Open source software in quantum computing

Mark Fingerhuth, Tomas Babej, Peter Wittek

PLOS ONE | https://doi.org/10.1371/journal.pone.0208561 December 20, 2018

The usage of Qiskit

Quantum “Hello World!” program via Qiskit

Y. Ma
yingjin.ma@sccas.cn or yingjin_ma@163.com
Dec. 4 2019

Open source software in quantum computing

Mark Fingerhuth, Tomas Babej, Peter Wittek

Mark Fingerhuth

Head of R&D - Co-Founder

Mark is one of the first applied quantum programmers. His published thesis was on implementing the first quantum machine learning algorithm on superconducting gate-based quantum computers.

Tomas Babej

CTO - Co-Founder

Tomas has a double MSc degree in computer science, with focus on machine learning, cybersecurity and quantum computing. Before cofounding ProteinQure, he mastered the art of software engineering at Red Hat.

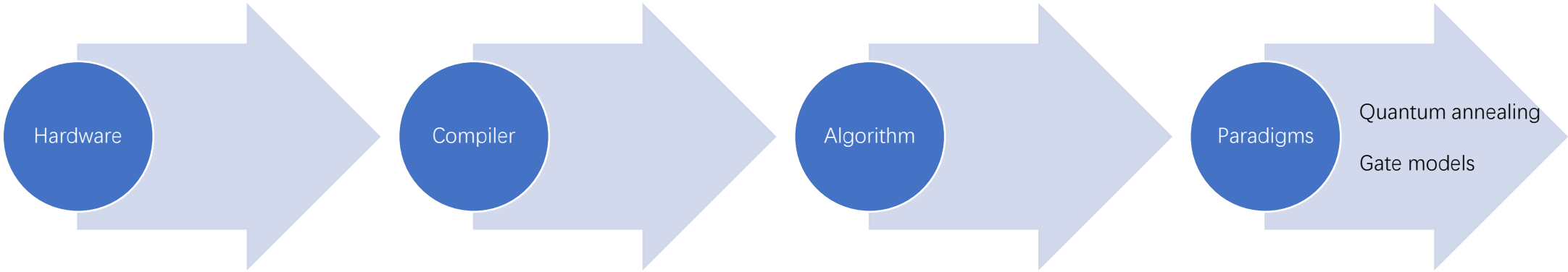
Peter Wittek

Assistant Professor in the University of Toronto working on quantum-enhanced machine learning and applications of high-performance learning algorithms in quantum physics.

Cofounder of the Quantum Open Source Foundation

The main thing/idea for this article :

1. review a wide range of open source software for quantum computing



"covering all stages of the quantum toolchain"

2. evaluation of each project

Why open source in quantum computing?

Reproducibility

a core tenet of science

Impact and publicity

crucial for both scientific and commercial endeavors

Building a community and ecosystem

steep learning curve that needs to be overcome, therefore it is in the best interest of quantum hardware companies to get more developers involved

Gaining credit and increasing human capital

Software projects in quantum computing

Discrete variable gate-model quantum computing

- ✓ bits are replaced by qubits
- ✓ logical transformations by a finite set of unitary gates
- ✓ Most popular in hardware

Continuous variable gate-model quantum computing

- ✓ qubits are replaced by qumodes
- ✓ closer to the physics way of thinking, e.g. in quantum optics
- ✓ Most popular language in describing circuits

Adiabatic quantum computation

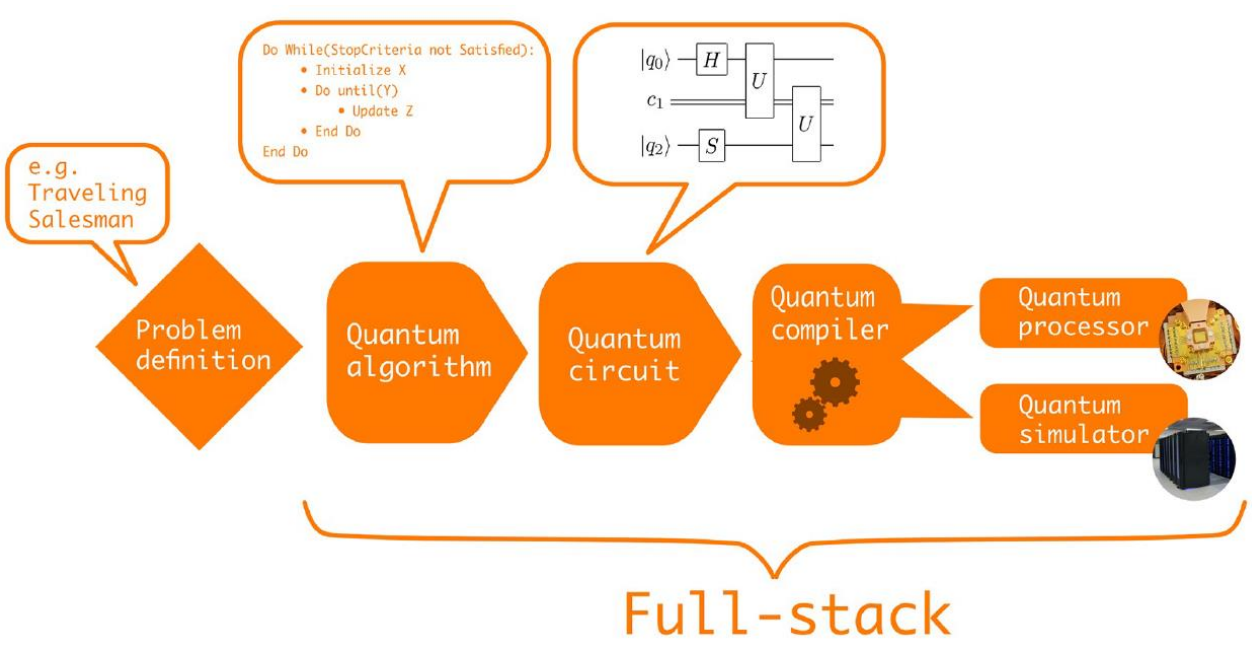
- ✓ Quantum annealing devices
- ✓ Only for some understanding of statistical physics

Quantum simulators

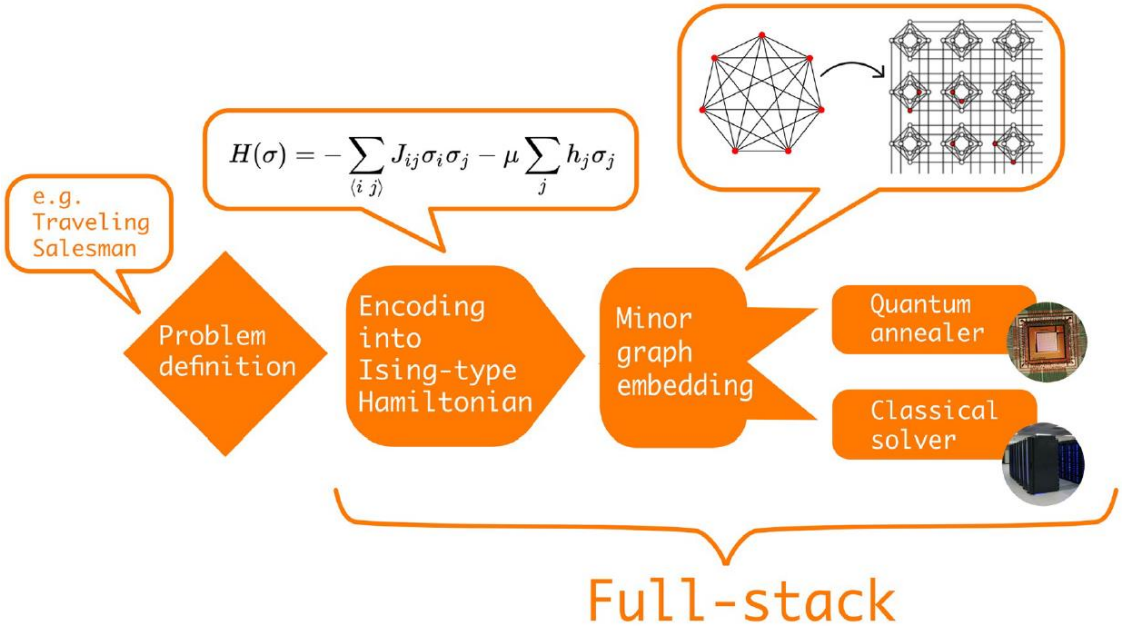
- ✓ Original motivation behind quantum computing
- ✓ Application-specific

Visualization of a typical quantum algorithm workflow

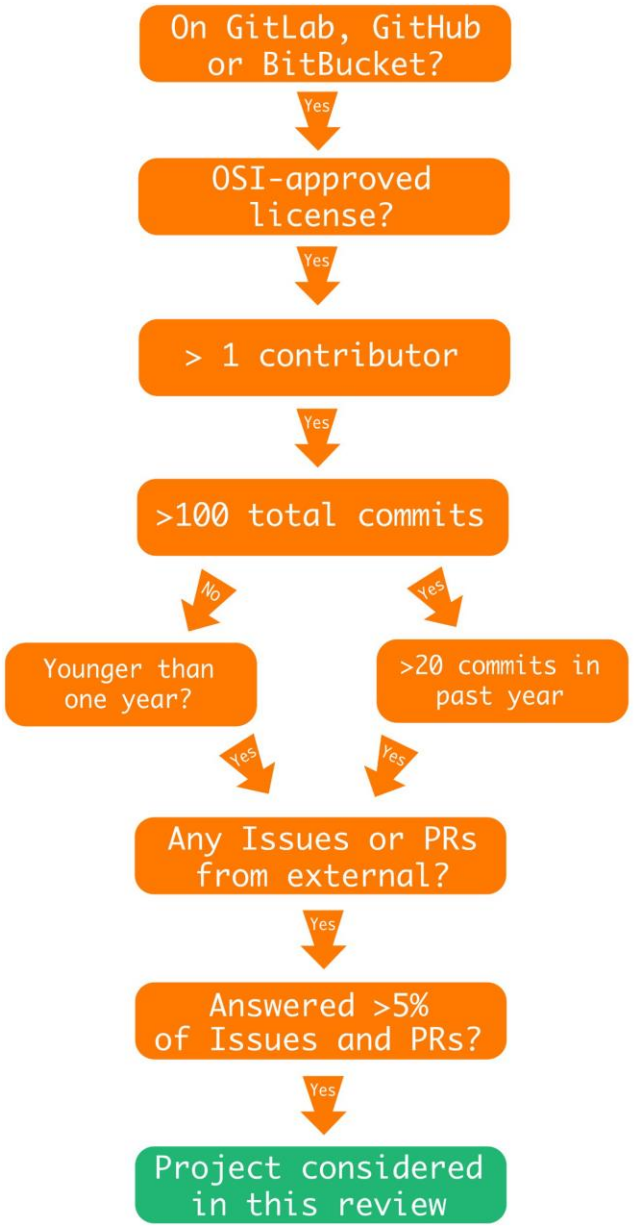
On a gate-model quantum computer:



On a quantum annealer:



Their standard for considering/evaluating open source Quantum Computing software:



Projects considered:

Qiskit

Name	Tagline	Programming language	Licence	Supported OS
Cirq	Framework for creating, editing, and invoking Noisy Intermediate Scale Quantum (NISQ) circuits.	Python	Apache-2.0	Windows, Mac, Linux
Cliffords.jl	Efficient calculation of Clifford circuits in Julia.	Julia	MIT	Windows, Mac, Linux
dimod	Shared API for Ising/quadratic unconstrained binary optimization samplers.	Python	Apache-2.0	Windows, Linux, Mac
dwave-system	Basic API for easily incorporating the D-Wave system as a sampler in the D-Wave Ocean software stack.	Python	Apache-2.0	Linux, Mac
FermiLib	Open source software for analyzing fermionic quantum simulation algorithms.	Python	Apache-2.0	Windows, Mac, Linux
Forest (pyQuil & Grove)	Simple yet powerful toolkit for writing hybrid quantum-classical programs.	Python	Apache-2.0	Windows, Mac, Linux
OpenFermion	The electronic structure package for quantum computers.	Python	Apache-2.0	Windows, Mac, Linux
ProjectQ	An open source software framework for quantum computing.	Python, C++	Apache-2.0	Windows, Mac, Linux
PyZX	Python library for quantum circuit rewriting and optimisation using the ZX-calculus.	Python	GPL-3.0	Windows, Mac, Linux
QGL.jl	A performance orientated QGL compiler.	Julia	Apache-2.0	Windows, Mac, Linux
Qbsolv	Decomposing solver that finds a minimum value of a large quadratic unconstrained binary optimization problem by splitting it into pieces.	C	Apache-2.0	Windows, Linux, Mac
Qiskit Terra & Aqua	Quantum Information Science Kit for writing experiments, programs, and applications.	Python, C++	Apache-2.0	Windows, Mac, Linux
Qiskit Tutorials	A collection of Jupyter notebooks using Qiskit.	Python	Apache-2.0	Windows, Mac, Linux
Qiskit.js	Quantum Information Science Kit for JavaScript.	JavaScript	Apache-2.0	Windows, Mac, Linux
Qrack	Comprehensive, GPU accelerated framework for developing universal virtual quantum processors.	C++	GPL-3.0	Linux, Mac
Quantum Fog	Python tools for analyzing both classical and quantum Bayesian networks.	Python	BSD-3-Clause	Windows, Mac, Linux
Quantum++	A modern C++11 quantum computing library.	C++, Python	MIT	Windows, Mac, Linux
Qubiter	Python tools for reading, writing, compiling, simulating quantum computer circuits.	Python, C++	BSD-3-Clause	Windows, Mac, Linux
Quirk	Drag-and-drop quantum circuit simulator for your browser to explore and understand small quantum circuits.	JavaScript	Apache-2.0	Windows, Mac, Linux
reference-qvm	A reference implementation for a Quantum Virtual Machine in Python.	Python	Apache-2.0	Windows, Mac, Linux
ScaffCC	Compilation, analysis and optimization framework for the Scaffold quantum programming language.	C++, Objective C, LLVM	BSD-2-Clause	Linux, Mac
Strawberry Fields	Full-stack library for designing, simulating, and optimizing continuous variable quantum optical circuits.	Python	Apache-2.0	Windows, Mac, Linux
XACC	eXtreme-scale Accelerator programming framework.	C++	Eclipse PL-1.0	Windows, Mac, Linux
XACC VQE	Variational quantum eigensolver built on XACC for distributed, and shared memory systems.	C++	BSD-3-Clause	Windows, Mac, Linux

quantum processing unit (QPU, Hardware)

Feature overview:

Name	Quantum computing paradigm	Quantum algorithms	Quantum circuits	Quantum compiler	Quantum computer simulator	QPU backend	Full-stack
Cirq	Discrete gate model	✓	✓	✓	✓	✗	✓
Cliffords.jl	Discrete gate model	✗	✓	✗	✓	✗	✗
FermiLib	Discrete gate model	✓	✗	✗	✗	✗	✗
Forest (pyQuil & Grove)	Discrete gate model	✓	✓	✓	✓	✓	✓
OpenFermion	Discrete gate model	✓	✓	✗	✗	✗	✗
ProjectQ	Discrete gate model	✓	✓	✓	✓	✓	✓
PyZX	Discrete gate model	✗	✗	✓	✗	✗	✗
QGL.jl	Discrete gate model	✗	✗	✓	✗	✗	✗
Qiskit Terra & Aqua	Discrete gate model	✓	✓	✓	✓	✓	✓
Qiskit Tutorials	Discrete gate model	✓	✗	✗	✗	✗	✗
Qiskit.js	Discrete gate model	✓	✓	✓	✓	✓	✓
Qrack	Discrete gate model	✗	✓	✓	✓	✗	✗
Quantum Fog	Discrete gate model	✓	✓	✗	✗	✗	✗
Quantum++	Discrete gate model	✗	✓	✗	✓	✗	✗
Qubiter	Discrete gate model	✓	✓	✓	✓	✓	✓
Quirk	Discrete gate model	✓	✓	✗	✓	✗	✗
reference-qvm	Discrete gate model	✗	✓	✗	✓	✗	✗
ScaffCC	Discrete gate model	✗	✗	✓	✗	✗	✗
Strawberry Fields	Continuous gate model	✓	✓	✓	✓	✗	✓
XACC	Discrete gate model	✓	✓	✓	✓	✓	✓
XACC VQE	Discrete gate model	✓	✗	✗	✗	✗	✗

Qiskit

Name	Hardware platform	Hamiltonian generation	Minor embedding	Post- processing	Classical solver	QPU backend	Full-stack
dimod	Quantum annealing	✗	✓	✓	✓	✓	✗
dwave-system	Quantum annealing	✗	✓	✓	✓	✓	✗
Qbsolv	Quantum annealing	✗	✗	✗	✓	✓	✗

Why Qiskit “split”?

Qiskit API documentation

Qiskit is an open-source framework for working with quantum computers at the level of circuits, pulses, and algorithms.

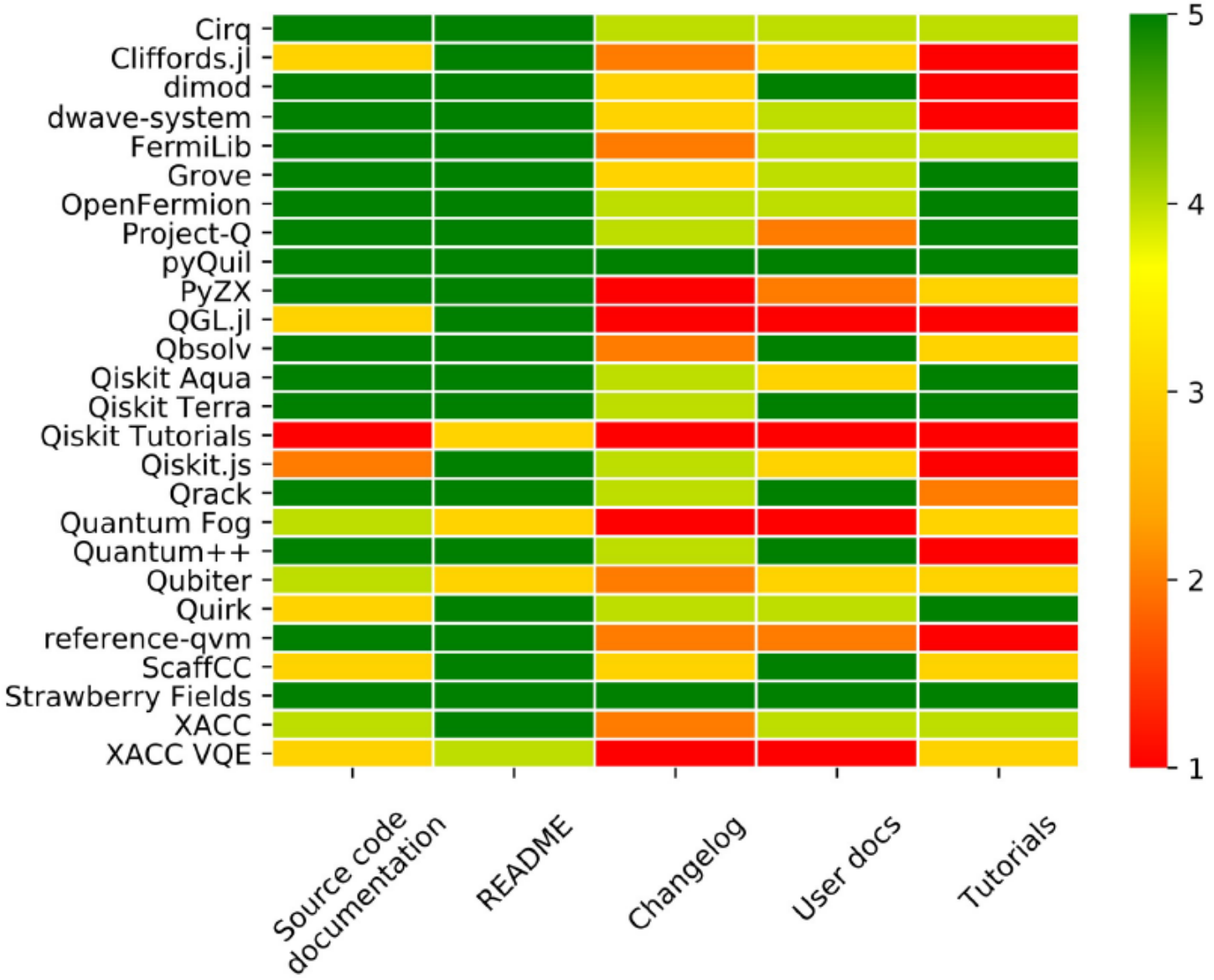
A central goal of Qiskit is to build a software stack that makes it easy for anyone to use quantum computers. However, Qiskit also aims to facilitate research on the most important open issues facing quantum computation today.

You can use Qiskit to easily design experiments and run them on simulators and real quantum computers.

Qiskit consists of four foundational elements:

- Qiskit Terra: Composing quantum programs at the level of circuits and pulses with the code foundation.
- Qiskit Aer: Accelerating development via simulators, emulators, and debuggers
- Qiskit Ignis: Addressing noise and errors
- Qiskit Aqua: Building algorithms and applications

Heatmap of documentation analysis results:



Evaluation results for the community analysis:

Project	Roadmap	Releases	Contributors	User-discussion channels	Developer-discussion channels	Public review processs	Community profile
Cirq	✗	✓	28	Stack Exchange	-	E+I	4/7
Cliffords.jl	✗	✓	7	-	-	E	3/7
dimod	✗	✓	11	Forum	-	E+I	5/7
dwave-system	✗	✓	6	Forum	-	E+I	4/7
FermiLib	✗	✓	10	-	-	E+I	3/7
Forest - Grove	✗	✓	24	Slack	Slack	E+I	3/7
Forest - pyQuil	✗	✓	46	Slack	Slack	E+I	3/7
OpenFermion	✗	✓	26	-	-	E+I	3/7
ProjectQ	✗	✓	10	-	-	E+I	3/7
PyZX	✗	✗	3	-	-	-	3/7
QGL.jl	✗	✗	3	-	-	E+I	3/7
Qbsolv	✗	✓	18	Forum	-	E+I	5/7
Qiskit Aqua	✗	✓	14	Forum	-	E+I	7/7
Qiskit Terra	✓	✓	67	Forum, Slack	Slack	E+I	7/7
Qiskit Tutorials	✗	✗	37	-	-	E+I	3/7
Qiskit.js	✗	✓	4	Forum	-	E	7/7
Qrack	✗	✓	2	-	-	E+I	3/7
Quantum Fog	✗	✗	2	-	-	E	3/7
Quantum++	✗	✓	3	Gitter	-	E	5/7
Qubiter	✗	✗	2	-	-	E	3/7
Quirk	✗	✓	3	-	-	E	4/7
reference-qvm	✗	✓	8	-	-	E+I	3/7
ScaffCC	✗	✓	7	-	-	E	3/7
Strawberry Fields	✗	✓	5	Slack	Slack	E+I	7/7
XACC	✗	✗	6	-	-	E	4/7
XACC VQE	✗	✗	2	-	-	E	3/7

Evaluation results for the static analysis of each project and its source code:

pull requests (PR)

Name	Version control system	Issue tracking system	Issues/ PRs	Attention rate	Average response time (days)	Test suite	Code coverage	Complexity
Cirq	Git	GitHub	448/686	0.54	2.6	✓	94%	2.99
Cliffords.jl	Git	GitHub	6/12	0.33	<1	✓	-	-
dimod	Git	GitHub	110/201	0.30	5.3	✓	94%	2.96
dwave-system	Git	GitHub	54/72	0.24	8.2	✓	87%	3.47
FermiLib	Git	GitHub	24/134	0.31	<1	✓	99%	2.43
Forest - Grove	Git	GitHub	53/130	0.51	17.7	✓	72%	3.25
Forest - pyQuil	Git	GitHub	293/385	0.41	10.6	✓	88%	2.65
OpenFermion	Git	GitHub	137/345	0.61	1.3	✓	100%	2.46
ProjectQ	Git	GitHub	84/198	0.75	4.0	✓	100%	4.02
PyZX	Git	GitHub	6/2	0.80	<1	✓	51%	4.42
QGL.jl	Git	GitHub	17/13	0.75	130.6	✓	-	-
Qbsolv	Git	GitHub	50/85	0.17	22.2	✓	95%	-
Qiskit Aqua	Git	GitHub	43/141	0.20	1.8	✓	67%	3.04
Qiskit Terra	Git	GitHub	526/713	0.11	16.0	✓	76%	2.56
Qiskit Tutorials	Git	GitHub	94/274	0.40	8.6	✗	-	-
Qiskit.js	Git	GitHub	19/8	0.33	4.4	✓	66%	-
Qrack	Git	GitHub	7/78	0.07	8.7	✓	87%	-
Quantum Fog	Git	GitHub	17/1	1.00	<1	✗	0%	3.32
Quantum++	Git	GitHub	8/45	0.88	<1	✓	72%	-
Qubiter	Git	GitHub	14/3	0.75	<1	✗	0%	-
Quirk	Git	GitHub	286/131	0.96	<1	✓	-	-
reference-qvm	Git	GitHub	6/14	0.44	75.6	✓	80%	3.99
ScaffCC	Git	GitHub	15/11	0.18	10.1	✓	-	-
Strawberry Fields	Git	GitHub	16/20	0.73	1.2	✓	97%	2.70
XACC	Git	GitHub	65/14	0.65	<1	✓	-	-
XACC VQE	Git	GitHub	22/4	0.33	8.8	✓	-	-

Conclusions :

These open source projects lowers the barrier to learn quantum computing

It reflects the same process that happened in machine learning

Lack of standardization in the field

Multiple players develop competing software platforms

Lack of stand-alone quantum compilers

Most compilers are either proprietary, closed-source or absorbed into quantum full-stack libraries

More @ live website (https://qosf.org/)

The usage of Qiskit

The easiest way :

1. Search “anaconda python”
2. Download the Python3.7 version
3. Install it via “sh Anaconda3-2019.10-Linux-x86_64.sh”
    a. 可修改默认安装目录
    b. 建议安装完后允许其自动shell配置
    c. 同时“conda config –set auto_activate_base false”避免自动环境激活
4. “conda create –n qiskit-py37 python=3.7”
5. “conda activate qiskit-py37”
6. “pip install qiskit”

includes: **qiskit qiskit_terra qiskit_aer qiskit-ibmq-provider qiskit_ignis qiskit_aqua**
marshmallow scipy networkx jsonschema numpy psutil sympy ply nest_asynic
websockes cvxopt Quandl fastdtw docplex scikit_learn h5py pyscf dlx six
decorator attrs importlib_metadata pyrsistent mpmath urllib3 idna chardet
cryptography ntlm_auth python_detautil pyasn1 ndg_httpsclient inflection
pyOpenSSL pandas more_itertools docloud joblib zipp cffi pytz pycparse

The usage of Qiskit

Additional packages are suggested :

(especially when you follow the tutorials in YouTube of Qiskit)

(in the “qiskit-py37” environments)

1. “conda install jupyter matplotlib”
2. “jupyter notebook”

The usage of Qiskit

jupyter Untitled Last Checkpoint: 2 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

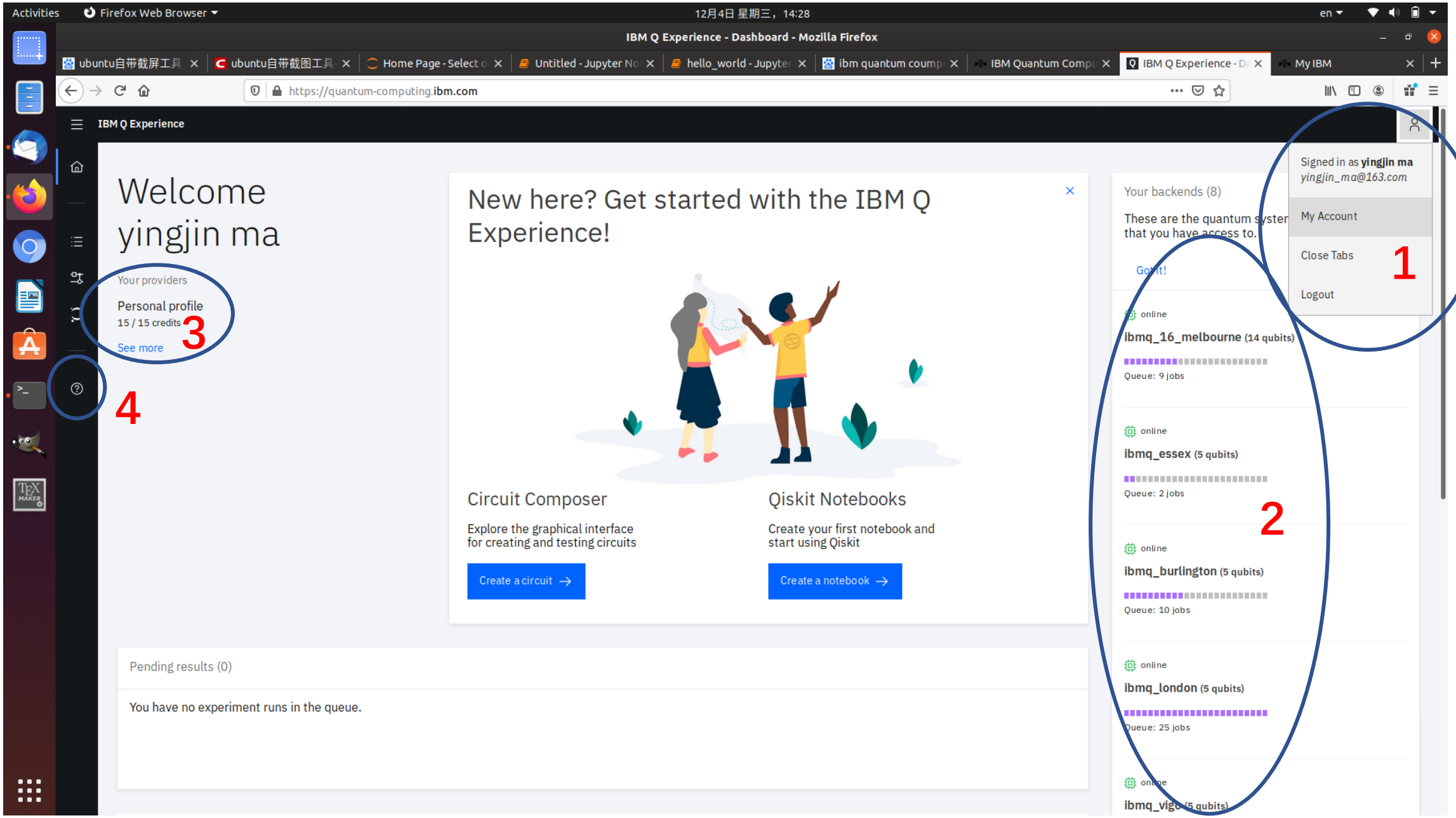
```
In [1]: import qiskit

In [2]: qiskit.__qiskit_version__
Out[2]: {'qiskit-terra': '0.10.0',
         'qiskit-aer': '0.3.2',
         'qiskit-ignis': '0.2.0',
         'qiskit-ibmq-provider': '0.3.3',
         'qiskit-aqua': '0.6.1',
         'qiskit': '0.13.0'}

In [3]: from qiskit import IBMQ

In [4]: IBMQ.save_account('71c49e2ff11cc9911467babb0e9c48bfc4b5283c8a571656ec4d0972da654cfd9b6c3c1deba2bf381577c4a5e7a528e4d2a569c

In [5]: IBMQ.load_account()
Out[5]: <AccountProvider for IBMQ(hub='ibm-q', group='open', project='main')>

In [ ]:
```

Remote QPU servers

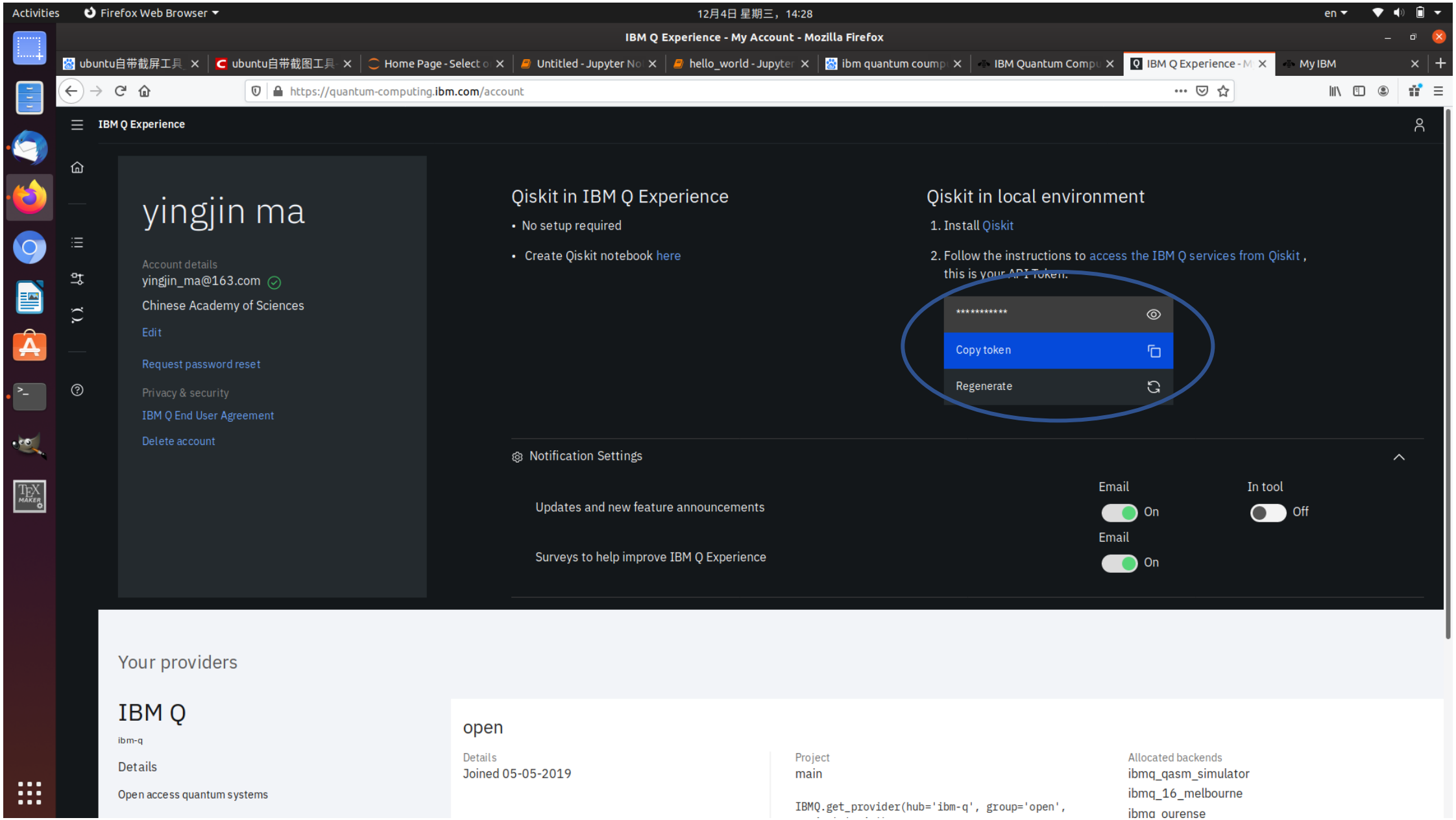
Your account

https://quantum-computing.ibm.com

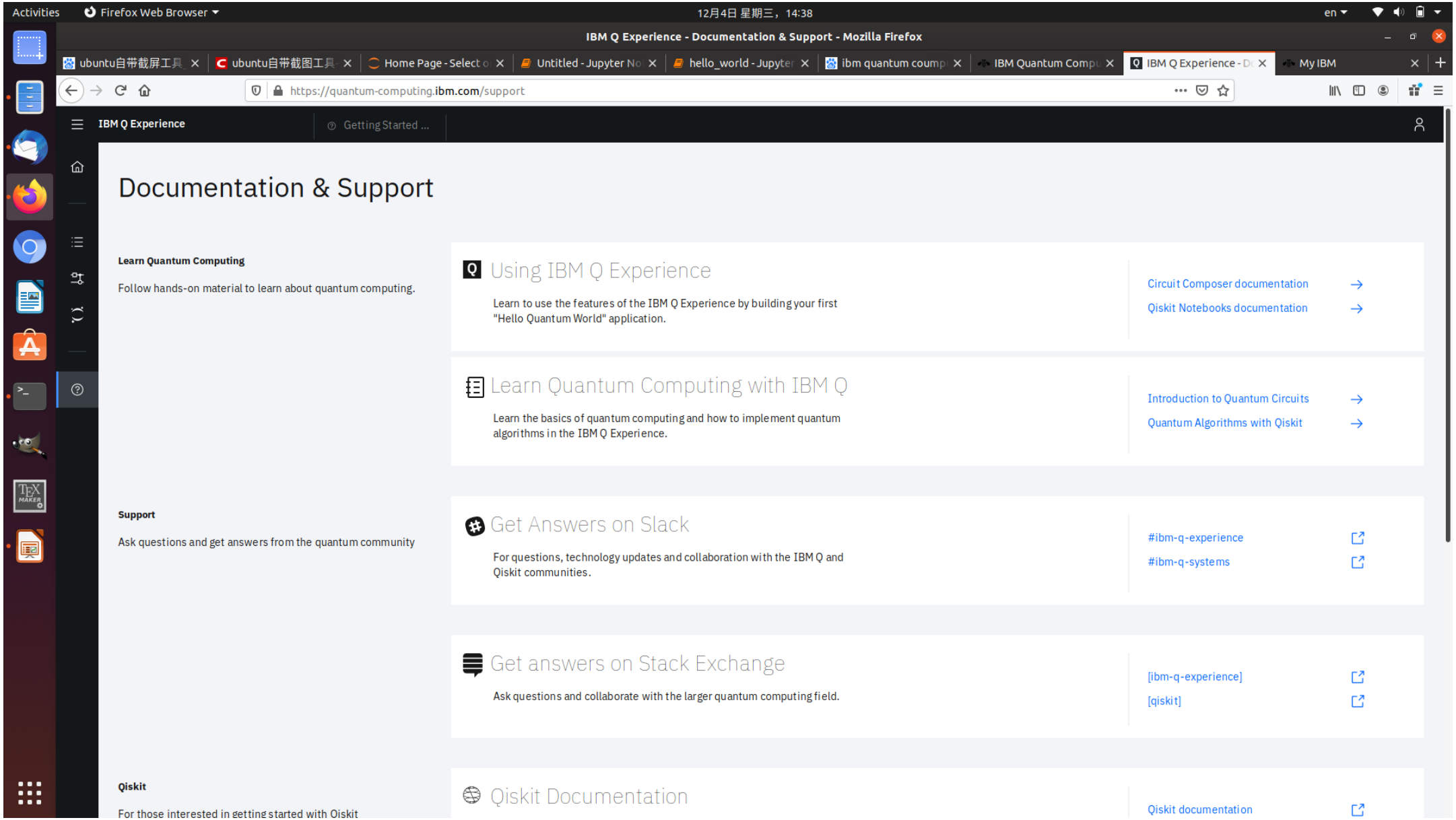
https://quantum-computing.ibm.com （Overview）



https://quantum-computing.ibm.com （Account/Token）

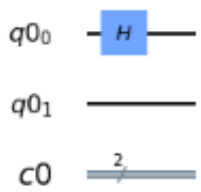


https://quantum-computing.ibm.com (Documentation & Support)



Quantum “Hello World!” program via Qiskit

```
In [1]: from qiskit import *

In [2]: qr = QuantumRegister(2)

In [3]: cr = ClassicalRegister(2)

In [4]: circuit = QuantumCircuit(qr,cr)

In [5]: %matplotlib inline

In [6]: circuit.draw()
Out[6]:
q0_0: |0>
q0_1: |0>
 c0_0: 0
 c0_1: 0

In [7]: circuit.h(qr[0])
Out[7]: <qiskit.circuit.instructionset.InstructionSet at 0x7feef38e1250>

In [8]: circuit.draw(output='mpl')
Out[8]:
```

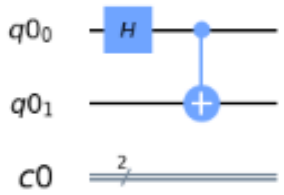



```
In [9]: circuit.cx(qr[0],qr[1])
Out[9]: <qiskit.circuit.instructionset.InstructionSet at 0x7feef38ff090>
```

Quantum “Hello World!” program via Qiskit

```
In [10]: circuit.draw(output='mpl')
```

Out[10]:

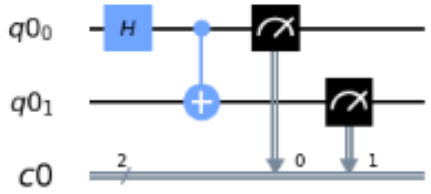



```
In [11]: circuit.measure(qr,cr)
```

```
Out[11]: <qiskit.circuit.instructionset.InstructionSet at 0x7feef381bdd0>
```

```
In [12]: circuit.draw(output = 'mpl')
```

Out[12]:

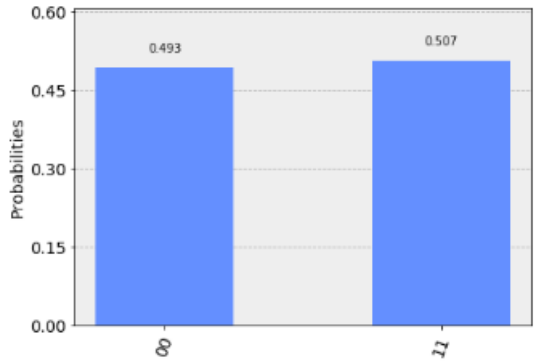



```
In [13]: simulator = Aer.get_backend('qasm_simulator')
```

```
In [14]: result = execute(circuit, backend = simulator).result()
```

```
In [15]: from qiskit.tools.visualization import plot_histogram
```

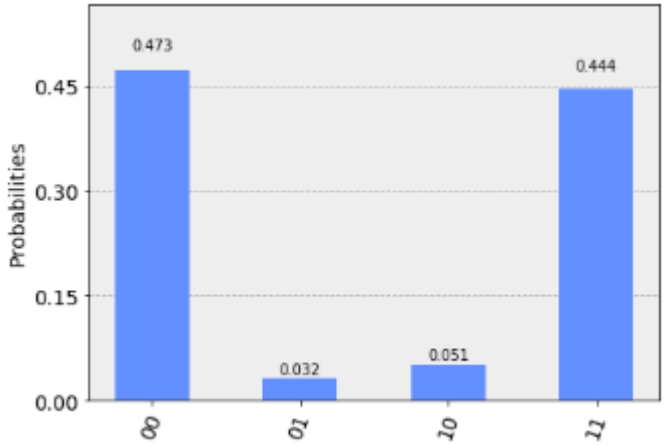
```
In [16]: plot_histogram(result.get_counts(circuit))
```



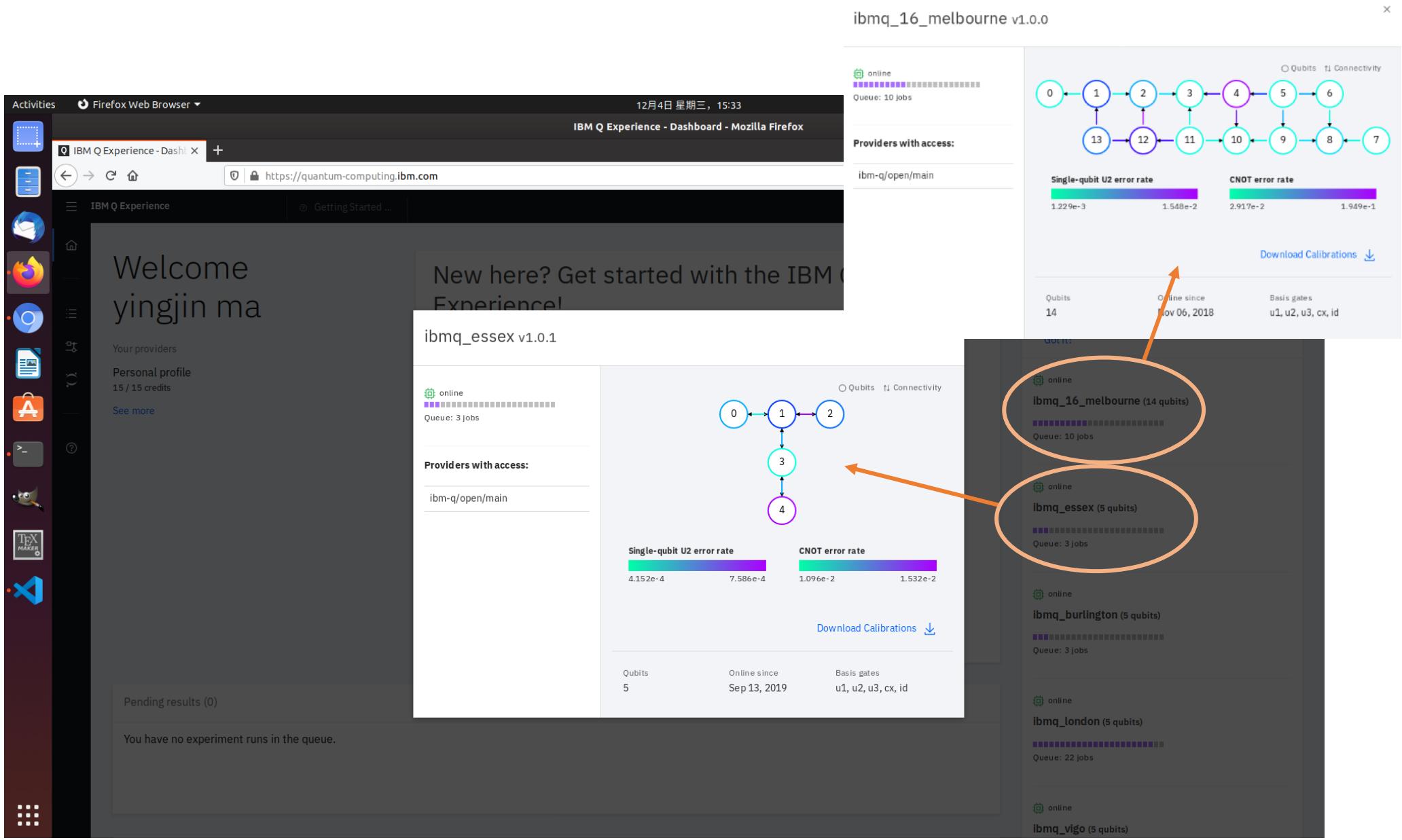
Quantum “Hello World!” program via Qiskit

```
In [18]: IBMQ.load_account()
Out[18]: <AccountProvider for IBMQ(hub='ibm-q', group='open', project='main')>

In [19]: provider = IBMQ.get_provider('ibm-q')

In [20]: qcomp = provider.get_backend('ibmq_essex')

In [21]: job = execute (circuit, backend=qcomp)

In [22]: from qiskit.tools.monitor import job_monitor

In [23]: job_monitor(job)
         Job Status: job has successfully run

In [24]: result = job.result()

In [25]: plot_histogram(result.get_counts(circuit))
```

(remote server)



Quantum “Hello World!” program via Qiskit （remote server）



Quantum “Hello World!” program via Qiskit (jobs status)