PREFECT

An Intro to Workflow Orchestration

Kevin Kho

Budapest
Data Forum

Overview

01 **Simple Problem**
Building a Dogecoin Tracker

02 **Workflow Orchestration**
Eliminating Negative Engineering with Prefect

03 **Constructing Prefect Flows**
Learn How to Make Prefect Flows

04 **Execution Demo**
Using Prefect and Dask to parallelize Flows

Motivation

- Track DOGE prices
- Alert me when there’s a dip

coindesk

NEWS LEARN TV VIDEOS PODCASTS RESEARCH EVENTS

Bitcoin 24h $40,349.25 -10.60%
Ethereum 24h $3,041.81 -11.50%
XRP 24h $1.46 -3.98%
Stellar 24h $0.588148 -11.08%
Cardano 24h $1.80 -14.01%

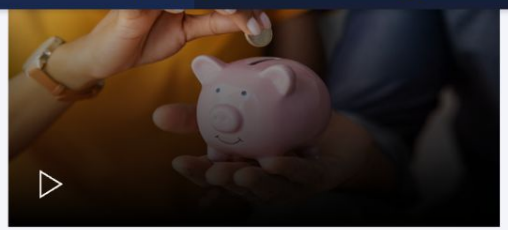
INDEXES TOP ASSETS

Linear Log

EXPORT DATA

05/18/2021 to 05/19/2021

1h 12h 1d 1w 1m 3m 1y all



Saving For Retirement? There's a New Player in Crypto IRA

May 14, 2021

Market Wrap: Elon Taketh Away – Bitcoin Continues Fall as Options Traders Pile Into Puts

Daniel Cawrey May 17, 2021

Musk Learns the Hard Way: Crypto Doesn't Need a Savior

David Morris May 17, 2021

Want to Buy Dogecoin? Read This First

Dogecoin Tracker

- Construct URL
- Get Request
- Detect Dips
- Slack Notification

doge.py

```
def format_url(coin: str= "DOGE") -> str:
    url =
"https://production.api.coindesk.com/v2/price/values/"
    start_time = (datetime.now() -
timedelta(minutes= 10)).isoformat(timespec= "minutes")
    end_time =
datetime.now().isoformat(timespec= "minutes")
    params =
f"?start_date= {start_time} &end_date= {end_time} &ohlc=fal
se"
    return url + coin + params

def get_data(coin: str= "DOGE") -> pd.DataFrame:
    prices = re.get(format_url(coin))
    prices = prices.json()[ 'data'][ 'entries' ]
    data = pd.DataFrame(prices, columns=[ "time",
"price"])
    return data
```

Dogecoin Tracker

- Construct URL
- Get Request
- Detect Dips
- Slack Notification

doge.py

```
def detect_dip(data, threshold =  10):
    peak = data['price'].max()
    bottom = data['price'].min()
    dip = 100 - (bottom/peak)*100
    if dip > threshold:
        return True
    else:
        return False

def post_to_slack():
    pass
```

Dogecoin Tracker

- Running the script

- What if the API fails?
- What if I wanted track another coin?
- Off schedule runs
- Can get notifications if it fails?

doge.py

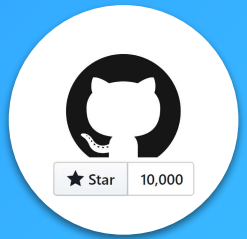
```
while True:
    time.sleep(300)
    data = get_data()
    is_dip = detect_dip(data)
    if is_dip:
        post_to_slack()
```

Negative Engineering

- Failure Logic
- Retries when APIs go down
- Malformed Data
- Notifications
- Failure Messages

PREFECT

- Open-source workflow orchestration
- Python-based
- Native Dask integration
- Modern data stack
- Very active community
- Prefect Cloud/Prefect Server



6400+

6400+

Prefect Tasks

- Flows are comprised of Tasks

doge_with_prefect.py

```
@task
def get_data(coin="DOGE") -> pd.DataFrame:
    prices = re.get(format_url(coin))
    prices = prices.json()['data']['entries']
    data = pd.DataFrame(prices, columns=["time",
"price"])
    return data

@task
def detect_dip(df: pd.DataFrame, threshold =  10):
    peak = df['price'].max()
    bottom = df['price'].min()
    dip = 100 - (bottom/peak)*100
    if dip > threshold:
        return True
    else:
        return False
```

Prefect Flows

- First-class passing of data
- Automatic setting of dependencies
- Local testing

doge_with_prefect.py

```
with Flow("to-the-moon") as flow:
    data = get_data()
    is_dip = detect_dip(data)
    post_to_slack()


flow.run()
```

Retries

- Retries can be set on a task level

doge_with_prefect.py

```
@task(max_retries = 3,
retry_delay=timedelta(minutes= 1))
def get_data(coin="DOGE") -> pd.DataFrame:
    prices = re.get(format_url(coin))
    prices = prices.json()['data']['entries']
    data = pd.DataFrame(prices, columns=[ "time",
"price"])
    return data

@task
def detect_dip(df: pd.DataFrame, threshold = 10):
    peak = df['price'].max()
    bottom = df['price'].min()
    dip = 100 - (bottom/peak)*100
    if dip > threshold:
        return True
    else:
        return False
```

Prefect Parameters

- Parameters are special tasks that receive user inputs

doge_with_prefect.py

```
@task(max_retries = 3,
retry_delay=timedelta(minutes= 1))
def get_data(coin="DOGE") -> pd.DataFrame:
    prices = re.get(format_url(coin))
    prices = prices.json()[ 'data']['entries']
    data = pd.DataFrame(prices, columns=[ "time",
"price"])
    return data

with Flow("to-the-moon") as flow:
    coin = Parameter( "coin", default="DOGE")
    data = get_data( coin)
    is_dip = detect_dip(data)
    post_to_slack()

flow.run()
```

Prefect Parameters

- We can also have ad-hoc runs with Parameters

doge_with_prefect.py

```
@task(max_retries = 3,
retry_delay=timedelta(minutes= 1))
def get_data(coin="DOGE") -> pd.DataFrame:
    prices = re.get(format_url(coin))
    prices = prices.json()['data']['entries']
    data = pd.DataFrame(prices, columns=[ "time",
"price"])
    return data

with Flow("to-the-moon") as flow:
    coin = Parameter( "coin", default="DOGE")
    data = get_data(coin)
    is_dip = detect_dip(data)
    post_to_slack()

flow.run(parameters=dict(coin= "BTC"))
```

Posting to Slack

- Easier to do with Prefect

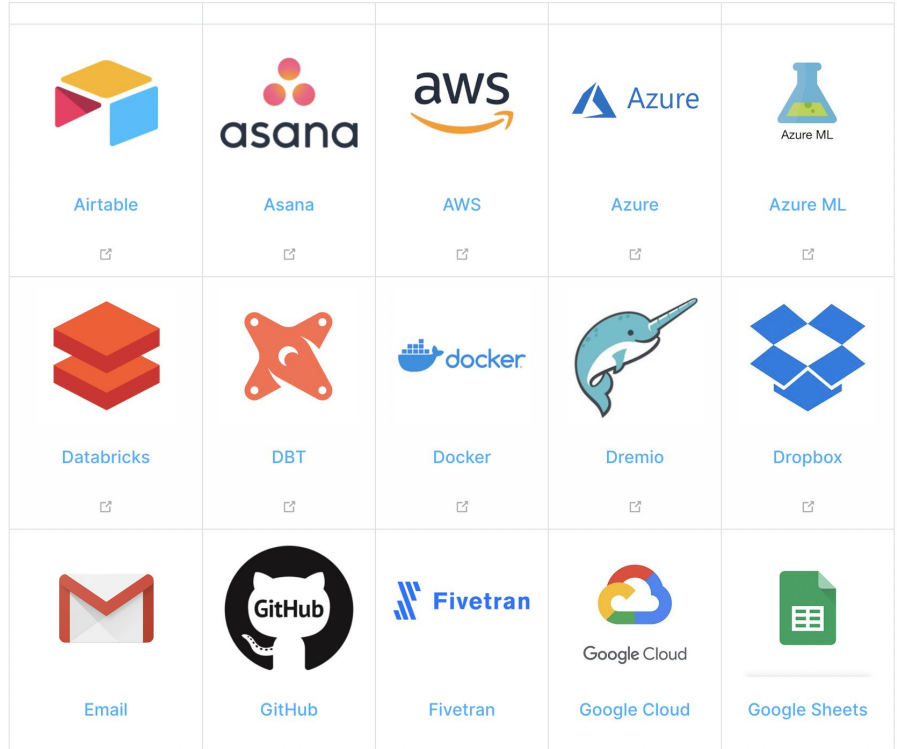
doge_with_prefect.py

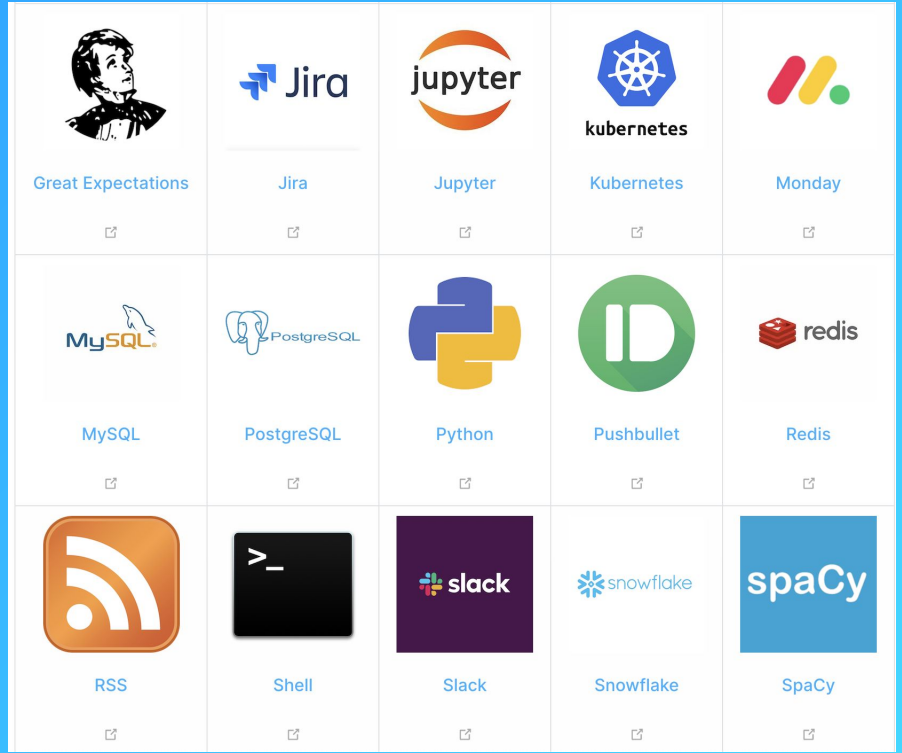
```
@task(max_retries = 3,
retry_delay=timedelta(minutes= 1))
def get_data(coin="DOGE") -> pd.DataFrame:
    prices = re.get(format_url(coin))
    prices = prices.json()['data']['entries']
    data = pd.DataFrame(prices, columns=[ "time",
"price"])
    return data

with Flow("to-the-moon") as flow:
    coin = Parameter( "coin", default="DOGE")
    data = get_data(coin)
    is_dip = detect_dip(data)
    post_to_slack()

flow.run()
```

Task Library





Posting to Slack

- Use the Task library

doge_with_prefect.py

```
@task(max_retries = 3,
retry_delay=timedelta(minutes= 1))
def get_data(coin="DOGE") -> pd.DataFrame:
    prices = re.get(format_url(coin))
    prices = prices.json()[ 'data']['entries']
    data = pd.DataFrame(prices, columns=[ "time",
"price"])
    return data


from prefect.tasks.notifications import SlackTask
post_to_slack = SlackTask(
            message="There has been a dip in DOGE
price.",
            webhook_secret="SLACK_WEBHOOK_URL")
with Flow("to-the-moon") as flow:
    coin = Parameter( "coin", default="DOGE")
    data = get_data(coin)
    is_dip = detect_dip(data)
```

Secrets

- Store secret values in the Prefect Cloud

doge_with_prefect.py

```
@task(max_retries = 3,
retry_delay=timedelta(minutes= 1))
def get_data(coin="DOGE") -> pd.DataFrame:
    prices = re.get(format_url(coin))
    prices = prices.json()[ 'data']['entries']
    data = pd.DataFrame(prices, columns=[ "time",
"price"])
    return data

from prefect.tasks.notifications import SlackTask
post_to_slack = SlackTask(
          message="There has been a dip in DOGE
price.",
          webhook_secret="SLACK_WEBHOOK_URL")
with Flow("to-the-moon") as flow:
    coin = Parameter( "coin", default="DOGE")
    data = get_data(coin)
    is_dip = detect_dip(data)
```

Conditional Logic

- The case task here serves as an if-else

doge_with_prefect.py

```
while True:
    time.sleep(300)
    data = get_data()
    is_dip = detect_dip(data)
    if is_dip:
        post_to_slack()

with Flow("to-the-moon") as flow:
    coin = Parameter("coin", default="DOGE")
    data = get_data(coin)
    is_dip = detect_dip(data)
    with case(is_dip, True):
        post_to_slack()

flow.run()
```

Register

- Register the Flow to Prefect Cloud

doge_with_prefect.py

```
with Flow("to-the-moon") as flow:
    coin = Parameter("coin", default="DOGE")
    data = get_data(coin)
    is_dip = detect_dip(data)
    with case(is_dip, True):
        post_to_slack()

flow.register("tutorial")
```

Schedules

- CronClock
- IntervalClock

- Clocks
- Filters
- Adjustments

doge_with_prefect.py

```
now = datetime.datetime.utcnow()

clock1 = clocks.IntervalClock(start_date=now,
                              interval=timedelta(minutes= 5),
                              parameter_defaults={ "coin": "DOGE"})

schedule = Schedule(clocks=[clock1])



with Flow("to-the-moon") as flow:
    coin = Parameter( "coin", default="DOGE")
    data = get_data(coin)
    is_dip = detect_dip(data)
    with case(is_dip, True):
        post_to_slack()

flow.schedule = schedule
flow.register("tutorial")
```

PREFECT

- Flows and Tasks
- Parameters
- Retries
- Task Library
- Secrets
- Conditional Logic
- Register
- Scheduling

Prefect Mapping

- We can use mapping for parallel execution
- Depth first execution

doge_with_prefect.py

```
@task
def get_data(coin="DOGE") -> pd.DataFrame:
    prices = re.get(format_url(coin))
    prices = prices.json()['data']['entries']
    data = pd.DataFrame(prices, columns=["time",
"price"])
    return data

with Flow("to-the-moon") as flow:
    coin = Parameter("coin", default=["DOGE", "BTC",
"ETH"])
    data = get_data.map(coin)
    is_dip = detect_dip.map(data)
    dip = task(lambda x: max(x))(is_dip)
    with case(dip, True):
        post_to_slack()
```

Dask Executor

- The Dask engine will be used for parallel execution

doge_with_prefect.py

```
with Flow("to-the-moon") as flow:
    coin = Parameter("coin", default=["DOGE", "BTC",
"ETH"])
    data = get_data.map(coin)
    is_dip = detect_dip.map(data)
    dip = task(lambda x: max(x))(is_dip)
    with case(dip, True):
        post_to_slack()

flow.executor = LocalDaskExecutor()
flow.run()
```

Coiled Executor

- Create a software environment with Coiled
- The DaskExecutor will spin up the cluster and spin it down

Coiled

doge_with_prefect.py

```
import coiled

coiled.create_software_environment(
   name="prefect",
   conda={"channels": ["conda-forge"],
          "dependencies": ["python=3.8.0",
"dask=2021.04.0", "distributed=2021.04.0", "prefect"]},
   )

executor = DaskExecutor(
   cluster_class=coiled.Cluster,
   cluster_kwargs={
       "software": "kvnkho/prefect",
       "shutdown_on_close": True,
       "name": "prefect-cluster",
   },
)
flow.executor = executor
```

PREFECT

- Workflow Orchestration
- Eliminate negative engineering
- Prefect
  - Flows and Tasks
  - Parameters
  - Mapping
  - Execution on a Dask Cluster
- Prefect Cloud

Git repo: https://github.com/PrefectHQ/prefect

Slack: https://prefect.io/slack

Prefect: https://www.prefect.io/

Careers

- https://www.prefect.io/about/company/#careers