AIRFLOW SUMMIT

Scaling to 1,000 Dags:

Idelic's Blueprint for Airflow Automation and Reliability

Matt McCormack & Matt Stavinga

Who are we?

Matt Stavinga

- Senior Software Engineer
- 6 years at Idelic / Descartes

Matt McCormack

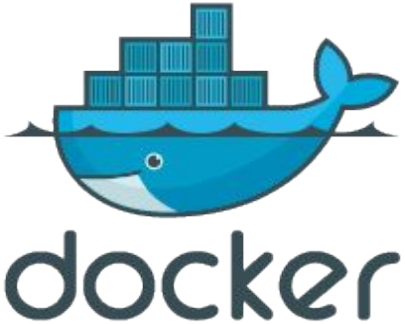
- Senior Data Engineer
- 6.5 years at Idelic / Descartes

What do we do?

- Safety Suite
  - Data management software geared towards trucking fleets
  - Data used in machine learning algorithms to identify risky behaviors / drivers in a fleet
- Extract various types of data
  - HR data
  - Telematics data
  - Accident data
- Various extraction sources
  - Spreadsheets (sent to us in aws s3 via sftp, externally hosted sftp, email)
  - APIs
  - Queue Based (external queues / aws sqs)
  - Screen scrape websites

Where we started

- Core Tasks – Python scripts
- Orchestration
  - PostgreSQL – Config, Scheduling, Logs
  - Celery: Broker – RabbitMQ, Backend – Redis
  - Monitor UI – Flower
  - Workers – Custom Docker Containers
- Problems
  - Observability
  - Development
  - Scalability
  - Deployment



How did we decide on Airflow ?

- Other tools that we considered
  - Cadence, Conductor, Luigi
- Why we chose Airflow
  - Scalability
  - Native python
  - Ease of deployment
  - Community Support
  - Observability

Initial Airflow Setup

- Each pipeline had its own DAG template file
  - Lots of custom files to manage
- Required a custom python DAG generation script to generate the DAGs in our airflow instance
  - ~10-15 minutes for all DAGs to appear in Airflow UI
- 3 Deployments – Production, Staging, Testing
  - Each had their own config
- Job config was contained in a JSON file on the repo
  - Required code changes

Initial Airflow Setup - Struggles

- Deploys required to create new DAGs
    - Generates 1 Dag per file
- Largely isolated tasks
- Custom code per DAG template
- Less modular code
- Config changes had change management (in repo) but required code deploys to update

Standardized Framework Design Theme

Standardization -> Abstraction -> Scalability

How did we scale up to 1000 DAGs ?
Standardized framework design

- Modernized DAG creation
    - Reads from Config API instead of static files
    - 1:1 file per job, per customer -> 1:1 file per job
- Increase task modularity
    - File Extraction
    - Transform from Spec
- Standardization of DAG tags
- Abstraction of task group creation
    - Task dependencies

How did we scale up to 1000 DAGs ?

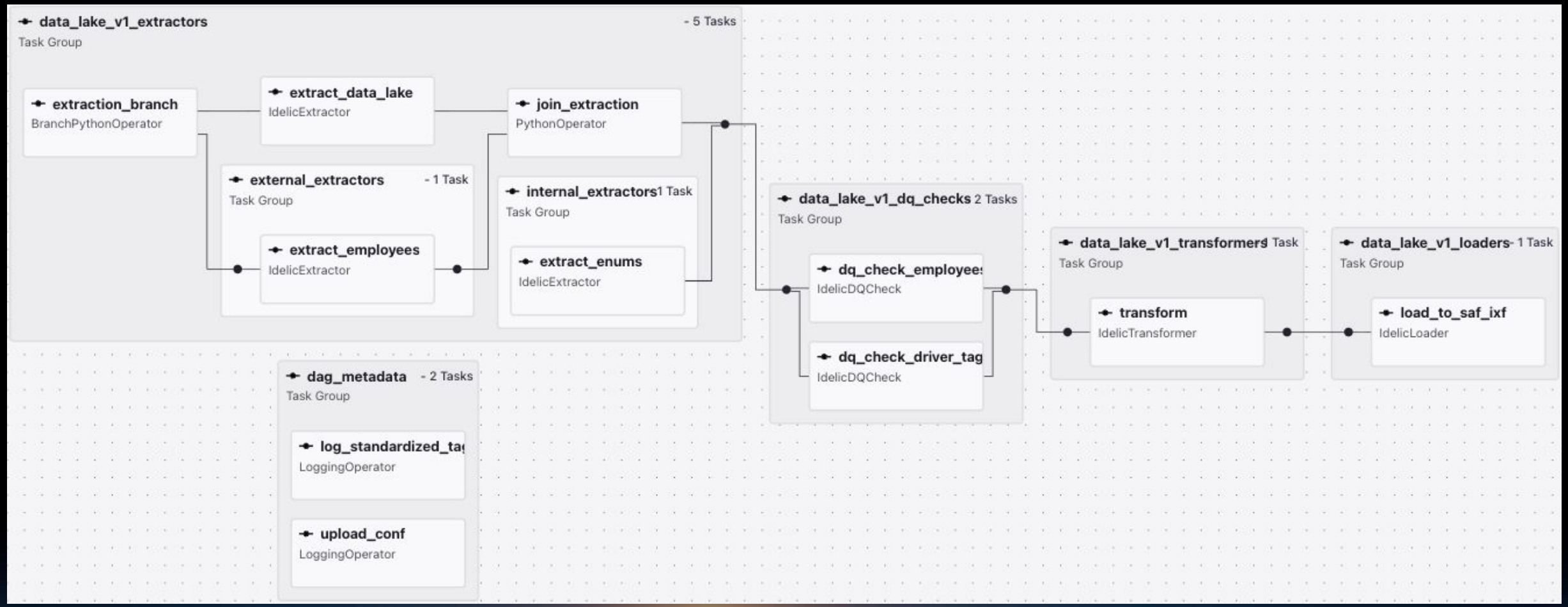
Standardized framework design

- Standardized Config Naming
- Data Quality Checks
  - Schema validation
    - Row level
    - Column Level
  - Defined Quality based on
    - Downstream Data requirements
    - Transformer Expectations
- Data Lake
  - AWS
    - S3 – parquet files

Side by side of Code -> Airflow

```
13  job = DataLakeJob(
14      'employee_test_integration',
15      EmployeeJobConfig,
16      [
17          pull_tag,
18          ModelTypeTag.EMPLOYEES,
19          ModelTypeTag.DOT_DOCUMENTS,
20          IntegrationMethodTag.REST_API
21      ]
22  )
23
24  with job:
25      create_extractors([extract_employees])
26      create_dynamic_schema()
27      create_transformers([transform])
28      create_loaders()
29      standard_flow()
30
31  job_aliases = job.get_job_aliases(enabled_key='employees.enabled')
32  job.create_dags(job_aliases)
```

Side by side of Code -> Airflow



How did we scale up to 1000 DAGs ?

CI / CD Automation

- Deploys are in Jenkins
- Automated testing airflow deployment creation
  - Jira connection / process
- Copy server deployments
  - Helped us scale up by streamlining testing
  - Allowed us to test changes safely with real customer data

How did we scale up to 1000 DAGs ?

Misc

- AI QA automation
  - First pass for QA
- Smoke Test Dag
  - Validate any framework / environment changes
- Metrics
  - Shared utilities to automatically push metrics to DataDog

Future Planned Work

- Migrating the rest of our DAG templates to the standardized DAG format
- Project “Quick Start”
    - Customer facing, self-service automation
    - Kicking DAG runs off from our UI
    - Human in the loop automation
        - Allow for Data validation prior to loading into our internal system
    - AI-Enablements
        - Chat bot style self-service onboarding
- Customers viewing DAG Run status / Run history
    - Customers can self QA their own data

Any Questions?

Scaling to 1,000 DAGs: Idelic's Blueprint for Airflow Automation and Reliability

Thank You!