Mastering Data Flow:

Empower Your Projects with Prefect's Pipeline Magic

Dr Adam Hill

Before we get started ...

Download the code
https://bit.ly/pydata-Prefect-WS

Run:

```
docker-compose up --build --force-recreate --remove orphans
```

Create a free account:

- MongoDB Atlas – https://www.mongodb.com/cloud/atlas/register

Comply Advantage

We are recruiting!

Help us fight financial crime and tackle a $1.4 trillion problem!

Senior Data Scientist (London) https://bit.ly/CA-SDS-LON

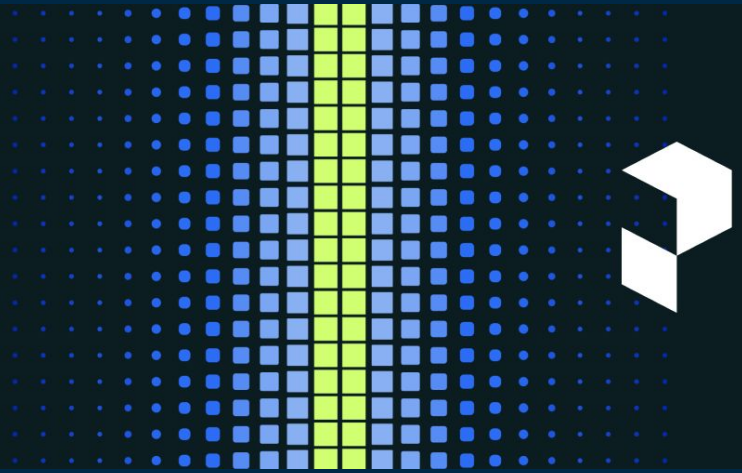
Senior Data Scientist (Lisbon) https://bit.ly/CA-SDS-LIS

AIM

By the end of this session you will:

- Understand what Prefect is
- Build and execute tasks and flows
- Have scheduled a flow using deployment
- Have a grasp of what else can be done
- Had some fun 😆

PREFECT



V3 was released only two weeks ago!

Adam Hill

tldr;

COMPLY ADVANTAGE

Senior Data Scientist

Financial Crime Fighter

Day Job

Charity

DataKind volunteer

DataDive volunteer; Data Ambassador; Committee member

DataKind

Evangelist

Me

Former Royal Society Entrepreneur in Residence

Recovering astrophysicist

UNIVERSITY OF Southampton

THE ROYAL SOCIETY

Find me at

@astroadamh

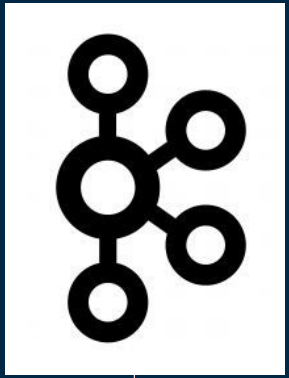
www.linkedin.com/in/adambenhill/

Horsewithapointyhat.com

First lesson: External dependencies

To demo interfacing with external systems the original plan was to use the Upstash free-tier serverless Kafka

upstash



Last week they switched off the service for new users!

But kudos to Upstash that they engaged to help me out when I flagged my problem

Prefect Technologies
data-warehouse

- Dashboard
- Flow Runs
- Flows
- Deployments
- Work Pools
- Blocks
- Variables
- Automations
- Event Feed
- Event Webhooks
- Artifacts
- Settings

Help

Bill Palombi
bill@prefect.io

Dashboard

All tags | 8h | 24h | 1w

Flow Runs

1,091 total

11 | 2 | 1078 | 0 | 0

Flows with failed or crashed runs

orbit-to-bigquery 1
3h 50m ago

orbit-to-bigquery > private-numbat auto-scheduled
Failed 2023/09/01 11:45:07 AM 1m 54s 10 task runs
Deployment Orbit to BigQuery
Work Pool kubernetes-prd-data-warehouse Work Queue default

run-census-sync 1
3h 53m ago

main-orchestrator 1
3h 59m ago

delete-cloud1-tenant 2
19h 23m ago

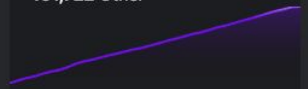
cloud2-to-data-warehouse 6

Task Runs

39,658
39,649 Completed 99.98%
9 Failed 0.02%

Events

407,844
5,668 Block
454 Worker
401,722 Other



Active Work Pools

kubernetes-legacy-data-warehouse 150 total

Polled	Work Queues	Late runs	Completes
33s ago	●	0 (10s avg.)	98.67% + 0.9

kubernetes-prd-data-warehouse 852 total

Polled	Work Queues	Late runs	Completes
18s ago	●●●●●●	0 (11s avg.)	99.65% + 1.9

kubernetes-stg-data-warehouse 76 total

Polled	Work Queues	Late runs	Completes
18s ago	●●●●●●	0 (10s avg.)	93.42% - 4.3

Calling a task from a flow

Use the `@task` decorator to designate a function as a task. Calling the task creates a new task run:

```
from prefect import flow, task

@task
def my_task():
    print("Hello, I'm a task")

@flow
def my_flow():
    my_task()
```

Let’s create our first flow!

Adding detail to the task

```python
import datetime
from prefect import flow, task

def generate_task_name():
    date = datetime.datetime.now(datetime.timezone.utc)
    return f"{date:%A}-is-a-lovely-day"

@task(name="My Example Task",
      description="An example task for a tutorial.",
      task_run_name=generate_task_name)
def my_task(name):
    pass

@flow
def my_flow():
    # creates a run with a name like "Thursday-is-a-lovely-day"
    my_task(name="marvin")
```

Logging out of the box

log_prints=True

We could have achieved the exact same outcome by using Prefect's convenient `log_prints` keyword argument in the `flow` decorator:

```
@flow(log_prints=True)
def get_repo_info(repo_name: str = "PrefectHQ/prefect"):
    ...
```

repo_info.py

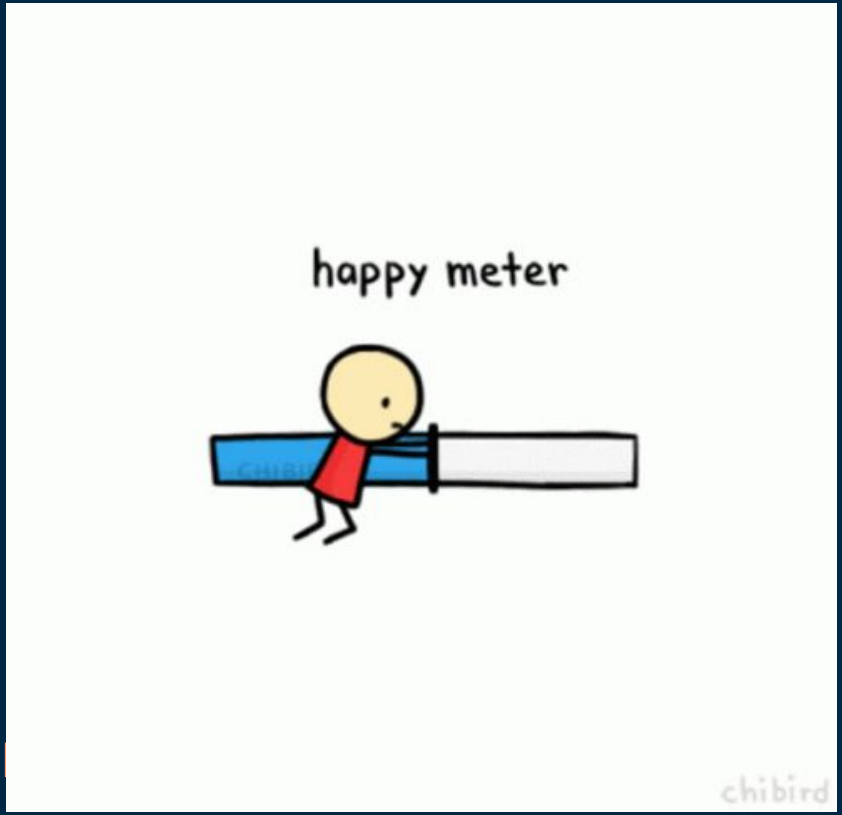
```
import httpx
from prefect import flow, get_run_logger


@flow
def get_repo_info(repo_name: str = "PrefectHQ/prefect"):
    url = f"https://api.github.com/repos/{repo_name}"
    response = httpx.get(url)
    response.raise_for_status()
    repo = response.json()
    logger = get_run_logger()
    logger.info("%s repository statistics 🤓:", repo_name)
    logger.info(f"Stars 🌟 : %d", repo["stargazers_count"])
    logger.info(f"Forks 🍴 : %d", repo["forks_count"])
```

Let’s start building

Our objective is to build an engine that will monitor regularly for new “tweets” about airlines, move that data into a Kafka message queue. From there we will pick it up, process it, run a sentiment analysis model over it. And finally we will store the original data and the sentiment calculations in a MongoDB in the Cloud!

So to make our lives easier let’s start breaking down the problem …

DEMO / BUILDING



Aside: Testing

If all our functions are decorated with @task and @flow, how can we test the raw function output?

Consider our first implementation ...

```python
from prefect import task, flow

@task(name="Addition operator")
def add(a, b):
    return a + b

@task(name="Squaring function")
def square_num(num):
    return num ** 2

@flow(log_prints=True, name="Demo 1")
def add_and_square(a:int = 2, b:int = 3):
    add_result = add(a, b)
    square_result = square_num(add_result)
    print(f"({a} + {b}) squared =
{square_result}")
```

Aside: Testing

```python
from solution.s01_my_first_flow import add, add_and_square


# Test the `add` task directly using core logic
def test_add():
    result = add.fn(2, 3)  # Bypassing Prefect's task layer
    assert result == 5, f"Expected {number_data['expected_sum']}, got
{result}"

# Test the flow directly, bypassing the Prefect orchestration
def test_add_and_square_sysout_flow(capsys):
    _ = add_and_square.fn(2, 5)  # Run flow logic directly
    captured = capsys.readouterr()  # Capture print output
    assert str(25) in captured.out, "Flow output does not match expected
value"
```

Key problem: Scheduling

```
if __name__ == "__main__":
    get_repo_info.serve(
        name="my-first-deployment",
        cron="* * * * *",
        tags=["testing", "tutorial"],
        description="Given a GitHub repository, logs repository statistics for that
repo.", version="tutorial/deployments",
    )
```

Key problem: Secrets Management

```python
from prefect.blocks.system import Secret

# Create a Secret block with database credentials
credentials = {
    "username": "your_db_username",
    "password": "your_db_password",
    "server": "your_db_server"
}

secret_block = Secret(value=credentials)

# Save the block with a unique name
secret_block.save(name="db_credentials", overwrite=True)
```

Key problem: Secrets Management

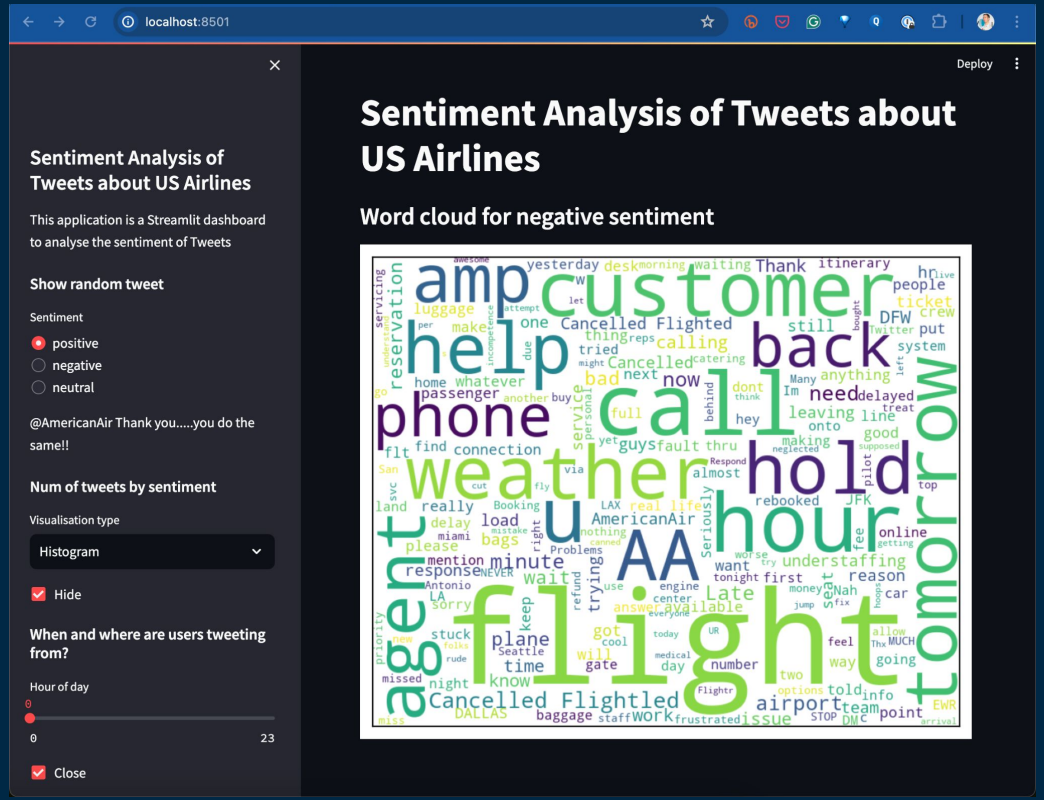
```python
from prefect import flow, task
from prefect.blocks.system import Secret

@task
def connect_to_database():
    # Retrieve the Secret block by name
    db_credentials = Secret.load("db_credentials").get()

    username = db_credentials['username']
    password = db_credentials['password']
    server = db_credentials['server']

    print(f"Connecting to database at {server} with user {username}")
    # Add logic to connect to the database using the credentials
```

See what we've built …

Go to http://localhost:8501

From docker you are running a Streamlit app connected to your Atlas database.

You can see the data change as the code runs…



What we didn't have time for ...

- Parallel processing; e.g. dask can be used to run jobs in parallel
- Triggers & Alerts
- Plugging into K8s
- Using third-party plug-ins e.g. dbt-runner, docker-runner etc.

Lots more to learn!

We are recruiting @ Comply Advantage

Senior Data Scientist (London) https://bit.ly/CA-SDS-LON

Senior Data Scientist (Lisbon) https://bit.ly/CA-SDS-LIS

Adam Hill

tldr;

COMPLY ADVANTAGE

Senior Data Scientist

Financial Crime Fighter

Former Royal Society Entrepreneur in Residence

Recovering astrophysicist

UNIVERSITY OF Southampton

THE ROYAL SOCIETY

Thank you for listening

DataKind volunteer

DataDive volunteer; Data Ambassador; Committee member

DataKind

Find me at

@astroadamh

www.linkedin.com/in/adambenhill/

Horsewithapointyhat.com