IBM Quantum
Getting Started Resources

IBM Quantum Network

IBM **Quantum**

Qiskit: Getting Started

Qiskit Installation – setup python environment and install Qiskit

Qiskit Youtube Channel - "How to Install Qiskit"

Introduction to Qiskit - Tutorial #1 – example of the Qiskit workflow with each step explained in detail

Getting Started Tutorial 1 – example of the Qiskit workflow using IBM Quantum Lab Jupyter notebook

General Qiskit Tutorials list – tutorials in Quantum Lab broken down into several categories

General Learning Resources

Qiskit Textbook

Qiskit Release Notes

Access: Getting Started

Quantum Tools – quantum services and cloud programming tools

- Documentation – User guides with multiple paths to get started
- Quantum Composer – graphical programming tool to work with quantum circuits
- Quantum Lab – cloud-enabled programming environment using online Jupyter notebooks

Access Systems with your account - overview of how to use your IBM Quantum account to access the systems and simulators available in IBM Quantum

System Configuration – Explains IBM Quantum system information

IBM Quantum Services – View the status and details of IBM Quantum's systems and simulators, and track which are available

Job Flow: 1. Creation -> 2. Submission -> 3. Queuing -> 4. Result

1. Jobs that are run on quantum systems must first be composed as a QuantumCircuit or Pulse Schedule in either a local or cloud-hosted Jupyter notebook or on IBM Quantum via the Circuit Composer or Quantum Lab.
   - In order to execute the job, the circuit or schedule must be transpiled according to the (static) backend configuration, i.e. the native gate set and connectivity of the backend. At higher levels of the transpiler, (dynamic) backend properties may also be taken into account when mapping circuits or schedules to real hardware, i.e. T_1, T_2, gate errors, etc. that are benchmarked after each automatic calibration (currently performed once every 24 hours). These will be taken into account in the transpiler with the *initial_layout* parameter.
2. The job is then assembled as a Python dictionary called a Qobj and sent to the backend. The availability and details of IBM Quantum programs, systems, and simulators are viewable on the IBM Quantum Services page.
   - Your specific provider determines which backends you have access to.
   - Each job is a submission which may consist of multiple different circuits/schedules with each circuit/schedule repeated multiple times (shots) to gather good statistics.
3. Once the job is submitted to an IBM backend via the cloud, it enters the backend's queue with:
   - Fair share queue - Normal job submission calculated from dynamic priority
   - Priority Mode - Jump to the front of the queue
   - Dedicated Mode - Sole access to a given backend
   - While you can monitor the total pending jobs on your backends, because of the queuing dynamics, this may not accurately reflect how long your job will actually take to run.
4. Each shot contains measurement pulses where the phase imparted on the measurement pulse determines the state of the qubit that was measured. This signal may be returned as a raw voltage waveform (meas_level=0 in Qiskit Pulse), an integrated signal determined by a default or selectable kernel or meas_level=1, or a discriminated signal (count) using the default or selectable discriminator. This data is returned via the result() method of the IBMQJob.

Recommendations and Guidance (tips & tricks)

Recommended Workflow

- Test and debug your circuits by first executing on simulators, such as the *qasm_simulator*
  - This simulator is very fast and will not ping usage of actual hardware
- Consider choosing the smallest machine with most ready availability for the job in terms of qubit numbers
  - Job size is more limited on certain (e.g. public) machines (75 vs 900 circuits/schedules)
  - Run with less shots initially (i.e., 1024 vs max=8192)
- Once you are ready for prime time, bundle jobs together to increase efficiency
  - Get a run-time estimate by submitting a representative job before bundling them
    - This can be found at https://quantum-computing.ibm.com/results/<job_id> under "Running"
  - See example notebook (upon request) or 1 Minute Qiskit How-to for demonstrating how to bundle jobs
- Log in to IBM Quantum to manage and view job status or use Qiskit for job monitoring
- Many other useful tips and tricks as part of 1 Minute Qiskit series
- Understand the mapping of the circuit qubits to hardware via the transpiler and the role of specifying an *initial_layout* or *optimization_level* to achieve best results
- Explore using advanced techniques including Mid-circuit measurement and Qiskit Runtime

Publication Guidance

- Any publications that make use of Qiskit should reference it as here
- Publications that make use of IBM Quantum Services should follow citing backend usage guidelines and acknowledge access through their Hub; for example, one can do so by stating "Access to the IBM Quantum Services was obtained through the IBM Quantum Hub at CERN" in the Acknowledgements section.

IBM Quantum Resources

Learning and Usage Tools

Intro to Quantum Computing + Hardware Online Course

- A comprehensive two-week course on quantum computing, hardware and much more – equipped with videos, notes and labs

Qiskit Online Open-Source Textbook

- Interactive advanced text on quantum algorithms and computation based on Qiskit

Qiskit Youtube Channel

- **Qiskit Foundations & Algorithms** (learn to program with Qiskit)
- **Qiskit Live: Circuit Sessions** (value and use of quantum circuits)
- **Quantum Information Science Seminar** (research topic deep-dive)
- **SuperPosition Series** (explores how individuals became Qiskit developers)
- **1 Minute Qiskit** (Qiskit tips & tricks)

IBM Quantum Tools

- The leading quantum services and cloud programming tools to help accelerate quantum research and applications
- Quantum Composer – a feature-rich graphical programming tool to work with quantum circuits
- Quantum Lab – a cloud-enabled programming environment featuring Qiskit tutorials

Qiskit.org

- The leading, Python-based software development kit (SDK) for open-source quantum development that allows you to build and deploy applications using powerful quantum services
- Documentation, Tutorials, Events, Metal – device design

Publications & Demos

- IBM Institute for Business Value
  - Research, reports, & insights on quantum computing use-cases in specific industries, including: finance, manufacturing, airlines, life sciences, healthcare, chemicals/petroleum
- IBM Quantum Papers
  - List of publications from IBM & Quantum Network partners
- Quantum Applications
  - Finance
  - Chemistry/Life Science
  - AI/ML
  - Optimization

Recent Announcements

- Roadmap for Scaling Quantum Technology
- Roadmap for Building an Open Quantum Software Ecosystem
- Video presentation of the IBM Quantum Development Roadmap
- A New OpenQASM for a New Era of Dynamic Circuits
- Updates and new features in IBM Quantum

Social Media & Partner Programs

IBM Quantum Blog | Medium | Slack

Twitter | GitHub | Stack Exchange

Qubit[x] Qubit - quantum education

IBM **Quantum**

IBM Quantum Network members are enabled through various engagement and support programs

Dedicated Qiskit Support

- Answer questions through github, stack exchange, slack & email

- Provide feedback and suggested efficiencies in Qiskit

ibmq@us.ibm.com

Continuing Education

- **Ask-the-Researcher Webinars**
  - Scientific seminars on published work (past recordings available)

- **Teach-the-Researcher Algorithm courses**
  - Multi-day deep dive courses on specific algorithms to enable research
  - Topics include VQE, QML, Quantum Error Correction
  - Recordings of previous courses are available

- **IBM Quantum Summit**
  - Annual gathering of the full IBM Quantum Network