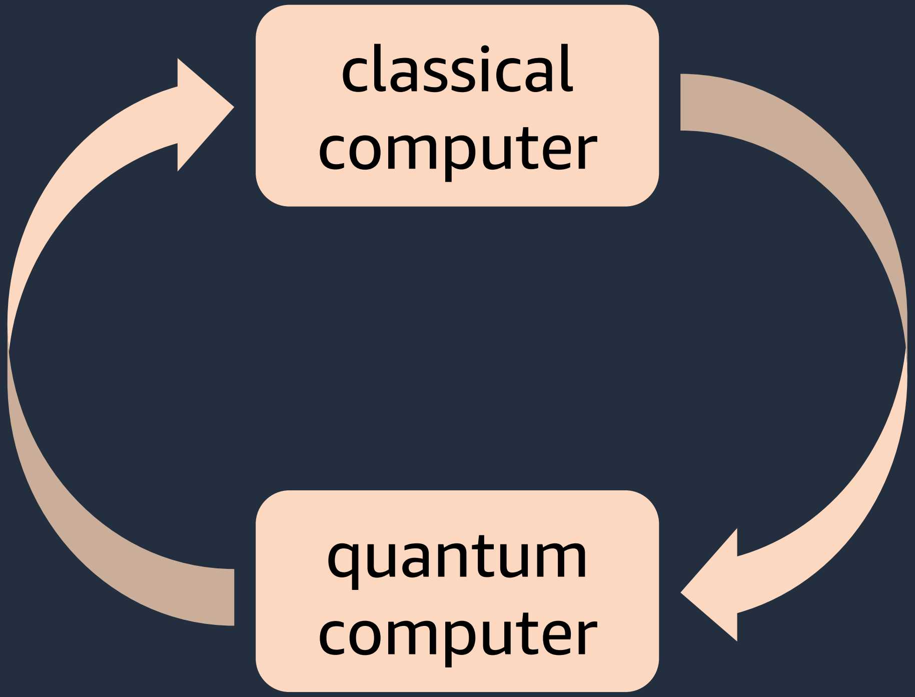
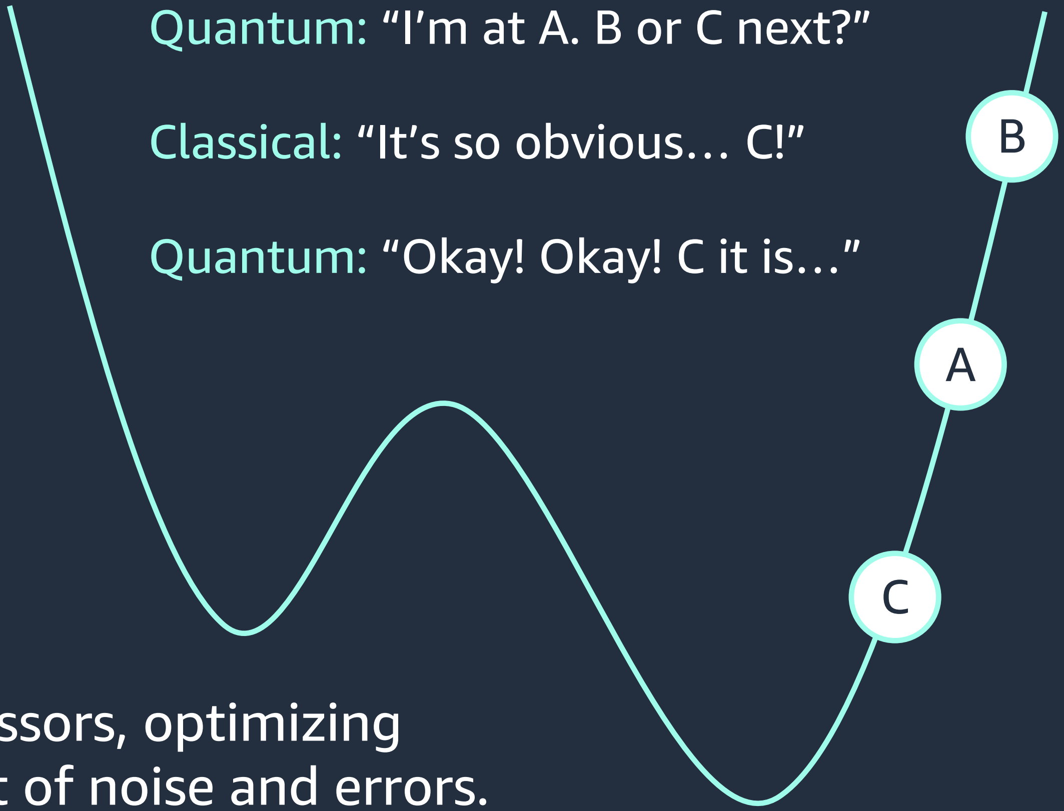
Amazon Braket Hybrid Jobs

Hybrid Classical-Quantum Algorithms





Quantum: “I’m at A. B or C next?”

Classical: “It’s so obvious… C!”

Quantum: “Okay! Okay! C it is…”

Classical and quantum parts act as co-processors, optimizing quantum algorithms to minimize the impact of noise and errors.

Hybrid Algorithms with Amazon Braket

Amazon Braket SDK

Write your own hybrid algorithms

Growing list of examples

PennyLane

Python library for differentiable programming of quantum computers

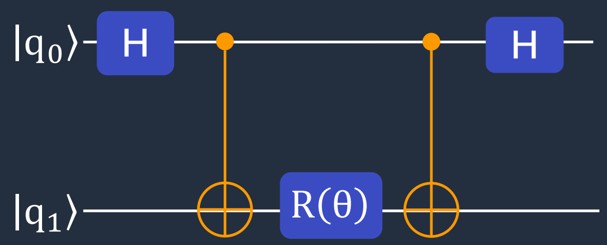
Compatible with Amazon Braket Hybrid Jobs

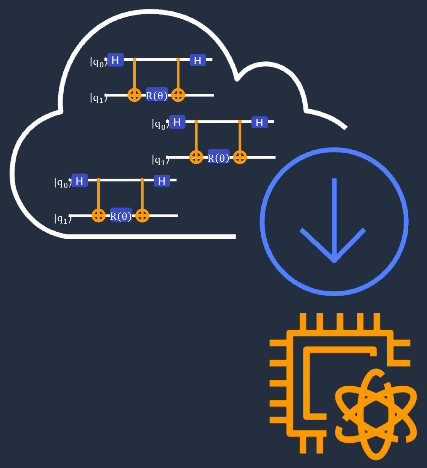
Braket Hybrid Jobs

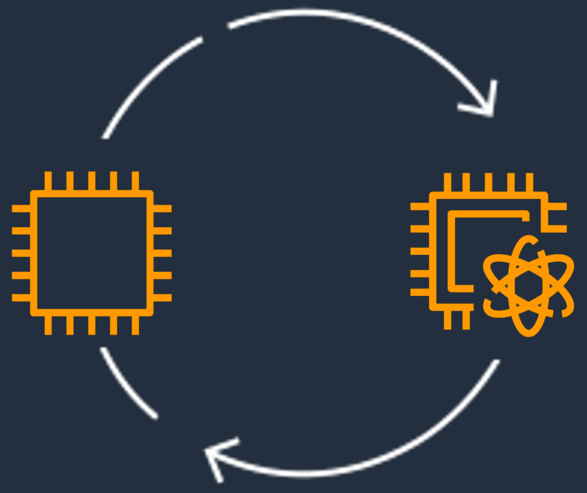
Priority access to QPUs

Job instance (EC2) that orchestrate quantum-classical interactions

Shots, Tasks, and Jobs







Shot

Single execution of quantum operation on a device

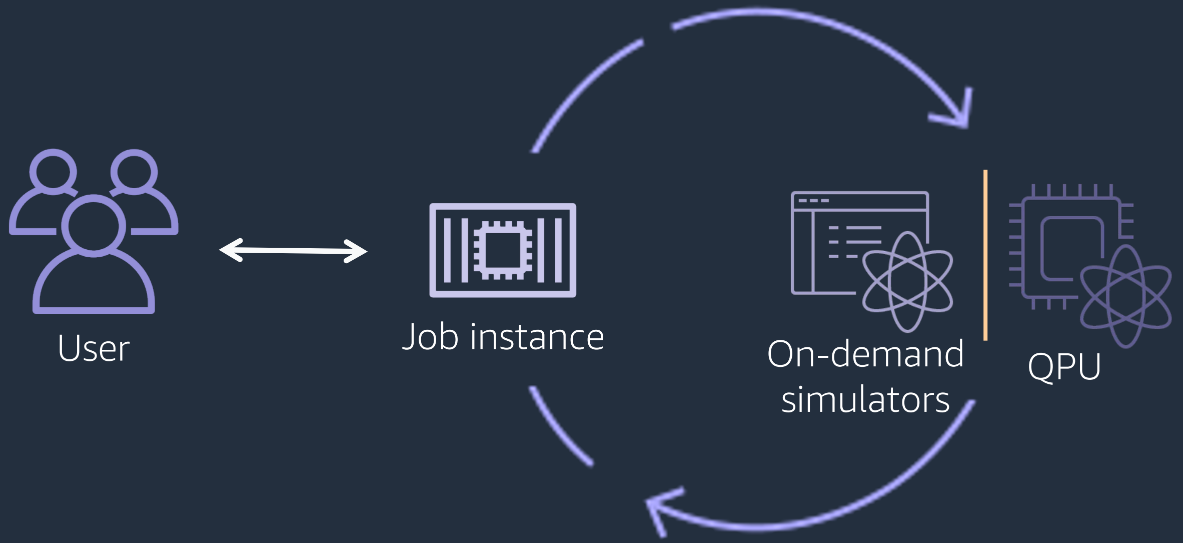
Quantum Task

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

Hybrid Job

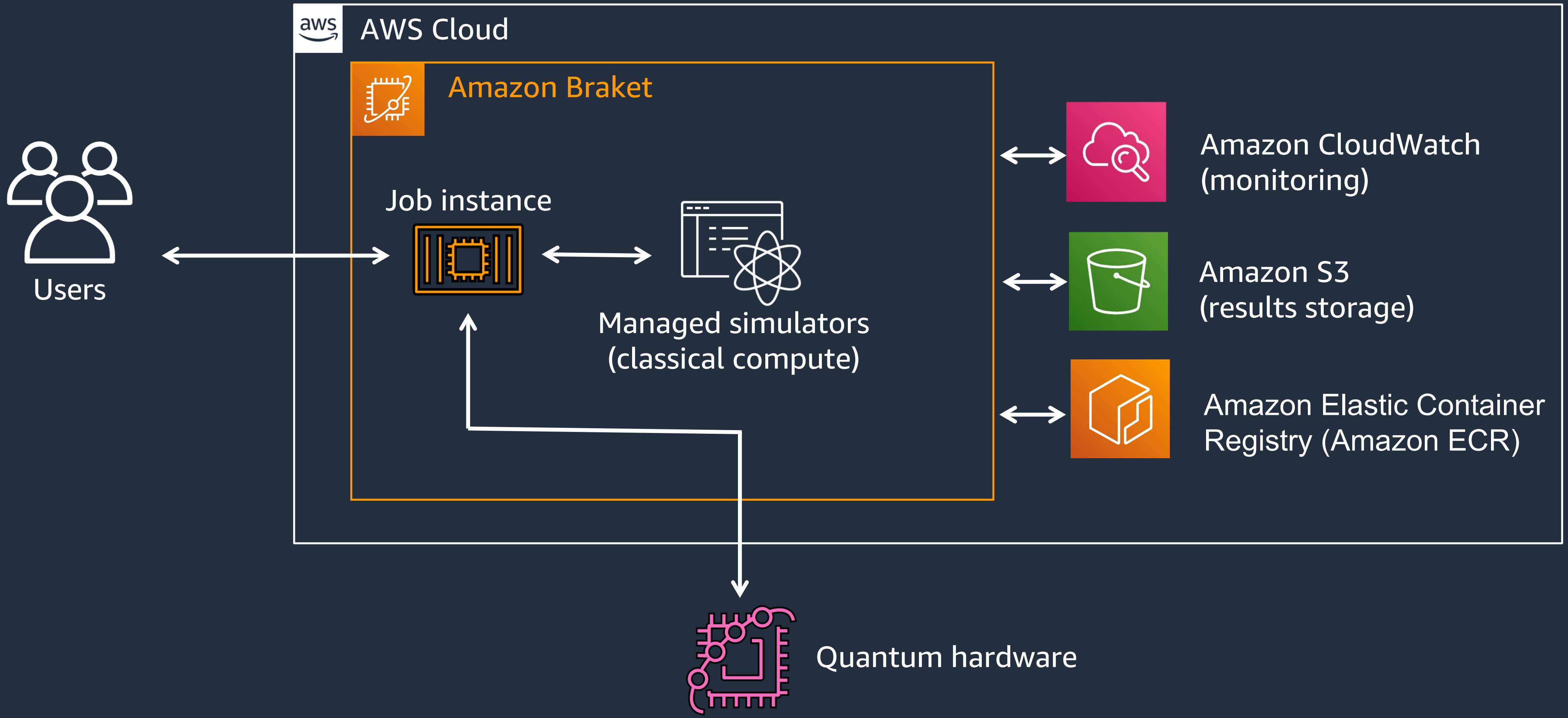
Sequence of classical and quantum compute cycles (10s to 1,000s of tasks per job)

Amazon Braket Hybrid Jobs



- Fully-managed support for quantum-classical workloads
- Spin, run, release, and pay only for what you use
- Live insights of algorithm metrics
- Priority access to the QPU

Amazon Braket Hybrid Jobs



AWS Cloud

Amazon Braket

Users

Job instance

Managed simulators
(classical compute)

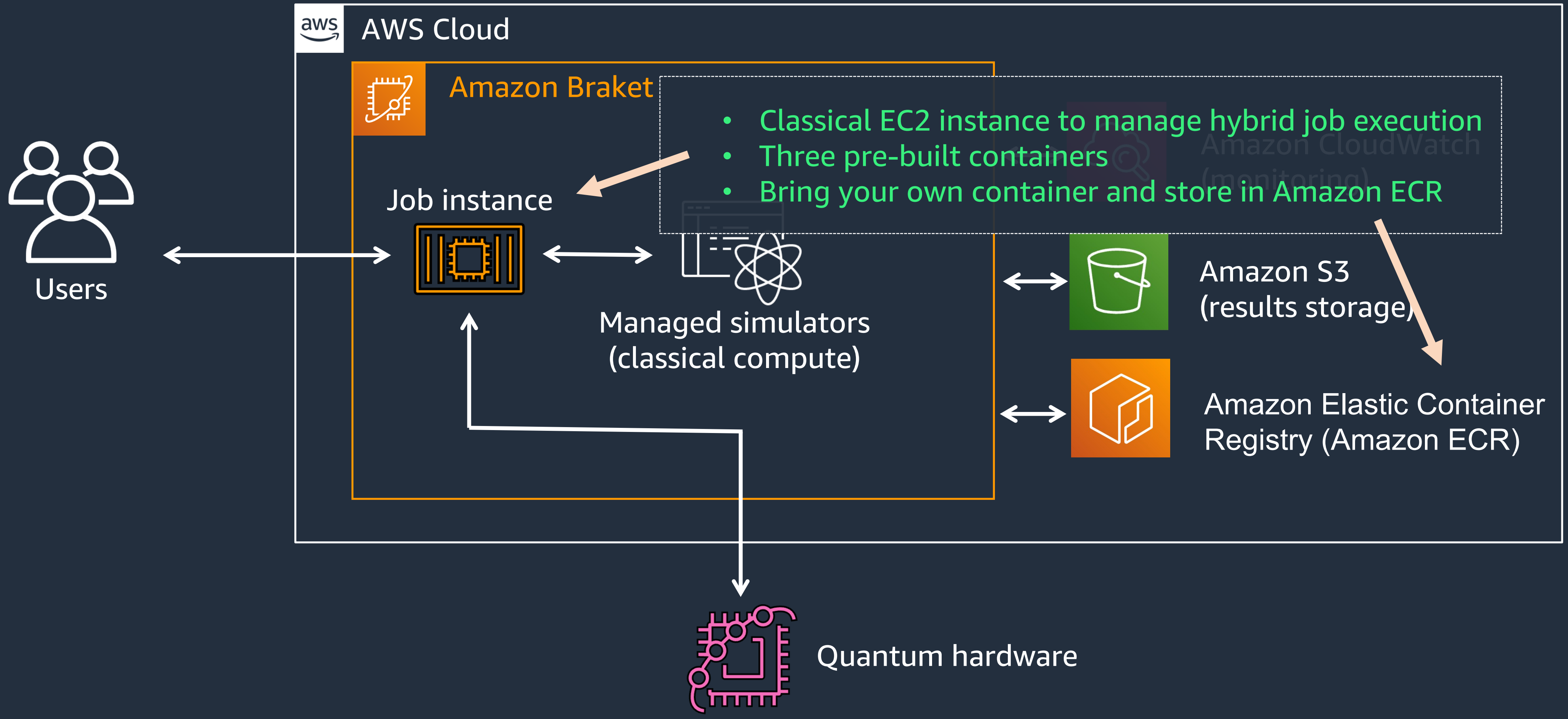
Amazon CloudWatch
(monitoring)

Amazon S3
(results storage)

Amazon Elastic Container
Registry (Amazon ECR)

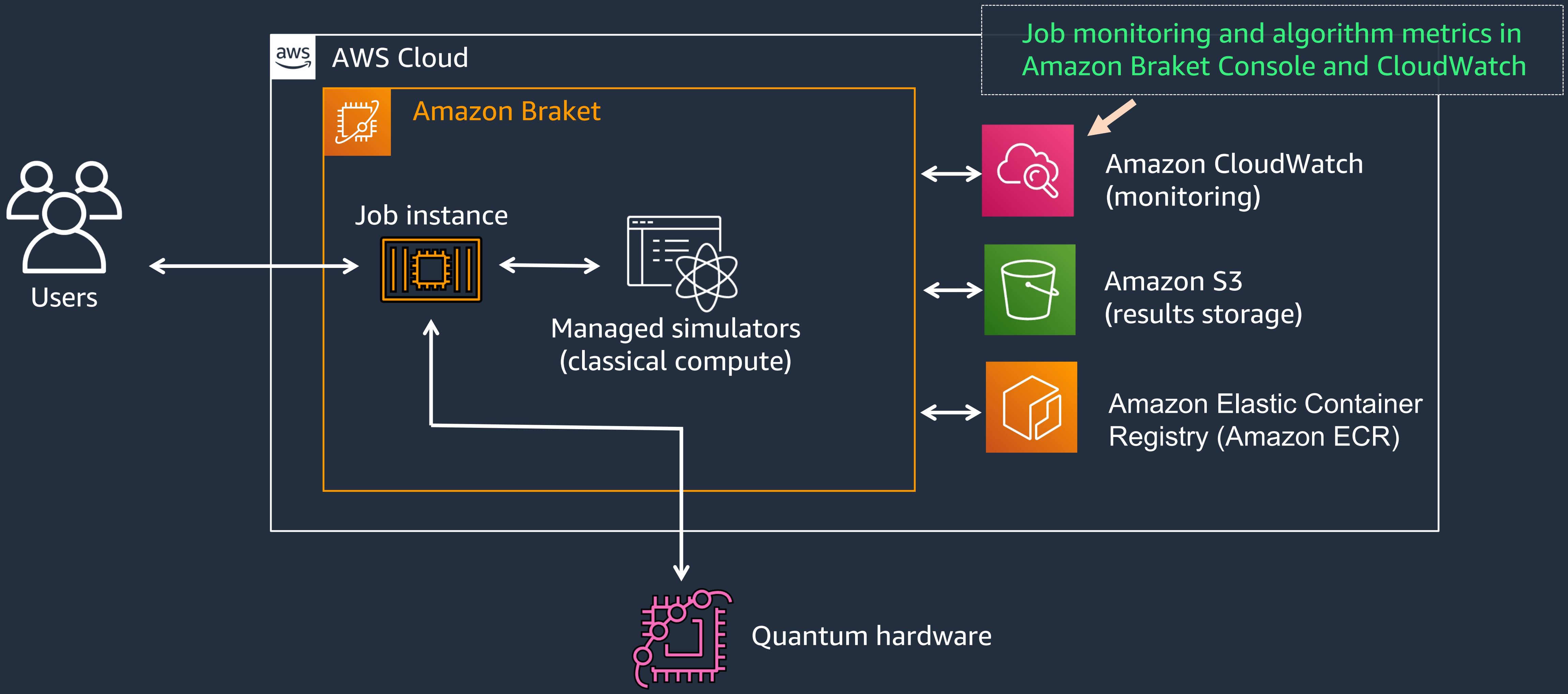
Quantum hardware

Amazon Braket Hybrid Jobs: customized containers



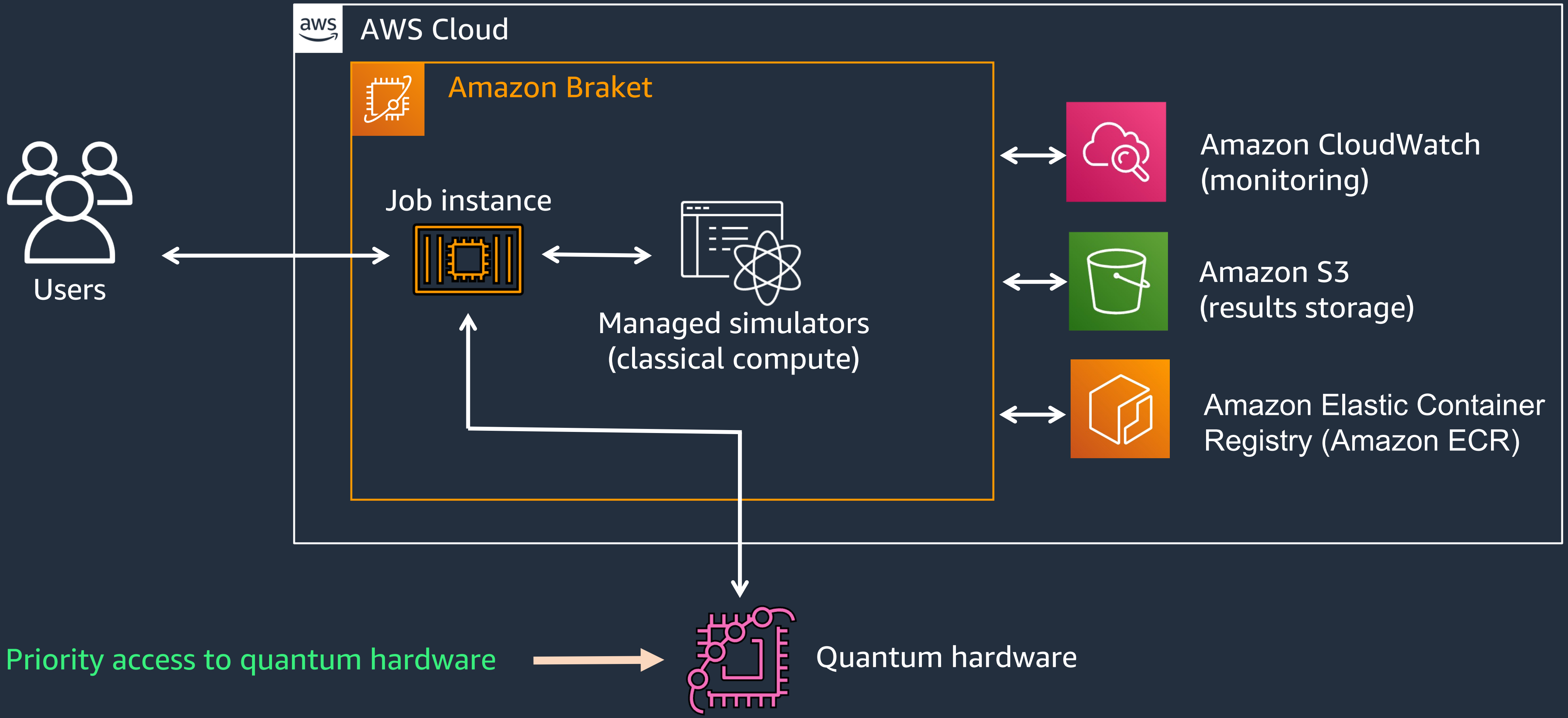
- Classical EC2 instance to manage hybrid job execution
- Three pre-built containers
- Bring your own container and store in Amazon ECR

Amazon Braket Hybrid Jobs: custom metrics



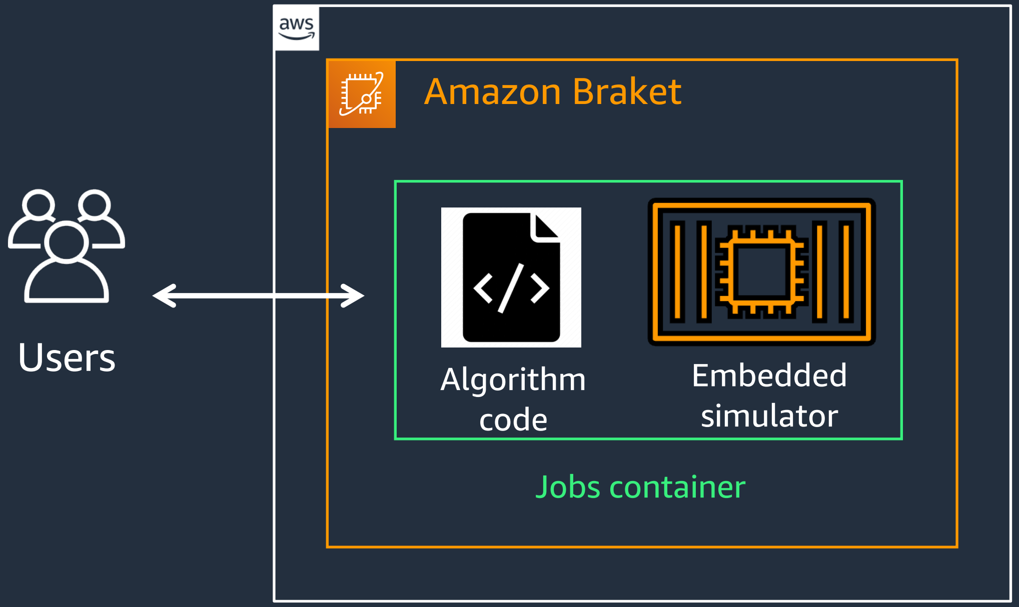
Job monitoring and algorithm metrics in Amazon Braket Console and CloudWatch

Amazon Braket Hybrid Jobs: priority access



Priority access to quantum hardware → Quantum hardware

Embedded Simulators for Hybrid Jobs



Speed up demanding algorithms

- Bring code and simulator together in a single container for faster communication and distributed scale-out.
- Reduce number of iterations and lower memory usage, decreasing running costs.
- lightning.qubit, lightning.gpu using the Nvidia cuQuantum library or BYOC

Hands-on Lab