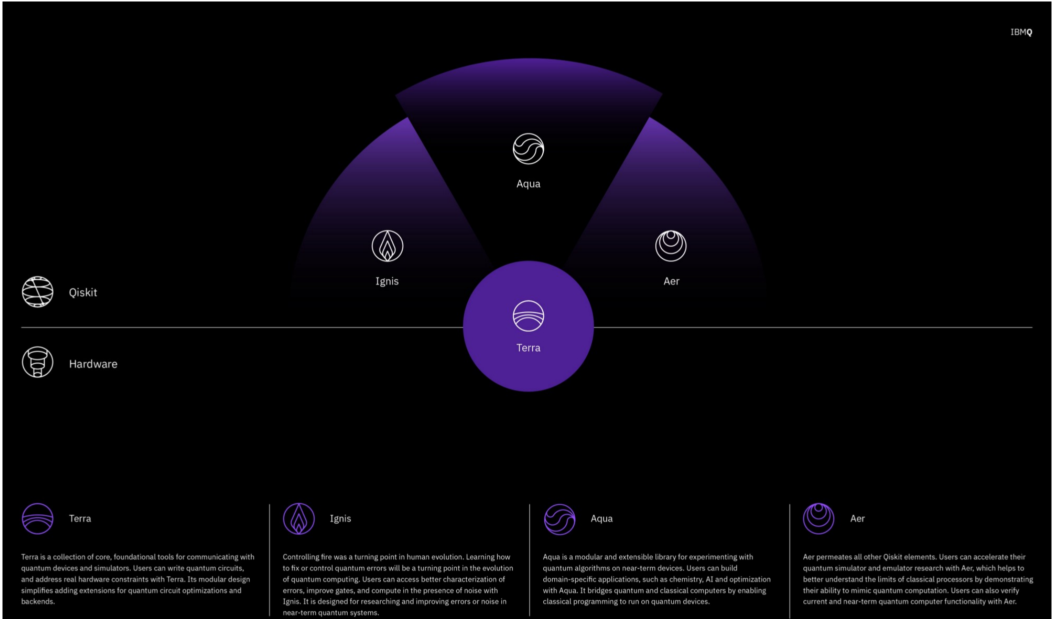
What’s new in Qiskit 2021

Matthew Treinish

Oct 9, 2021

The Elements



Terra

Terra is a collection of core, foundational tools for communicating with quantum devices and simulators. Users can write quantum circuits, and address real hardware constraints with Terra. Its modular design simplifies adding extensions for quantum circuit optimizations and backends.

Ignis

Controlling fire was a turning point in human evolution. Learning how to fix or control quantum errors will be a turning point in the evolution of quantum computing. Users can access better characterization of errors, improve gates, and compute in the presence of noise with Ignis. It is designed for researching and improving errors or noise in near-term quantum systems.

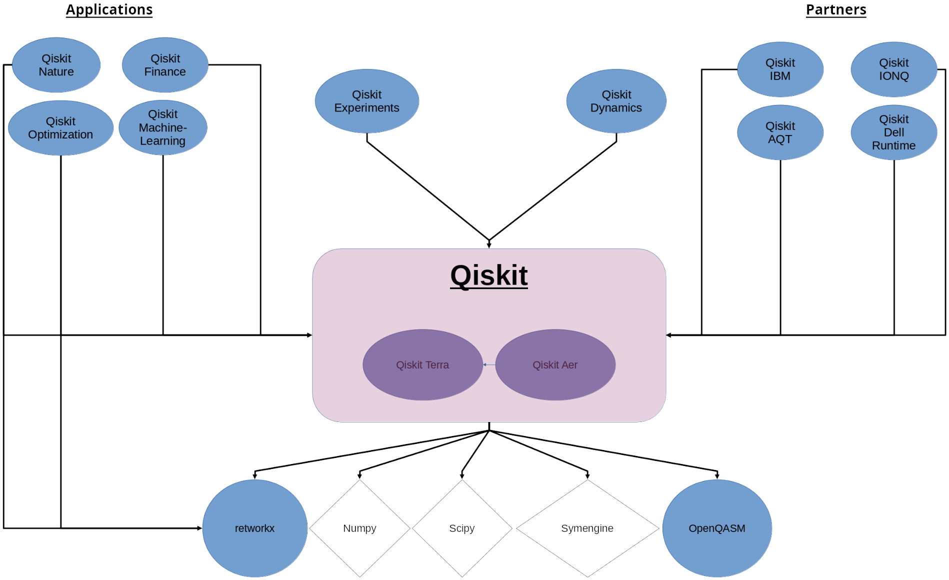
Aqua

Aqua is a modular and extensible library for experimenting with quantum algorithms on near-term devices. Users can build domain-specific applications, such as chemistry, AI and optimization with Aqua. It bridges quantum and classical computers by enabling classical programming to run on quantum devices.

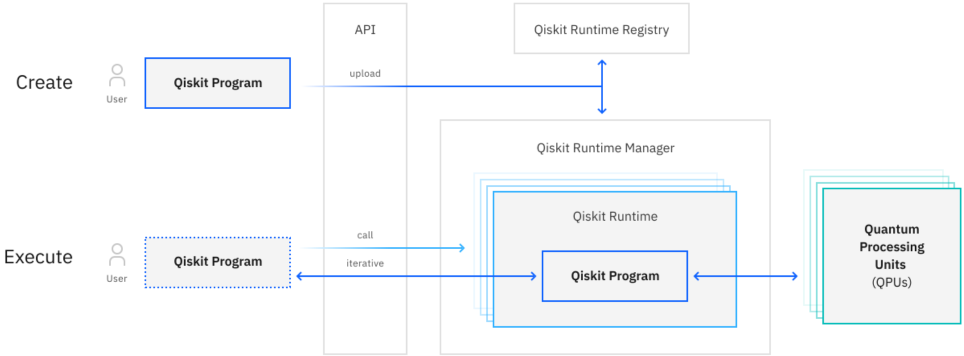
Aer

Aer permeates all other Qiskit elements. Users can accelerate their quantum simulator and emulator research with Aer, which helps to better understand the limits of classical processors by demonstrating their ability to mimic quantum computation. Users can also verify current and near-term quantum computer functionality with Aer.

Qiskit Moving Foward



Qiskit Runtime



QPY Serialization

- Qiskit native binary serialization format for QuantumCircuit
- Designed to be backwards compatible moving forward (new Qiskit can always load old QPY files)
-

```python
import gzip
from qiskit.circuit.library import EfficientSU2
from qiskit.circuit import QuantumCircuit
from qiskit.circuit import Parameter

from qiskit.circuit.qpy_serialization import load, dump

gamma = Parameter("$\\gamma$")
qc = QuantumCircuit(4)
for i in range(4):
    qc.h(i)
qc.append(EfficientSU2(4), [0, 1, 2, 3])
for pair in [(0, 1), (1, 2), (2, 3), (3, 0)]:
    qc.rzz(2 * gamma, *pair)
    qc.barrier()
qc.measure_all()

with gzip.open("test.qpy.gz", "wb") as qpy_file:
    dump(qc, qpy_file)

with gzip.open("test.qpy.gz", "rb") as qpy_file:
    loaded_circuit = load(qpy_file)[0]
```

Improving the interface to backends

- Making continual improvements to Qiskit's interface with backends
- Now designed to be hardware and vendor agnostic to enable
-

```python
from qiskit.circuit.library import QuantumVolume
from qiskit.compiler import transpile

from qiskit.test.mock import FakeMontreal

backend = FakeMontreal()
backend.set_options(method="stabilizer")

qc = QuantumVolume(10)
qc.measure_all()
tqc = transpile(
    qc, backend, optimization_level=3, routing_method="sabre",
    layout_method="sabre"
)
results = backend.run(tqc, shots=1e6).result()
print(results.get_counts())
```

Control Flow

- The next Qiskit release will add support for basic control flow
-

QASM3

- Working on making QASM3 integrated with Qiskit
- Starting with support for exporting OpenQASM3 from a QuantumCircuit in the next Qiskit release
- Moving forward support for parsing OpenQASM3 into a QuantumCircuit object will be added in the future