# Qiskit Quantum Software Stack

Junye Huang
Quantum Developer Advocate

# Outline

Qiskit

## What is Qiskit?

## What can you use Qiskit for?

# Qiskit SDK Architecture

**High level applications**

### Qiskit Nature
For applications relating to simulating quantum mechanical systems and natural phenomena.

### Qiskit Finance
For applications relating to financial modeling.

### Qiskit Optimization
For applications relating to optimization problems.

### Qiskit Machine Learning
For applications relating to machine learning.

**Low level applications**

Qiskit

### Qiskit Metal
For designing quantum hardware and processors.

### Qiskit Dynamics
For building, transforming, and solving time-dependent models of quantum systems.

### Qiskit Experiments
For running quantum experiments with a library of characterization, calibration, and verification experiments.

**Core Capabilities**

### Qiskit Terra
For building and transforming quantum circuits and operators at the level of gates or pulses.

**Simulator**

### Qiskit Aer
For simulating quantum circuits on classical hardware.

**Hardware providers**

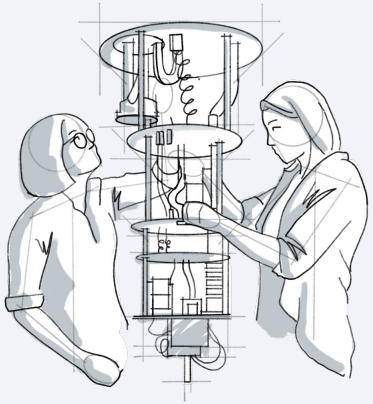
### IBM
IBM Quantum systems

### AQT
AQT systems

### IonQ
IonQ systems

Qiskit can connect to many other systems

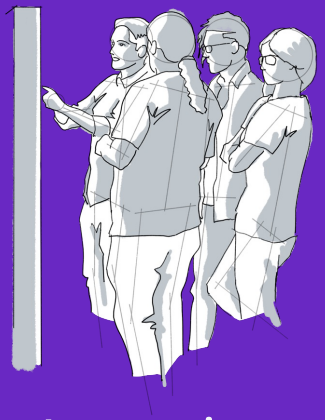
# Qiskit for



Research

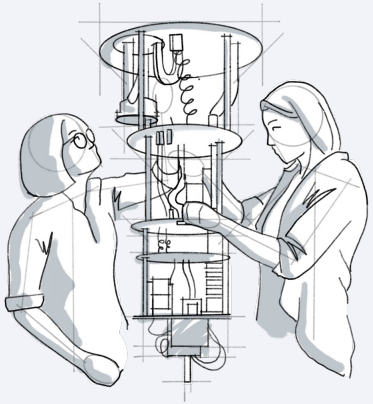
```
from qiskit import QuantumCircuit, execute
from qiskit import Aer, IBMQ
from qiskit.providers.aer.noise import NoiseModel

# Choose a real device to simulate from IBMQ provider
provider = IBMQ.load_account()
backend = provider.get_backend('ibmq_vigo')
coupling_map = backend.configuration().coupling_map

# Generate an Aer noise model for device
noise_model = NoiseModel.from_backend(backend)
basis_gates = noise_model.basis_gates

# Generate 3-qubit GHZ state
num_qubits = 3
circ = QuantumCircuit(3, 3)
circ.h(0)
circ.cx(0, 1)
circ.cx(1, 2)
circ.measure([0, 1, 2], [0, 1 ,2])

# Perform noisy simulation
backend = Aer.get_backend('qasm_simulator')
job = execute(circ, backend,
              coupling_map=coupling_map,
              noise_model=noise_model,
              basis_gates=basis_gates)
result = job.result()

print(result.get_counts(0))
```

Development

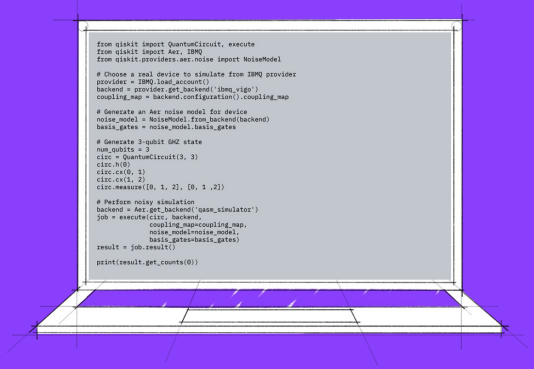


Learning

# Qiskit for

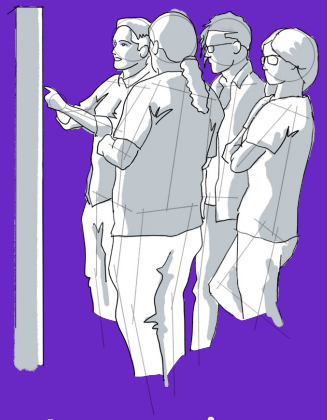


## Research



```
from qiskit import QuantumCircuit, execute
from qiskit import Aer, IBMQ
from qiskit.providers.aer.noise import NoseModel

# Choose a real device to simulate from IBMQ provider
provider = IBMQ.load_account()
backend = provider.get_backend('ibmq_vigo')
coupling_map = backend.configuration().coupling_map

# Generate an Aer noise model for device
noise_model = NoiseModel.from_backend(backend)
basis_gates = noise_model.basis_gates

# Generate 3-qubit GHZ state
num_qubits = 3
circ = QuantumCircuit(3, 3)
circ.h(0)
circ.cx(0, 1)
circ.cx(1, 2)
circ.measure([0, 1, 2], [0, 1 ,2])

# Perform noisy simulation
backend = Aer.get_backend('qasm_simulator')
job = execute(circ, backend,
              coupling_map=coupling_map,
              noise_model=noise_model,
              basis_gates=basis_gates)
result = job.result()

print(result.get_counts(0))
```

## Development

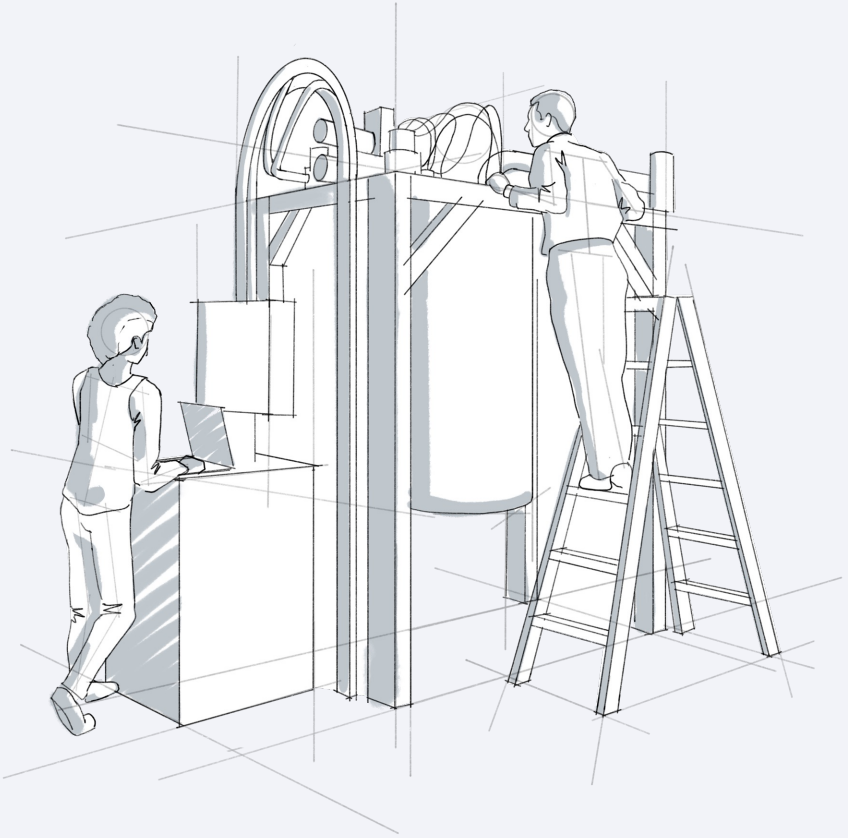


## Learning

Since launching in Mar 2017

Qiskit has enabled

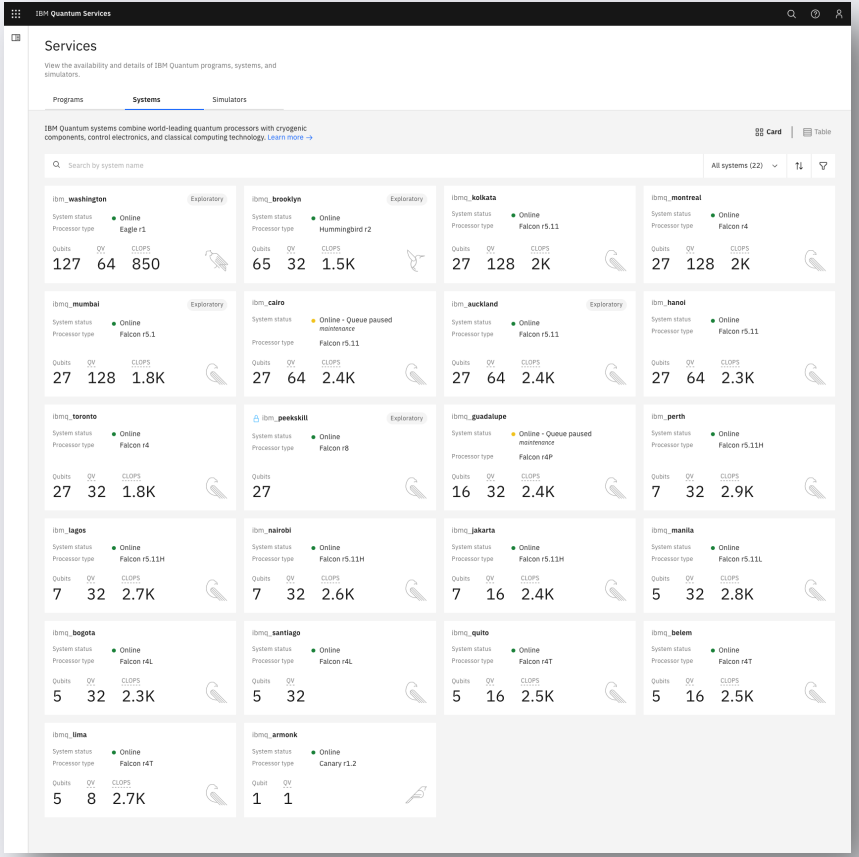
# 1400+ research papers*



*based on the usage of IBM Quantum systems

# Access to a wide range of quantum systems

Qiskit




**Hardware providers**

| IBM | AQT | IonQ | |
|---|---|---|---|
| IBM Quantum systems | AQT systems | IonQ systems | Qiskit can connect to many other systems |

Cross-platform support

- **Superconducting:** IBM Quantum, Rigetti
- **Trapped ions:** AQT, IonQ, Quantinuum
- **Spin qubits:** Quantum Inspire
- **Cold atoms:** Qiskit Cold Atom provider
- **Platform**: Azure Quantum, AWS Braket
- **Simulator**: cuQuantum (NVIDIA)

# Qiskit Pulse for pulse-level control

With pulses, you can dig deep into the heart of a quantum device and study the system as if you were physically present in the lab.

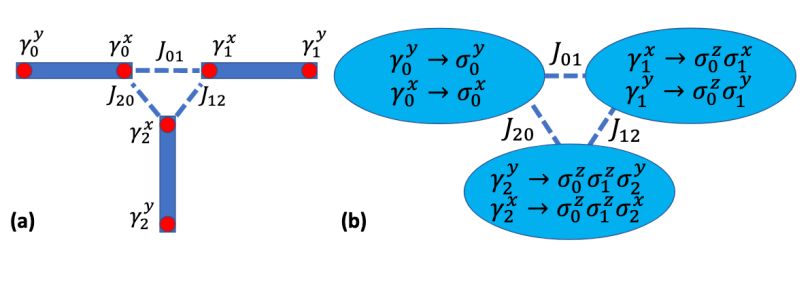
Core Capabilities

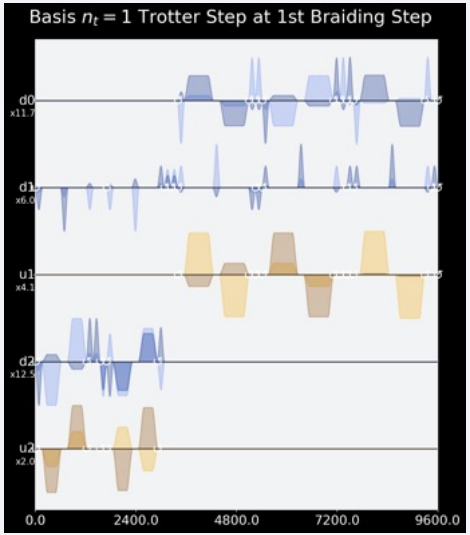
**Qiskit Terra**

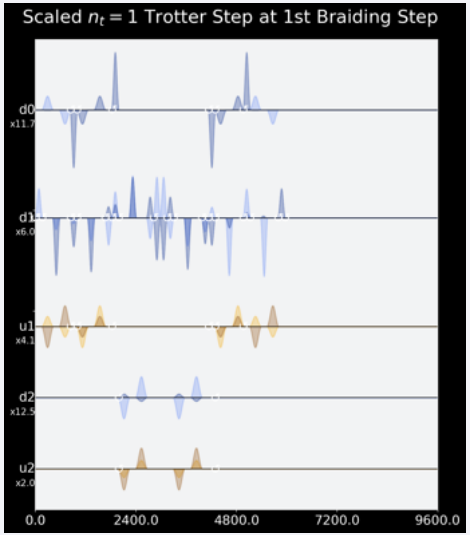
For building and transforming quantum circuits and operators at the level of gates or pulses.

**Highlight**

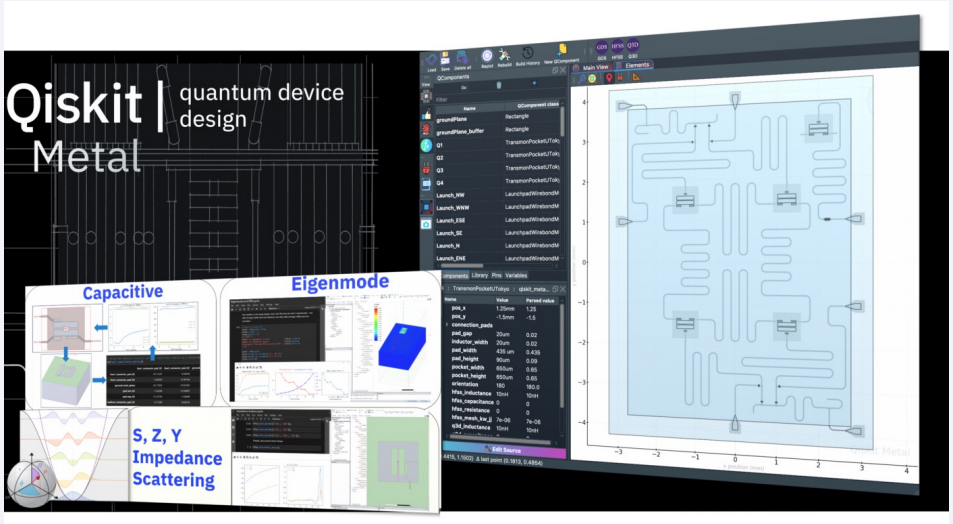
- Simulating the dynamics of braiding of Majorana zero modes using an IBM Quantum computer with Qiskit Pulse [arXiv:2012.11660]
- Qiskit Pulse allowed the authors to overcome the device's noise with specially-crafted controlled gates.



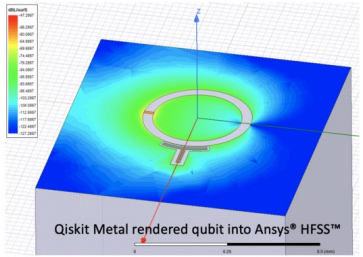
# Qiskit Metal for designing quantum processors



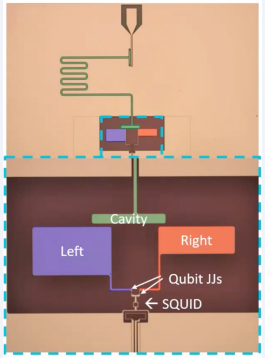

Qiskit Metal is the first EDA tool specifically for quantum computers used for designing, modelling and analyzing performance of quantum processors.

Community driven and experimentally tested by IBM Quantum and the community

Promote more knowledge sharing in the quantum hardware community




An 8-qubit chip designed by C. Warren, Amr Osman, and team (Chalmers) with Qiskit Metal.

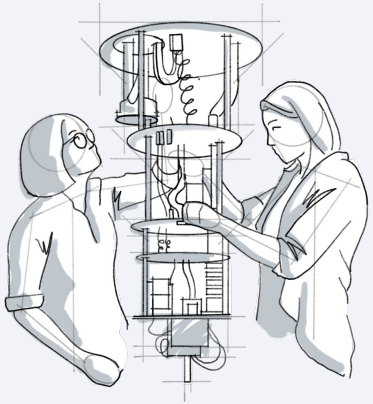



Two transmon qubits with a SQUID coupler designed by Zachary Parrott (UCB & NIST) with Qiskit Metal.

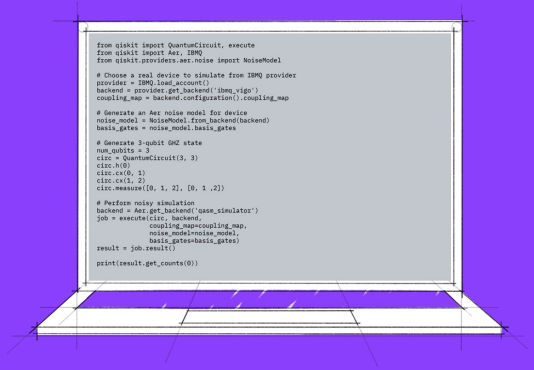
**Qiskit Metal**

For designing quantum hardware and processors.

# Qiskit for

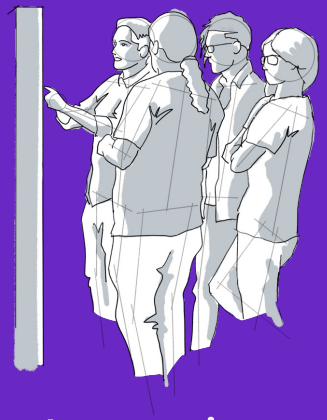


Research



```
from qiskit import QuantumCircuit, execute
from qiskit import Aer, IBMQ
from qiskit.providers.aer.noise import NoseModel

# Choose a real device to simulate from IBMQ provider
provider = IBMQ.load_account()
backend = provider.get_backend('ibmq_vigo')
coupling_map = backend.configuration().coupling_map

# Generate an Aer noise model for device
noise_model = NoiseModel.from_backend(backend)
basis_gates = noise_model.basis_gates

# Generate 3-qubit GHZ state
num_qubits = 3
circ = QuantumCircuit(3, 3)
circ.h(0)
circ.cx(0, 1)
circ.cx(1, 2)
circ.measure([0, 1, 2], [0, 1 ,2])

# Perform noisy simulation
backend = Aer.get_backend('qasm_simulator')
job = execute(circ, backend,
              coupling_map=coupling_map,
              noise_model=noise_model,
              basis_gates=basis_gates)
result = job.result()

print(result.get_counts(0))
```

**Development**



Learning

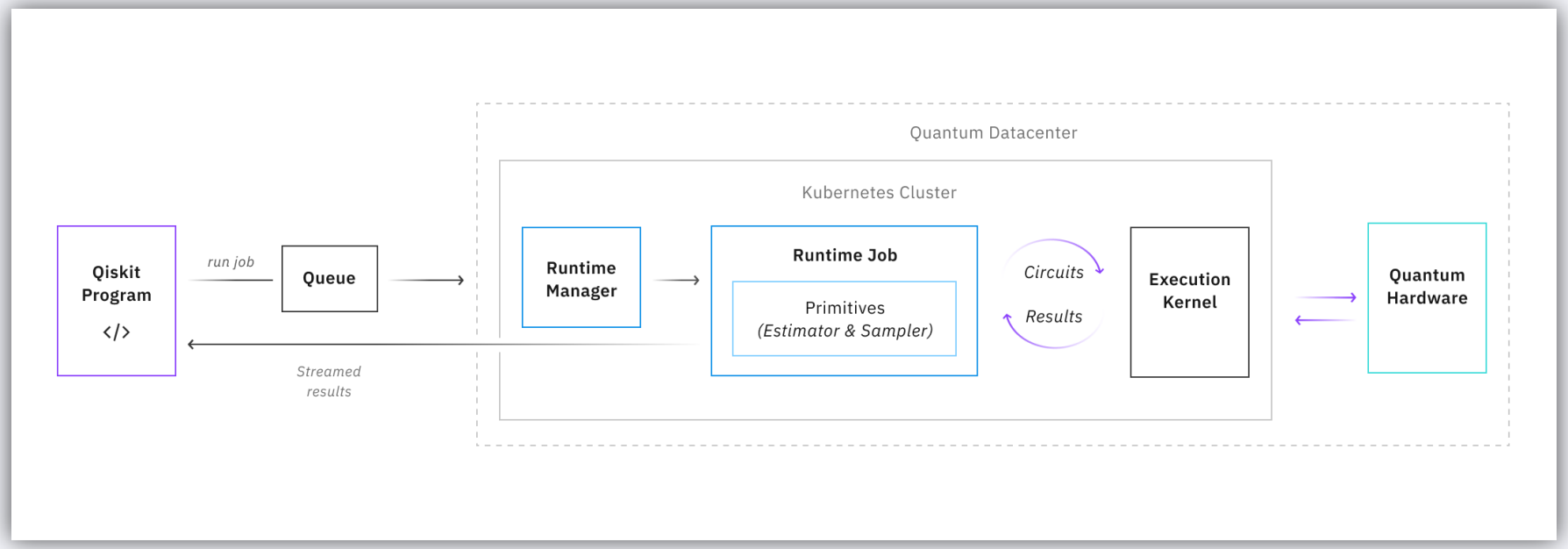
# Development Roadmap

Executed by IBM
On target

IBM **Quantum**

| | 2019 | 2020 | 2021 | 2022 | 2023 | 2024 | 2025 | Beyond 2026 |
|---|---|---|---|---|---|---|---|---|
| | Run quantum circuits on the IBM cloud | Demonstrate and prototype quantum algorithms and applications | Run quantum programs 100x faster with Qiskit Runtime | Bring dynamic circuits to Qiskit Runtime to unlock more computations | Enhancing applications with elastic computing and parallelization of Qiskit Runtime | Improve accuracy of Qiskit Runtime with scalable error mitigation | Scale quantum applications with circuit knitting toolbox controlling Qiskit Runtime | Increase accuracy and speed of quantum workflows with integration of error correction into Qiskit Runtime |
| Model Developers | | | | | Prototype quantum software applications → | | Quantum software applications | |
| | | | | | | | Machine learning \| Natural science \| Optimization | |
| Algorithm Developers | | Quantum algorithm and application modules | | | Quantum Serverless | | | |
| | | Machine learning \| Natural science \| Optimization | | | | Intelligent orchestration | Circuit Knitting Toolbox | Circuit libraries |
| Kernel Developers | Circuits | | Qiskit Runtime | | | | | |
| | | | | Dynamic circuits | Threaded primitives | Error suppression and mitigation | | Error correction |
| System Modularity | Falcon 27 qubits | Hummingbird 65 qubits | Eagle 127 qubits | Osprey 433 qubits | Condor 1,121 qubits | Flamingo 1,386+ qubits | Kookaburra 4,158+ qubits | Scaling to 10K-100K qubits with classical and quantum communication |
| | | | | | Heron 133 qubits x p | Crossbill 408 qubits | | |

# Qiskit Runtime




Qiskit Runtime is a quantum computing service and programming model that allows users to optimize workloads and efficiently execute them on quantum systems at scale. The programming model extends the existing interface in Qiskit with a set of new primitive programs.

# Qiskit application modules

Qiskit

### Qiskit Nature

For applications relating to simulating quantum mechanical systems and natural phenomena.

### Qiskit Finance

For applications relating to financial modeling.

### Qiskit Optimization

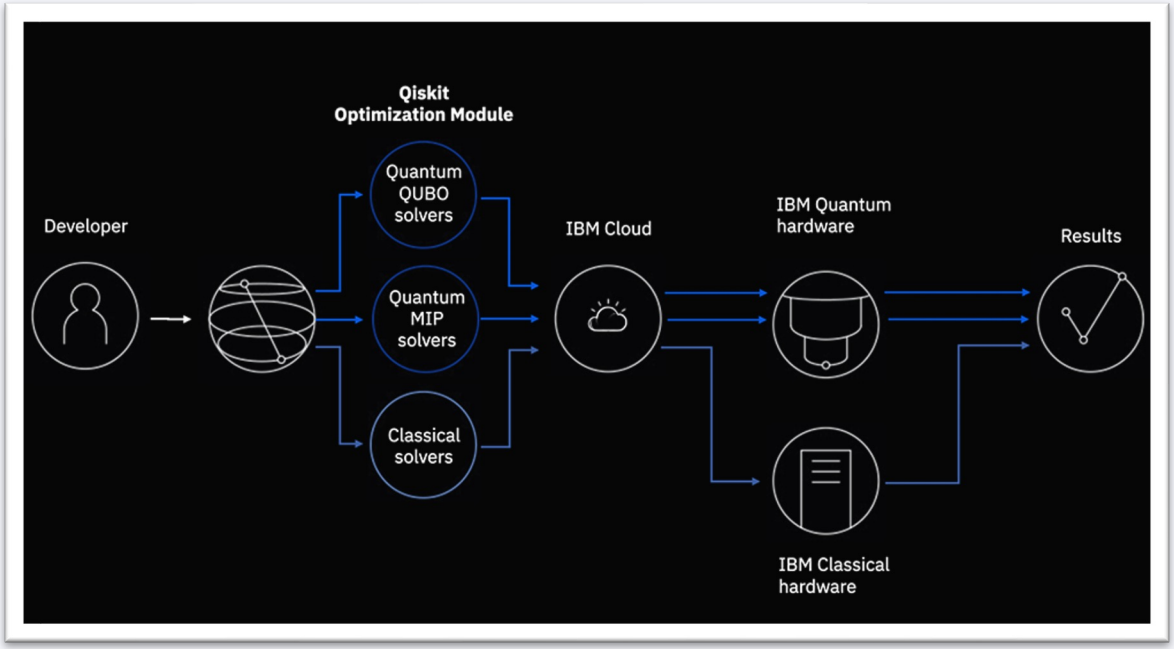
For applications relating to optimization problems.

### Qiskit Machine Learning

For applications relating to machine learning.

**Frictionless development vision:**

a programming environment where the intricacies of the underlying technology are no longer a concern to users



# Qiskit Ecosystem

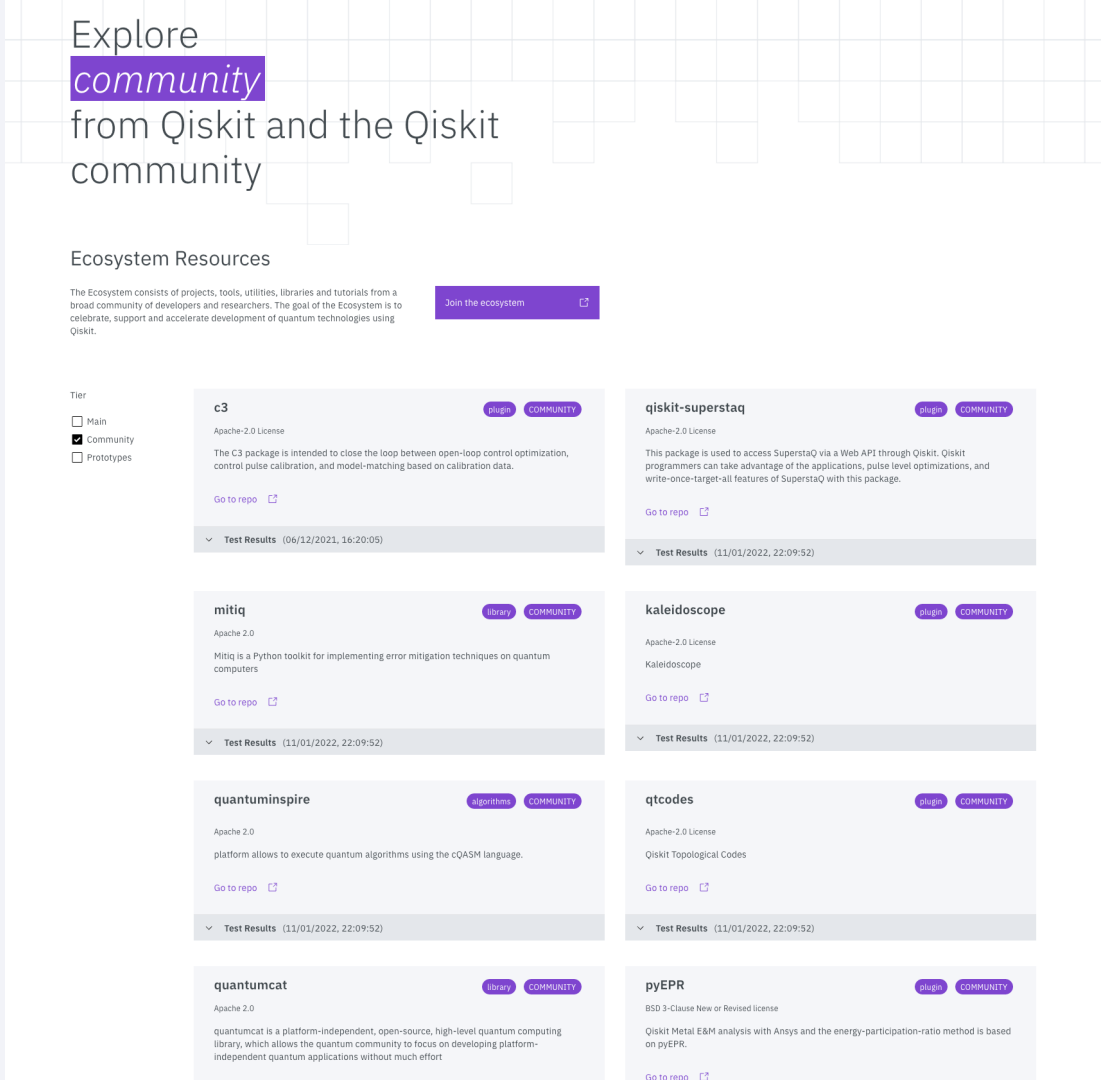
Qiskit

- is open source
- has a culture of openness
- has a vibrant community

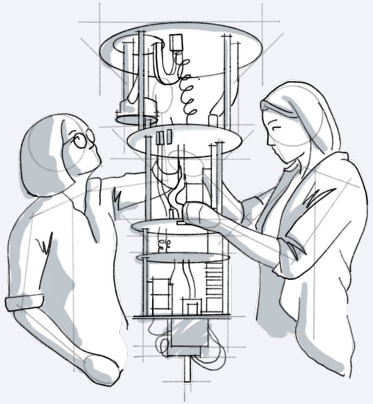
**Qiskit Ecosystem** (qiskit.org/ecosystem)

The Ecosystem consists of projects, tools, utilities, libraries and tutorials from a broad community of developers and researchers.

The goal of the Ecosystem is to celebrate, support and accelerate development of quantum technologies using Qiskit.



# Qiskit for



Research

```
from qiskit import QuantumCircuit, execute
from qiskit import Aer, IBMQ
from qiskit.providers.aer.noise import NoiseModel

# Choose a real device to simulate from IBMQ provider
provider = IBMQ.load_account()
backend = provider.get_backend('ibmq_vigo')
coupling_map = backend.configuration().coupling_map

# Generate an Aer noise model for device
noise_model = NoiseModel.from_backend(backend)
basis_gates = noise_model.basis_gates

# Generate 3-qubit GHZ state
num_qubits = 3
circ = QuantumCircuit(3, 3)
circ.h(0)
circ.cx(0, 1)
circ.cx(1, 2)
circ.measure([0, 1, 2], [0, 1 ,2])

# Perform noisy simulation
backend = Aer.get_backend('qasm_simulator')
job = execute(circ, backend,
              coupling_map=coupling_map,
              noise_model=noise_model,
              basis_gates=basis_gates)
result = job.result()

print(result.get_counts(0))
```

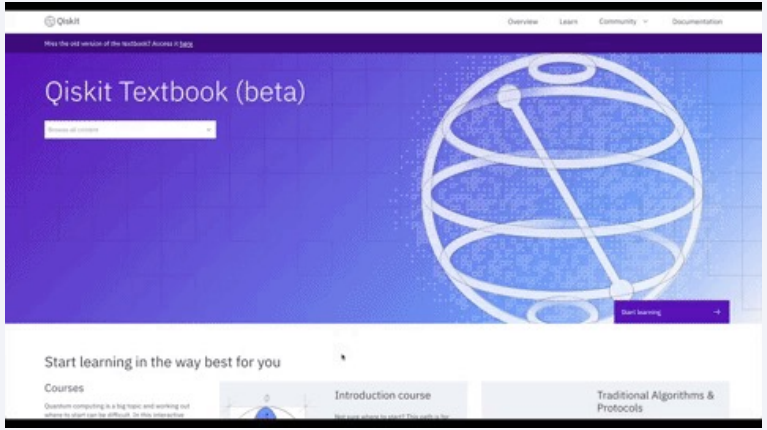
Development

**Learning**

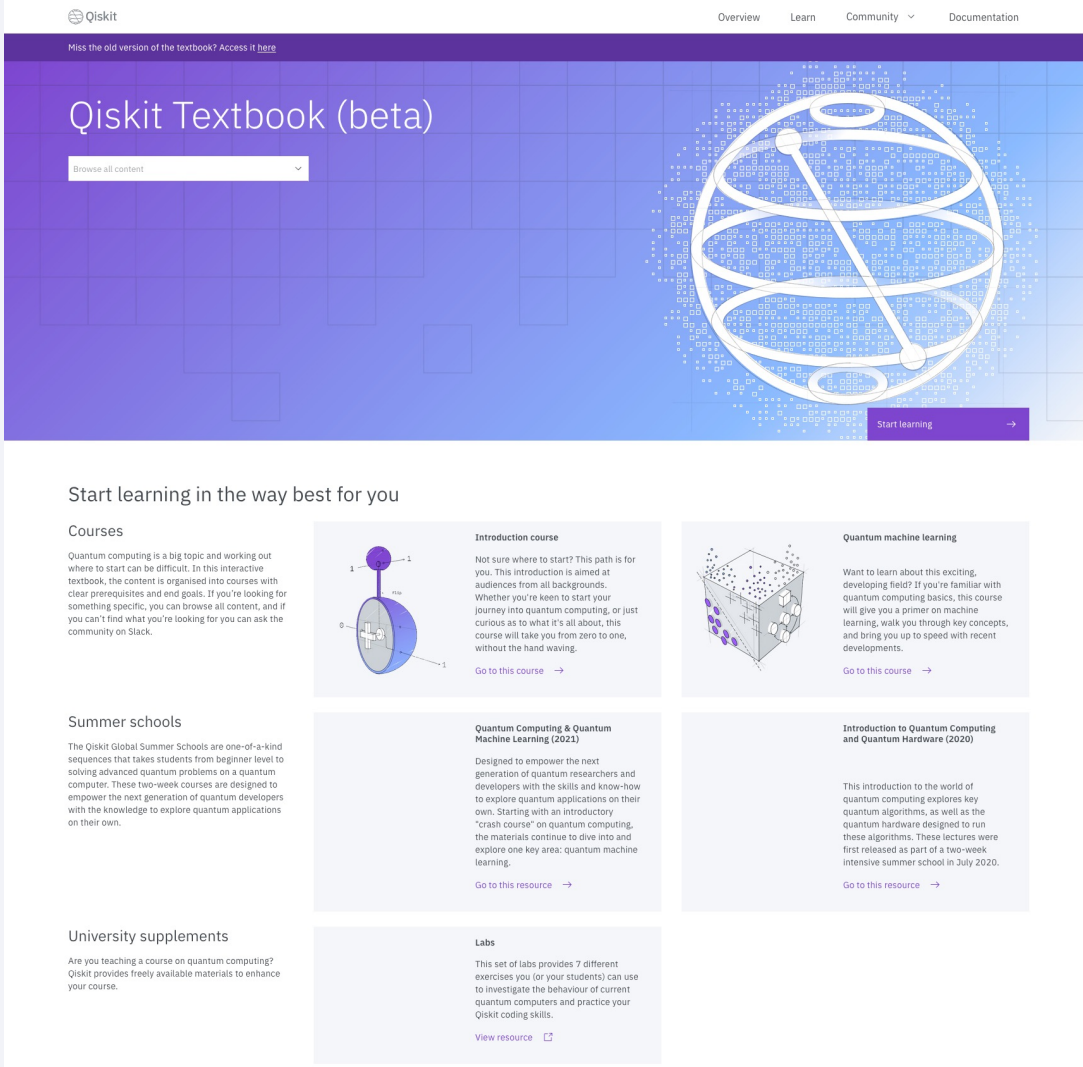
# Qiskit textbook

The Qiskit textbook (qiskit.org/textbook-beta) is an open-source, university-level quantum algorithms / computation course with Qiskit code implementations and interactive features

Top source of QC knowledge on the web (2nd only to Wikipedia)





# IBM Quantum Challenges

Qiskit

One-week long online quantum programming challenges (1000+ participants each time)

Featuring Qiskit advanced capabilities and dedicated quantum systems

- Sep 2019: Fall Challenge 2019
- May 2020: Spring Challenge 2020
- Sep 2020: Qiskit Challenge India
- Nov 2020: Fall Challenge 2020
- Feb 2021: ICPC Challenge
- May 2021: Spring Challenge 2021
- Sep 2021: Africa Challenge
- Oct 2021: Fall Challenge 2021
- May 2022: Spring Challenge 2022

## IBM Quantum Challenge generates better solutions than challenge creators thought possible

From May 20 to 26, 1,431 people from 76 countries participated in the IBM Quantum Challenge 2021, a celebration for 5-year anniversary of IBM Quantum and 40 years of quantum computing. Not only was the event a blast, but participants devised a more-efficient solution to one quantum computing problem than the problem's authors even thought possible.

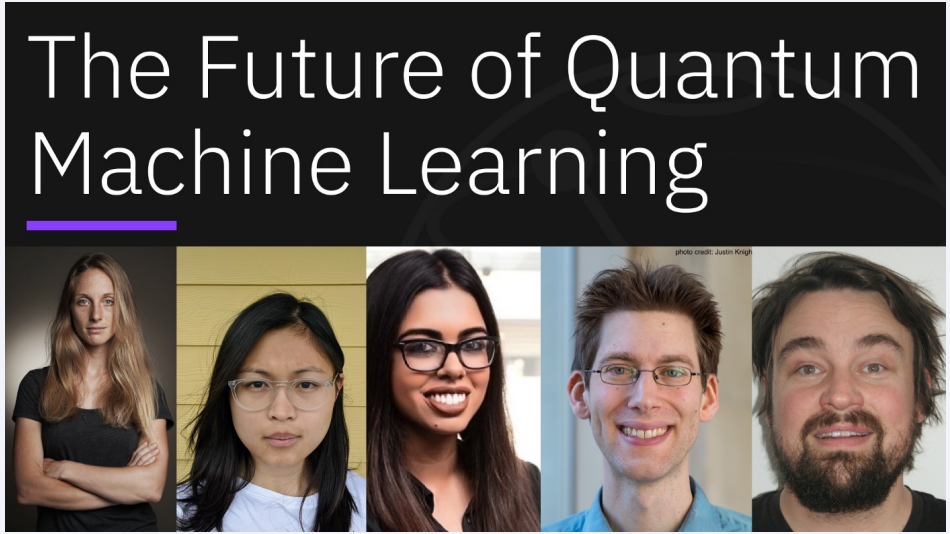
# Qiskit Global Summer Schools

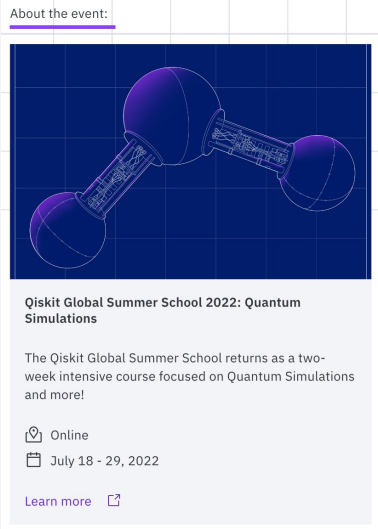
Qiskit

2-week intensive summer school (equivalent to one-semester course)

Largest quantum summer school (4000+ students for both 2020 and 2021)

QGSS 2022 will focus on quantum simulations. Early bird ran out in just a few hours! Registration will open again on June 2, 6pm ET. (Follow Qiskit Twitter for the latest announcement).

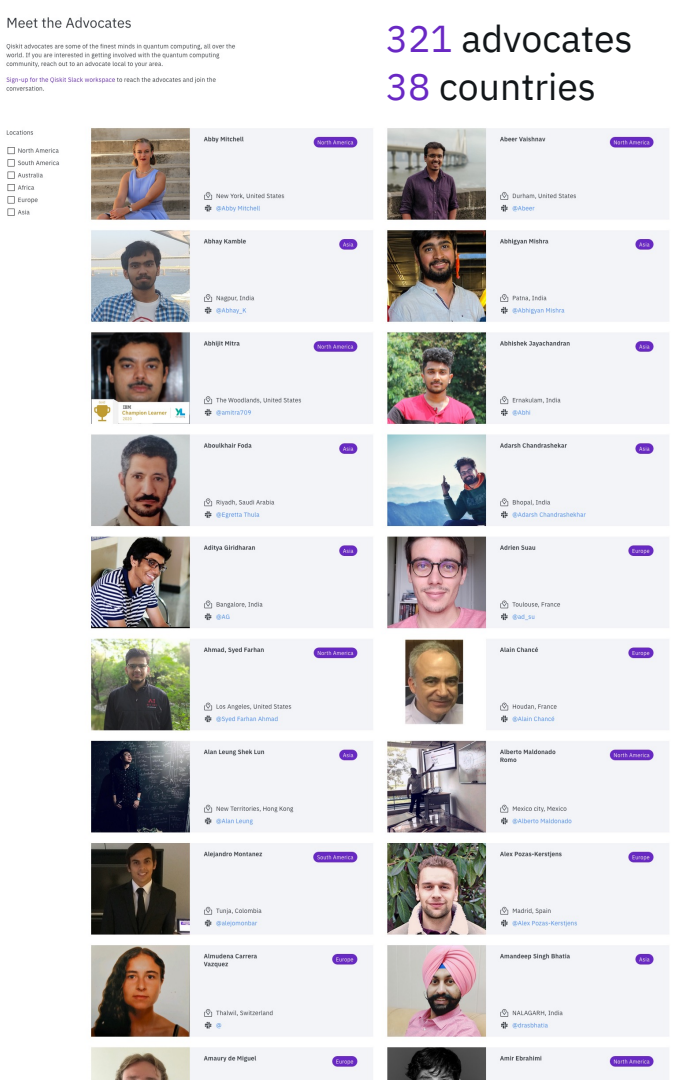
# Qiskit advocate program

The Qiskit advocate program (qiskit.org/advocates) is a global program that provides support to the individuals who actively contribute to the Qiskit Community.

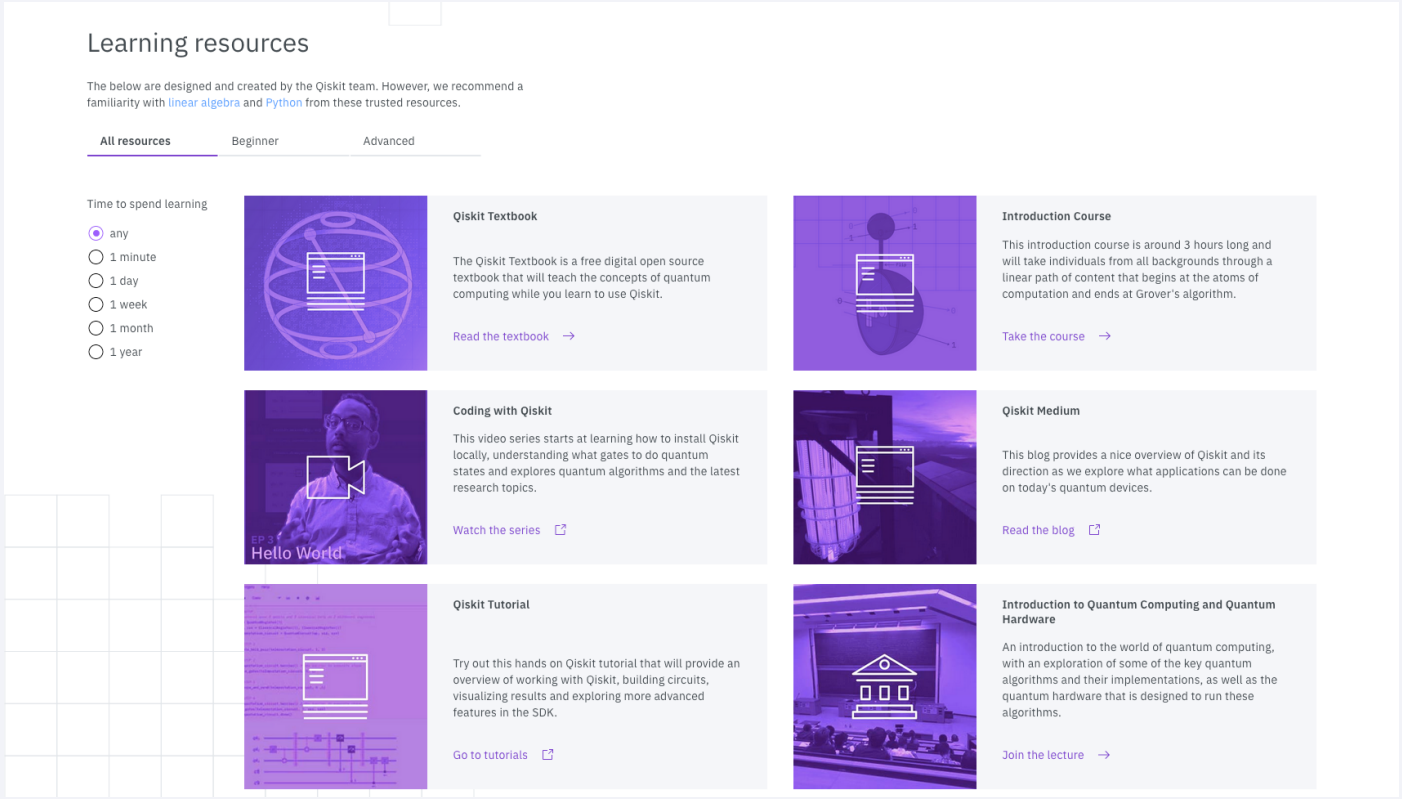
Benefits:

| | |
|---|---|
| **Mentorship** | **Network with experts and enthusiasts** |
| **Access to Qiskit core members and projects** | **Invitation to events** |

2022 application opens on June 1!



# Other learning tools and resources




Qiskit Learn ([qiskit.org/learn](qiskit.org/learn))

- Qiskit Textbook
- Qiskit Tutorials
- Qiskit YouTube
- Qiskit Medium
- QGSS recordings and materials

# Summary

# What is Qiskit?

**High level applications**

**Qiskit Nature**
For applications relating to simulating quantum mechanical systems and natural phenomena.

**Qiskit Finance**
For applications relating to financial modeling.

**Qiskit Optimization**
For applications relating to optimization problems.

**Qiskit Machine Learning**
For applications relating to machine learning.

**Low level applications**

Qiskit

**Qiskit Metal**
For designing quantum hardware and processors.

**Qiskit Dynamics**
For building, transforming, and solving time-dependent models of quantum systems.

**Qiskit Experiments**
For running quantum experiments with a library of characterization, calibration, and verification experiments.

**Core Capabilities**

**Qiskit Terra**
For building and transforming quantum circuits and operators at the level of gates or pulses.

**Simulator**

**Qiskit Aer**
For simulating quantum circuits on classical hardware.

**Hardware providers**

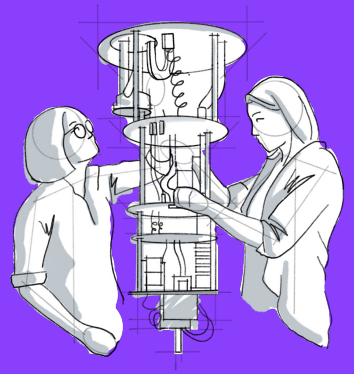
**IBM**
IBM Quantum systems

**AQT**
AQT systems

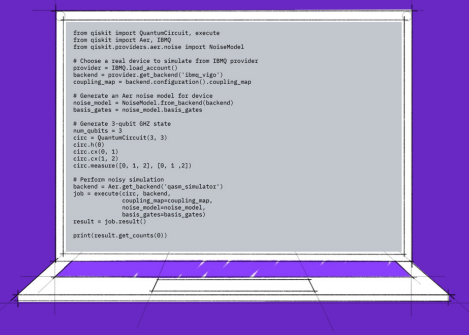
**IonQ**
IonQ systems

Qiskit can connect to many other systems

## Research



## Development




## Learning

# Thank you

Qiskit

Slides are available on
https://github.com/HuangJunye/presentations

Follow @Qiskit

 Twitter

 YouTube

 Slack

 Medium