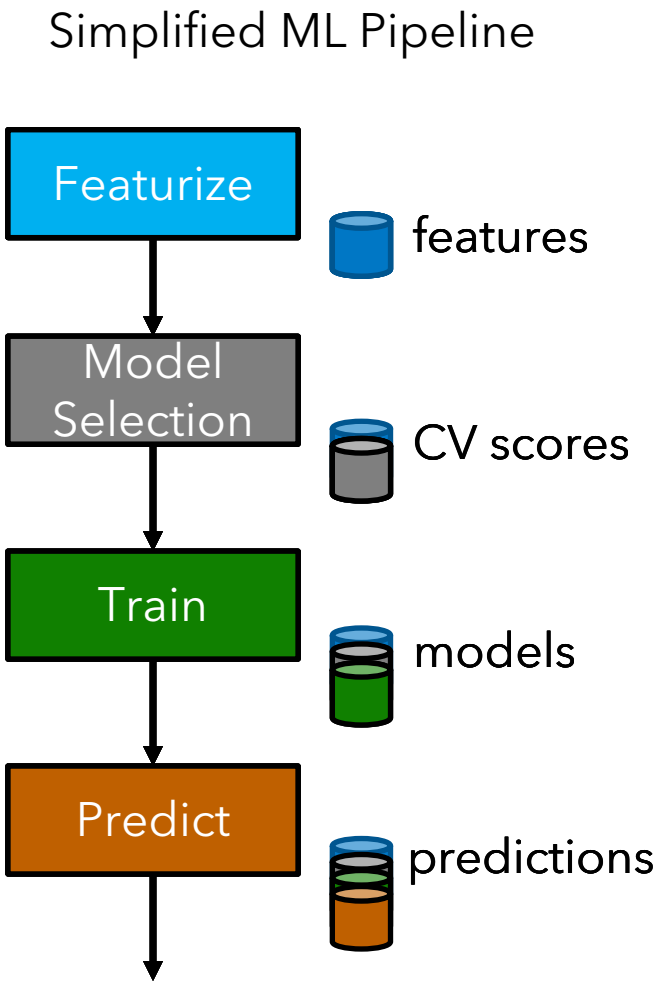
Disdat: Bundle Data Management for ML Pipelines

May 20, 2019

Ken Yocum, Sean Rowan, Jonathan Lunt – Intuit, Inc.

& Theodore Wong – 23andMe

ML Pipelines in Practice



Scenario:

- Team needs to create a new ML model
- Group of data scientists and engineers
- Many laptops, small and full dataset
- Agree on general pipeline

Need access to data artifacts to:

- Explore input / output data
- Tune by updating features / retrain on specific data
- Debug / reproduce errors
- Deploy models and share predictions

Data management challenges

Naming

- Logically identify an output, i.e. fin_model

Versioning

- Allow humans/pipelines to find “latest” output

Sharing

- Mechanism for moving / re-using data

Instead we get bespoke projects with:

- Ad-hoc naming convolved with versioning
  - `fin_model` becomes `fin_model_v1_20190520`
- Data scatter from sharing via email, messaging, AirDrop, Box
  - Across applications, local FS, DFS, etc.
- Reproduction instead of re-use

The VCS option

Git

- Popular, familiar, powerful + share via repos and version history.
- But not all VCS constructs may apply to data versioning
  - Humans often ”checkout” individual files, but pipelines?
  - Diff / Merge binary or text-based data, merge conflicts on model updates?
  - No metadata, e.g., lineage.

Goal: Enable naming, versioning, sharing with:

- Minimal prescription: many file types and processing systems
- Separate naming and versioning
- Simple semantics: Ex. notebook code for `get_latest_data(“fin_model”)`

Disdat

Two practical data abstractions

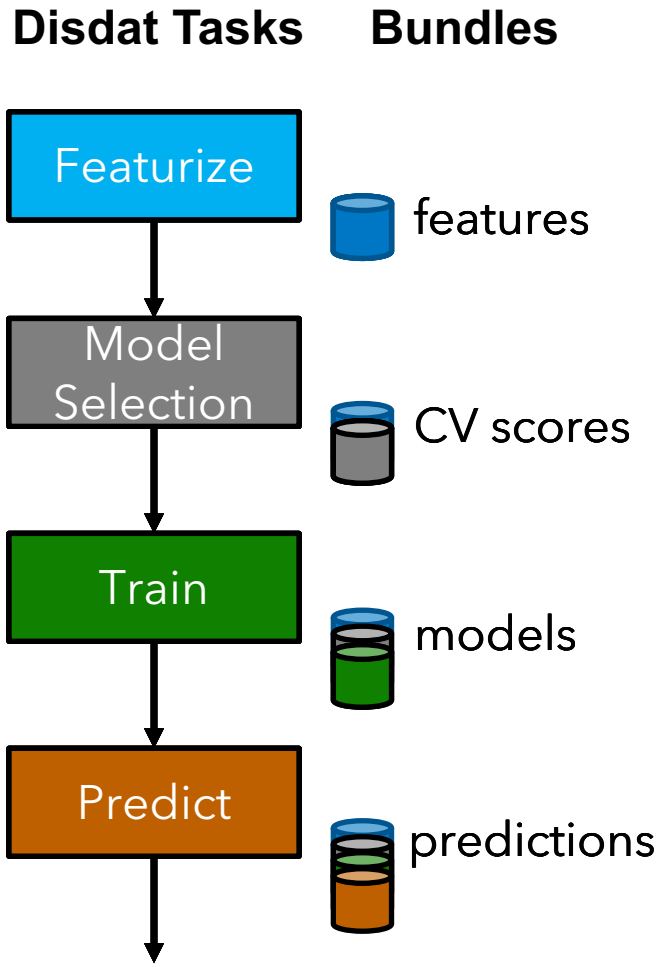
- Data bundle – Named collection of files and literals
- Data context – Named repository of bundles

Disdat API

- API for creating and sharing bundles in contexts
- Use from notebook or command line

Disdat pipelines

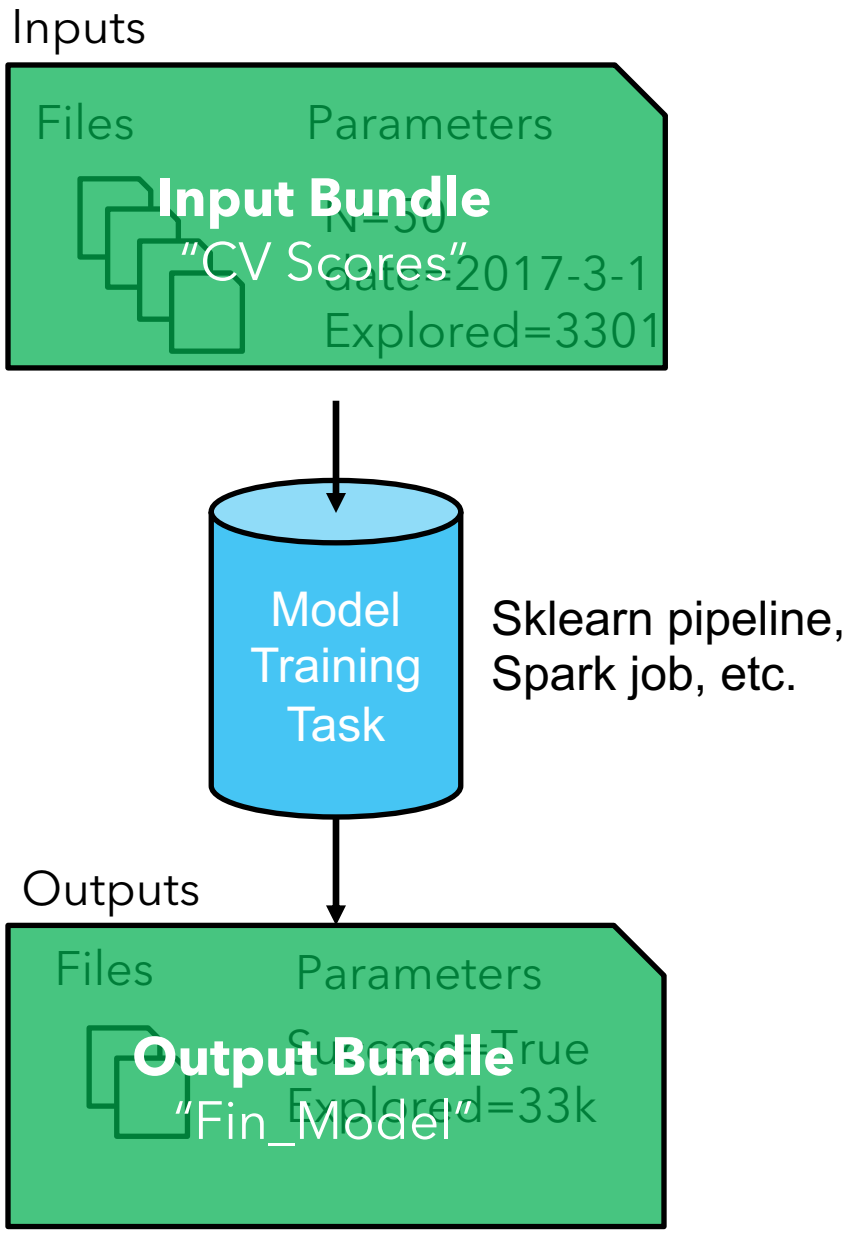
- Instrumented existing system -- Spotify's Luigi
- Consumes, produces, and publishes Bundles



Bundles: describing data sets

Bundle: an immutable collection of

- Named, typed arrays:
  - Links to local files or on cloud systems (S3)
  - Scalars (int, float, bool, etc.)
  - Pointers to other bundles
- Lineage and user-defined tags

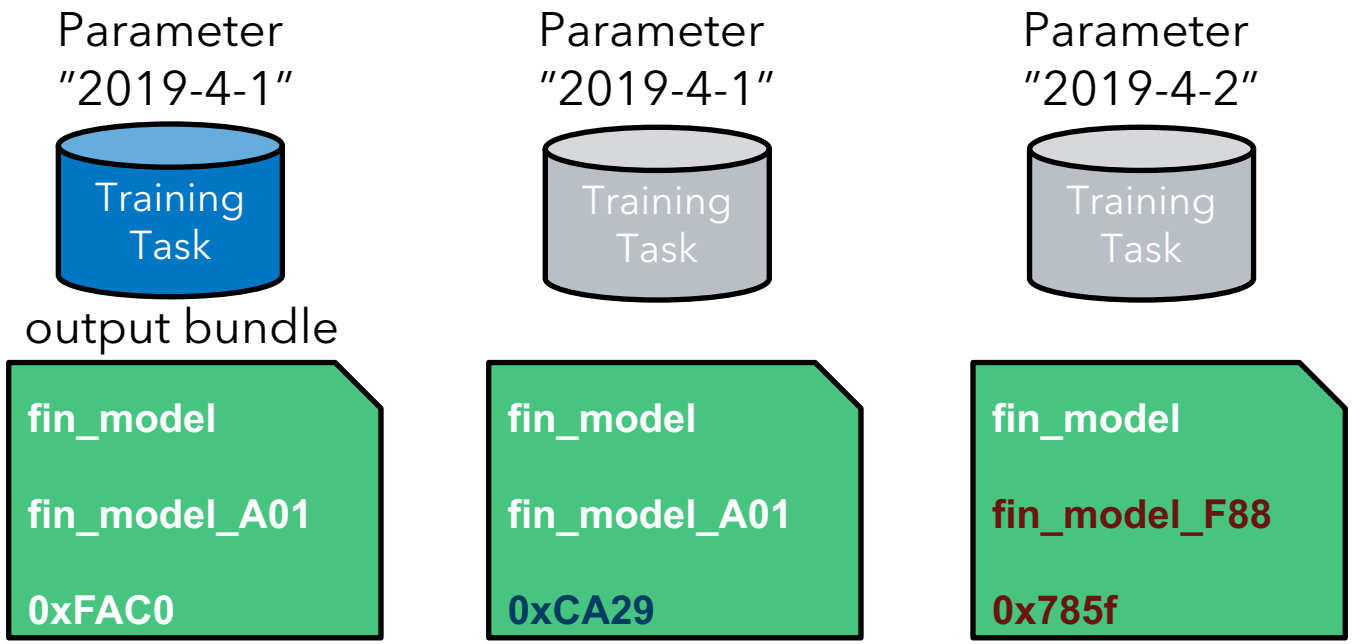


Bundle versioning

Determine how to name and find the “latest” bundle

3 Names

- Human name
- Processing name
- UUID



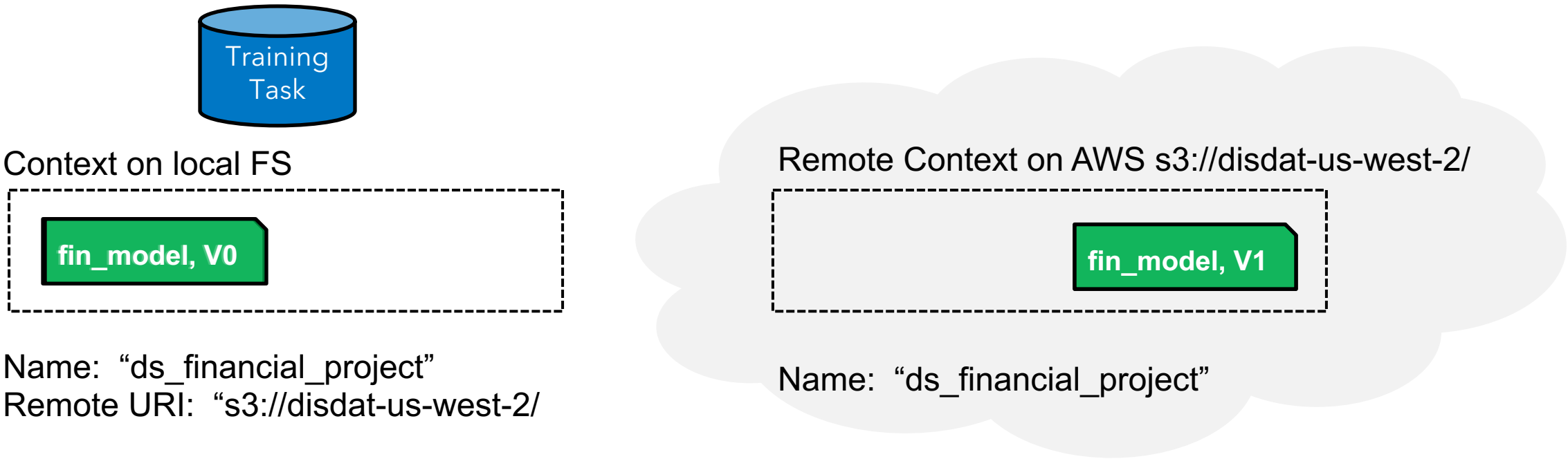
Human Query: `disdat.api.get(human_name=”fin_model”)` – **0x785f**

Pipeline Query: `disdat.api.get(proc_name=”fin_model_A01”)` – **0xCA29**

Sharing via data contexts

Data context: A collection of bundles

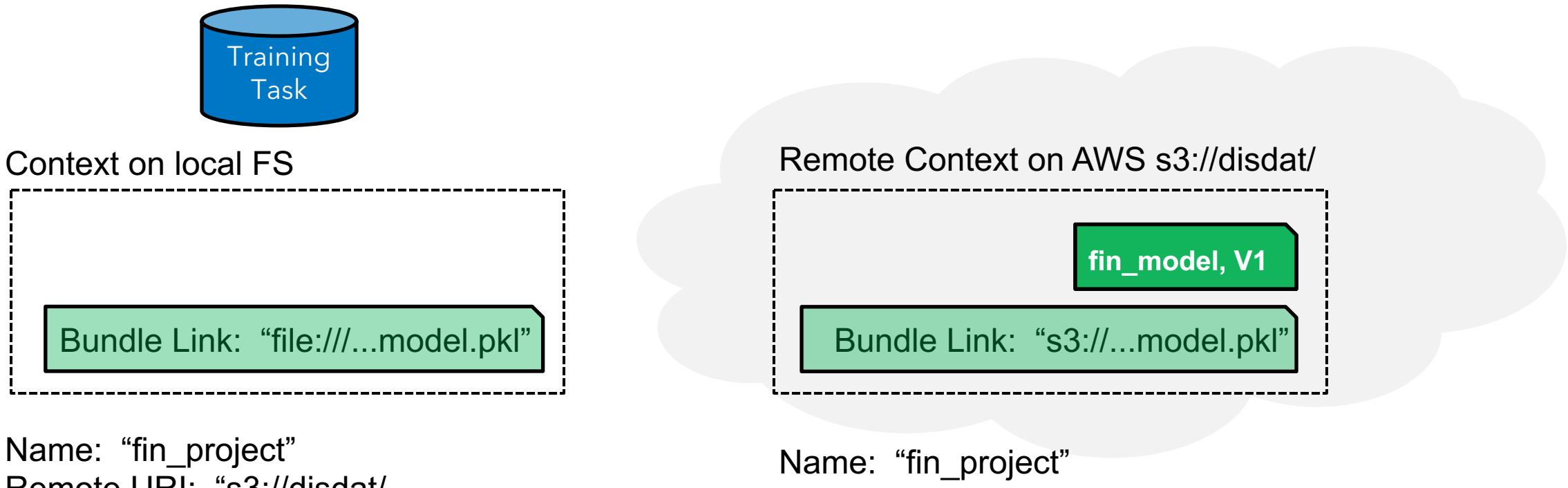
- Exists as a directory on local or cloud FS
- “push” or “pull” to/from contexts



Optimizing data transfers

Localizing bundles

- Pull meta data, but not linked files.
- On “localize” pull linked files



Scenario: ML Pipeline

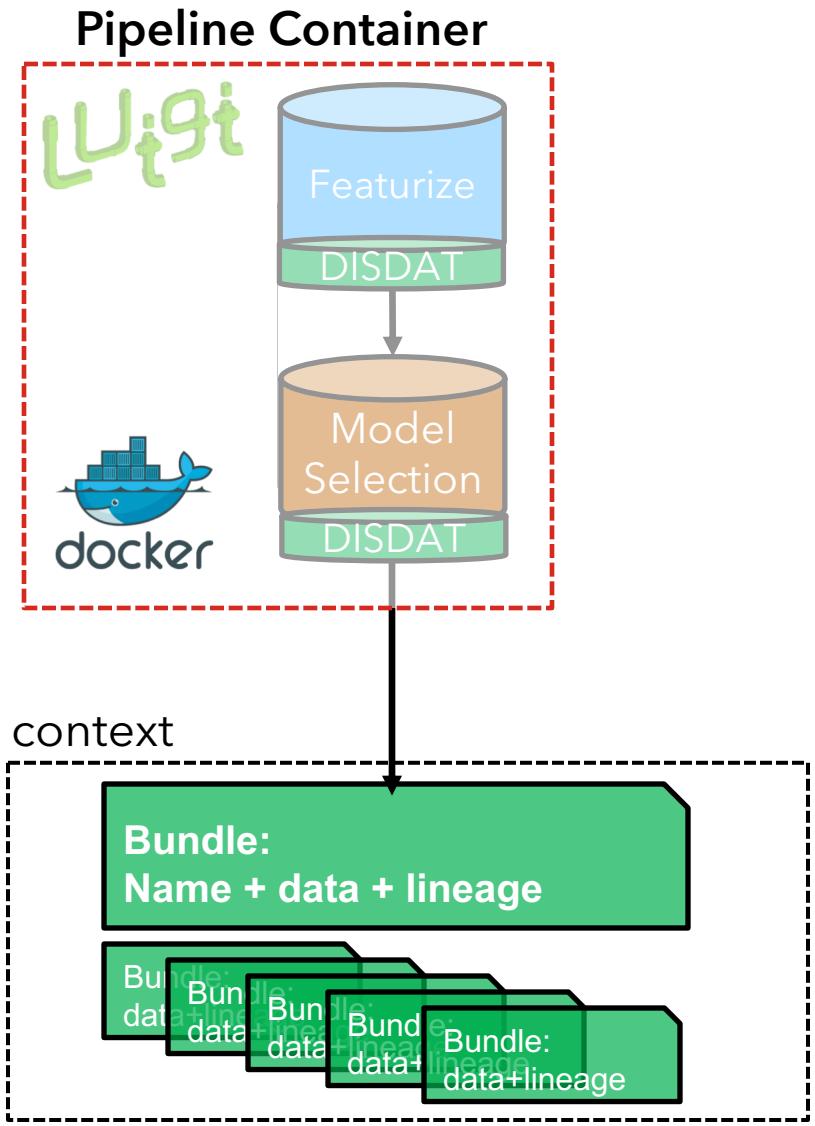
1.) Users wrote Disdat / Luigi tasks

2.) Each produces a *bundle:* name + files + lineage

3.) Bundles shared via *contexts*

But model selection is expensive

4.) Dockerize the pipeline and run on cloud.



Pipeline cloud execution

Pipeline as DAG of Disdat tasks

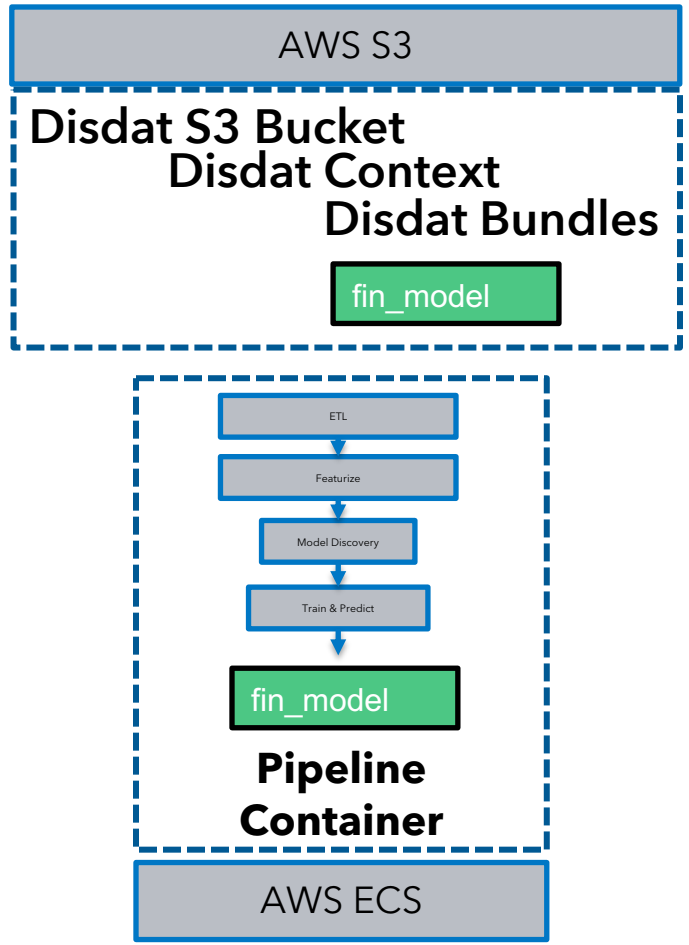
```
# Create container and push to AWS
api.dockerize('./setup.py',push=True)


# Run on AWS Batch
api.run(module.class, args, backend=AWSBatch)
```

Notebook

```
# Pull most recent bundle
api.pull('fin_project', 'fin_model', localize=True)

# Inspect bundle
bundle = api.get('fin_project', 'fin_model')
print('bundle data:', bundle.data)
```



Conclusion

Related work

Many examples: Palintir Foundry, FBLearner, Uber Michelangelo, Pachyderm, Datmo, DVC, MLFlow, PyML

- Closed and/or monolithic ecosystems
- Open-source projects / APIs / “collection-oriented” file types
- But leave naming and data management to user

Disdat introduces two practical abstractions to help

- Bundles the unit for versioning, lineage
- Contexts for managing and sharing bundles

Status

- https://github.com/kyocum/disdat

Final thanks to Intuit, Inc, and Human Longevity, Inc. (Jason Knight, Amalio Telenti)