

ALL ABOUT DEFERRABLES



ANDREW GODWIN // [@andrewgodwin](https://twitter.com/@andrewgodwin)

The background of the slide features a dark, star-filled night sky. In the foreground, the silhouettes of several tall evergreen trees are visible against the dark background.

Hi, I'm

Andrew Godwin

- Principal Engineer at **ASTRONOMER**
- Primary author of Deferrable Operators core
- Somehow at 15 years of writing Python



Why Deferrables?

Did we really need a whole new concept?

How They Work

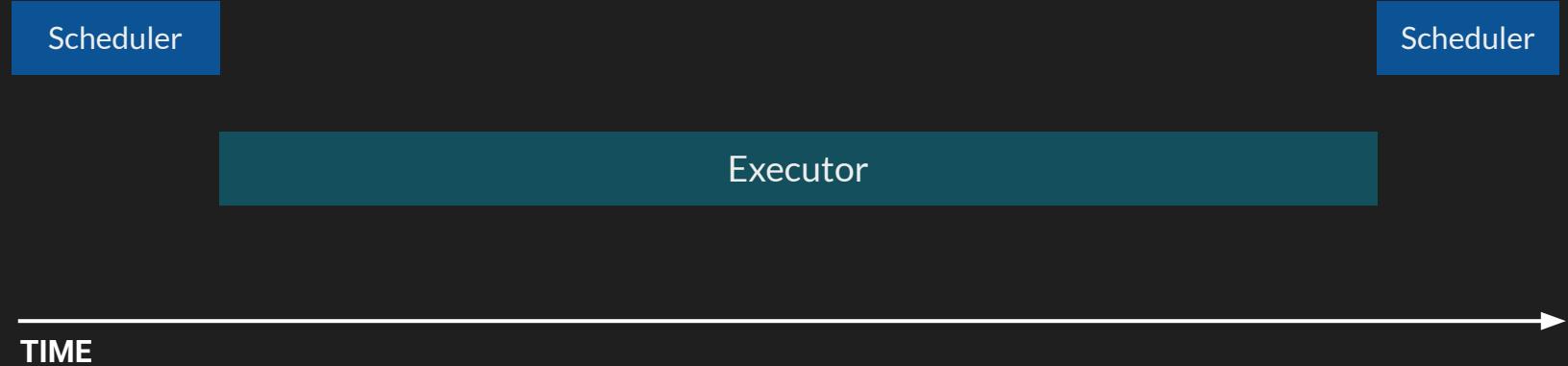
The answer is not just "Very Well, Thank You"

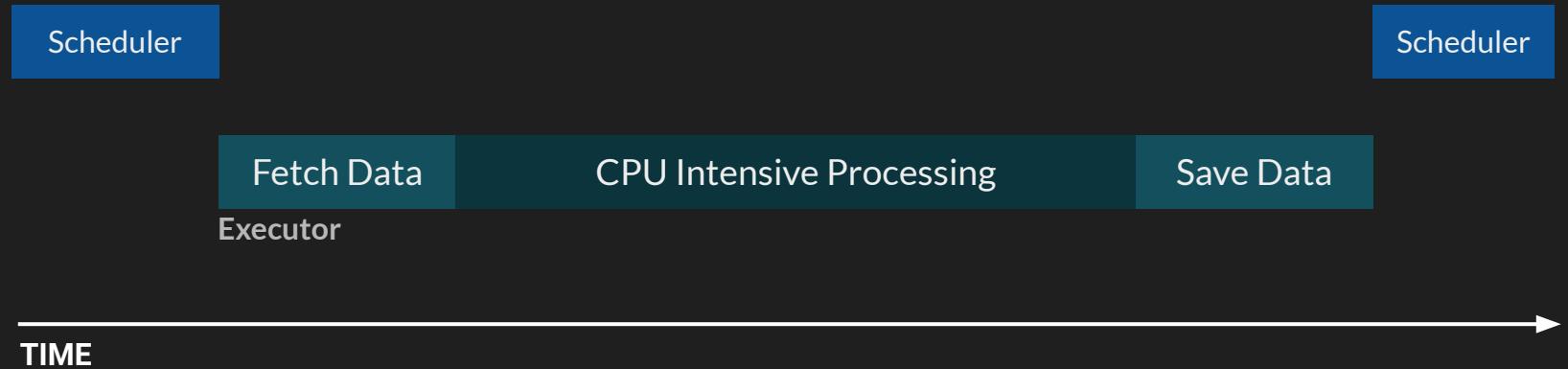
What's next?

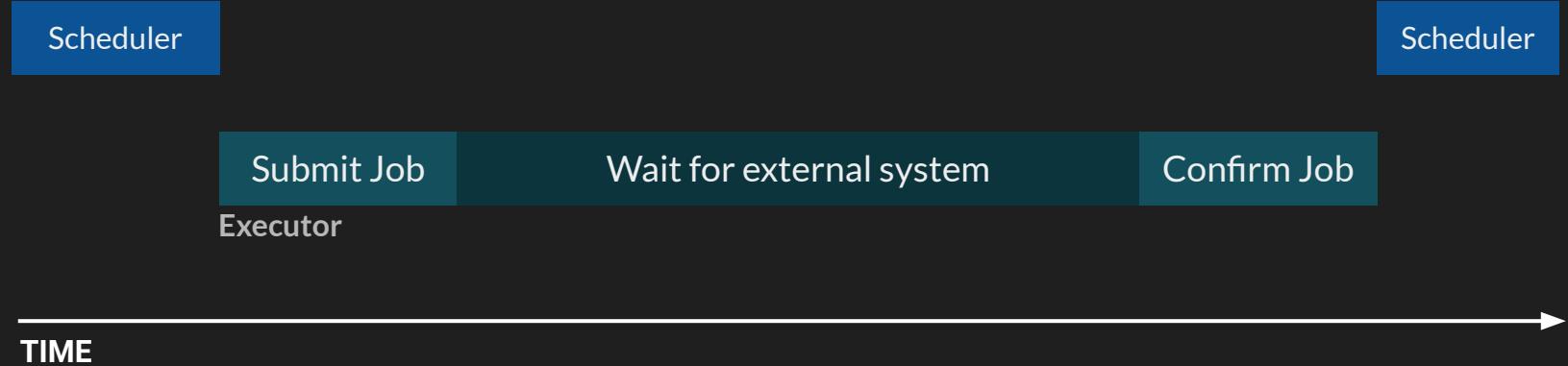
There's a lot more we can do with this

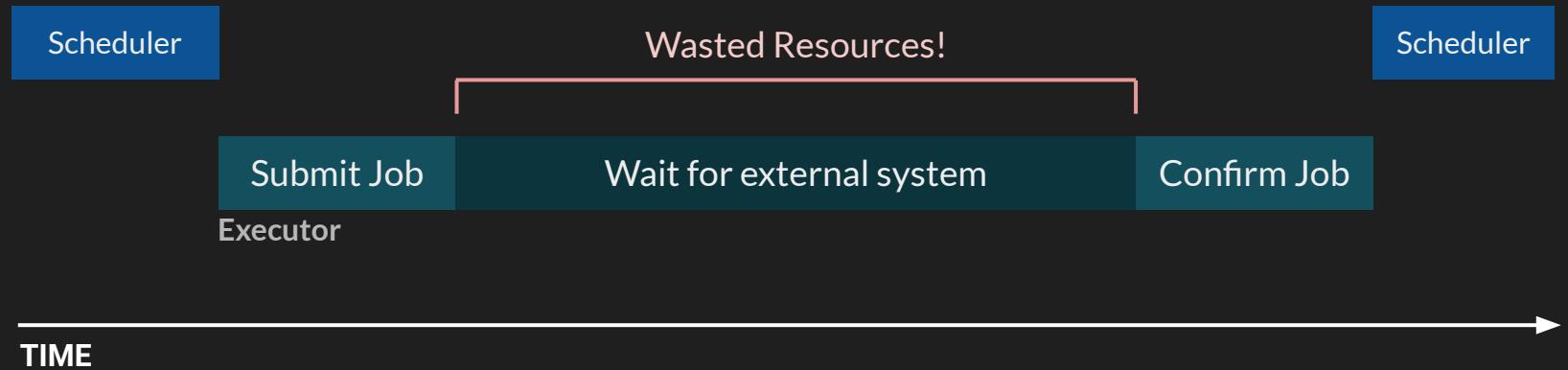
Why defer?

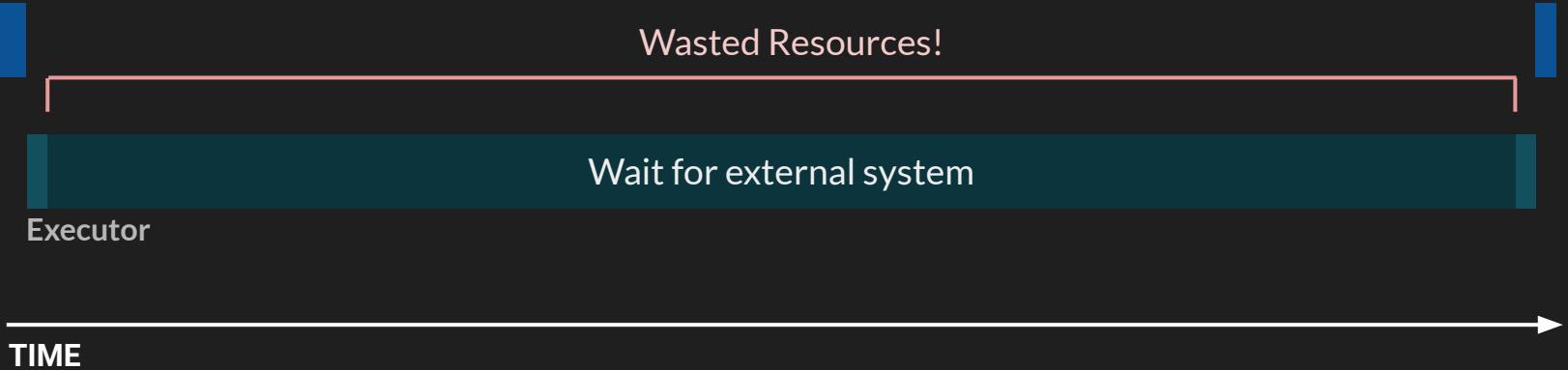
And what is deferring anyway?











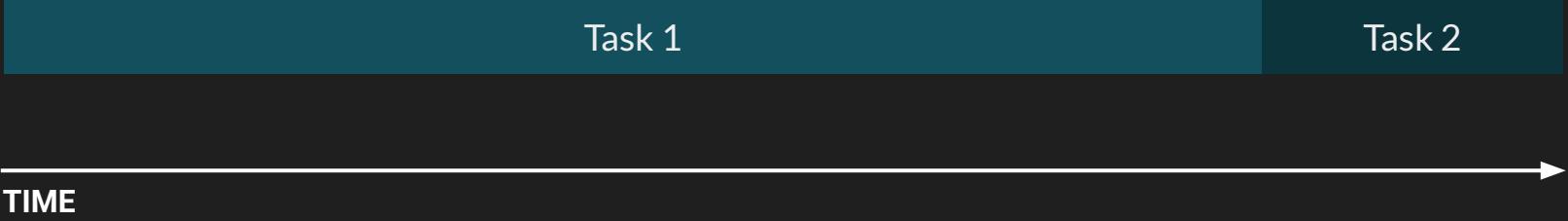
Opportunity Cost

A wasted slot on Celery, wasted reservation on Kubernetes



The Triggerer is asynchronous

It can run thousands of triggers at once!



A Gantt chart illustrating task scheduling over time. A horizontal axis at the bottom is labeled "TIME" and features a right-pointing arrow. Above the axis, two horizontal bars represent tasks. The first task, "Task 1", is a teal bar spanning from the start of the timeline to approximately the midpoint. The second task, "Task 2", is a teal bar starting immediately after Task 1 ends and extending to the end of the timeline. Both tasks are labeled with their respective names in white text.

Task 1

Task 2

TIME



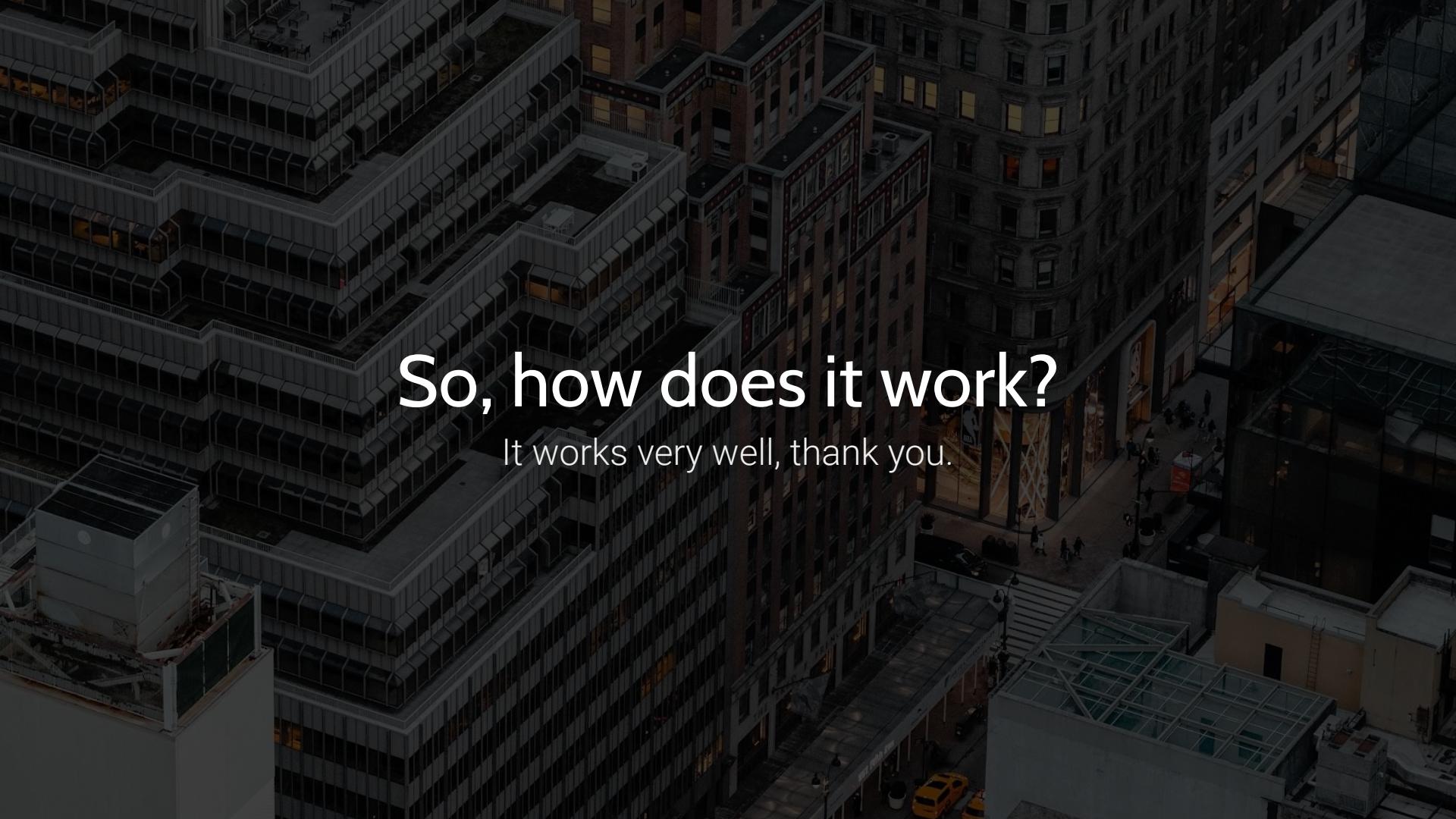
TIME

push button and wait
for signal opposite



The longer the wait, the better the saving

We've seen over a 90% reduction in resources for a 10 min wait

The background of the slide is a high-angle aerial photograph of a city street. It features several modern buildings with glass facades and a mix of older brick structures. A street runs diagonally from the bottom left towards the top right. People are visible walking on the sidewalks, and a few cars are on the street. The overall scene is a typical urban environment.

So, how does it work?

It works very well, thank you.

You hand off to a Trigger

This is a new class of workloads alongside Operators



More Restrictions

Triggers are more limited than Operators so they can be efficient

Must be asynchronous

So we can run thousands per CPU core

No persistent state

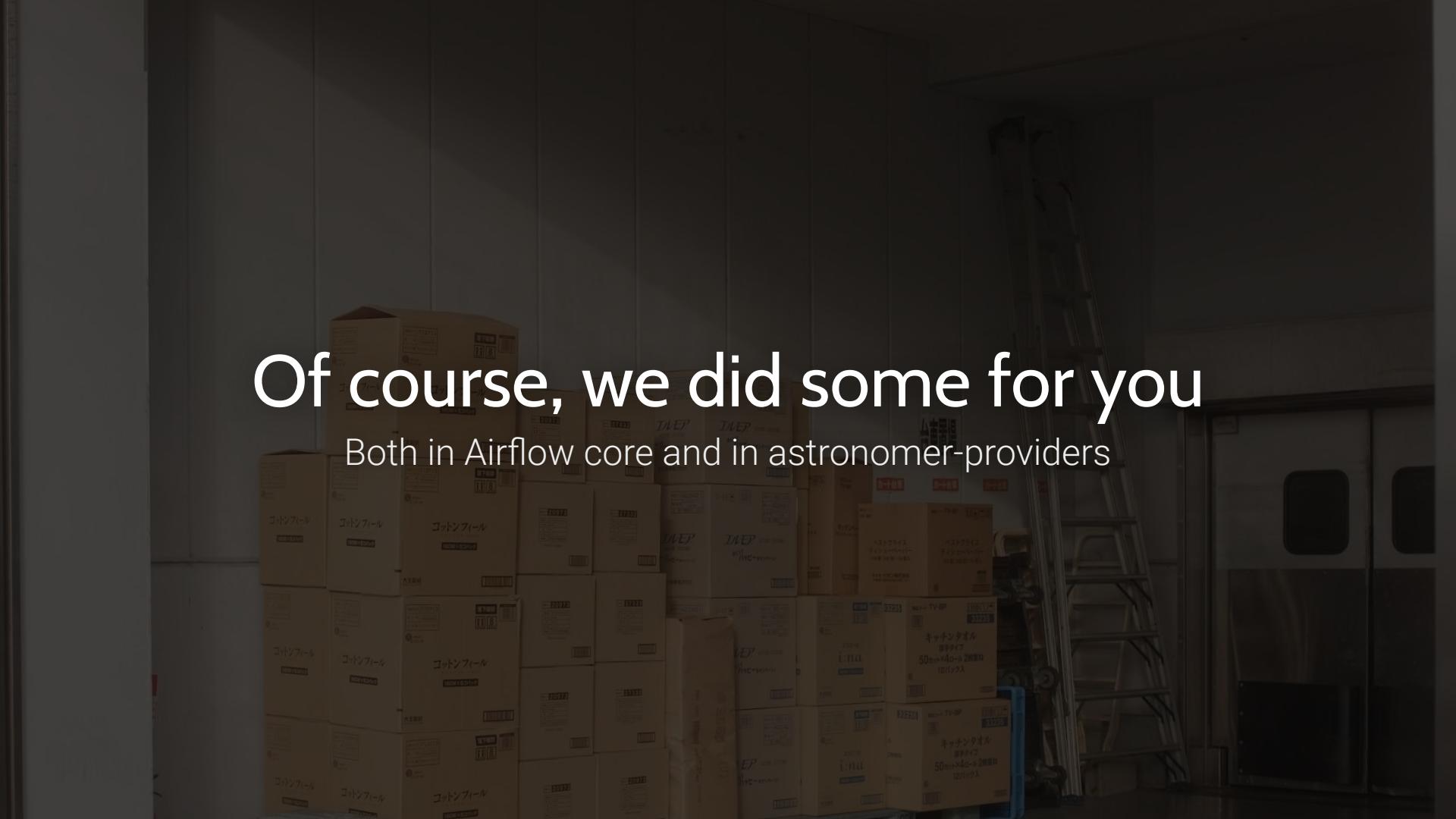
So we can shuttle them around between Triggerers as needed

Must support multiple copies

For reliability during network partitions

```
class DateTimeTrigger(BaseTrigger):  
  
    def __init__(self, moment: datetime.datetime):  
        super().__init__()  
        self.moment = moment  
  
    def serialize(self):  
        return ("mymodule.DateTimeTrigger", {"moment": self.moment})  
  
    @async def run(self):  
        while self.moment > timezone.utcnow():  
            await asyncio.sleep(1)  
            yield TriggerEvent(self.moment)
```

```
class WaitOneHourSensor(BaseSensorOperator):  
  
    def execute(self, context):  
        self.defer(  
            trigger=TimeDeltaTrigger(timedelta(hours=1)),  
            method_name="execute_complete",  
        )  
  
    def execute_complete(self, context, event=None):  
        # We have no more work to do here. Mark as complete.  
        return
```



Of course, we did some for you

Both in Airflow core and in astronomer-providers

Not everything can be deferred

It must be an external event/system with a portable identifier

What's next?

Turns out, Triggers are generally useful

More operator support for deferring

There's not a lot of reasons *not* to use it

Triggers for DAGs

Will likely play into the new Dataset work

Expanding async workload support

The triggerer should *really* be part of the Executor contract

Making more of Airflow `async`

It's not just the operators that sit there and idle a lot

The background image shows a wide, green valley surrounded by dark, forested mountains. A paved stone path leads from the foreground towards the center of the valley. The sky is overcast.

Airflow is forged by people like you.

Want to help with any of this? Get in touch!

Thanks.

Andrew Godwin

@andrewgodwin

andrew.godwin@astronomer.io