Notebooks as Functions with papermill.

Using Nteract Libraries
@github.com/nteract/

NETFLIX

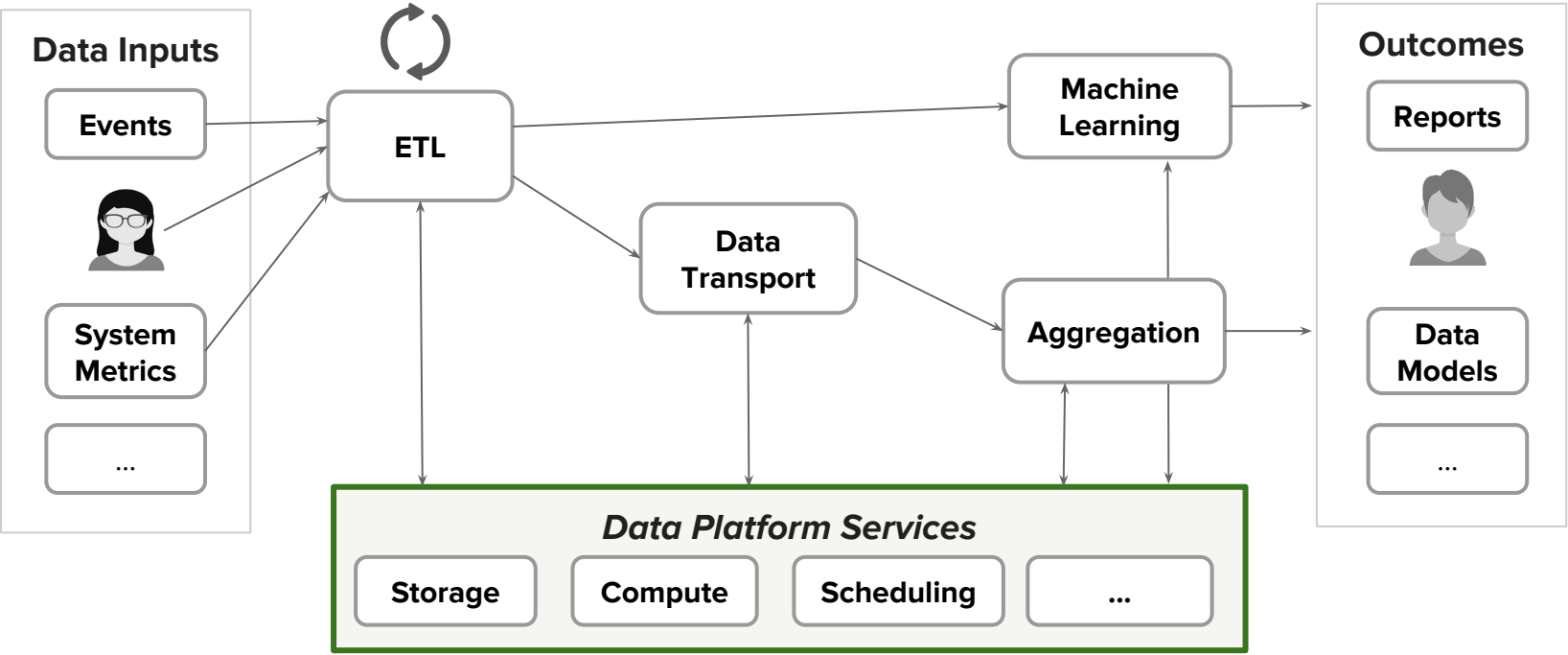
Speaker Details

Matthew Seal

Backend Engineer on the Big Data Platform Orchestration Team @ Netflix

NETFLIX

What does a data platform team do?




Data
Platform
Opens
Doors
...
not this
one

Open Source Projects

jupyter

nteract

Contributed to by

Jupyter Notebooks

What Are They?

NETFLIX

jupyter NB Basics Last Checkpoint: 3 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code git nbdiff

In [1]:
```
from IPython.display import Image
```

In [2]:
```
Image(filename='papermill.png')
```

Out[2]:

In [3]:
```
paper = 'paper'
mill = 'mill'
paper + mill
```

Out[3]: 'papermill'

NETFLIX

Notebooks.

A rendered REPL combining

- **Code**
- **Logs**
- **Documentation**
- **Execution Results.**

Useful for

- **Iterative Development**
- **Sharing Results**
- **Integrating Various API Calls**

nteract Useful Code.ipynb less than a minute

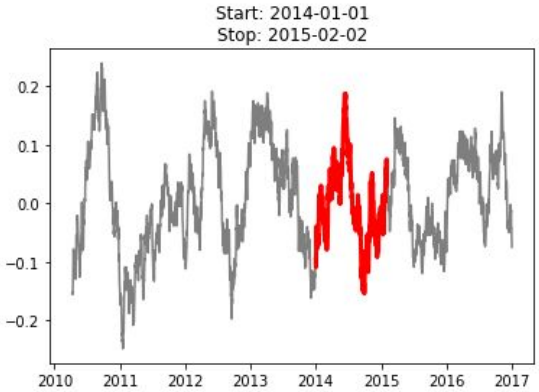
File Edit View Cell Runtime Help

Model Rendering

Some easy to read markdown to explain how some results and graphs were generated

```
[ ]  with open('~/models/recent.json', 'r') as f:
       model = json.load(f)
     # Confirm our model is valid
     assert 'version' in model
```

```
[13]  display({"image/png": model['results']['png']}, {}, raw=True)
```



A Breakdown

nteract Useful Code.ipynb

less than a minute

File Edit View Cell Runtime Help

Status / Save Indicator

Model Rendering

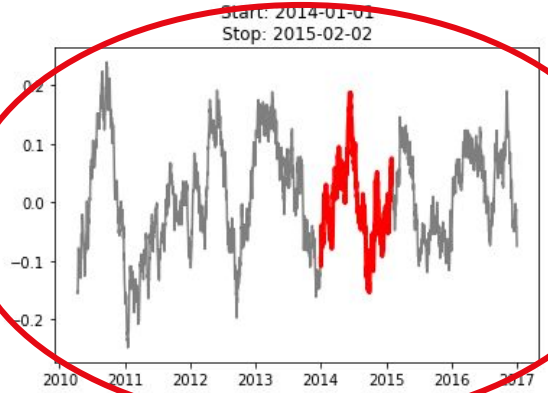
Some easy to read markdown to explain how some results and graphs were generated

Code Cell

```
[ ]  with open('~/models/recent.json', 'r') as f:
       model = json.load(f)
     # Confirm our model is valid
     assert 'version' in model
```

```
[13]  display({"image/png": model['results']['png']}, {}, raw=True)
```

Displayed Output



Wins.

- Shareable
- Easy to Read
- Documentation with Code
- Outputs as Reports
- Familiar Interface
- Multi-Language

nteract Useful Code.ipynb less than a minute

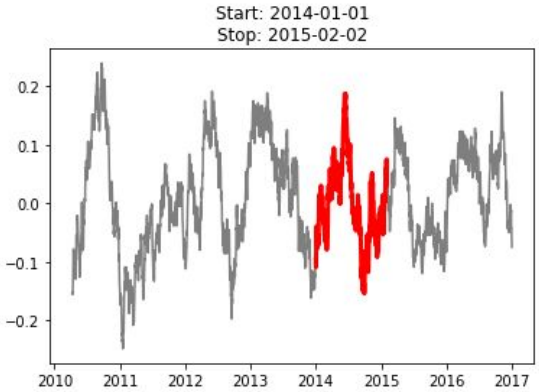
File Edit View Cell Runtime Help

Model Rendering

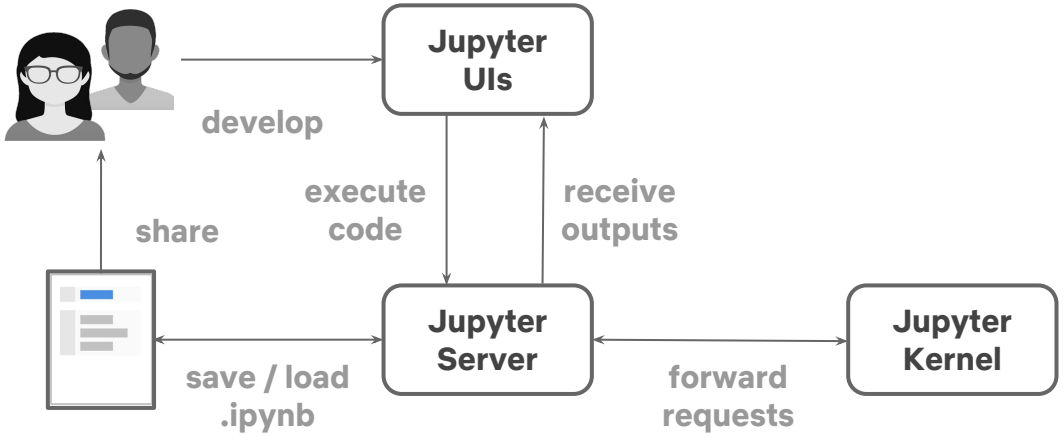
Some easy to read markdown to explain how some results and graphs were generated

```
[ ]  with open('~/models/recent.json', 'r') as f:
       model = json.load(f)
     # Confirm our model is valid
     assert 'version' in model
```

```
[13]  display({"image/png": model['results']['png']}, {}, raw=True)
```



Notebooks: A Repl Protocol + UIs



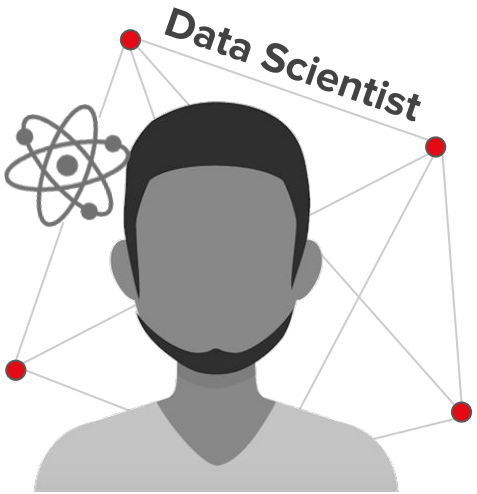
It's more complex than this in reality

Traditional Use Cases

Explore

NETFLIX

Exploring and Prototyping.



The Good.

Notebooks have several attractive attributes that lend themselves to particular development stories:

- **Quick iteration cycles**
- **Expensive queries once**
- **Recorded outputs**
- **Easy to modify**

The Bad.

But they have drawbacks, some of which kept Notebooks from being used in wider development stories:

- Lack of history
- Difficult to test
- Mutable document
- Hard to parameterize
- Live collaboration

Filling the Gaps

papermill

NETFLIX

Focus points to extend uses.

Things to preserve:

- Results linked to code
- Good visuals
- Easy to share

Things to improve:

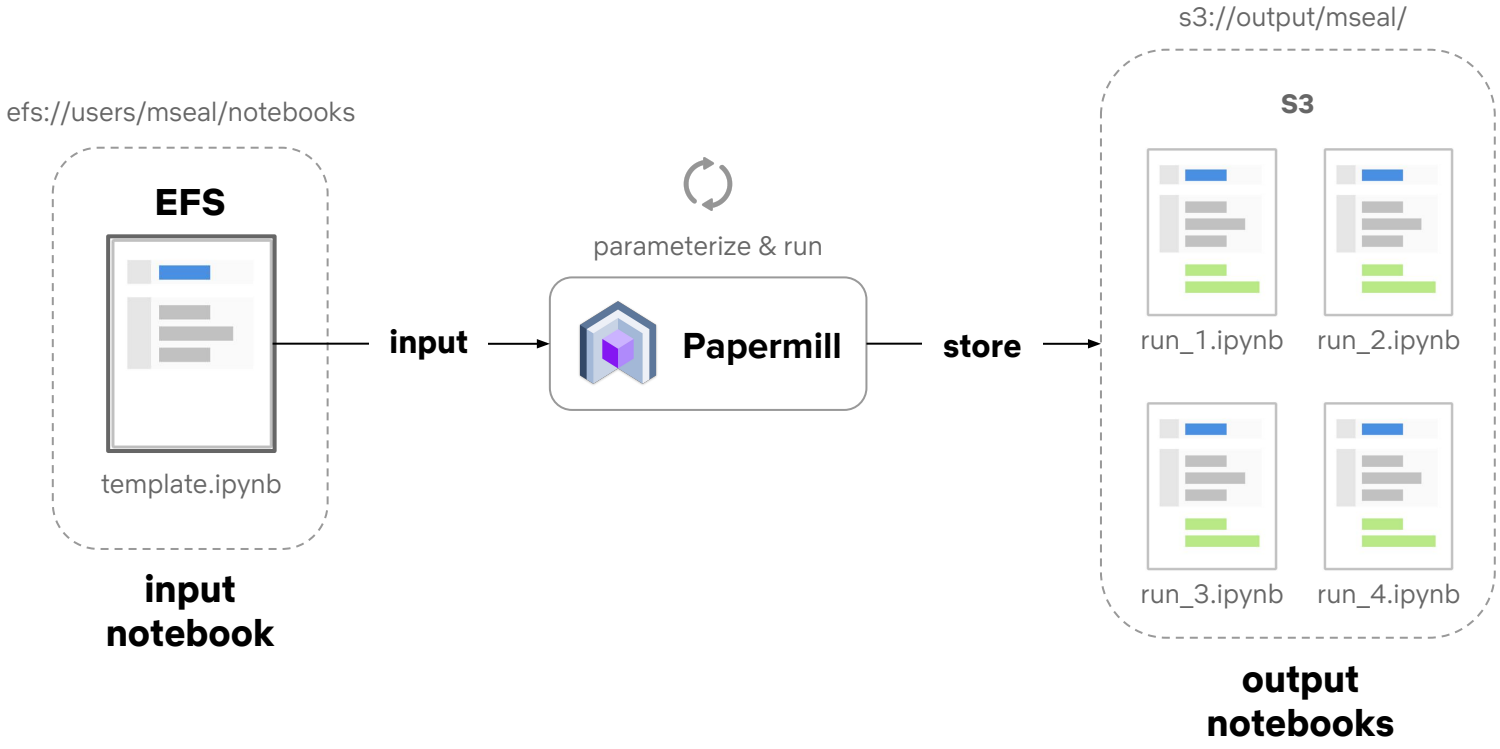
- Not versioned
- Mutable state
- Templating

Papermill

An **nteract** library

NETFLIX

A simple library for executing notebooks.



Choose an output location.

```
import papermill as pm

pm.execute_notebook('input_nb.ipynb', 'outputs/20190402_run.ipynb')



…

# Each run can be placed in a unique / sortable path
pprint(files_in_directory('outputs'))
```

```
outputs/
    ...
    20190401_run.ipynb
    20190402_run.ipynb
```

Add Parameters

```
# Pass template parameters to notebook execution
pm.execute_notebook('input_nb.ipynb', 'outputs/20190402_run.ipynb',
     {'region': 'ca', 'devices': ['phone', 'tablet']})
```

…

```
[2]  # Default values for our potential input parameters
     region = 'us'
     devices = ['pc']
     date_since = datetime.now() - timedelta(days=30)

[3]  # Parameters
     region = 'ca'
     devices = ['phone', 'tablet']
```

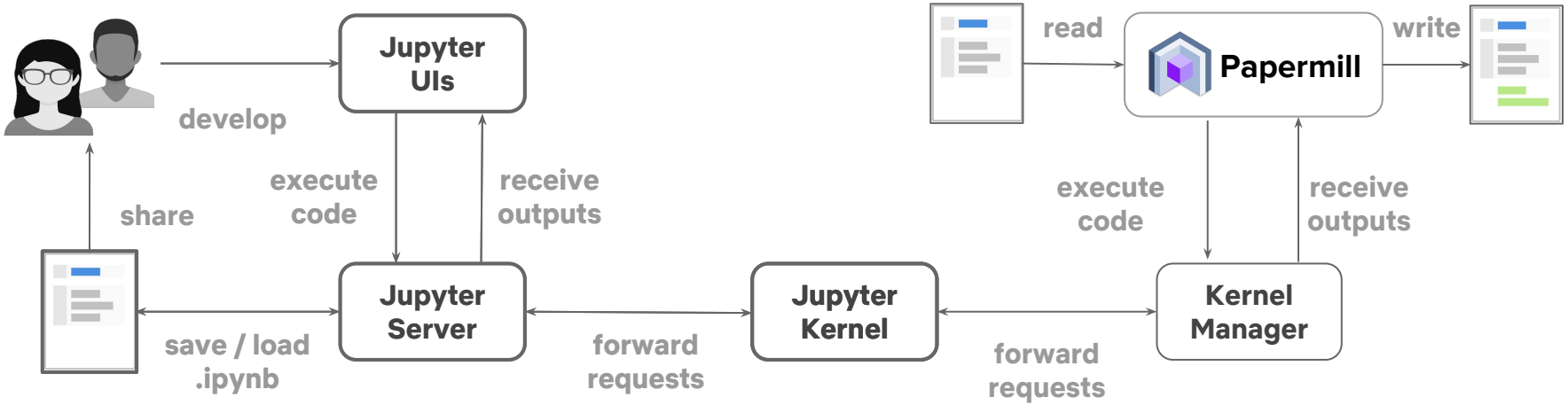
Also Available as a CLI

```
# Same example as last slide
pm.execute_notebook('input_nb.ipynb', 'outputs/20190402_run.ipynb',
     {'region': 'ca', 'devices': ['phone', 'tablet']})
```

…

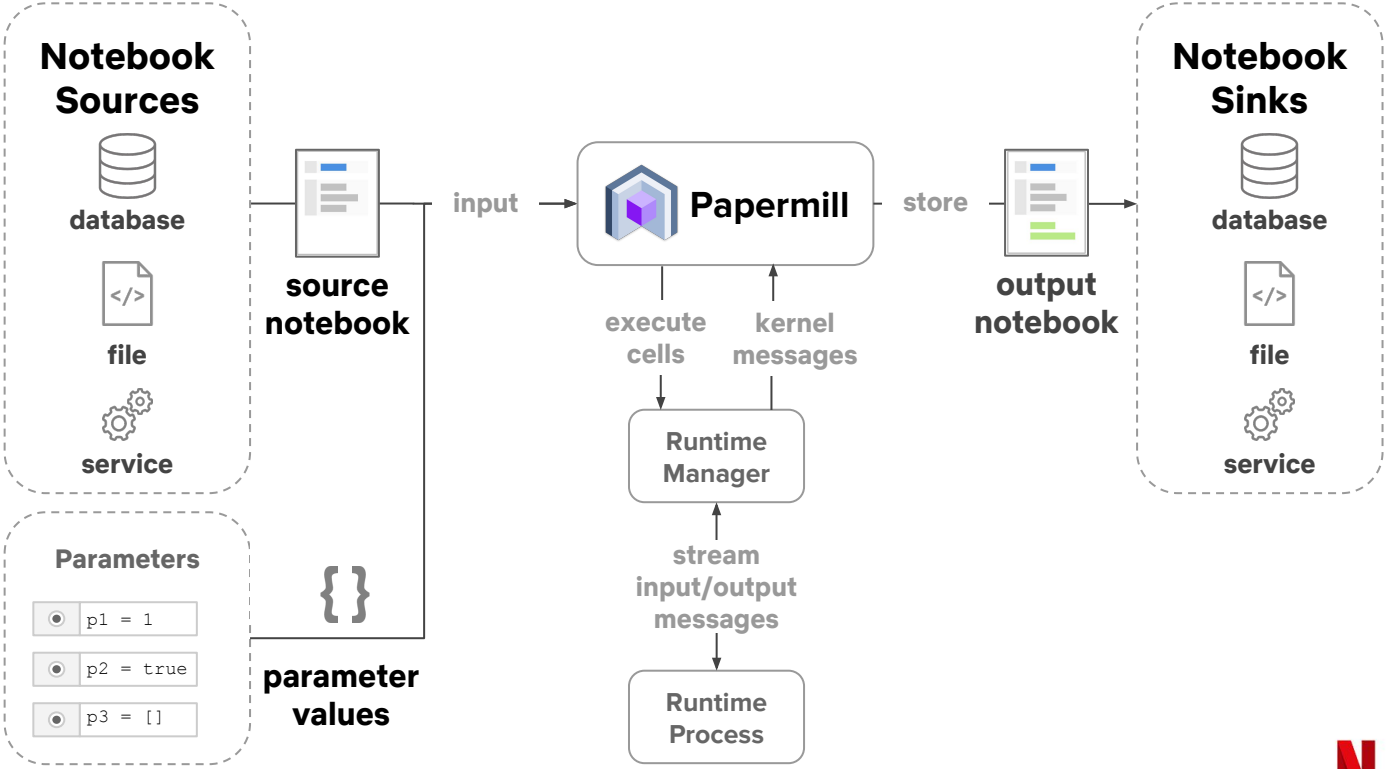
```
# Bash version of that input
papermill input_nb.ipynb outputs/20190402_run.ipynb -p region ca -y
'{"devices": ["phone", "tablet"]}'
```

Notebooks: Programmatically

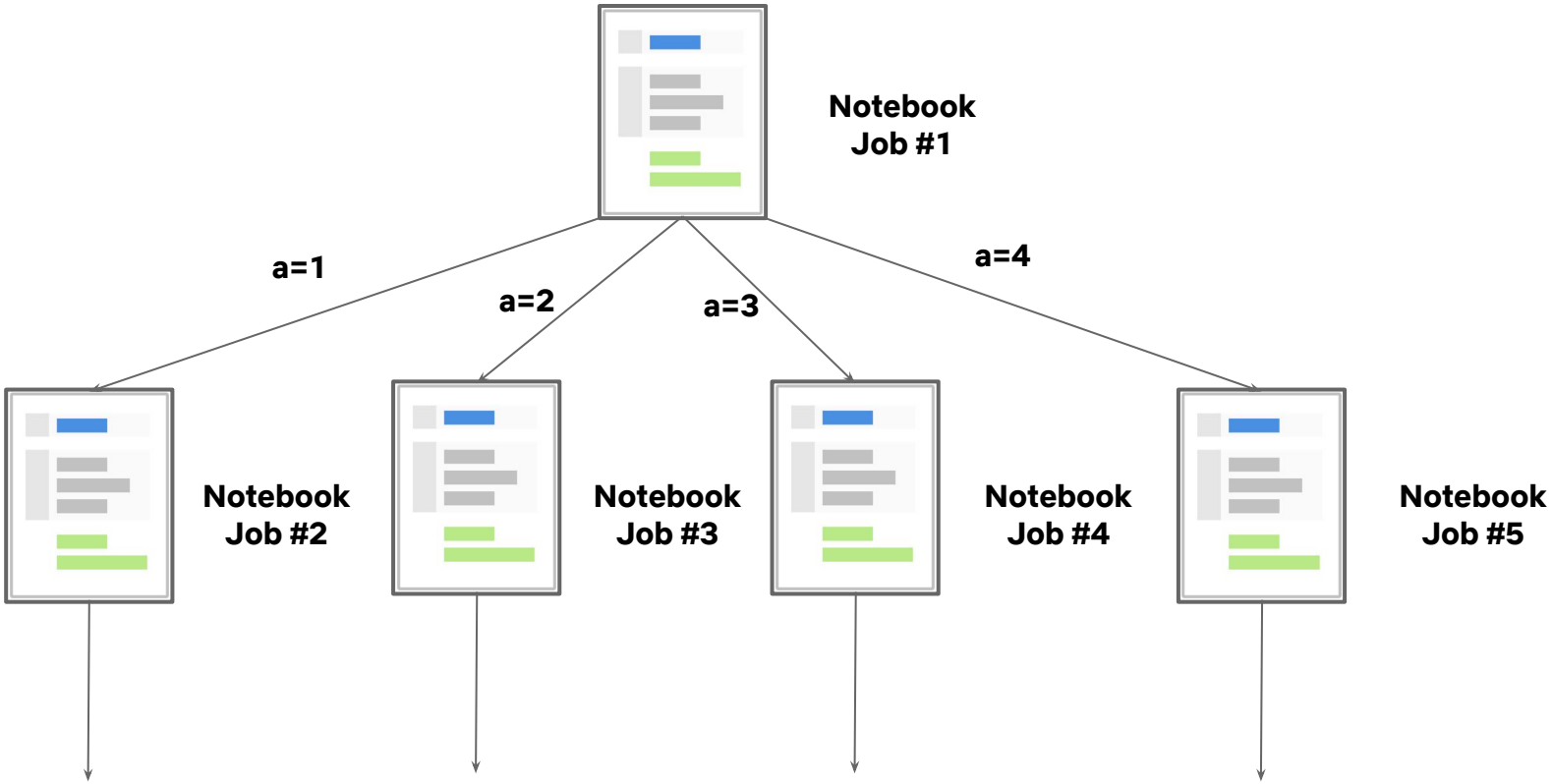


How it works a bit more.

- Reads from a source
- Injects parameters
- Launches a runtime manager + kernel
- Sends / Receives messages
- Outputs to a destination



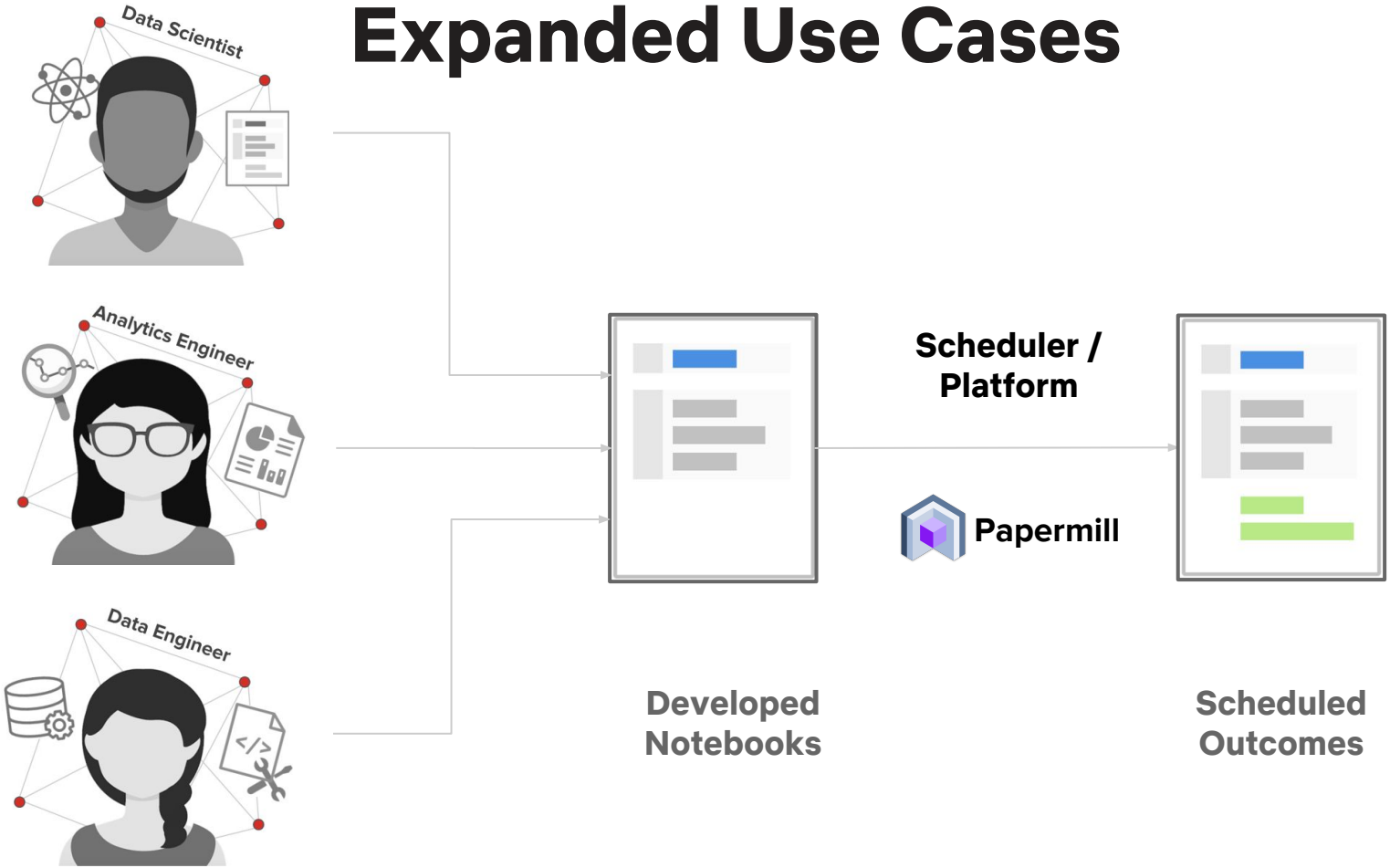
Parallelizing over Parameters.



New Users &

Extensions

Expanded Use Cases



Support for Cloud Targets

```
# S3
pm.execute_notebook(
     's3://input/template/key/prefix/input_nb.ipynb',
     's3://output/runs/20190402_run.ipynb')

# Azure
pm.execute_notebook(
     'adl://input/template/key/prefix/input_nb.ipynb',
     'abs://output/blobs/20190402_run.ipynb')

# GCS
pm.execute_notebook(
     'gs://input/template/key/prefix/input_nb.ipynb',
     'gs://output/cloud/20190402_run.ipynb')

# Extensible to any scheme
```

Plug n’ Play Architecture

New Plugin PRs Welcome

Entire Library is Component Based

```
# To add SFTP support you'd add this class
class SFTPHandler():
    def read(self, file_path):
        ...
    def write(self, file_contents, file_path):
        ...

# Then add an entry_point for the handler
from setuptools import setup, find_packages
setup(
     # all the usual setup arguments ...
     entry_points={'papermill.io': ['sftp://=papermill_sftp:SFTPHandler']}
)

# Use the new prefix to read/write from that location
pm.execute_notebook('sftp://my_ftp_server.co.uk/input.ipynb',
                    'sftp://my_ftp_server.co.uk/output.ipynb')
```

Diagnosing with Outputs

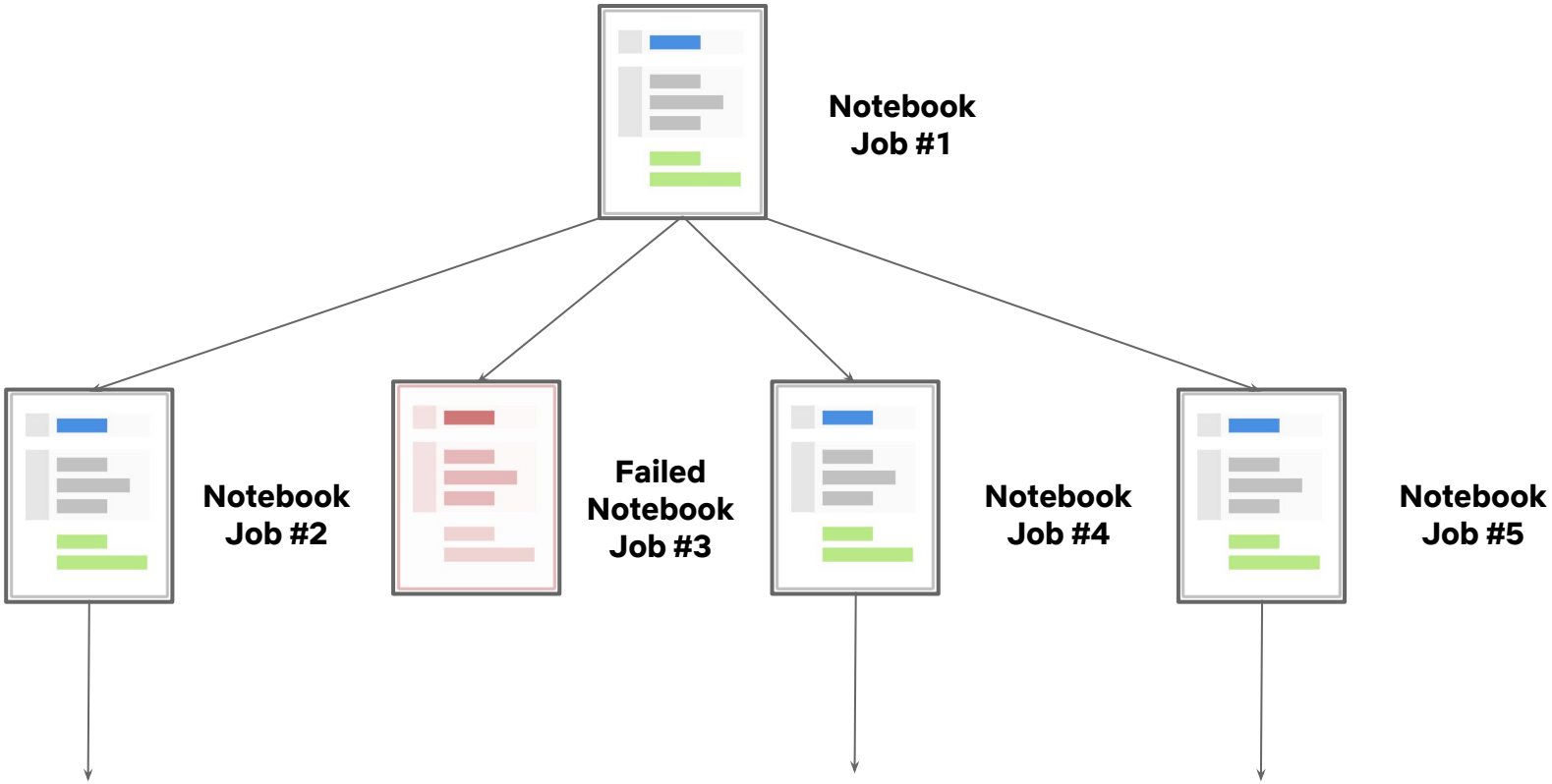
NETFLIX

Failed Notebooks

A better way to review outcomes

NETFLIX

Debugging failed jobs.



Failed outputs are useful.

Output notebooks are the place to look for failures. They have:

- **Stack traces**
- **Re-runnable code**
- **Execution logs**
- **Same interface as input**

Papermill - Parametrized

[3]
```
# Default parameter values
run_date = datetime.today().strftime('%Y_%M_%d')
region_code = 'us'
```

[4]
```
# Injected papermill parameters (specified by user)
region_code = 'fr'
```

[5]
```
script = video_agg_template.format(region=region_code, date=run_date)
print(script)
```

```
CREATE TABLE vid_agg_fr_2018_05_08 AS
(
   SELECT
      device_type,
      count(*)
   FROM
      video_with_region_2018_05_08
   GROUP BY
      device_type
   WHERE
      region_code = 'fr'
)
```

[6]
```
job = SparkJob.script(script).execute()
job.watch_stderr()
```

Find the issue.

```
[12]  job = SparkJob.script(script).execute()
```

```
ConnectionError: HTTPConnectionPool(host='genie.typo', port=80): Max retries exceeded with
url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at
0x7f11ec303dd8>: Failed to establish a new connection: [Errno -2] Name or service not
known',))
```

Test the fix.

```
[21]  # Default parameter values
      run_date = datetime.today().strftime('%Y_%M_%d')
      region_code = 'us'
      cluster_hostname = 'https://genie.corrected.net'
```

...

```
[22]  job = SparkJob.host(cluster_hostname).script(script).execute()
      job.watch_stderr()
```

```
Job succeeded!
```

Update the notebook.

Changes to the notebook experience.

Adds notebook isolation

- Immutable inputs
- Immutable outputs
- Parameterization of notebook runs
- Configurable sourcing / sinking

and gives better control of notebook flows via library calls.

How notebooks

Integrate

NETFLIX

Notebooks Are Not Libraries

Try to not treat them like a library

NETFLIX

Notebooks are good integration tools.

Notebooks are good at connecting pieces of technology and building a result or taking an action with that technology.

They're unreliable to reuse when complex and when they have a high branching factor.

Some development guidelines.

- Keep a low branching factor
- Short and simple is better
- Keep to one primary outcome
- (Try to) Leave library functions in libraries
    - Move complexity to libraries

Tests via papermill

Integration testing is easy now

NETFLIX

Controlling Integration Tests

```
# Linear notebooks with dummy parameters can test integrations
pm.execute_notebook('s3://commuter/templates/spark.ipynb',
     's3://commuter/tests/runs/{run_id}/spark_output.ipynb'.format(
          run_id=run_date),
     {'region': 'luna', 'run_date': run_date, 'debug': True})
…

[3]  # Parameters
     region = 'luna'
     run_date = '20180402'
     debug = True

[4]  spark.sql('''
          insert into {out_table} select * from click_events
          where date = '{run_date}' and envt_region = '{region}'
     '''.format(run_date=run_date, enc_region=region,
                out_table='test/reg_test if debug else 'prod/reg_' + region))
```

Other Ecosystem

Goodies

NETFLIX

Host of libraries.

To name a few:

- **nbconvert**
- **commuter**
- **nbformat**
- **bookstore**
- **scrapbook**
- **...**

See jupyter and nteract githubs to find many others

NETFLIX

Scrapbook

Save outcomes inside your notebook

NETFLIX

Adds return values to notebooks

```
[1]  # Inside your notebook you can save data by calling the glue function
     import scrapbook as sb
     sb.glue('model_results', model, encoder='json')

…

# Then later you can read the results of that notebook by "scrap" name
model = sb.read_notebook('s3://bucket/run_71.ipynb').scraps['model_results']

…

[2]  # You can even save displays and recall them just like other data outcomes
     sb.glue('performance_graph', scrapbook_logo_png, display=True)
```

Commuter

A read-only interface for notebooks

NETFLIX

Commuter Read-Only Interface

Browse Discover

home › bdp › latest › notebooks › etl › PrestoNotebook.ipynb

The `Presto` job type allows you to execute Presto jobs using Genie.

```
[ ]  import kragle as kg
     import aws.s3 as s3
```

```
[ ]  from kragle.parameters.genie import defaults as genie
     from kragle.parameters.presto import defaults as presto
```

```
[ ]  job = kg.genie.PrestoJob().apply_namespaces(genie=genie, presto=presto)
     print("job: {})".format(job))
```

```
[ ]  rj = job.execute()
     print(rj.gaze())
```

```
[ ]  rj.wait()
     print(rj.gaze())
     rj.raise_for_status()
```

No kernel / resources required

NETFLIX

Notebooks

At Netflix

NETFLIX

A strategic bet!

We see notebooks becoming a common interface for many of our users.

We've invested in notebook infrastructure for developing shareable analysis resulting in many thousands of user notebooks.

And we've changed over 10,000 jobs which produce upwards of 150,000 queries a day to run inside notebooks.

We hope you enjoyed the session

Thanks

NETFLIX

Questions?

https://slack.nteract.io/
https://discourse.jupyter.org/