Quantum Computing
A Practical Perspective

Marco Venere

marco.venere@polimi.it

November 20th, 2024
Politecnico di Milano

POLITECNICO MILANO 1863
NECST laboratory

Agenda

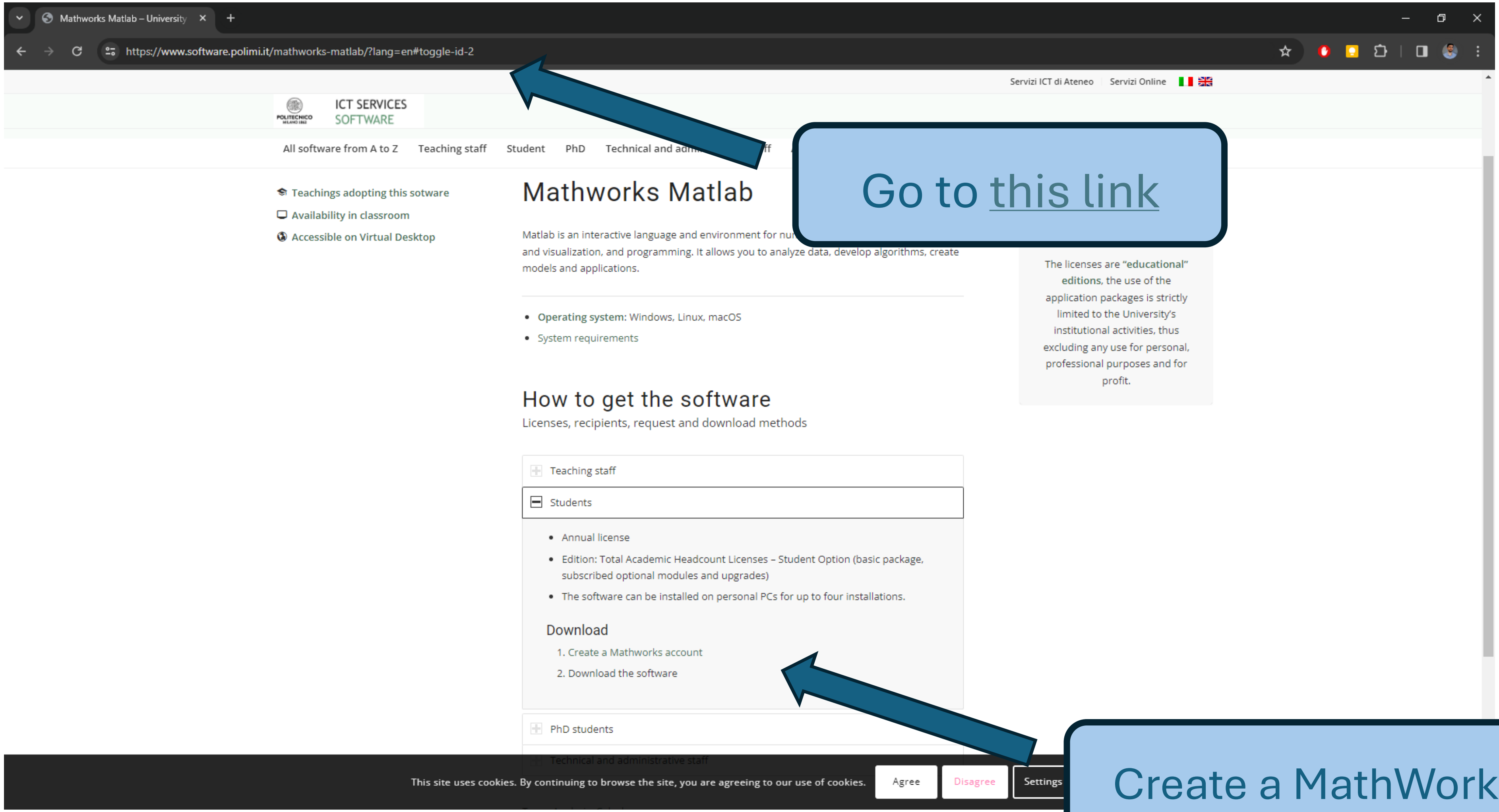
- November 7th → Theory Recap on Quantum Computing
- November 20th → Initial Setup and First Experiments
- November 21st → Grover's Algorithm
- November 25th → Combinatorial Optimization
- November 28th → VQE, QNN, QMC
- December 3rd → Quantum Error Correction & Mitigation – Projects Presentation

(it may be subject to variations)

Initial Setup: Steps To Do

- Get a MathWorks MATLAB license from polimi
- Install MATLAB R2024b on your local machine
- Install MATLAB Support Package for Quantum Computing
- Sign up to IBM Quantum

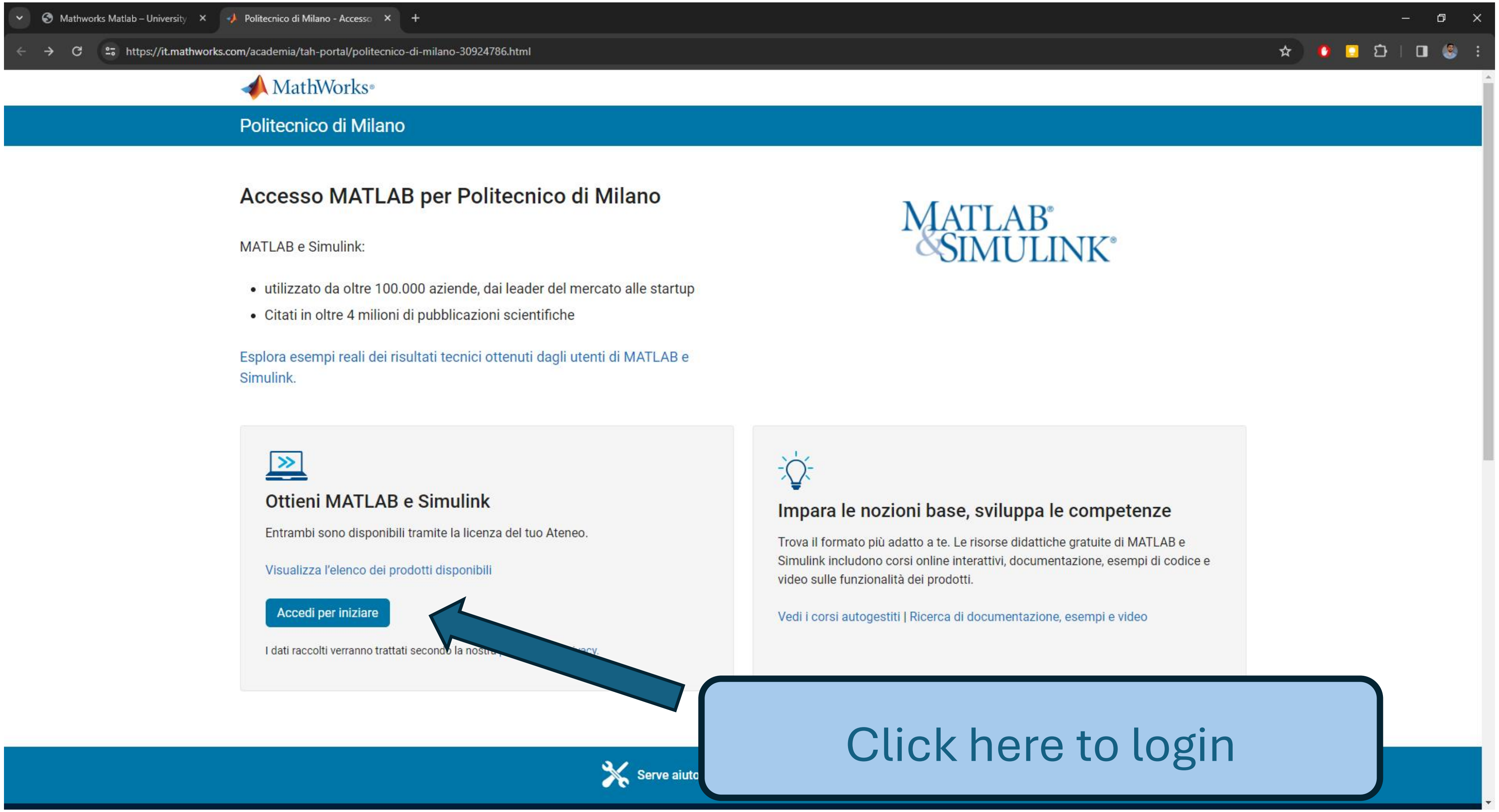
Get a license



Go to this link

Create a MathWorks Account

Get a license

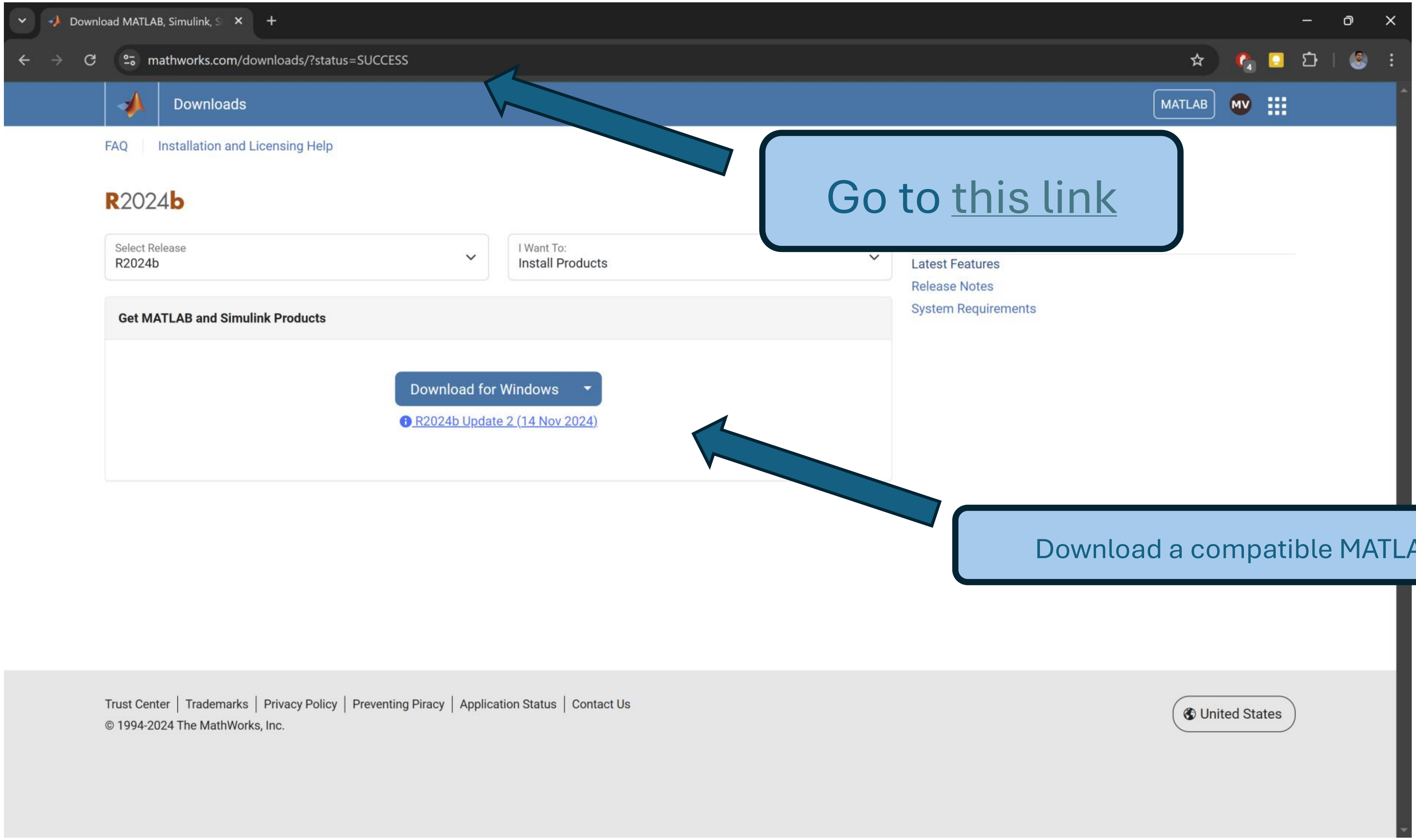


Click here to login

Get a license

- Use your @mail.polimi.it e-mail as a MathWorks account
- Follow all the steps and fill in the required data
- At the end of the process, your account should be correctly enabled with a valid license

Install MATLAB R2024b



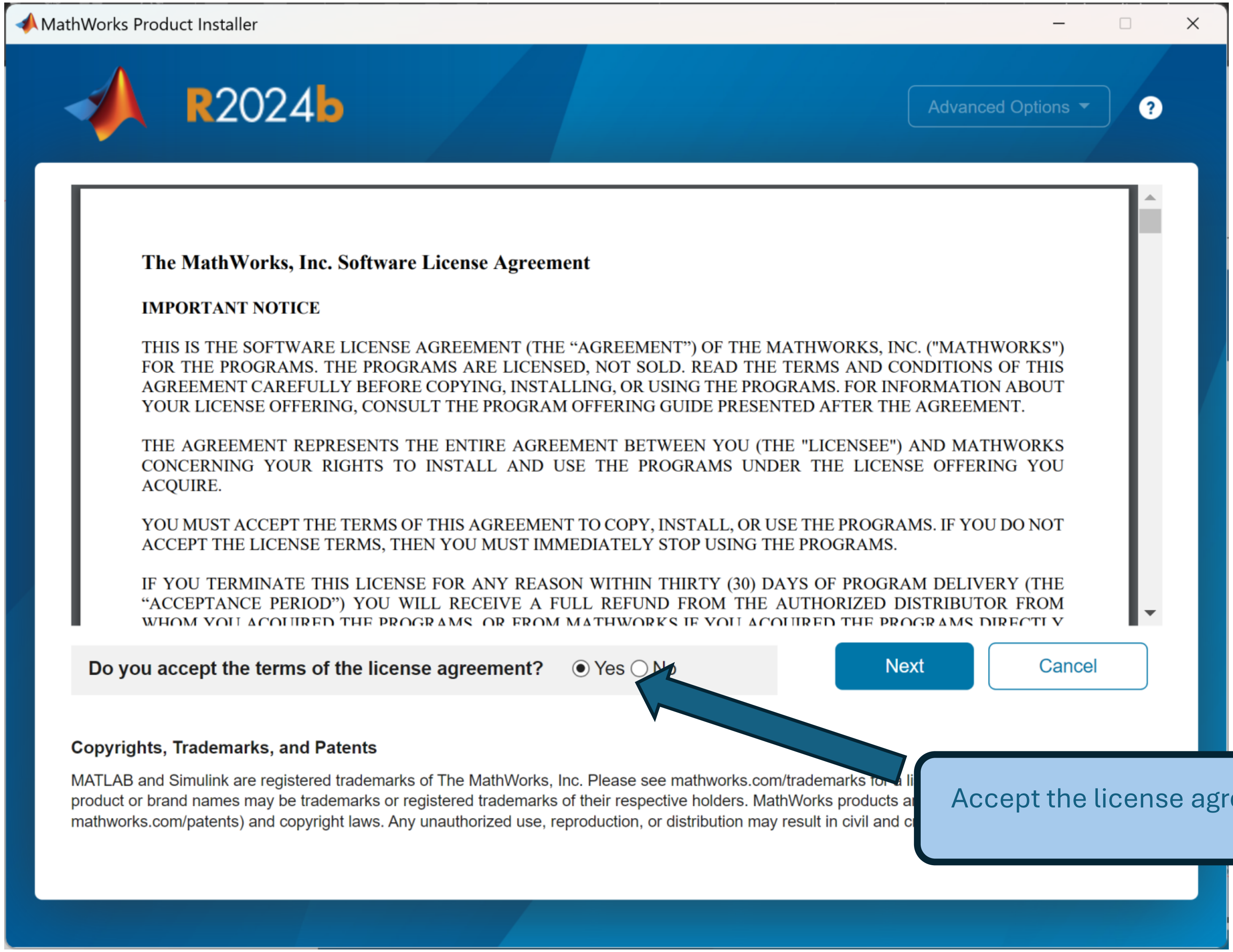
Go to [this link]

Download a compatible MATLAB version

Install MATLAB R2024b

Login with your @mail.polimi.it (Polimi SSO login)

Install MATLAB R2024b



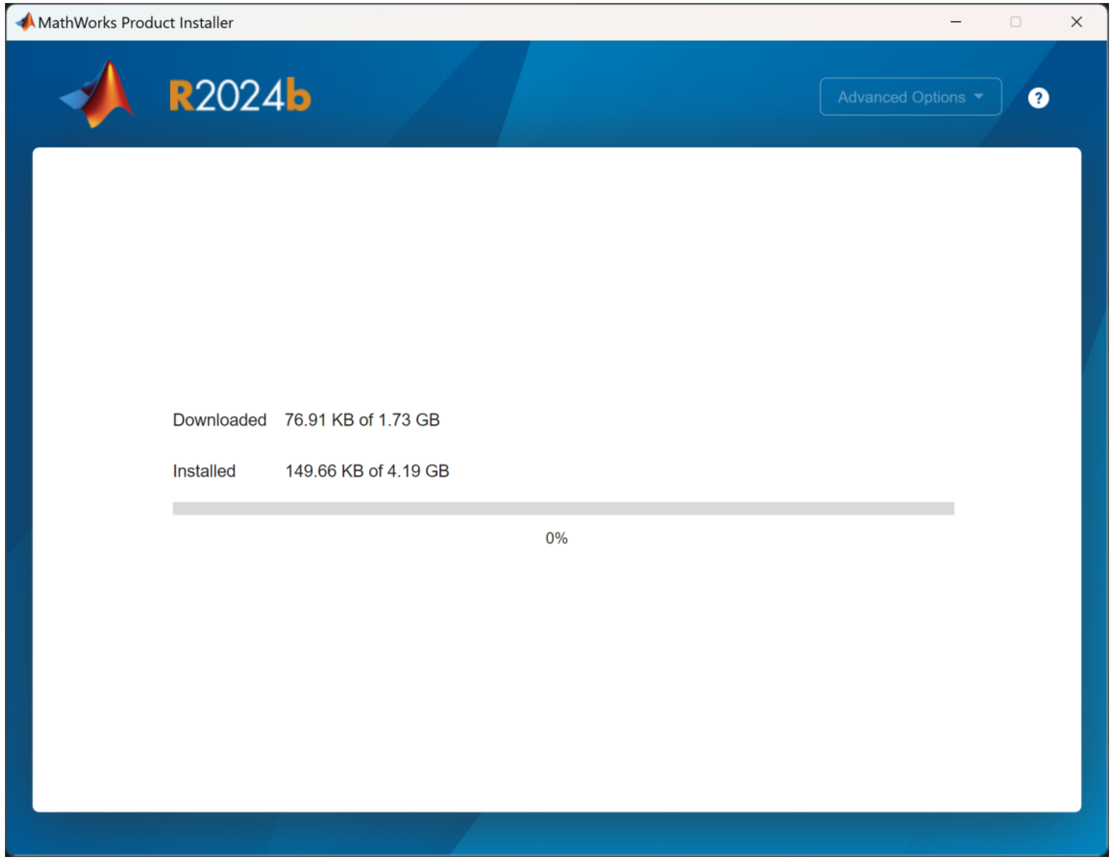
Accept the license agreement (after carefully reading it)

Install MATLAB R2024b

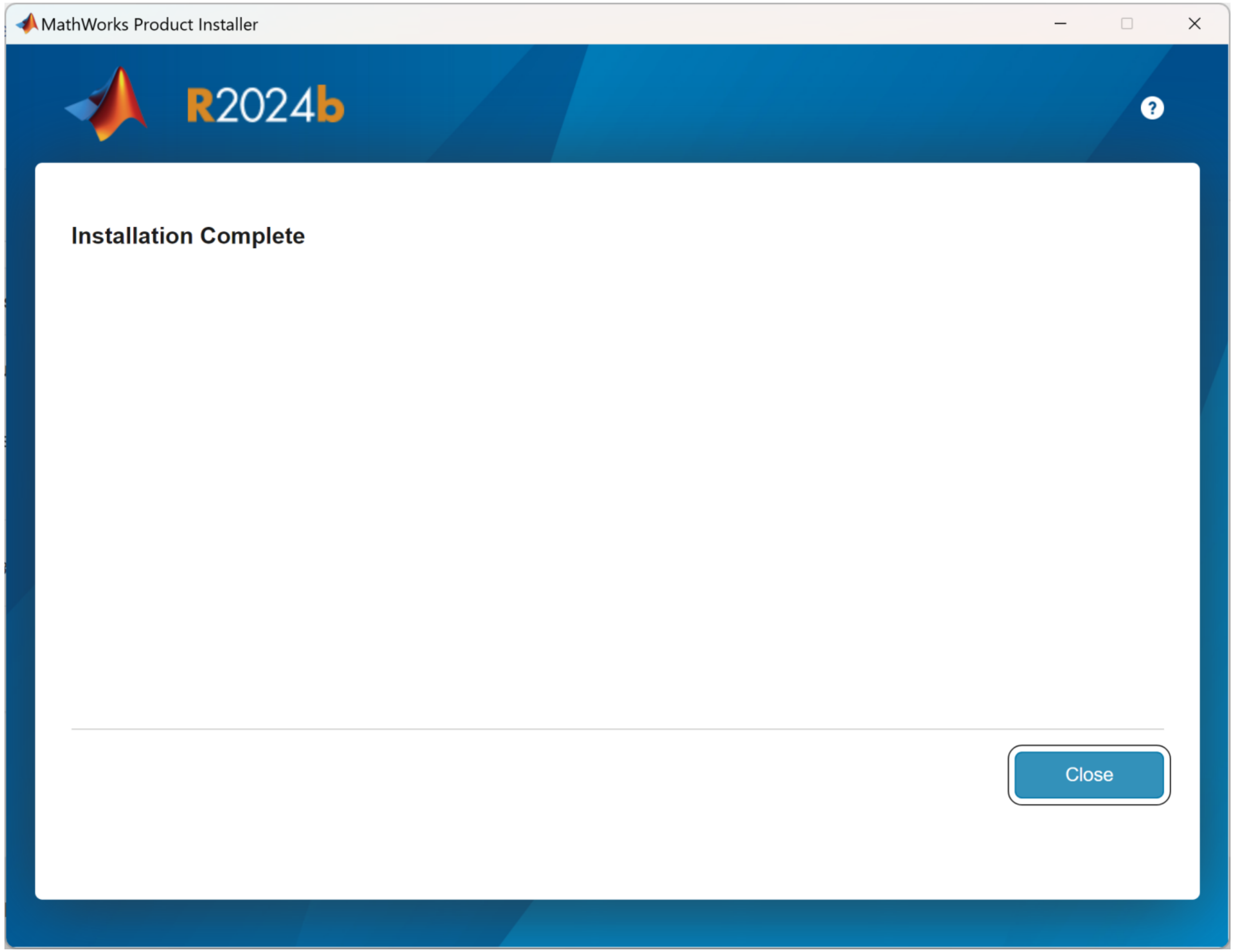
- Press “Next” a bunch of times and then “Begin Install”

Install MATLAB R2024b

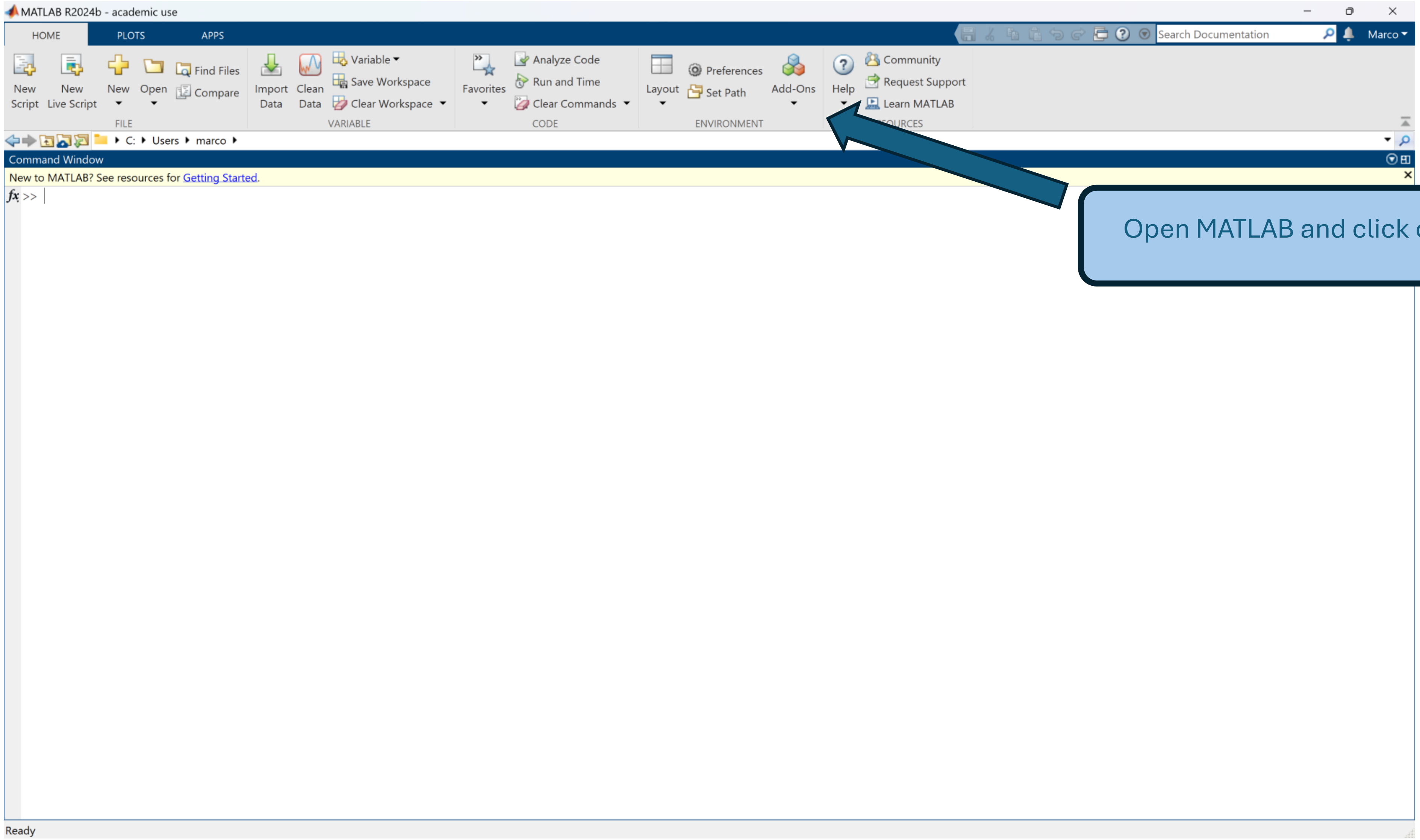
- Press “Next” a bunch of times and then “Begin Install”



Install MATLAB R2024b

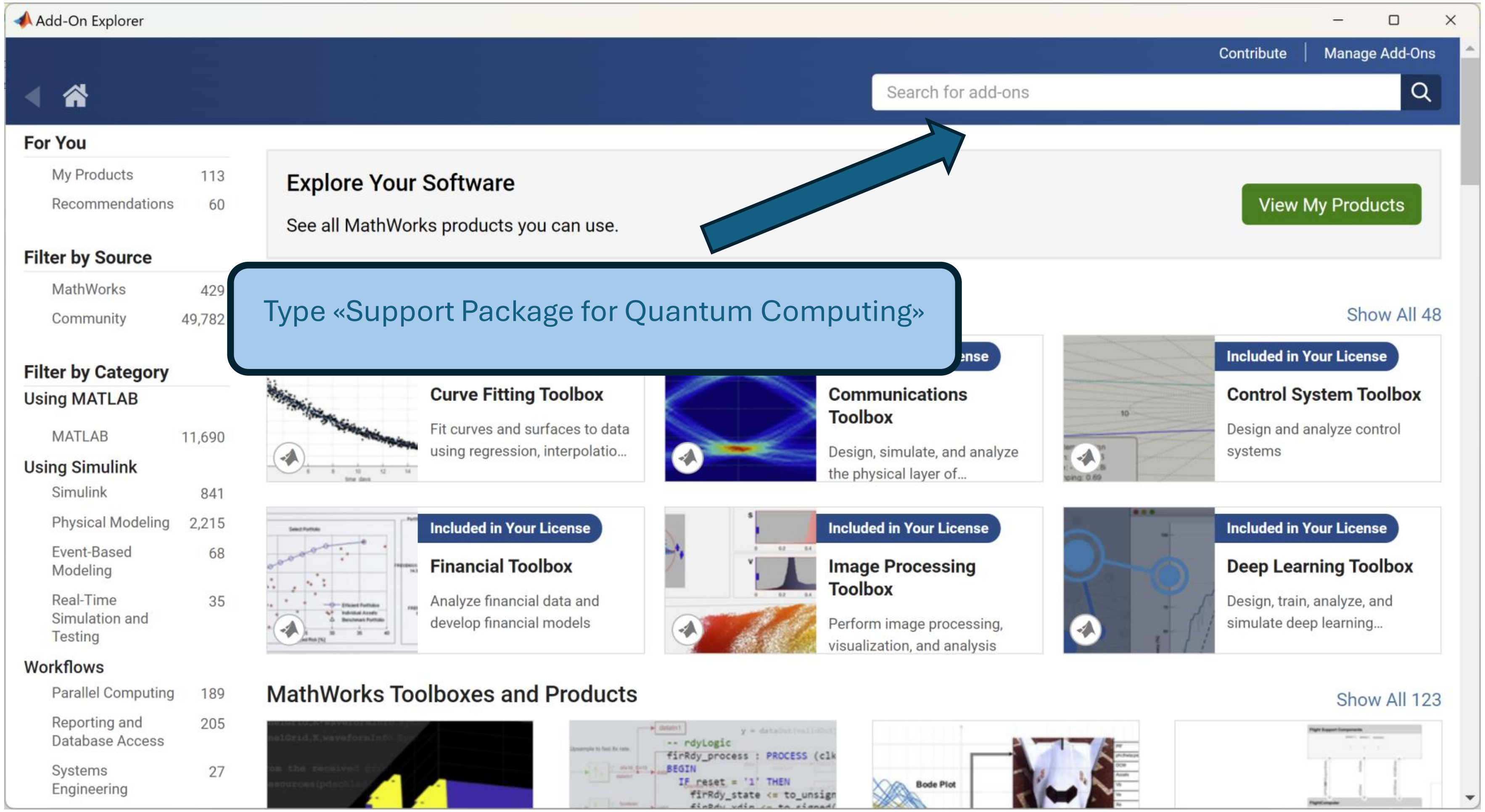


Install MATLAB Support Package for QC



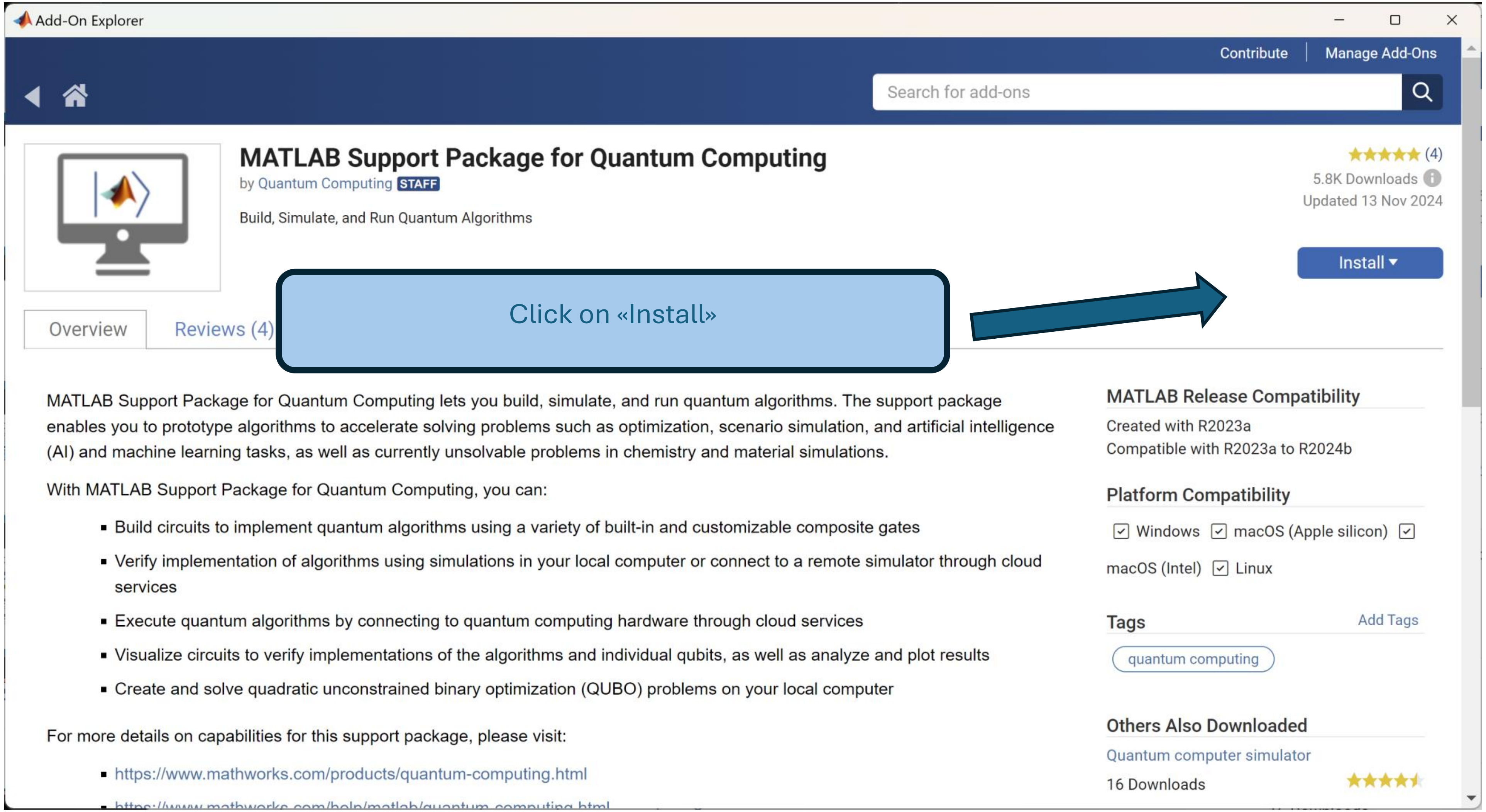
Open MATLAB and click on «Get Add-Ons»

Install MATLAB Support Package for QC

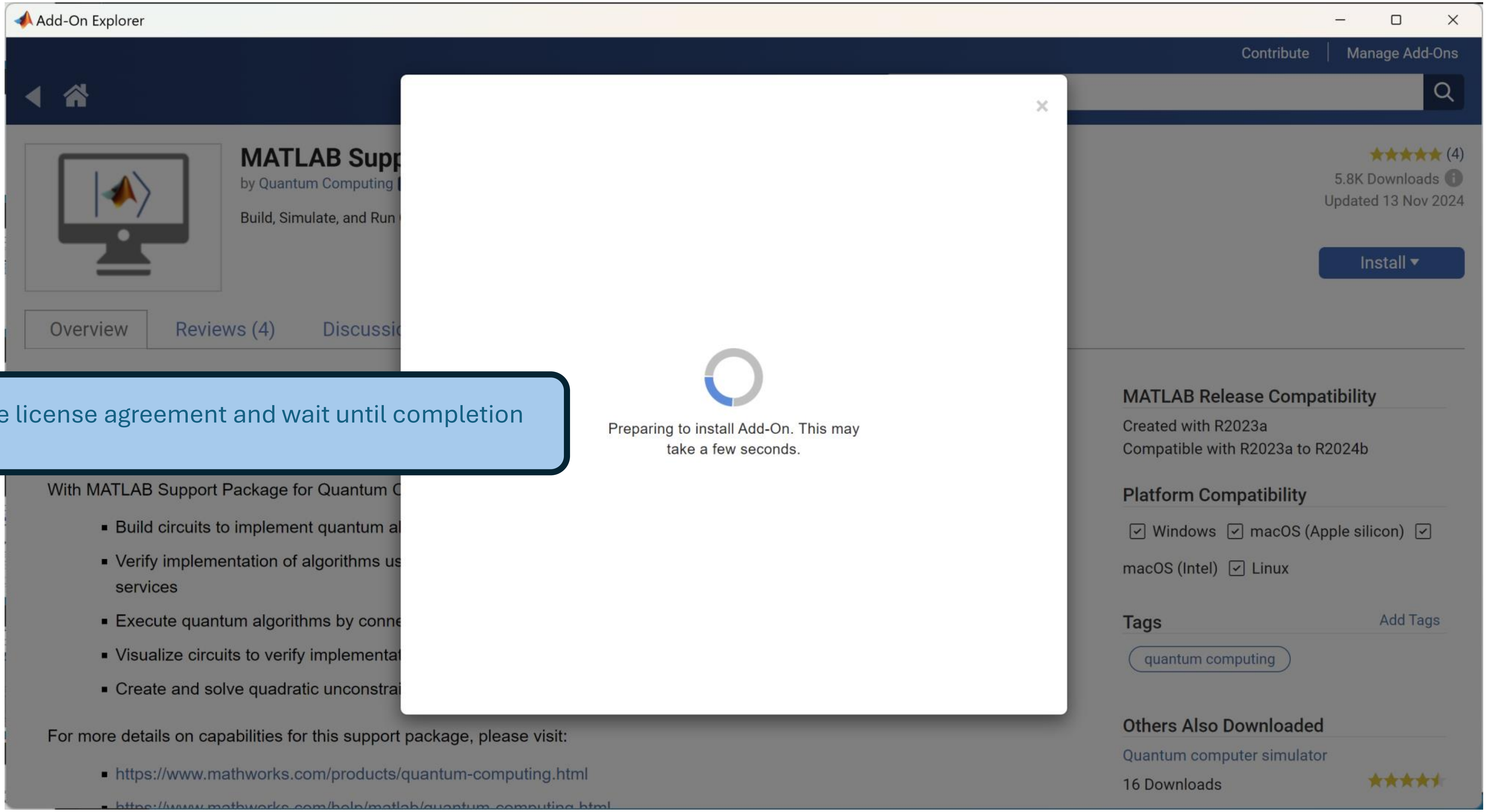


Type «Support Package for Quantum Computing»

Install MATLAB Support Package for QC



Install MATLAB Support Package for QC



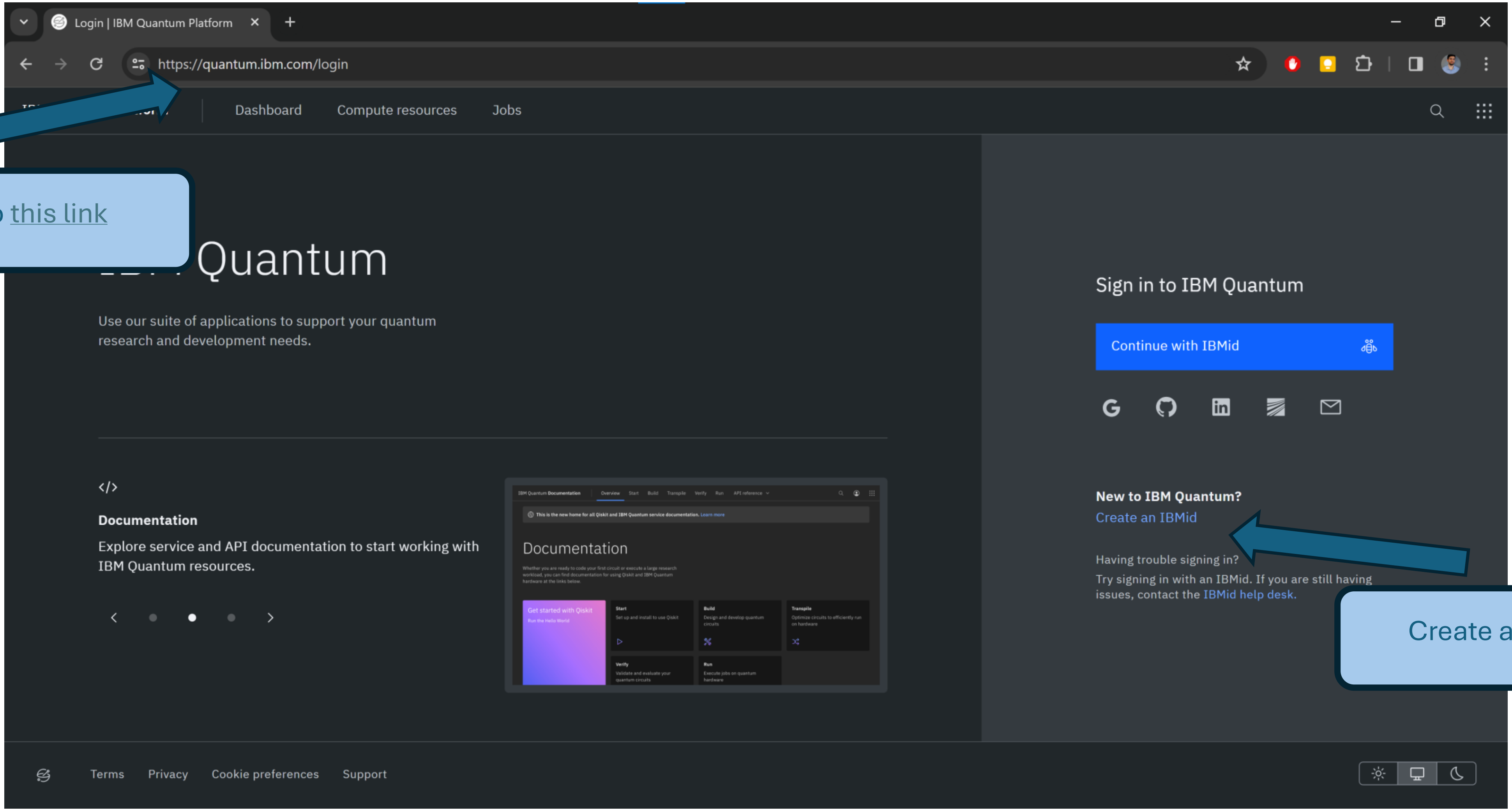
Accept the license agreement and wait until completion

MATLAB OnRamp

In case you don’t know how to use MATLAB, please follow this easy tutorial: link

You will find useful material and basics on programming with MATLAB.

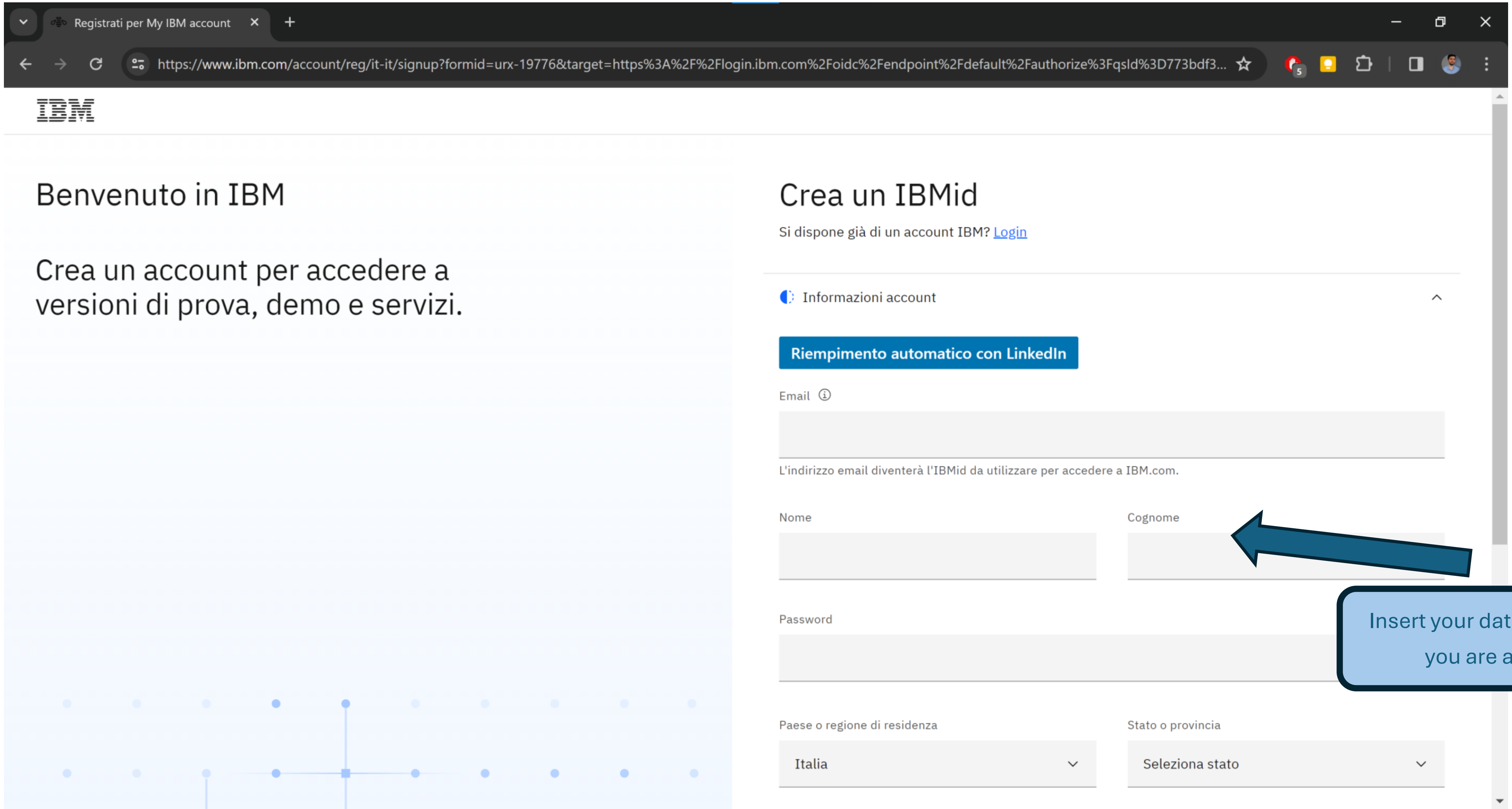
Sign up to IBM Quantum



Go to [this link](https://quantum.ibm.com/login)

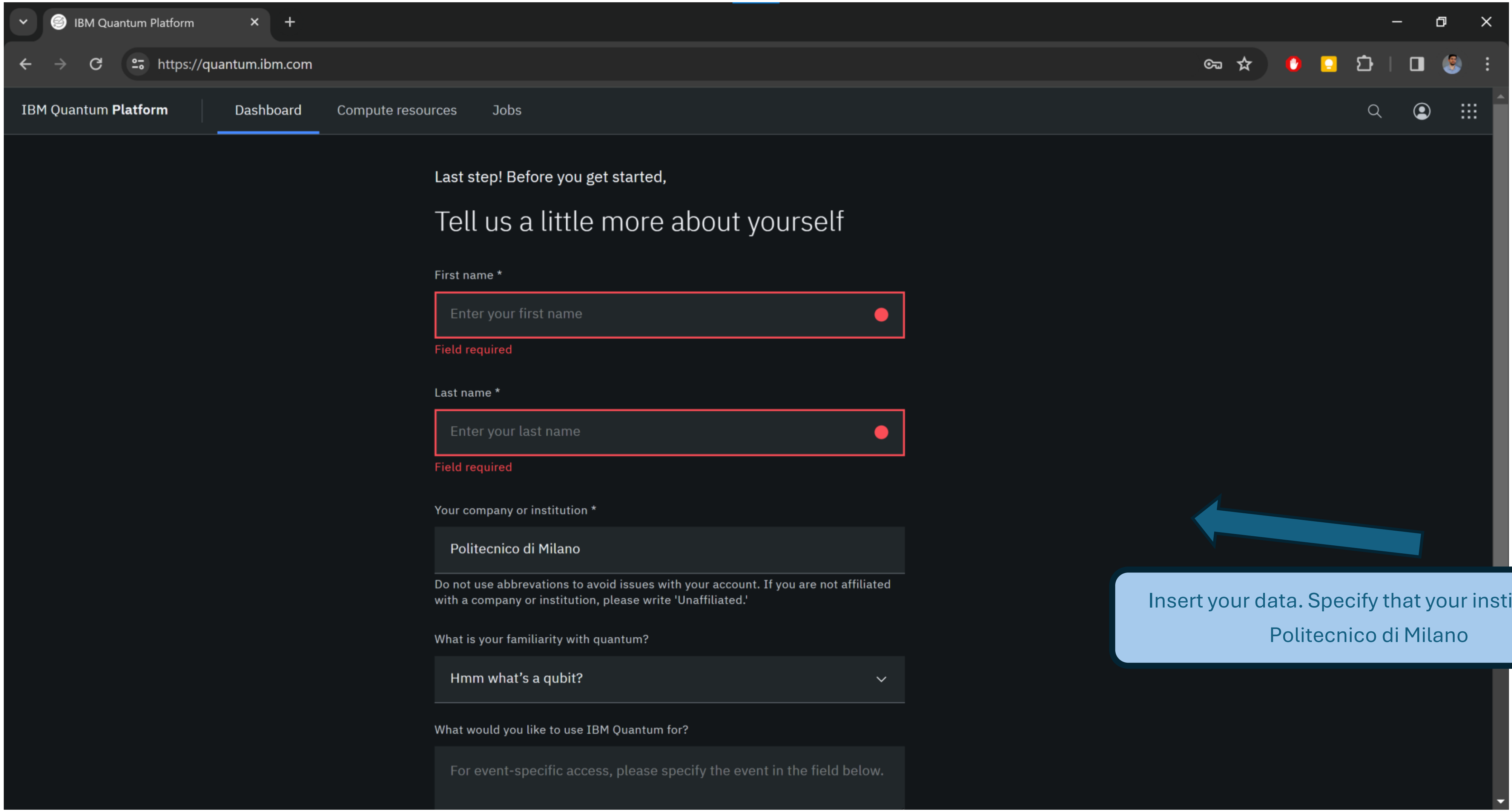
Create an IBMid

Sign up to IBM Quantum



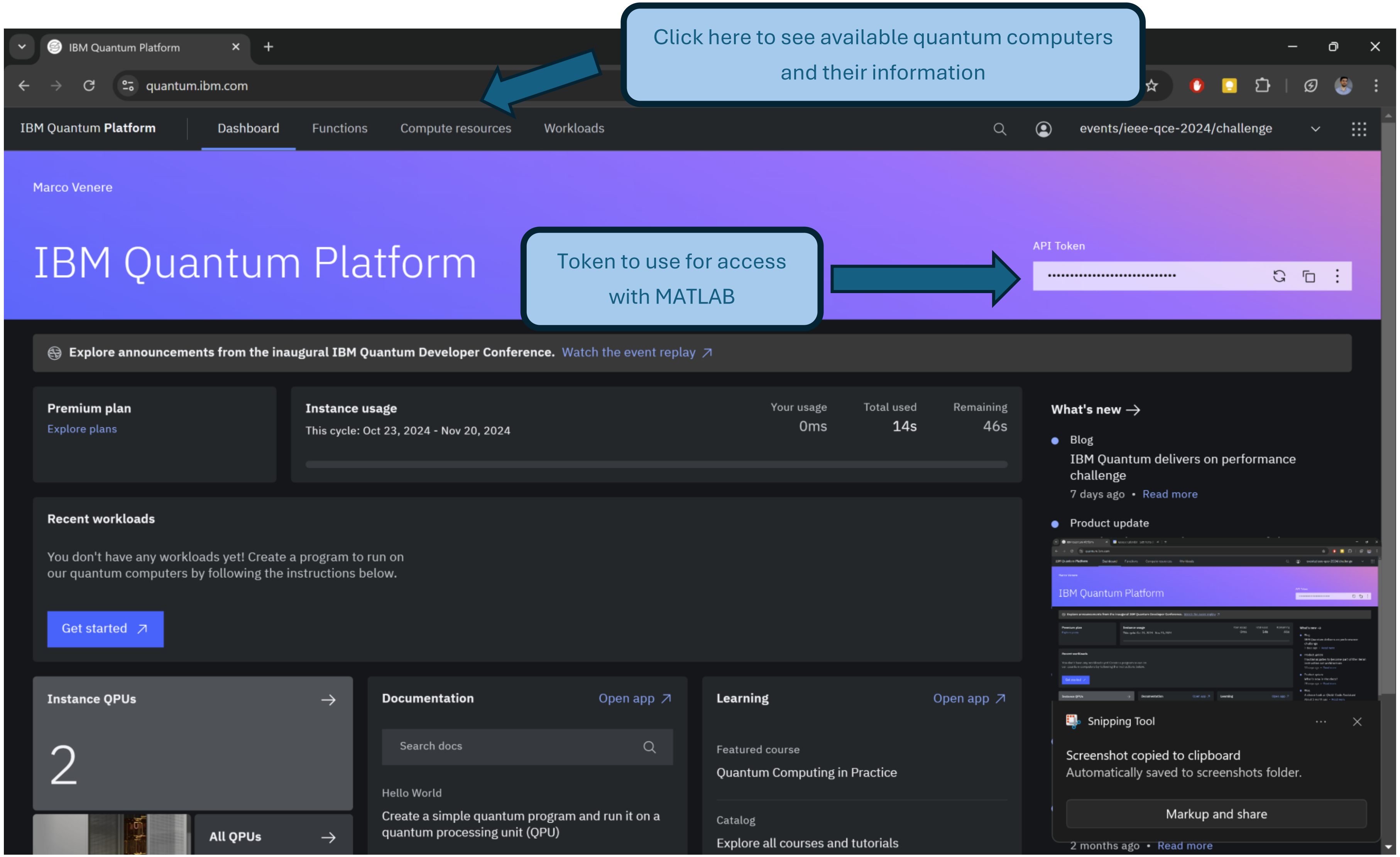
Insert your data. Specify that you are a student

Sign up to IBM Quantum

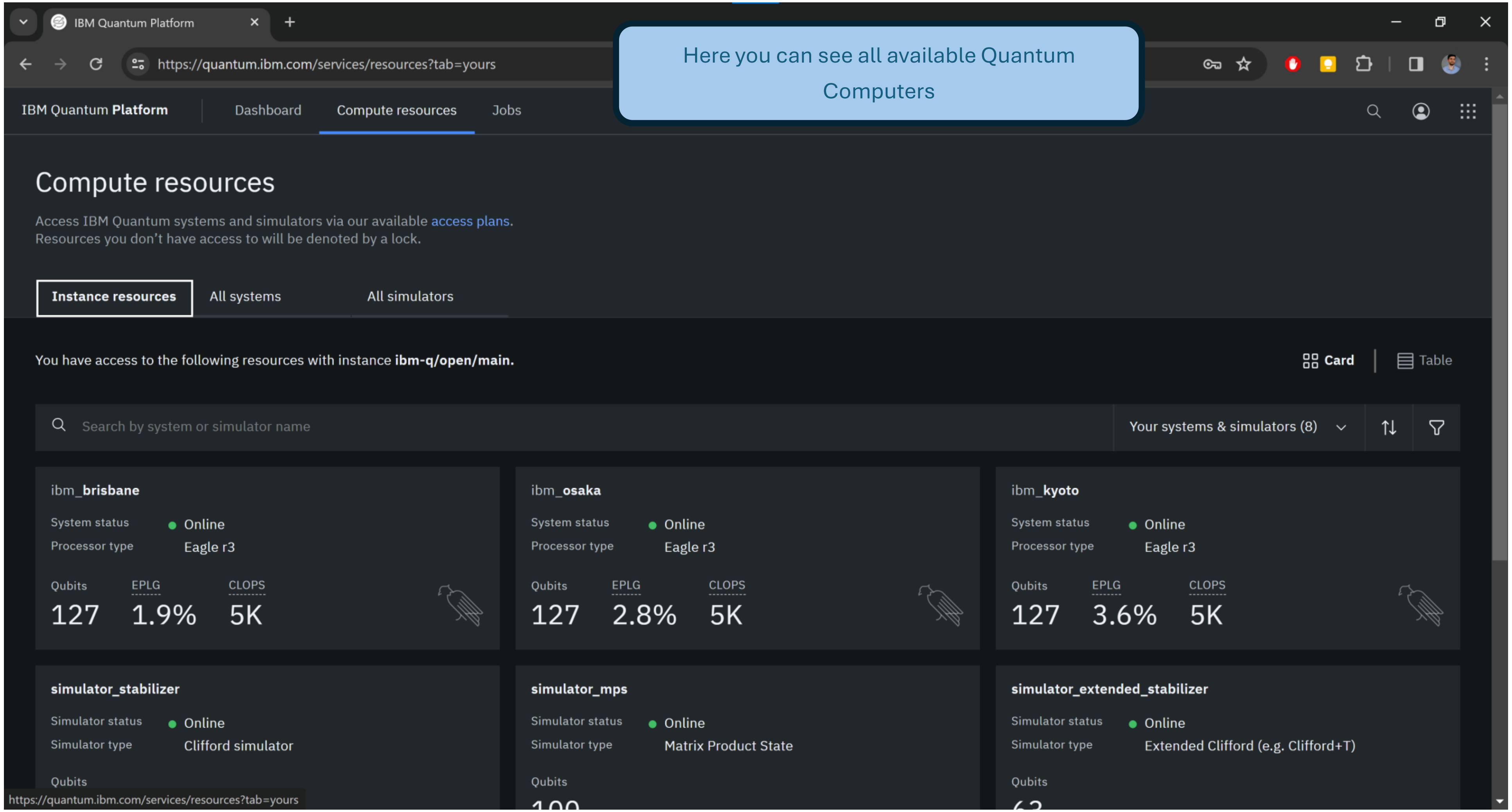


Insert your data. Specify that your institution is Politecnico di Milano

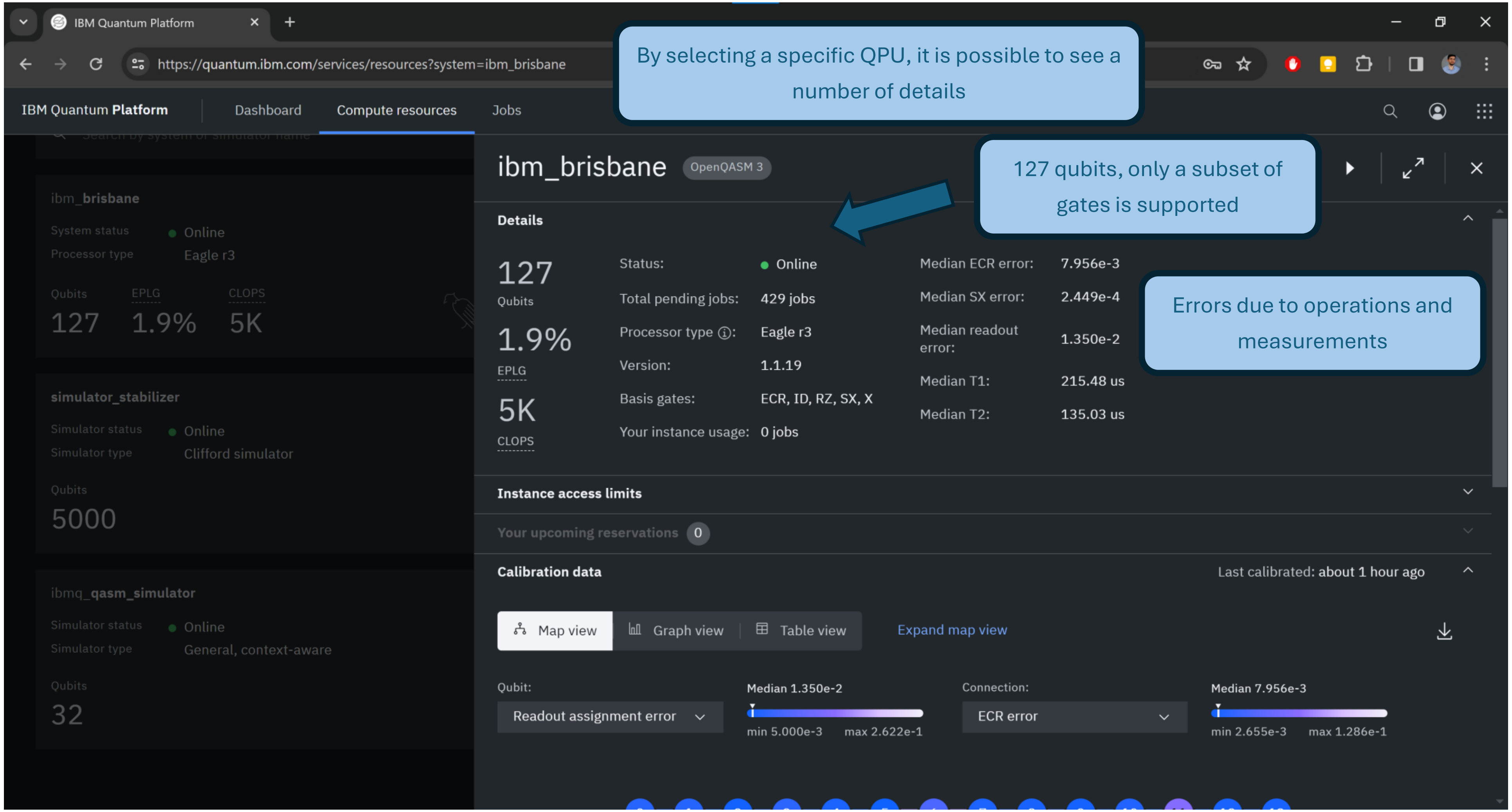
IBM Quantum Dashboard



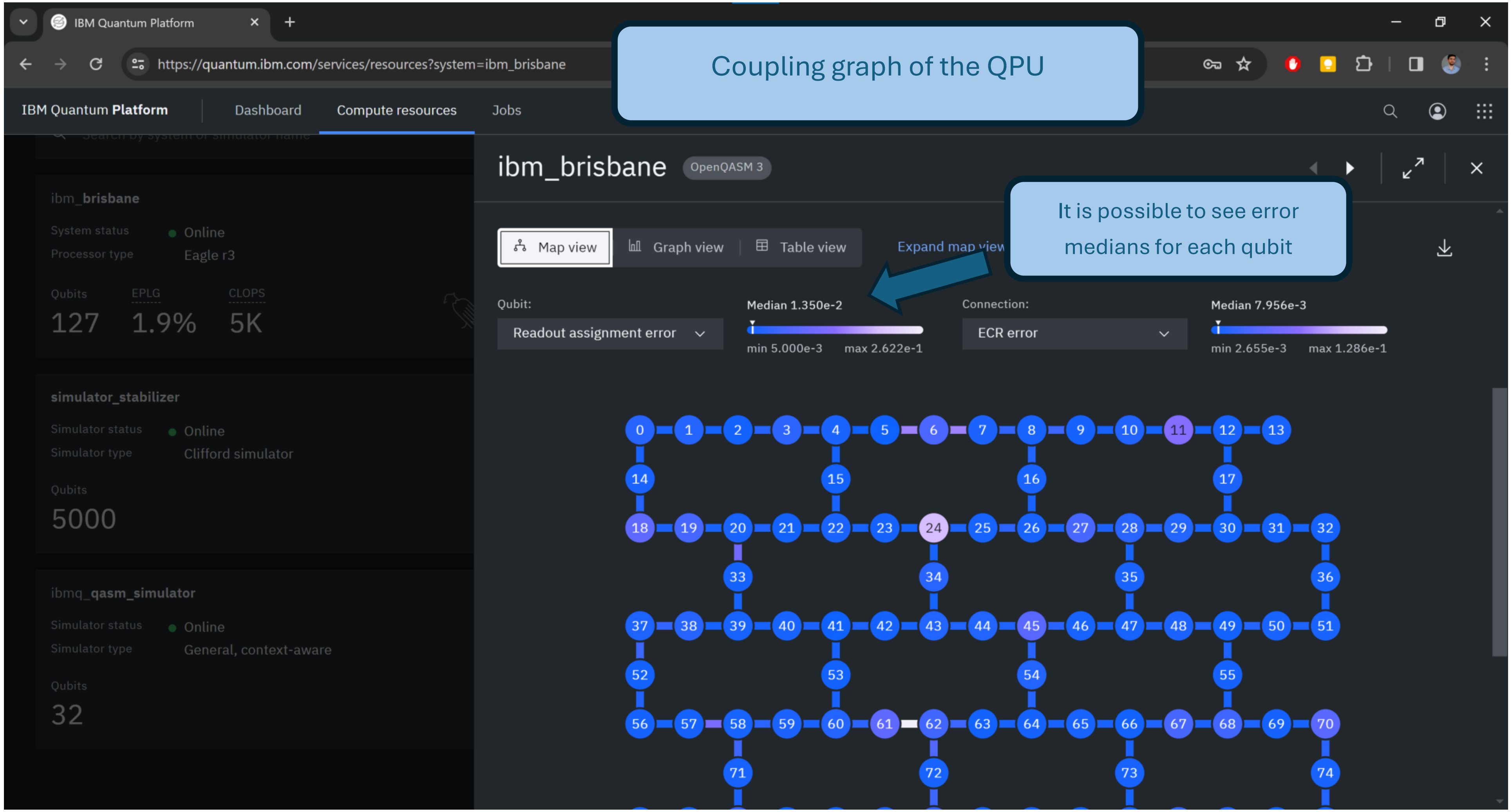
IBM Quantum Dashboard



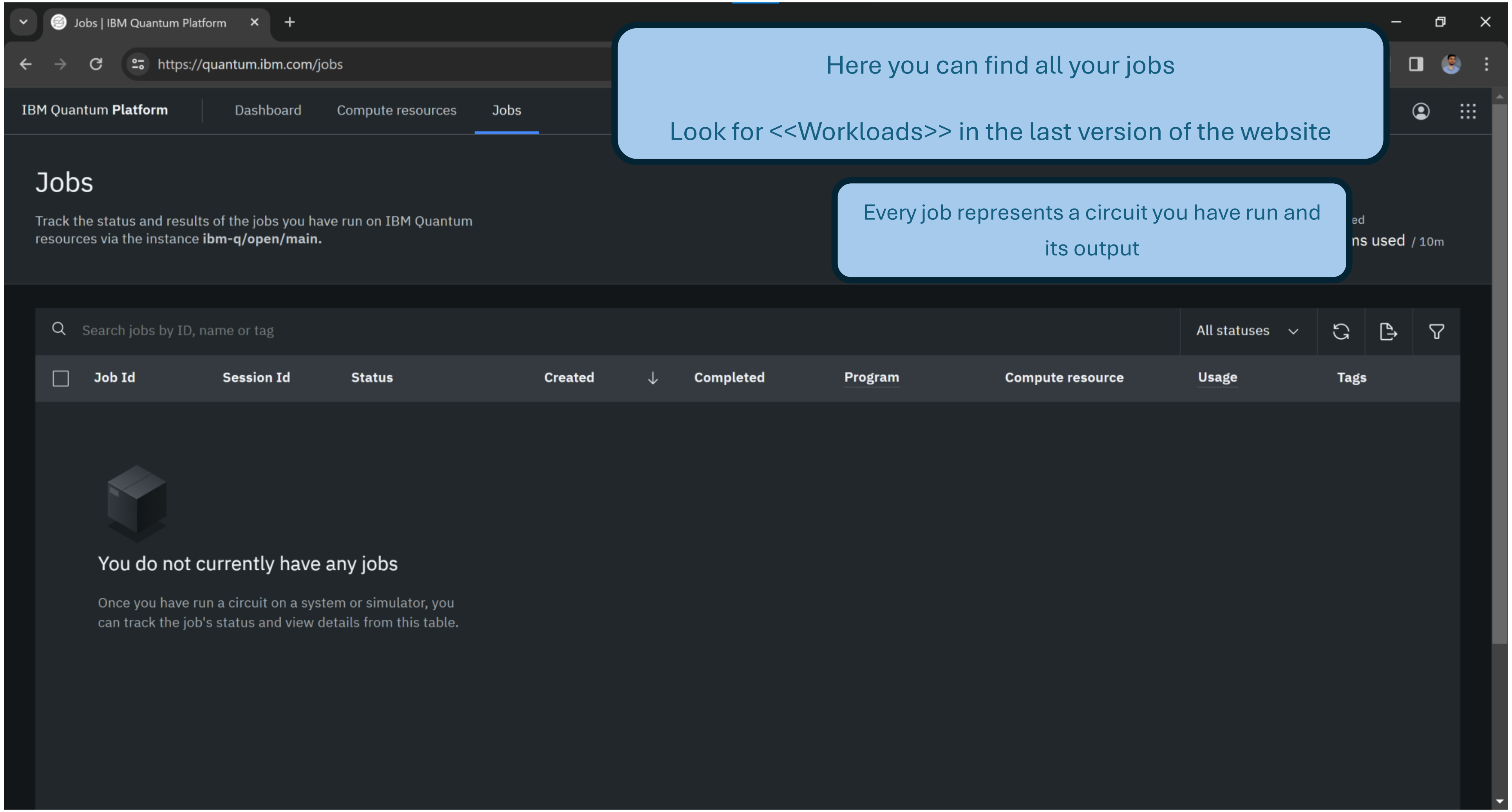
IBM Quantum Dashboard



IBM Quantum Dashboard



IBM Quantum Dashboard

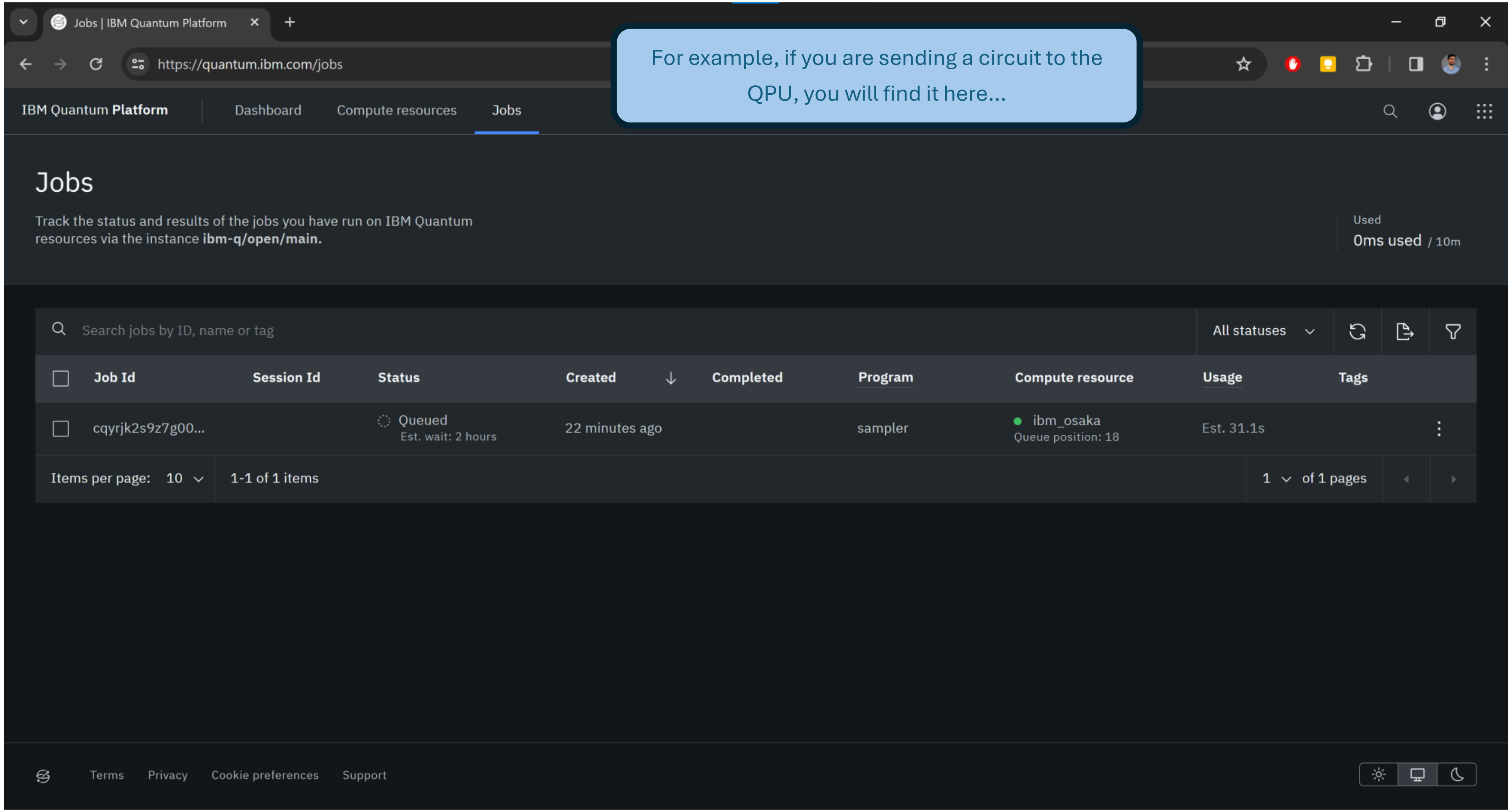


Here you can find all your jobs

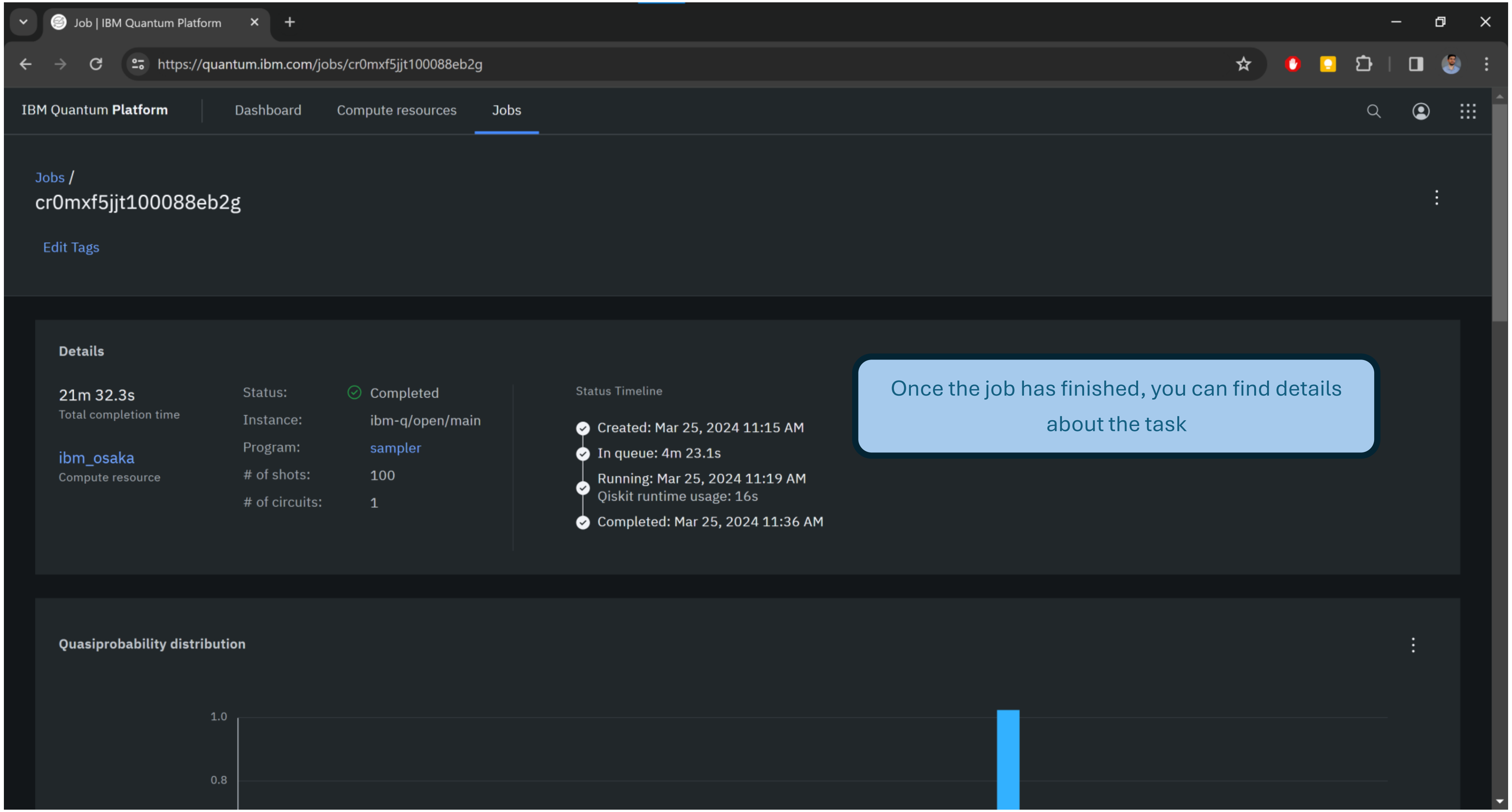
Look for <<Workloads>> in the last version of the website

Every job represents a circuit you have run and its output

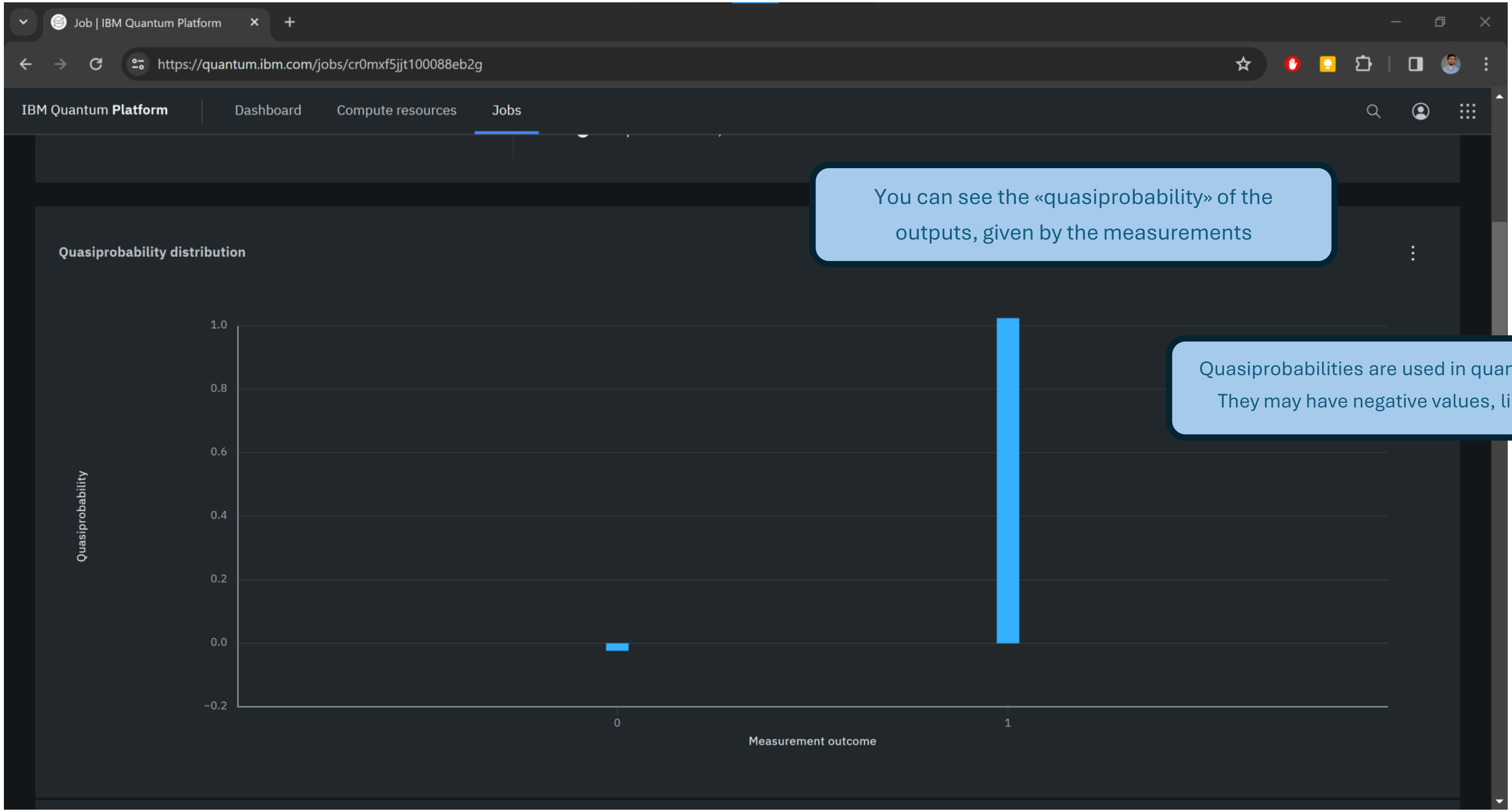
IBM Quantum Dashboard



IBM Quantum Dashboard



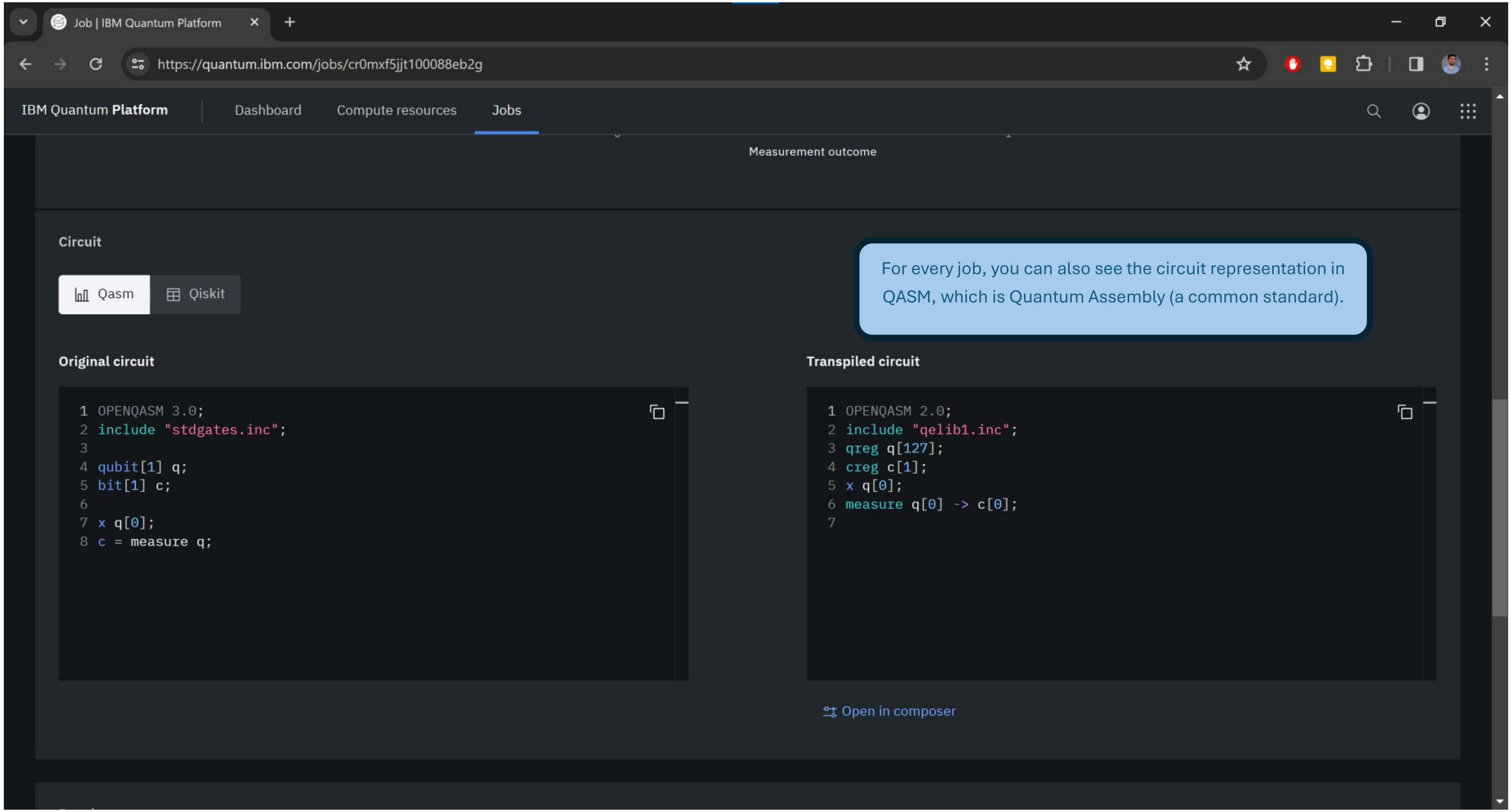
IBM Quantum Dashboard



You can see the «quasiprobability» of the outputs, given by the measurements

Quasiprobabilities are used in quantum mechanics. They may have negative values, like in this case.

IBM Quantum Dashboard



Now your setup should be working!

Next Steps:

1. Create our first Quantum Circuits
2. Simulate them classically
3. Run them on real quantum hardware

Create our first Quantum Circuits

We need to create a **quantumCircuit** object....

Docs here: quantumCircuit

Let's do it now with a LiveScript!

More Quantum Algorithms

We are also going to see some examples of famous quantum algorithms:

- Quantum Teleportation
- Quantum Fourier Transform

Quantum Teleportation

Recap: Alice and Bob share a pair of entangled qubits $|\Phi^+\rangle_{AB}$.

Alice also possesses a qubit $|\psi\rangle_{A'} = \alpha|0\rangle_{A'} + \beta|1\rangle_{A'}$:

$$|\psi\rangle_{A'}|\Phi^+\rangle_{AB} = \left(\alpha|0\rangle_{A'} + \beta|1\rangle_{A'}\right)\frac{|00\rangle_{AB}+|11\rangle_{AB}}{\sqrt{2}}$$

Alice measures her two qubits using the Bell states, and Bob's qubit becomes one of

$|\psi\rangle_B, Z|\psi\rangle_B, X|\psi\rangle_B, XZ|\psi\rangle_B$

Alice sends two classical bits and Bob reconstructs $|\psi\rangle_B$.

A whole qubit has been teleported using a pair of entangled qubits and two classical bits.

Quantum Teleportation

Let's implement it on MATLAB!

We need:

- A circuit with 3 qubits
- An initialization with entanglement + generic qubit $|\psi\rangle$
- A Bell measurement for Alice's qubits
- The application of Z and X gates conditioned by Alice measurement

Quantum Fourier Transform

The Quantum Fourier Transform is the quantum analogue of the Discrete Fourier Transform (DFT).

It's very useful for a number of quantum algorithms, e.g., Shor's algorithm for integer factorization, discrete logarithm, quantum phase estimation, and algorithms for the hidden subgroup problem.

There is a computational advantage in computing the QFT: indeed, we can apply a DFT on 2^n amplitudes by using only $O(n^2)$ Hadamard gates and controlled phase shift gates. The classical approach would instead require $O(n2^n)$ operations.

Quantum Fourier Transform

Classical DFT: it maps a vector $(x_0, x_1, \dots, x_{N-1}) \in \mathbb{C}^N$ to another vector $(y_0, y_1, \dots, y_{N-1}) \in \mathbb{C}^N$

$$y_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n \omega_N^{-nk}, \omega_N = e^{\frac{2\pi i}{N}}, k = 0, 1, 2, \dots, N-1$$

QFT: it maps a quantum state $|x\rangle = \sum_{i=0}^{N-1} x_i |i\rangle$ to another quantum state $|y\rangle = \sum_{i=0}^{N-1} y_i |i\rangle$, where $N = 2^n$, which means that the state is spread across n different qubits.

$$y_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n \omega_N^{nk}, \omega_N = e^{\frac{2\pi i}{N}}, k = 0, 1, 2, \dots, N-1$$

The sign of the exponential varies based on different conventions. ω_N^{nk} represents a rotation.

Quantum Fourier Transform

For example, if $|x\rangle$ is a basis state, we can define the whole QFT operation as:

$$QFT: |x\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{xk} |k\rangle$$

where xk is the scalar product between the bitstring x and the bitstring k.

E.g., $x = 11001$ and $\mathrm{k} = 10011 \rightarrow xk = 1 \cdot 1 + 1 \cdot 0 + 0 \cdot 0 + 0 \cdot 1 + 1 \cdot 1$

Inverse Quantum Fourier Transform

The inverse of the QFT is also defined:

$$x_k = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} y_k \omega_N^{-nk}, \qquad n = 0, 1, 2, \dots, N-1$$

Notice the difference in the sign of the phase.

Practical Implementation of QFT

Given n qubits, whose state is given by a vector of 2^n components, the computation of the QFT can be given by using the following $2^n \times 2^n$ matrix:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \dots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \dots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \dots & \omega^{(N-1)(N-1)} \end{pmatrix}$$

On a practical level, this circuit can be implemented by using a number of Hadamard gates and controlled rotation gates. Let's look at it on MATLAB…

Thank you for your attention!

Quantum Computing
A Practical Perspective

Marco Venere

marco.venere@polimi.it

MathWorks®

November 20th, 2024
Politecnico di Milano

POLITECNICO MILANO 1863
NECST laboratory