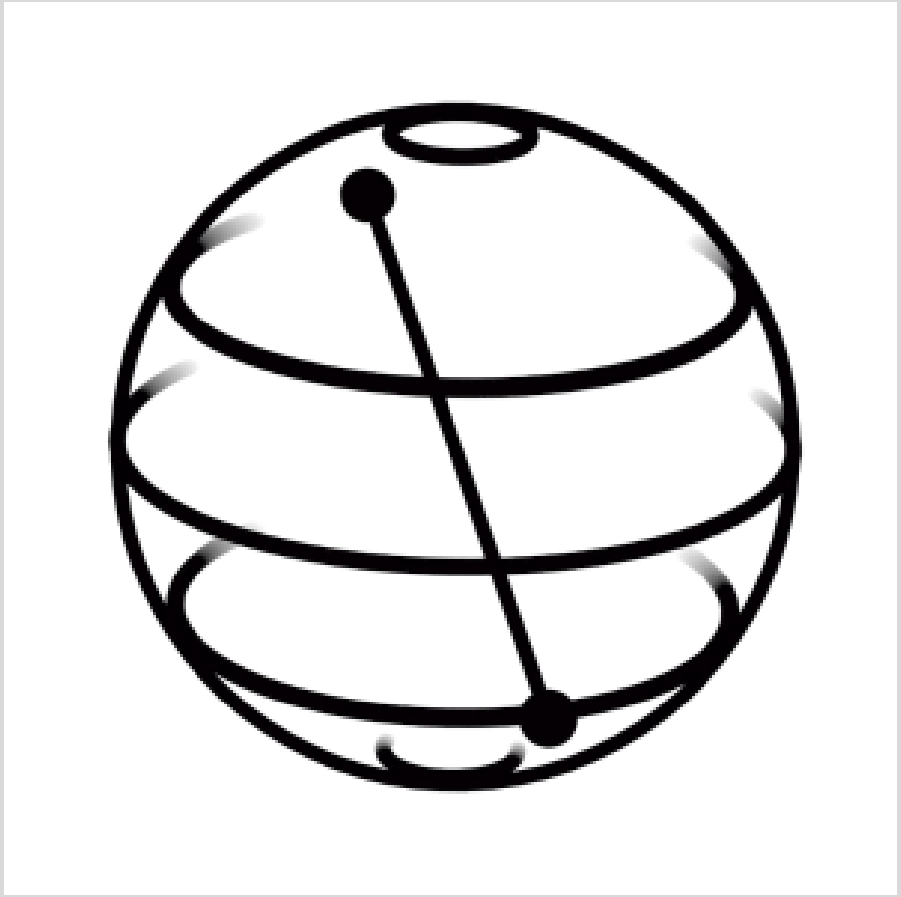

Using the IBMQ Provider

G. Byrd - PPoPP Tutorial - Feb 17, 2019

IBMQ Provider

- provider: accesses backends and provides backend objects
- backend: runs the quantum circuit
- job: keeps track of the submitted job

IBMQ provides access to the following public backends

- **ibmqx4** -- 5-qubit machine
- **ibmq_16_melbourne** -- 16-qubit hardware
- **ibmq_qasm_simulator** -- cloud-based simulator

First Step: Set up account

```
from qiskit import IBMQ


IBMQ.save_account('...')  # paste your IBM-Q API token
# saves in a configuration file -- only need to do this once

IBMQ.load_accounts()    # enables saved accounts
```

Second Step: Build Circuit

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit


q = QuantumRegister(2)
c = ClassicalRegister(2)
qc = QuantumCircuit(q, c)

qc.h(q[1])         # Hadamard on first qubit
qc.cx(q[1],q[0])   # CNOT to entangle


qc.measure(q,c)
```

Third Step: Submit job

```
from qiskit import IBMQ, execute


IBMQ.backends()   # list of available backend objects
[<IBMQBackend('ibmqx4') from IBMQ()>, <IBMQBackend('ibmq_16_melbourne')
from IBMQ()>, <IBMQBackend('ibmq_qasm_simulator') from IBMQ()>]
backend = IBMQ.get_backend('ibmqx4')


job = execute(qc, backend, shots=512) # compile and run
# defaults shots = 1024
```

Fourth Step: Monitor, get results

```
from qiskit.tools.monitor import job_monitor


job_monitor(job)  # loop to show status of job
# not necessary...


result = job.result()  # waits for job to finish
print(result.get_counts())
```