How to reduce your dbt data pipeline costs with DuckDB

Table of contents

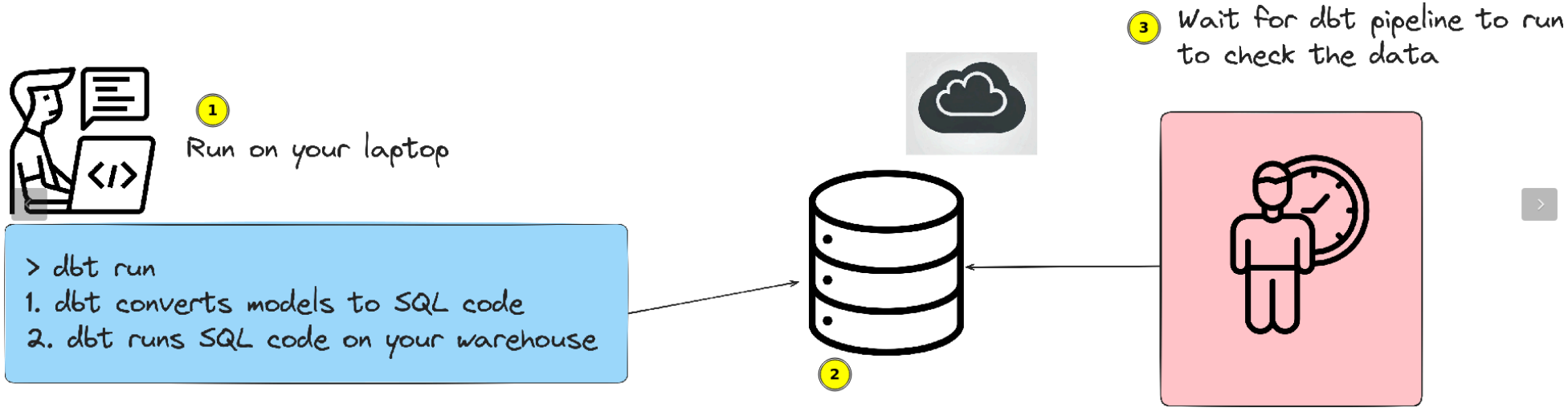
- DuckDB + Ephemeral VMs = dirt cheap data processing.
- Migrate one (or a few related pipelines) at a time.
- Watch out for integration points!

Introduction

- Name: Joseph Machado
- Work exp: 10 years in Data engineering & platform
- Blog: startdataengineering.com

DuckDB + Ephemeral VMs = dirt cheap data processing.

Standard dbt + warehouse workflow



- The connection settings are defined in the `profiles.yml` file.

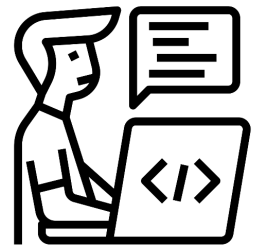
```
sde_dbt_tutorial:
  target: dev
  outputs:
    dev:
      type: postgres
      threads: 20
      host: localhost
      port: 5432
      user: dbt
      pass: password1234
      dbname: dbt
      schema: your_name_warehouse

    prod:
      type: postgres
      threads: 20
      host: localhost
      port: 5432
      user: dbt
      pass: password1234
      dbname: dbt
      schema: warehouse
```

Connection setting for dev env

Connection setting for prod env

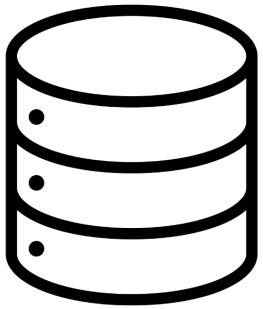
dbt + duckdb local workflow

Run on your laptop



> dbt run
1. dbt converts models to SQL code
2. dbt runs SQL code on your warehouse

DuckDB



Serverless data pipeline workflows

- Serverless dbt + duckdb workflow: Start VM -> Pull data into VM -> Process data -> Dump data into destination -> Spin down VM.

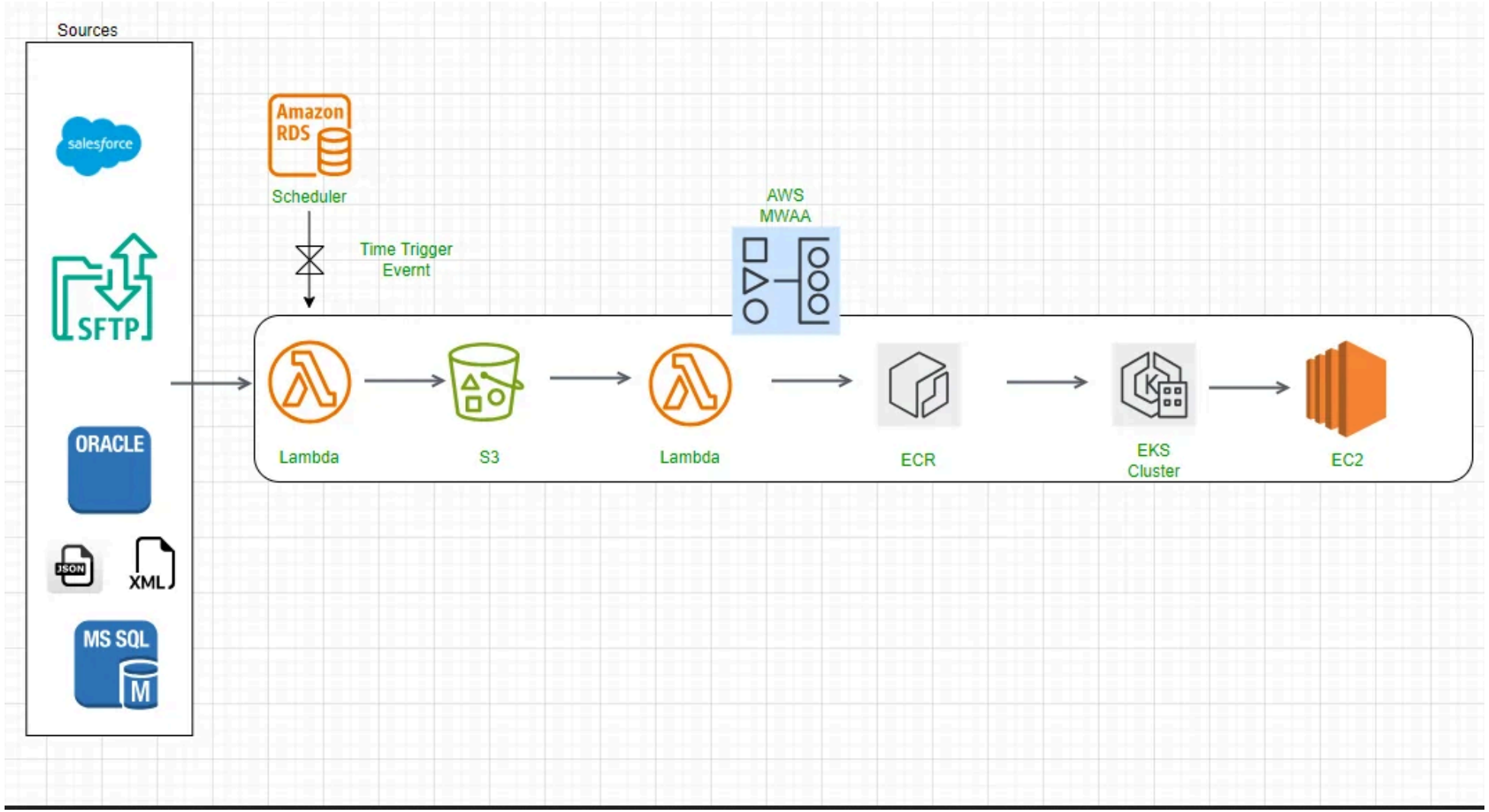
- **Note**: Pulling data can be done effectively with DuckDB extensions

Ephemeral servers are inexpensive

- Most cloud providers allow you to rent virtual machines charged by the hour (e.g., EC2).
- If you are not under tight time constraints, you can bid spot VMS at a much lower rate than on-demand VMS.
- Pulling data into VM (e.g. EC2) is cheap.
- Fast data transfer speed (especially with cpp optimized duckdb extensions).

Real project cost computation

- Real data infra (**ref**). Details:
  1. **Number of jobs**: 800 independent data pipelines with about 80 source systems
  2. **Number of files**: ~400
  3. **Average size per file**: ~1GB
- Objective: Reducing overall costs to between $3,000 and $5,000 per month, processing about 400GB per day across 800 jobs.



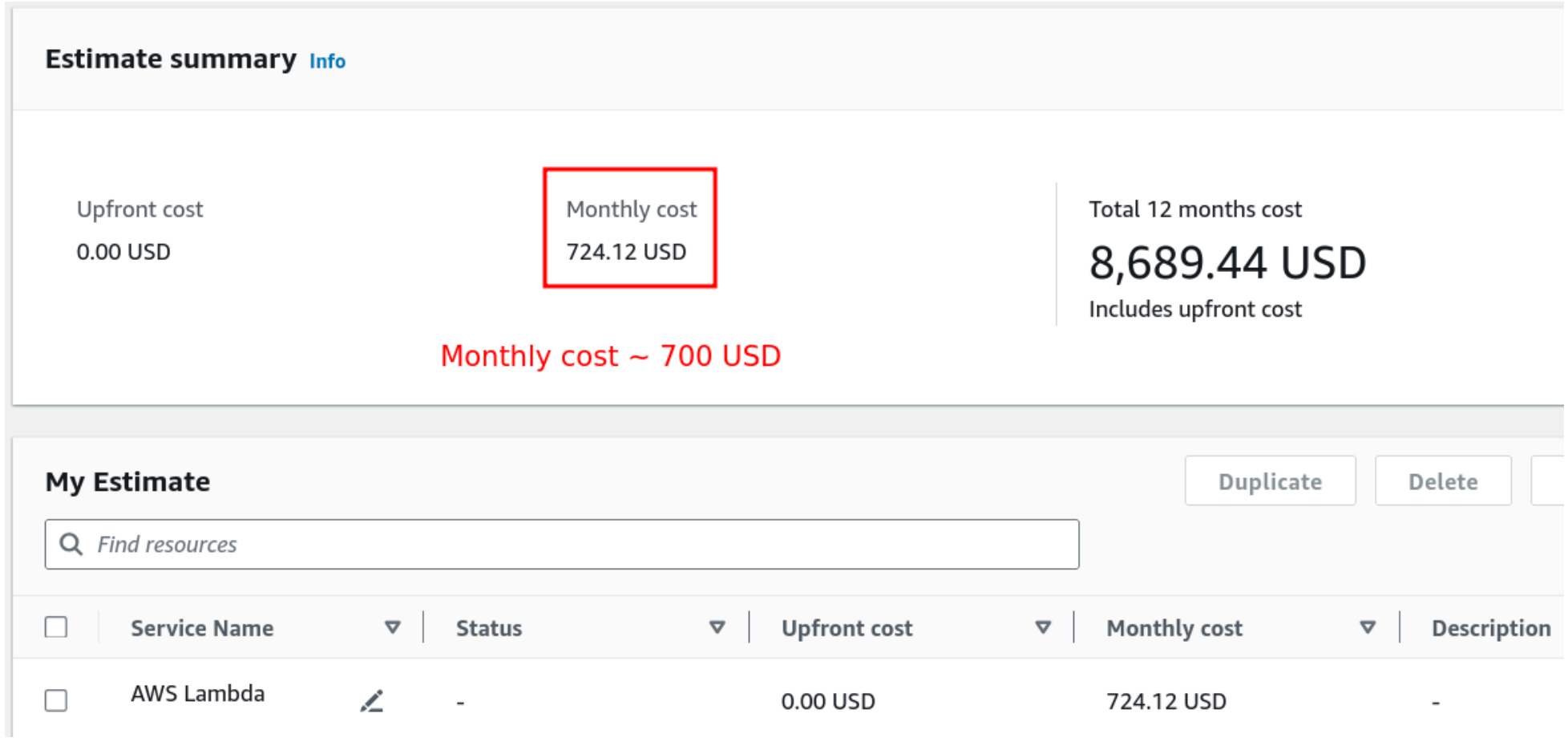
Architecture

Cost projection with serverless duckdb + dbt

- **Execution time (sample)**: A non-optimized, expensive pipeline (fact-fact join, joining all dimensions) takes about 2 minutes to run.

  1. Input: ~2GB
  2. Github codespace machine: RAM: 8GB & Cores: 2
  3. Output: ~13GB

- Serverless for 800 jobs ~ **700 USD**.
- Data transfer costs & IP costs are negligible at this scale.


Estimate summary Info
Upfront cost
0.00 USD
Monthly cost
724.12 USD
Total 12 months cost
8,689.44 USD
Includes upfront cost
Monthly cost ~ 700 USD
My Estimate
Duplicate
Delete
Find resources
Service Name
Status
Upfront cost
Monthly cost
Description
AWS Lambda
-
0.00 USD
724.12 USD
-

cost

Save time and money by keeping the feedback loop short

- With standard dist. Data proc systems, you will need to wait a while (set up cluster, start job, etc.) before you see the results.
- With dbt + duckdb, you can run locally without a complex setup and see results instantly (check out the buenavista package for viewing results when a dbt pipeline is running in duckdb).

Migrate one (or a few related pipelines) at a time.

Your first migration will involve some work

- Setting up infra to run, EC2/AWS Lambda/ECS, k8s, etc instead of connecting to a db/engine.
- **Code changes**: Changing db-specific functions, e.g., MERGE and date functions.
- **Input data**: You have to read the data from source systems or from warehouse where the input data exists.

- **Error handling & Debugging**: If you are processing in memory (without persisting intermediate datasets) and your data processing fails, you will have to re-run the entire pipeline.
- **Permissions** are set at the service level, not the USER/ROLE level, as in most data processing systems.
- **Logging system metrics**: most cloud VMs have this setup, e.g., AWS Cloudwatch. In addition to metrics logging you will not be able to see query history.

With a template to migrate, the rest of the migration will be more straightforward

- With the infra in place, the migration will be simple.
- Migrate at off-peak times. For example, if you are in e-commerce, don’t migrate during Thanksgiving, or if you are in finance, don’t migrate at the end of fiscal year reporting.

- Data pipeline migrations are tricky!
  1. **`Code`**: Ensure the code has unit/integration tests (not just DQ checks)
  2. **`Data`**: Ensure data has sufficient DQ checks and validate data between old and new systems for a defined period before switching over.

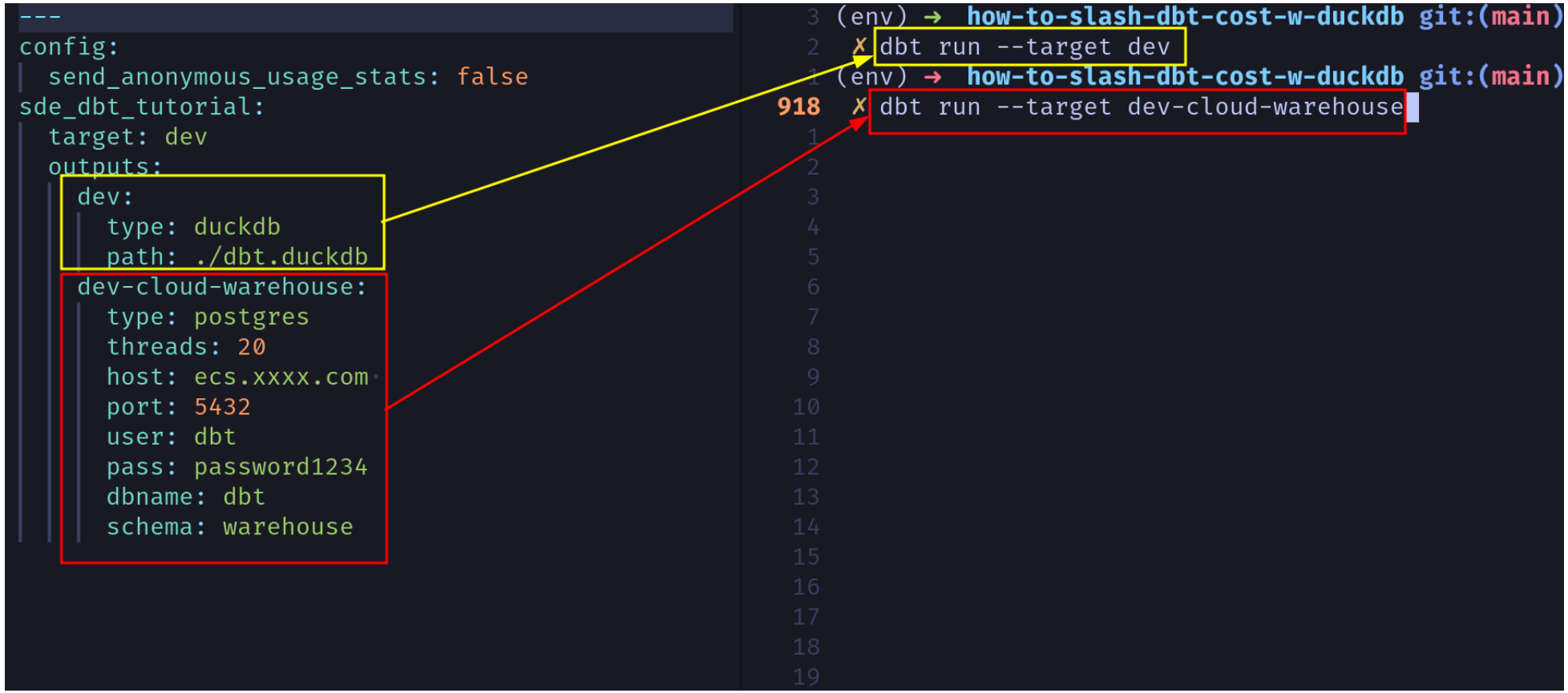
Watch out for pipelines that fully reprocess huge tables.

- If you have pipelines that involve aggregating historical data (e.g., for anomaly detection, handling late-arriving events), you need to handle them. Use one of the methods below:
- `Aggregate past n periods of data` instead of reaggregating the entire data set. For example, if your pipeline processes sales data that comes in every day, instead of reprocessing the past n years' worth of data, consider reprocessing the past 3/6 months of data (depending on later arriving data for your business use case).

- **`Store aggregated data`** in a separate location. For example, if you count rows in a dataset every run, store the counts in a separate table so you don’t have to recompute them for historical data each time.

Multi environment setup

- There may be cases where you may want to run some pipelines with duckdb and the rest with your existing warehouse.
- You can use dbt's `profile.yml` and dbt cli to indicate which pipeline should run where.
- Note that this introduces significant complexities with debugging & maintanance & establishing SOT!
- But you can alleviate a lot of issues if you are using a catalog (e.g. iceberg catalog)that most OLAP dbs support.


---
config:
send_anonymous_usage_stats: false
sde_dbt_tutorial:
target: dev
outputs:
dev:
type: duckdb
path: ./dbt.duckdb
dev-cloud-warehouse:
type: postgres
threads: 20
host: ecs.xxxx.com
port: 5432
user: dbt
pass: password1234
dbname: dbt
schema: warehouse
(env) → how-to-slash-dbt-cost-w-duckdb git:(main)
✗ dbt run --target dev
(env) → how-to-slash-dbt-cost-w-duckdb git:(main)
✗ dbt run --target dev-cloud-warehouse

Watch out for integration points!

Data permissions are defined at the service level.

- Most data processing systems have comprehensive data access controls crucial for data governance.
- With dbt + duckdb, we must handle data access at a service (E.g., AWS Lambda can access a specific S3 bucket, etc.) level.
- The inability to handle data permissions at a row level (like what Snowflake can offer) can sometimes be a deal breaker (PII, Sensitive info, etc).

Some popular tools don’t officially support DuckDB yet

1. elementary (GH issue)
2. greatexpectations

You need to do some work to dump data into Vendor warehouses

- With dbt, you usually create the output dataset as well.
- When using duckdb for processing, you must dump the data into a destination system (cloud store or another database).
- You will need a system to create the output tables/dump to the cloud store via extensions or code.

Excited About

1. Motherduck makes serverless infra for proc and analytics simple
2. Ibis dataframe with DuckDB backend
3. Tight Python integrations with Client API
4. dbt duckdb
5. Buena vista library for accessing duckdb tables when dbt is running

Questions