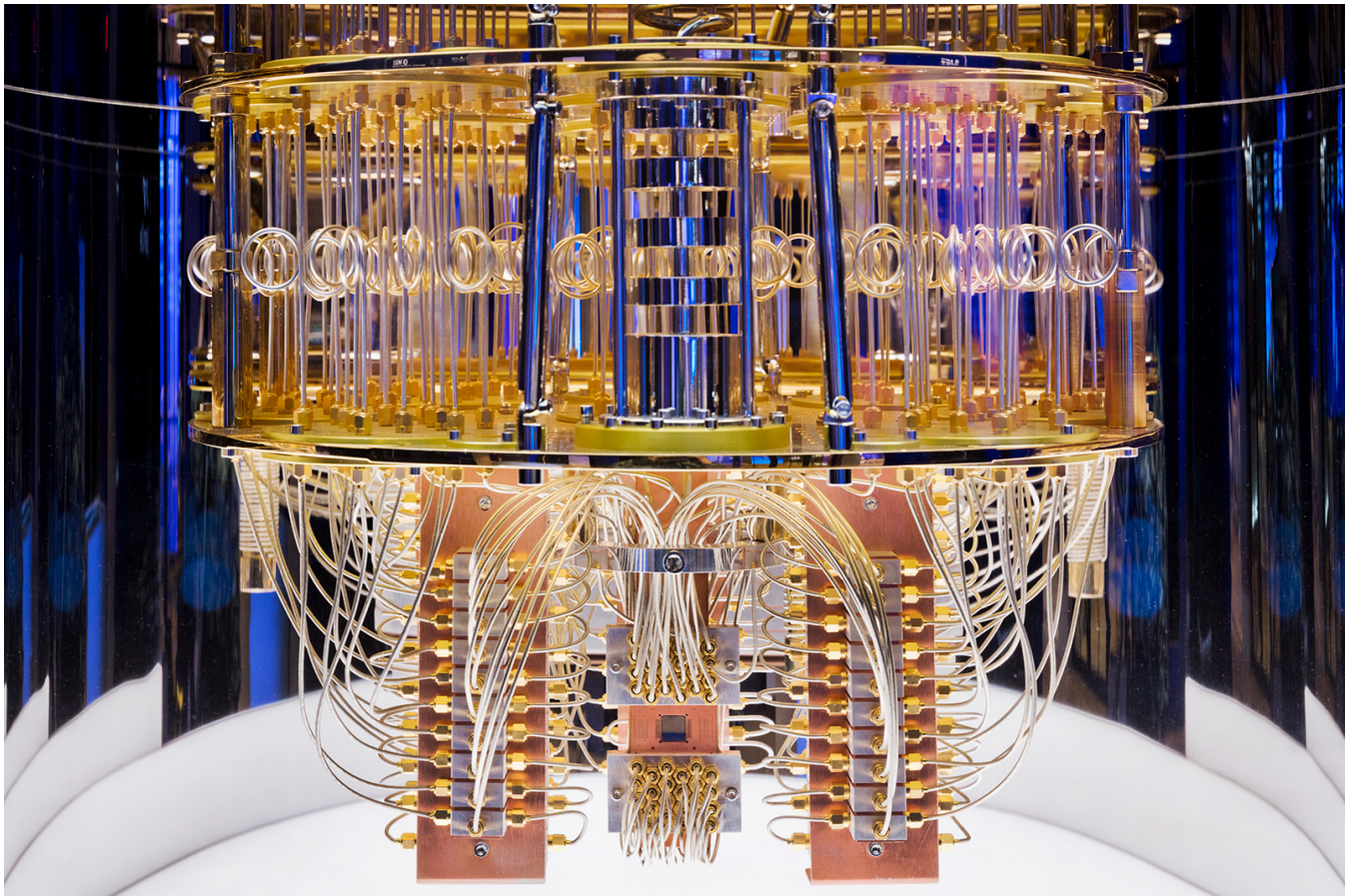


# Unlocking Quantum Computing: A Step-by-Step Guide to Executing Code on Real Quantum Computers

## How To Run Code On A Real Quantum Computer..?

Quantum computers are now a reality, and you can access them for free through platforms like IBM's cloud-based quantum computing services. To get started, you'll need to register for IBM's quantum services by creating an IBMid account and verifying your email address.

Unlock the future of computing with quantum technology. Quantum computers harness the power of the smallest particles to solve complex problems faster and more efficiently than ever before. Step into the realm of limitless possibilities and accelerate innovation with quantum computing.




Source: IBM Quantum Computing, (Source: [IBM Quantum Platform \(https://quantum-computing.ibm.com/\)](https://quantum-computing.ibm.com/)).

## Create an IBM Quantum Platform Account

To begin utilizing IBM's quantum computing services, the initial step involves registering for an IBMid account. Simply select the [Create an IBMid account \(https://quantum-computing.ibm.com/\)](https://quantum-computing.ibm.com/) option and furnish the required information. Once you have successfully confirmed your email address, you will be

IBM Quantum Platform Link: <https://quantum-computing.ibm.com/>



Welcome to IBM

Create an account to access trials, demos, and services.

### Create an IBMid

Already have an IBM account? [Log in](#)

Account information

E-mail

Your email address will become your IBMid, which you'll use to log into IBM.com.

First name

Last name

Password

Country or region of residence

Pakistan

State or province

Select State

Next

Create an Account, (Source: [IBM Quantum Platform \(https://quantum-computing.ibm.com/\)](https://quantum-computing.ibm.com/)).

## Login into IBM Quantum Platform Account

If you have created an account you can directly login into it by using credential

← → ↻ quantum-computing.ibm.com

IBM Quantum Platform | Dashboard | Compute Resources | Jobs

# IBM Quantum

Use our suite of applications to support your quantum research and development needs.

Platform

Copy your API token, track jobs, and view quantum compute resources.

Recent jobs

Job ID	Status	Created	Run	Compute resource
ibmq_16mq_v1	Completed	1 day ago	1 day ago	ibmq_16mq_v1
ibmq_16mq_v1	Completed	1 day ago	1 day ago	ibmq_16mq_v1

Your systems

42

IBM Quantum Platform

Recent jobs

What's new

Sign in to IBM Quantum

Continue with IBMid

G

New to IBM Quantum?

Create an IBMid

Having trouble signing in?

Try signing in with an IBMid. If you are still having issues, contact the [IBMid help desk](#).

Login into the IBM Quantum Platform Account, (Source: [IBM Quantum Platform \(https://quantum-computing.ibm.com/\)](https://quantum-computing.ibm.com/)).

# Dashboard of IBM Quantum Platform

The Dashboard offers a comprehensive summary of your reserved IBM Quantum Services. Of utmost significance, it displays a section labeled "Your Systems." To access a complete list, kindly click on "View All" and subsequently opt for "Your Systems" within the filtering dropdown menu located immediately adjacent to the search bar.

The screenshot displays the IBM Quantum Platform dashboard for user Khuram Shahzad. The dashboard is divided into several sections:

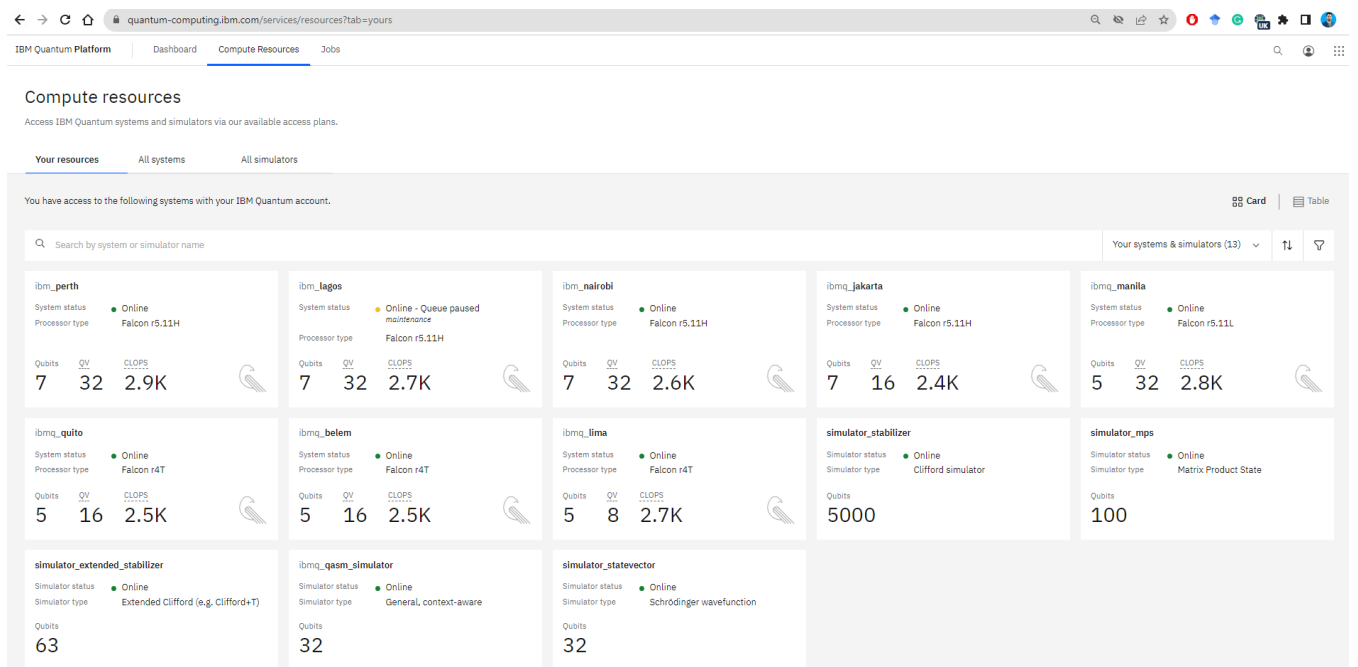
- Recent Jobs:** A table showing job status and completion details.
- Your systems:** A section showing 8 systems and 5 simulators.
- Documentation:** A section with a search bar and links to Qiskit Runtime, Introduction to primitives, and Dynamic Circuits.
- Learning:** A section with links to IBM Quantum Composer and IBM Quantum Lab.
- What's new:** A section with updates on product updates, service alerts, and quantum news.

Job ID	Status	Created	Completed	Compute resource
ck2le41r6vdfs1djqv0	Queued Est. wait: 4 minutes	17 minutes ago		ibmq_perth Queue position: 1
ck2l3ht1r6vdfs1cdaug	Failed - Instruction bfunc is not supported	40 minutes ago	21 minutes ago	ibmq_perth
ck2jst1r6vdfs1819tg	Queued Est. wait: 1 month	About 2 hours ago		ibmq_quito Queue position: 238
ck2jrc14fd8h6bp9l3ag	Queued Est. wait: 1 month	About 2 hours ago		ibmq_quito Queue position: 237
ck2jko51r6vdfs175gl0	Cancelled	About 2 hours ago	About 2 hours ago	ibmq_quito

IBM Quantum Dashboard, (Source: [IBM Quantum Platform \(https://quantum-computing.ibm.com/\)](https://quantum-computing.ibm.com/)).

## Availability of Quantum Computer and Simulator

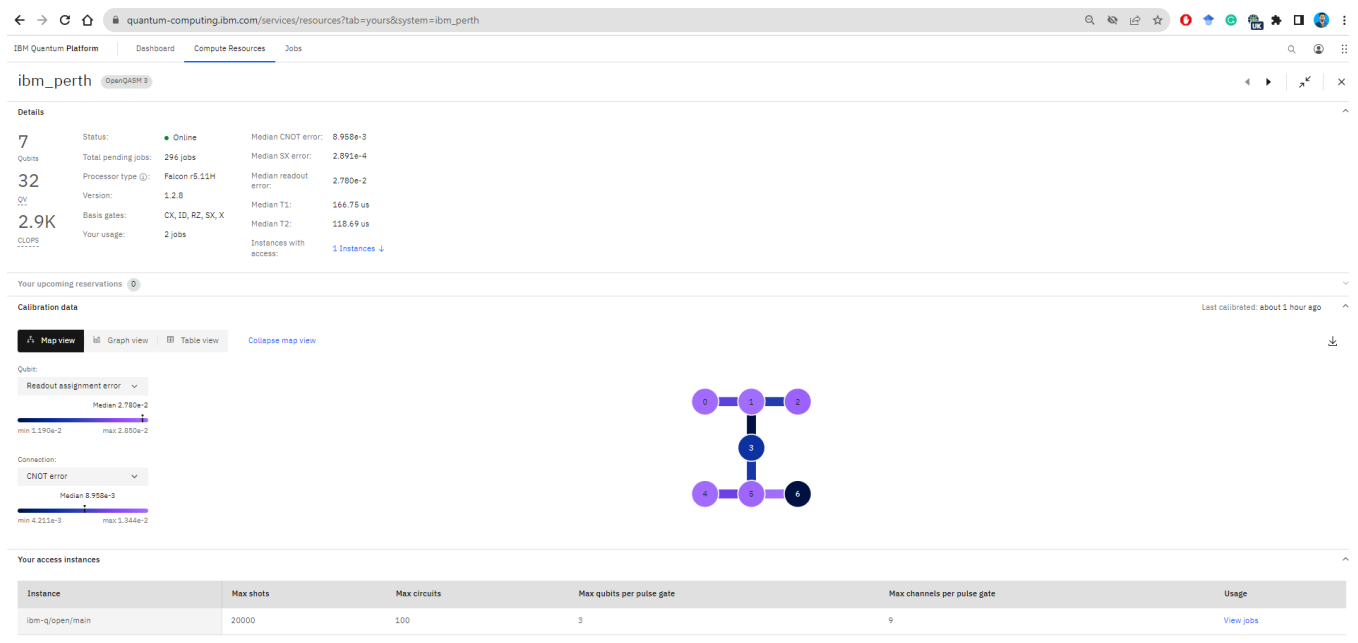
This action [Resources \(https://quantum-computing.ibm.com/services/resources?tab=yours\)](https://quantum-computing.ibm.com/services/resources?tab=yours) will provide you with an overview of the available quantum computing systems at your disposal. While these systems may possess only a limited number of qubits, they are entirely adequate for initiating your quantum computing endeavors.



IBM Quantum Platform Resources, (Source: [IBM Quantum Platform \(https://quantum-computing.ibm.com/\)](https://quantum-computing.ibm.com/)).

## Configuration of Quantum Device

Upon selecting a specific system, you will gain access to a comprehensive overview of its architectural and configurational details. Notably, the diagram positioned in the lower-right corner elucidates the interconnections among the qubits within the system. An inherent constraint to be mindful of when engaging with an authentic quantum computer is our ability to exclusively entangle qubits that share a direct connection with one another.



Quantum Device configuration, (Source: [IBM Quantum Platform \(https://quantum-computing.ibm.com/\)](https://quantum-computing.ibm.com/)).

# Let's Start Coding in Jupyter Notebook or in IBM Quantum Lab.

We start with a few imports and then, we define the circuit.

Note: IBM has stopped its quantum hardware services in Pakistan, you can try it using a proxy

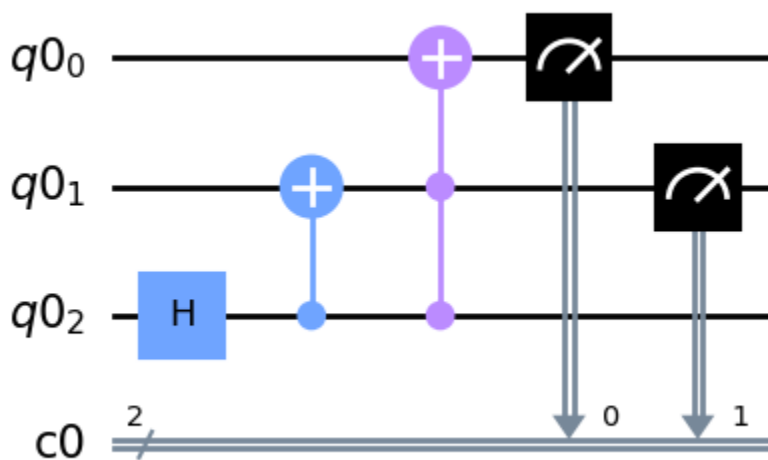
## A code Sample: Implementation of Left Shift

```
In [1]: from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister, IBMQ, execute, transpile
from qiskit.tools.monitor import job_monitor
from qiskit.tools.visualization import plot_histogram

qr = QuantumRegister(3)
cr = ClassicalRegister(2)

circuit = QuantumCircuit(qr, cr)
circuit.h(qr[2])
circuit.cx(qr[2],qr[1])
circuit.ccx(qr[2],qr[1],qr[0])
circuit.measure(qr[0], cr[0])
circuit.measure(qr[1], cr[1])
circuit.draw('mpl')
```

Out[1]:



## Load your IBM Quantum Account using API Token

Prior to executing the code, it is imperative that we establish a connection with the **ibm\_perth** device. To achieve this, we must first load our account by utilizing your API token. Your API token can be acquired through the IBM Quantum dashboard and is characterized by its considerable length. It is of utmost importance to exercise vigilance in safeguarding this token.

```
In [3]: # replace TOKEN with your API token string (https://quantum-computing.ibm.com/)
# IBMQ.save_account("TOKEN", overwrite=True)
# provider = IBMQ.load_account()
# Suppress warnings
IBMQ.save_account('TOKEN', overwrite=True)
IBMQ.load_account() # Load account from disk
IBMQ.providers()    # List all available providers
```

```
Out[3]: [<AccountProvider for IBMQ(hub='ibm-q', group='open', project='main')>]
```

## List of available Quantum Devices

```
In [4]: provider = IBMQ.get_provider(hub='ibm-q')
provider.backends()
```

```
Out[4]: [<IBMQSimulator('ibmq_qasm_simulator') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQBackend('ibmq_lima') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQBackend('ibmq_belem') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQBackend('ibmq_quito') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQSimulator('simulator_statevector') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQSimulator('simulator_mps') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQSimulator('simulator_extended_stabilizer') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQSimulator('simulator_stabilizer') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQBackend('ibmq_jakarta') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQBackend('ibmq_manila') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQBackend('ibmq_lagos') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQBackend('ibmq_nairobi') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQBackend('ibmq_perth') from IBMQ(hub='ibm-q', group='open', project='main')>]
```

## Make a connection with IBM Device

Next, we connect to the provider and get the Preth backend. If you want to connect to other backends, you may need to select another hub, group, and project. But IBM will let you know when they grant you access to any other provider.

Then, we need to transpile the circuit. In this step, Qiskit rewrites your code to match the device's requirements. For instance, if we entangled two qubits that are not connected physically, Qiskit would rewrite the circuit to adhere to the hardware requirements.

Finally, we assemble the circuit and send it to the backend that returns a job object.



```
In [5]: # Get backend for experiment
provider = IBMQ.get_provider(hub='ibm-q', group='open', project='main')
backend = provider.get_backend('ibm_perth')

# prepare the circuit for the backend
mapped_circuit = transpile(circuit, backend=backend)
qobj = assemble(mapped_circuit, backend=backend, shots=1024)

# execute the circuit
job = backend.run(qobj)
```

## Live Job Status and Its ID

The job object lets you manage your request. For instance, you can obtain its status

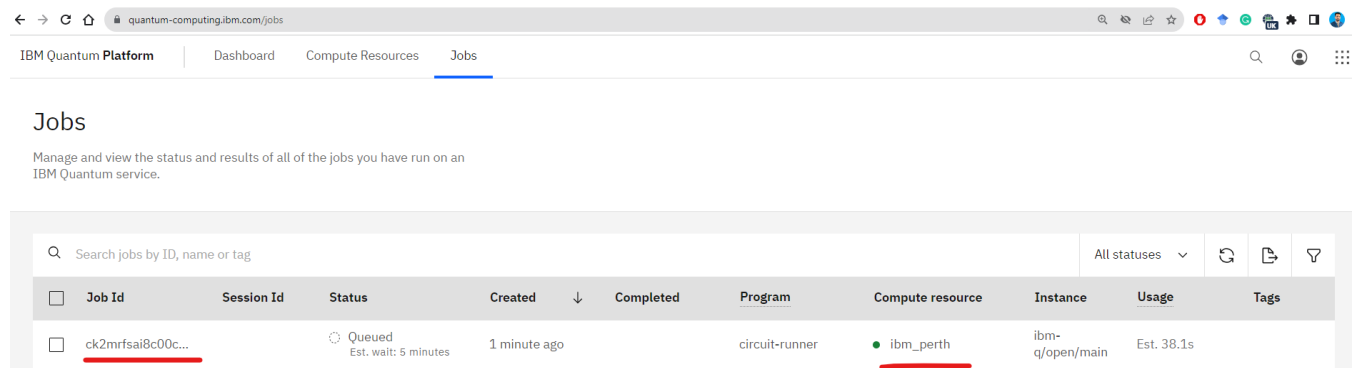
```
In [6]: job.status()
```

```
Out[6]: <JobStatus.QUEUED: 'job is queued'>
```

```
In [7]: job.job_id()
```

```
Out[7]: 'ck2mrfsai8c00cggfaj0'
```

Let's have another look at the IBM Quantum dashboard. You should now have an entry in the **Jobs**-list. When you click on it, you get the details. There, you also see when your circuit is scheduled to run. In my case, it takes about half an hour.



Job Id	Session Id	Status	Created	Completed	Program	Compute resource	Instance	Usage	Tags
ck2mrfsai8c00c...		Queued Est. wait: 5 minutes	1 minute ago		circuit-runner	ibm_perth	ibm-q/open/main	Est. 38.1s	

(Source: IBM Quantum Platform, (Source: [IBM Quantum Platform \(https://quantum-computing.ibm.com/\)](https://quantum-computing.ibm.com/))).

## Get the result from Quantum Device

Of course, we're interested in the results of the job once completed. But maybe you don't want to wait that long. So, you may want to retrieve the job later.

```
In [8]: job = provider.get_backend('ibm_perth').retrieve_job('ck2mrfsai8c00cggfaj0')
```

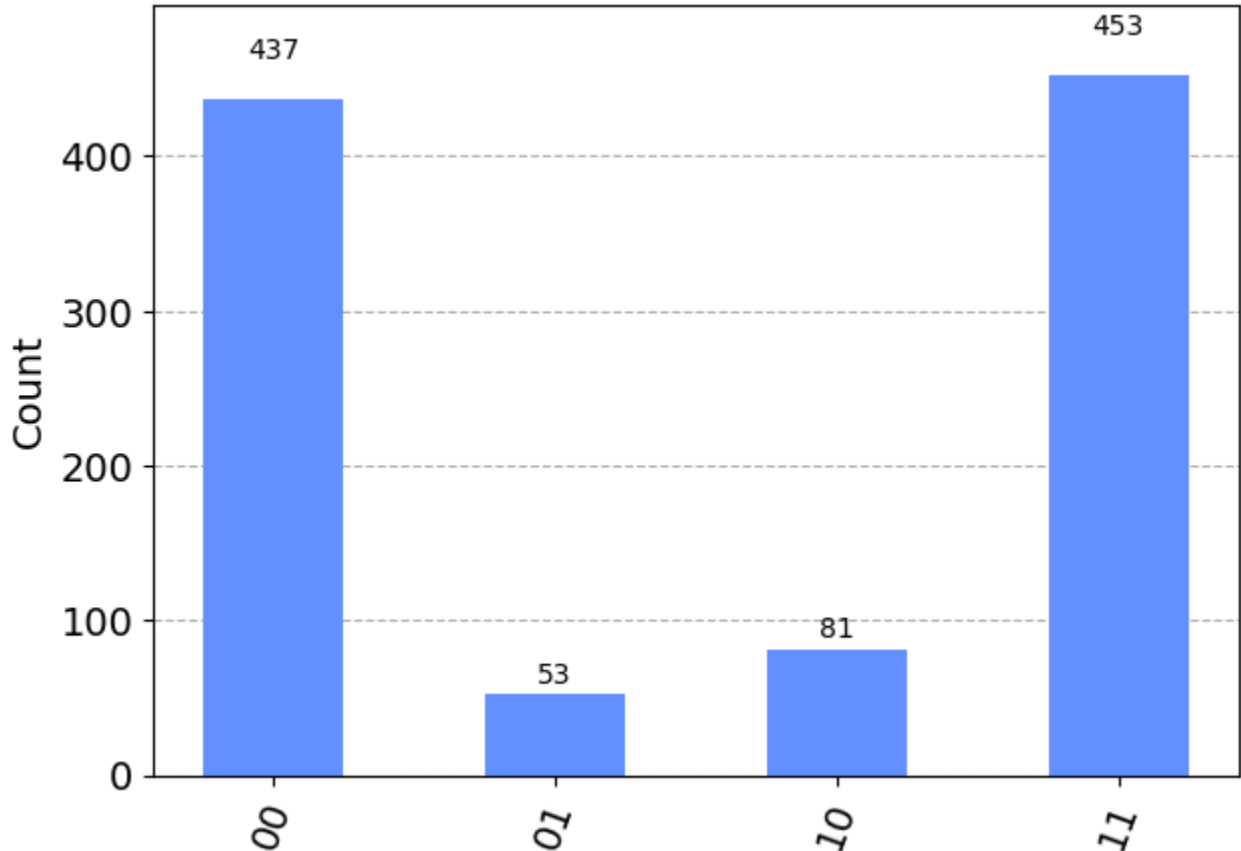
```
In [10]: result = job.result()
counts = result.get_counts()
print(counts)
```

```
{'00': 437, '01': 53, '10': 81, '11': 453}
```

The bitstrings indicate the measured values of the qubits (read from right to left). We can see that the upper qubits at positions four and five are not always 0. This may seem strange because we didn't do anything with them. So, they should remain at the default state of 0. Unfortunately, qubits are very sensitive to noise.

```
In [13]: plot_histogram(counts)
```

Out[13]:



The result show that state  $|00\rangle$  and  $|11\rangle$  have high probability than state  $|01\rangle$  and  $|10\rangle$

## Summary

Quantum computing is now accessible through platforms like IBM's cloud-based services. To begin, register for an IBMid account and verify your email.

Once registered, use the Dashboard to explore available IBM Quantum Services, focusing on "Your Systems." To see all systems, click "View All" and select "Your Systems" from the filter. These systems have limited qubits but suffice for initial quantum computing.

Clicking on a system reveals its architecture and qubit connections; you can only entangle physically connected qubits.



For real quantum computing, create a quantum circuit, typically in a Jupyter notebook. Import necessary libraries, define the circuit, and connect to a quantum device. Use an API token from the IBM Quantum dashboard for account authentication, ensuring token security. Connect to the provider and select the backend, such as Quito.

Transpile the circuit to adapt it to hardware requirements, ensuring compatibility. Assemble the circuit, send it to the backend, and receive a job object for request management. Track job status, runtime, and job ID via the IBM Quantum dashboard.

Upon job completion, retrieve results, representing qubit measurements. Note that qubits may yield

In [ ]: