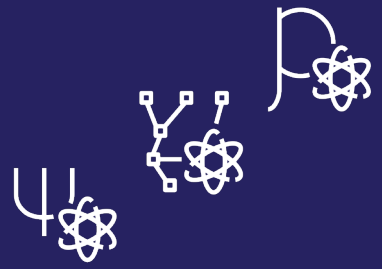
Amazon Braket Workshop iQuHack 2024

Amazon Braket – the AWS quantum computing service

A fully managed service that makes it easy for scientists and developers to explore quantum computing

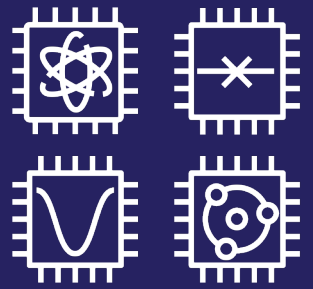
Build

- Amazon Braket SDK
- Jupyter notebooks
- Command line interface



Test

- Local simulators for rapid testing
- High-performance simulators



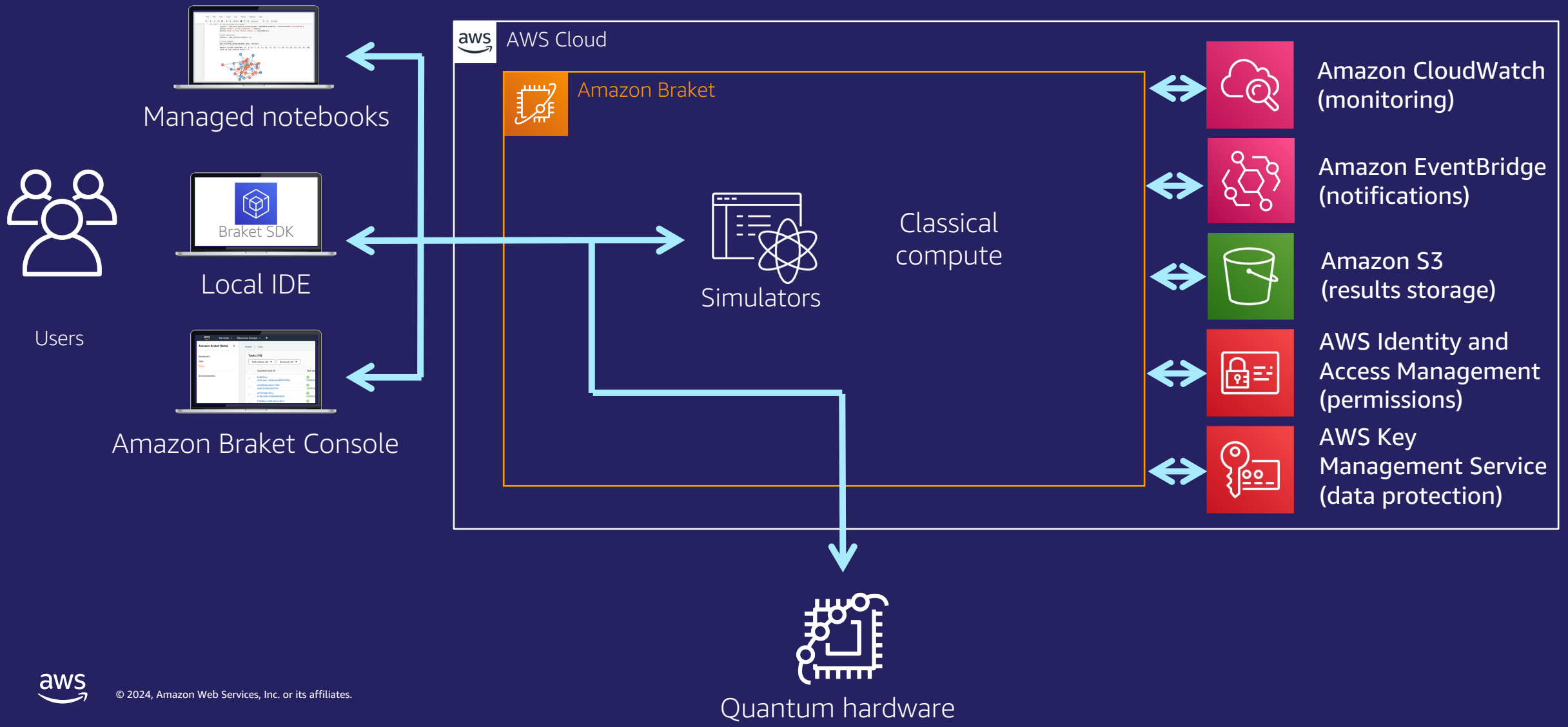
Run

- Access multiple quantum computers
- Combine quantum and classical resources

Analyze

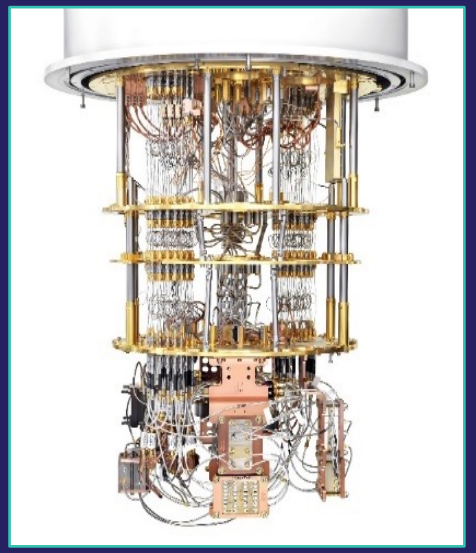
- Monitor algorithms in almost real time
- Analyze algorithm results and performance

Amazon Braket architecture



Available Quantum Computers

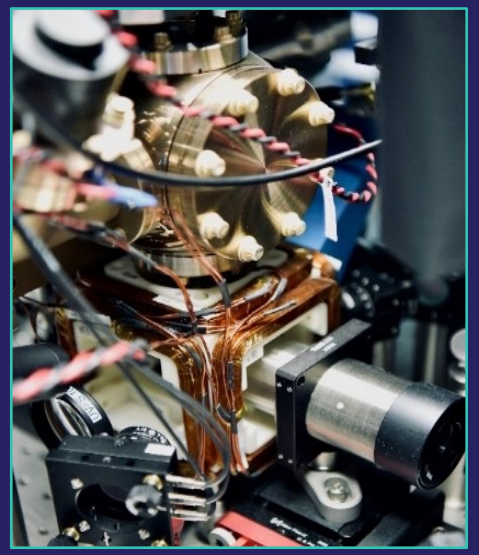
IONQ



rigetti

OQC

(Oxford Quantum Circuits)

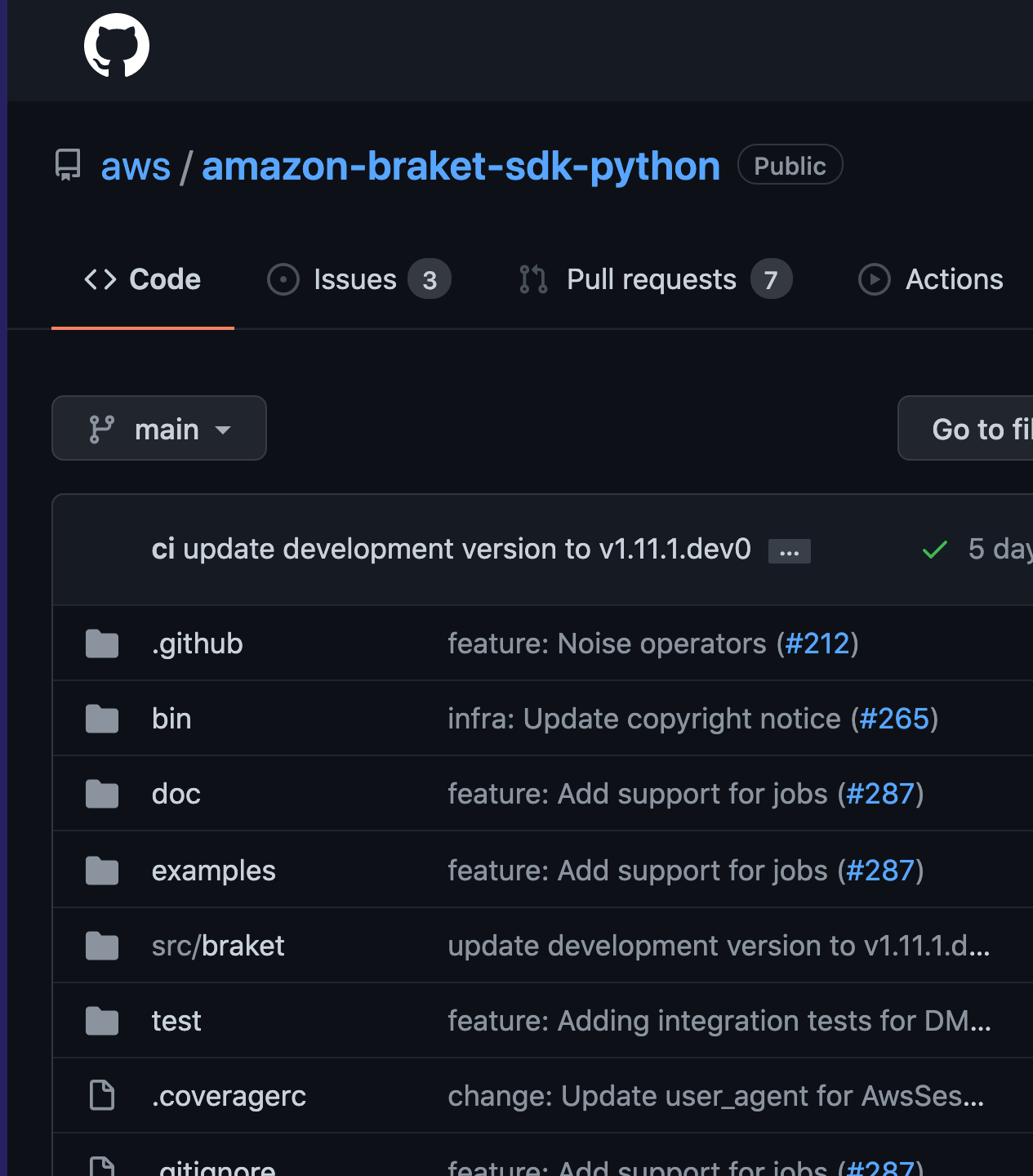


IQuEra Computing Inc.

The Amazon Braket SDK

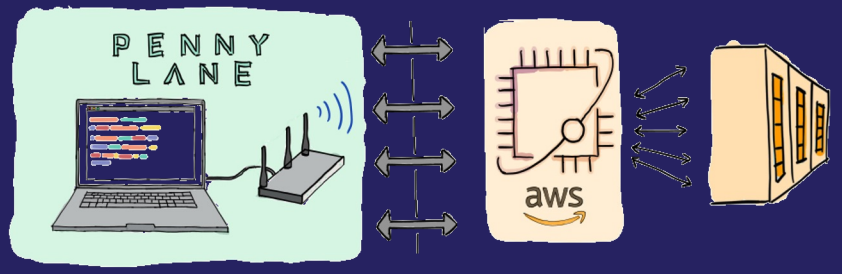
Open source Python library

- Design and build quantum circuits
- Submit tasks and jobs to devices
- Track and monitor their execution
- Local installation
  - pip install amazon-braket-sdk
- Three local simulators (free)
  - braket_sv (pure state vector simulation)
  - braket_dm (noisy density matrix simulation)
  - braket_ahs (analog Hamiltonian simulation)



The Amazon Braket SDK

Enhance functionality with open-source plugins



A cross-platform Python library for differentiable programming of quantum computers.

Train a quantum computer the same way as a neural network.

https://pennylane.ai/qml

Run Amazon Braket code from Qiskit development toolkit

https://github.com/qiskit-community/qiskit-braket-provider

qiskit_connector.py

```python
from qiskit_braket_provider import BraketLocalBackend

local_simulator = BraketLocalBackend()
task = local_simulator.run(circuit, shots=1000)

```

Get training at your own pace with the Amazon Braket Badge!

AWS Quantum Technologies Blog

Introducing the Amazon Braket Learning Plan and Digital Badge

by James Whitfield, Sebastian Hassinger, and Zia Mohammad | on 27 NOV 2023 | in Amazon Braket, Quantum Technologies | Permalink | Share

Available today, quantum computing developers, educators, and enthusiasts can learn the foundations of quantum computing on Amazon Web Services (AWS) with the Amazon Braket Digital Learning Plan and earn their own Digital badge – at no additional cost. You earn the badge after completing a series of learning courses and scoring at least 80% on an assessment measuring your knowledge of Amazon Braket, the quantum computing service of AWS.

Quantum computing is an exciting new field. Today, Amazon Braket enables developers to explore early applications of this technology by providing access to a range of different quantum hardware and tooling for programming quantum computers and running hybrid quantum-classical algorithms. As an emerging technology, customers asked us for a better way to get started.

To address this need, together with AWS Training and Certification, we are launching the Amazon Braket Learning Plan which is designed to provide you with the foundational knowledge needed to quickly get up and running with Braket. If you already know how to use Amazon Braket, you can jump right in and take the assessment.

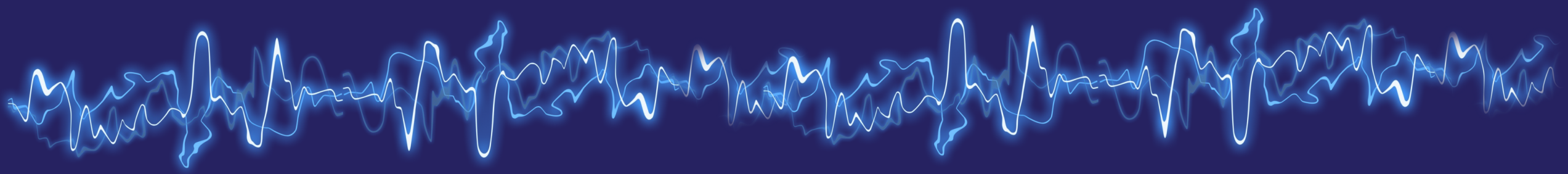
Amazon Braket Learning Plan

Demo:

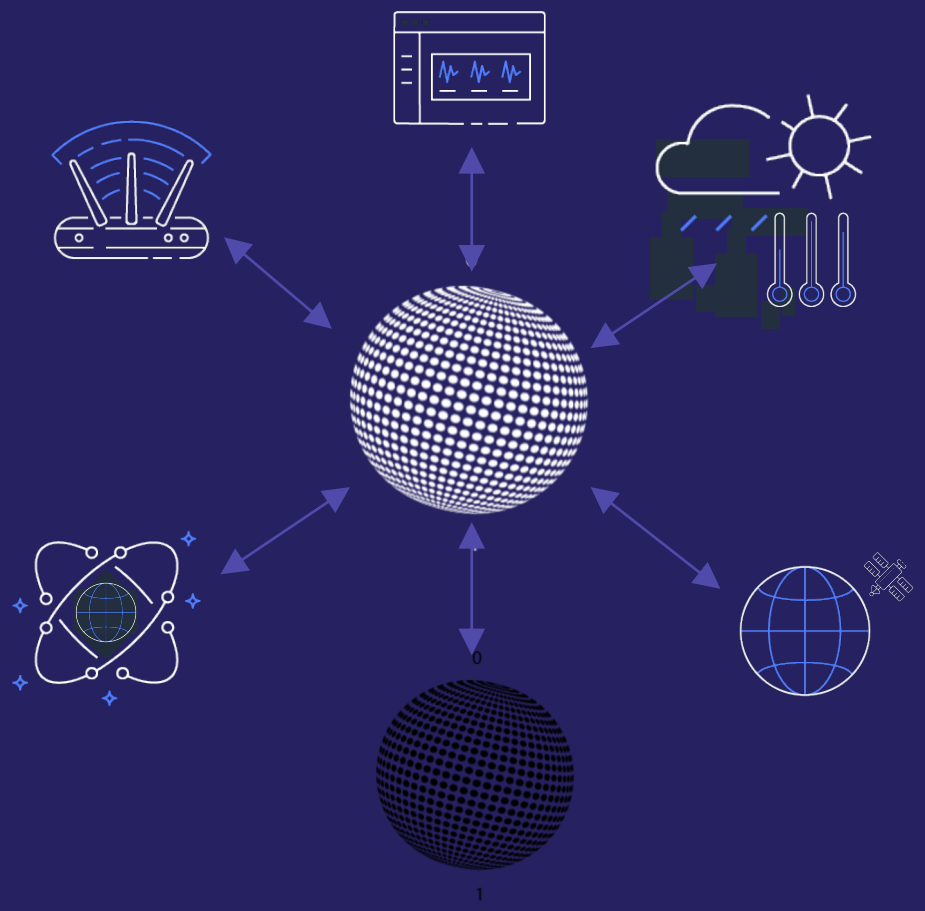
Accessing Amazon Braket Jupyter Notebooks with QBraid

Running your first tasks on simulators and QPUs

Noise in Quantum Computing



Basic concepts



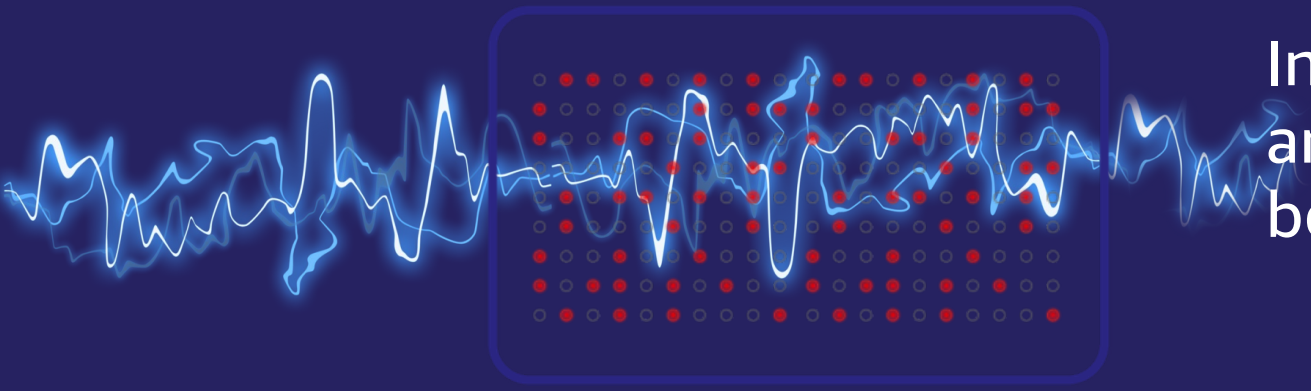
Noise causes computers to malfunction, making computations hard.

Qubits are susceptible to noise from external factors or other qubits.

Decoherence is the degradation of the qubit state, resulting on randomized or erased information.

Classical vs. Quantum

In a digital computer, transistor lifespan is 1,000,000,000(*) years at 1,000,000,000(*) operations per second.

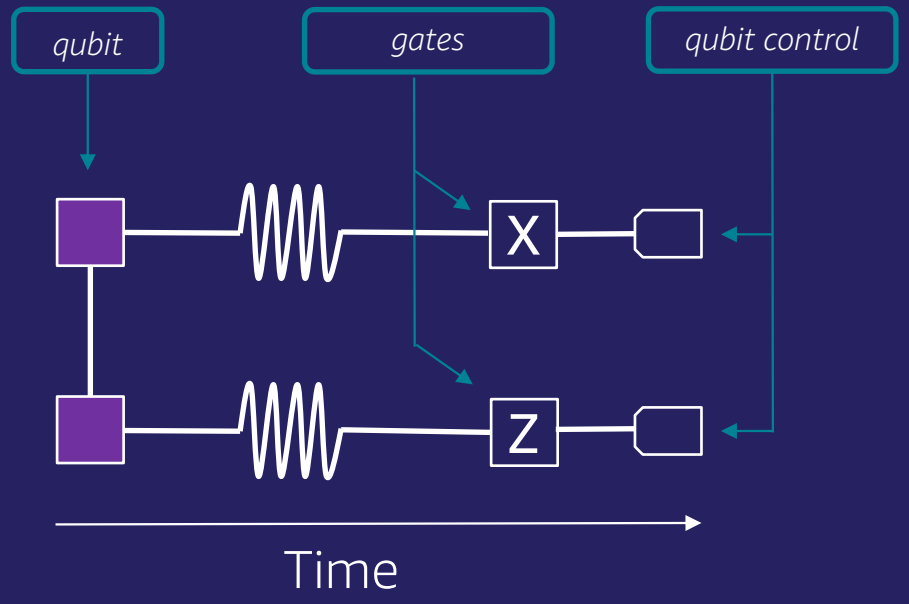


In a quantum computer, the qubit has an average lifespan of 0.001 seconds. before collapsing.

(*) Read as "one thousand million" in some countries

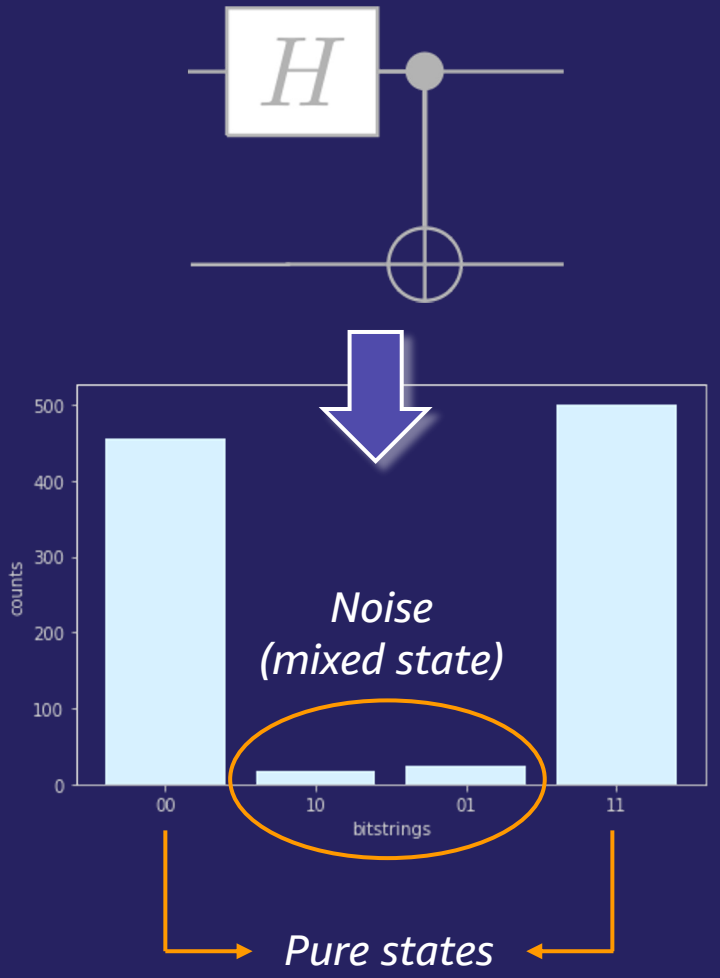
The circuit model

$$H(t)|\psi(t)\rangle = i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle$$



Operator	Gate(s)	Matrix
Pauli-X (X)	X	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)	Y	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)	Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)	H	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)	S	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)	T	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)	Z	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

Pure and mixed quantum states



Logical gates are the toolbox to describe pure quantum states.

Noise and decoherence cannot be described using this framework.

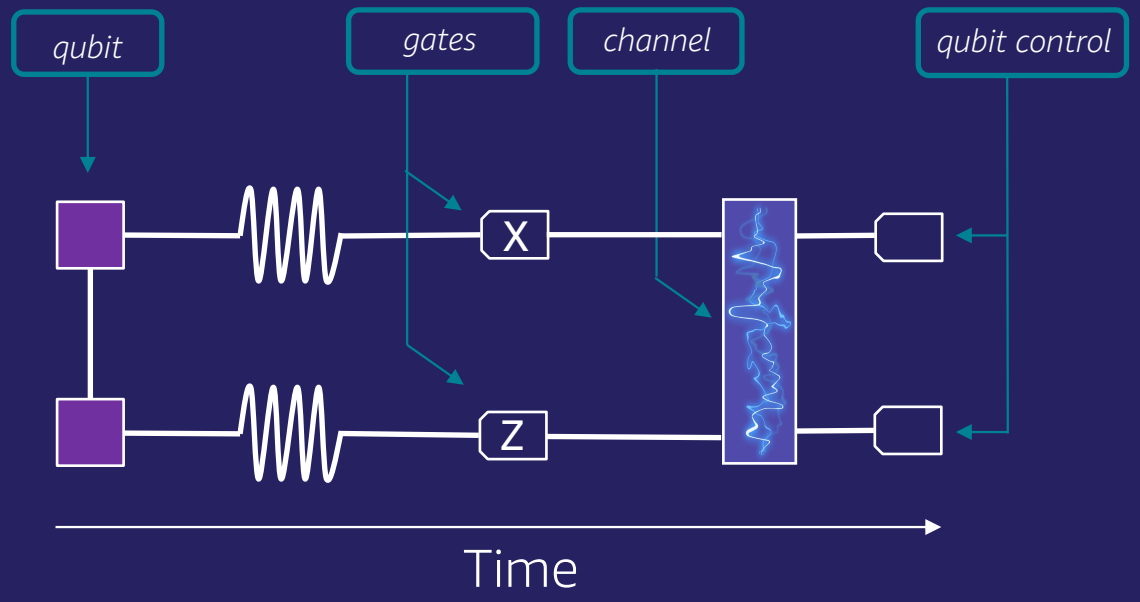
The density matrix (ρ) formalism provides the same language to describe any quantum state.

Gates and channels

Quantum channels add flexibility and allow noise and mixed states representation.

They describe the quantum system time evolution as a density matrix.

Amazon Braket follows the Kraus representation to represent transformations.



$$\mathcal{N}(\rho) = \sum_i s_i K_i \rho K_i^\dagger$$

Amazon Braket Simulators

Local simulator (SDK)

Pre-installed with Braket SDK

Fast and convenient prototyping

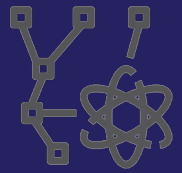
Number of qubits based on hardware

State Vector simulator (SV1)

Quantum circuit with up to 34 qubits

Stores the full wave function state

Concurrency: Default 35, max 50



Tensor Net. simulator (TN1)

Quantum circuit with up to 50 qubits

Encodes quantum circuits into a structured graph

Concurrency: Default 10, max 10.

Density Matrix simulator (DM1)

Quantum circuit with up to 17 qubits

Run multiple circuits in parallel with noise simulation

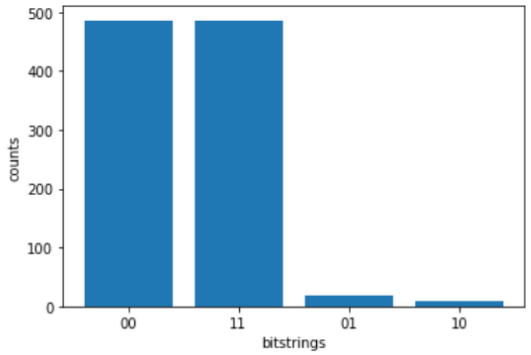
Concurrency: Default 35, max 50.

The Local Simulator

```
# define a noise channel
noise = Noise.BitFlip(probability=0.01)

# add noise to every gate in the circuit
bell.apply_gate_noise(noise)

# select the local noise simulator
device = LocalSimulator('braket_dm')

# run the circuit on the local simulator
task = device.run(bell, shots = 1000)

# visualize the results
result = task.result()
counts = result.measurement_counts
print('measurement results:', counts)
```

```
measurement results: Counter({'00': 486, '11': 485, '01': 19, '10': 10})
```

```
# plot using Counter
plt.bar(counts.keys(), counts.values());
plt.xlabel('bitstrings');
plt.ylabel('counts');
```

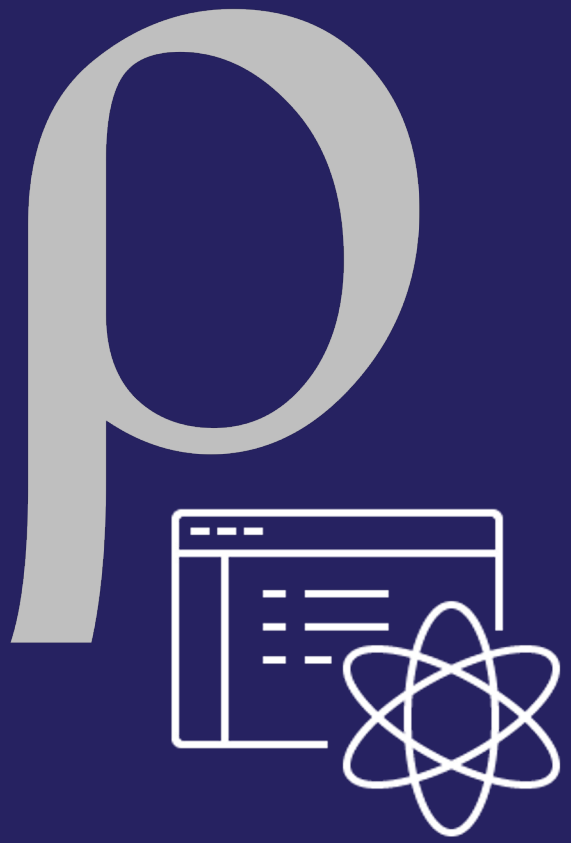


Suitable for fast prototyping, it can run for free in a local environment.

Noise can be applied to the entire circuit or to specific gates and qubits.

We instantiate it with ‘braket_dm’ parameter.

The Density Matrix Simulator (DM1)



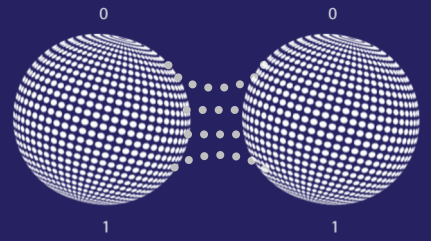
Stores the full density matrix of the system and sequentially applies gates and noise operations of the circuit.

Runtime scales linearly with the number of operations and exponentially with the number of qubits.

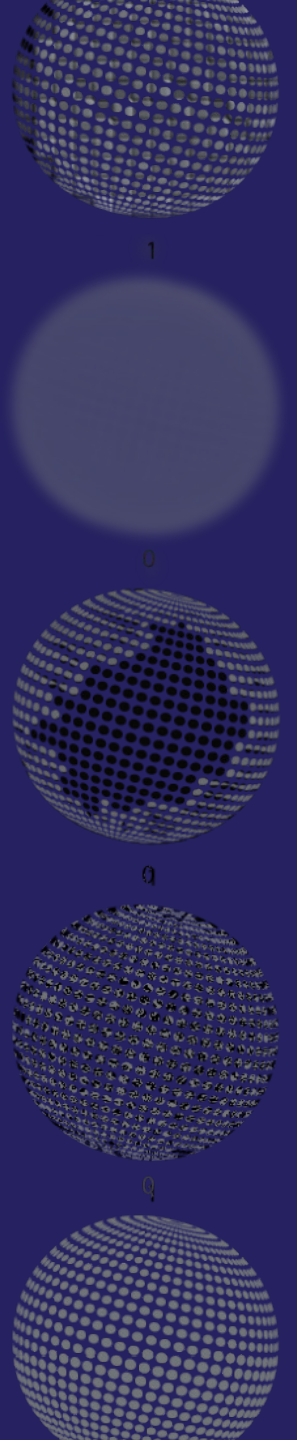
Results are represented as sample, expectation, variance and probability.

Predefined noise channels

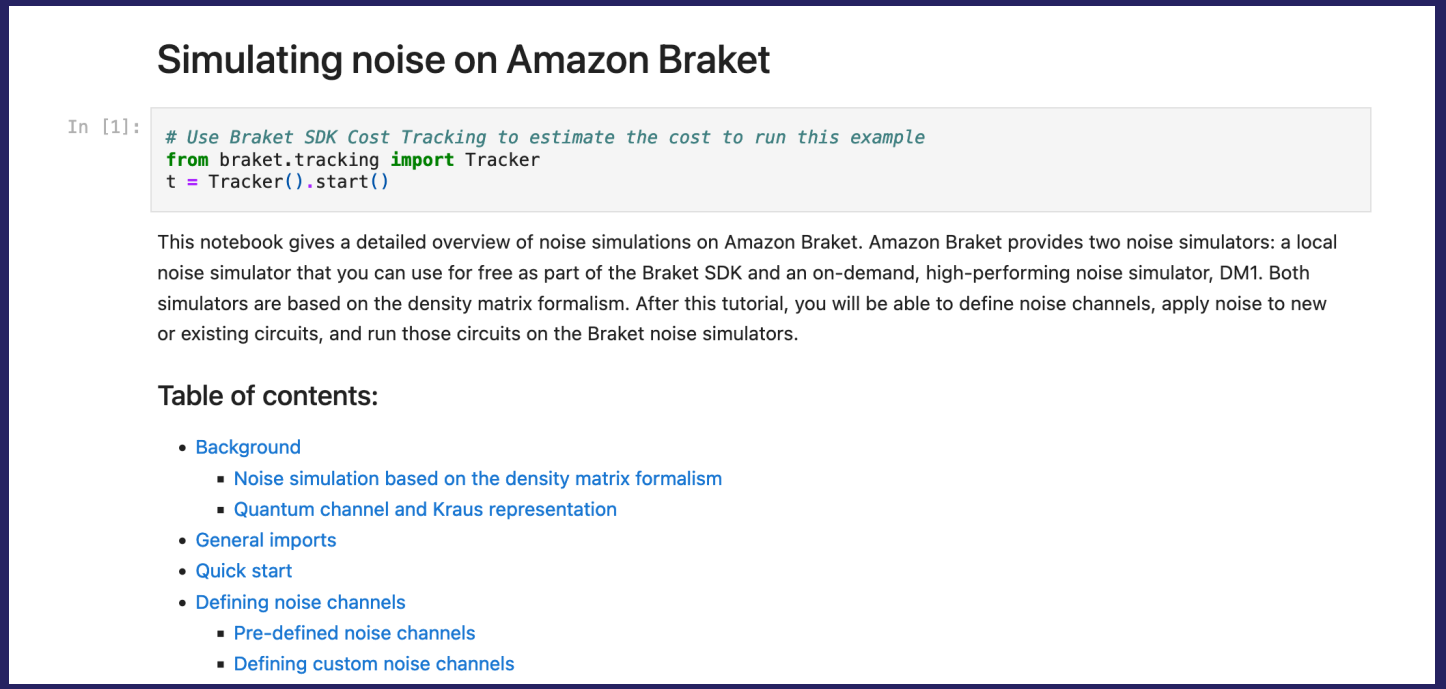
- Bit Flip
- Phase Flip
- Pauli Channel
- Phase Damping
- Amplitude Damping
- Depolarizing



- Two Qubit Depolarizing
- Two Qubit Dephasing



Build and bring your own noise



Simulating noise on Amazon Braket

In [1]:

```
# Use Braket SDK Cost Tracking to estimate the cost to run this example
from braket.tracking import Tracker
t = Tracker().start()
```

This notebook gives a detailed overview of noise simulations on Amazon Braket. Amazon Braket provides two noise simulators: a local noise simulator that you can use for free as part of the Braket SDK and an on-demand, high-performing noise simulator, DM1. Both simulators are based on the density matrix formalism. After this tutorial, you will be able to define noise channels, apply noise to new or existing circuits, and run those circuits on the Braket noise simulators.

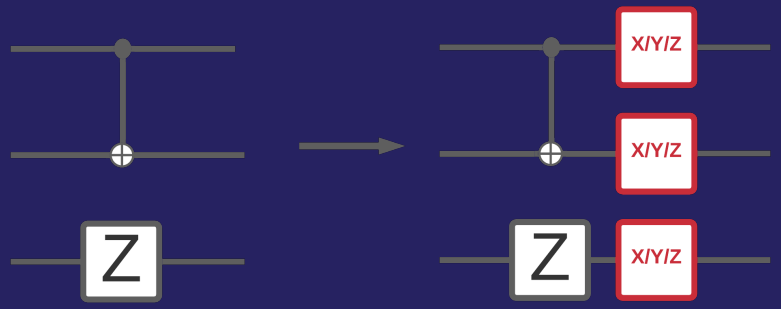
Table of contents:

You can define your own Kraus operators to represent various noise channels.

You can define a custom noise model to automatically apply certain noise channels to certain gates.

Custom Noise

1. Bottom up: Noise operations are applied as gates.



2. Global: Apply error to an entire existing circuit.

```
apply_gate_noise()

apply_initialization_noise()

apply_readout_noise()
```

Running variational algorithms with Amazon Braket Hybrid Jobs

Hybrid Algorithms with Amazon Braket

Amazon Braket SDK

Write your own hybrid algorithms

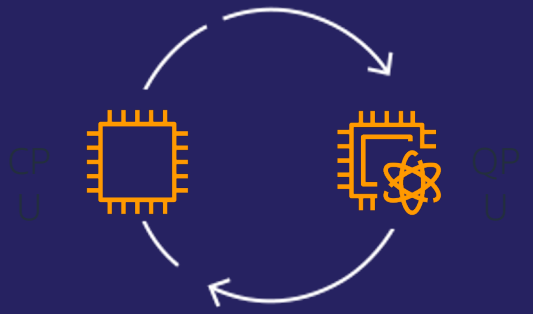
Growing list of examples

PENNY LANE

PennyLane

Python library for differentiable programming of quantum computers

Compatible with Amazon Braket Hybrid Jobs

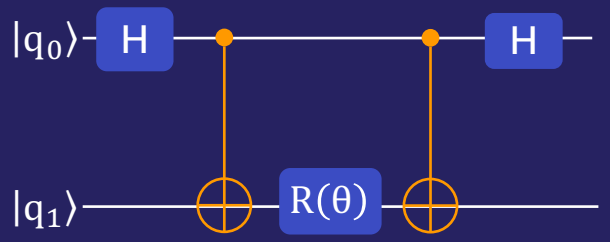


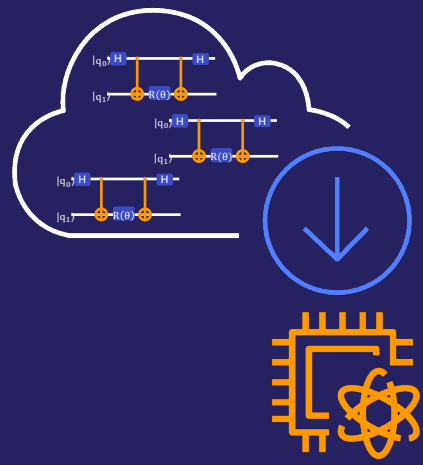
Braket Hybrid Jobs

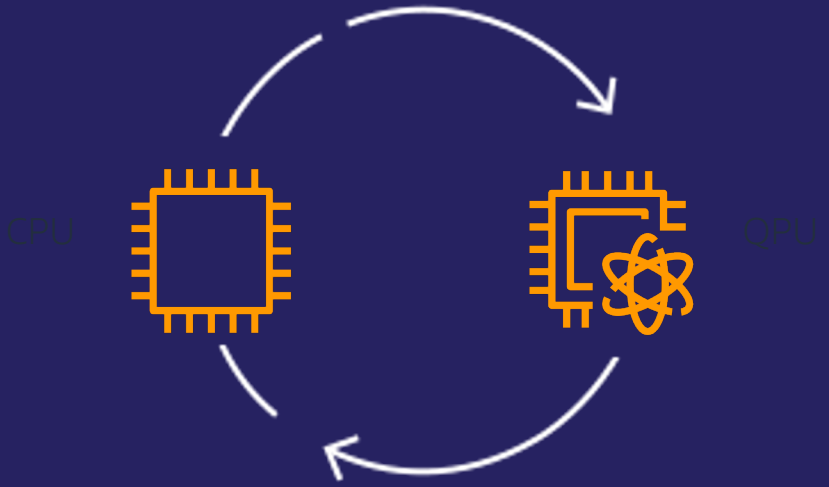
Priority access to QPUs

Job instance (EC2) that orchestrate quantum-classical interactions

Shots, tasks, and now Hybrid Jobs







Shot

Single execution of quantum operation on a device

Task

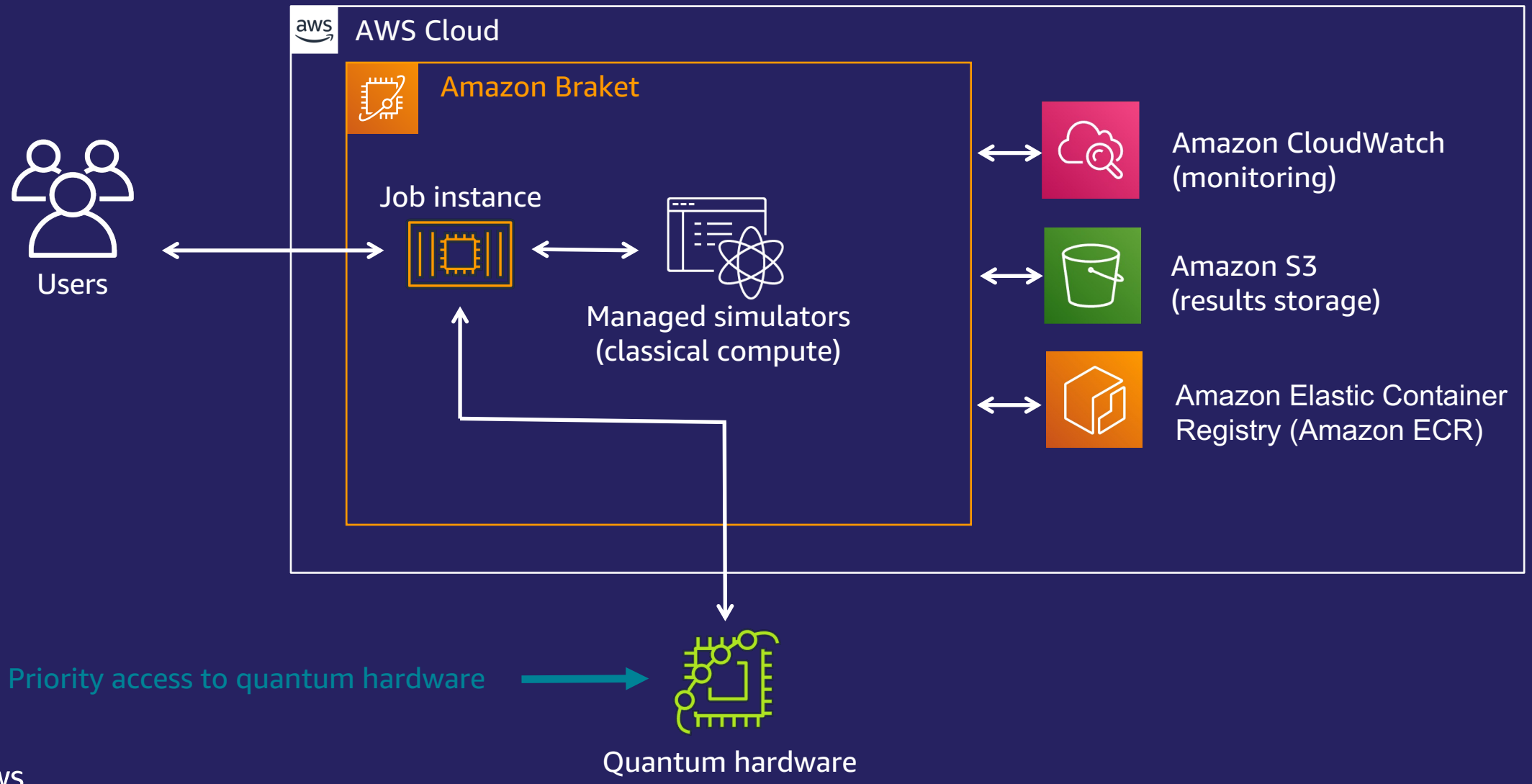
Series if repeated shots on a device (10s–10,000s shots per task)

Hybrid job

Sequence of classical and quantum compute cycles (10s to 1,000s of tasks per job)

docs.aws.amazon.com/braket/latest/developerguide/braket-jobs.html

Amazon Braket Hybrid Jobs

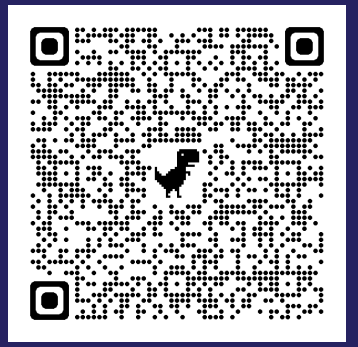


Useful tips for working with Amazon Braket

Amazon Braket Tips & Tricks

- Use the simulators to debug your algorithm and test it performs as expected.
- QPUs are scarce resources which may build up long queues – use the Amazon Braket queue visibility feature to understand when your task or Hybrid Job will run.
- If you plan to run a long series of tasks , use Hybrid Jobs to fire-and-forget your workload. Use the remote decorator to run your script in a Job with only a few code changes. Use high performance embedded simulators in Jobs.
- Use the Amazon Braket SDK's cost tracker to understand how many credits you've used.
- IonQ Harmony will process tasks albeit slowly.

Queue visibility blog post



Remote decorator blog post