Prefect for data pipelines in Python

Franklin Ventura

ResBaz 20210520

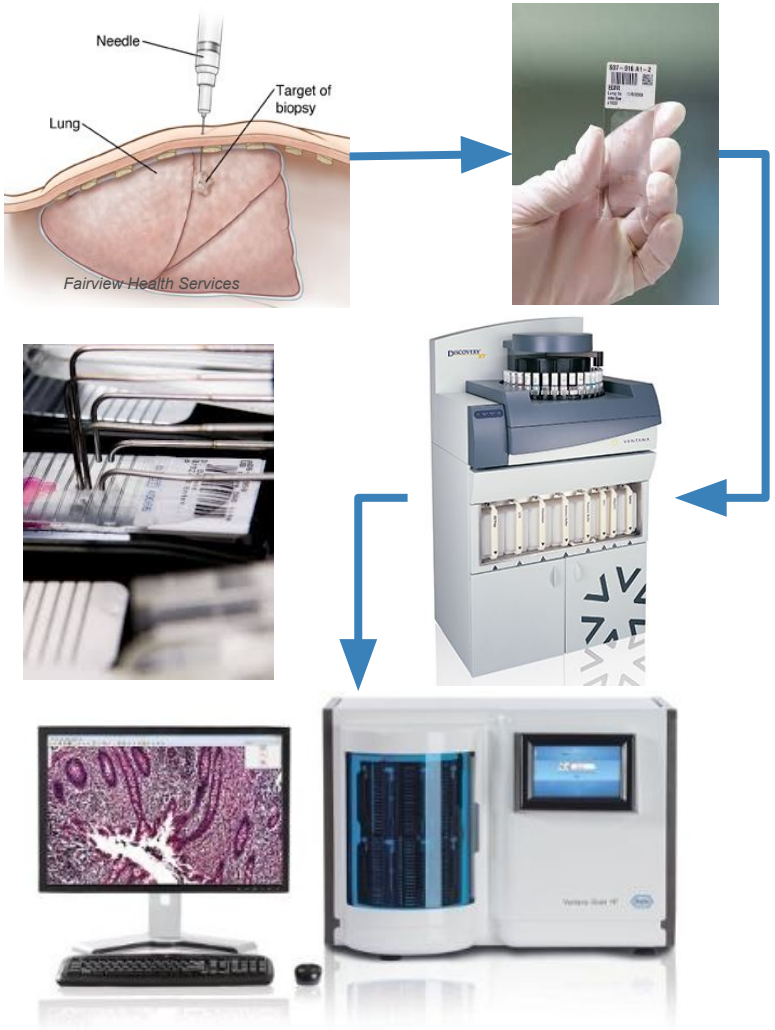
https://github.com/VenturaFranklin/resbazTucson2021-Prefect

PREFECT

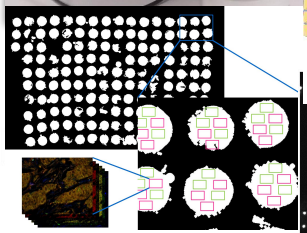
Franklin Ventura

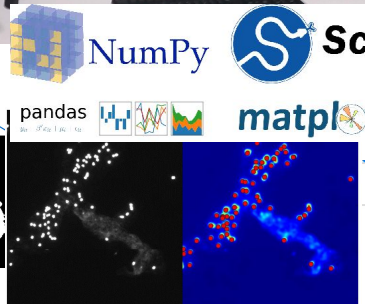
Roche Tissue Diagnostics

- BS Biomedical Engineering
- MS Software/Systems Engr
- 9 years
  - Senior Software Engineer
  - Research and Early Development (tRED)
  - Digital Pathology
  - Cancer Diagnostics
  - RTD Programming Club
    - Beginners & More Experienced
  - Microsoft TEALS
  - Software Carpentry Instructor
  - Tucson Python Language Enthusiasts (Local Meetup)



Variety of Python Applications at RTD



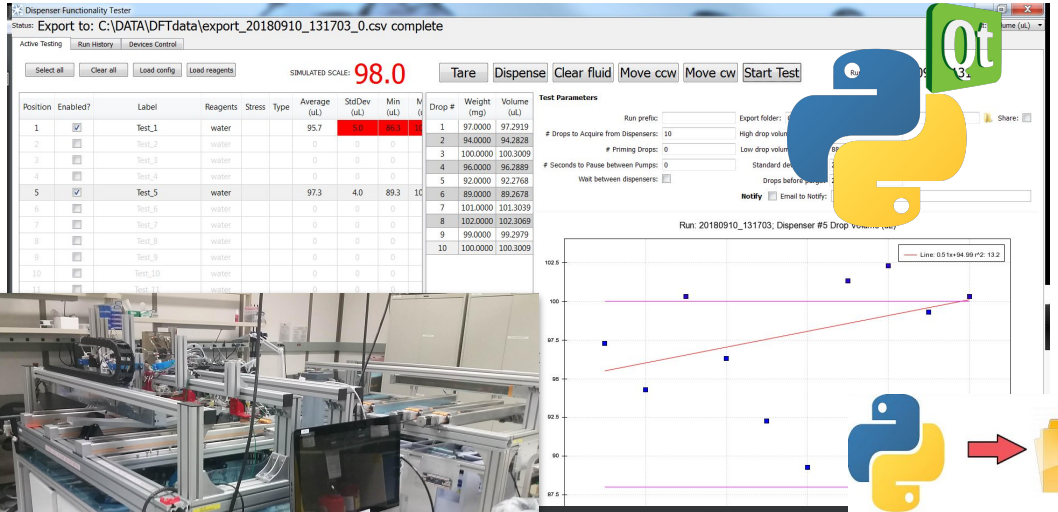


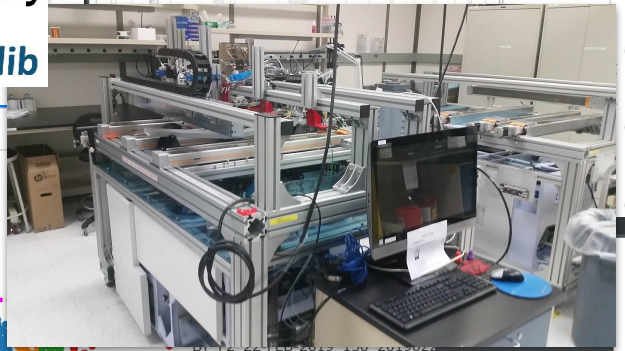
NumPy

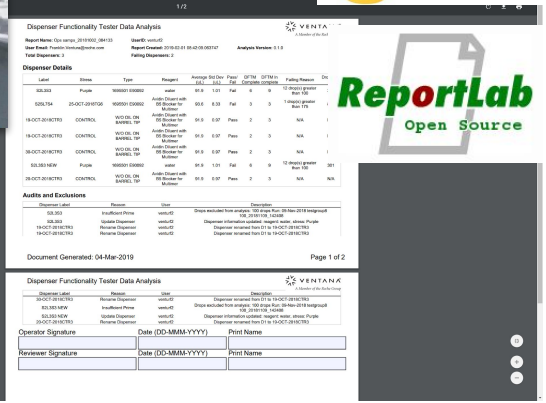
SciPy

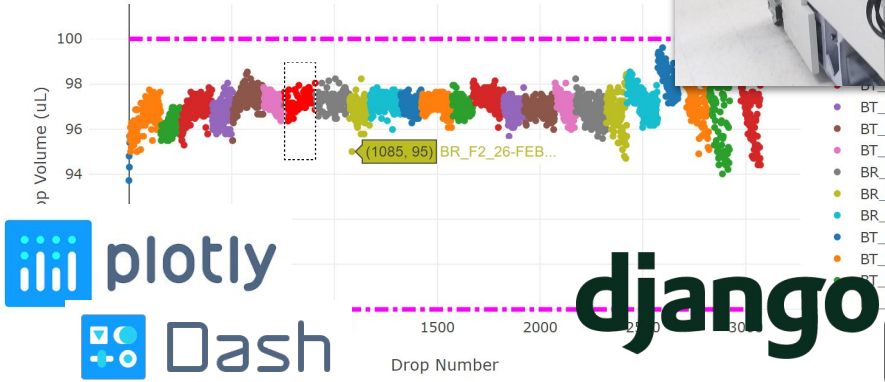
pandas

matplotlib









plotly

Dash

django

Selenium

ReportLab
Open Source

EXE

PREFECT

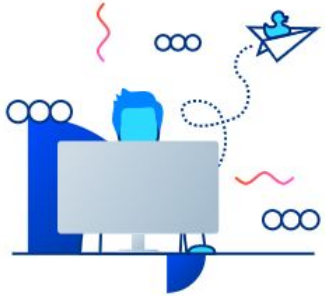
https://www.prefect.io/

- programmatically author, schedule, and monitor workflows
- Extract, Transform, and Load data processing expressed as code

organize and kick off machine learning jobs running on external Dask clusters

"Prefect takes care of scheduling, infrastructure, error handling, retries, logs, triggers, data serialization, parameterization, dynamic mapping, caching, concurrency, and more..."

STEP 01 – CORE



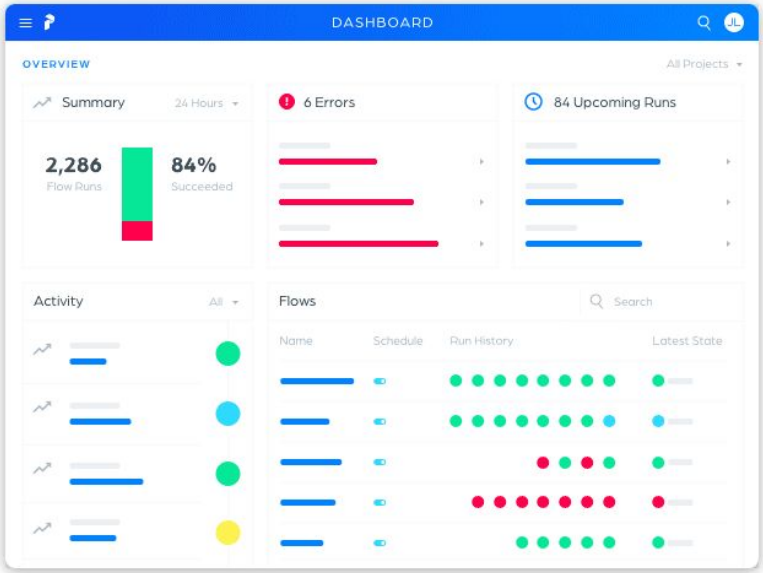
STEP 02 – CLOUD

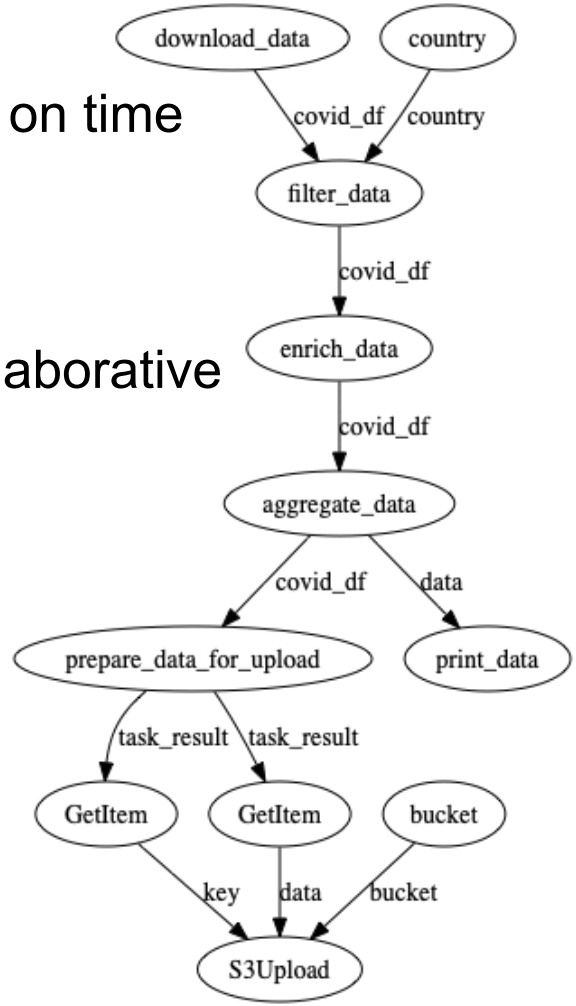
STEP 03 – CORE

STEP 04 – CLOUD

Why use Prefect at all?

- making sure everything is running smoothly on time
- scheduling and coordination
- recovering from failures
- workflows defined as code, become more maintainable, versionable, testable, and collaborative





Will Prefect Get My Data?

- Prefect only gets what you consciously decide to send them
- By default only function names, structure of DAG and log info is sent
- Cloud: [“hybrid execution keeps code and data completely private”]

OR

- Run your own prefect Server which comes with the UI

See:

[Prefect Server vs. Prefect Cloud - which should I choose?]

Prefect - Alternatives

Roche

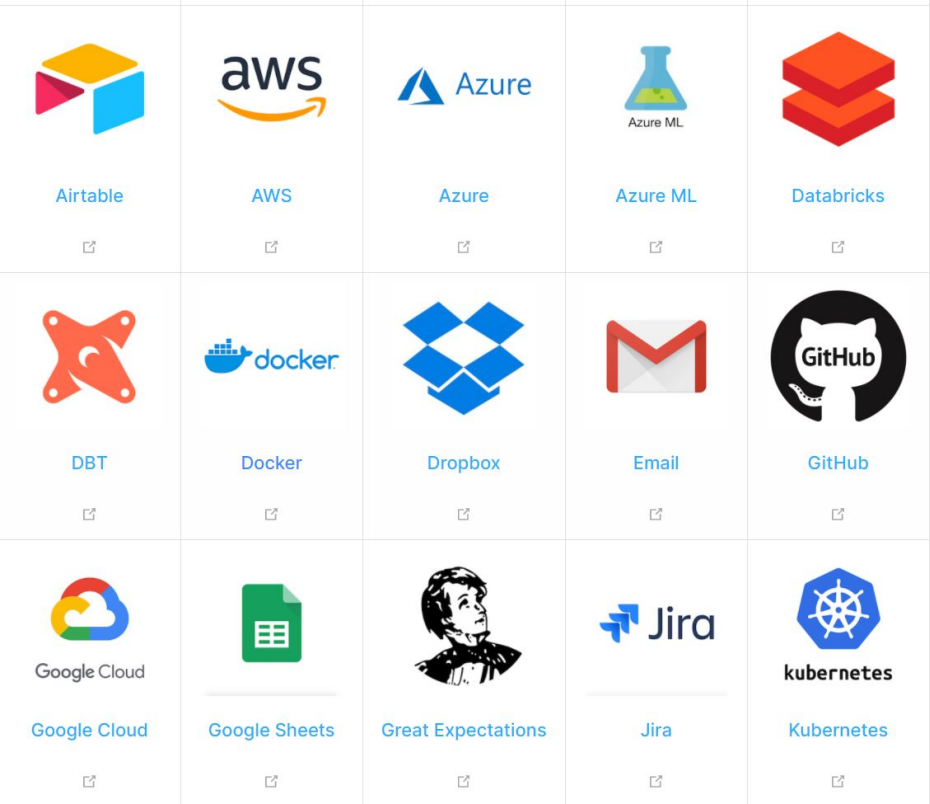
Apache Airflow

PREFECT

Class Based	**API**	Function Based
strict dependency on a specific time	**Scheduling**	any time, with any concurrency, for any reason
Flow not meant to change	**Parameters**	Allow slightly different flow
Second class citizen	**Data**	First class operation
Internal function	**Dynamic Workflows**	Task mapping

Prefect - Building a Flow

It’s just Python functions



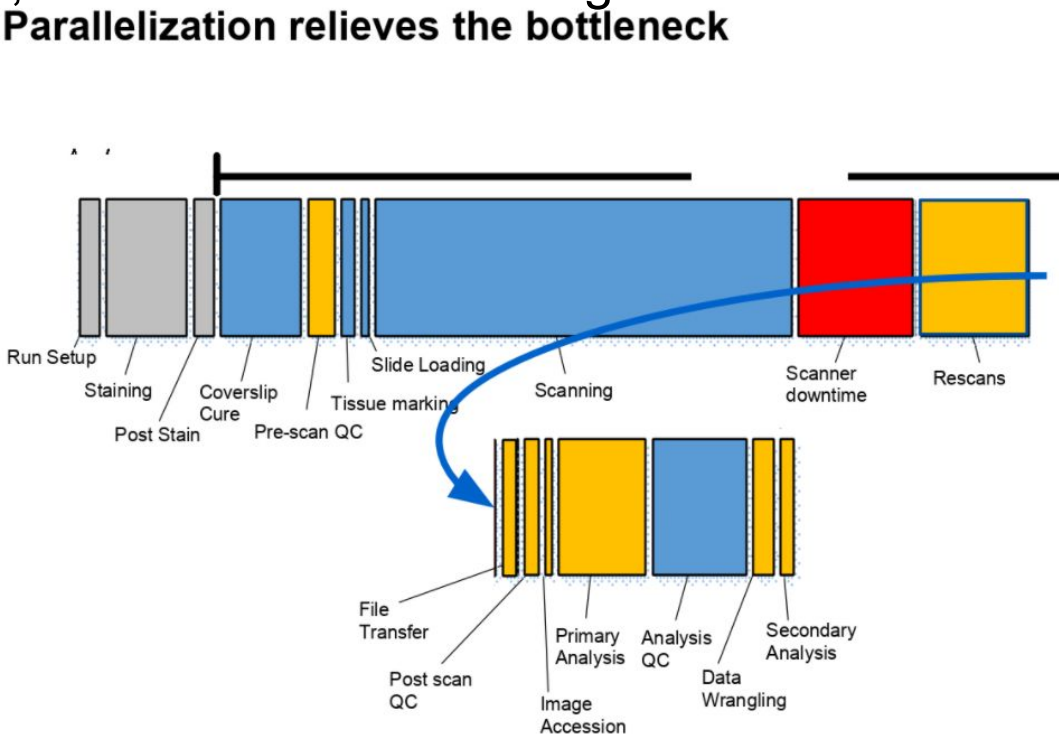

```
def test_flow():
  with Flow() as flow:
    StartContainer(**)
    WriteGsheetRow(**)
    status = WaitOnContainer(**)
    EmailTask(status, **)
    SlackTask(**)
    custom_cleanup(**)
```

Data

Orchestration

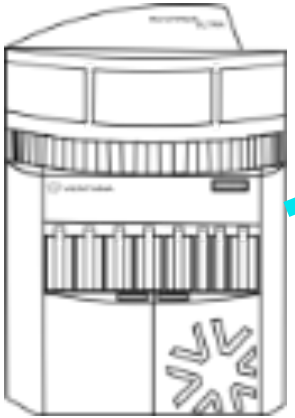
Transfer images from scanner; Track metadata for images

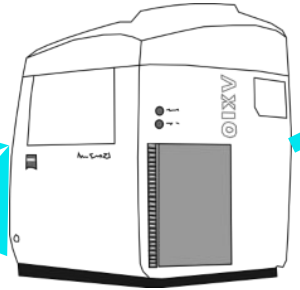
- Removing scanner bottle necks
- Reduction of touch points
- Flexible Metadata collection
- Data pipeline analytics, monitoring, and error tracking



Solution

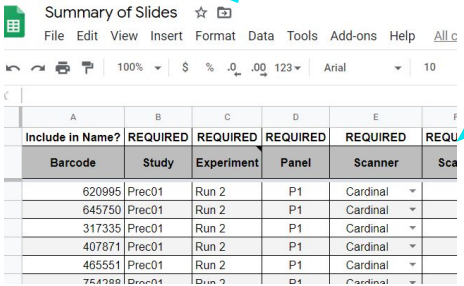
Process Improvement





PREFECT

Roche



Summary of Slides

File Edit View Insert Format Data Tools Add-ons Help All c

100% $ % .0 .00 123 Arial 10

	A	B	C	D	E	F
	Include in Name?	REQUIRED	REQUIRED	REQUIRED	REQUIRED	REQU
	Barcode	Study	Experiment	Panel	Scanner	Sca
	620995	Prec01	Run 2	P1	Cardinal	
	645750	Prec01	Run 2	P1	Cardinal	
	317335	Prec01	Run 2	P1	Cardinal	
	407871	Prec01	Run 2	P1	Cardinal	
	465551	Prec01	Run 2	P1	Cardinal	
	754288	Prec01	Run 2	P1	Cardinal	

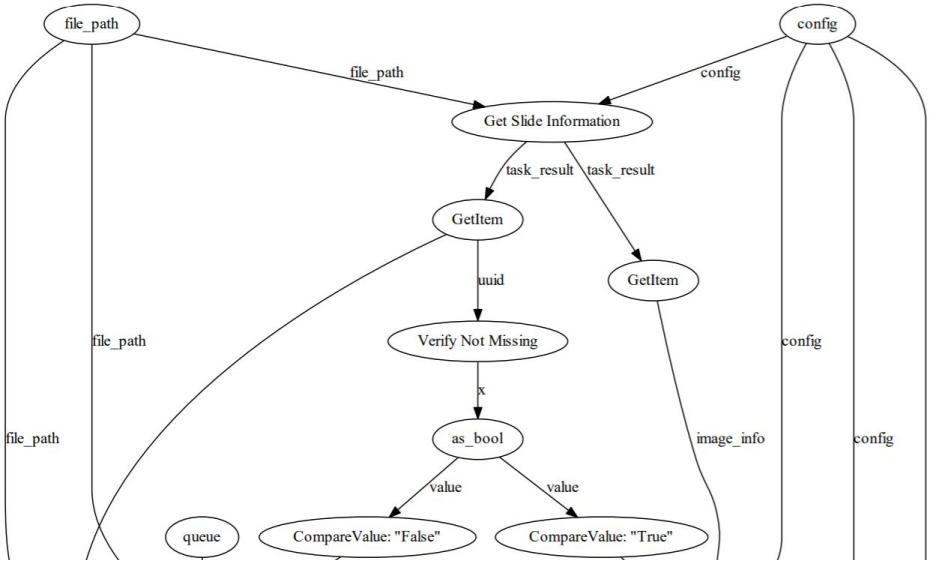
Data Pipeline

Clearly organize the various steps and the associated order

- Configure retry policies into individual tasks
- Set up alerting in the case of failures, retries, as well as tasks running longer than expected
- Comes with an intuitive UI with powerful tools for monitoring and managing jobs

[Version 2.5.0]

[Version 2.10.5]



Data Flow Orchestration

Track images through system - Gain Analytics

DASHBOARD

All Projects

All Projects

OVERVIEW FLOWS AGENTS

Run History

a month ago

an hour ago

Summary 24 Hours

In the last day

11 flow runs

90.9 % succeeded

1 Failed Flows 24 Hours

Scan Image Flow: Raven

2:19pm MST

0 Upcoming Runs

UPCOMING

LATE

No upcoming runs.

Data Flow Monitoring

Monitor systems

DASHBOARD

Aurora

OVERVIEW | FLOWS | AGENTS

Cardinal Agent
LocalLocalAgent

LAST QUERY

4:37pm | 6 seconds ago

CORE VERSION	TOKEN ID
0.12.0	27ce4d63-595f-4...

LABELS

azure-flow-storage, Cardinal, gcs-flow-storage, RTUMWXCOEXT0065, s3-flow-storage

Raven Agent
LocalLocalAgent

LAST QUERY

4:37pm | 10 seconds ago

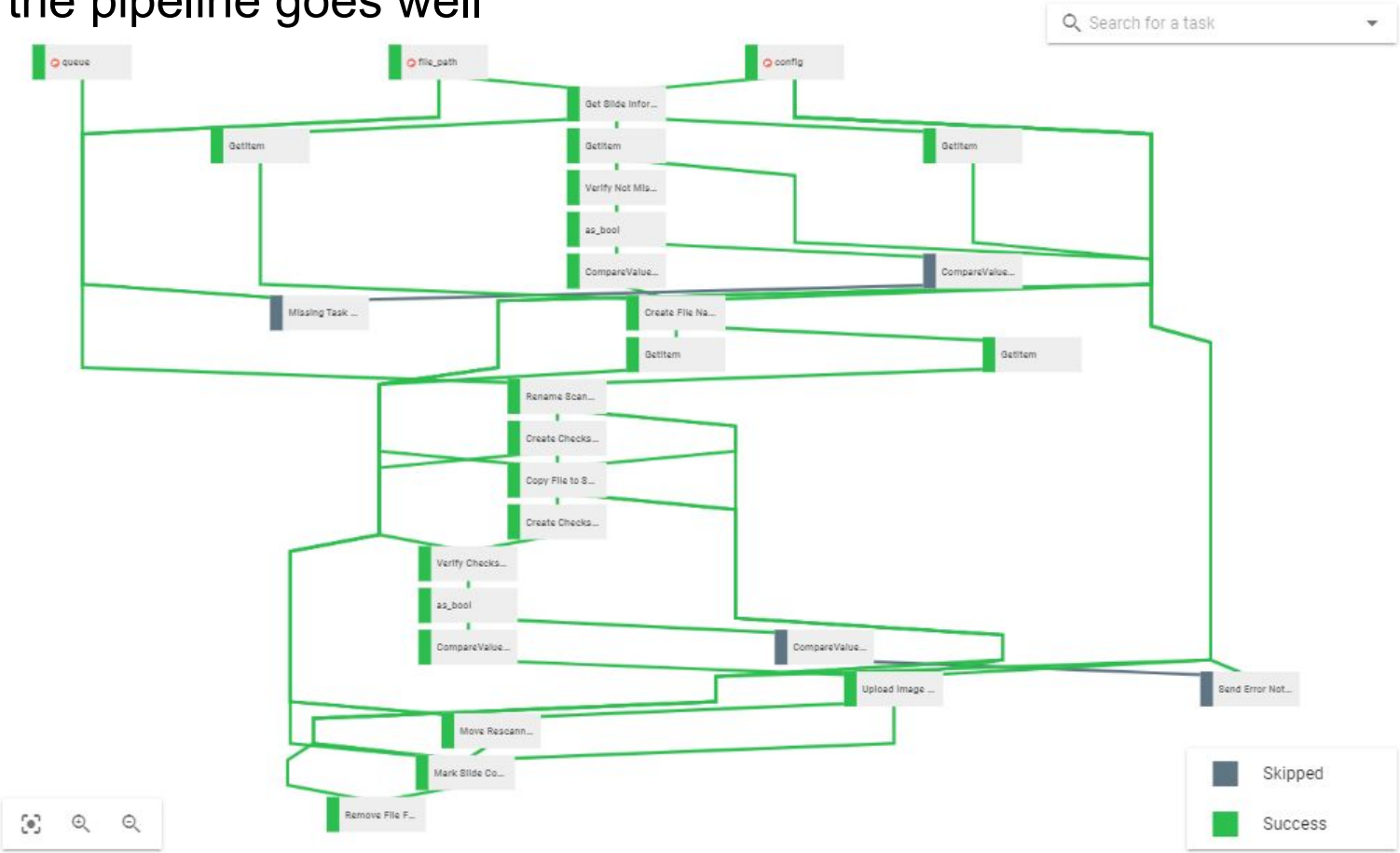
CORE VERSION	TOKEN ID
0.12.0	83f162a6-5690-4...

LABELS

azure-flow-storage, gcs-flow-storage, Raven, RTUMWXCOEXT0050, s3-flow-storage

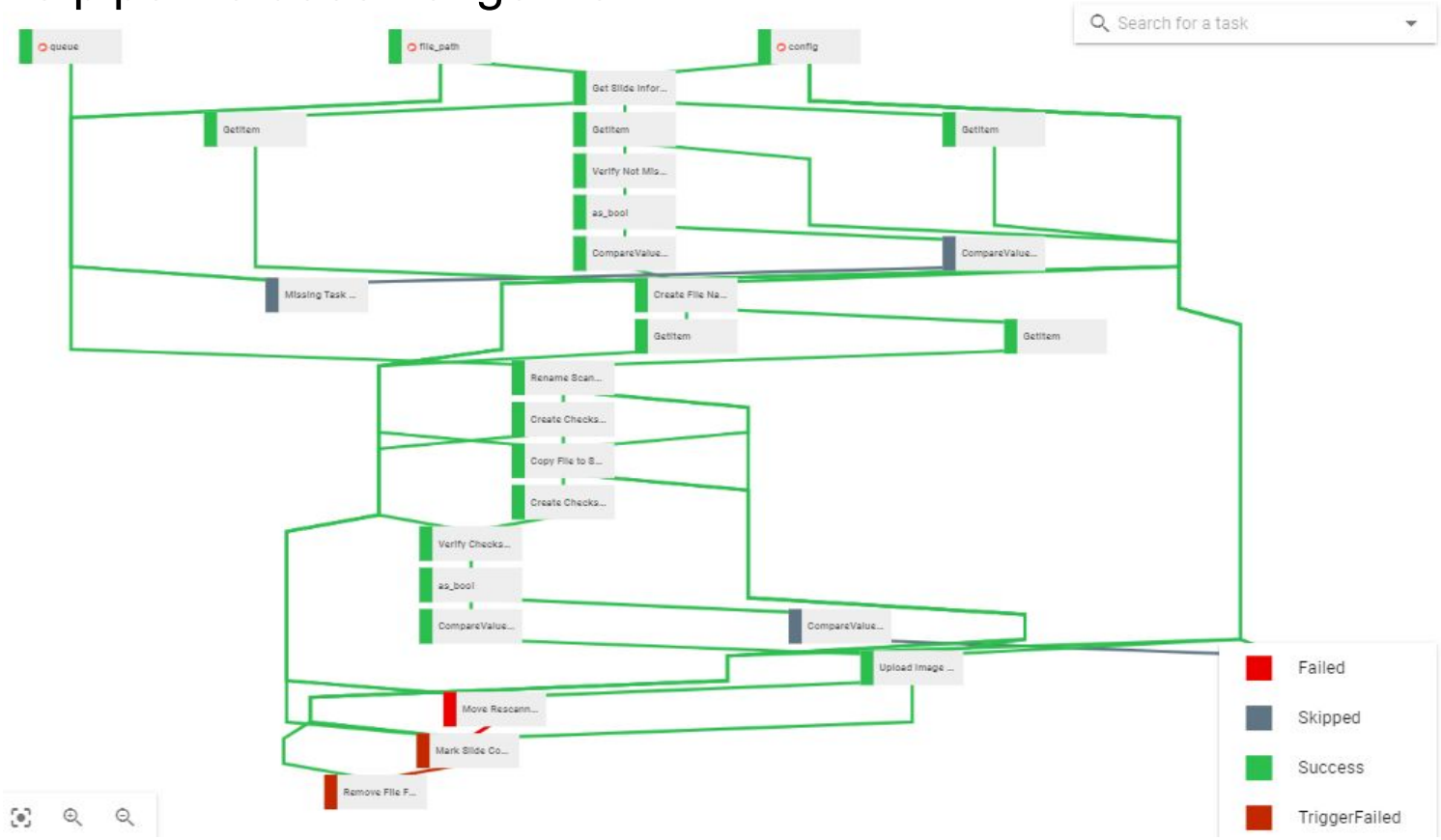
Data Pipeline Visualization

When the pipeline goes well



Data Pipeline Visualization

When the pipeline does not go well



Error Investigation

Dive into errors remotely, no combing through logs

545891 - Move Rescanned Images

Aurora › Scan Image Flow: Raven › 545891 ›

RESTART MARK AS

OVERVIEW LOGS

Move Rescanned Images

Failed

Flow Run

545891

Failed

Last State Message

[6 Aug 2020 2:19pm]: Unexpected error: FileNotFoundError('No image found for feb90bff5f5f73db')

Started	6 Aug 2020 2:19pm
Ended	6 Aug 2020 2:19pm
Duration	1 second
Updated	6 Aug 2020 2:19pm
Class	FunctionTask
Result Type	NoResultType
Result Location	None

Dependencies

3 Upstream • 2 Downstream

Show Cards

GetItem
Duration: <1s

config
Duration: <1s

Upload Image
Duration: 1s

Search for a task

Task Run
545891 - Move Rescan...
Failed - 6 Aug 2020 2:1...

Task
Move Rescanned Images

Task Run State Message:
Unexpected error: FileNotFoundError('No image found for feb90bff5f5f73db')

Max retries: 0

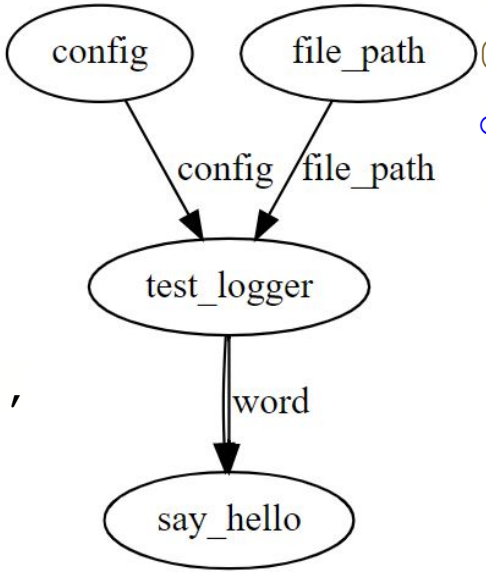
Class: FunctionTask

Activity

Prefect - Hello World Simple Example

```
def define_test_flow():
  with Flow() as flow:
    file = Parameter("file_path")
    config = Parameter("config")
    word = test_logger(file, config)
    hello(word,
          upstream_tasks=[word])
    return flow
```

```
flow = define_test_flow()
flow.visualize()
flow.run(file_path="TEST_PATH",
         config={1: "Test"})
```




```
@task
def test_logger(file,
config):
  logger.debug(
    f"Received {file}")
  return "Hello"
@task(log_stdout=True)
def hello(word):
  print(f"Say, {word}.")
```


Roche
PREFECT
Questions?
Prefect or Other

Doing now what patients need next