Welcome to quantum !

Qiskit is an open-source quantum computing software development framework for working with quantum computers at the level of extended quantum circuits, operators, and primitives.

- IBM Quantum Platform and its content
  - Dashboard, documentation, learning
- How to install and use Qiskit
  - Set up, build, run, study a job
  - Qiskit-ibm-runtime and its hardware
- Your access with qiskit and basic troubleshooting

IBM **Quantum**

IBM Quantum Platform
https://quantum.ibm.com

Access to your account via the platform, the documentation about Qiskit and Qiskit Runtime and learning materials

Github organization
https://github.com/Qiskit

Find all repositories related to qiskit, such as the documentation, runtime, qiskit itself, etc.

Slack workspace
https://qisk.it/join-slack

Open slack workspace with all the Qiskit community, lots of channels dedicated to specific subjects

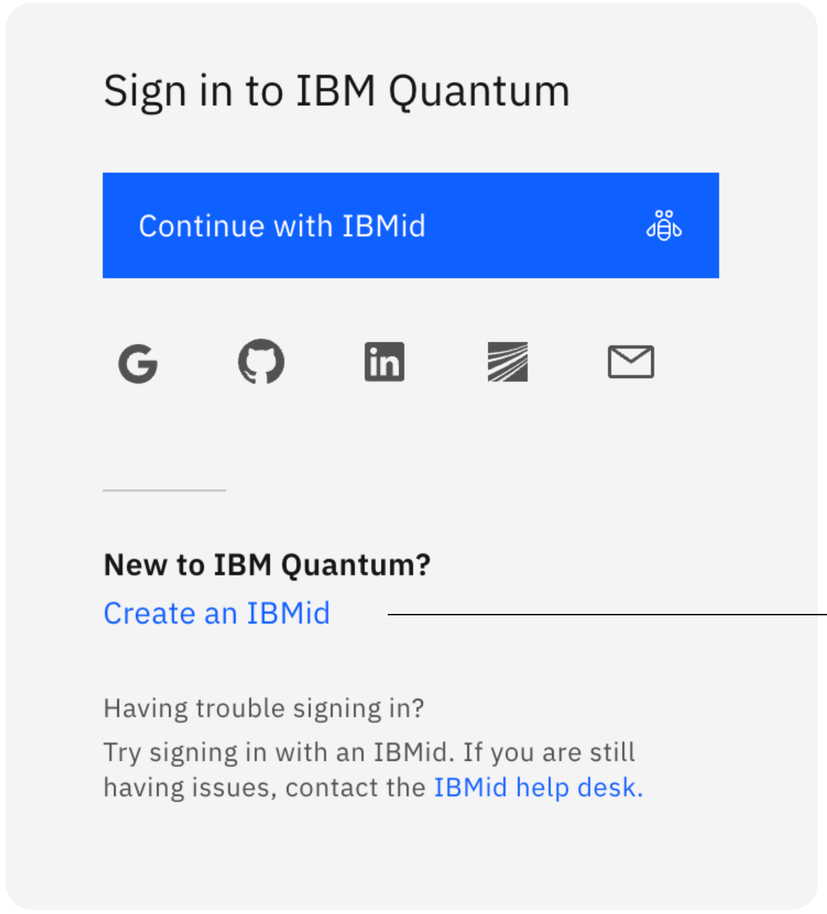
Youtube channel
https://www.youtube.com/Qiskit

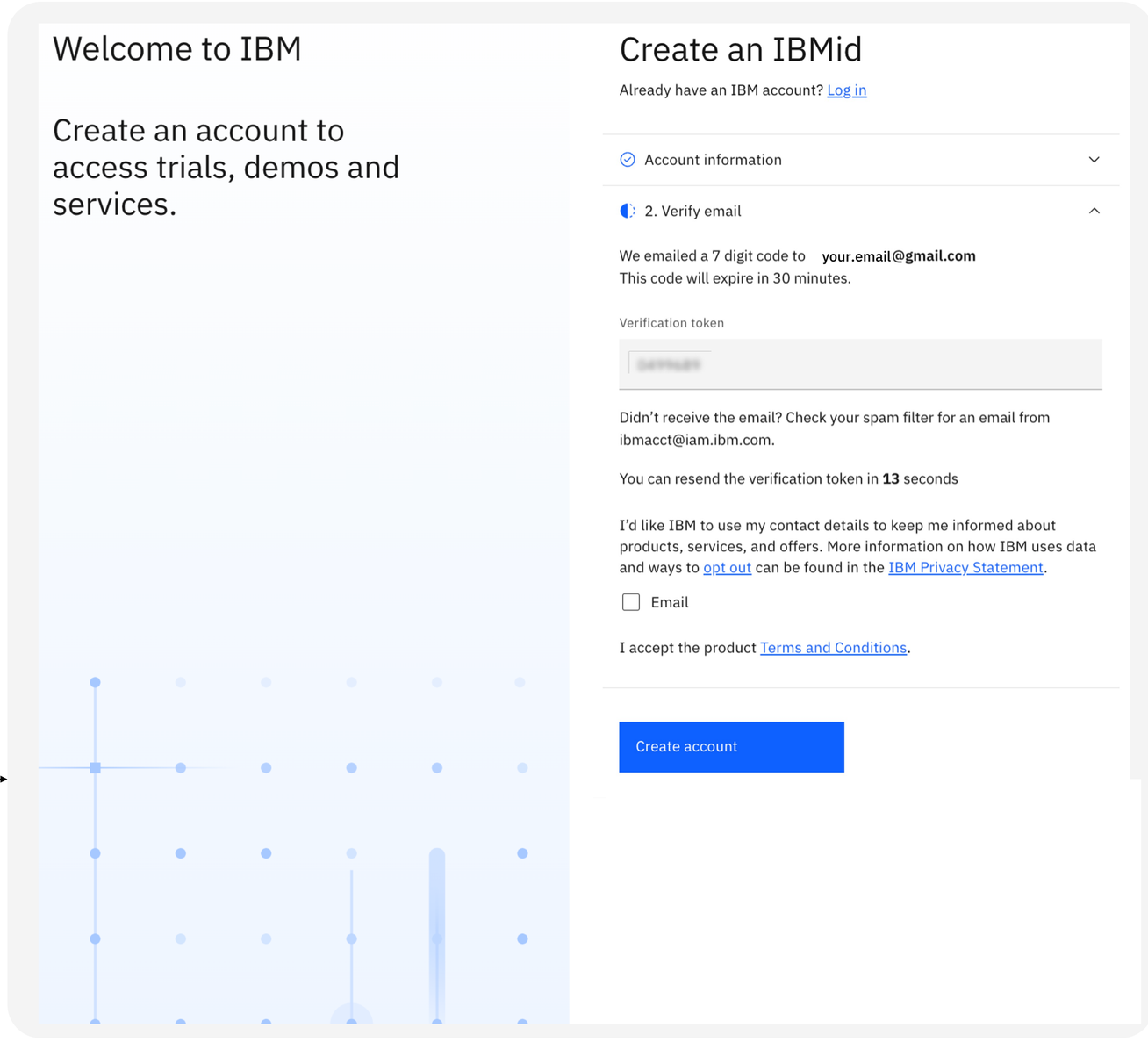
Videos related to QI/QC and Qiskit, access to seminars, paper reviews, tips about Qiskit, etc.

IBM Quantum Platform: getting started

Go to the IBM Quantum Platform
https://quantum.ibm.com

and create an IBMid to run your own circuits!





IBM Quantum **Platform** | Dashboard | Compute resources | Jobs | Experiments | ibm-q-internal/support/core-team

Lena Perennes

IBM Quantum Platform

API Token

Switch applications

Platform
Jobs and compute resources

Documentation
Qiskit and API docs

Learning
Courses, Lab and Composer

Recent jobs View all

0 Pending

16340 Completed jobs

Job ID	Status	Created	Completed	Compute resource
	Completed	7 days ago	7 days ago	ibm_torino
	Completed	10 days ago	10 days ago	ibmq_kolkata
	Completed	10 days ago	10 days ago	ibmq_kolkata
	Completed	10 days ago	10 days ago	ibmq_kolkata
	Completed	10 days ago	10 days ago	ibmq_kolkata

What's new →

- Product update — Updates to Learning -- new Learning catalog! — 2 months ago • Read mo
- Product update — Introducing ibm_osaka — 2 months ago • Read mo
- Product update — Qiskit.org redirects an — 3 months ago • Read mo
- Product update — Journey toward utility: Open plan users — 3 months ago • Read mo
- Product update — New URL strategy on I — 3 months ago • Read mo
- Product update — Updates to our quality — 3 months ago • Read mo

Instance systems →

22

Simulators →

5

Documentation Open app ↗

Search docs

Hello World
Create a simple quantum program and run it on a quantum system

Qiskit Runtime
Introduction to primitives

Learning Open app ↗

Catalog New
Explore all courses and tutorials

IBM Quantum Composer
Graphically build circuits

IBM Quantum Lab
Develop quantum experiments

Overview of the platform : https://quantum.ibm.com

IBM Quantum Documentation

Documentation includes guidance for those starting their quantum journey:

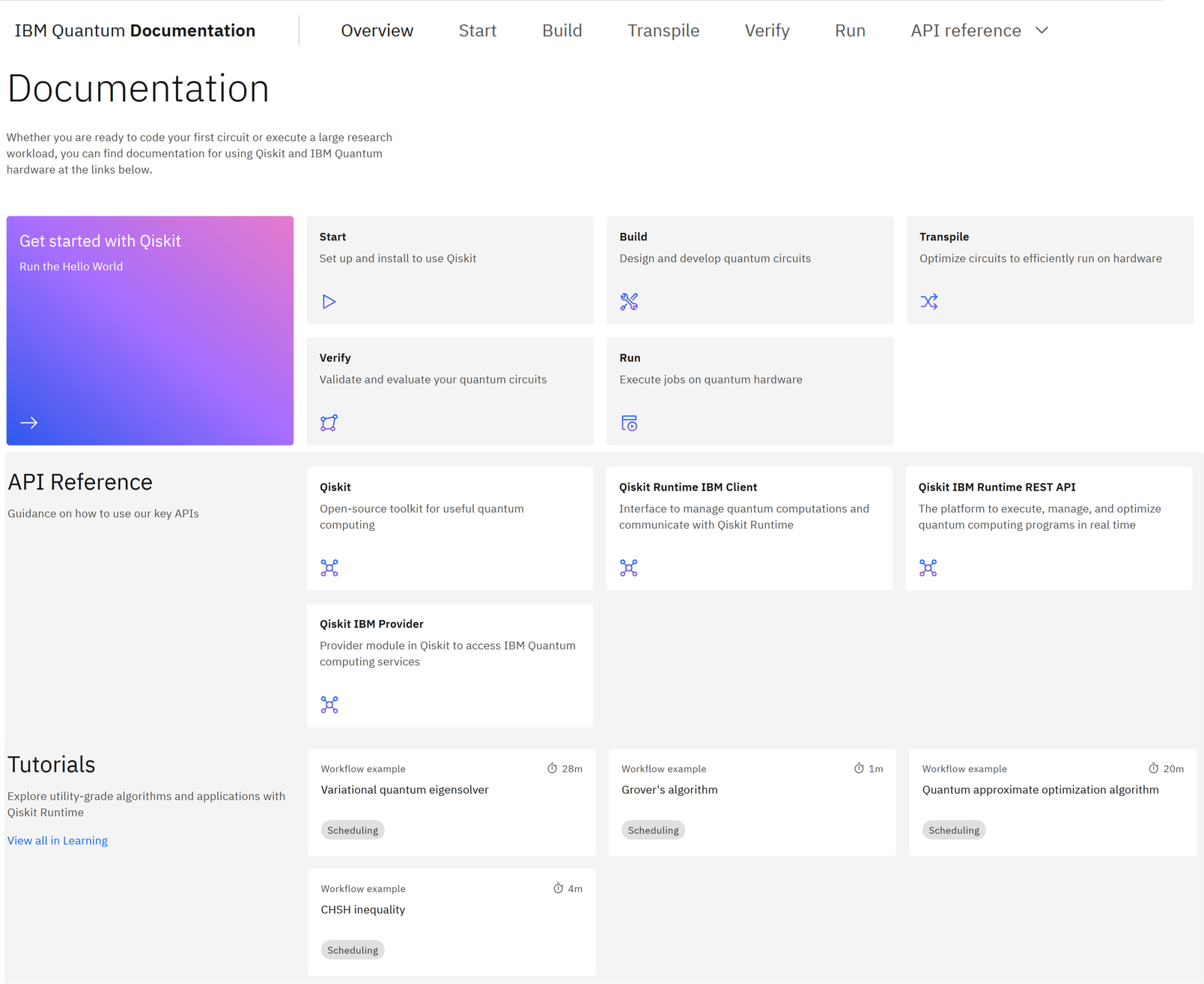
- Start: Starting with Qiskit
- Build: Building quantum circuits
- Transpile: Optimization of circuits to quantum hardware
- Verify: Evaluation of outputs
- Run: Executing jobs on quantum hardware

Documentation also provides useful resources for technical practitioners on:

- Qiskit
- Runtime
- Providers

https://docs.quantum.ibm.com

IBM Quantum



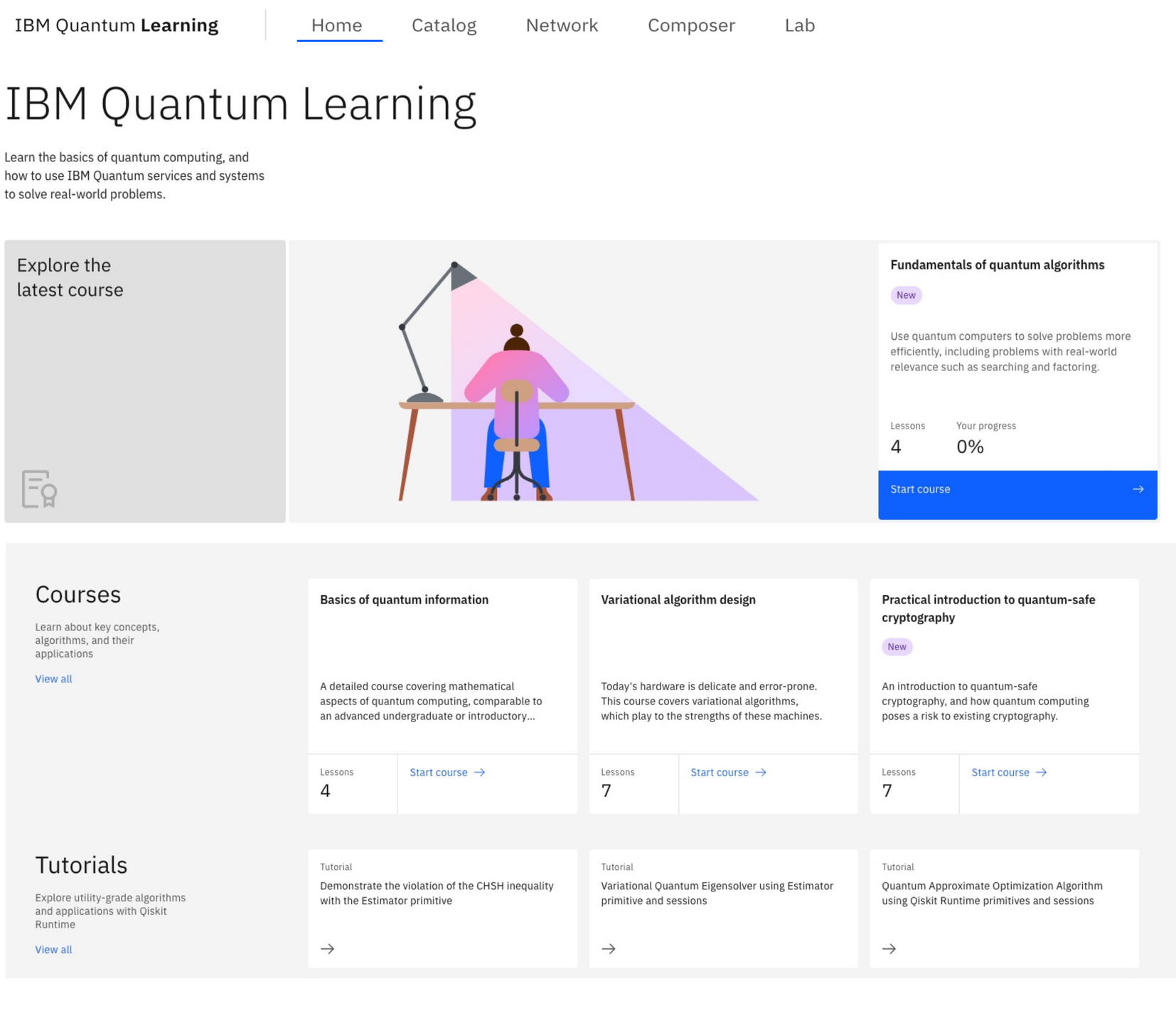
IBM Quantum Learning

Interactive courses, tutorials, and tools to help you build a foundation in quantum.

Experiment with the latest IBM Quantum hardware and software technology.

Badging available for some courses!

https://learning.quantum.ibm.com/



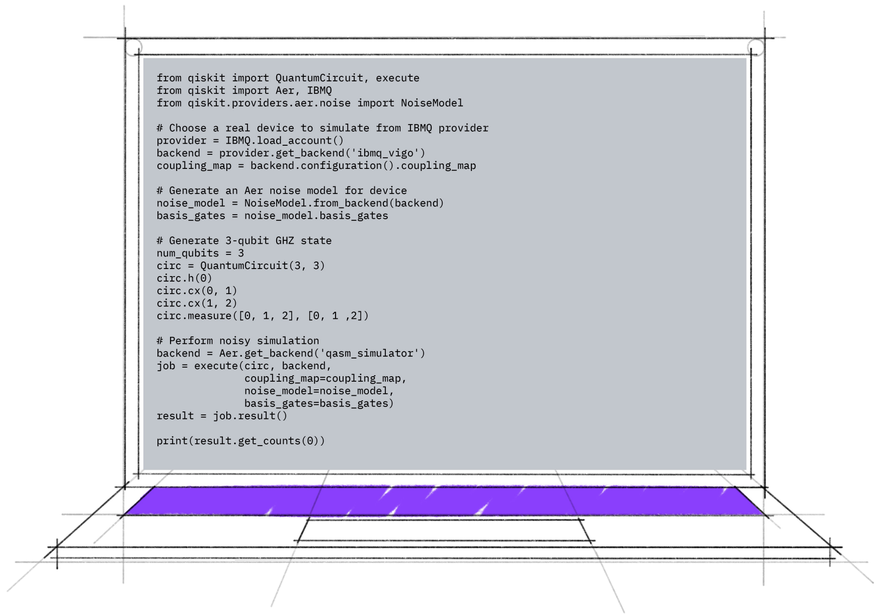
Set up your account with Qiskit

How to use Qiskit

Option 1 : install Qiskit locally

- Install Python, 3.8 or later compatible
- Recommended to use virtual environments (python env, Anaconda, miniconda, etc.) along with **jupyter development env**
  - isolated space to work with Python for a specific purpose, without affecting the "base" Python environment on your machine
- Install qiskit and qiskit-ibm-runtime
- More details available here https://docs.quantum.ibm.com/start/install

- If you already have qiskit packages installed, make sure you have all latest versions
  - Careful not to upgrade qiskit 0.x to 1.x in the same env



```
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))
```

How to set up the account locally qiskit-ibm-runtime

qiskit-ibm-runtime :

- Access to managed services through the Qiskit Runtime primitives
- For high quality probability distribution or expectation values, without having to optimize the circuits yourself
- Migrate to using Qiskit Runtime primitives https://docs.quantum.ibm.com/api/migration-guides/qiskit-runtime
- API token available https://quantum.ibm.com/account

```python
from qiskit_ibm_runtime import QiskitRuntimeService

token='<MY_API_TOKEN>'
instance='hub/group/project'
QiskitRuntimeService.save_account(token=token,
                                  instance=instance,
                                  channel='ibm_quantum')

service = QiskitRuntimeService(channel='ibm_quantum')

# See your overall access via instances
service.instances()

# Access to backends
service.backends()
service.get_backend('ibm_torino')
service.least_busy(simulator=False, operational=True)

# Access to jobs
service.job('my_job_id')
service.jobs(backend='ibm_torino')
```

Getting started with Qiskit

Hello World with Qiskit

Qiskit

How to create a quantum circuit to prepare the quantum state we want ?

We want to create the following quantum state : $|\psi\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$

Entrée [4]:

```
1 from qiskit import QuantumCircuit
2
3 bell = QuantumCircuit(2,2)
4 bell.h(0)
5 bell.cx(0,1)
6 bell.measure(range(2), range(2))
```

Out[4]:

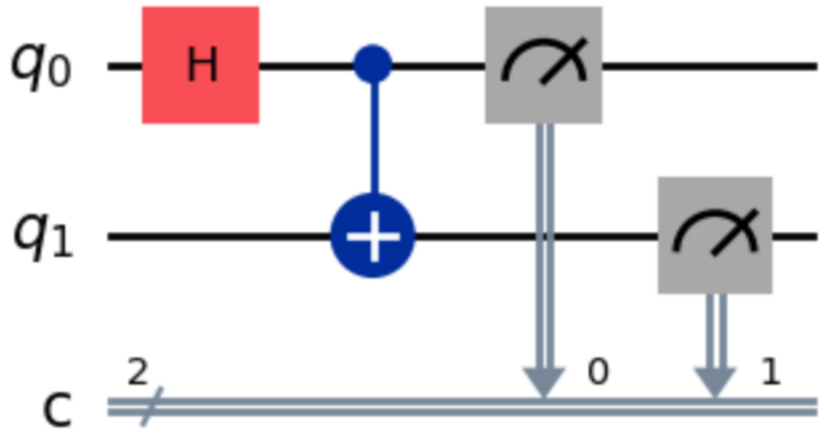
```
<qiskit.circuit.instructionset.InstructionSet at 0x7fd3a27bf2e0>
```

Hello World with Qiskit

Qiskit

```
[2]: bell.draw('mpl')
```

[2]:



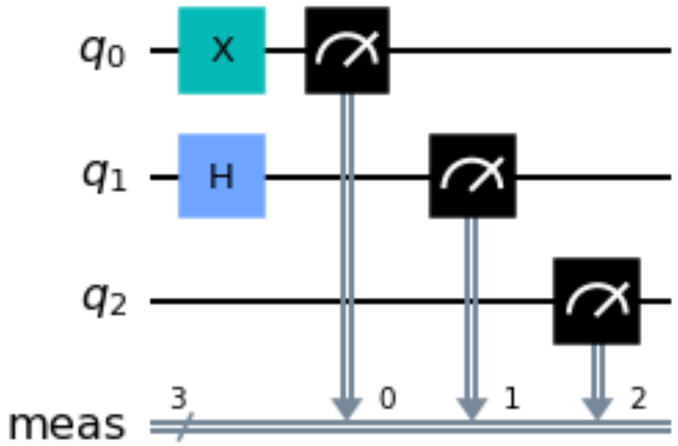
- ➔ Hadamard gate (H gate) to induce superposition
- ➔ CNOT gate (also known as CX gate) to induce entanglement
- ➔ The measure to collect information on our final quantum state

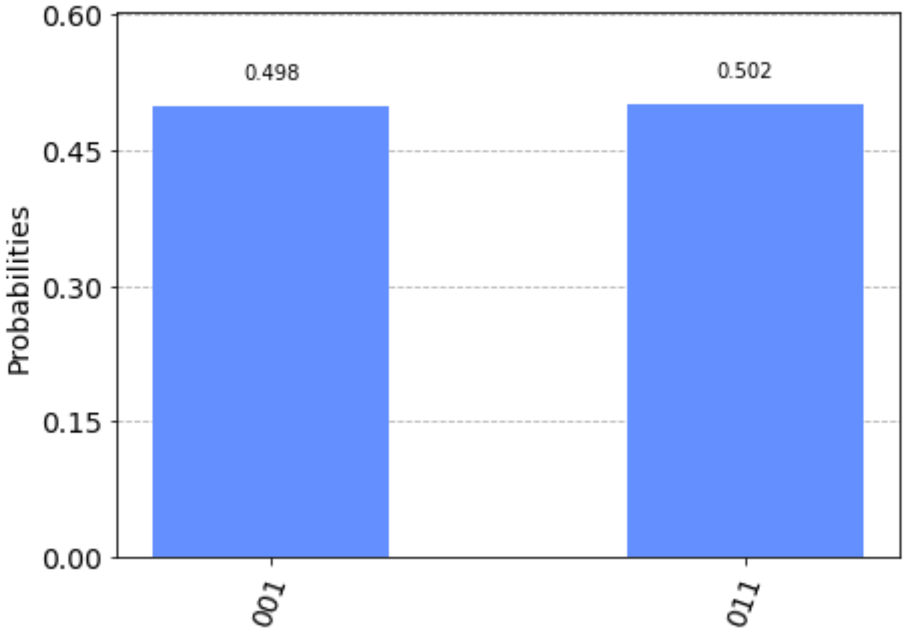
Qiskit bit ordering

- In most of the textbook/litterature in quantum computing ➔ big endian notation : lsb to the left
- Qiskit bit and qubit ordering ➔ little endian notation : lsb to the right

Qubits ordering : $|q_n q_{n-1} \dots q_1 q_0\rangle$

Bits ordering : $b_n b_{n-1} \dots b_1 b_0$





Hello World with Qiskit

Qiskit

How to run our circuit to check we have good results ?

[22]:
```
from qiskit_ibm_runtime import QiskitRuntimeService, SamplerV2 as Sampler
# load your instance along with the backendsavailable to you
service = QiskitRuntimeService(channel='ibm_quantum')
backend = service.backend('ibm_brisbane')
```

[23]:
```
from qiskit_aer import AerSimulator
# load an ideal simulator
sim = AerSimulator()
```

[24]:
```
from qiskit.transpiler.preset_passmanagers import generate_preset_pass_manager
pass_manager = generate_preset_pass_manager(optimization_level=3, backend=backend)
pass_manager_sim = generate_preset_pass_manager(optimization_level=3, backend=sim)
```

Hello World with Qiskit

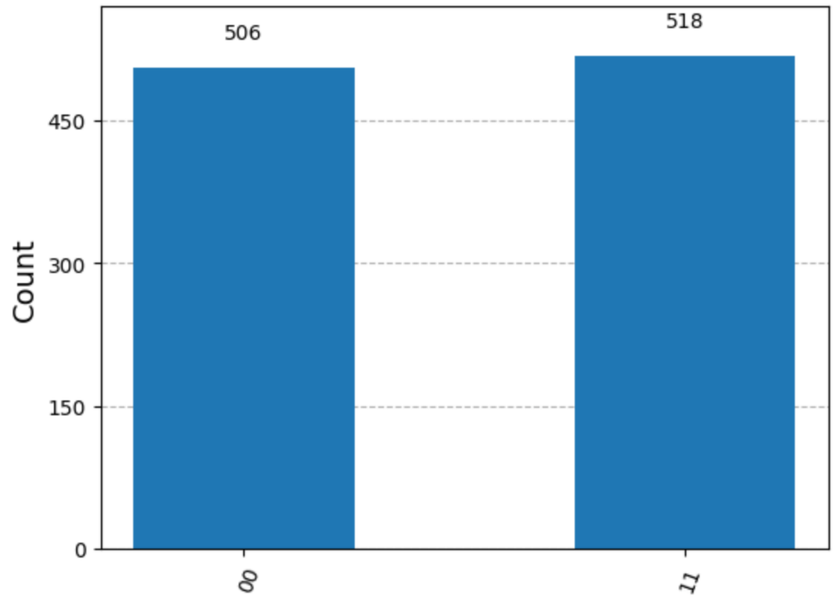
How to visualize the results ?

Ideal classical simulation

```
[20]: trans = pass_manager_sim.run(bell)
      job = sim.run(trans)
```

```
[21]: from qiskit.visualization import plot_histogram
      counts = job.result().get_counts()
      plot_histogram(counts)
```

[21]:

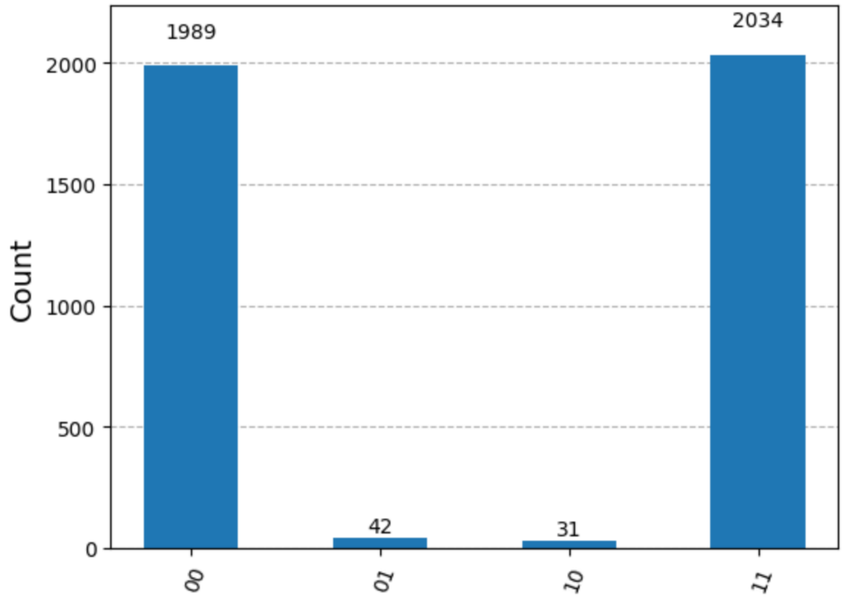


Real quantum hardware

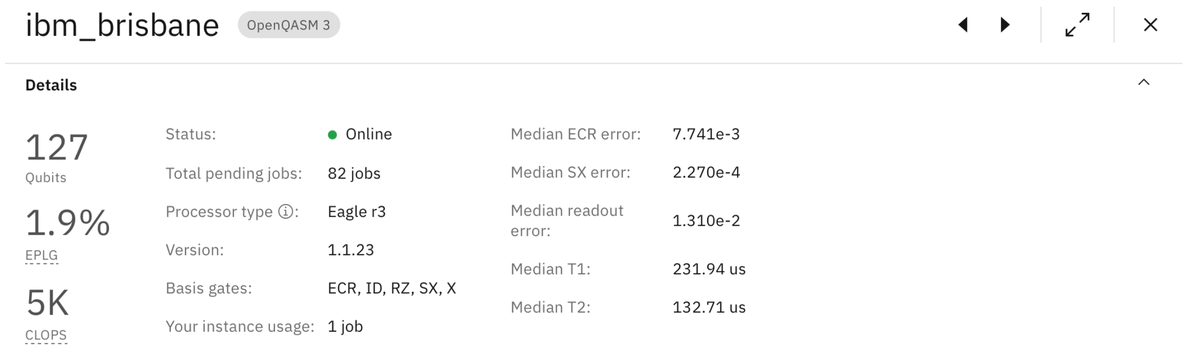
```
[14]: trans = pass_manager.run(bell)
      job = Sampler(backend=backend).run((trans,))
```

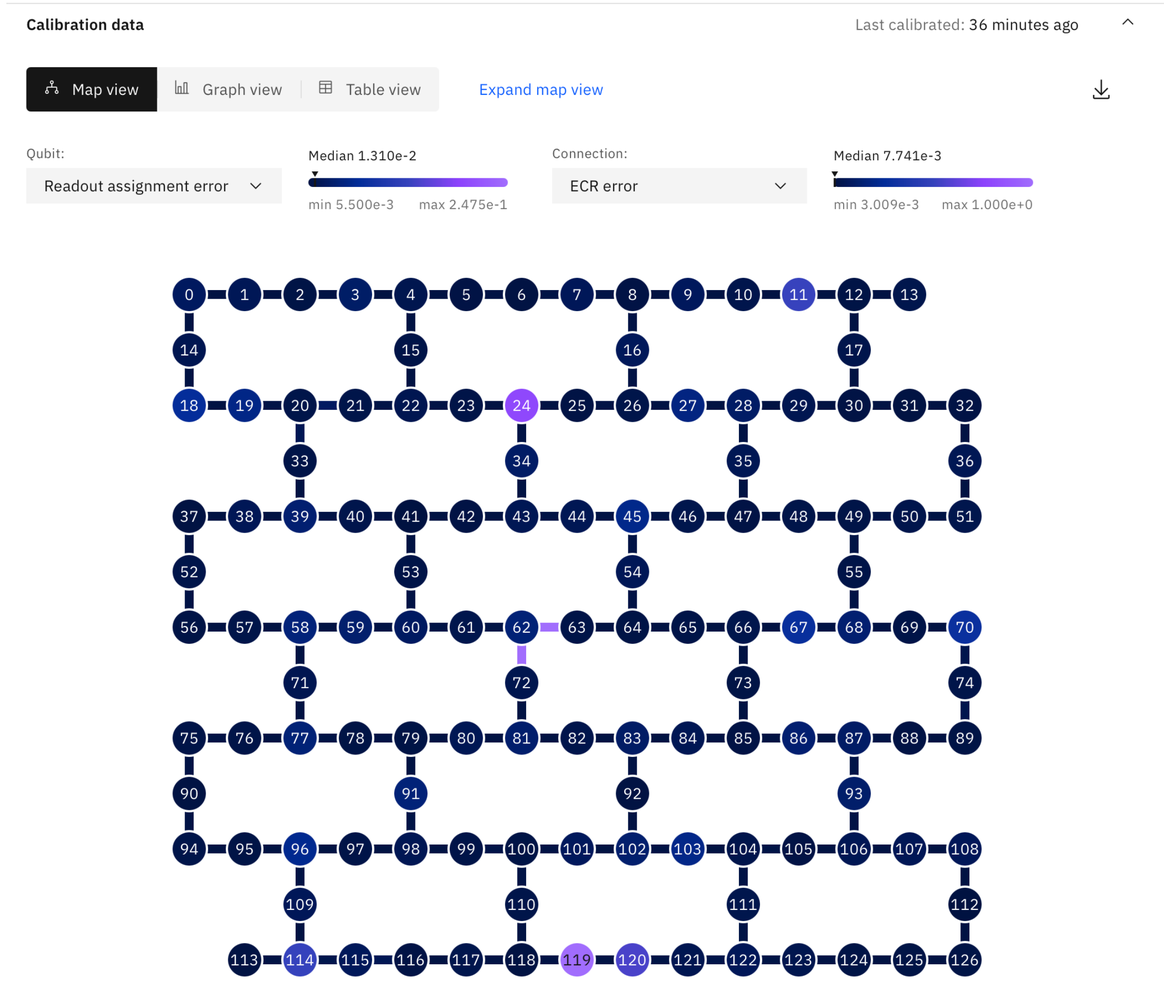
```
[19]: from qiskit.visualization import plot_histogram
      counts = job.result()[0].data.c.get_counts()
      plot_histogram(counts)
```

[19]:



Characteristics of a real backend





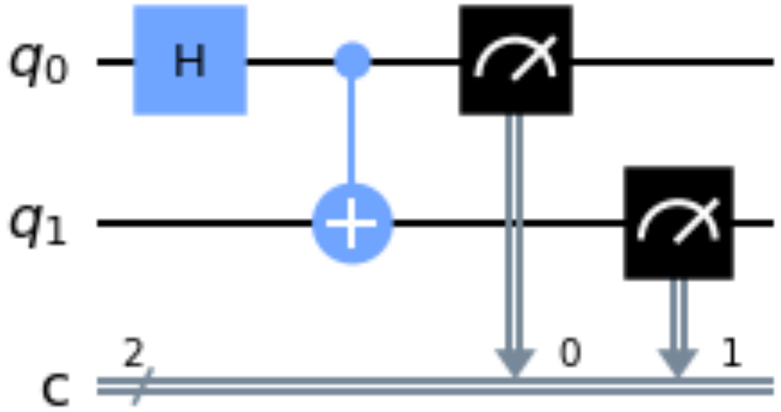
- Topology of the backend
- Errors related to gates and measures
- Characteristics related to qubits
- More details available here
  https://docs.quantum.ibm.com/run/processor-types

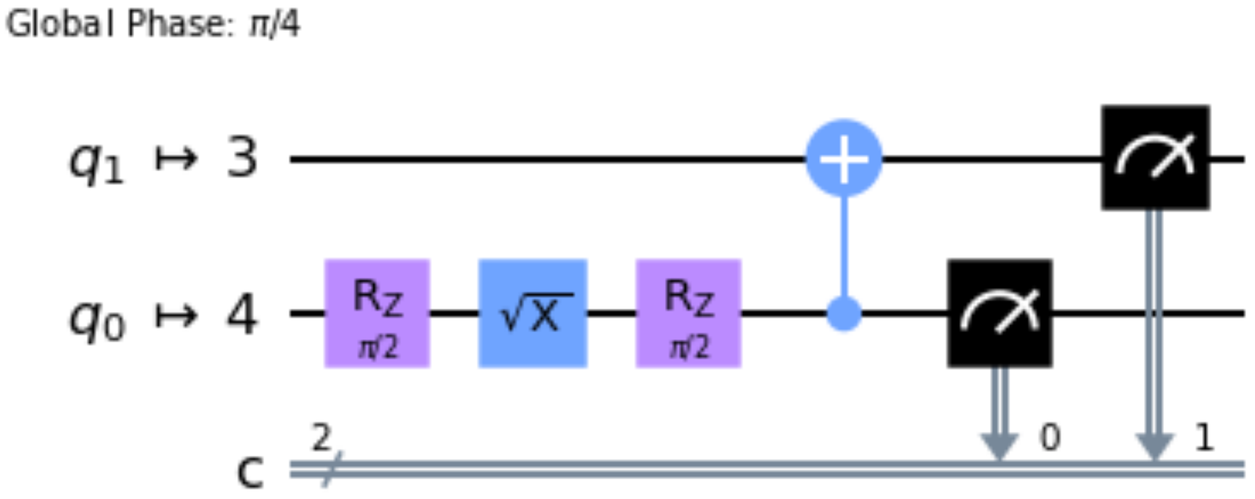
Running on a real backend

Qiskit

Considering all this different characteristics, the quantum computer will not be able to understand an abstract quantum circuit we just created without some modifications.

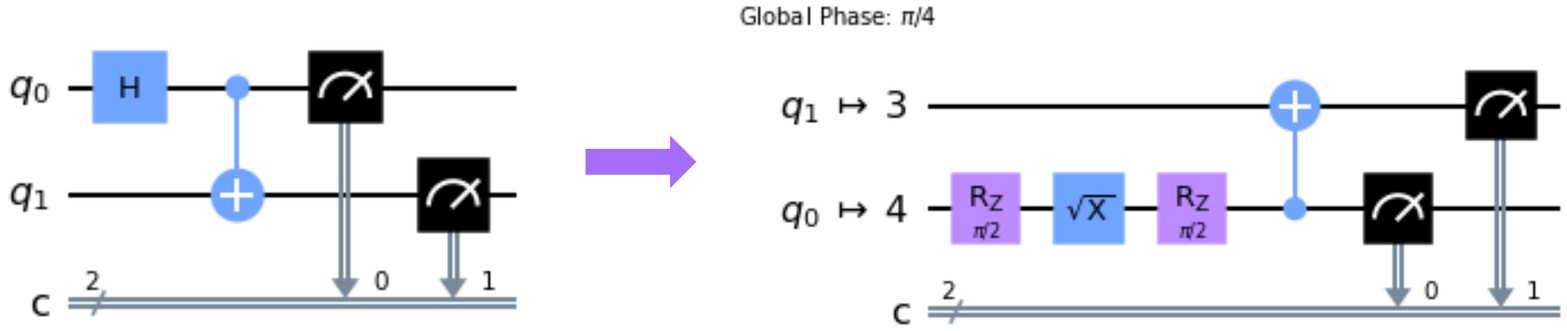
➔ The circuit needs to be re-written, translated in order to run smoothly and in an efficient way on the quantum computer





Qiskit – The transpile process

- Transpilation is the process of rewriting a given input circuit to match the topology of a specific quantum device, and optimize the circuit instructions for execution on noisy quantum systems.
- The transpiler is designed for modularity and extensibility. Its main goal is to write new circuit transformations (known as **transpiler passes**), and combine them with other existing passes, greatly reducing the depth and complexity of quantum circuits
- There are six stages: init, layout, routing, translation, optimization, scheduling
- More details available here https://docs.quantum.ibm.com/transpile



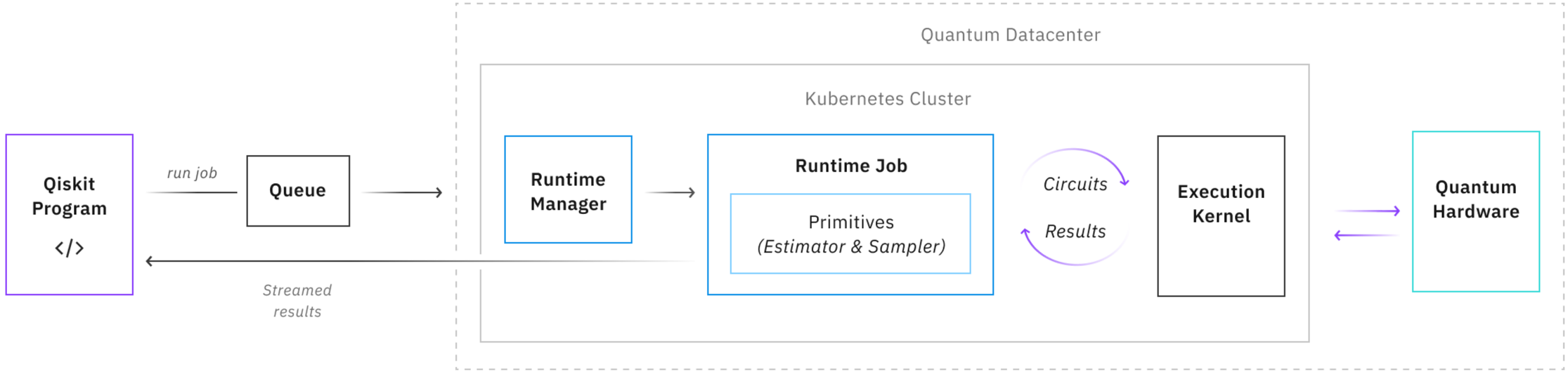
Study a job from beginning to end

Study a job from beginning to end Qiskit IBM Runtime

What is Qiskit runtime?

New model for quantum computing access : Qiskit Runtime is a new architecture that streamlines computations requiring many iterations. These experiments will execute significantly faster within its improved hybrid quantum/classical process

Documentation https://docs.quantum.ibm.com/api/qiskit-ibm-runtime/runtime_service



Study a job from beginning to end
Understanding primitives

Sampler :

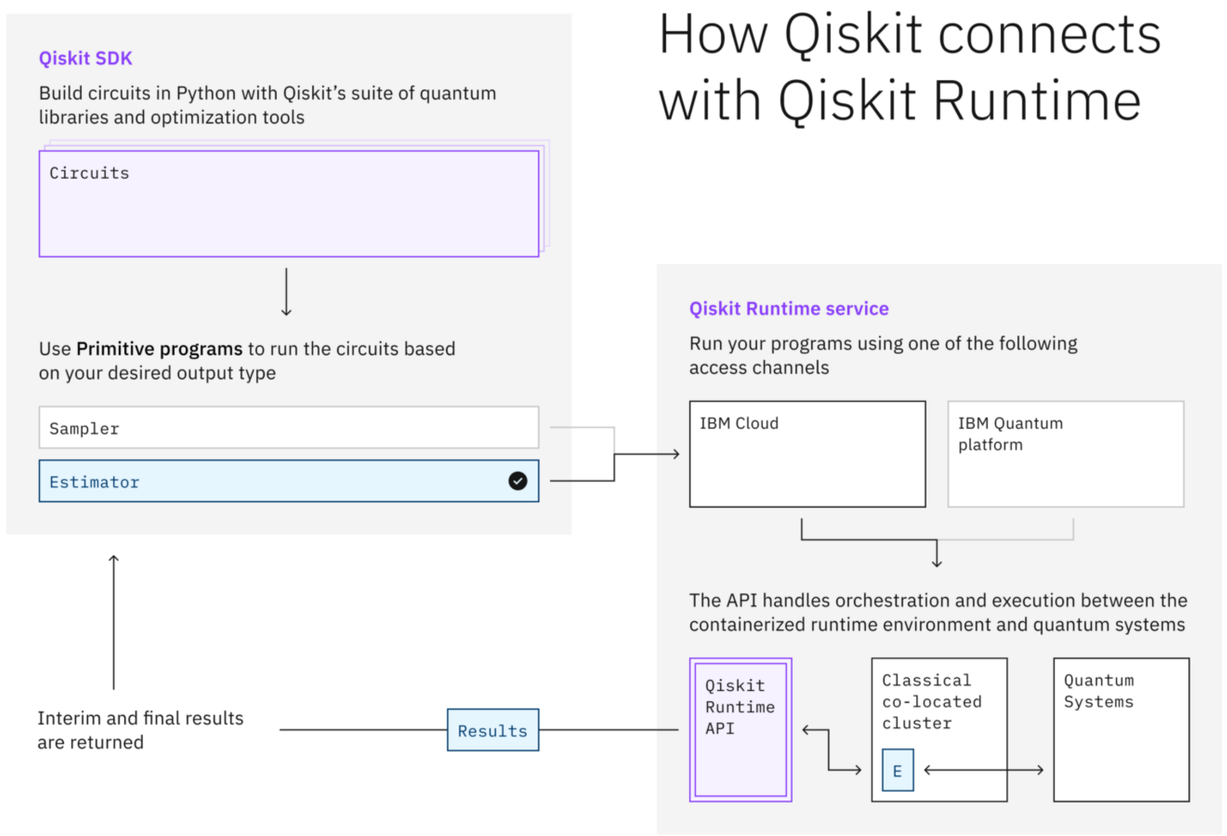
- takes circuits and returns the corresponding sampled bitstrings from the measurement

Estimator :

- computes expectation values of observables with respect to states prepared by quantum circuits

Introduction to primitives
https://docs.quantum.ibm.com/run/primitives

Introduction to execution mode
https://docs.quantum.ibm.com/run/sessions



Check your access with Qiskit

Access to instances and backends information

Account access

- Save and delete accounts
- Check all saved accounts
- Look at your access through the instances

```python
from qiskit_ibm_runtime import QiskitRuntimeService

QiskitRuntimeService.save_account(token=token)

QiskitRuntimeService.saved_accounts()

QiskitRuntimeService.active_account()

QiskitRuntimeService.delete_account()

# See you access
service = QiskitRuntimeService(channel='ibm_quantum')
service.instances()
```

Backend view

- See all backends you have access though your account
- Filter backends to your liking
- Get a specific backend

```python
from qiskit_ibm_runtime import QiskitRuntimeService

service = QiskitRuntimeService(channel='ibm_quantum')

service.backends()

service.backends(filters=lambda x: ("rz" in x.basis_gates))

service.backends(simulator=False,
                 operational=True,
                 min_num_qubits=30)

service.least_busy(simulator=False)

service.get_backend(name='ibm_torino')
```

Job load

- Access to jobs through filtering
- Get a specific job
- See all info about one job

```python
from qiskit_ibm_runtime import QiskitRuntimeService

service = QiskitRuntimeService(channel='ibm_quantum')

service.jobs(limit=None,
             backend_name='ibm_torino',
             instance='hub/group/project',
             pending=False)

job = service.job('MY_JOB_ID')

job.creation_date
job.metrics()
job.program_id()
job.properties()
job.result()
job.session_id
```

How to do basic troubleshooting in Qiskit

How to do basic troubleshooting in Qiskit

If you ever encounter an error within your code, there could be multiple reasons and causes to this. It can either be software or hardware-related, it might come from another package qiskit uses, etc. Here are some first steps you can take to try and find it

Error message

Lots of clues as to why the error is appearing can be found in the error message. That way you can pinpoint whether it's coming from the code, or Qiskit, or if it's just a deprecation warning.

Error code

Within the error message, you might have an error code related. You can check the error code registry to know what the error means and either solve it directly or **contact us** for more information if needed

Packages versions

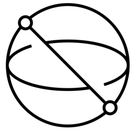
Code can sometimes break because of version upgrades and changes within qiskit or runtime/provider packages. Make sure you have the latest version. You can check on github or pypi to check this.

Technical support

I'll be here all day to help with any technical issue or question you might be having about our software, hardware.

IBM Quantum Challenge 2024

Learn how to use **Qiskit 1.0** and build toward **utility-scale** quantum experiments.



The IBM Quantum Challenge is the annual code challenge focused on using Qiskit. It consists of a series of Jupyter notebooks that ask you to learn new features, use Qiskit, and solve problems.

Jun 05 — Jun 14, 2024

Sign up using your IBM Quantum Network email address → challenges.quantum.ibm.com

Participation is free, so join and take part in the IBM Quantum Challenge !

Key information

- **Adapt to utility-scale workloads** by shifting away from the IBM Quantum Lab and instead running your code locally.
- **Earn a Credly badge** to showcase your achievement by completing the challenge.
- **Read more** here → https://www.ibm.com/quantum/blog/2024-quantum-challenge

IBM **Quantum** Network

Thank you

Léna Pérennès
IBM Quantum Support team
lena.perennes@ibm.com

Qiskit – useful links for the hackathon

- IBM Quantum Platform https://quantum.ibm.com
- Documentation https://docs.quantum.ibm.com
- Learning https://learning.quantum.ibm.com
- Intro to start with qiskit
  https://docs.quantum.ibm.com/start
- Qiskit documentation
  https://docs.quantum.ibm.com/api/qiskit
- Qiskit-ibm-runtime documentation
  https://docs.quantum.ibm.com/api/qiskit-ibm-runtime
- Migration guide
  https://docs.quantum.ibm.com/api/migration-guides
- All tutorials
  https://learning.quantum.ibm.com/catalog/tutorials

Qiskit – links to go further

- Qiskit-aer documentation https://qiskit.github.io/qiskit-aer/
- Dynamic circuit introduction https://docs.quantum.ibm.com/build/classical-feedforward-and-control-flow
- Openqasm3 introduction https://docs.quantum.ibm.com/build/interoperate-qiskit-qasm3
- Information on backend characteristics https://quantumcomputing.stackexchange.com/questions/17054/characteristics-of-the-ibm-quantum-computer/
- Circuit library https://docs.quantum.ibm.com/build/circuit-library
- Build circuit https://docs.quantum.ibm.com/build/circuit-construction
- Build operator https://docs.quantum.ibm.com/build/operators-overview
- Transpile https://docs.quantum.ibm.com/transpile
- Pulse https://docs.quantum.ibm.com/build/pulse
- Build noise model https://docs.quantum.ibm.com/verify/building_noise_models