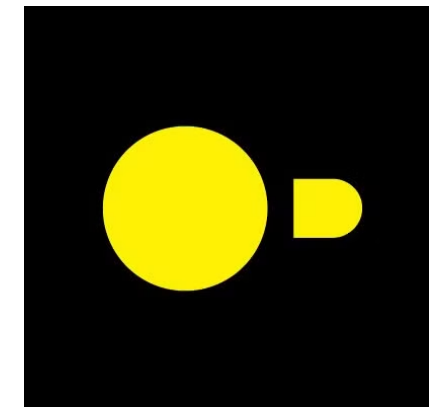
- DuckDB runs the aggregate queries **~2.3x** faster than PySpark at the pod level.
- Specs
 - Argo workflows on a single node EKS cluster
 - t3a.2xlarge (8 vCPU, 32GB mem)
- Execution time is mean over 5 runs
- github.com/junaidrahim/duckdb-with-argo-wf

Comparison of DuckDB and PySpark Execution Times



Horizontal Scaling, Dynamic Resources, Retries and much more

- Argo workflows are full of features you would expect from a battle-tested job orchestrator
 - Fan-in, Fan-out DAGs
 - Dynamic resource allocation using `podSpecPatch`
 - Robust retry mechanisms
 - Recursive DAGs
- Simpler further reporting and dashboarding
 - evidence.dev
 - observablehq.com
 - rilldata.com
 - motherduck.com
 - etc.



Thank You.



Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

[Create a presentation \(It's free\)](#)