Open-Source Tools for Platform Agnostic Quantum Computing

FOSDEM'25

Harshit Gupta
Quantum Software Engineer
qBraid

Agenda

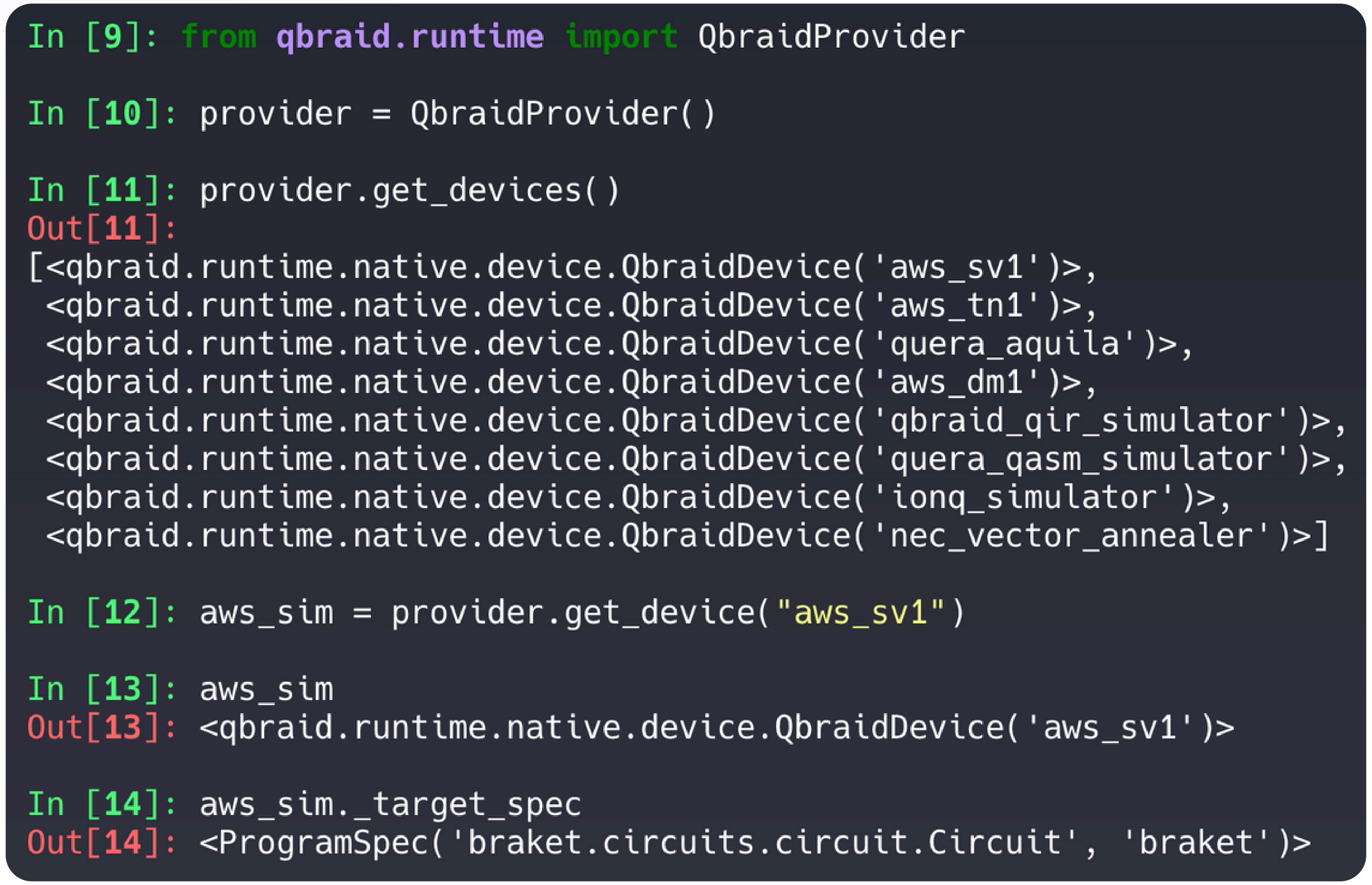
1. About qBraid
2. qBraid SDK
   - Runtime
   - Transpiler
3. PyQASM

qBraid | OVERVIEW

ONE-STOP PLATFORM ENABLING QUANTUM COMPUTING

qBraid | LAB

QUANTUM SOFTWARE DEVELOPMENT IN THE CLOUD



```
In [9]: from qbraid.runtime import QbraidProvider

In [10]: provider = QbraidProvider()

In [11]: provider.get_devices()
Out[11]:
[<qbraid.runtime.native.device.QbraidDevice('aws_sv1')>,
 <qbraid.runtime.native.device.QbraidDevice('aws_tn1')>,
 <qbraid.runtime.native.device.QbraidDevice('quera_aquila')>,
 <qbraid.runtime.native.device.QbraidDevice('aws_dm1')>,
 <qbraid.runtime.native.device.QbraidDevice('qbraid_qir_simulator')>,
 <qbraid.runtime.native.device.QbraidDevice('quera_qasm_simulator')>,
 <qbraid.runtime.native.device.QbraidDevice('ionq_simulator')>,
 <qbraid.runtime.native.device.QbraidDevice('nec_vector_annealer')>]

In [12]: aws_sim = provider.get_device("aws_sv1")

In [13]: aws_sim
Out[13]: <qbraid.runtime.native.device.QbraidDevice('aws_sv1')>

In [14]: aws_sim._target_spec
Out[14]: <ProgramSpec('braket.circuits.circuit.Circuit', 'braket')>
```

aws

IBM

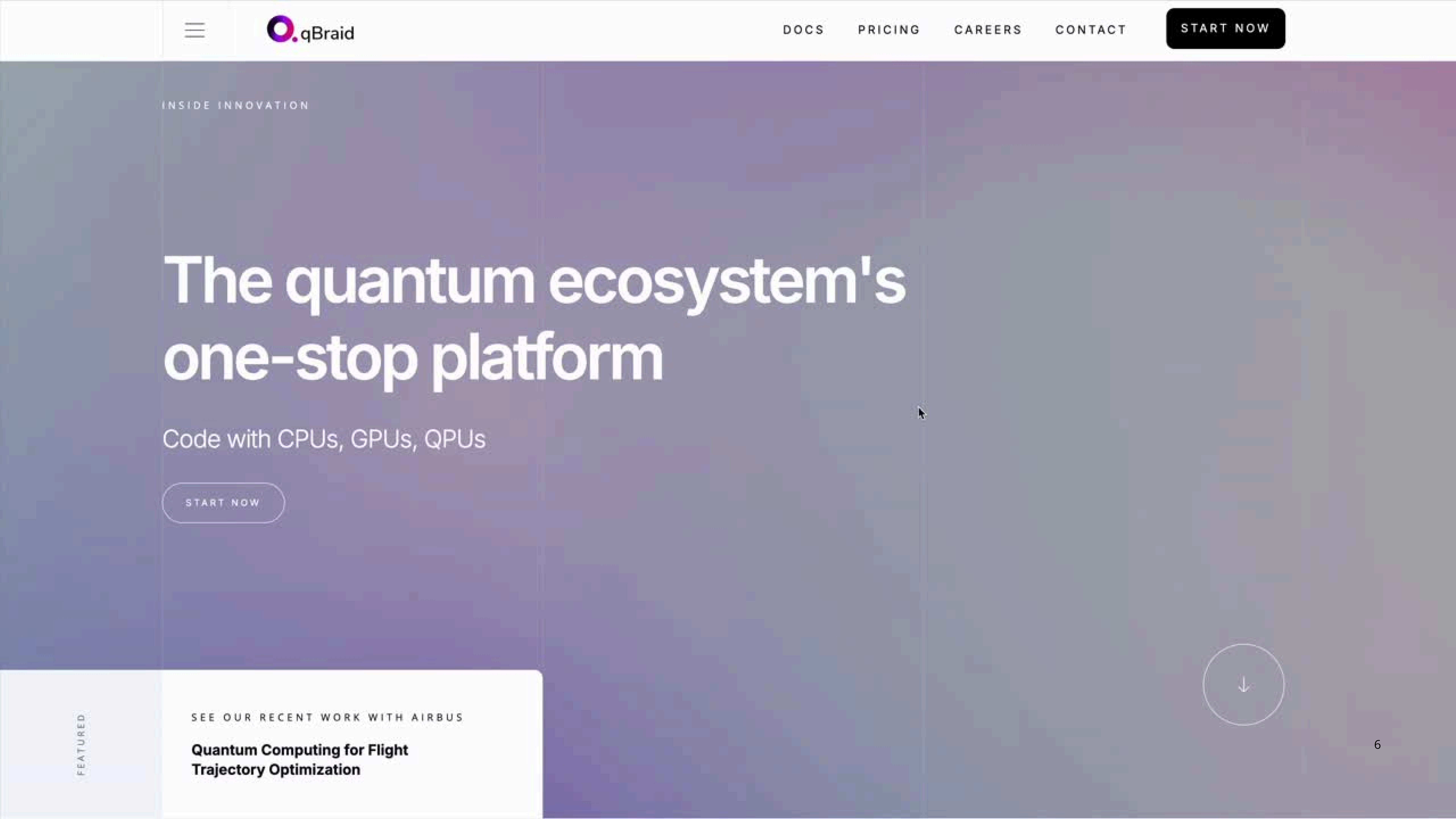
Avoid fickle installation procedures for quantum software python packages

qBraid | SDK

Integrate with the qBraid SDK to run quantum jobs on hardware seamlessly using cirq, pyqir, qasm, qiskit, pyquil and braket

With qBraid, quantum hardware or software is available out of the box with little to no installation

Demo



INSIDE INNOVATION

The quantum ecosystem's one-stop platform

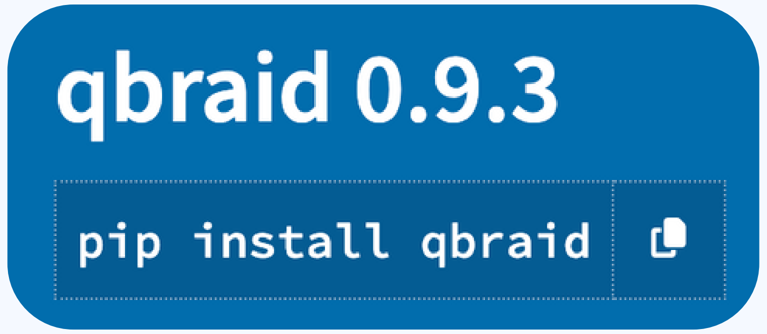
Code with CPUs, GPUs, QPUs

SEE OUR RECENT WORK WITH AIRBUS

Quantum Computing for Flight Trajectory Optimization

Open-Source Tools

qBraid SDK



qbraid 0.9.3

```
pip install qbraid
```

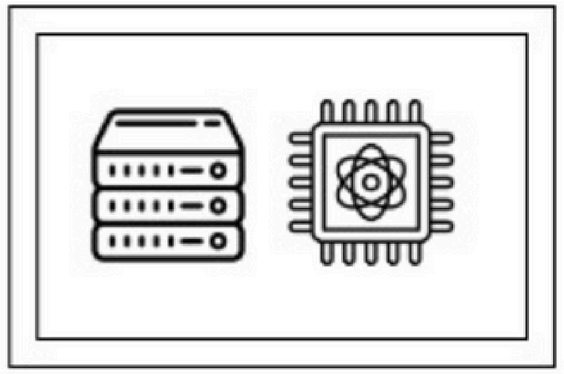
PLATFORM AGNOSTIC QUANTUM SOFTWARE DEVELOPMENT

qbraid.runtime

STREAMLINING QUANTUM WORKLOADS

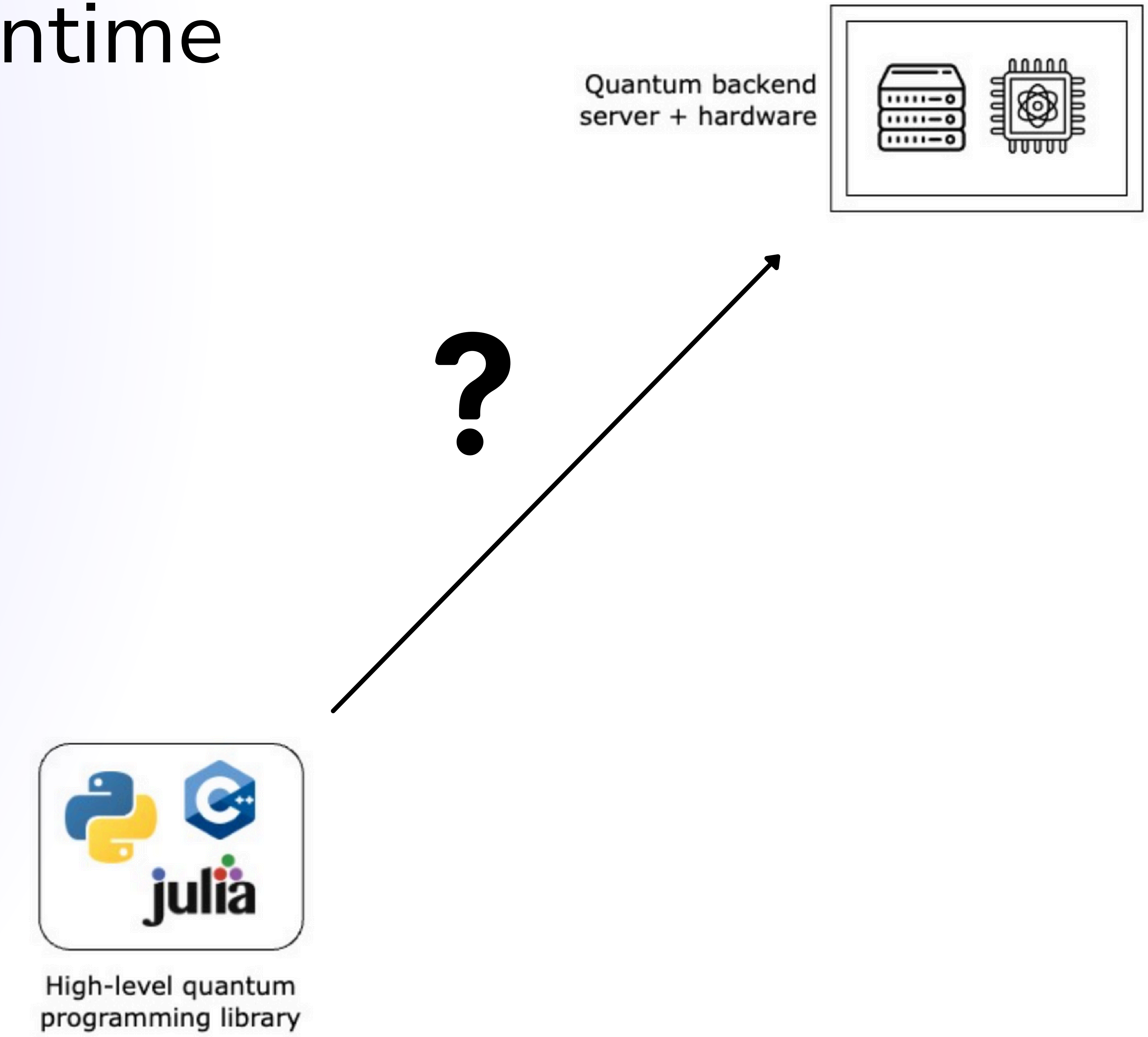
qbraid.runtime

Quantum backend
server + hardware



High-level quantum
programming library

qbraid.runtime

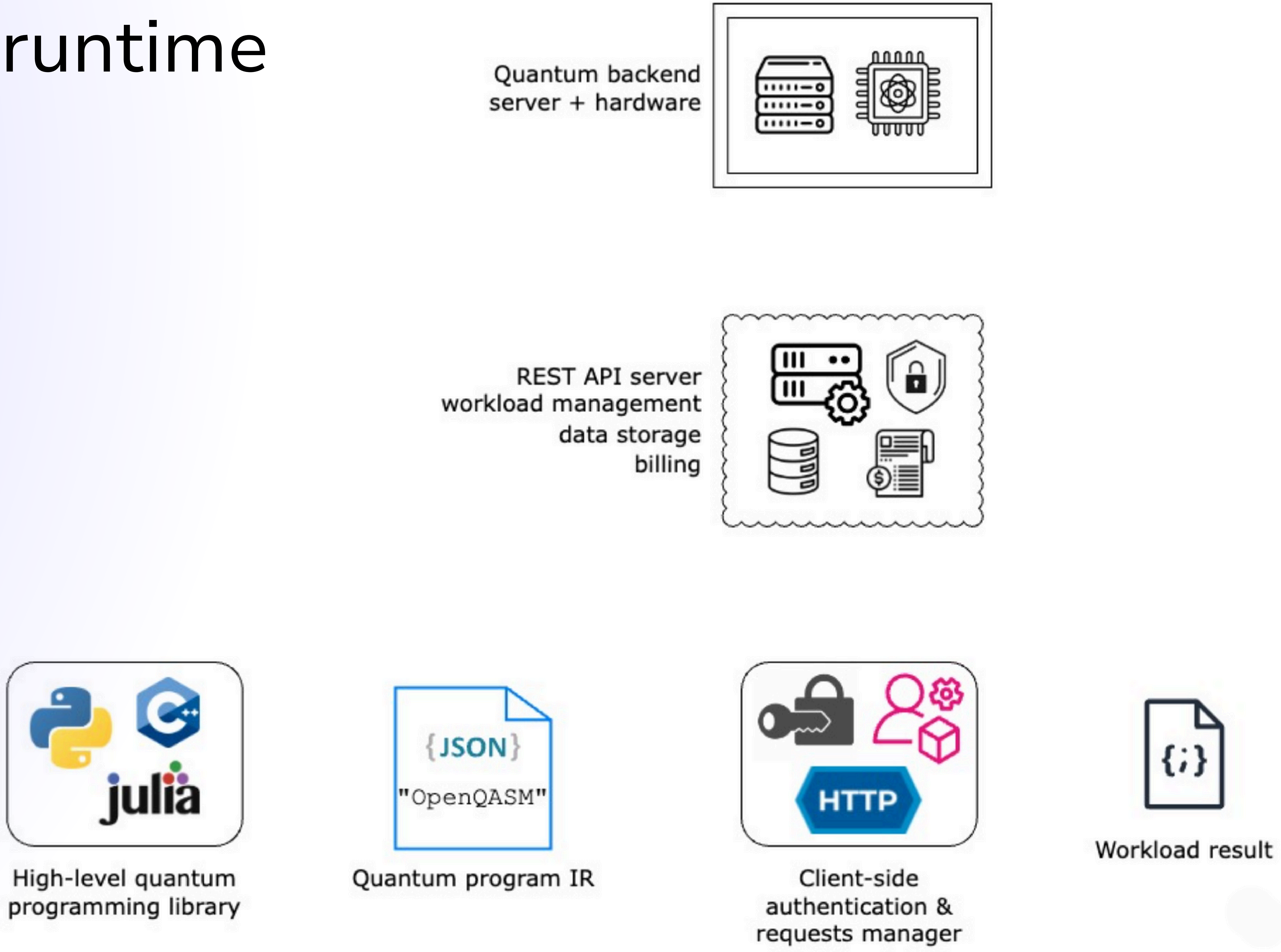


Quantum backend server + hardware

?

High-level quantum programming library

qbraid.runtime



Quantum backend server + hardware

REST API server
workload management
data storage
billing

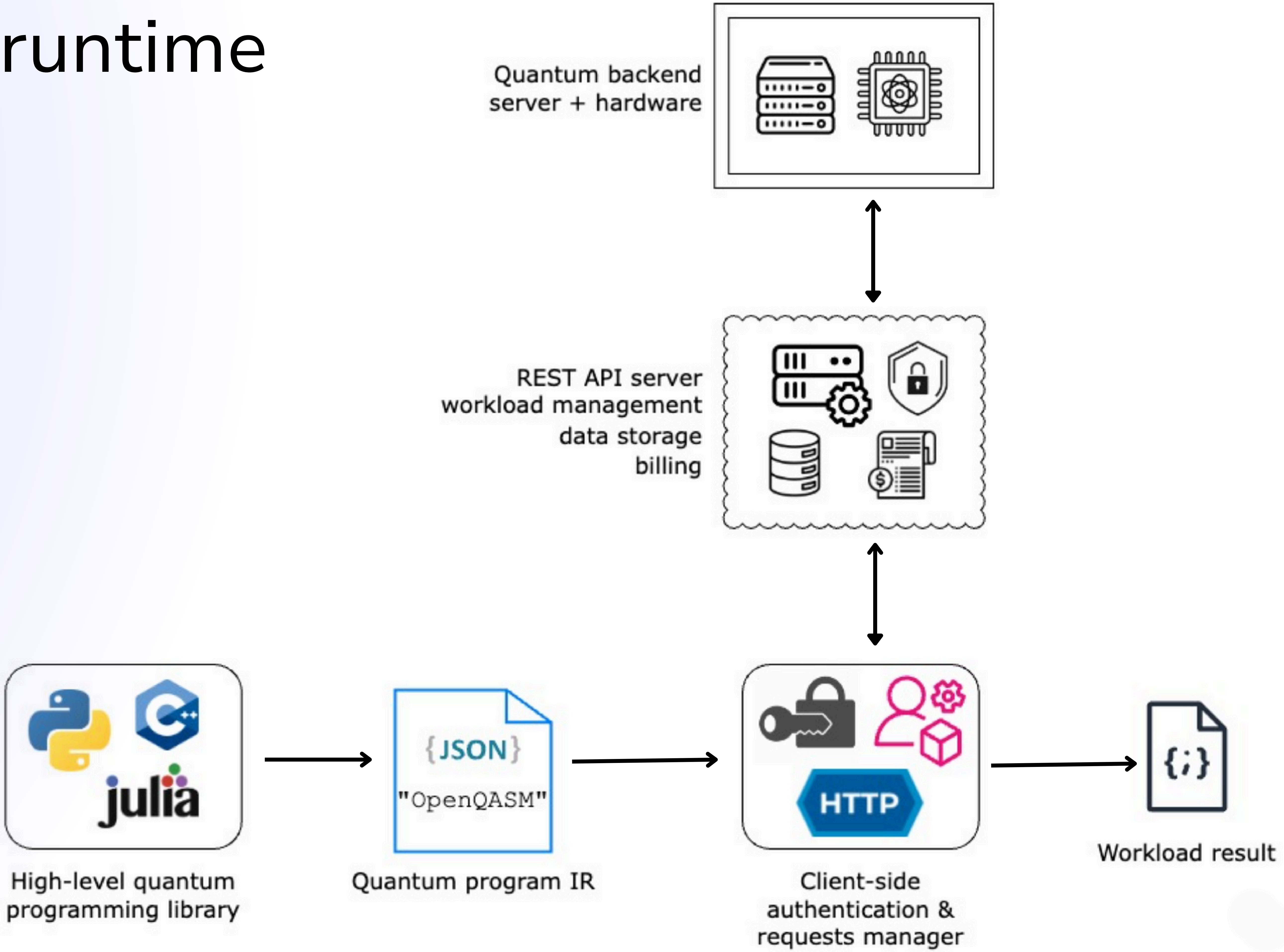
High-level quantum programming library

Quantum program IR

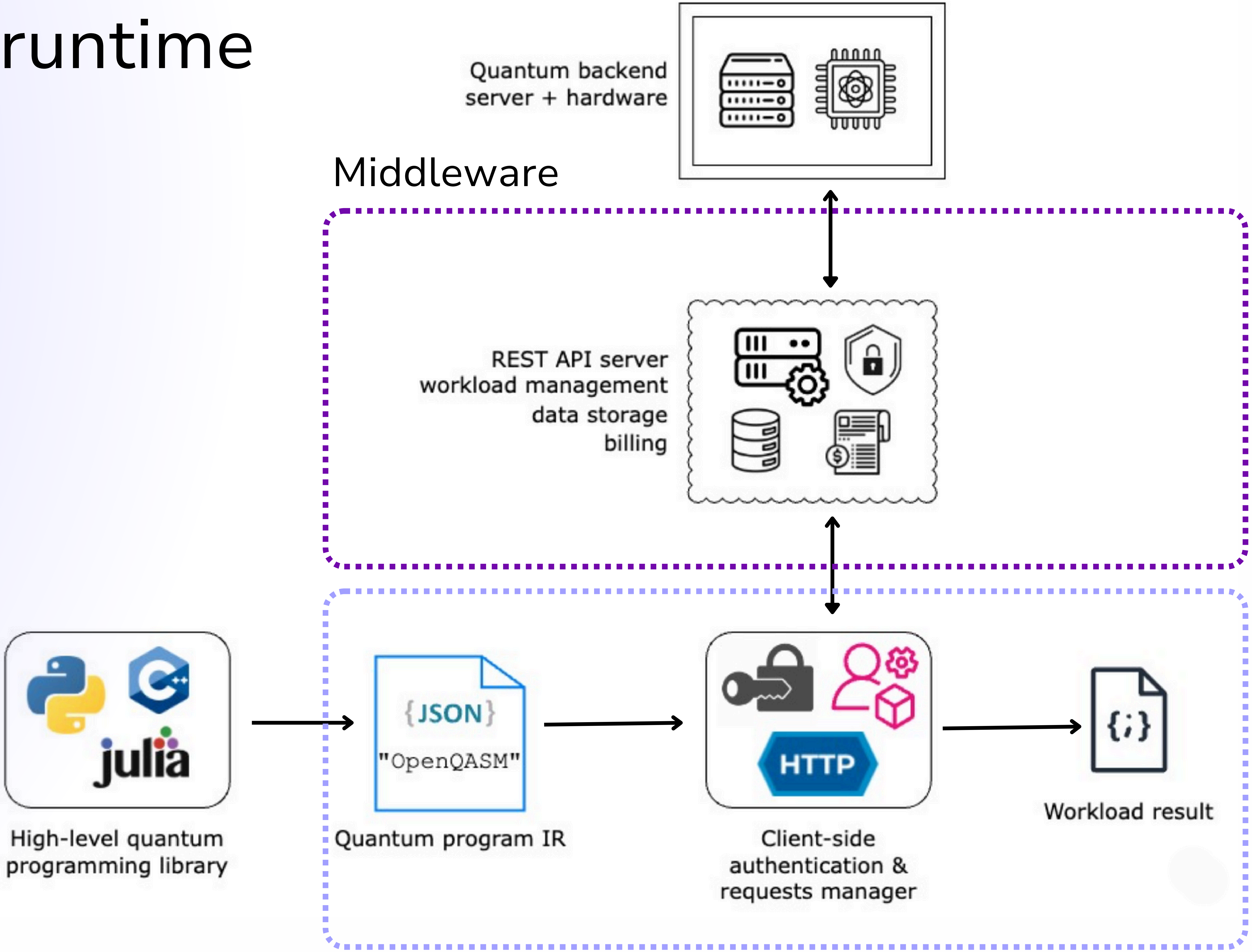
Client-side authentication & requests manager

Workload result

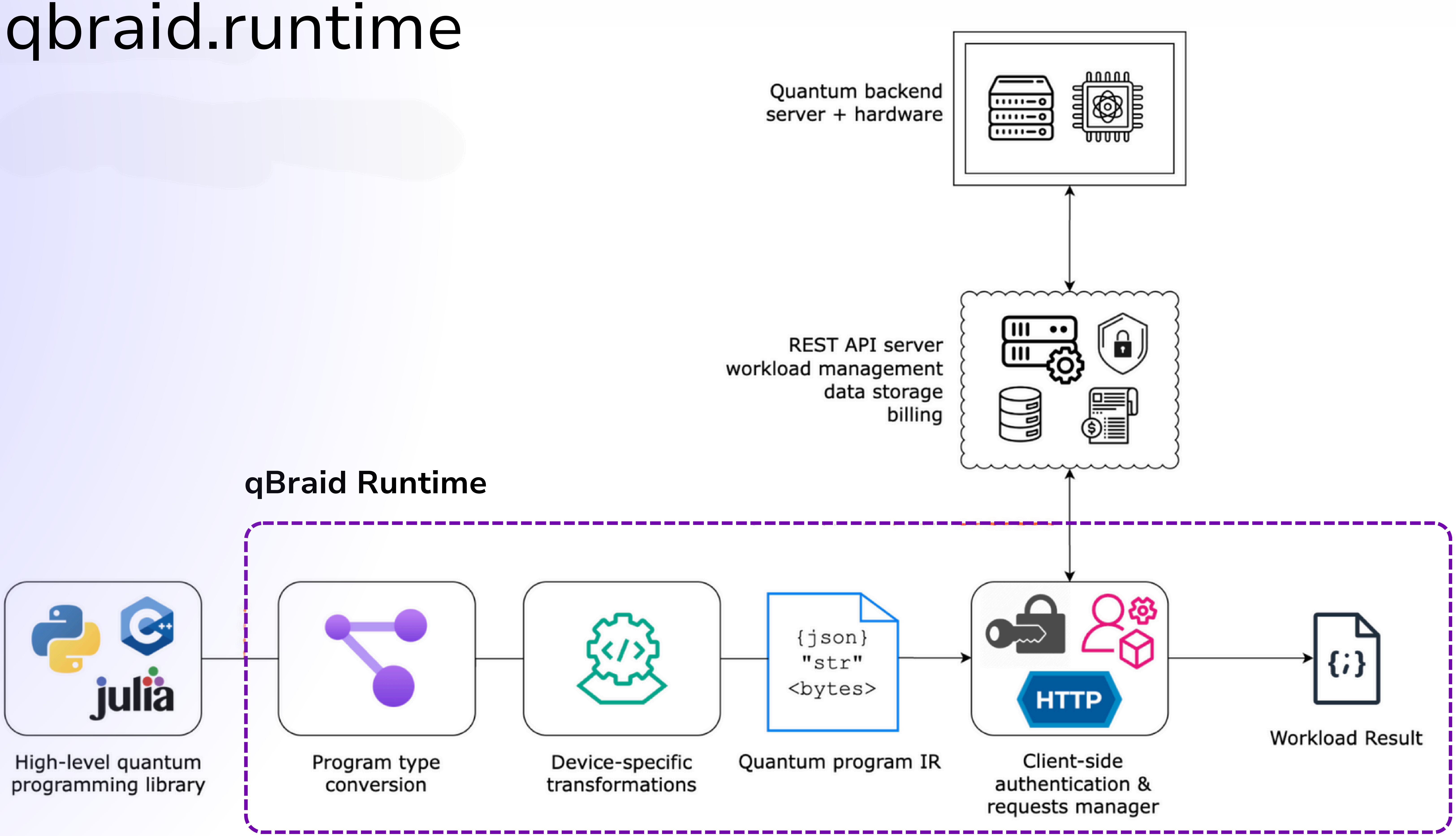
qbraid.runtime



qbraid.runtime

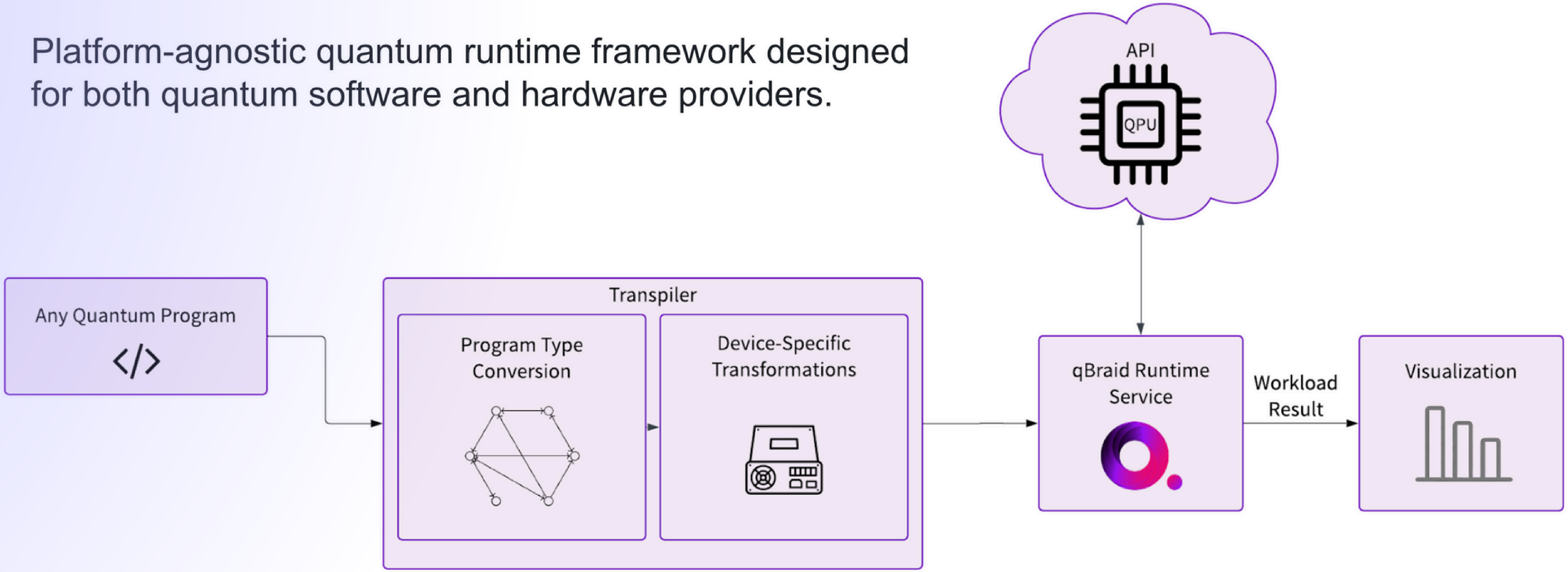


qbraid.runtime



qbraid.runtime

Platform-agnostic quantum runtime framework designed for both quantum software and hardware providers.

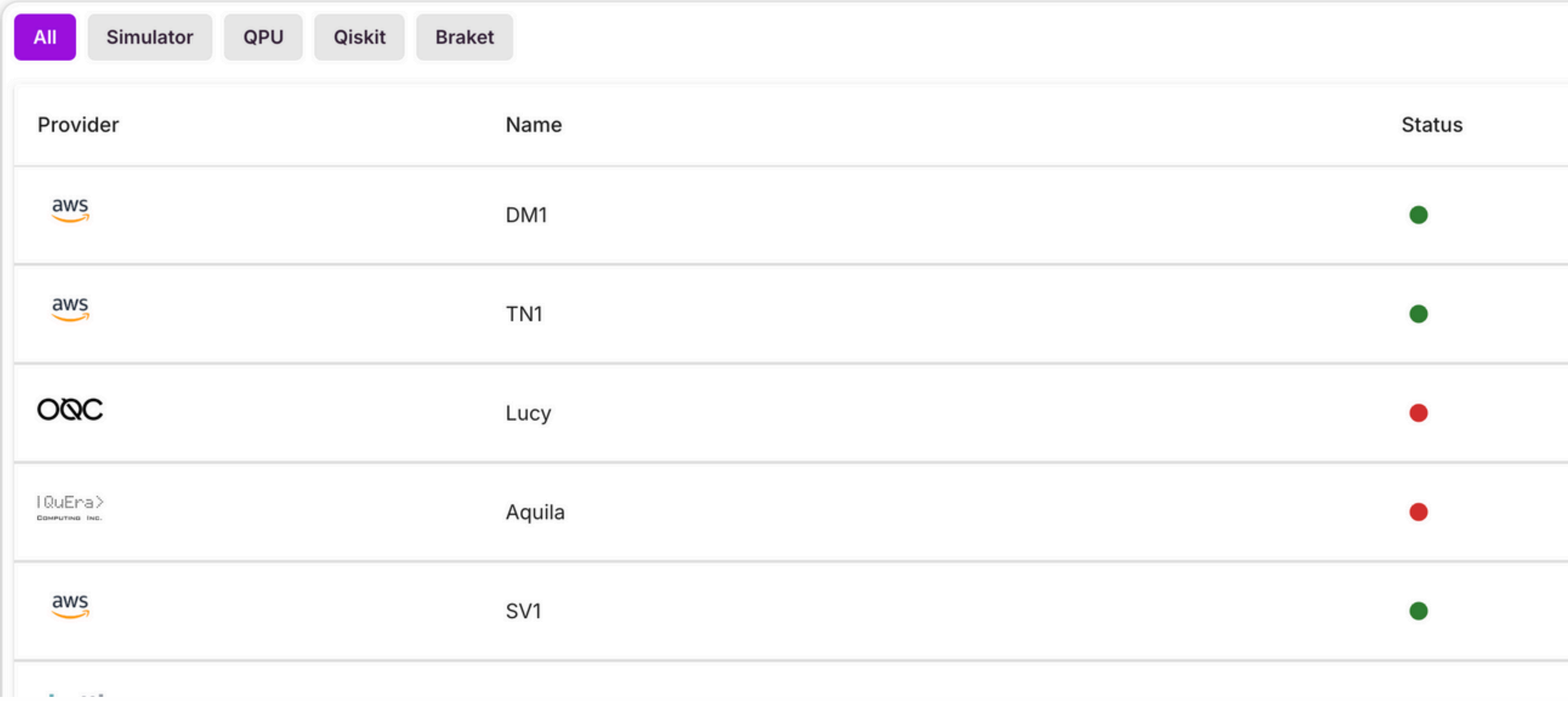


Demo

AGNOSTIC QUANTUM DEVELOPMENT

Code in any package and execute on any backend available with qBraid

Runtime User Guide

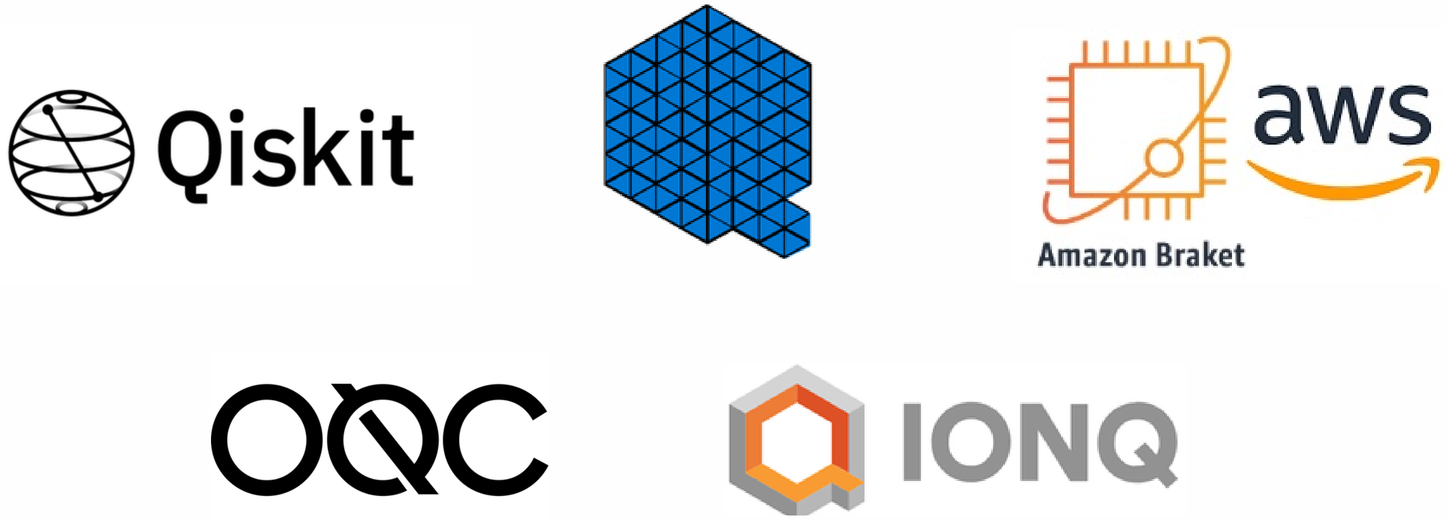


All | Simulator | QPU | Qiskit | Braket

Provider	Name	Status
aws	DM1	●
aws	TN1	●
OQC	Lucy	●
QuEra Computing Inc.	Aquila	●
aws	SV1	●

CURRENT PROVIDERS

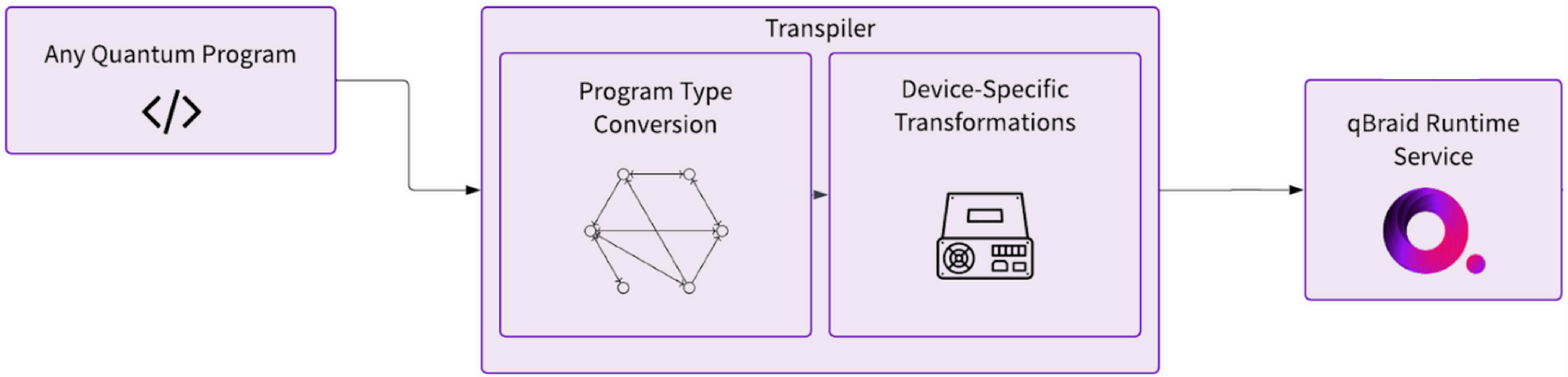
Choose from 5+ different runtime quantum providers and execute circuits across multiple backends



qbraid.transpiler

SEAMLESS QUANTUM PROGRAM CONVERSIONS

WHERE DOES IT FIT IN ?



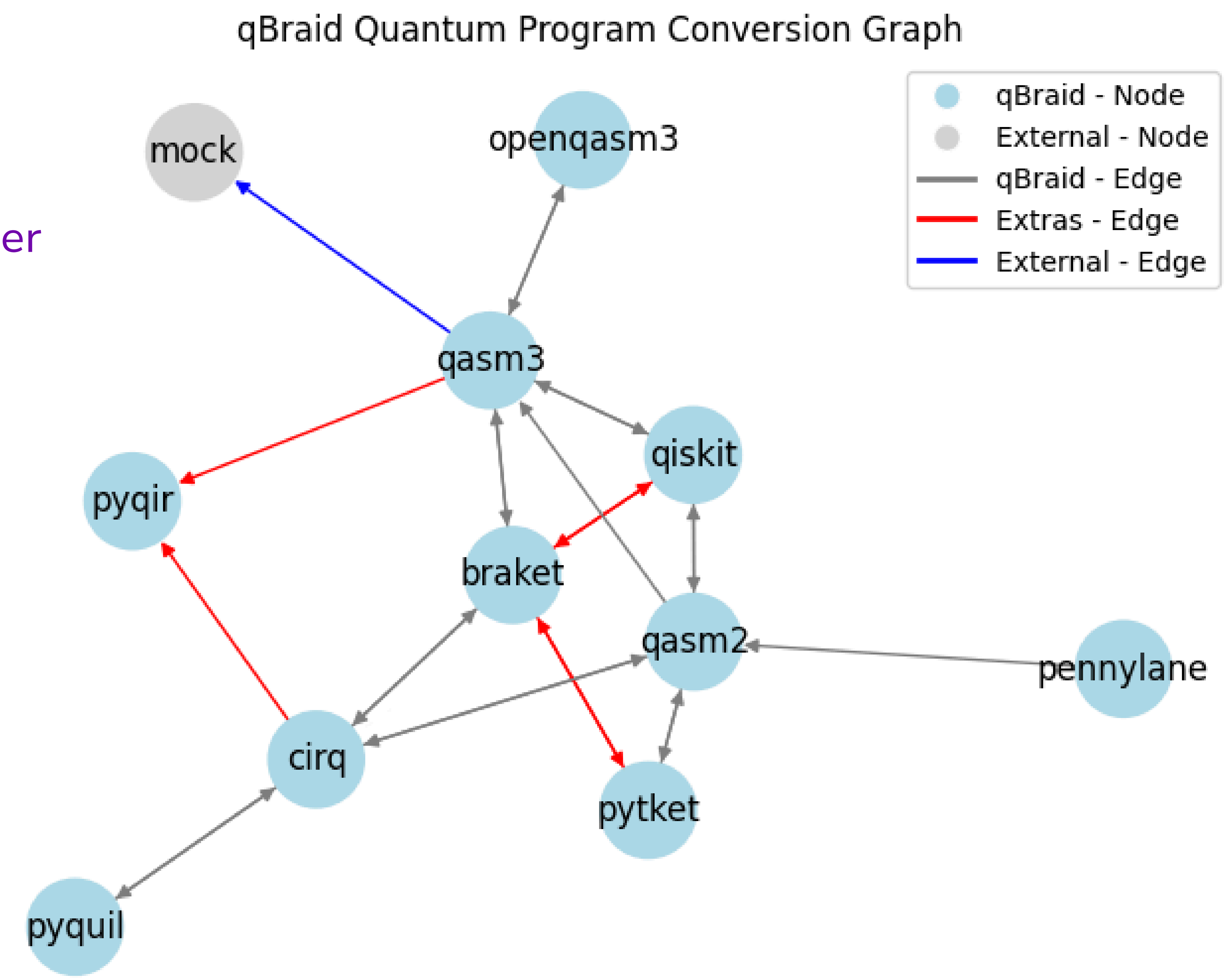
Convert from one target package to another

Enables access to optimization routines internal to a package

Transform quantum programs to QPU requirements

TRANSPILER MODEL

- qBraid has built a graph based transpiler model
- Each package is defined as a node in the transpiler and an edge is a conversion path
- If a path exists from source to the destination package, qBraid will convert the circuit
- Our tests ensure that unitary equivalence is maintained while converting circuits across packages
- Can easily execute a Qiskit circuit on an Amazon backend and vice versa

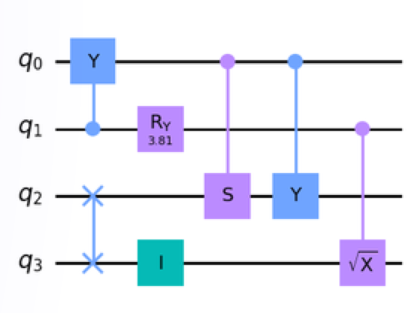


Check out the code!

CONVERSIONS

WRITING A CIRCUIT WITH QISKIT AND CONVERTING TO DIFFERENT PROVIDERS

Qiskit



→

```
qbraid.transpile (src_circuit, target)
```

aws

Amazon Braket

Cirq

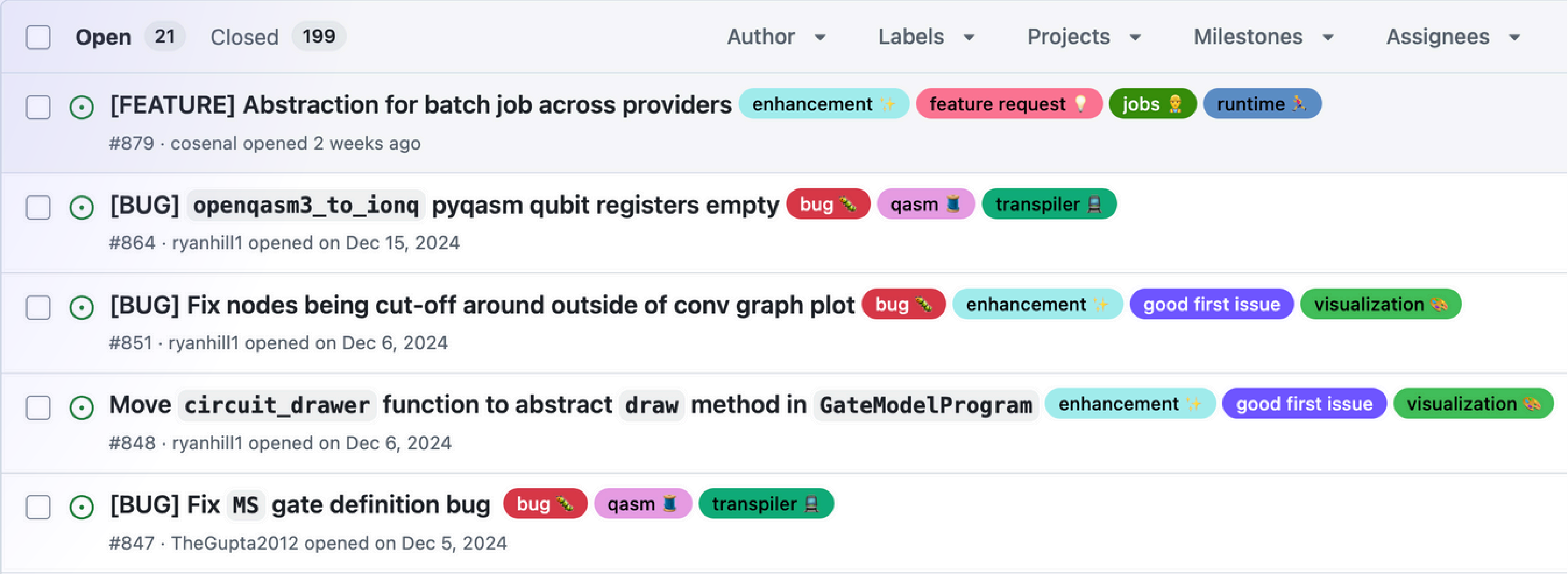
OpenQASM

PENNYLANE

and more!

Demo Notebook

COMMUNITY



□ Open 21 Closed 199 Author ▾ Labels ▾ Projects ▾ Milestones ▾ Assignees ▾

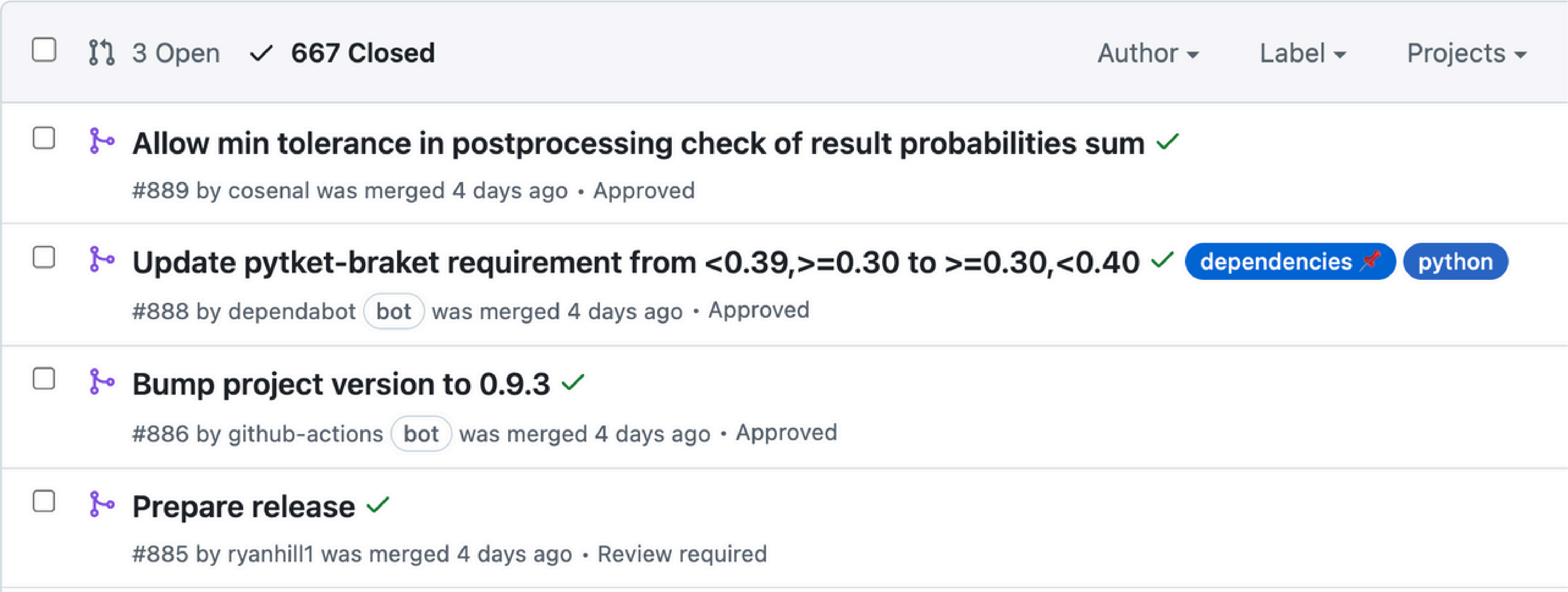
□ ○ [FEATURE] Abstraction for batch job across providers enhancement ✨ feature request 💡 jobs 👷 runtime 🏃
#879 · cosenal opened 2 weeks ago

□ ○ [BUG] `openqasm3_to_ionq` pyqasm qubit registers empty bug 🐛 qasm 🧵 transpiler 🚉
#864 · ryanhill1 opened on Dec 15, 2024

□ ○ [BUG] Fix nodes being cut-off around outside of conv graph plot bug 🐛 enhancement ✨ good first issue visualization 🎨
#851 · ryanhill1 opened on Dec 6, 2024

□ ○ Move `circuit_drawer` function to abstract `draw` method in `GateModelProgram` enhancement ✨ good first issue visualization 🎨
#848 · ryanhill1 opened on Dec 6, 2024

□ ○ [BUG] Fix `MS` gate definition bug bug 🐛 qasm 🧵 transpiler 🚉
#847 · TheGupta2012 opened on Dec 5, 2024



□ 3 Open ✓ 667 Closed Author ▾ Label ▾ Projects ▾

□ Allow min tolerance in postprocessing check of result probabilities sum ✓
#889 by cosenal was merged 4 days ago • Approved

□ Update pytket-braket requirement from <0.39,>=0.30 to >=0.30,<0.40 ✓ dependencies 📌 python
#888 by dependabot bot was merged 4 days ago • Approved

□ Bump project version to 0.9.3 ✓
#886 by github-actions bot was merged 4 days ago • Approved

□ Prepare release ✓
#885 by ryanhill1 was merged 4 days ago • Review required

unitary hack

qosf

Mentorship Programme

PyQASM

pyqasm 0.1.0

```
pip install pyqasm
```

OPENQASM PROGRAM TRANSFORMATION AND ANALYSIS

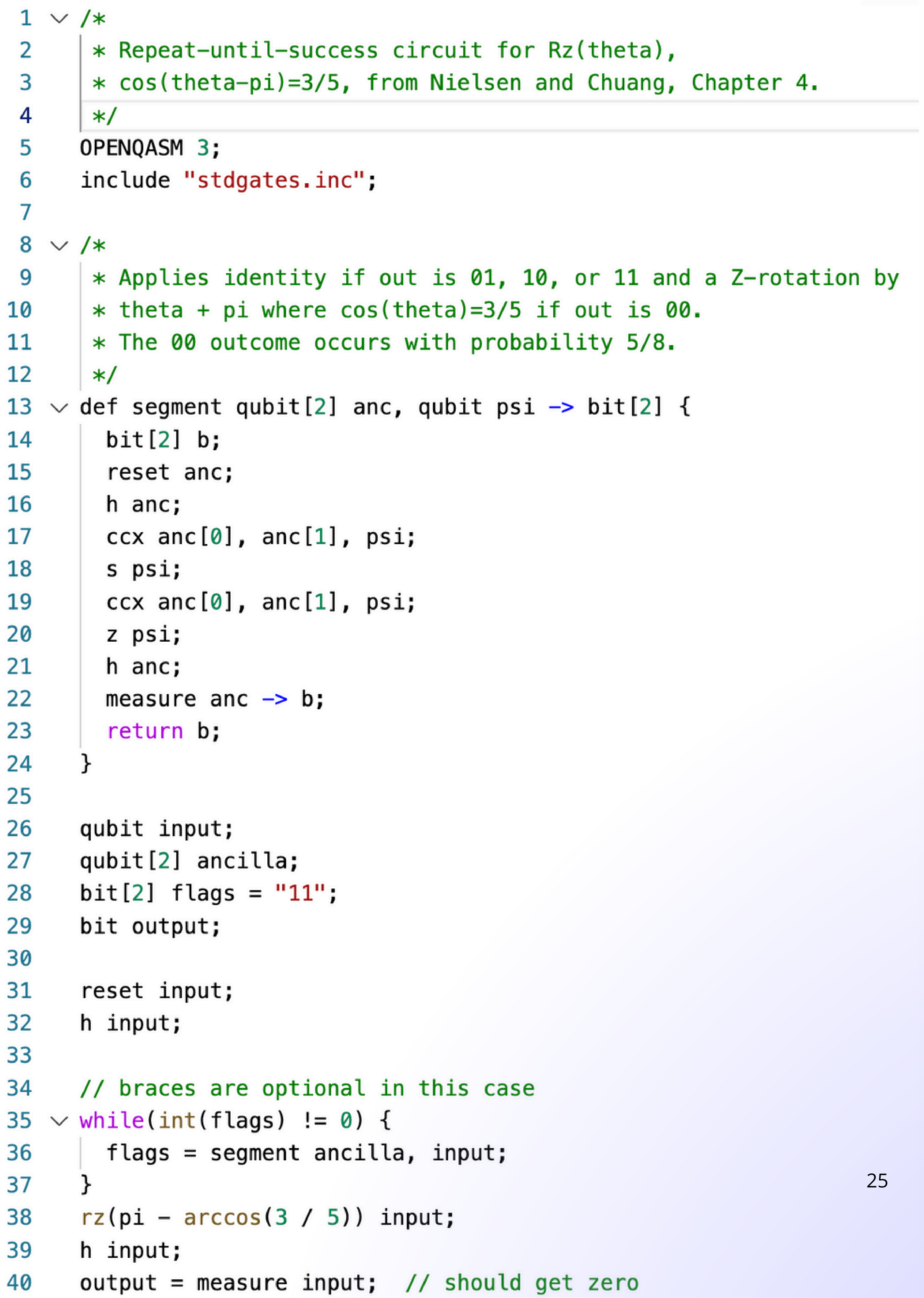
MOTIVATION

OpenQASM (Quantum Assembly Language) is one of the most popular Intermediate Representation (IR) in quantum

QASM v3.x has a very extensive grammar which includes classical control flow support, types, subroutines and more

Lacks a tool with comprehensive semantic analysis and compilation

Most tools like Qiskit provide QASM support *coupled* with the package



```
/*
 * Repeat-until-success circuit for Rz(theta),
 * cos(theta-pi)=3/5, from Nielsen and Chuang, Chapter 4.
 */
OPENQASM 3;
include "stdgates.inc";

/*
 * Applies identity if out is 01, 10, or 11 and a Z-rotation by
 * theta + pi where cos(theta)=3/5 if out is 00.
 * The 00 outcome occurs with probability 5/8.
 */
def segment qubit[2] anc, qubit psi -> bit[2] {
  bit[2] b;
  reset anc;
  h anc;
  ccx anc[0], anc[1], psi;
  s psi;
  ccx anc[0], anc[1], psi;
  z psi;
  h anc;
  measure anc -> b;
  return b;
}

qubit input;
qubit[2] ancilla;
bit[2] flags = "11";
bit output;

reset input;
h input;

// braces are optional in this case
while(int(flags) != 0) {
  flags = segment ancilla, input;
}
rz(pi - arccos(3 / 5)) input;
h input;
output = measure input;  // should get zero
```

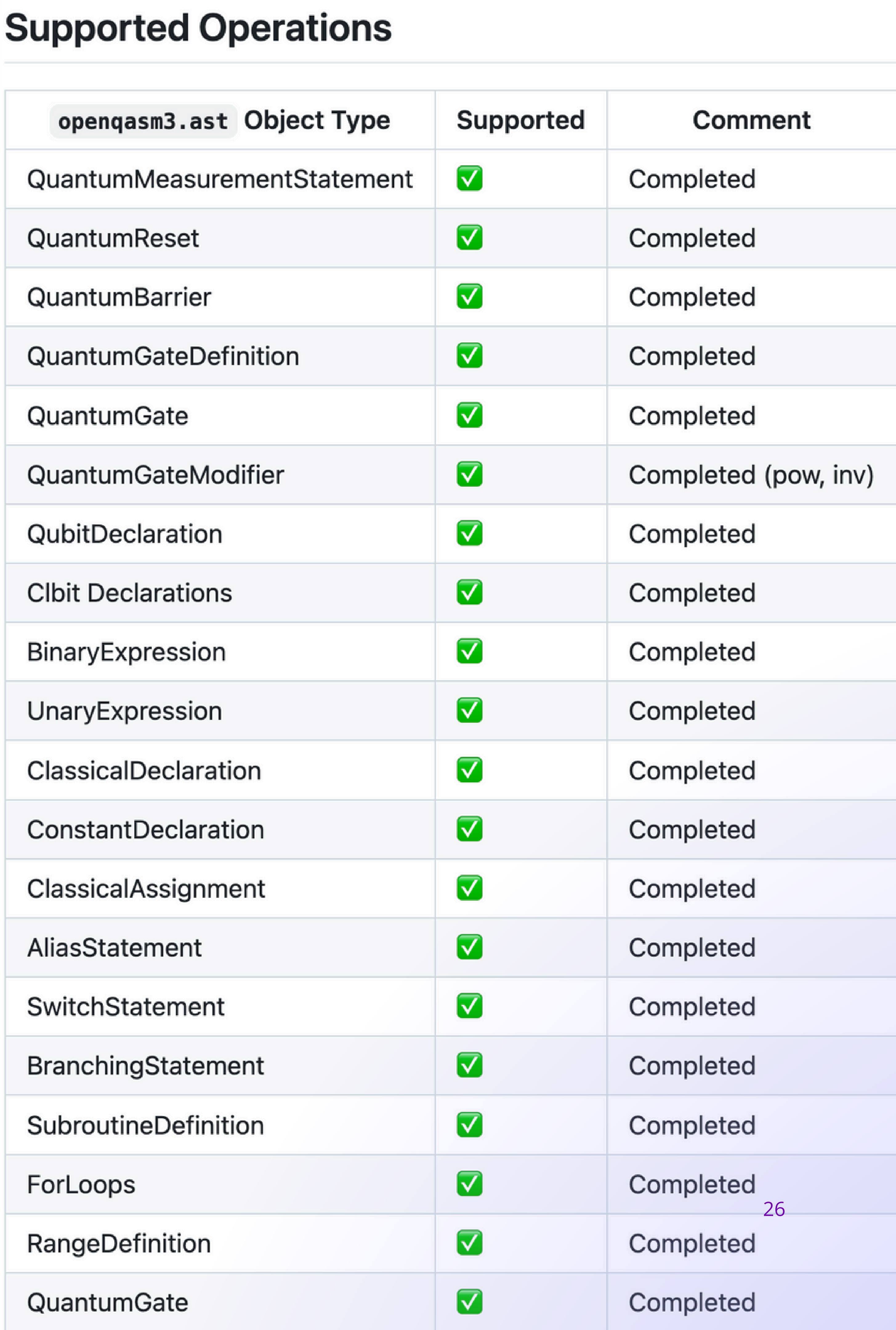
CURRENT FEATURES

Extensive support for semantic analysis of QASM v2.x and v3.x

Program unrolling and validation with support for loops, branches, subroutines and custom gate definitions

Support for program transformation and analysis

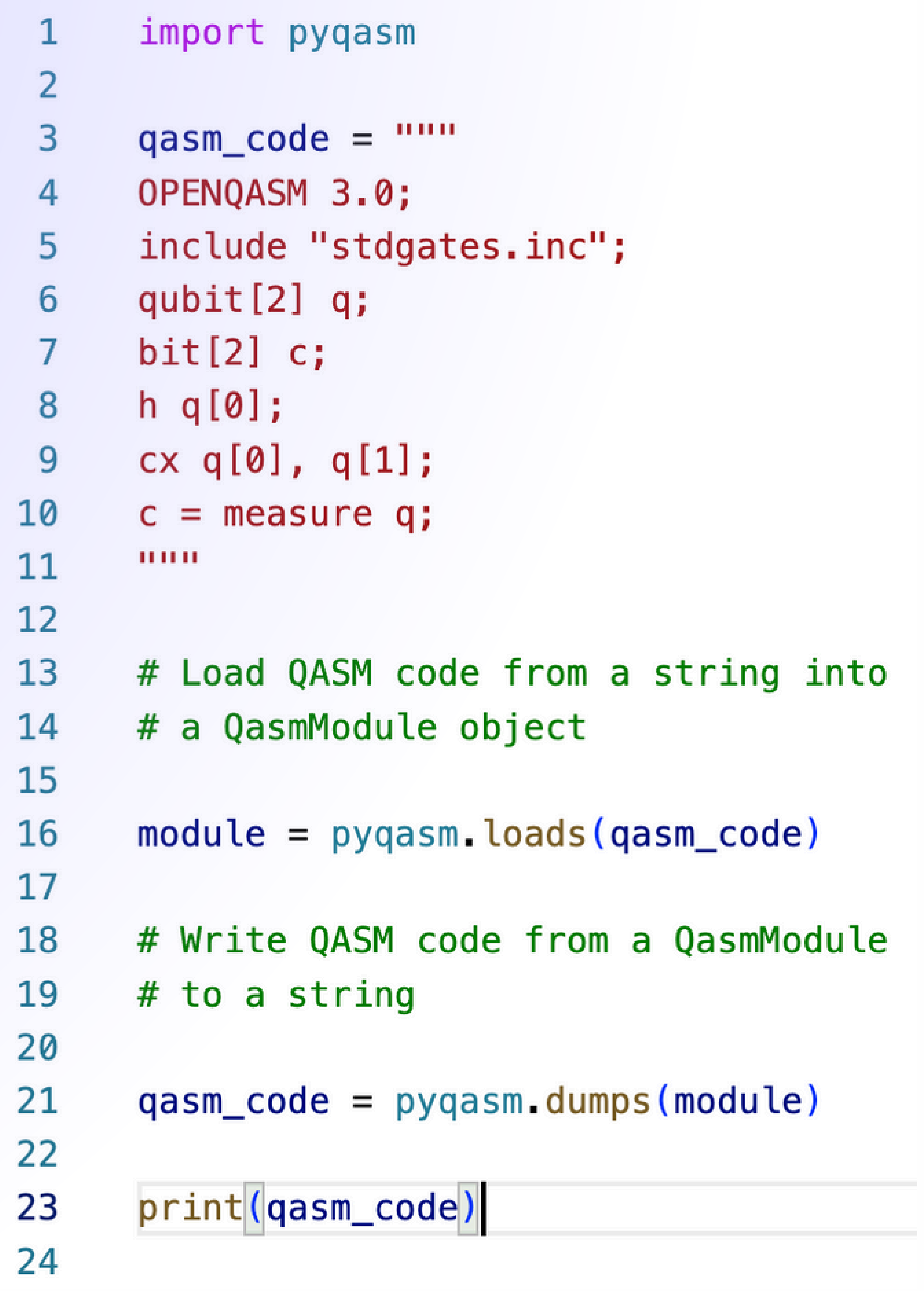
CLI tool providing semantic validation in terminal



Supported Operations

`openqasm3.ast` Object Type	Supported	Comment
QuantumMeasurementStatement	✅	Completed
QuantumReset	✅	Completed
QuantumBarrier	✅	Completed
QuantumGateDefinition	✅	Completed
QuantumGate	✅	Completed
QuantumGateModifier	✅	Completed (pow, inv)
QubitDeclaration	✅	Completed
Clbit Declarations	✅	Completed
BinaryExpression	✅	Completed
UnaryExpression	✅	Completed
ClassicalDeclaration	✅	Completed
ConstantDeclaration	✅	Completed
ClassicalAssignment	✅	Completed
AliasStatement	✅	Completed
SwitchStatement	✅	Completed
BranchingStatement	✅	Completed
SubroutineDefinition	✅	Completed
ForLoops	✅	Completed
RangeDefinition	✅	Completed
QuantumGate	✅	Completed

USAGE



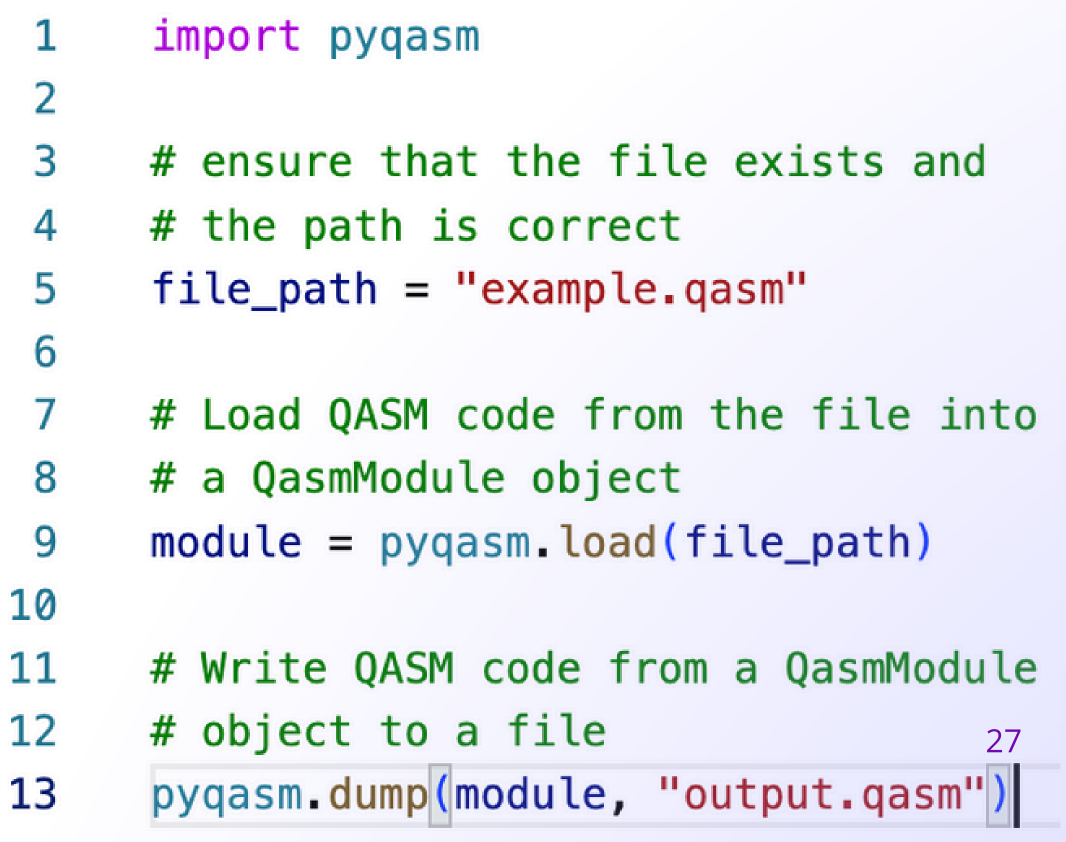
```python
import pyqasm

qasm_code = """
OPENQASM 3.0;
include "stdgates.inc";
qubit[2] q;
bit[2] c;
h q[0];
cx q[0], q[1];
c = measure q;
"""

# Load QASM code from a string into
# a QasmModule object

module = pyqasm.loads(qasm_code)

# Write QASM code from a QasmModule
# to a string

qasm_code = pyqasm.dumps(module)

print(qasm_code)
```

loads and dumps

The loads and dumps functions are used to read QASM code from a string and write QASM code to a string, respectively.

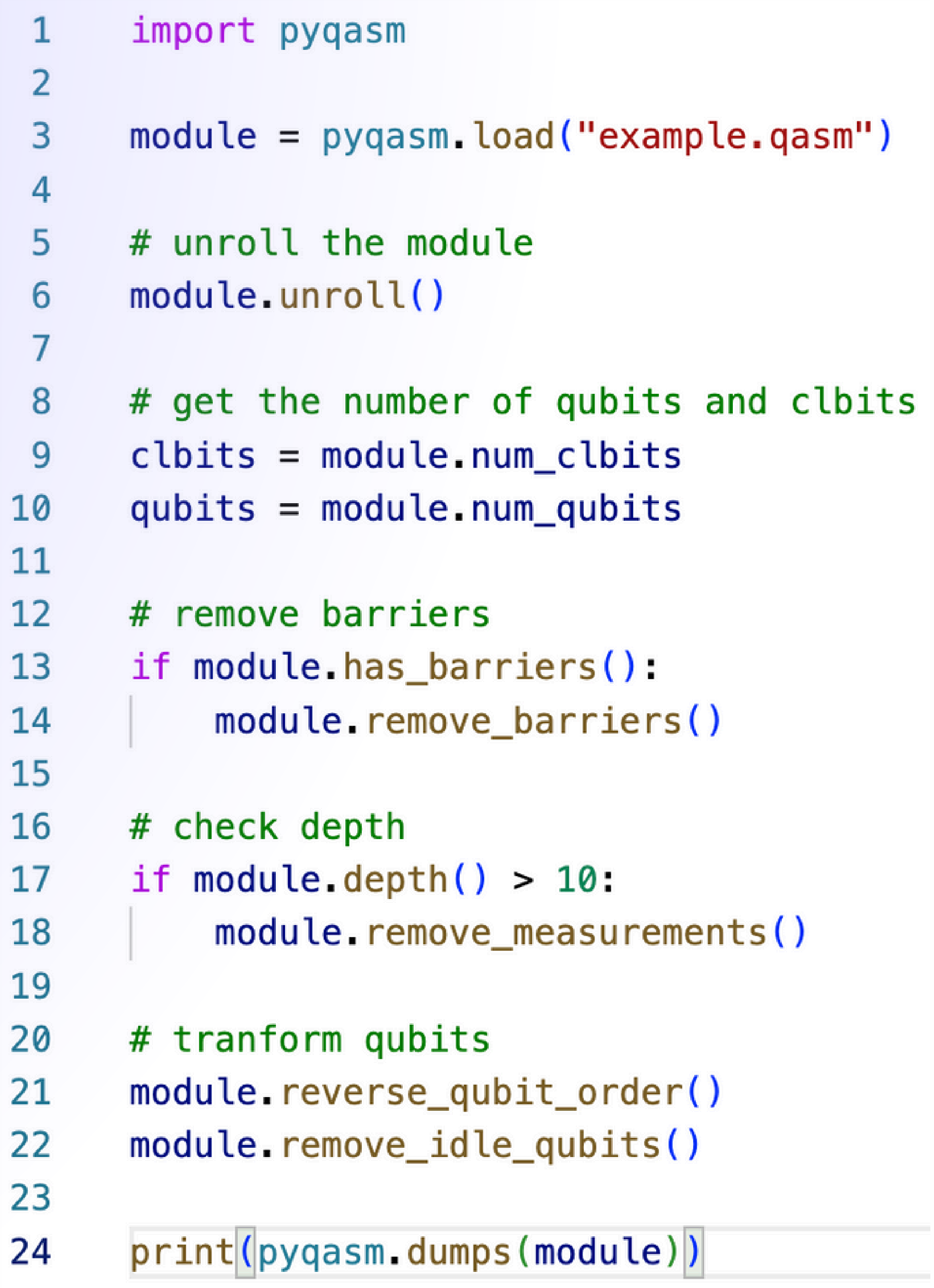
load and dump

Similar functionality as above, with QASM files are source and sink



```python
import pyqasm

# ensure that the file exists and
# the path is correct
file_path = "example.qasm"

# Load QASM code from the file into
# a QasmModule object
module = pyqasm.load(file_path)

# Write QASM code from a QasmModule
# object to a file
pyqasm.dump(module, "output.qasm")
```

USAGE



```python
import pyqasm

module = pyqasm.load("example.qasm")

# unroll the module
module.unroll()

# get the number of qubits and clbits
clbits = module.num_clbits
qubits = module.num_qubits

# remove barriers
if module.has_barriers():
    module.remove_barriers()

# check depth
if module.depth() > 10:
    module.remove_measurements()

# tranform qubits
module.reverse_qubit_order()
module.remove_idle_qubits()

print(pyqasm.dumps(module))
```

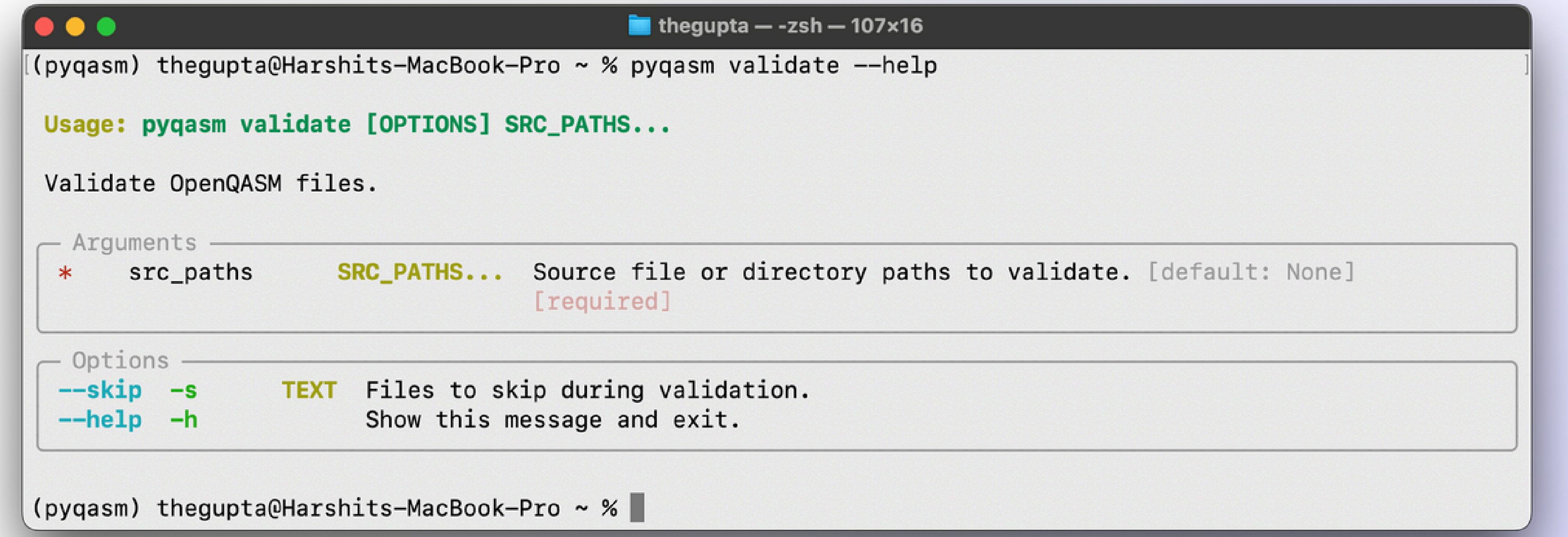
Operation Chaining

Pyqasm stores the program repr inside a QasmModule object

All operations are performed *in-place*, unless specified otherwise, and can be *chained*

CLI Tool

Use the CLI tool for validating single / multiple QASM programs



```
(pyqasm) thegupta@Harshits-MacBook-Pro ~ % pyqasm validate --help

 Usage: pyqasm validate [OPTIONS] SRC_PATHS...

 Validate OpenQASM files.

╭─ Arguments ───────────────────────────────────────────────────────────────────────────────────╮
│ *    src_paths      SRC_PATHS...  Source file or directory paths to validate. [default: None]  │
│                                   [required]                                                   │
╰────────────────────────────────────────────────────────────────────────────────────────────────╯
╭─ Options ─────────────────────────────────────────────────────────────────────────────────────╮
│ --skip  -s      TEXT  Files to skip during validation.                                         │
│ --help  -h            Show this message and exit.                                              │
╰────────────────────────────────────────────────────────────────────────────────────────────────╯

(pyqasm) thegupta@Harshits-MacBook-Pro ~ %
```

References

- qBraid-SDK
- qbraid.runtime docs
- pyqasm
- qbraid-lab
- openqasm spec
- qbraid.runtime demo
- Unitary Hack ‘24

Contact

Email Address

contact@qbraid.com

Phone Number

(248) 795 9039

Twitter

@qbraid_official

Our Website

qbraid.com

Location

111 S Wacker Dr, Chicago, IL 60606,
United States

Linkedin

qbraid_official

Access Code for qBraid

EHNU6626

Thank you!