In [1]:
```
from qiskit import IBMQ
API_TOKEN ='bf24308d4b9628ea5d8f4a213416453f23eb280986828dbe5d3b691a9a9cd0b40b0b40a2e7741cc22abc447b4feb2ba88a4d51f4d2512c7bca3aa964ef6b1a2c'
IBMQ.save_account(API_TOKEN)

Public_Provider = IBMQ.load_account()
Public_Provider.backends()

backend_publicism = Public_Provider.get_backend('ibmq_qasm_simulator')
backend_publiclon = Public_Provider.get_backend('ibmq_ourense')
print(backend_publicism)
print(backend_publiclon)

from qiskit import QuantumRegister, QuantumCircuit, ClassicalRegister
%matplotlib inline

from qiskit import BasicAer, execute
from qiskit.tools.monitor import job_monitor
from qiskit.tools.visualization import plot_histogram, plot_bloch_multivector
import numpy as np
from qiskit.extensions import Initialize
```

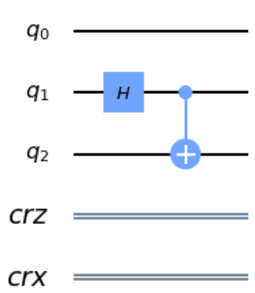
```
configrc.store_credentials:WARNING:2020-09-12 21:45:38,942: Credentials already present. Set overwrite=True to overwrite.
```

```
ibmq_qasm_simulator
ibmq_ourense
```

In [2]:
```
qr = QuantumRegister(3, name="q")
crz = ClassicalRegister(1, name="crz")
crx = ClassicalRegister(1, name="crx")
teleportation_circuit = QuantumCircuit(qr, crz, crx)
```

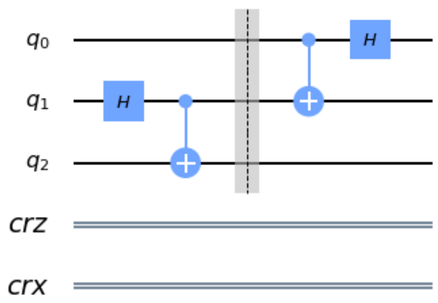
In [3]:
```
teleportation_circuit.h(qr[1])
teleportation_circuit.cx(qr[1], qr[2])
teleportation_circuit.draw(output = 'mpl')
```

Out[3]:



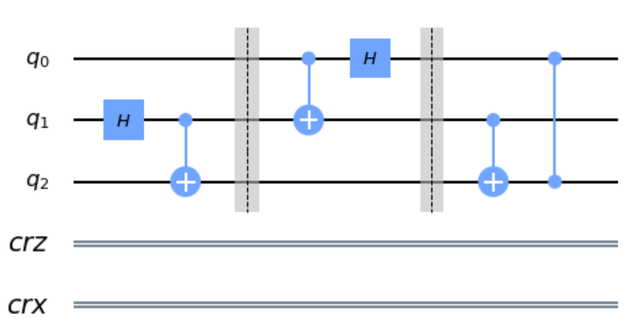
In [4]:
```
teleportation_circuit.barrier() # Use barrier to separate steps
teleportation_circuit.cx(qr[0],qr[1])
teleportation_circuit.h(qr[0])
teleportation_circuit.draw(output = 'mpl')
```

Out[4]:



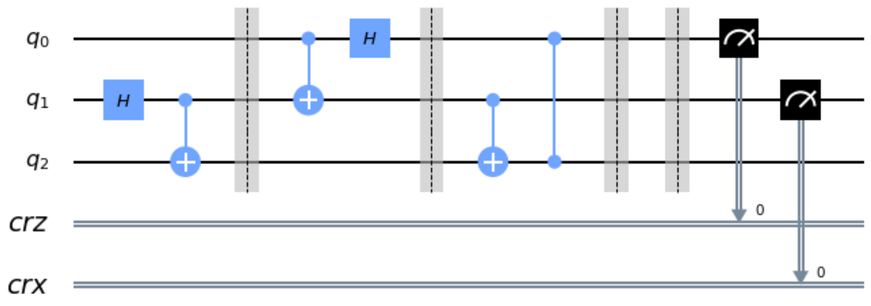
In [5]:
```
teleportation_circuit.barrier() # Use barrier to separate steps
teleportation_circuit.cx(qr[1],qr[2])
teleportation_circuit.cz(qr[0],qr[2])
teleportation_circuit.draw(output = 'mpl')
```

Out[5]:



In [7]:
```
teleportation_circuit.barrier() # Use barrier to separate steps
teleportation_circuit.measure(qr[0],crz)
teleportation_circuit.measure(qr[1],crx)
teleportation_circuit.draw(output = 'mpl')
```

Out[7]:



In [8]:
```
job = execute(teleportation_circuit,backend_publiclon)
job_monitor(job)
```

```
Job Status: job has successfully run
```

In [9]:
```
result = job.result()
```

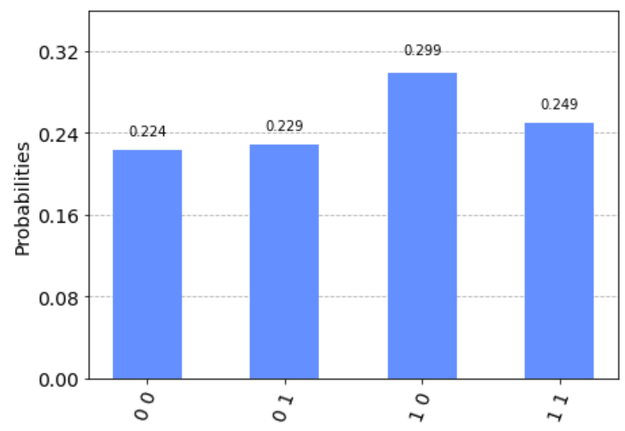
In [10]:
```
count = result.get_counts()
```

In [11]:
```
print(count)
```

```
{'0 0': 229, '0 1': 234, '1 1': 255, '1 0': 306}
```

In [12]:
```
plot_histogram(count)
```

Out[12]:



In [ ]: