The Executor class

Part of the Globus Compute SDK

Preferred approach to submitting tasks and collecting results

Subclass of `concurrent.futures.Executor`

- Typically used as a context manager
- `.submit()` returns a `Future`
- Automatically wait for results

```python
from globus_compute_sdk import Executor

with Executor(endpoint_id="...") as gce:
    fut = gce.submit(func, *args, **kwargs)
    print(fut.result())
```
The ComputeFuture class

Subclass of `concurrent.futures.Future`

- `.result()` waits until upstream services return the result
- `.done()` returns a boolean indicating whether the result is ready

What’s the difference? `ComputeFuture` objects are associated with tasks.

```python
# Blocking
print(f"Result: {fut.result()}")

# Non-blocking
print(f"Status: {fut.done()}")

>>> print(fut.task_id)
3223b5fd-6fe4-4c78-bfe4-ec0950048559
```
A peak under the hood

Globus Compute Services

1. Compute Executor
   - Submit tasks
2. Web Service
   - Enqueue tasks
3. Compute Endpoint
   - Consume tasks
4. TG Results Queue
   - Enqueue results
5. EP Task Queue
   - Consume results
An example workflow

Create some functions

Submit tasks to an endpoint running on Polaris (ALCF)

Submit tasks to an endpoint running on Midway (UChicago), using the former results as arguments
Any questions?

GitHub: [https://github.com/funcx-faas/funcX](https://github.com/funcx-faas/funcX)
Slack: [https://funcx.slack.com/](https://funcx.slack.com/)