```python
# Import necessary modules (adds IBM runtime for real hardware)
from qiskit import QuantumCircuit, transpile
from qiskit_aer import AerSimulator
from qiskit_ibm_runtime import QiskitRuntimeService, Sampler
from qiskit.visualization import plot_histogram
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")  # Suppress minor warnings for cleaner output

# Your API token (replace with your real one from quantum.ibm.com)
API_TOKEN = 'Key'  # Keep this secret!

# Create the entanglement circuit (same as before: 2 qubits)
qc = QuantumCircuit(2, 2)
qc.h(0)  # Superposition on Q0
qc.cx(0, 1)  # Entangle with CNOT
qc.measure(0, 0)
qc.measure(1, 1)

# Choose backend: Real hardware (change if needed, e.g., 'ibm_nairobi' for faster queue)
BACKEND_NAME = 'ibm_brisbane'  # 127-qubit machine; check status online

try:
    # Step 1: Authenticate with IBM (saves token for future runs)
    service = QiskitRuntimeService(channel="ibm_quantum", token=API_TOKEN)
    print(f"Authenticated with IBM Quantum! Available backends: {service.backends()[:3]}"
)  # Show first few

    # Step 2: Select the real backend
    backend = service.backend(BACKEND_NAME)
    print(f"Using real hardware: {backend.name} ({backend.status().pending_jobs} jobs in
queue)")

    # Step 3: Use Sampler primitive for efficient execution (modern way)
    sampler = Sampler(backend=backend, service=service)

    # Transpile for the specific hardware (maps gates to native ones)
    qc_compiled = transpile(qc, backend=backend, optimization_level=1)  # Level 1 for
balance of speed/accuracy

    # Run with 1024 shots (real hardware may limit to 8192 max)
    job = sampler.run(qc_compiled, shots=1024)
    print(f"Job submitted! ID: {job.job_id()}. Monitor at: https://quantum.ibm.com/jobs/{
job.job_id()}")

    # Wait for results (this may take 5-30+ minutes due to queue)
    print("Waiting for job to complete... (Check the link above for real-time status)")
    result = job.result()
    counts = result.quasi_dists[0].binary_probabilities()  # Get counts from Sampler

except Exception as e:
    print(f"Real hardware issue: {e}")
    print("Falling back to simulator...")
    # Fallback to Aer simulator (your working setup)
    simulator = AerSimulator()
    qc_compiled = transpile(qc, simulator)
    from qiskit_aer import AerSimulator  # Ensure import if needed
    job = simulator.run(qc_compiled, shots=1024)
    result = job.result()
    counts = result.get_counts()

# Plot and print results (same for sim or real)
plot_histogram(counts)
plt.title("Entanglement Outcomes (Real Quantum Hardware or Simulator)")
plt.show()
print("Measurement outcomes:", counts)
if '01' in counts or '10' in counts:
```

```
    print("Note: Small '01'/'10' counts indicate real hardware noise—perfectly normal!")
else:
    print("Perfect correlation: Only '00' and '11'—quantum entanglement in action!")

```