Empowering Collaborative Data Workflows with Airflow and Cloud Services

Hoa Nguyen
Data Engineer
at New York Mets

Stanislaw Smyl
Data Engineer
at New York Mets

Data Engineering at the New York Mets

Data Engineering at NYM - Overview

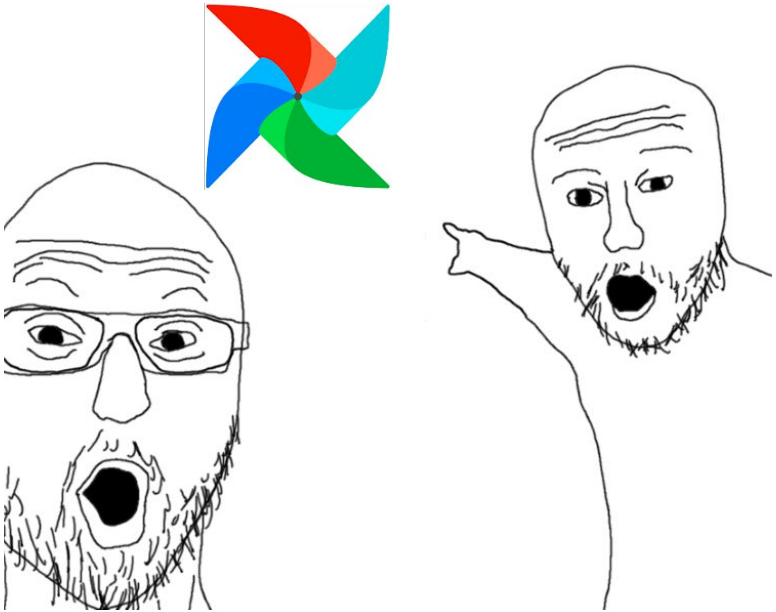
- From 0 to 4 Data Engineers in 2 years
- Google Cloud Platform
- Apache Airflow (Cloud Composer) at its core

Data Engineering at NYM - Baseball Data

- Biomechanical joint data
  - Position of every player
- > 45k games a year
- > 7 million pitches a year
- > 30k active players a year

Data Engineering at NYM - Apache Airflow

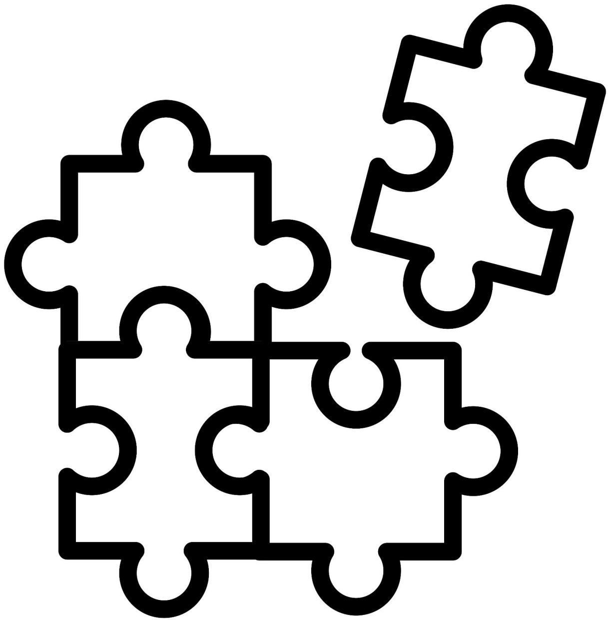
- At the core of Mets Data Platform
- ~ 150 DAGs
- Managed by Google as Cloud Composer
- Extensive integrations
  - 3rd party API
  - MLB



Data Workflows

Working with Others: Challenges

- Many processes are running locally without monitoring
- Code is developed individually: lacking code review and best practices
- Silos: Data management is decentralized, *some* people know *some* processes

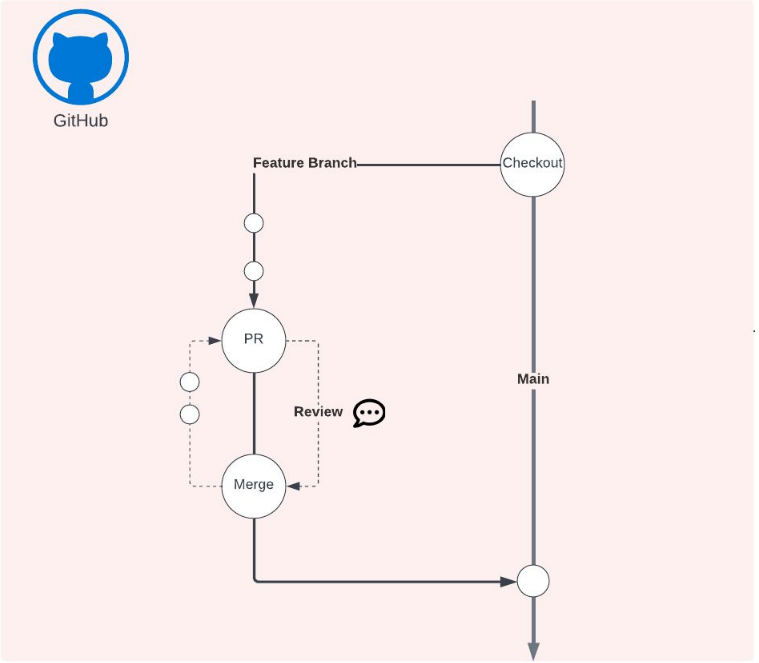


Solution: Data Workflows

A *simple* way of running *any* workflow on a schedule, which would let the DE team more easily manage the fun (Airflow) part, and allow all parties to collaborate

Code collaboration

- Perfect way to learn from each others
- Build reusable components and share with others
    - Python/R
- Each PR requires review from Data Engineer and Data Scientist before merge
- Flexible yet controlled deployments



Data Workflows: Airflow

Established monitoring practices

Loosely-coupled DAGs

Interdependent processes

Config-based dynamic DAGs

Fits all workflows

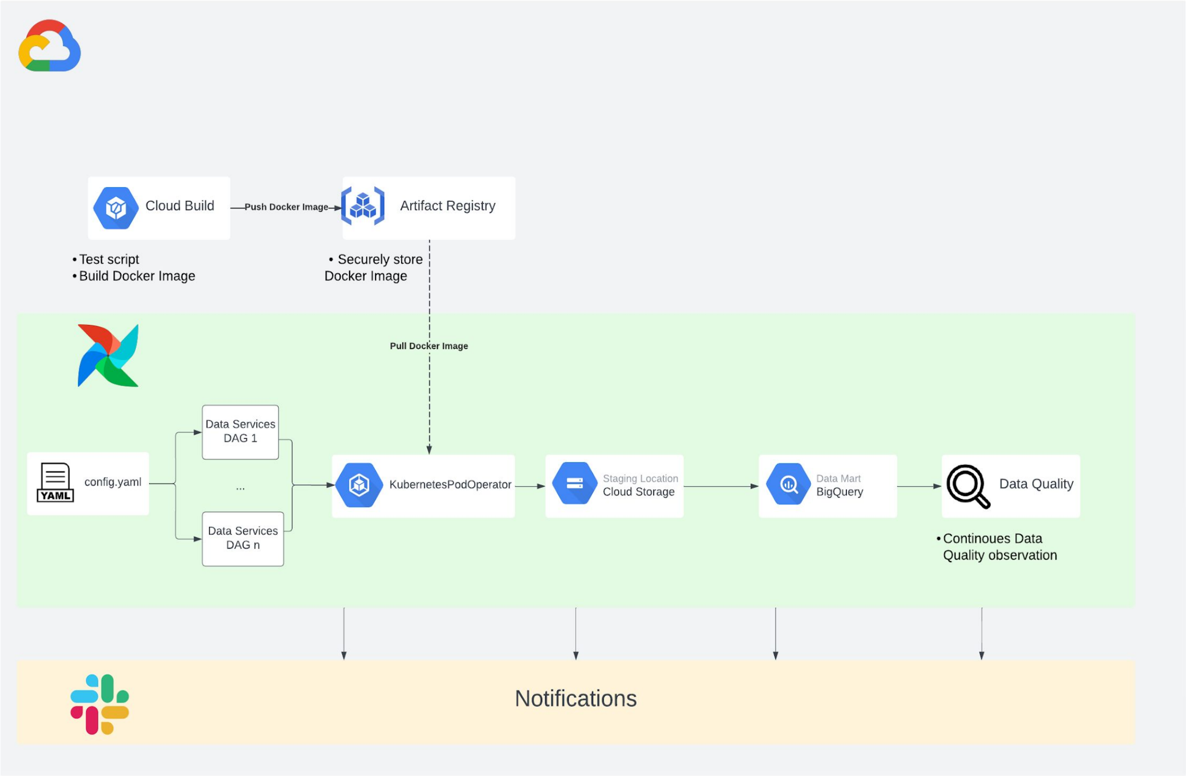
Kubernetes

Easy and fast scaling

Data Workflows: Process

- Script in any language submitted into Git along with templated documentation
    - Documentation is used for process discovery and sharing value with other departments
- Common shared libraries
    - Stress the importance of common interface
- Tests to be run pre-deployment and after scheduled processes finish
    - Data quality and observability at the core
- Monitoring
    - Shared Slack channel

Data Workflows: Process



Adding new Data Workflow

config.yaml

```yaml
image: "{region}-docker.pkg.dev/{project}/{name}:v1.1"
cmds:
  - "Rscript"
  - "/home/script_name.R"
staging_bucket: "bucket_name"
output:
  - type: "bq"
    location: "project_name.dataset_name.table_name"
    file: "output_file.csv"
doc_md: |
  DAG Documentation
schedule_interval: "@daily"
resources:
  limit_cpu: "2"
  limit_memory: "4G"
quality_checks:
  - collection_name: "{project_name}: Process Name"
dependents:
  - "dag1"
  - "dag2"
```

Parse

pydantic_model.py

```python
class Process(BaseModel):  # pylint: disable=R0903
    """
    Building blocks for a single process running within Data Workflows
    """

    cmds: List[str] = Field(..., description="Command to start the process in Docker image")
    staging_bucket: str = Field("bucket-name", description="Bucket where outputs are staged")
    output: List[Union[BQOutput, GCSOutput]] = Field(
        [],
        description="Where to output the files produced by this service. Currently BQ or GCS are
supported",
    )
    params: Dict[str, Dict[str, Any]] = Field({}, description="Parameters to DAG")
    doc_md: str = Field("", description="DAG's documentation to be displayed in Airflow UI")
    schedule_interval: Optional[str] = Field(None, description="Schedule Interval for Airflow DAG")
    resources: Resources = Field(..., description="Memory and CPU limits for running pod")
    models_to_materialize: List[str] = Field(
        [], description="dbt models to materialize after the process is run"
    )
    quality_checs: List[Check] = Field(
        [], description="Data Quality checks to be run after the process is run"
    )
    dependents: List[Dependent] = Field(
        [], description="List of dependents of this process"
    )
```

dag.py

```python
globals()[dag_id] = create_dag(**params)
```

Questions?

Thank you!

ssmyl@nymets.com
hnguyen@nymets.com