Tutorial - Quantum Web Services: development and deployment

Introduction

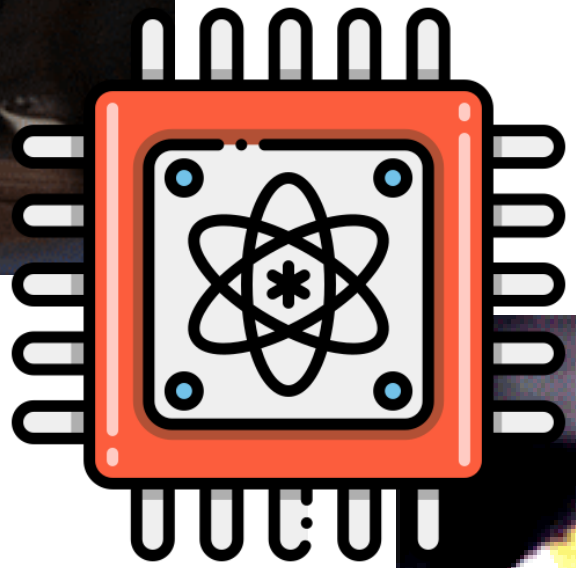
Javier Romero-Álvarez
Jaime Alvarado-Valiente
Enrique Moguel
José García-Alonso

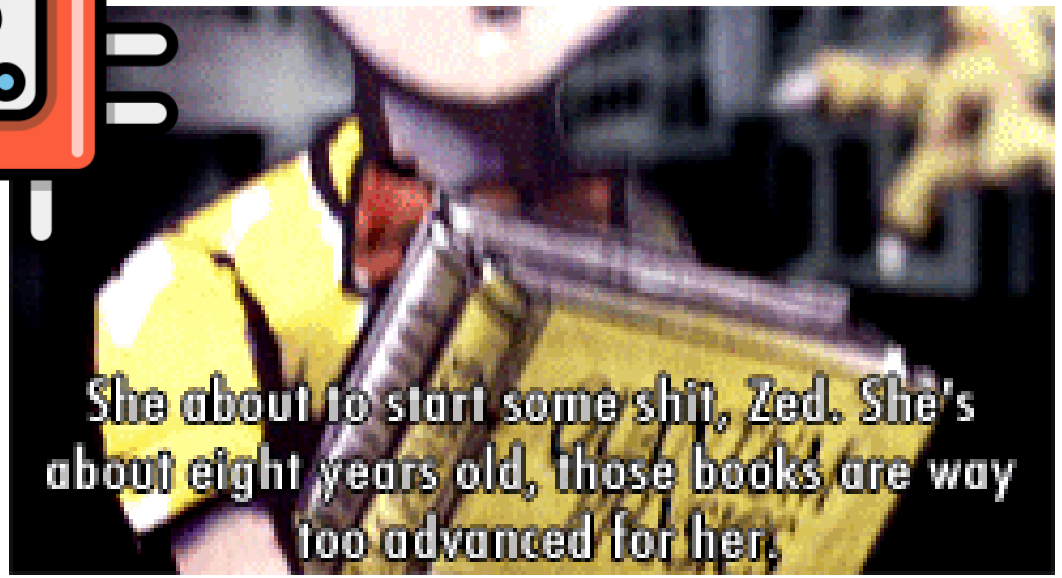
MINISTERIO DE CIENCIA E INNOVACIÓN
UNIÓN EUROPEA
FONDO EUROPEO DE DESARROLLO REGIONAL
AGENCIA ESTATAL DE INVESTIGACIÓN
JUNTA DE EXTREMADURA
Consejería de Economía, Ciencia y Agenda Digital
SPILab by Quercus SEG
Software Engineering Group
QUERCUS
UNIVERSIDAD DE EXTREMADURA

OUTLINE

1. Introduction to quantum computing
2. Quantum Programming
3. Quantum Web Services with AWS
4. Shor Service

Introduction to quantum computing



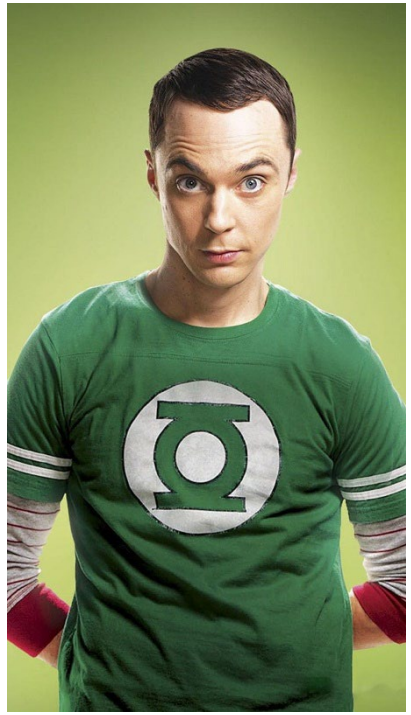


Introduction to quantum computing

1900s

- Einstein
- Planck
- Bohr
- Feynman
- Schrödinger
- …

Introduction to quantum computing



Theoretical physics → Experimental physics → Engineering

Introduction to quantum computing

- 1998 – Los Alamos National Laboratory – Massachussetts Institute of Technology
- First quantum computer

  2 qubits
- Nuclear Magnetic Resonance

IBM

Google

rigetti

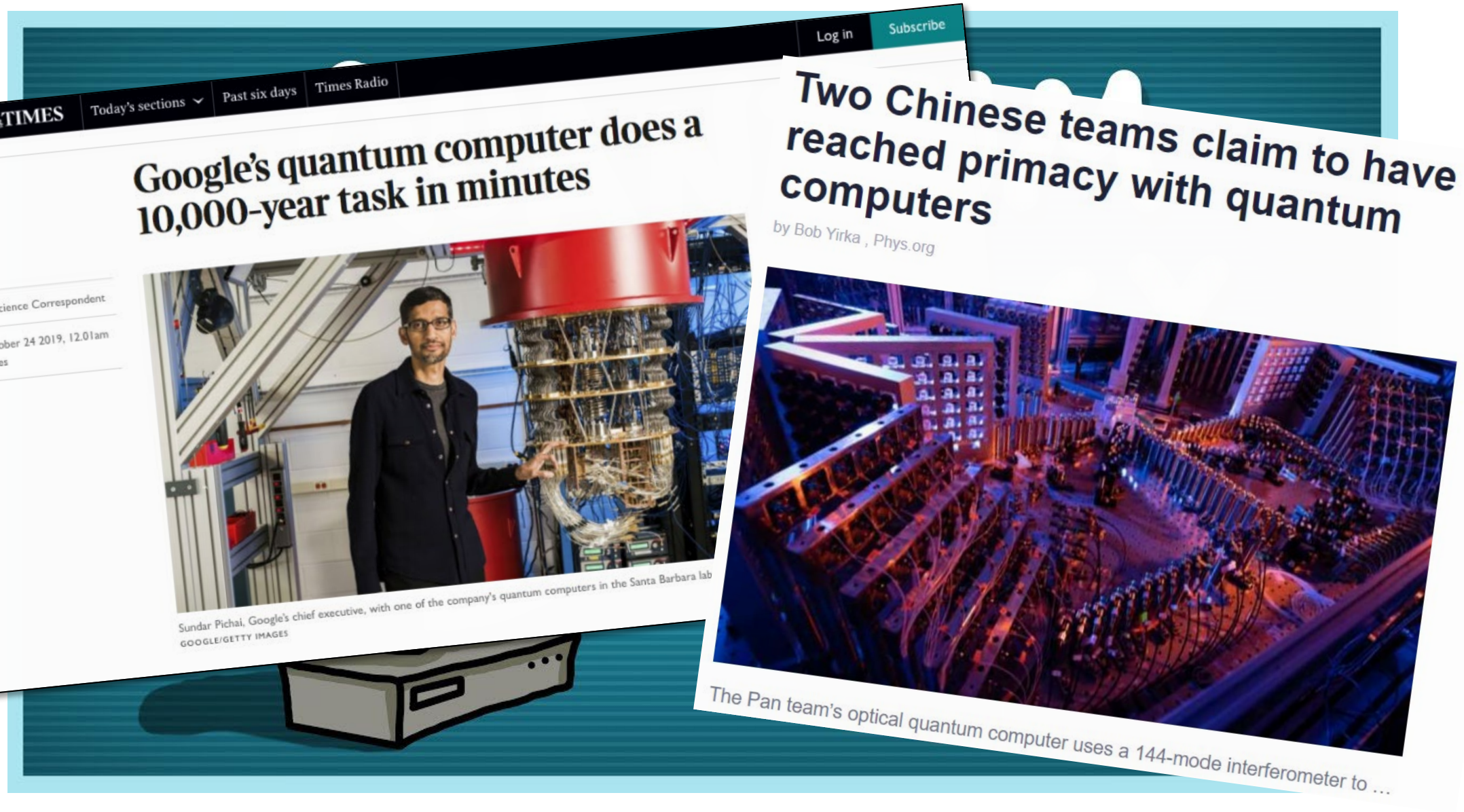
XANADU

IONQ

D:wave
The Quantum Computing Company™

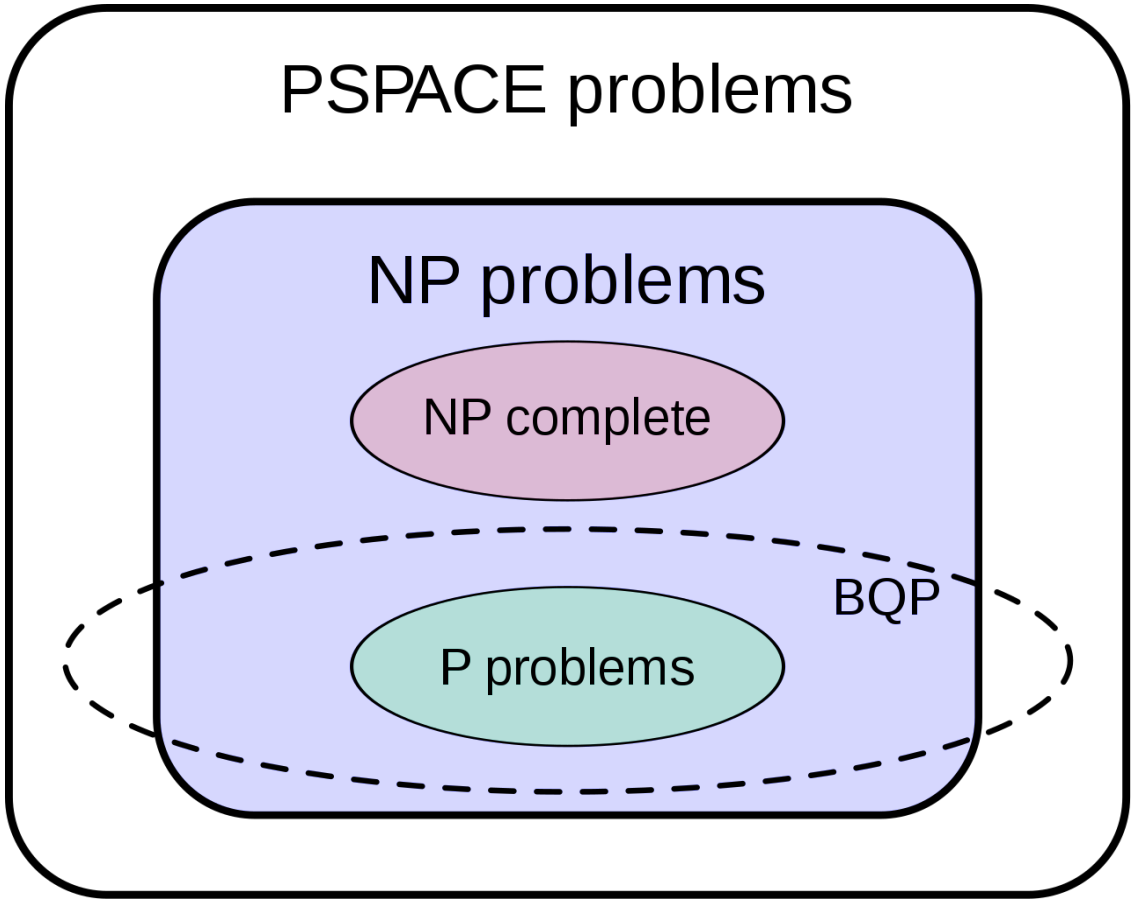

CénitS
LUSITANIA III

Quantum Supremacy

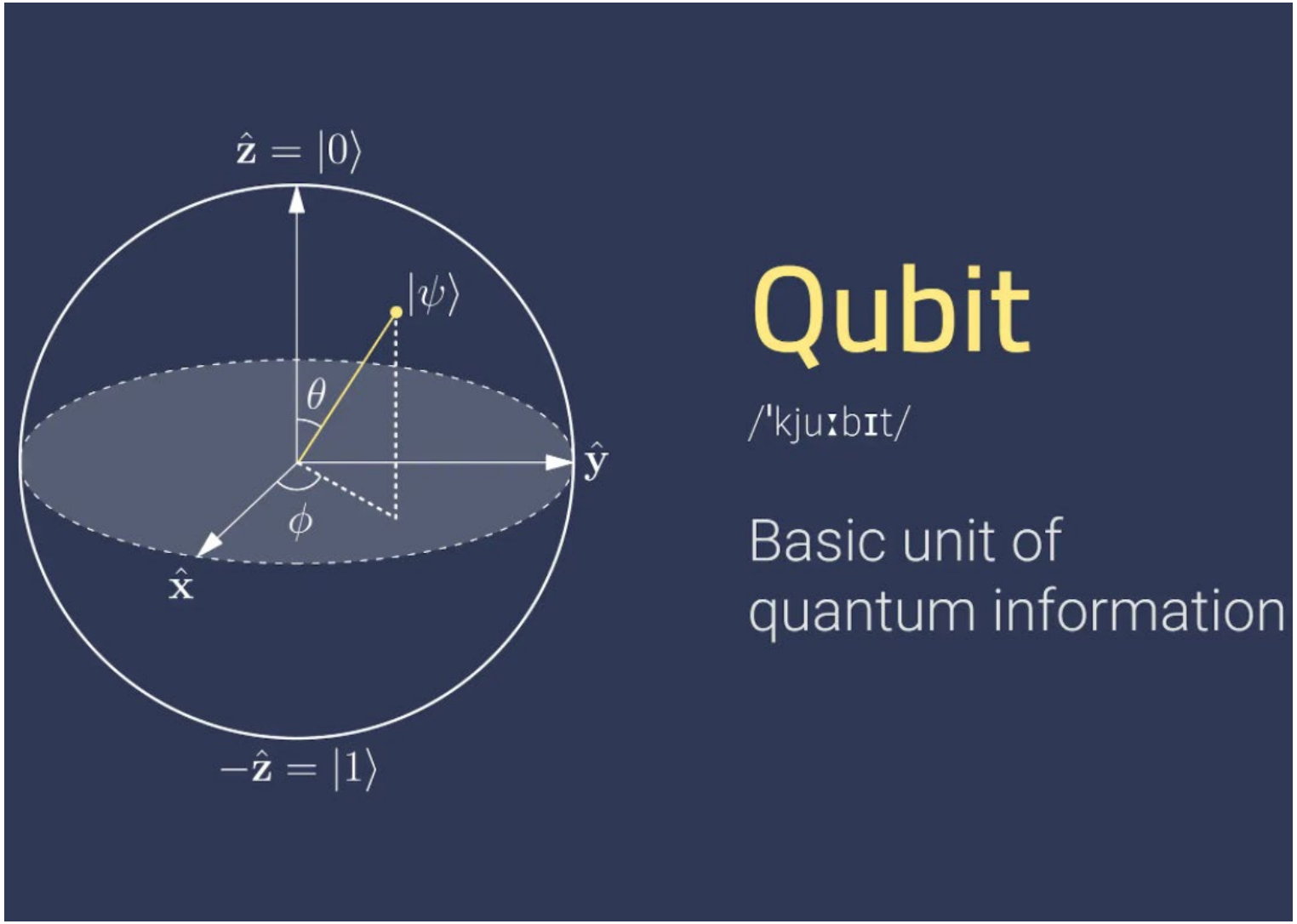


Computational complexity theory

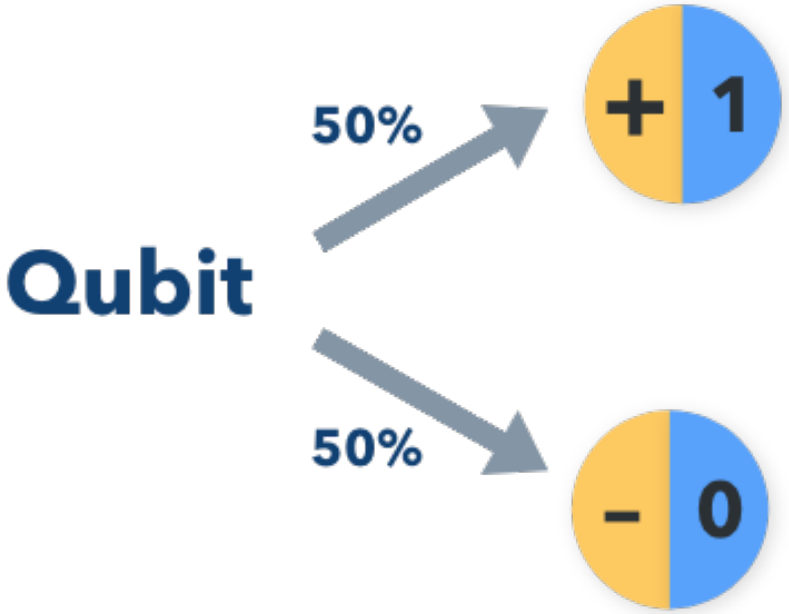
Bounded-error Quantum Polynomial time (BQP) is the class of decision problems solvable by a quantum computer in polynomial time, with an error probability of at most 1/3 for all instances



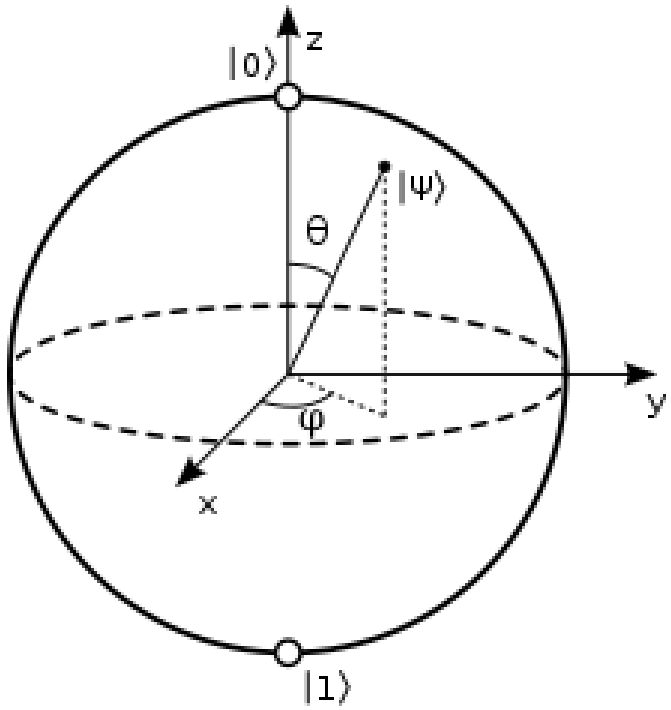
Qubit


$\hat{\mathbf{z}} = |0\rangle$
$|\psi\rangle$
θ
$\hat{\mathbf{y}}$
ϕ
$\hat{\mathbf{x}}$
$-\hat{\mathbf{z}} = |1\rangle$
Qubit
/'kju:bɪt/
Basic unit of
quantum information

Qubit collapse



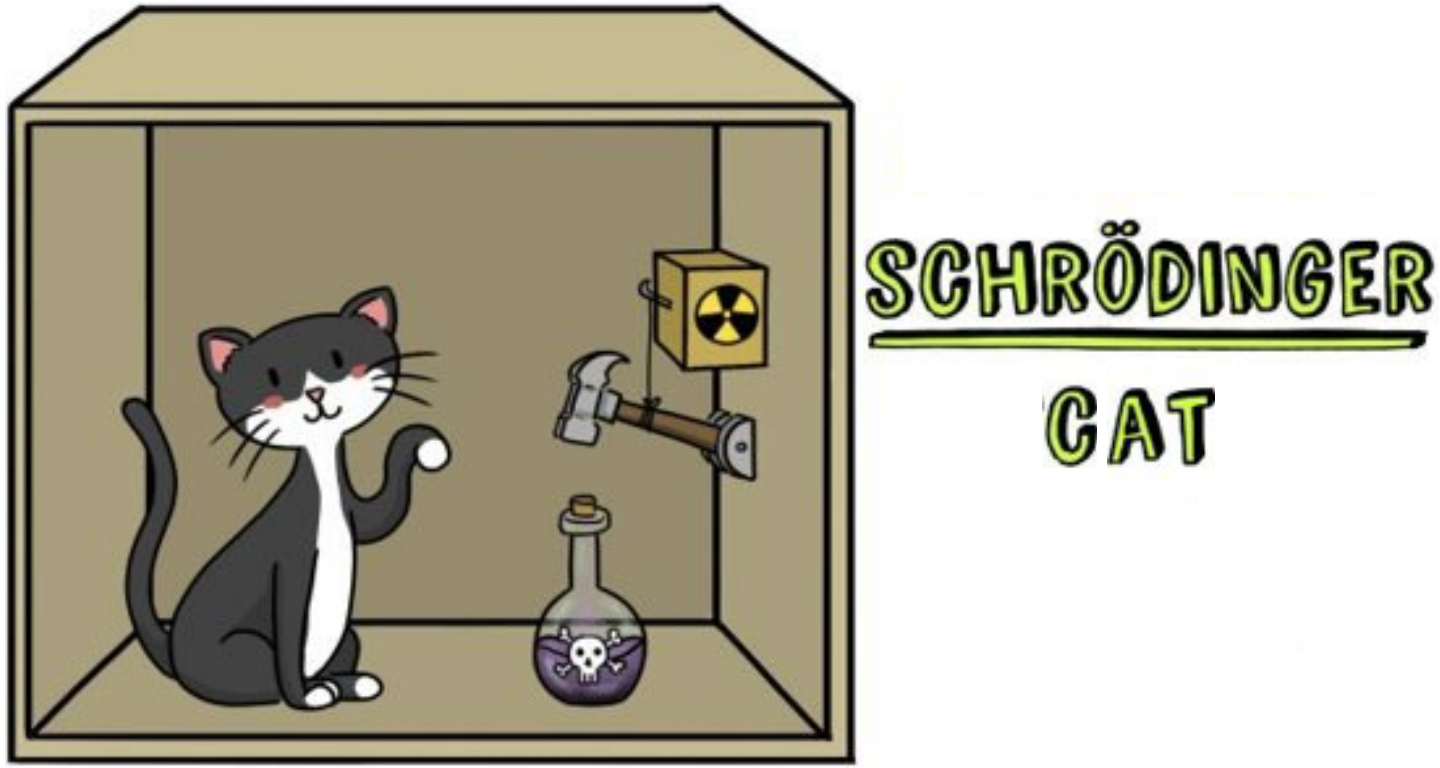
Qubit representation



$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \text{ and } |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Qubit superposition

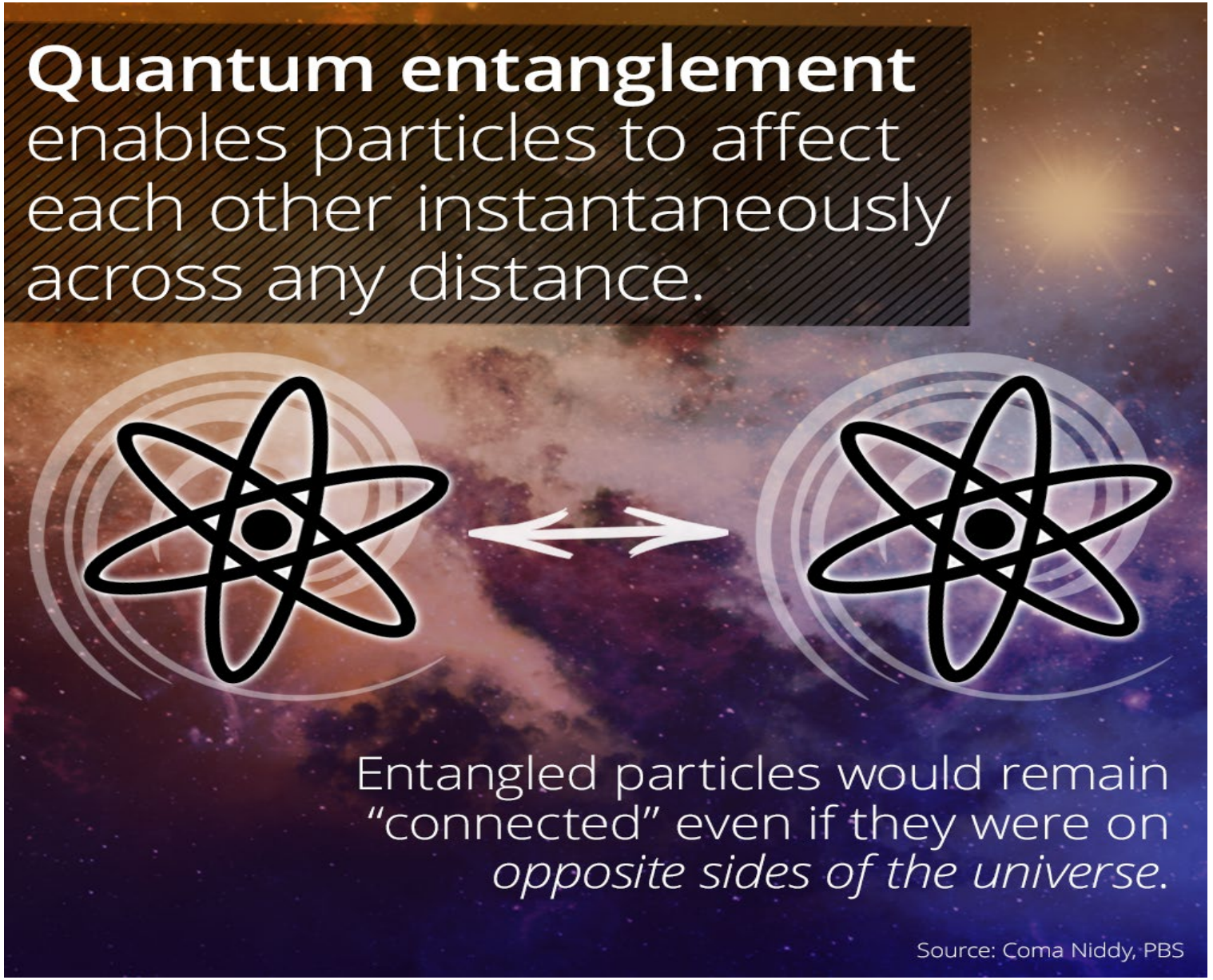


Madrid's quantum communication network deploys its first phase

Researchers at the Center for Computational Simulation at the Polytechnic University of Madrid are coordinating the Madrid quantum communication network, a European benchmark. They are researching quantum algorithms that will increase the security of critical applications in fields such as telecommunications, health care, electricity supply and public services, among others.

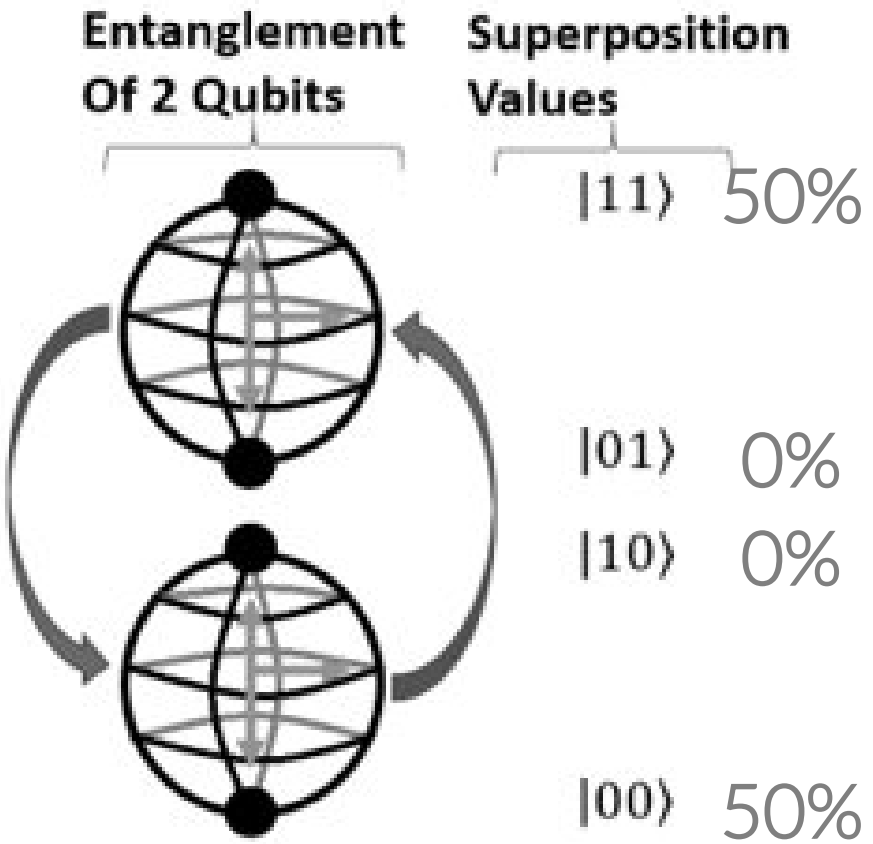
22.03.2021

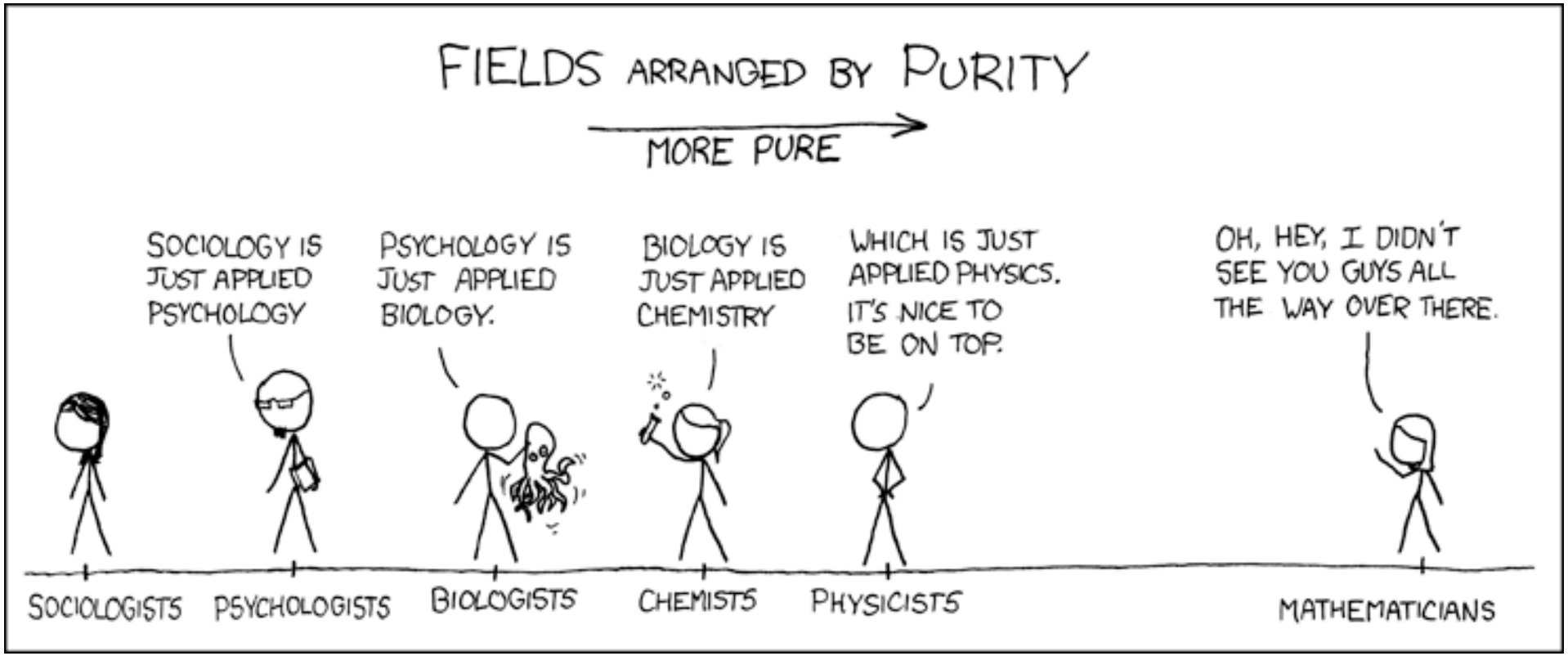
Qubit entanglement



Qubit entanglement

- Quantum entanglement is the fundamental resource of Quantum Physics, which is necessary to generate, in order to achieve the Maximum Effect in Quantum Computing
- **It cannot occur in classical bits**
- The quantum state of (n) qubits involves the superposition of all possible configurations




FIELDS ARRANGED BY PURITY
MORE PURE
SOCIOLOGY IS JUST APPLIED PSYCHOLOGY
PSYCHOLOGY IS JUST APPLIED BIOLOGY.
BIOLOGY IS JUST APPLIED CHEMISTRY
WHICH IS JUST APPLIED PHYSICS. IT'S NICE TO BE ON TOP.
OH, HEY, I DIDN'T SEE YOU GUYS ALL THE WAY OVER THERE.
SOCIOLOGISTS
PSYCHOLOGISTS
BIOLOGISTS
CHEMISTS
PHYSICISTS
MATHEMATICIANS

Quantum software

2 Algorithms based on the quantum Fourier transform
- 2.1 Deutsch–Jozsa algorithm
- 2.2 Bernstein–Vazirani algorithm
- 2.3 Simon's algorithm
- 2.4 Quantum phase estimation algorithm
- 2.5 Shor's algorithm
- 2.6 Hidden subgroup problem
- 2.7 Boson sampling problem
- 2.8 Estimating Gauss sums
- 2.9 Fourier fishing and Fourier checking

3 Algorithms based on amplitude amplification
- 3.1 Grover's algorithm
- 3.2 Quantum counting

4 Algorithms based on quantum walks
- 4.1 Element distinctness problem
- 4.2 Triangle-finding problem
- 4.3 Formula evaluation
- 4.4 Group commutativity

5 BQP-complete problems
- 5.1 Computing knot invariants
- 5.2 Quantum simulation
- 5.3 Solving a linear systems of equations

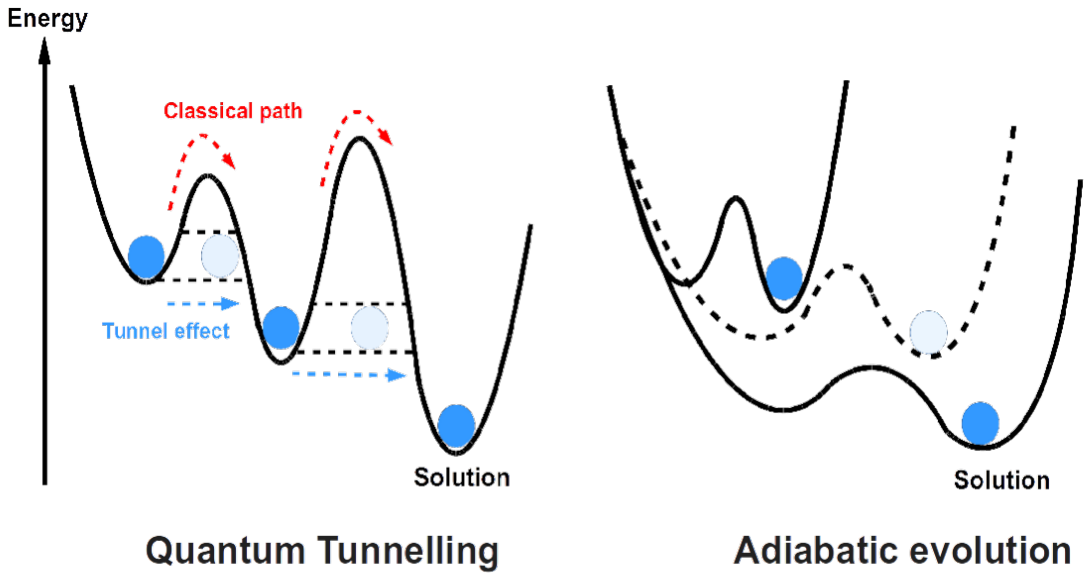
6 Hybrid quantum/classical algorithms
- 6.1 QAOA
- 6.2 Variational quantum eigensolver

QUANTUM PROGRAMMING

Types of programming

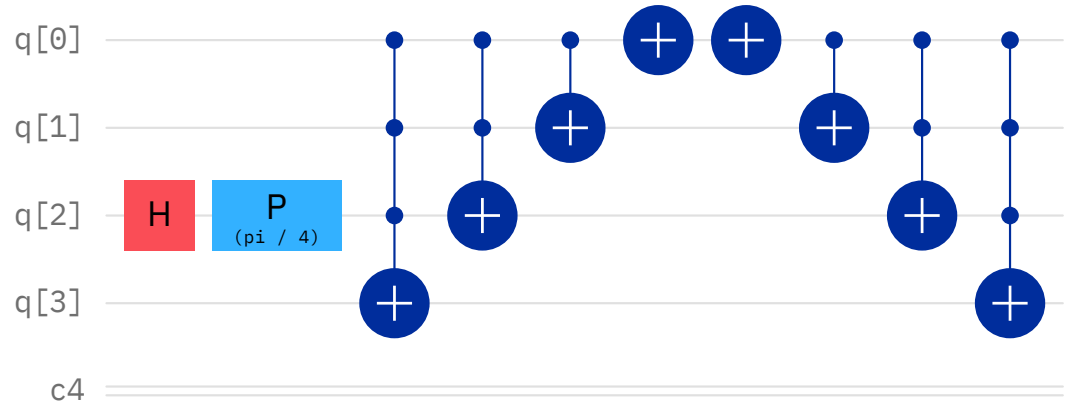
Quantum annealing (which also includes adiabatic quantum computation) is a quantum computing method used to find the optimal solution of problems involving many solutions

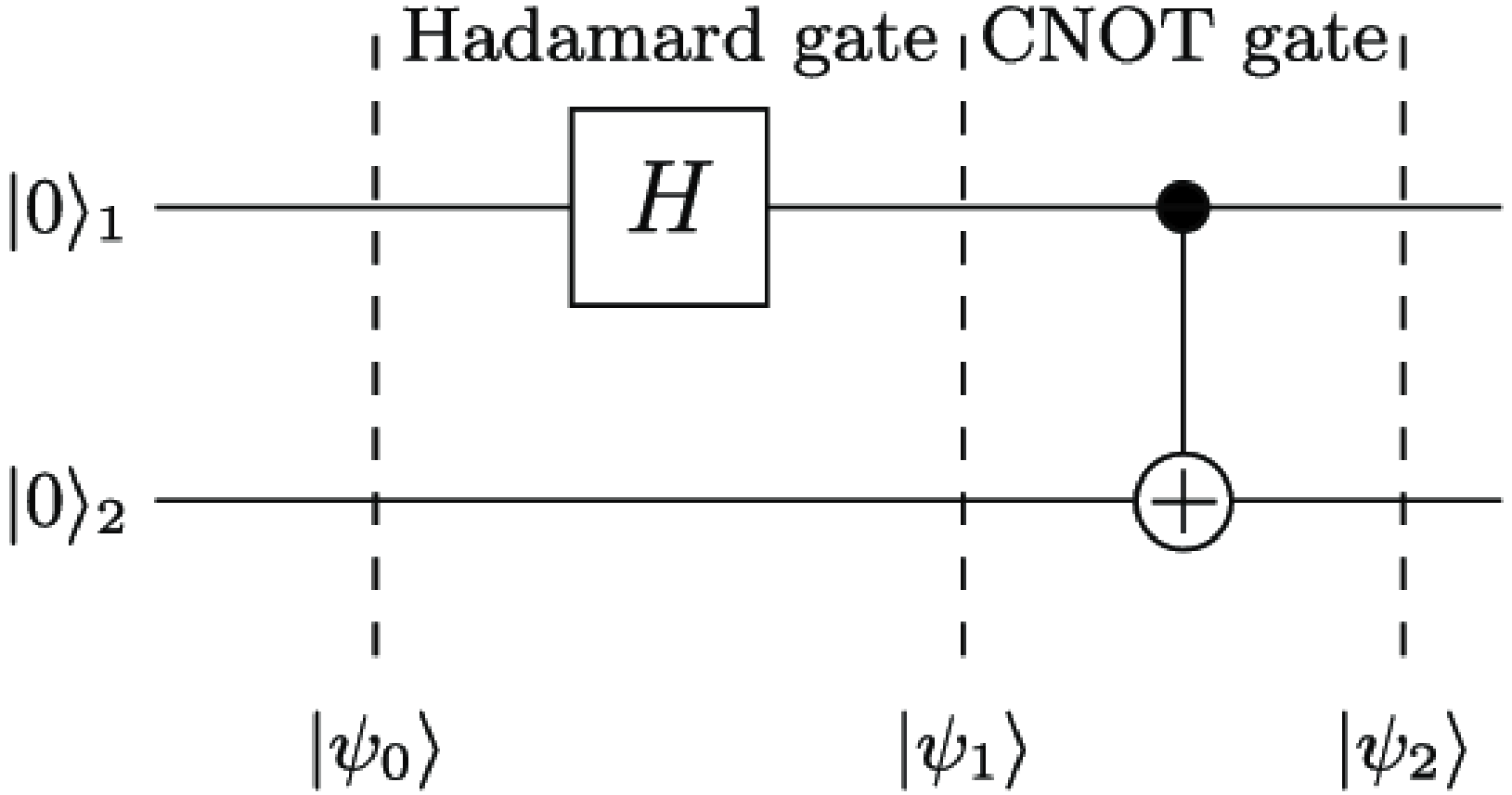
Universal quantum gate model is based on creating quantum structures using stable qubits and solving today's problems with **quantum circuits**



Quantum Tunnelling

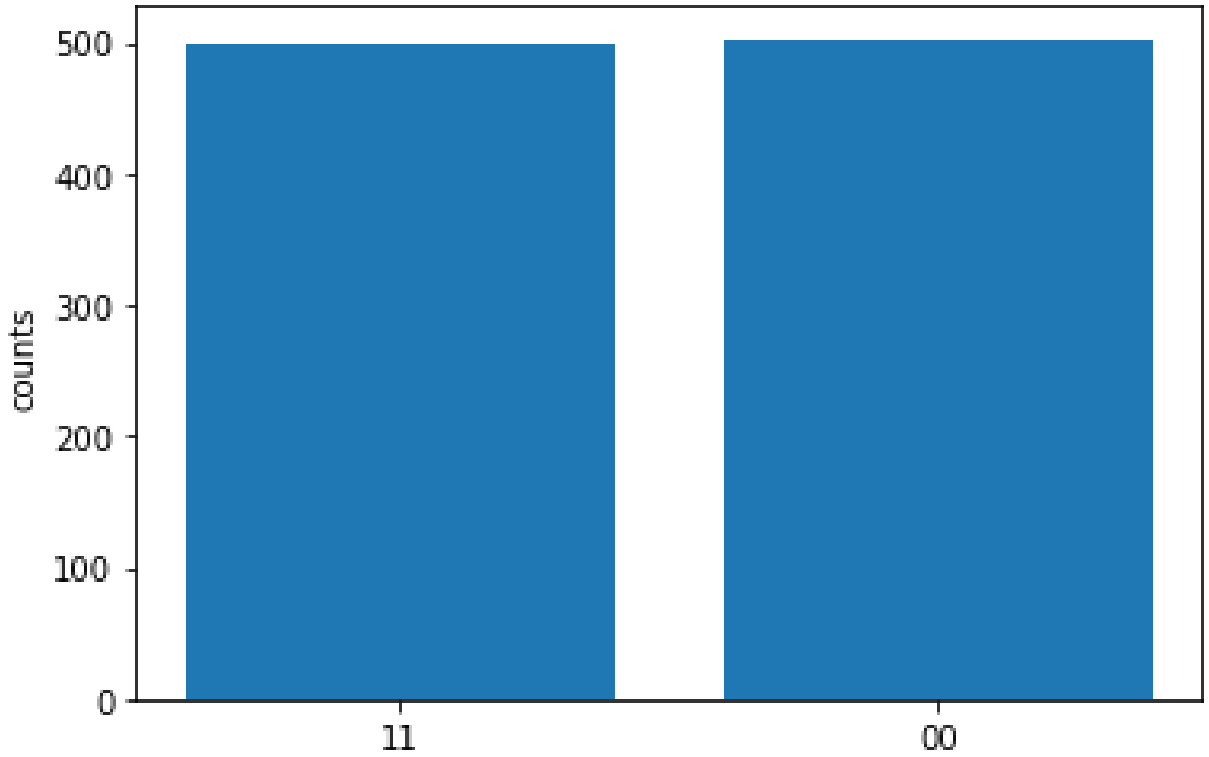
Adiabatic evolution



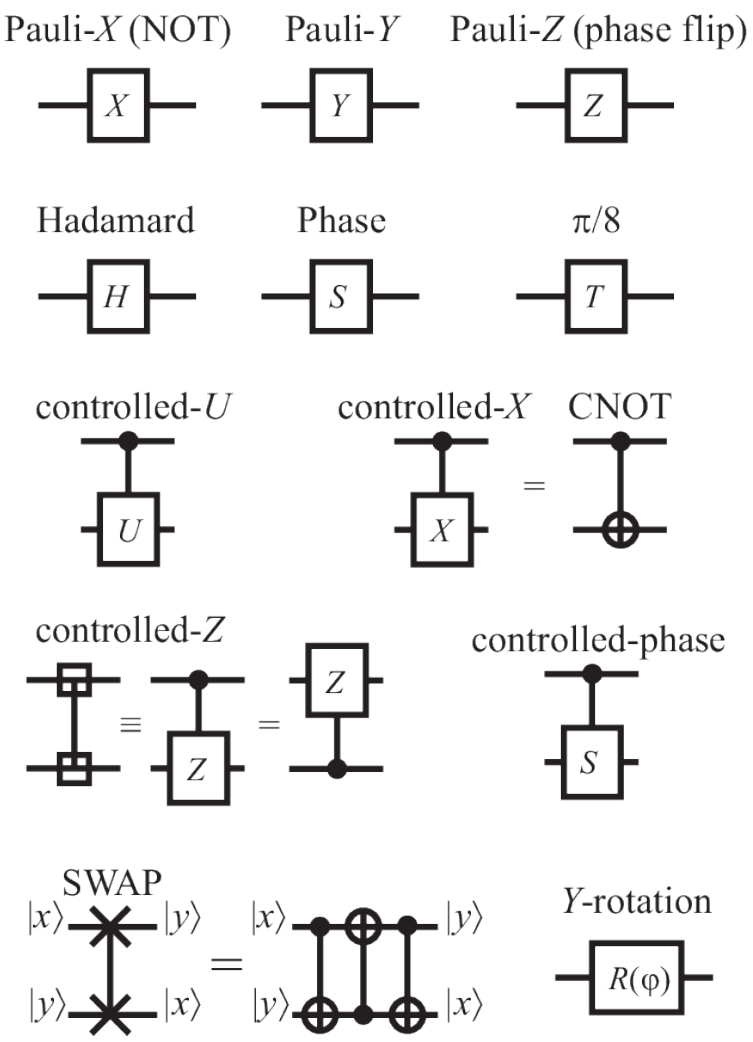

Hadamard gate
CNOT gate
$|0\rangle_1$
$|0\rangle_2$
H
$|\psi_0\rangle$
$|\psi_1\rangle$
$|\psi_2\rangle$

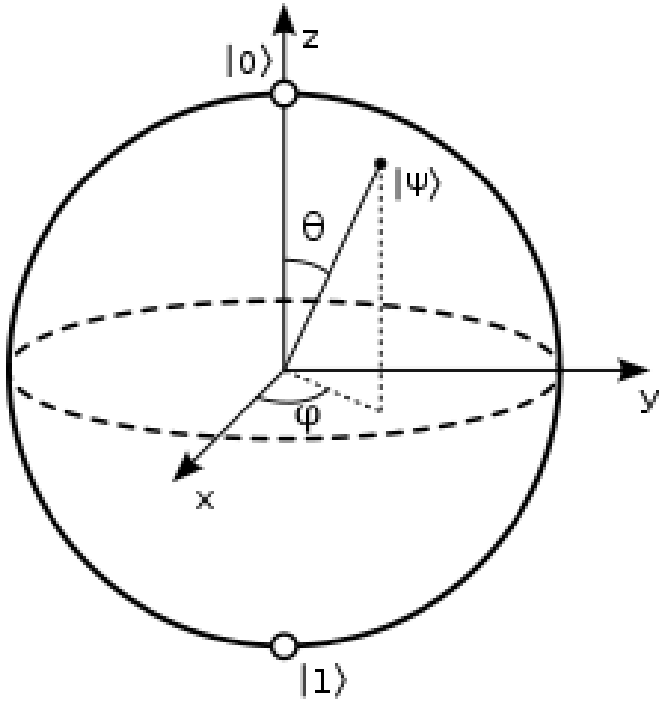
Quantum circuits

```
bell = Circuit().h(0).cnot(0, 1)
```



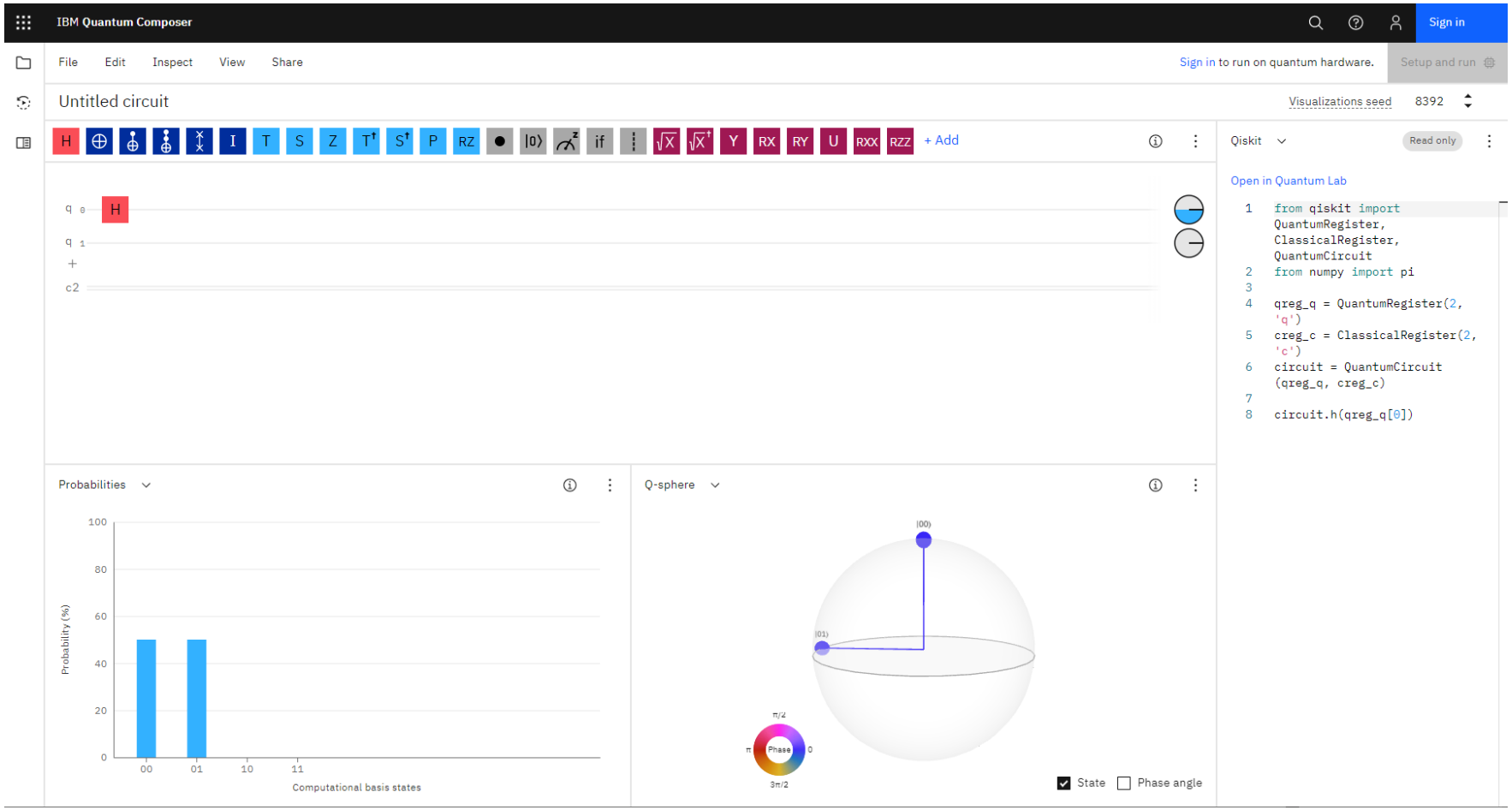
Quantum gates





QUANTUM PROGRAMMING

- IBM Quantum Composer: https://quantum-computing.ibm.com/composer



QUANTUM PROGRAMMING

Quantum Languages

Imperative languages

- OpenQASM by *IBM* for Qiskit and the IBM Q Exp
- Q# A language developed by *Microsoft*
- QCL One of the first implemented quantum programming languages
- Quantum pseudocode The first formalized language for description of quantum algorithms
- Q|SI> Is a platform embedded in .Net language supporting quantum programming in a quantum extension of while-language
- Q language Is the second implemented imperative quantum programming language
- qGCL Was defined by P. Zuliani in his PhD thesis. It's based on Guarded Command Language created by Edsger Dijkstra
- QMASM Quantum Macro Assemblera low-level language specific to quantum annealers such as the D-Wave
- Python (late 80s), although not an originally quantum programming language, is the most widely used, including by D-Wave, Google, Rigetti, IBM, Microsoft

Functional languages

- QFC and QPL QFC uses a flow chart syntax, whereas QPL uses a textual syntax.
- QML Is a Haskell-like quantum programming language by Altenkirch and Grattage.
- LIQUi|> Is a quantum simulation extension on the F# programming language.
- Quantum lambda calculi Are extensions of the classical lambda calculus introduced by Alonzo Church and Stephen Cole Kleene in the 1930s
- Quipper It is implemented as an embedded language, using Haskell as the host language

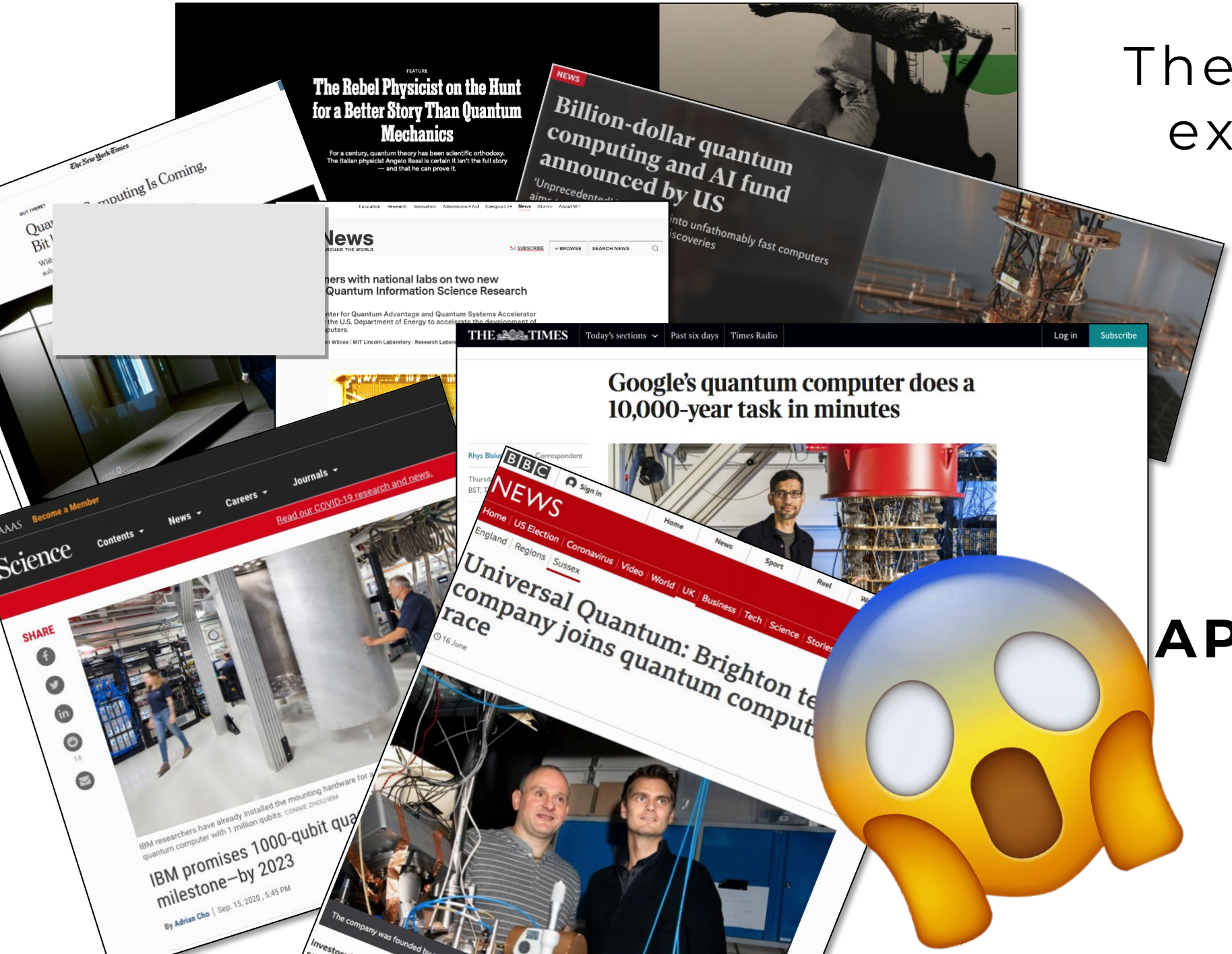
Quantum Service Providers

Cloud / QCaaS providers	Underlying technology providers
Amazon Braket	Simulators; Trapped Ion; Annealers; Superconducting
Microsoft Azure Quantum	QCs: Honeywell (Trapped Ion), IONQ (Trapped Ion), Superconducting; Optimisation: 1QBit Software, TOSHIBA Simulator (GPU), Microsoft Algorithms

Let’s go to the practical part!!

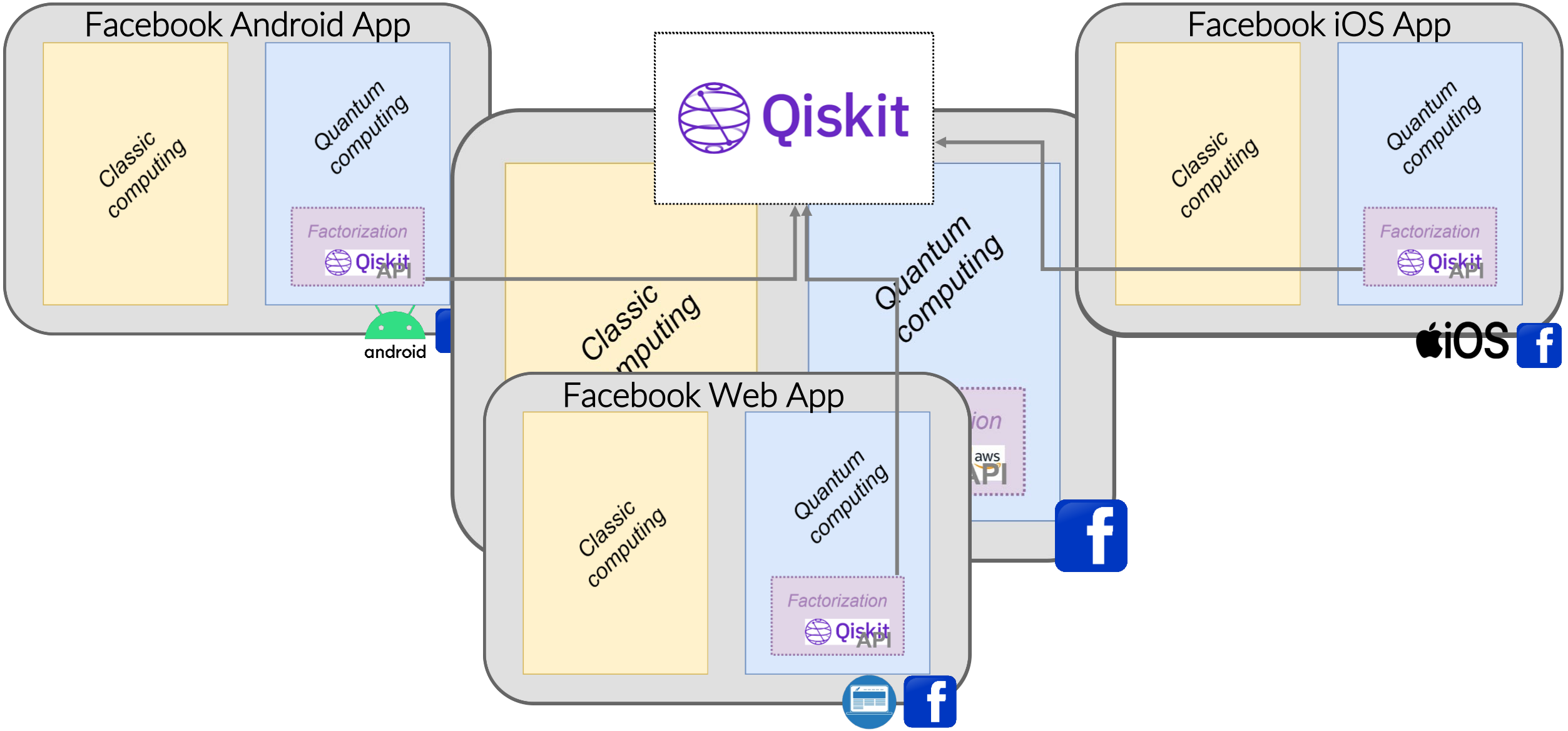
SCAN ME

Material of the examples and preparation of the environment

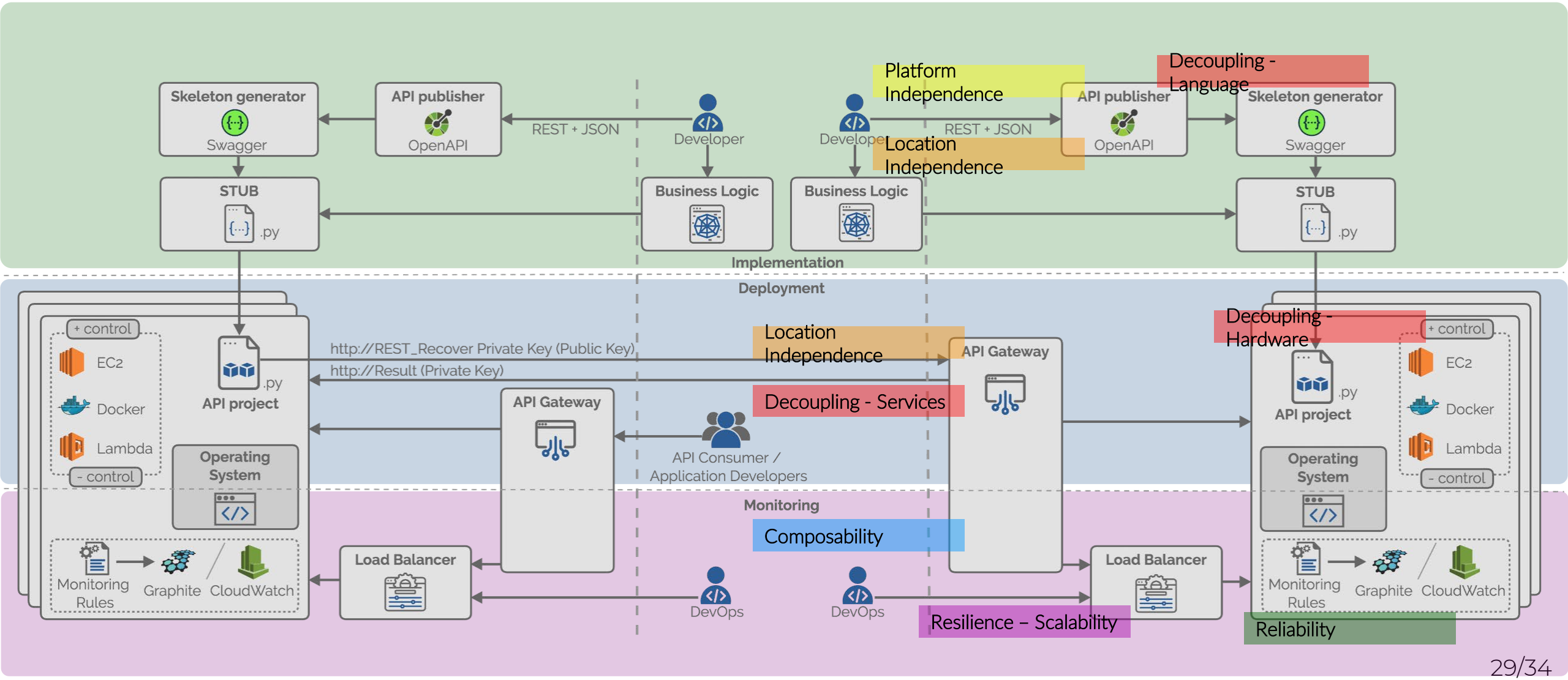


There are a lot of expectatives on Quantum Computing

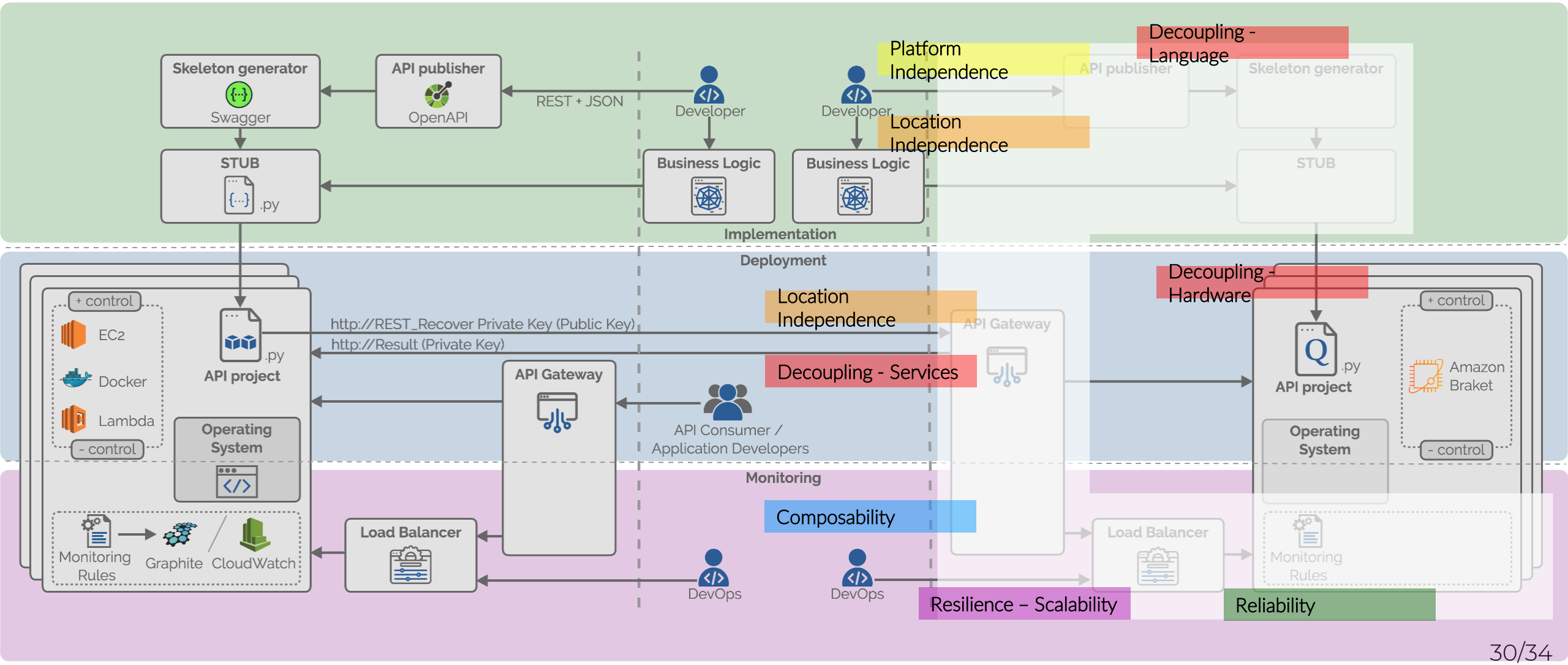
… BUT WE DON'T HAVE TOOLS TO BUILD REAL QUANTUM APPLICATIONS YET


Facebook Android App
Classic computing
Quantum computing
Factorization
Qiskit
API
android
Qiskit
Classic computing
Quantum computing
Facebook iOS App
Classic computing
Quantum computing
Factorization
Qiskit
API
iOS
Facebook Web App
Classic computing
Quantum computing
Factorization
Qiskit
API

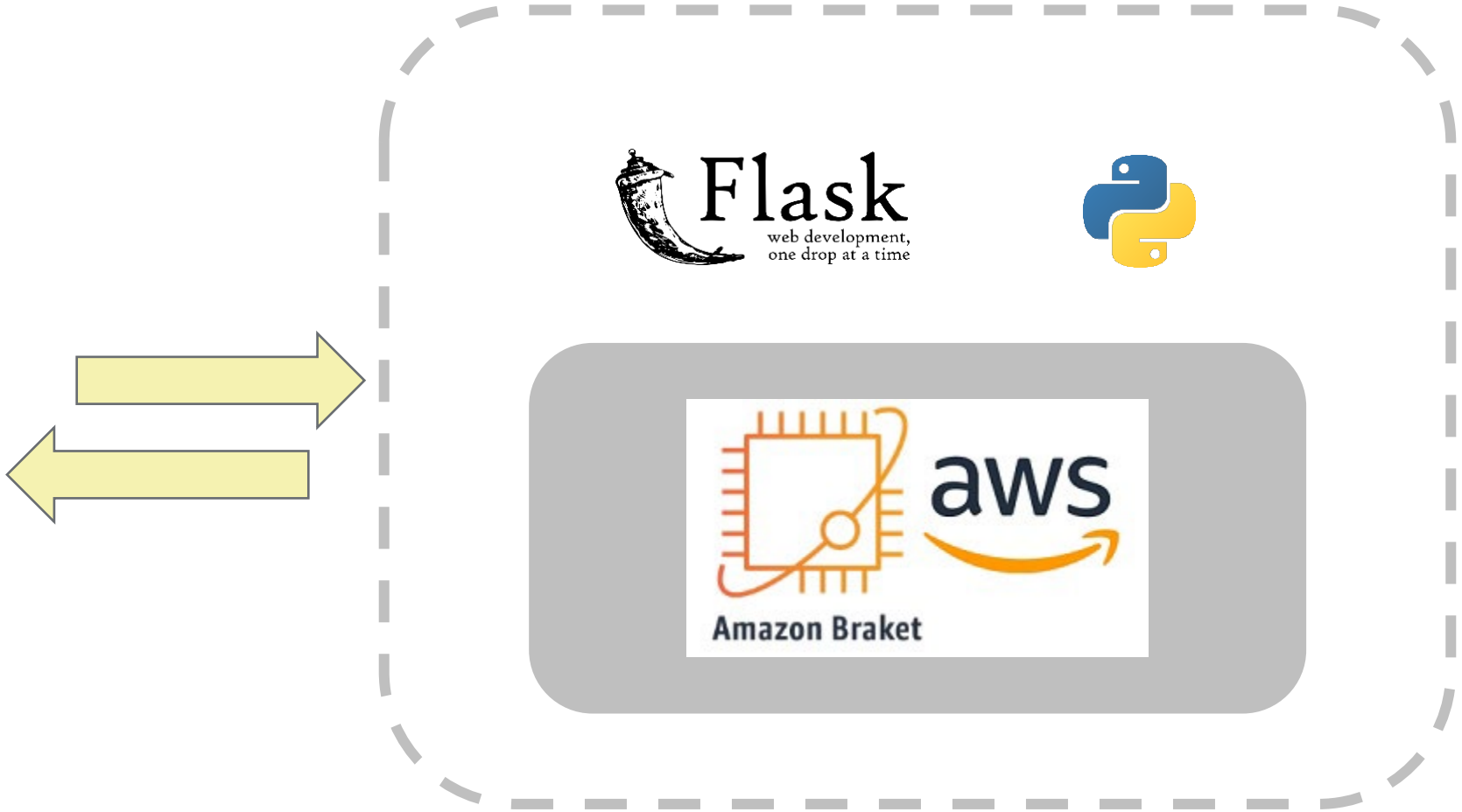
A good classical service implementation



Quantum service implementation

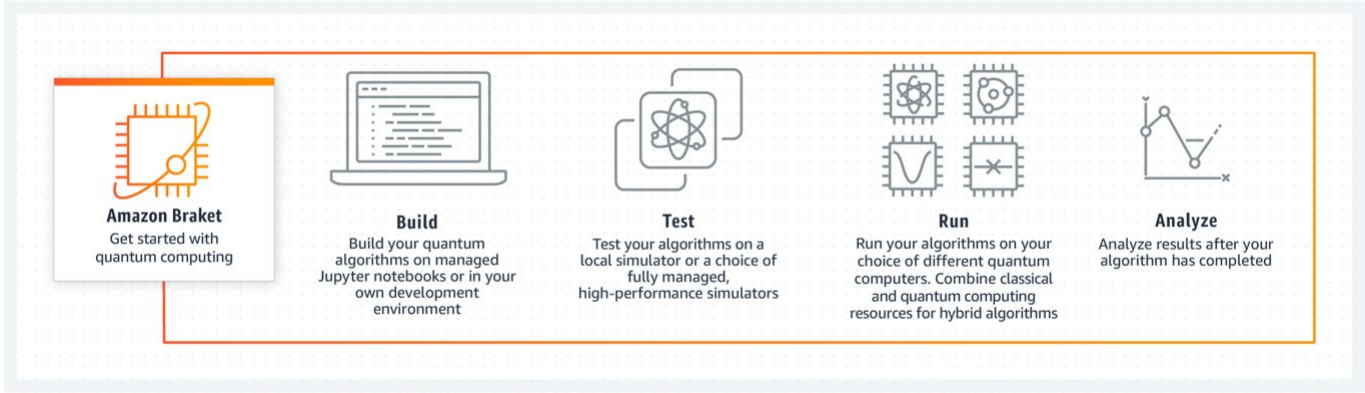


Quantum Web Services with AWS



Amazon Braket

- Amazon AWS Quantum Services Provider
- Accelerating quantum computing research
- Access to various quantum computers and simulators
- Cloud programming using Python
- Connection to local services through boto3



Flask

- Framework available in Python
- Creation of web applications
- Minimalist and "lines-of-code-saving" philosophy
- Based on Werkzeug's WSGI specification
- Jinja2 template engine
- BSD license

- **Most famous alternative to Django** due to its reduced learning curve
- Widely used for the **definition of web services** based on REST APIs

Other tools

Dwave Ocean

- Plugin for programming Dwave annealing machines

Pandas

- Library for data handling and data processing with Python

Matplotlib

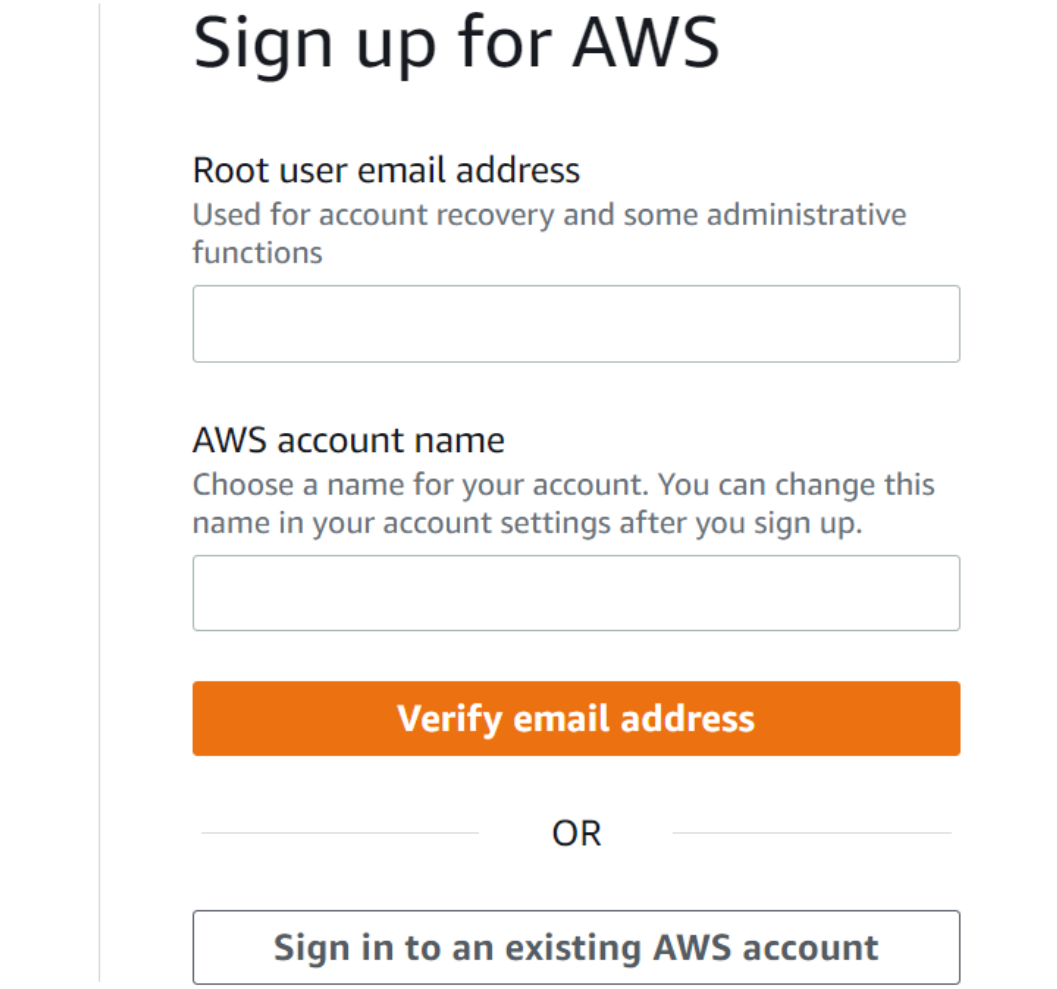
- Library for the creation of graphs and visual representations of data with Python.

ADVANCED STEPS: Setting up your environment with Amazon Braket

1. AWS account creation

1. Open AWS Web
2. Choose Create an AWS account
3. Fill in the information
4. Choose personal account
5. Fill in personal information

SCAN ME

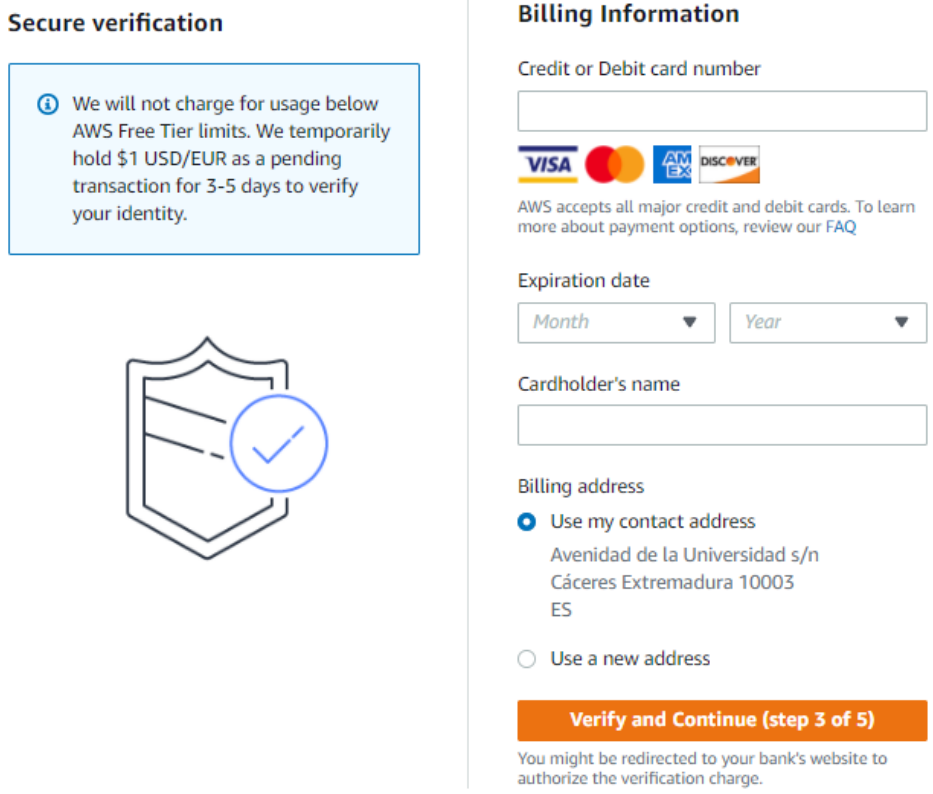


ADVANCED STEPS: Setting up your environment with Amazon Braket

1. AWS account creation

6. Enter payment method details

7. Verify telephone number

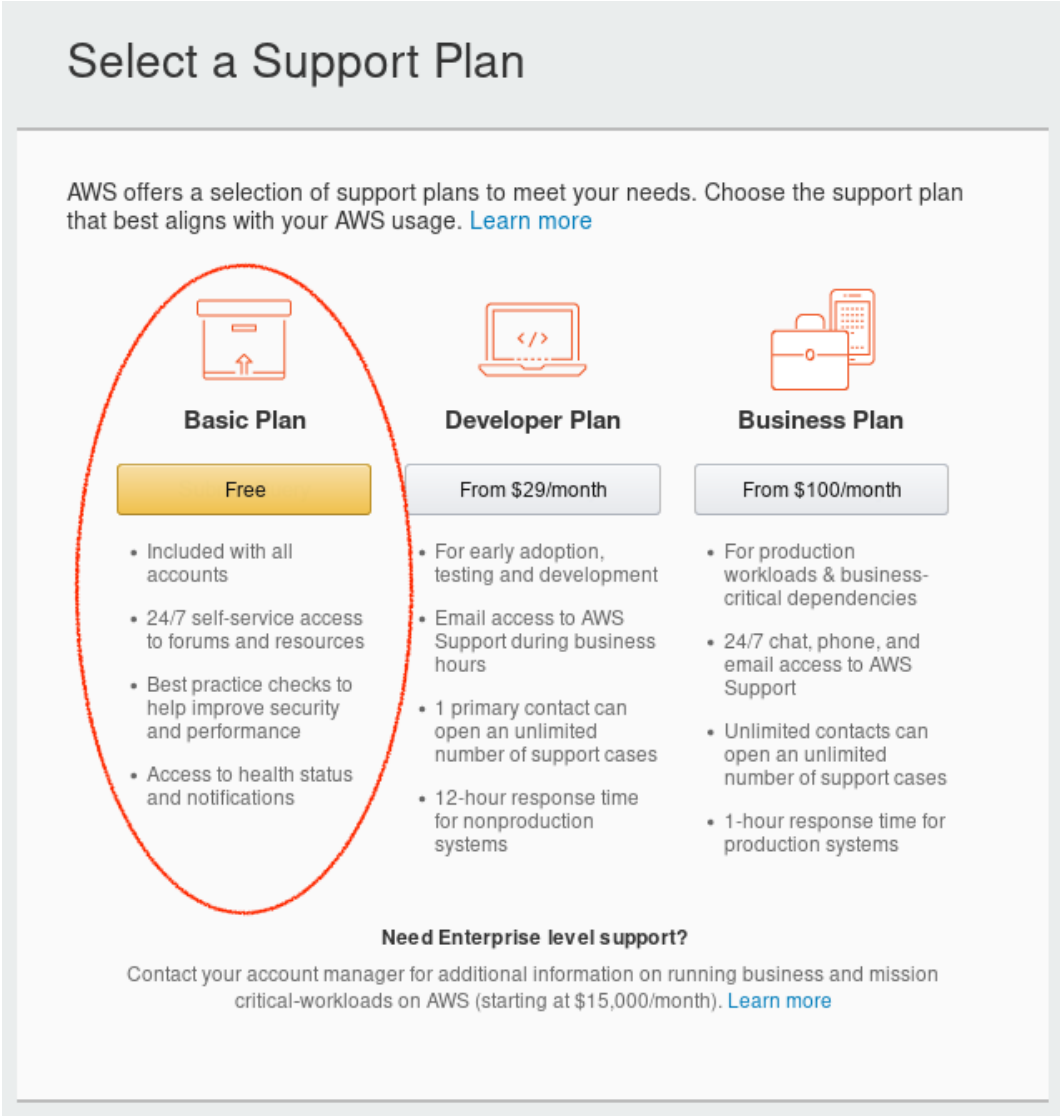


ADVANCED STEPS: Setting up your environment with Amazon Braket

1. AWS account creation

8. Choose Basic plan

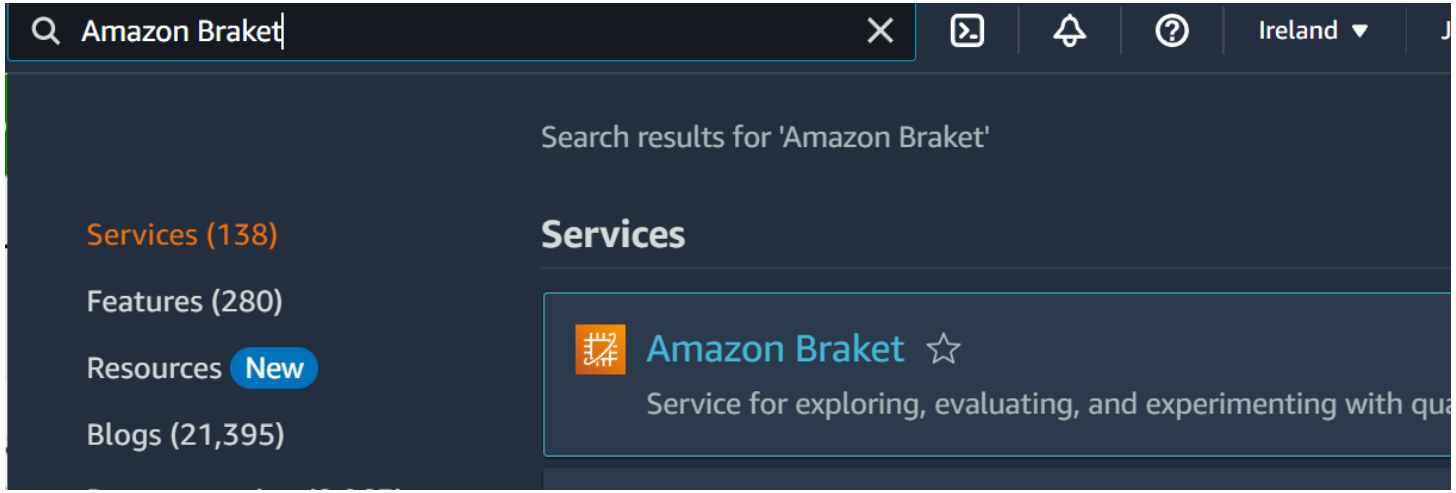
→ Login with "Root User"



ADVANCED STEPS: Setting up your environment with Amazon Braket

2. Using the AWS Braket service

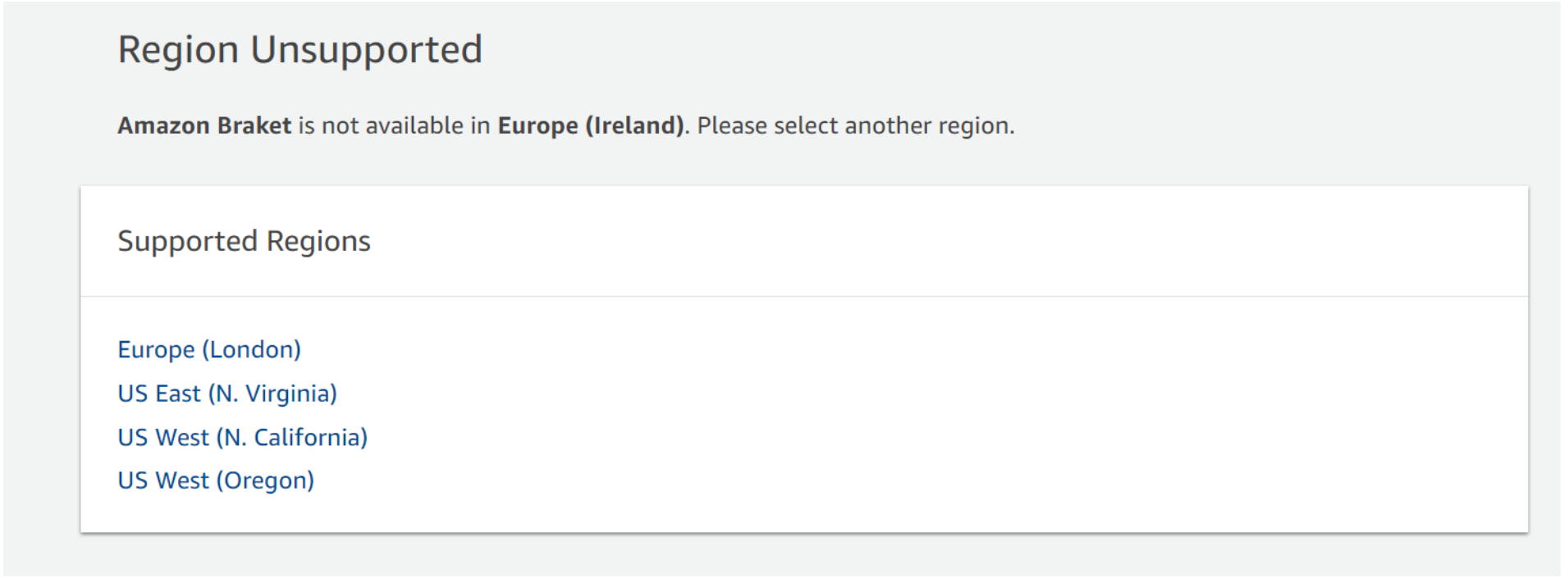
1. Search Amazon Braket in the Amazon Console



ADVANCED STEPS: Setting up your environment with Amazon Braket

2. Using the AWS Braket service

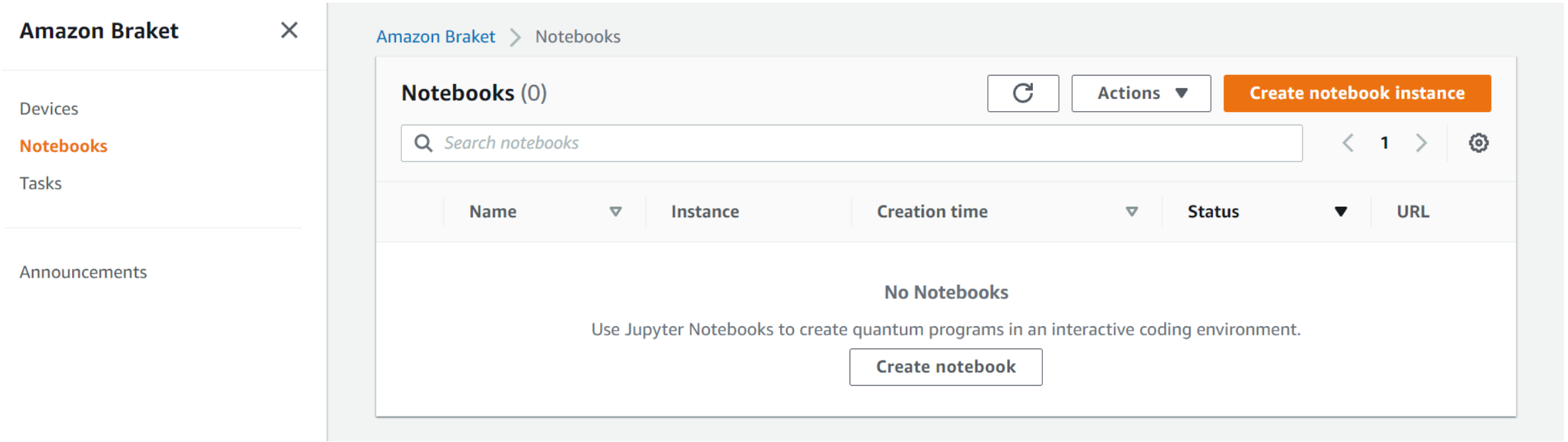
2. Select a valid region



ADVANCED STEPS: Setting up your environment with Amazon Braket

2. Using the AWS Braket service

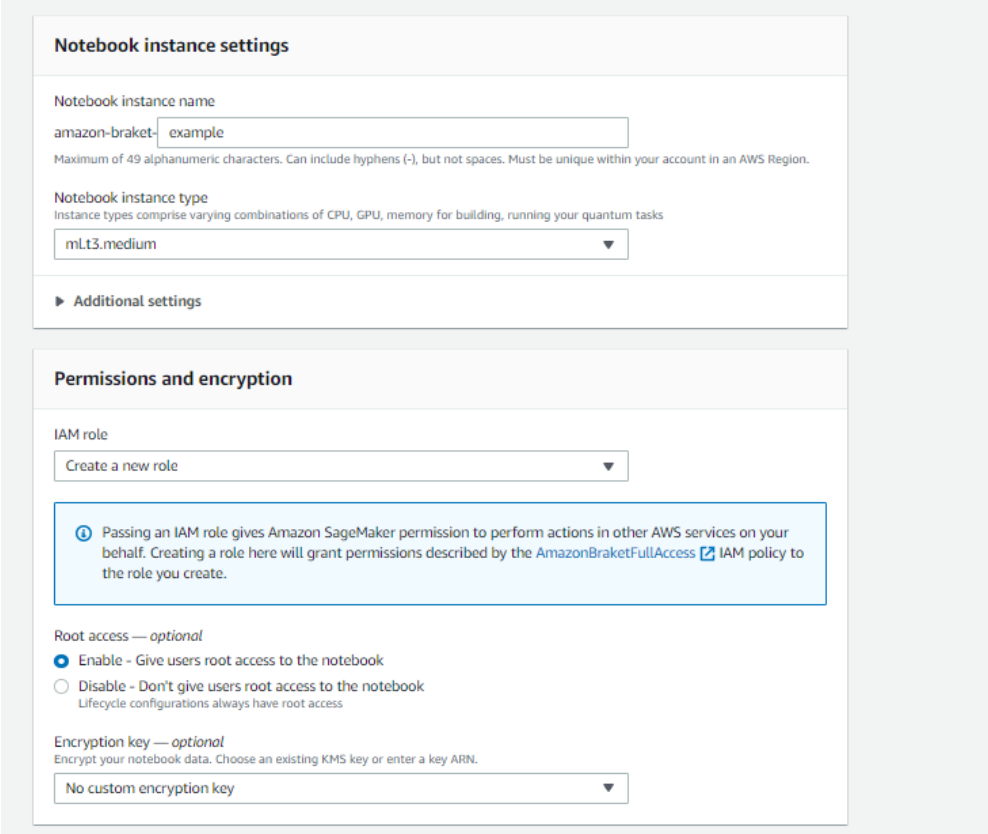
3. Create notebook instance



ADVANCED STEPS: Setting up your environment with Amazon Braket

2. Using the AWS Braket service

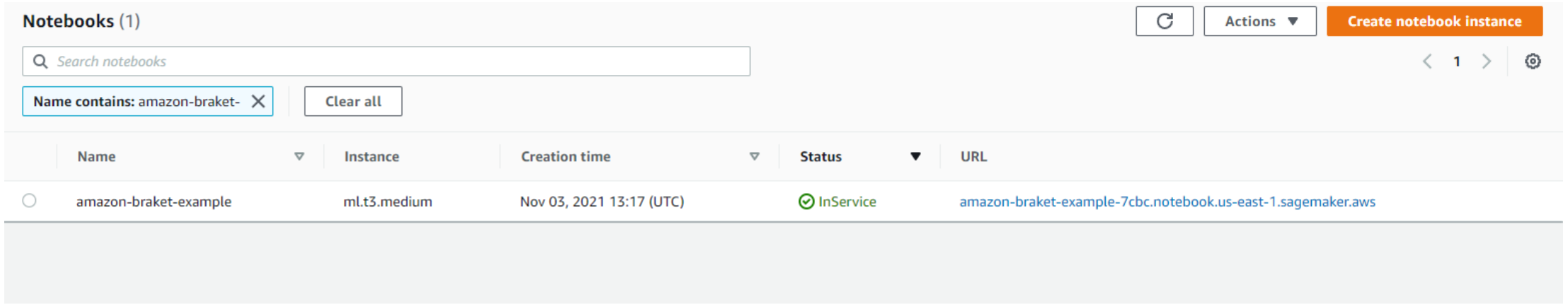
4. Select an instance of type *ml.t3.medium* and leave the rest of the options as default



ADVANCED STEPS: Setting up your environment with Amazon Braket

2. Using the AWS Braket service

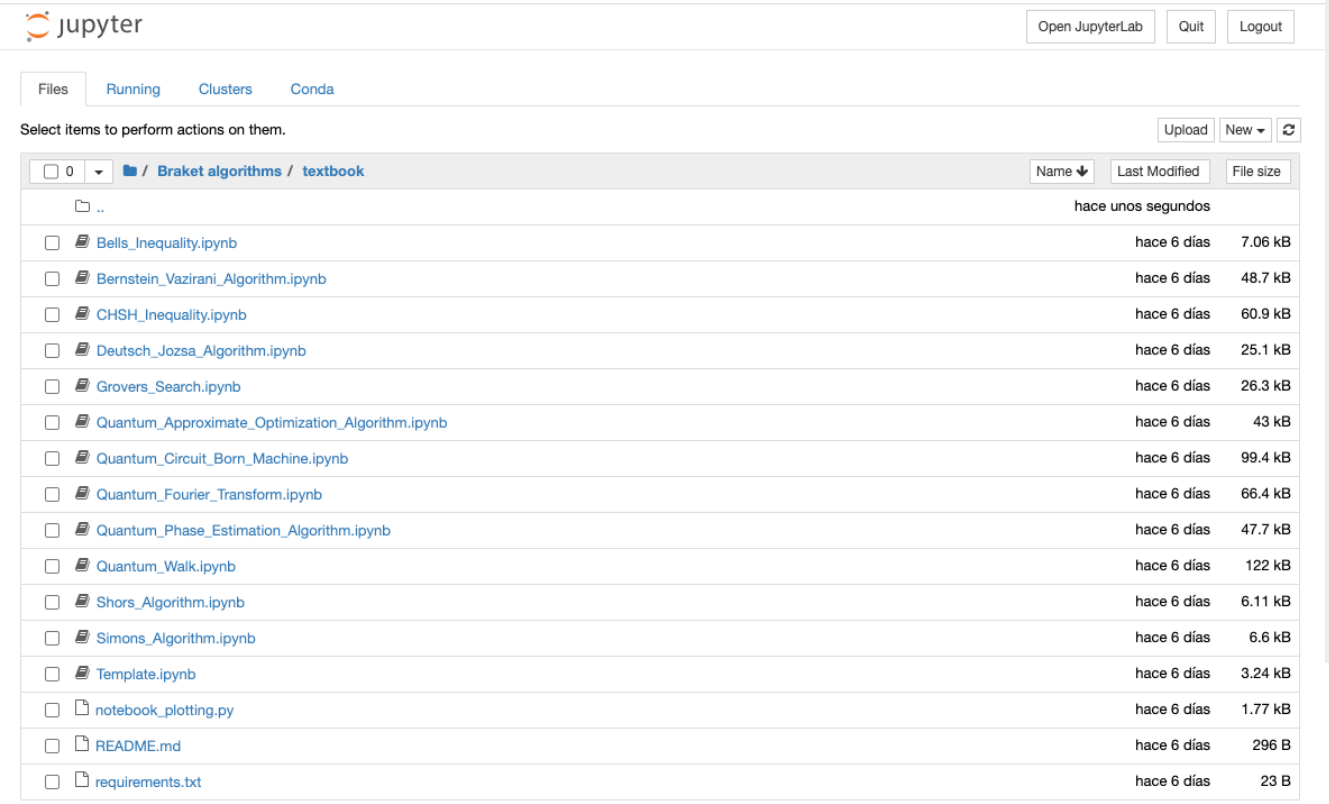
5. Check that the notebook has been created

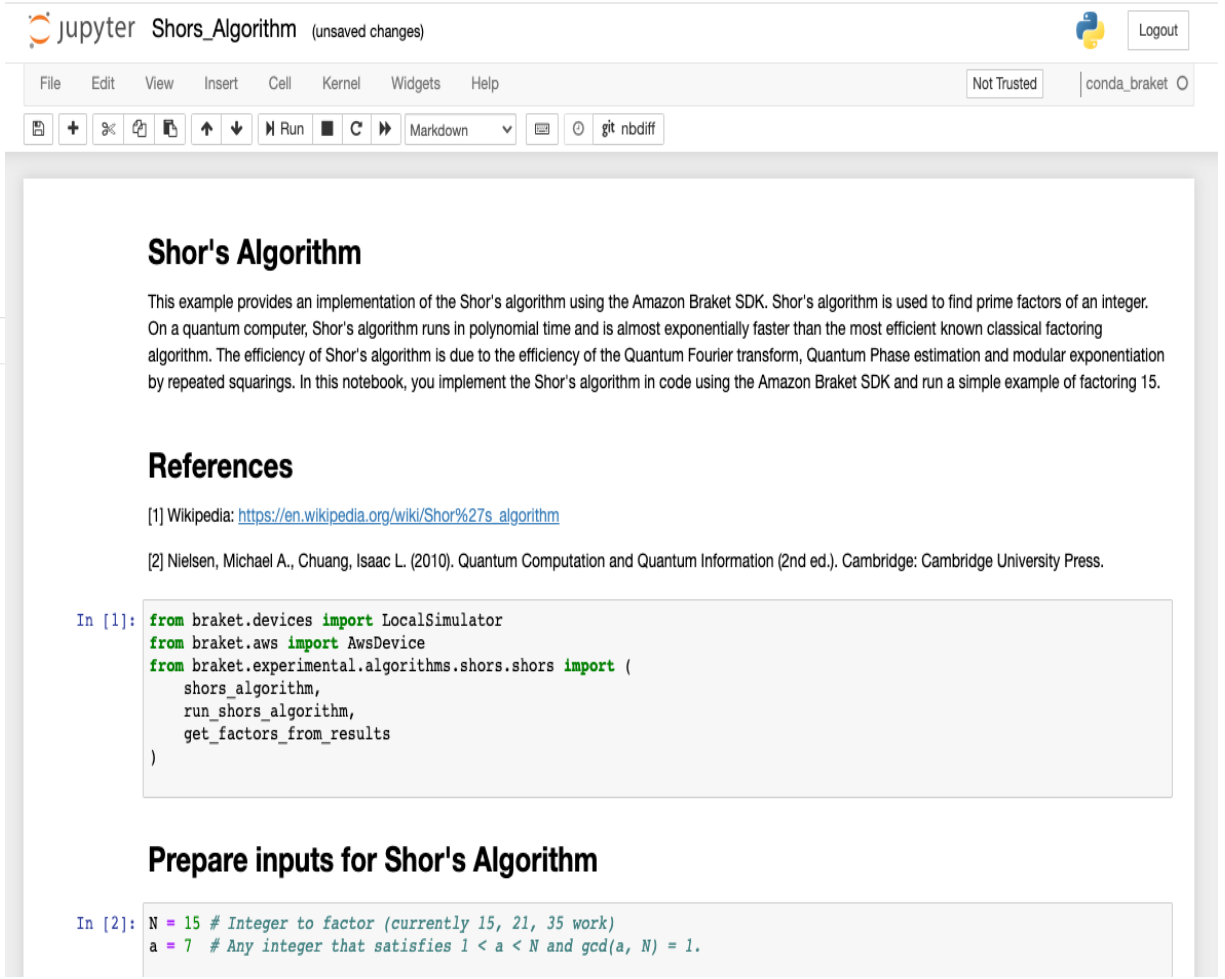


ADVANCED STEPS: Setting up your environment with Amazon Braket

2. Using the AWS Braket service

6. Execute notebook examples

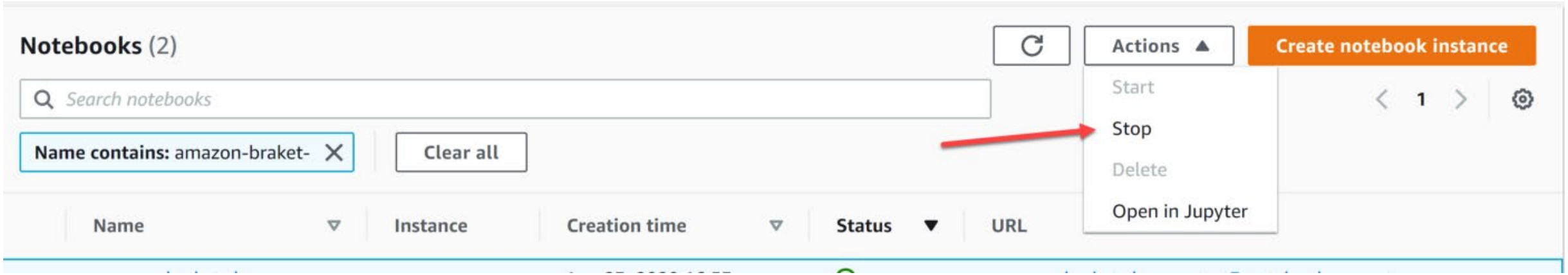




ADVANCED STEPS: Setting up your environment with Amazon Braket

2. Using the AWS Braket service

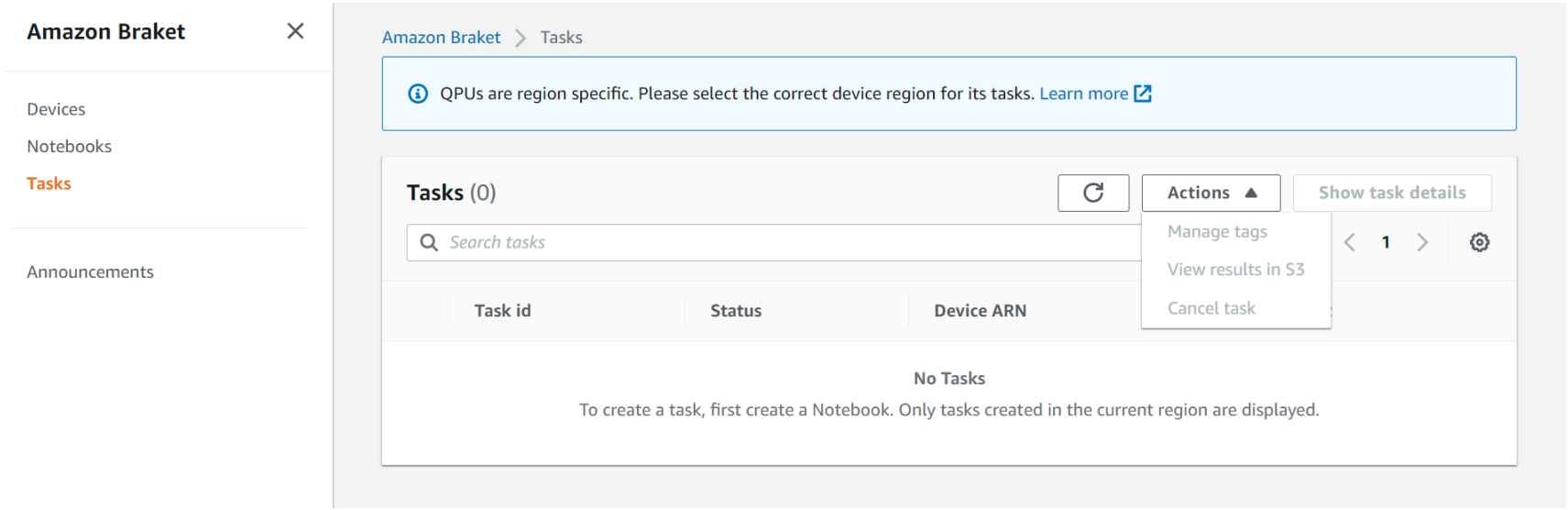
7. Stop notebook instance



ADVANCED STEPS: Setting up your environment with Amazon Braket

2. Using the AWS Braket service

8. Stop pending tasks



Local installation

1. Python 3 intallation

Windows

1. Go to the Windows Store
2. Search for Python 3.9
3. Click on get
4. Open a CMD
5. “python --version” o “python3 --version”

Ubuntu

1. sudo apt-get update
2. sudo apt-get install python3.9
3. “python --version” o “python3 --version”

MacOS

1. /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
2. export PATH="/usr/local/opt/python/libexec/bin:$PATH"
3. brew install Python
4. “python --version” o “python3 --version”

Local installation

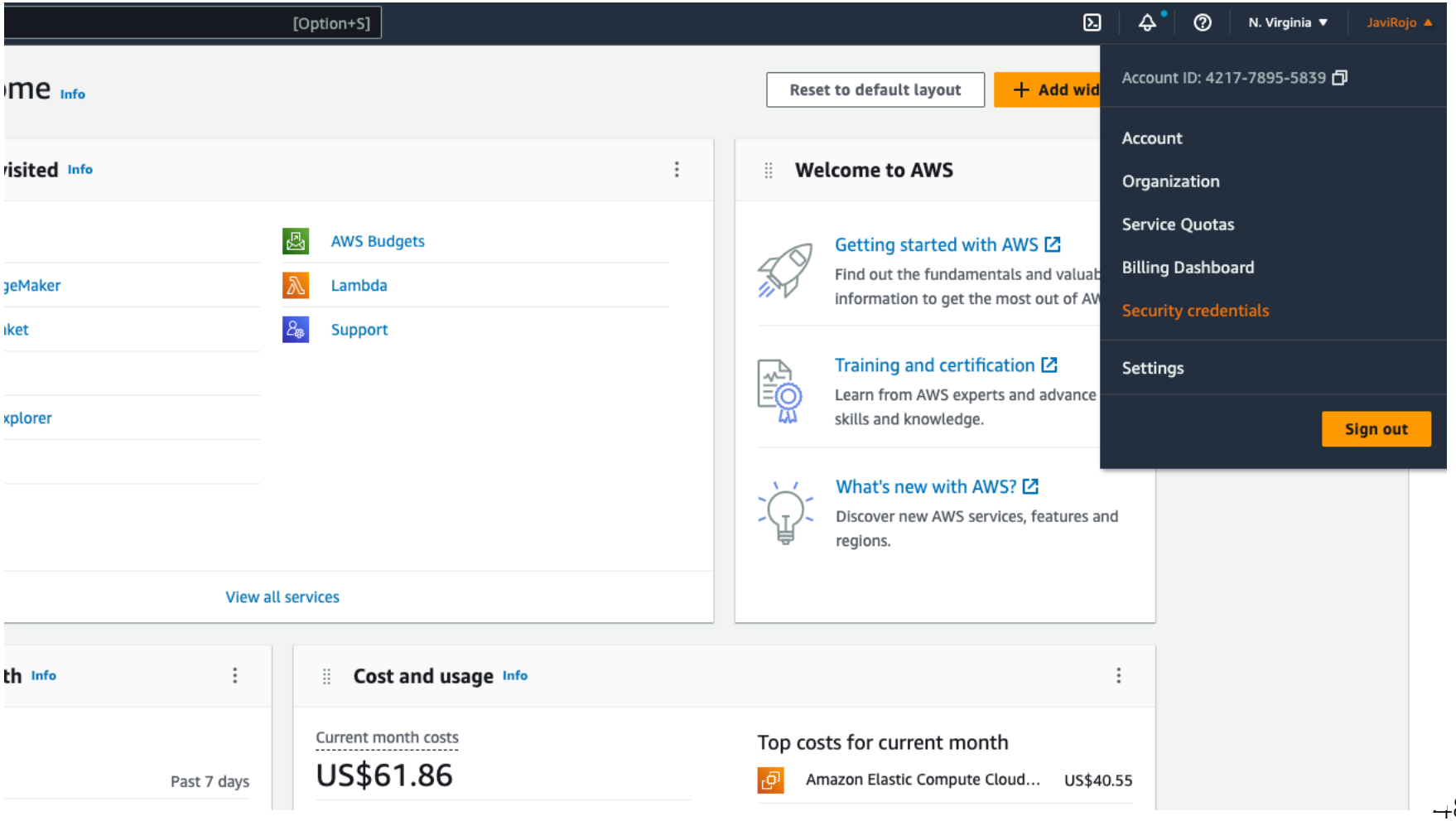
2. Installing the Amazon Braket SDK, Flask and other tools

```
cd amazon-braket-algorithm-library
pip install .
```

Local installation

3. Connecting our quantum services to the AWS account

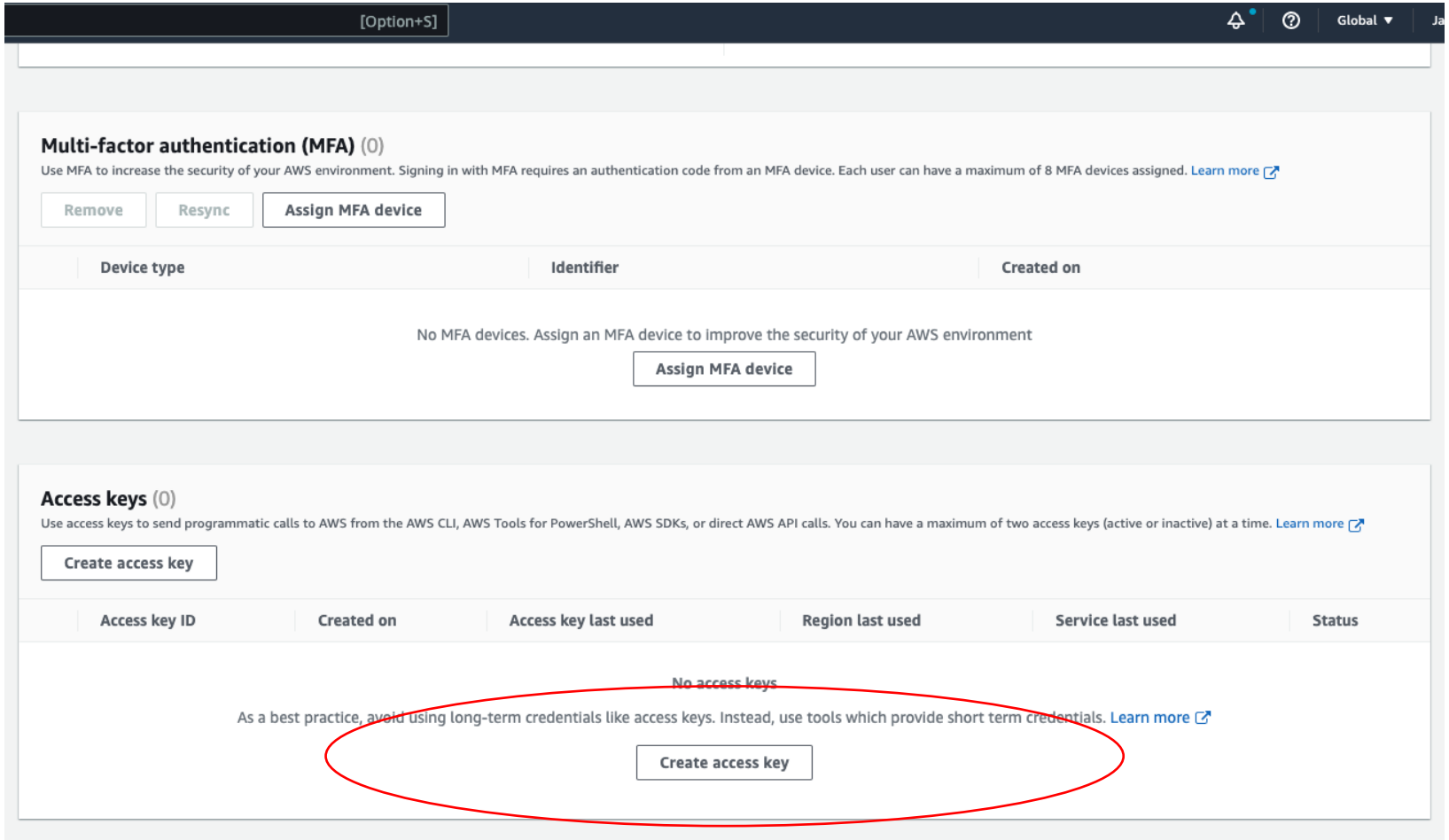
1. Get aws access key



Local installation

3. Connecting our quantum services to the AWS account

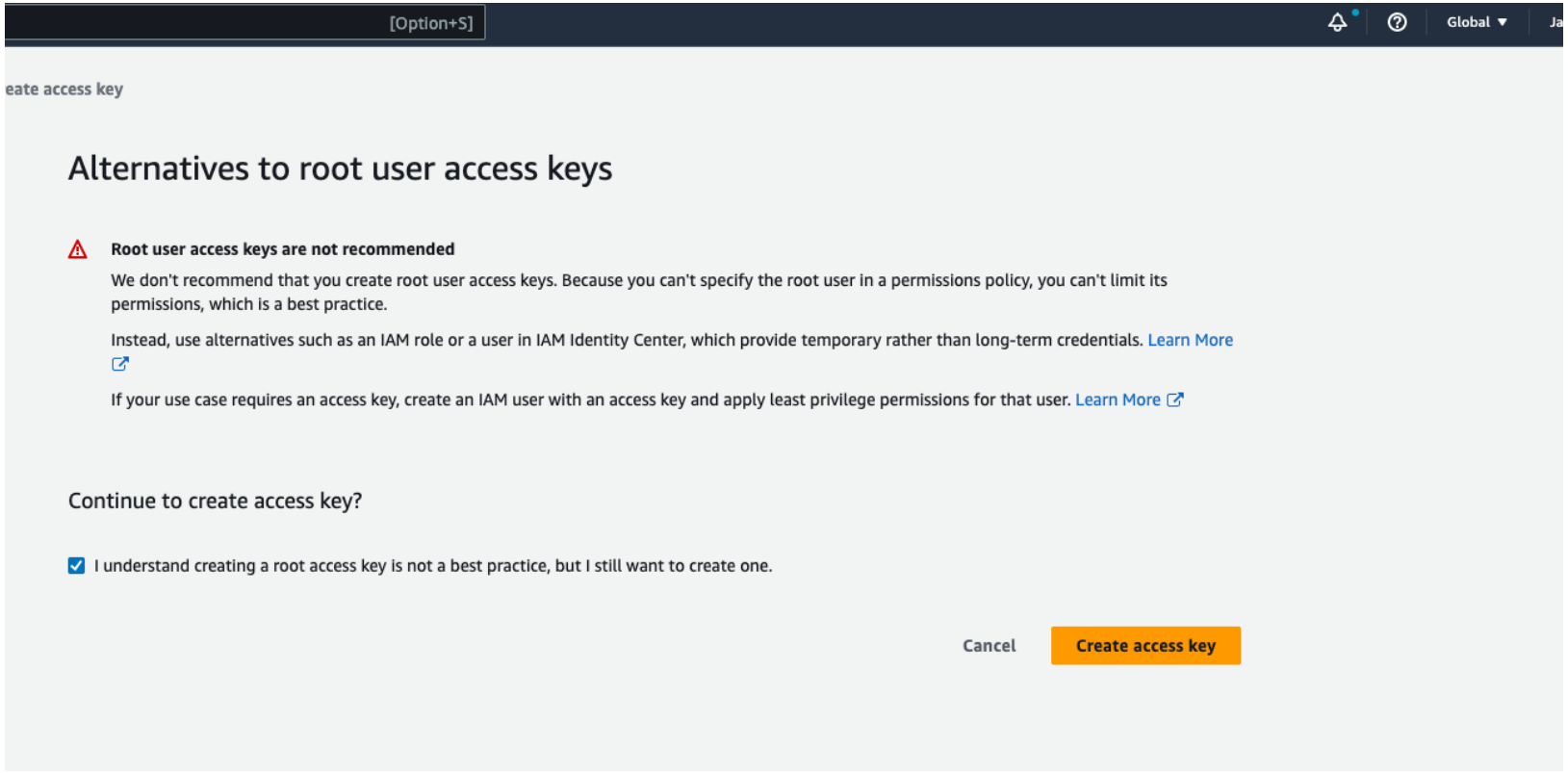
1. Get aws access key



Local installation

3. Connecting our quantum services to the AWS account

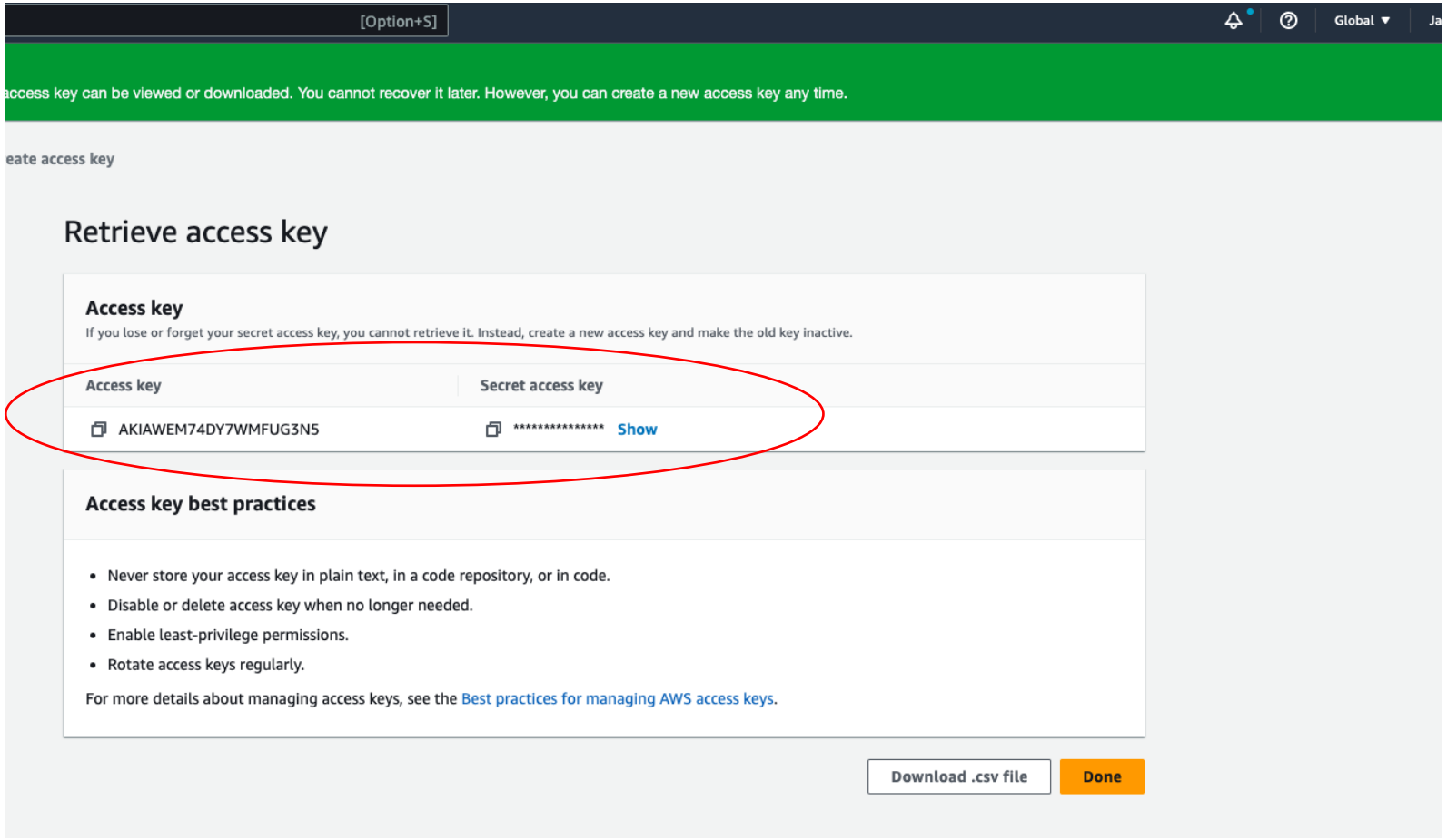
1. Get aws access key



Local installation

3. Connecting our quantum services to the AWS account

1. Get aws access key



Local installation

3. Connecting our quantum services to the AWS account

2. Entering passwords in our computer

2.1. Create folder ".aws" in root of our user

2.2. Create "credentials" and "config" files without extension in .aws folder

Local installation

3. Connecting our quantum services to the AWS account

File “credentials” (REPLACE access key id & secret access key)

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

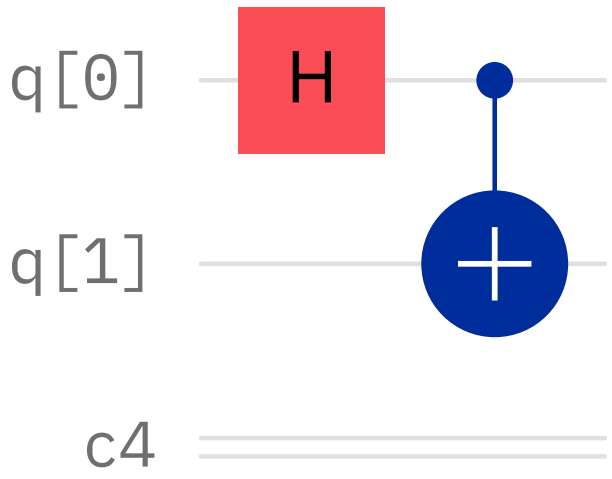
Local installation

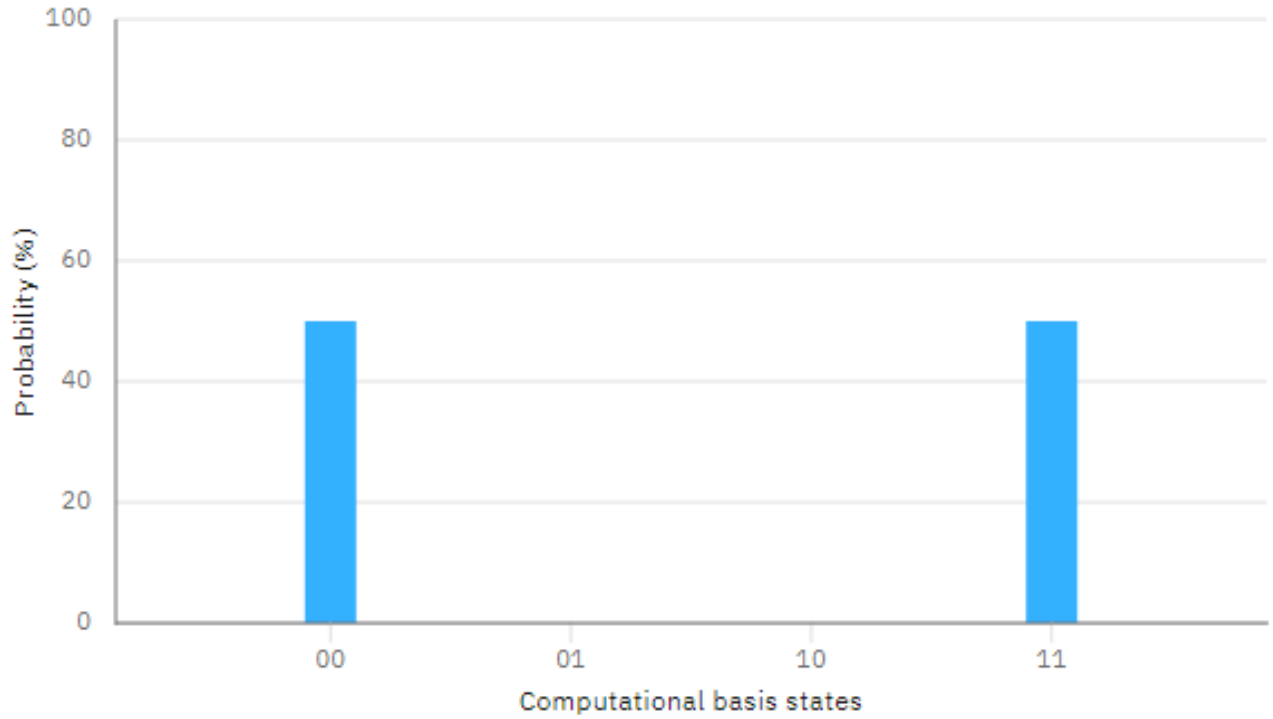
3. Conectar nuestros servicios cuánticos con la cuenta de AWS

File “config”

```
[default]
region=us-west-2
output=json
```

First quantum service: Bell state

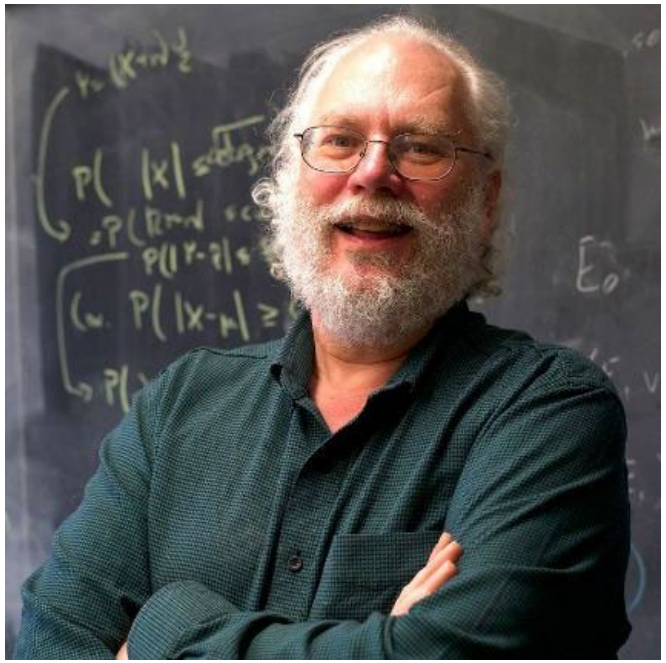




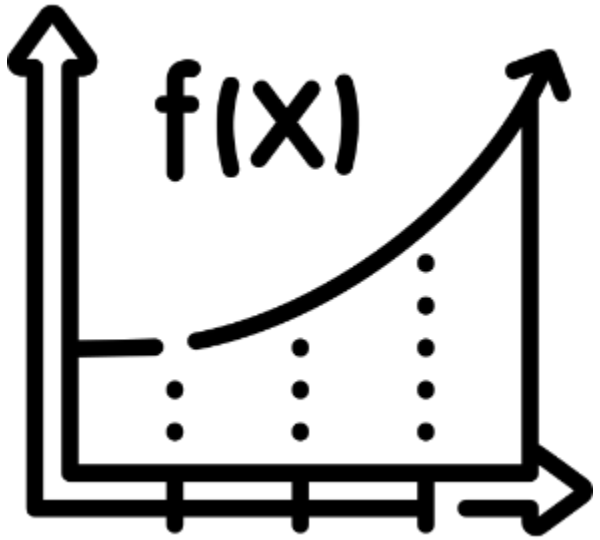
```
@app.route('/execute', methods=["get"])
def execute_quantum_task():
    # build a Bell state with two qubits.
    # Here 'cnot(control=0, target=1)' can be simplified as 'cnot(0,1)'
    bell = Circuit().h(0).cnot(control=0, target=1)

    # set up device
    device = LocalSimulator()

    # run circuit
    result = device.run(bell, shots=100).result()
    # get measurement shots
    counts = result.measurement_counts

    # print counts
    print(counts)
    # plot using Counter
    plt.clf()
    plt.bar(counts.keys(), counts.values())
    plt.xlabel('bitstrings')
    plt.ylabel('counts')
    plt.savefig("result.png")
    return send_file("result.png", mimetype='image/png')
```

- Shor's algorithm is a quantum computer algorithm for finding the prime factors of an integer

- It was developed in **1994** by the American mathematician **Peter Shor**



- Quantum algorithm for factoring a number N in O((log N)3) time and O(log N) space



- Is a method that allows finding prime factors of a number in an efficient way

- It can be carried out **classically** or using **quantum circuits**.

- This algorithm demonstrates that the integer factorization problem can be efficiently solved on a quantum computer.

- This algorithm is interesting because many public key cryptographies, such as **RSA**, are based on the product of two prime numbers.

- This kind of cryptography could become deprecated if the algorithm could be efficiently implemented on a quantum computer.

- Shor's algorithm was practically demonstrated in 2001 by IBM, which decomposed 15 into its factors 3 and 5, using a quantum computer with 7 qubits.

REUTERS SCIENCE 12.19.2001 02:15 PM

Big Blue Takes Quantum Step

IBM researchers make another advance in quantum computing, demonstrating "Shor's Algorithm," which can break large encryption codes.

NEW YORK -- IBM researchers said on Wednesday they have demonstrated a calculation that could be used to break complicated codes, marking a small step in the advance of quantum computing, a technology based on quantum mechanics.

IBM scientists will publish details in the scientific journal *Nature* on Thursday of the demonstration of "Shor's Algorithm," a method of factoring numbers that was developed in 1994 by AT&T scientist Peter Shor.

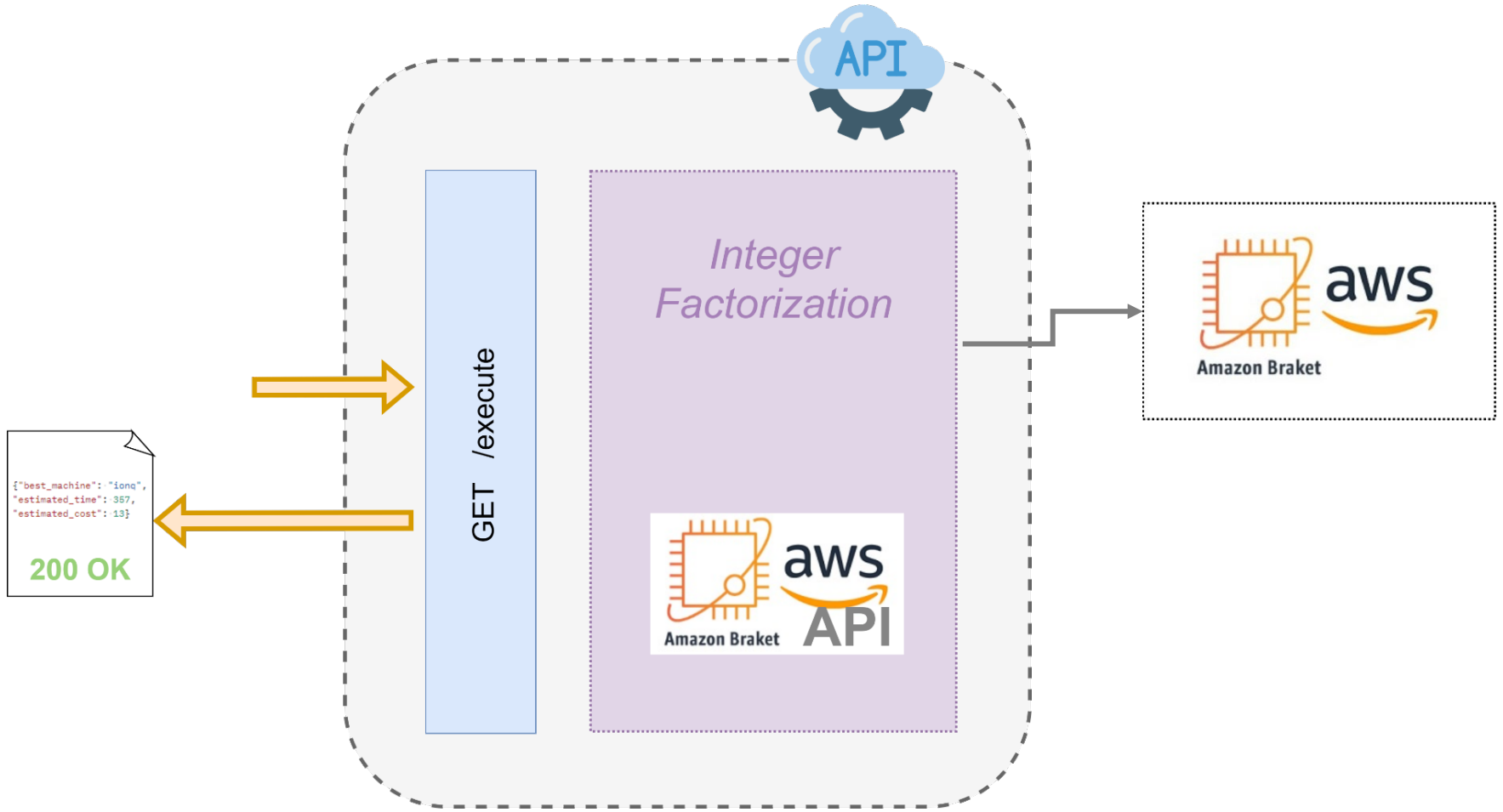
It was that algorithm, and the promise it holds for its ability to break large encryption codes, that spurred interest in quantum computing in the 1990s.

Quantum computing is one of several paths that researchers are taking as they strive to make smaller and smaller microchips. Under Moore's Law, which was set

WATCH

What the What Is Quantum Computing? We've Got You Covered

Shor Service



Valencia, D., García-Alonso, J., Rojo, J., Moguel, E., Berrocal, J., & Murillo, J. M. "Hybrid Clasical-Quantum Software Services Systems: Exploration of the Rough Edges.“ QUATIC'21. 2021.

Shor Service

Two different implementations for each simulator:

```
def shor(N, a):
# N = Integer to factor (currently 15, 21, 35 work)
# a = Any integer that satisfies 1 < a < N and gcd(a, N) = 1.


    shors_circuit = shors_algorithm(N, a)

    local_simulator = LocalSimulator()

    output = run_shors_algorithm(shors_circuit, local_simulator)

    guessed_factors = get_factors_from_results(output, N, a)

    return (guessed_factors)

@app.route('/shor_simulator', methods=['GET'])
def shor_simulator():
    N = int(request.args.get('N'))
    a = int(request.args.get('a'))
    response = shor(N, a)

    return jsonify(str(response))
```

```
@app.route('/shor_simulator_sv1', methods=['GET'])
def shor_simulator_sv1():
    N = int(request.args.get('N'))
    a = int(request.args.get('a'))
    shors_circuit = shors_algorithm(N, a)
    managed_sim = AwsDevice("arn:aws:braket:::device/quantum-simulator/amazon/sv1")
    output = run_shors_algorithm(shors_circuit, managed_sim)
    guessed_factors = get_factors_from_results(output, N, a)

    return jsonify(str(guessed_factors))
```

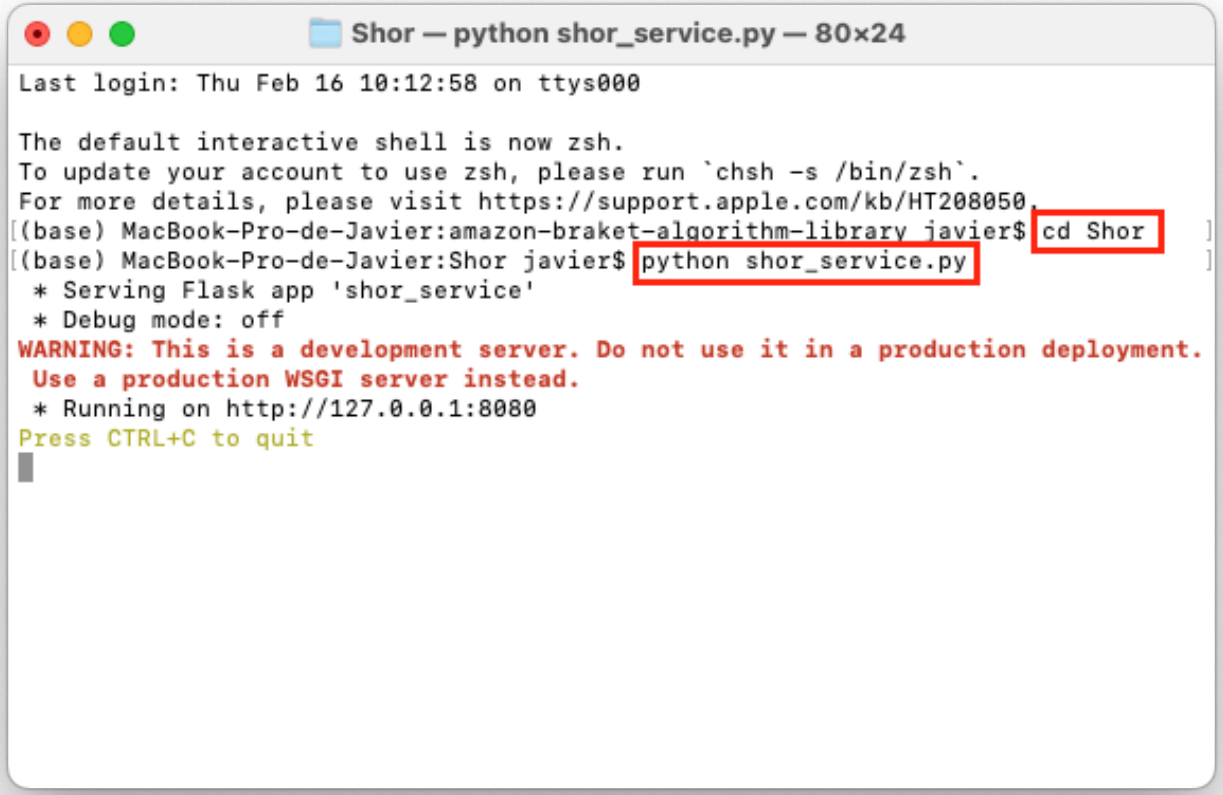
API Deployment

1. Run the follow commands in your system terminal:

- cd Shor
- python shor_service.py

  or

  python3 shor_service.py

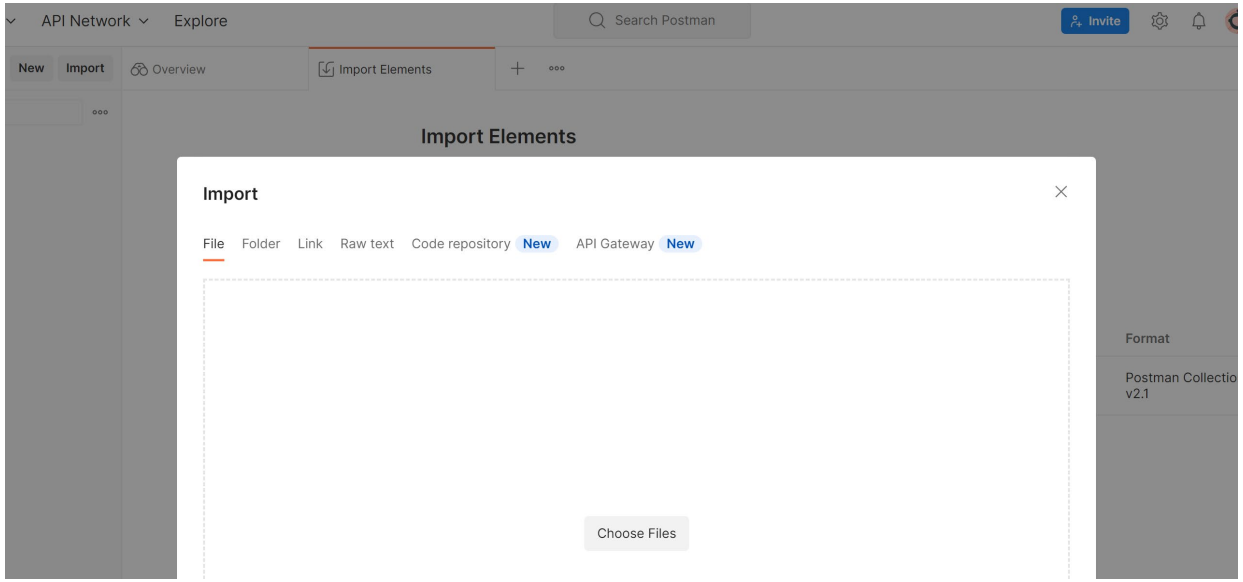


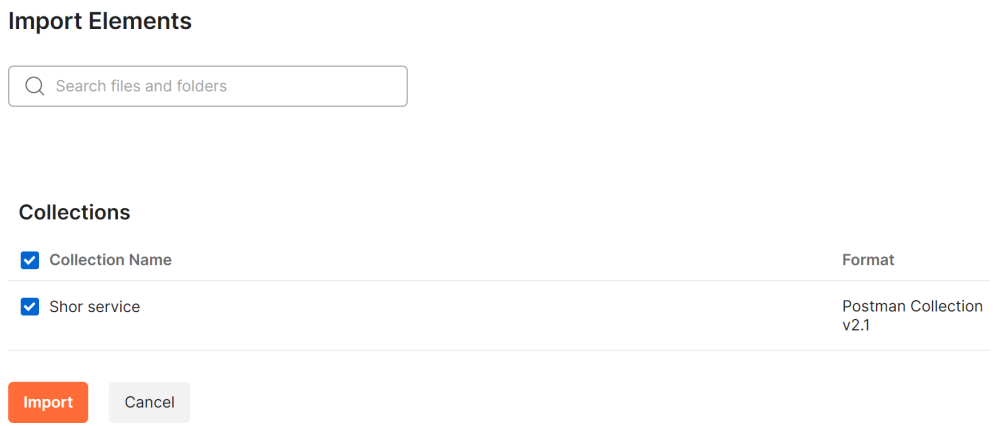
API Deployment

2. Download Postman:

https://www.postman.com/downloads/

3. Import collection file (Shor service.postman_collection.json) from Shor folder:



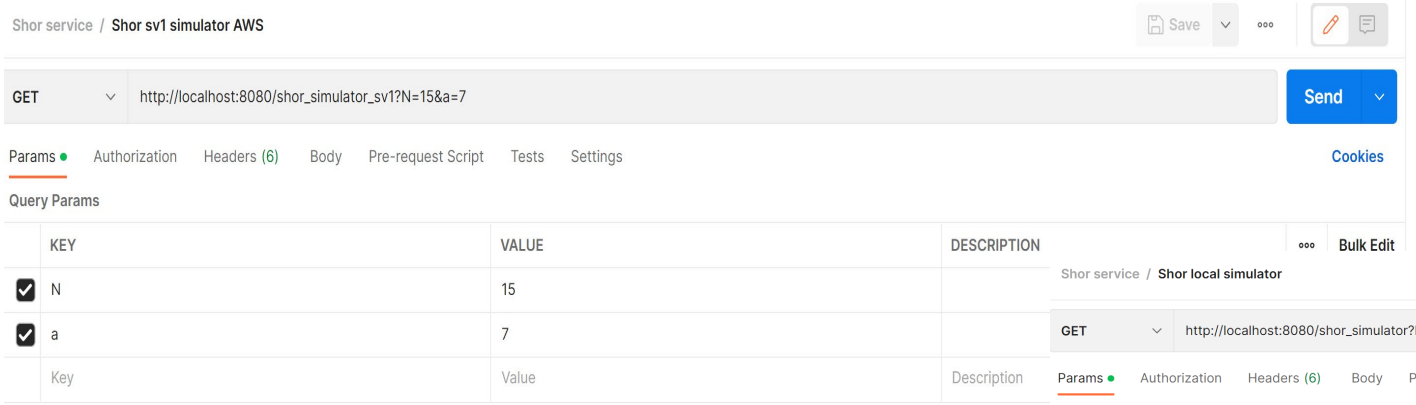


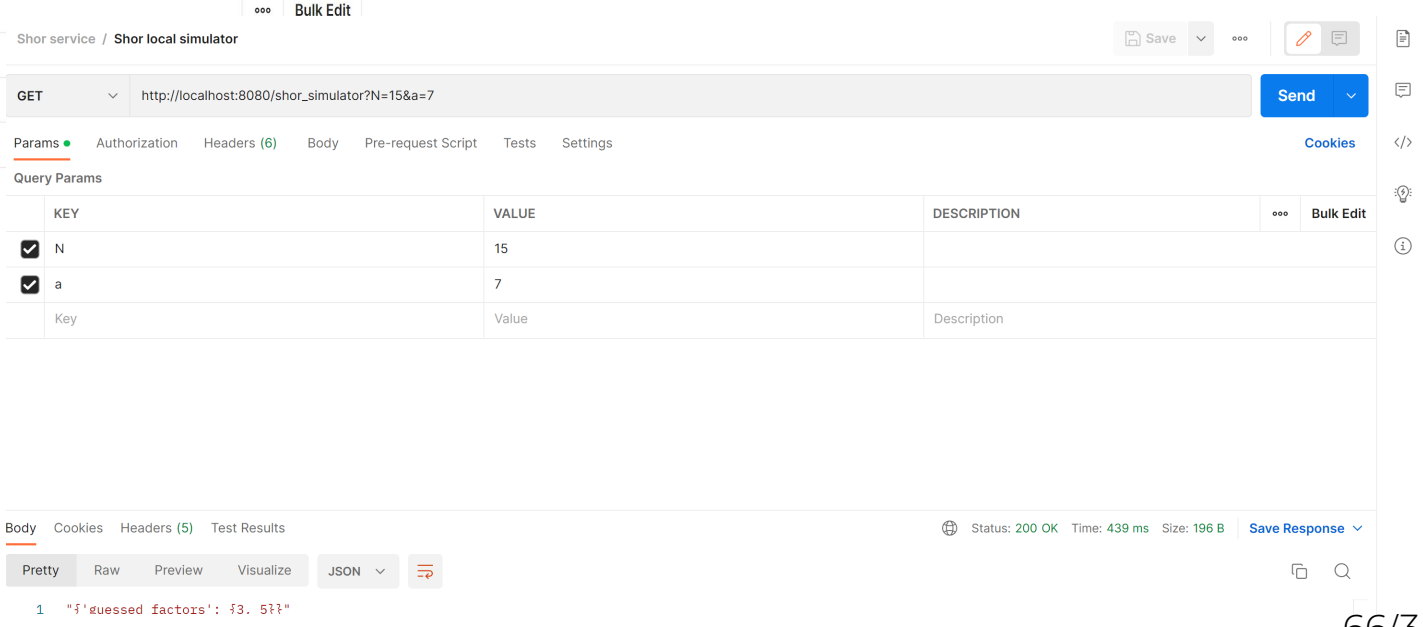
SHOR SERVICE

API Deployment

4. Test the shor service in postman:

- **Shor local simulator:** http://localhost:8080/shor_simulator?N=15&a=7
- **Shor sv1 simulator:** http://localhost:8080/shor_simulator_sv1?N=15&a=7





Quantum Web Services: development and deployment

Quantum OpenAPI

Javier Romero-Álvarez
Jaime Alvarado-Valiente
Enrique Moguel
José García-Alonso

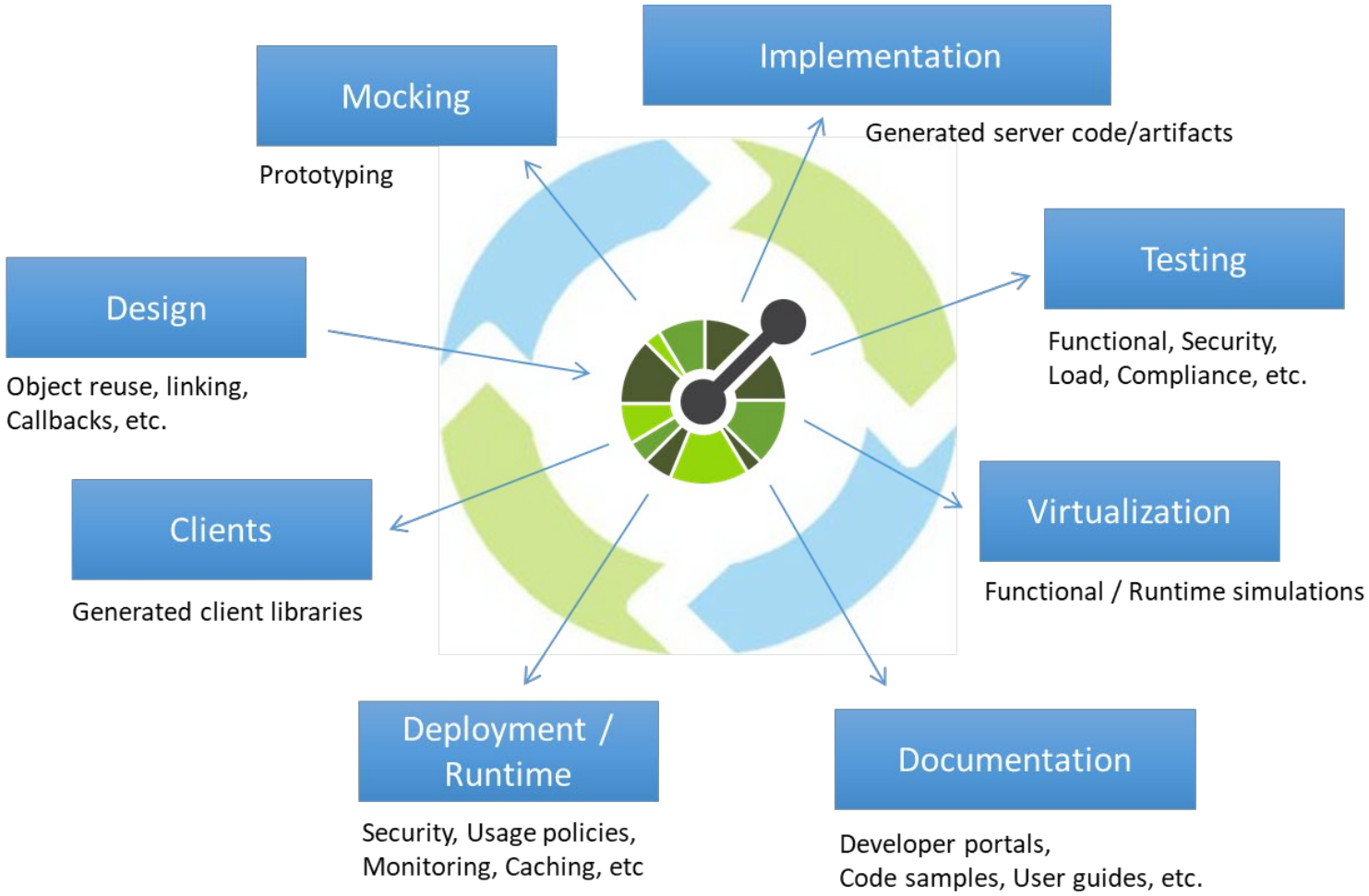
MINISTERIO DE CIENCIA E INNOVACIÓN
UNIÓN EUROPEA
FONDO EUROPEO DE DESARROLLO REGIONAL
AGENCIA ESTATAL DE INVESTIGACIÓN
JUNTA DE EXTREMADURA
Consejería de Economía, Ciencia y Agenda Digital
SPILab by Quercus SEG
Software Engineering Group
QUERCUS
UNIVERSIDAD DE EXTREMADURA

OUTLINE

1. Introduction
   a. OpenAPI
   b. GitHub Actions
2. Quantum service generation
3. Quantum service deployment
4. Workflow for the continuous deployment

INTRODUCTION

- The **OpenAPI** Initiative (OAI) was created by a consortium to standardise the way REST APIs are described.
- OpenAPI is a specification for describing, consuming and visualising RESTful web services.

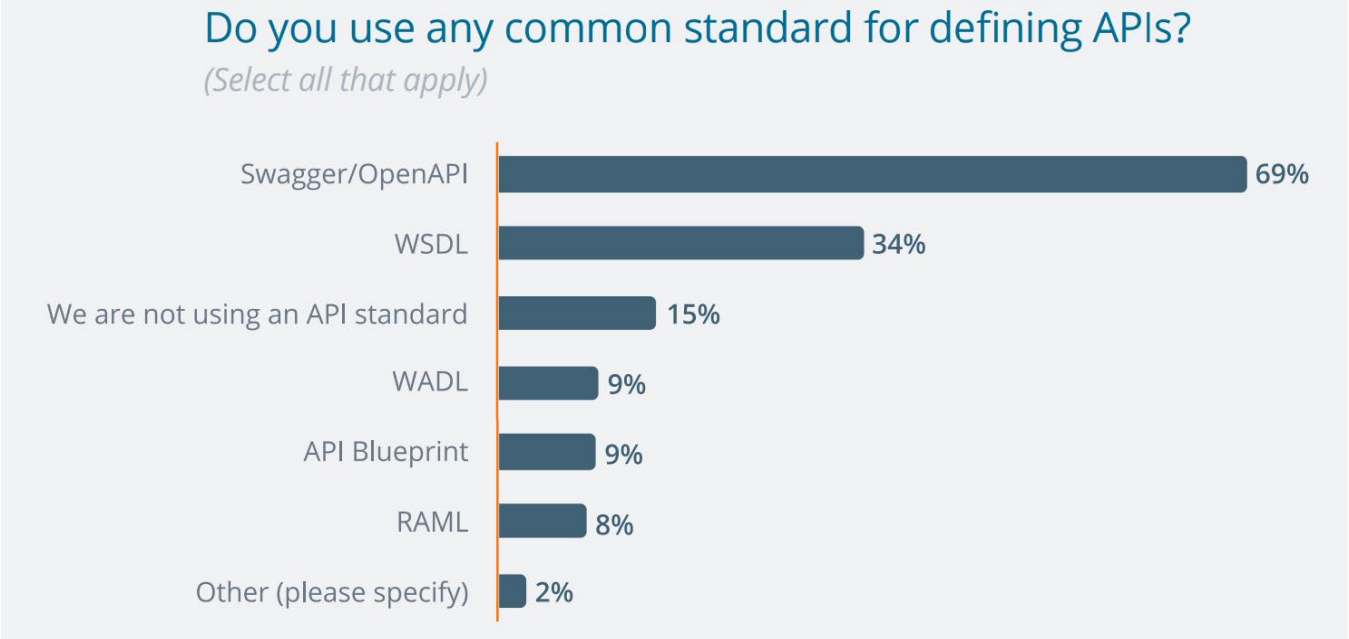


https://swagger.io/blog/api-strategy/benefits-of-openapi-api-development/

Introduction

Why OpenAPI?

OpenAPI is used and supported by large companies and, over the years, has established itself as the standard to use for building APIs for apps



https://smartbear.com/blog/a-look-at-the-key-takeaways-from state-of-api-2020/

INTRODUCTION

Open API Tools

- **Editor**

http://editor.swagger.io

- **API explorer**

http://petstore.swagger.io

- **Validator**

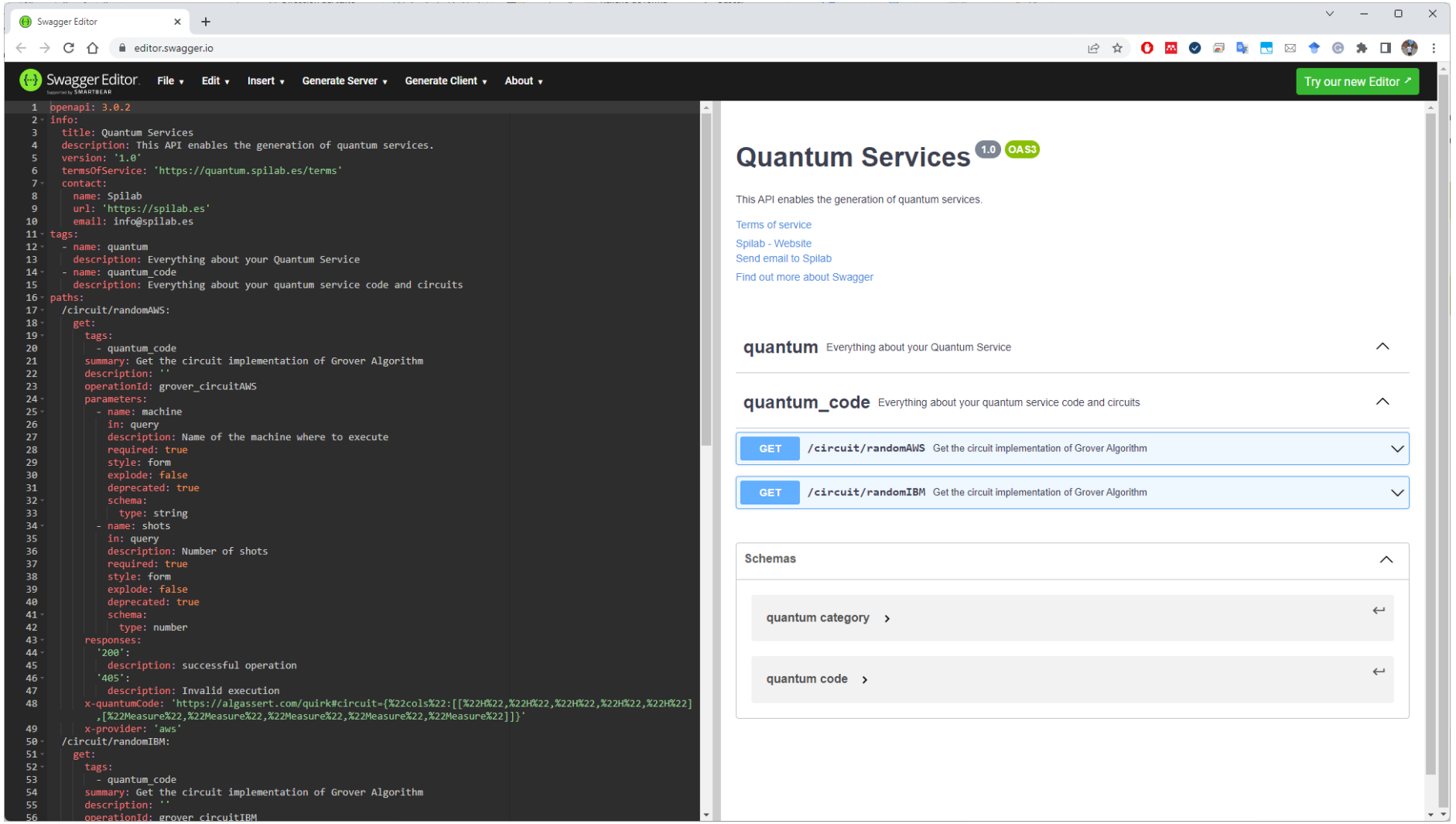
https://online.swagger.io/validator

- **Opensource generator:**
  - skeletons para backends
  - proxies para clientes o front-end

https://openapi-generator.tech/

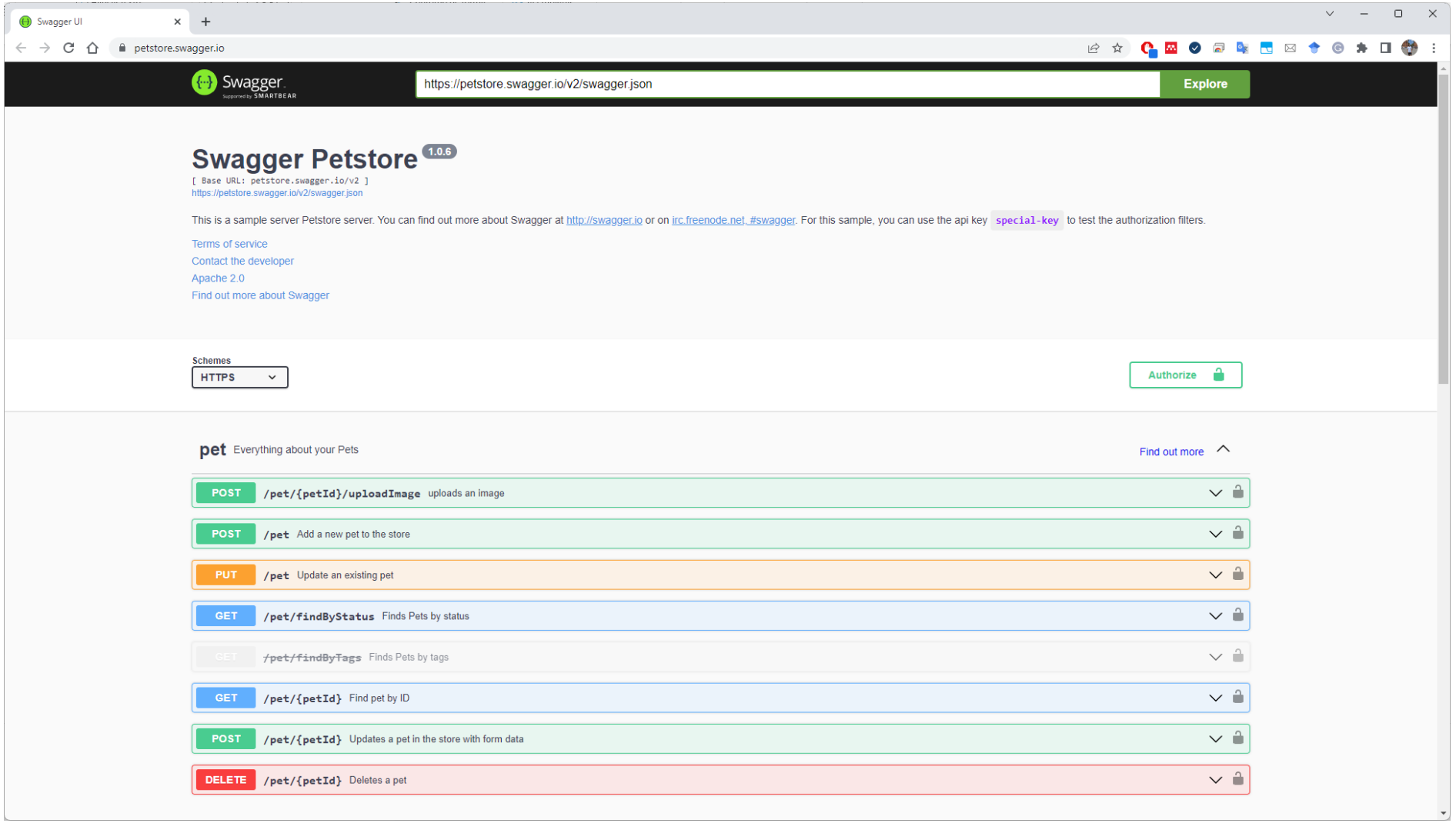
INTRODUCTION

- **Editor:** http://editor.swagger.io

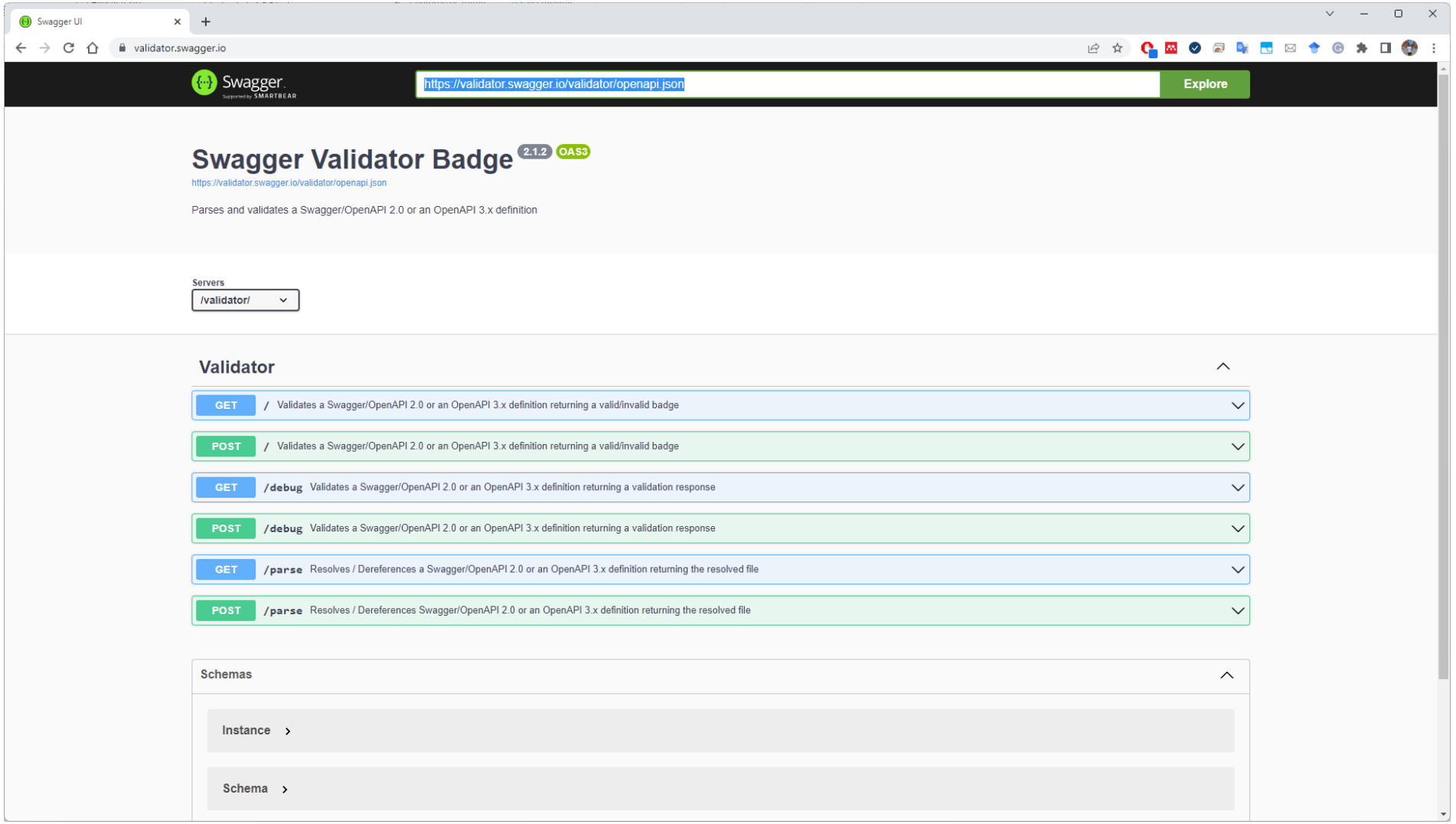


INTRODUCTION

- **API explorer:** http://petstore.swagger.io

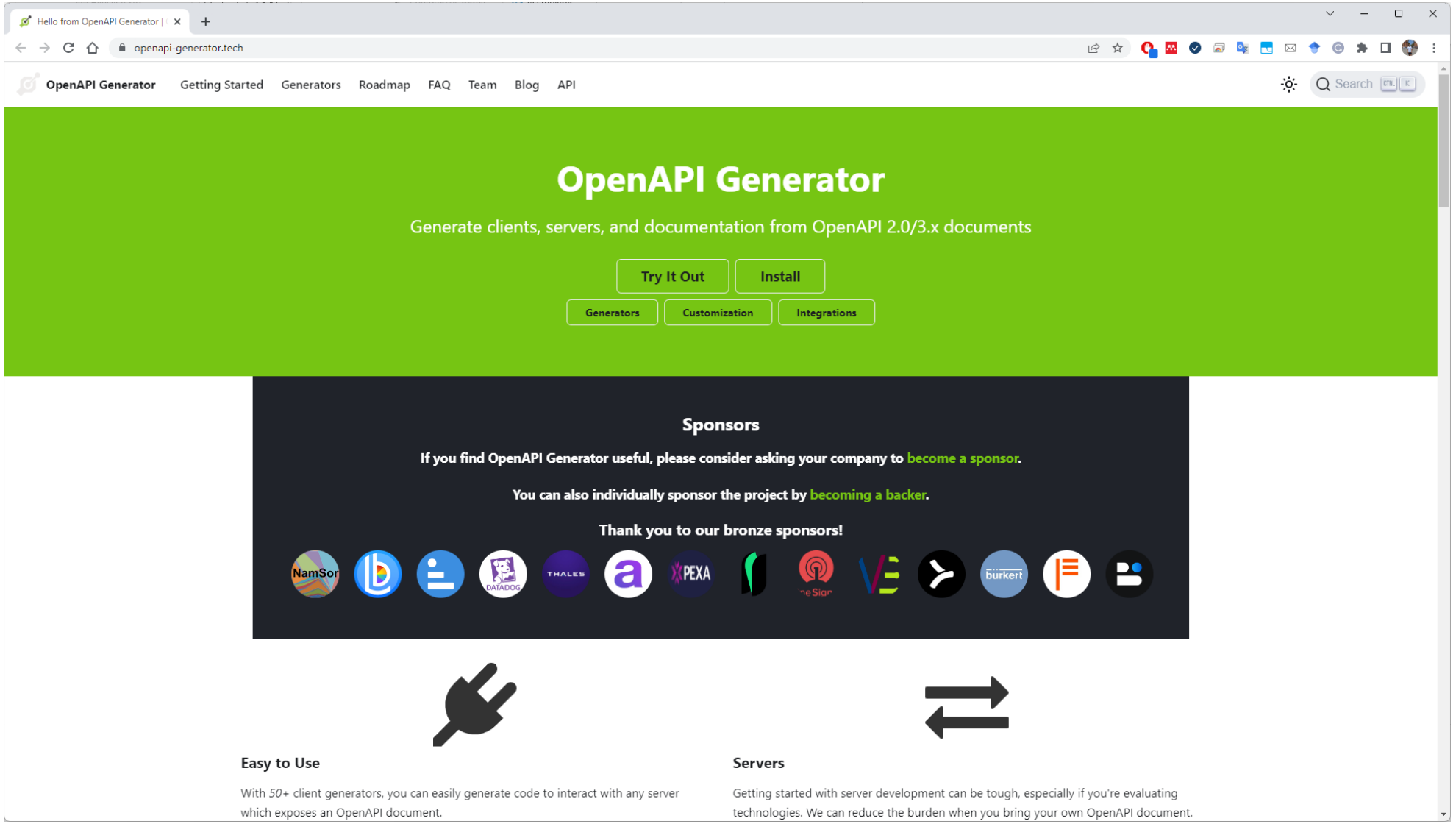


- **Validator:** https://online.swagger.io/validator



INTRODUCTION

- **Opensource generator:** https://openapi-generator.tech/



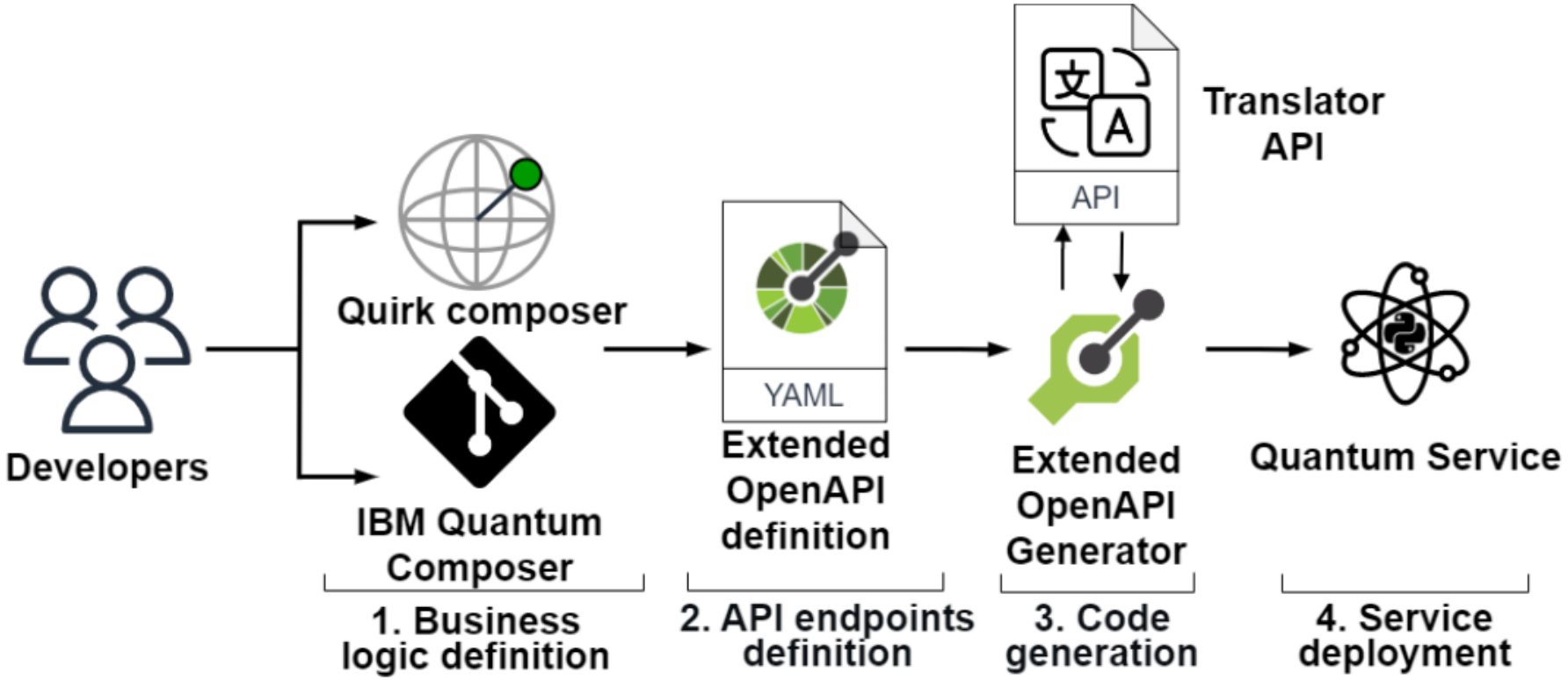
QUANTUM SERVICES GENERATION

To implement a classic service using OpenAPI, a developer needs to combine two main aspects:

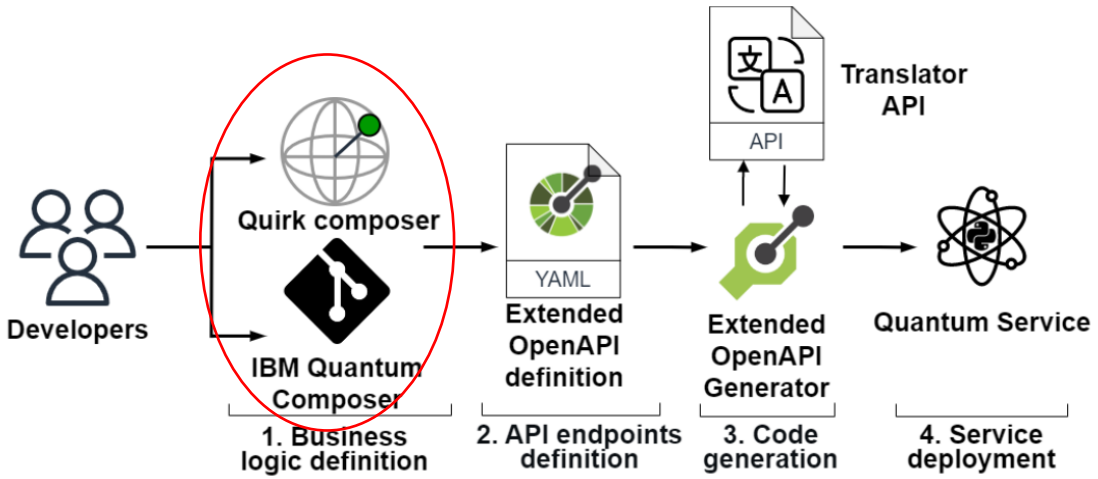
- **The business logic** of the service, which implements the functionality provided by the service.
- **The endpoint** of the service, which determines how an external client can invoke the service.

QUANTUM SERVICES GENERATION

Defining a model for the servitisation of quantum algorithms by extending the OpenAPI specification



QUANTUM SERVICES GENERATION

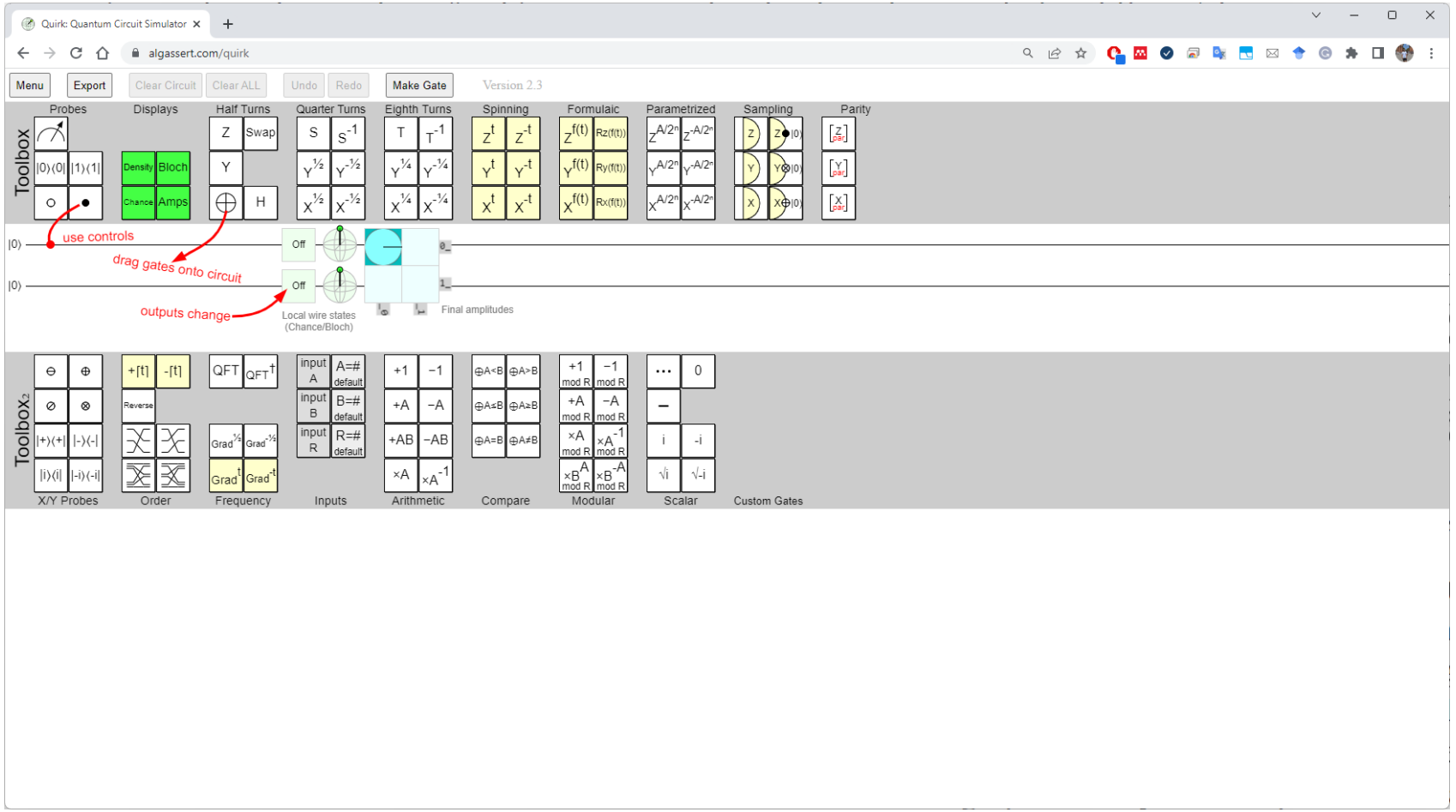


The first step is to define the business logic of the service as a quantum circuit using **Open Quirk**, indicating the Open Quirk link of the created circuit. Or directly indicating the link where is the **source code in Qiskit**

QUANTUM SERVICES GENERATION

OpenAPI for quantum algorithm servitisation

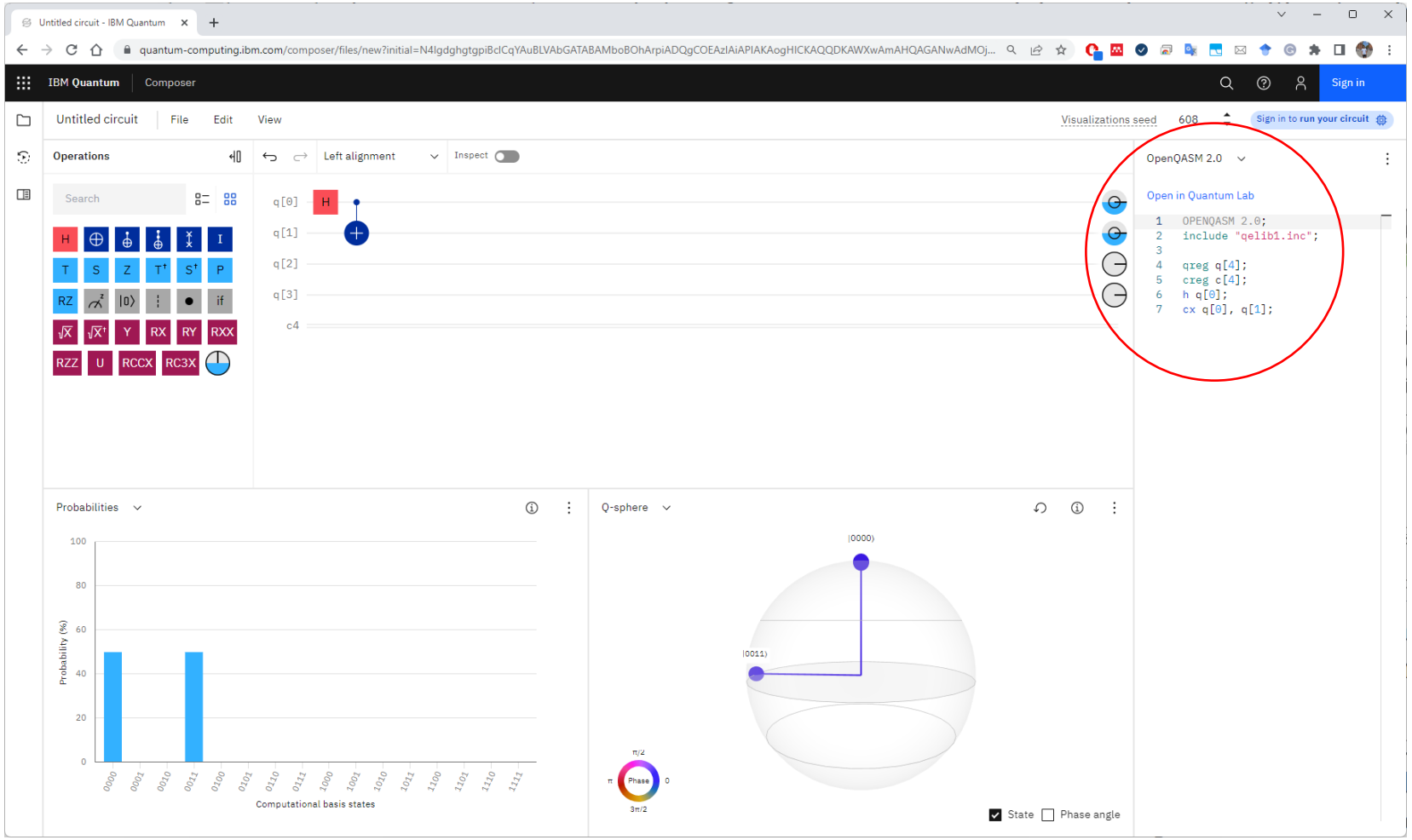
1. Definition of the circuit corresponding to the Quantum Algorithm with Quirk



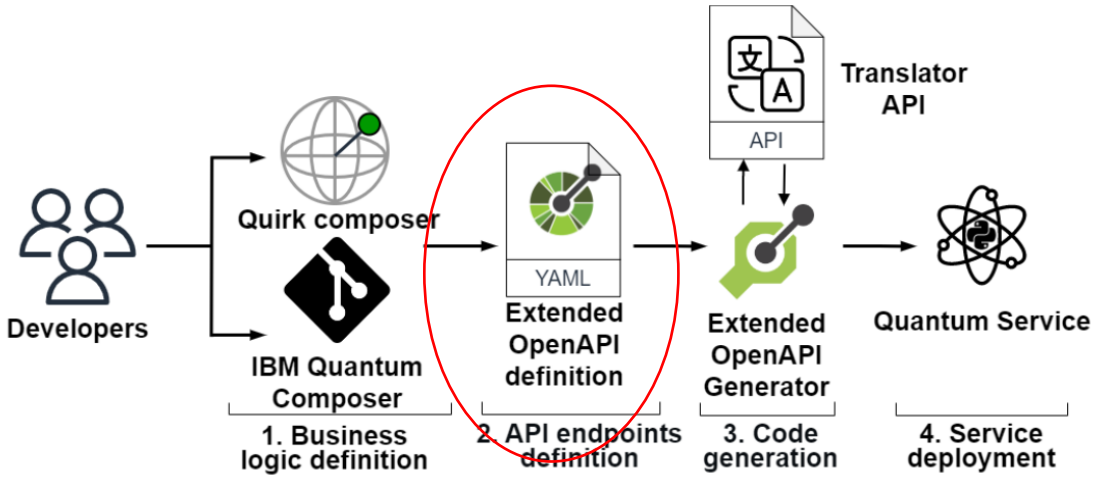
QUANTUM SERVICES GENERATION

OpenAPI for quantum algorithm servitisation

1. Definition of the circuit corresponding to the Quantum Algorithm with IBM Quantum Composer



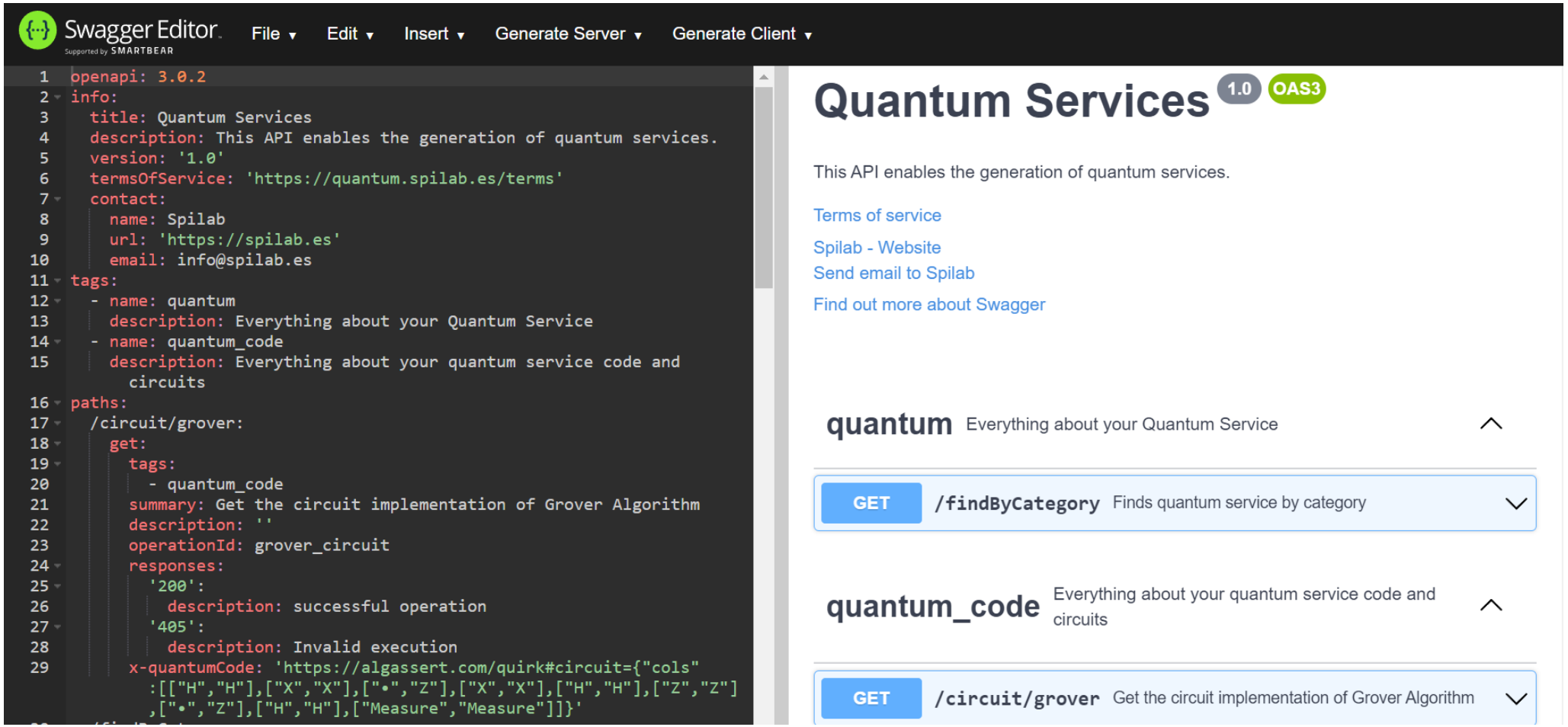
QUANTUM SERVICES GENERATION



In the second step, the quantum API is defined, for which an API contract must be established with OpenAPI.

QUANTUM SERVICES GENERATION

Definition of the API contract, in YAML format, using the Swagger Editor



QUANTUM SERVICES GENERATION

Two custom variables, one to indicate the link to the quantum circuit, and the other to indicate the provider where we want to run it.

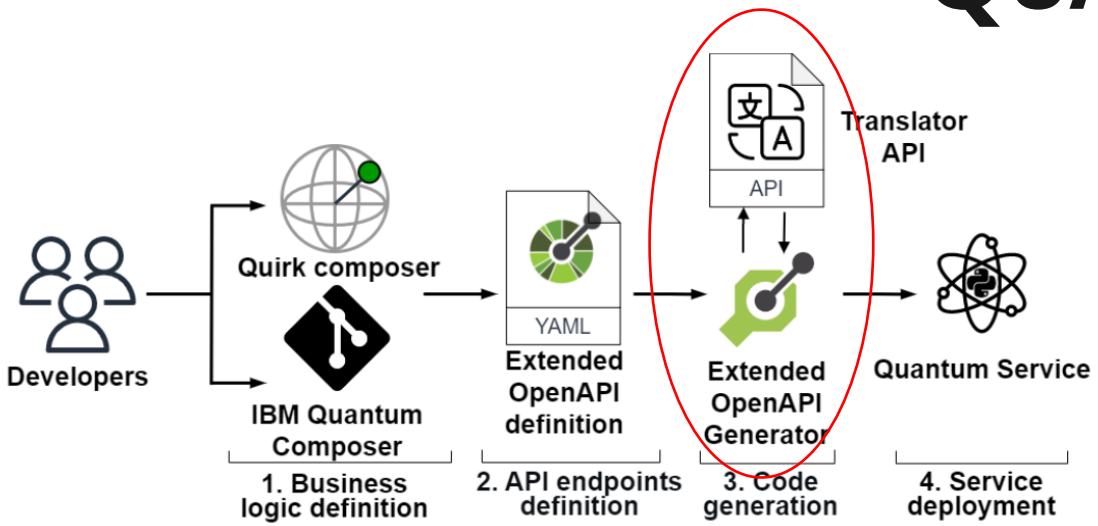
```
37            required: true
38            style: form
39            explode: false
40            deprecated: true
41            schema:
42              type: number
43          responses:
44            '200':
45              description: successful operation
46            '405':
47              description: Invalid execution
48          x-quantumCode: 'QISKIT-URL'
49          x-quantumProvider: 'aws'
50      /circuit/ShorIBMqiskit:
51        get:
52          tags:
```

QUANTUM SERVICES GENERATION

Two custom variables, one to indicate the link to the quantum circuit, and the other to indicate the provider where we want to run it.

```
paths:
  /circuit/randomAWS:
    get:
      tags:
        - quantum_code
      summary: Get the circuit implementation of Grover Algorithm
      description: ''
      operationId: grover_circuitAWS
      parameters:
        - name: machine
        - name: shots
      responses:
        '200':
          description: successful operation
        '405':
          description: Invalid execution
      x-quantumCode: 'QISKIT-URL'
      x-quantumProvider: 'aws'
```

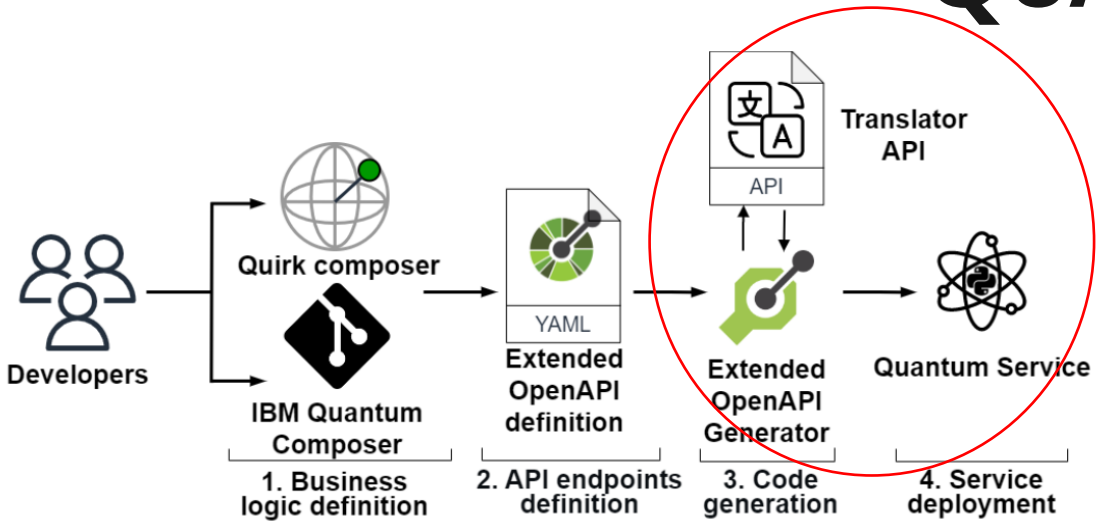
QUANTUM SERVICES GENERATION



In the third step, the extension of the OpenAPI Code Generator is used to generate the source code of the quantum services.

It uses our Translator API to compose the code from the provided quantum circuit for the selected provider.

QUANTUM SERVICES GENERATION



PATH PARAMETERS

*framework string	python-quantum

framework

BODY DATA (required)

parameters

EXAMPLE MODEL application/json

```
{
  "openAPIUrl":
"https://raw.githubusercontent.com\QuantumOpenAPI\main\openapi_quant
um.yaml",


}
```

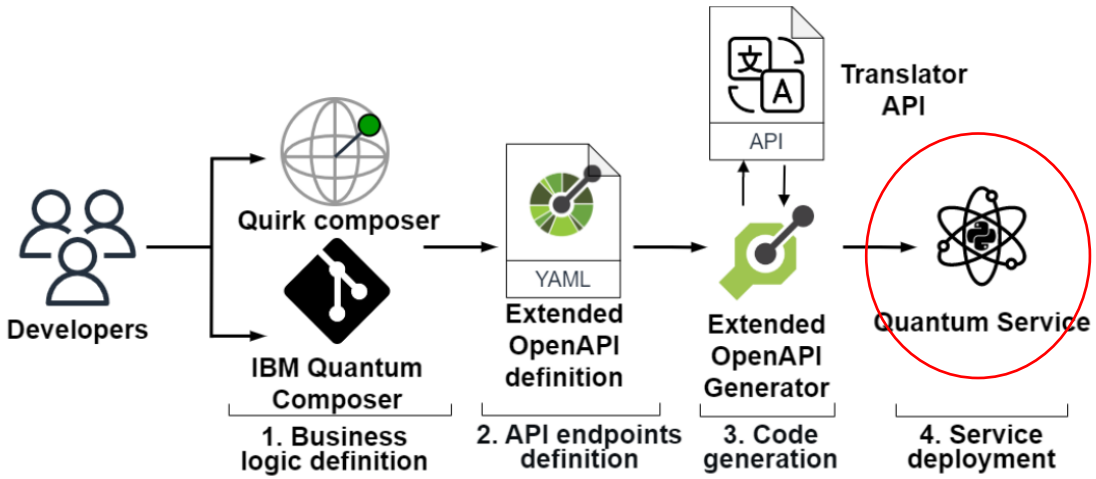
```python
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit, execute,IBMQ

def random_circuit(api_token):  # noqa: E501

    """Get the circuit implementation for random numbers

    # noqa: E501

    :param api_token: API Token
    :type api_token: str

    :rtype: None
    """
    IBMQ.enable_account(api_token)
    provider = IBMQ.get_provider(hub='ibm-q')

    q = QuantumRegister(16,'q')
    c = ClassicalRegister(16,'c')
    circuit = QuantumCircuit(q,c)
    circuit.h(q) # Applies hadamard gate to all qubits
    circuit.measure(q,c) # Measures all qubits

    backend = provider.get_backend('ibmq_qasm_simulator')
    job = execute(circuit, backend, shots=1)

    result = job.result()
    counts = result.get_counts(circuit)

    return counts
```

Qiskit libraries for quantum computing

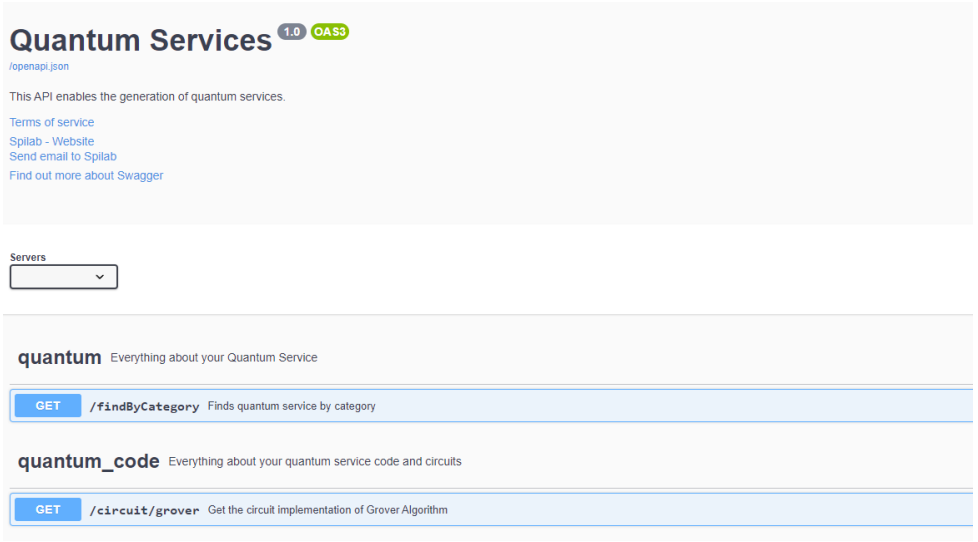
Classical wrapping endpoint service

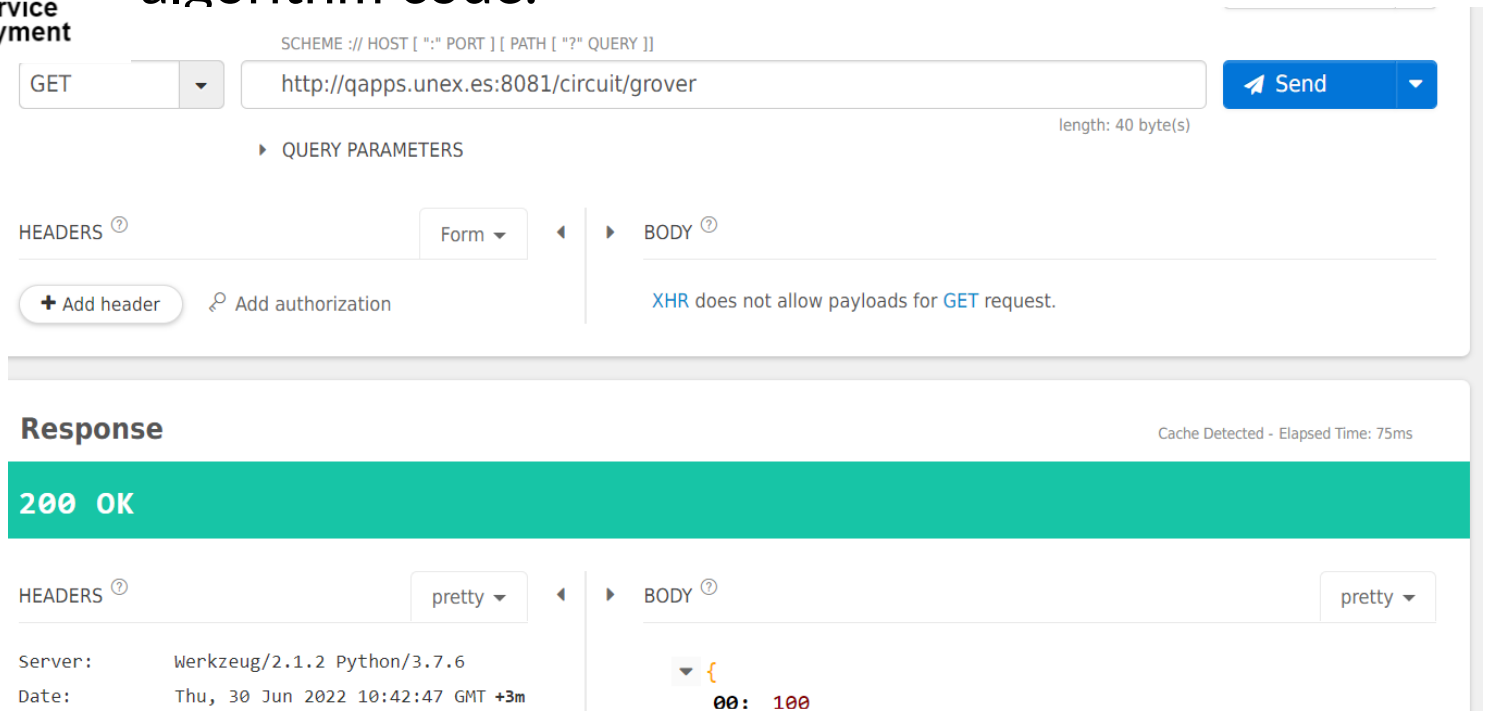
Random Numbers quantum circuit

QUANTUM SERVICES GENERATION

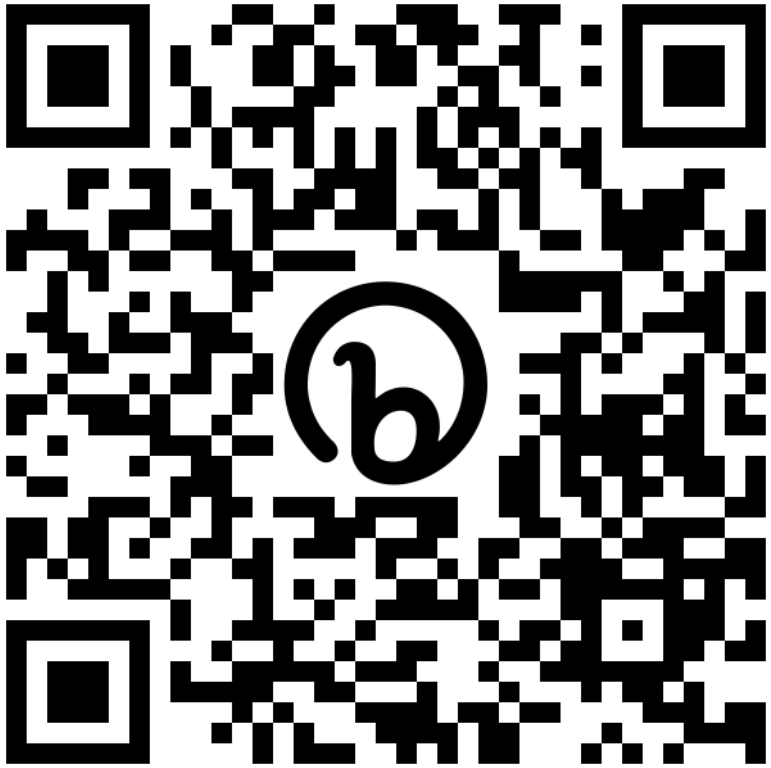


These generated quantum services can be deployed and invoked (fourth step) via RESTful calls to the endpoints encapsulating the quantum algorithm code.





Hands-on!



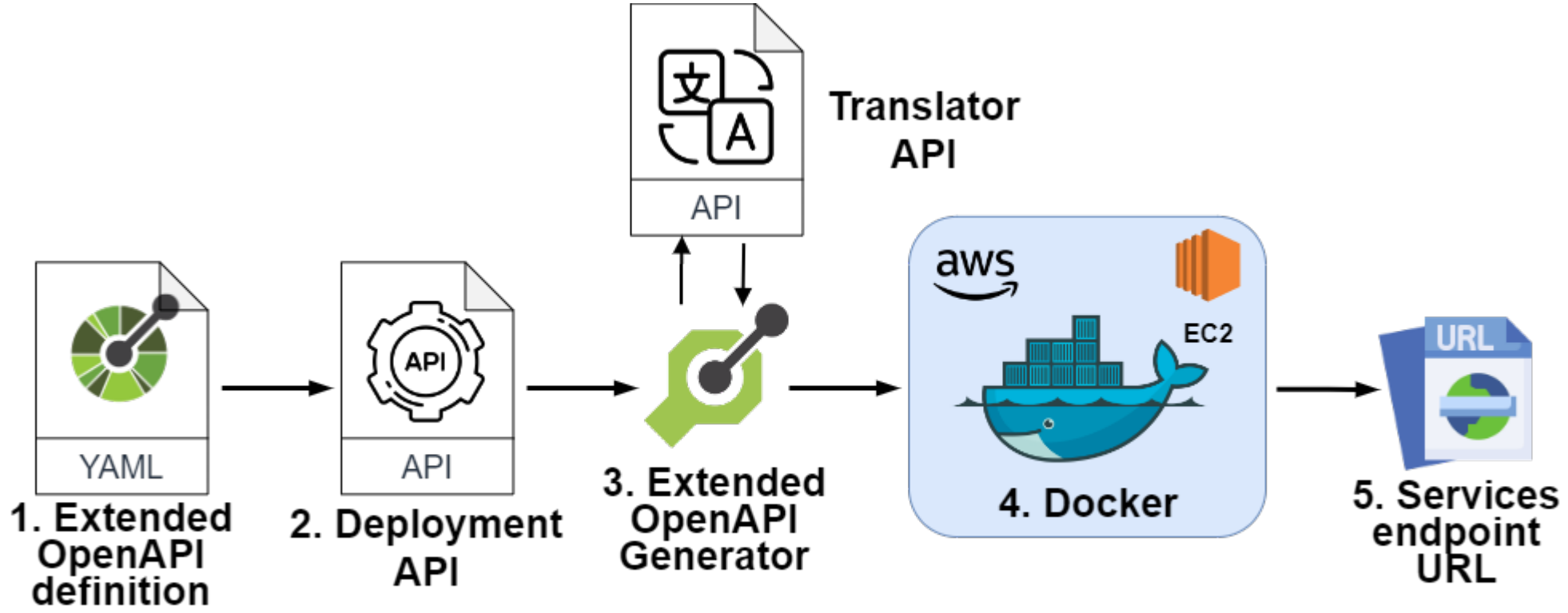
bit.ly/icweQuantumTutorial

Quantum service deployment

Once we have generated quantum services, which can be deployed manually, **why not do something to automate it?**

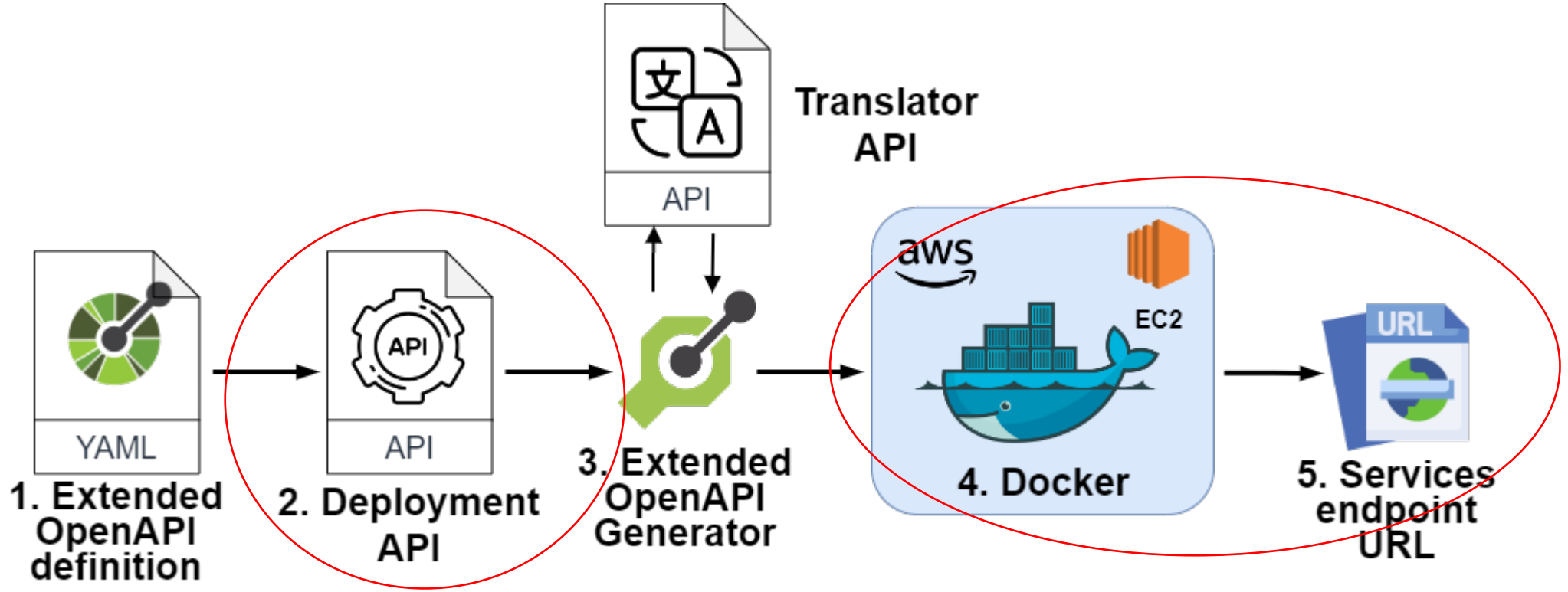
Quantum service deployment

Once we have generated quantum services, which can be deployed manually, why not do something to automate it?

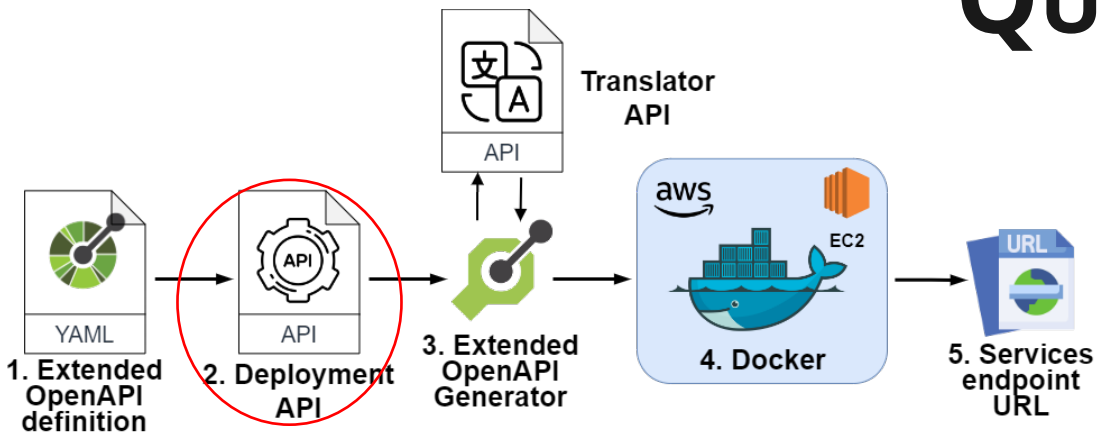


Quantum service deployment

Once we have generated quantum services, which can be deployed manually, why not do something to automate it?

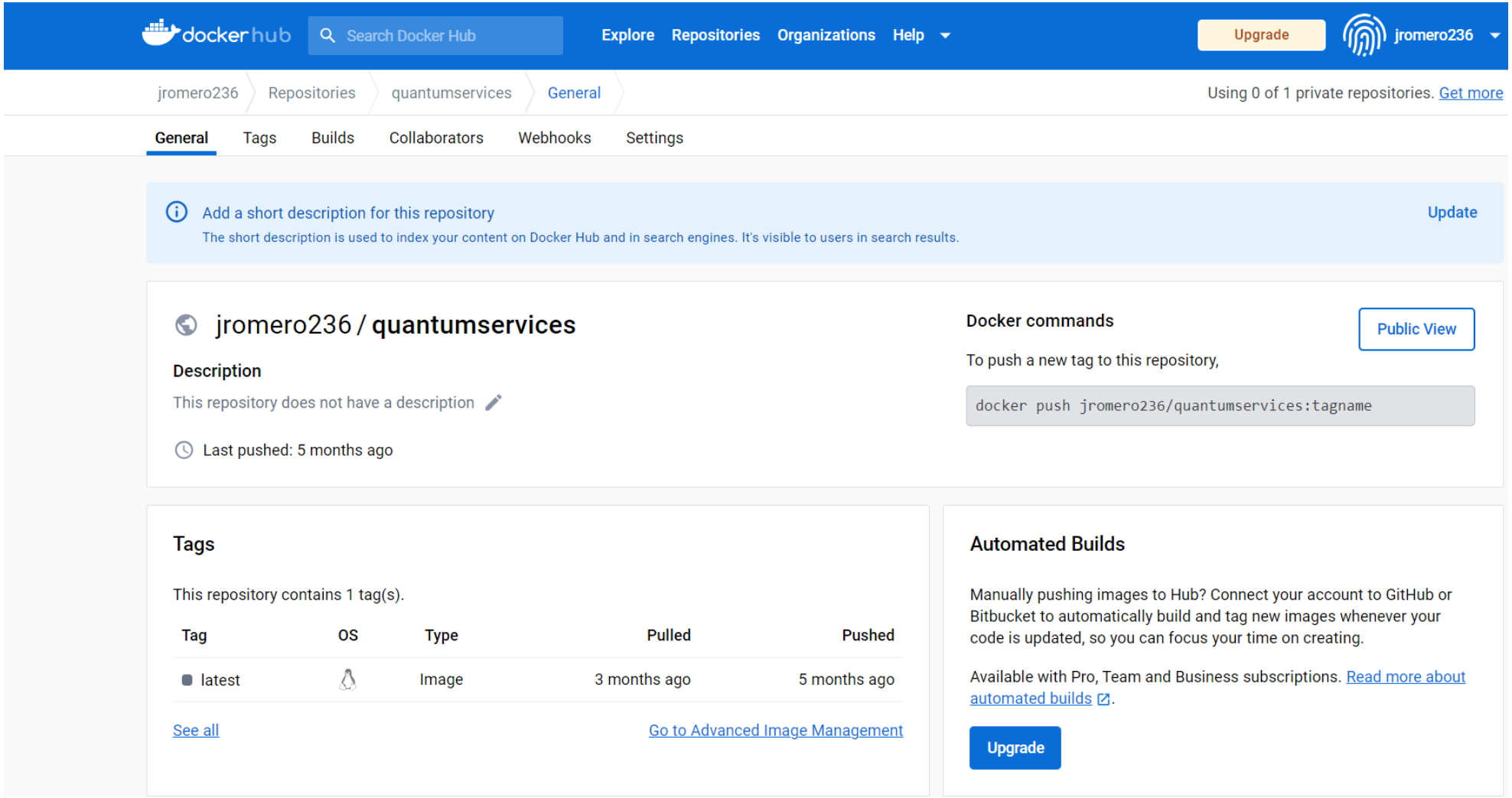


QUANTUM SERVICE DEPLOYMENT

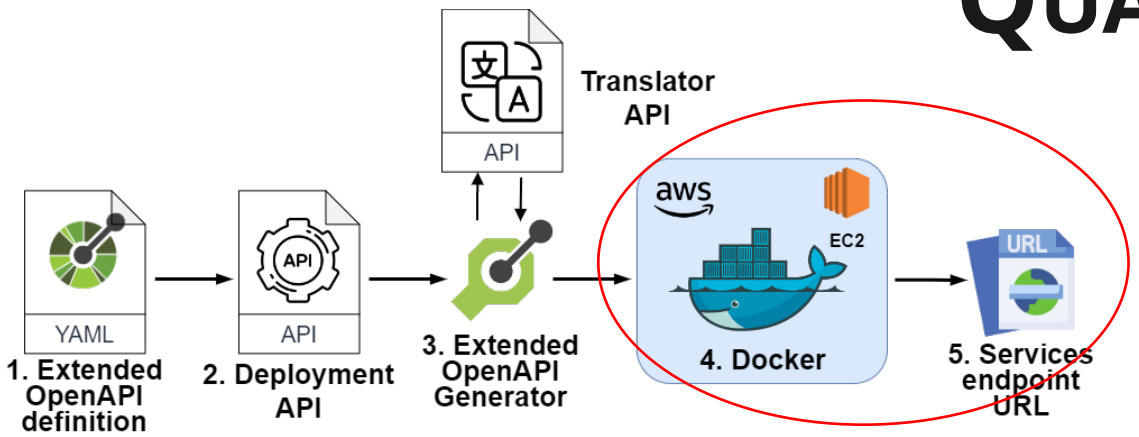


The Deployment API receives the YAML file based on the modification of the previous OpenAPI specification and the necessary parameters for the configuration of the providers (e.g. AWS keys).

QUANTUM SERVICE DEPLOYMENT



QUANTUM SERVICE DEPLOYMENT



Body Cookies Headers (5) Test Results Status: 200 OK Time: 8.62 s Size: 304 B Save Response

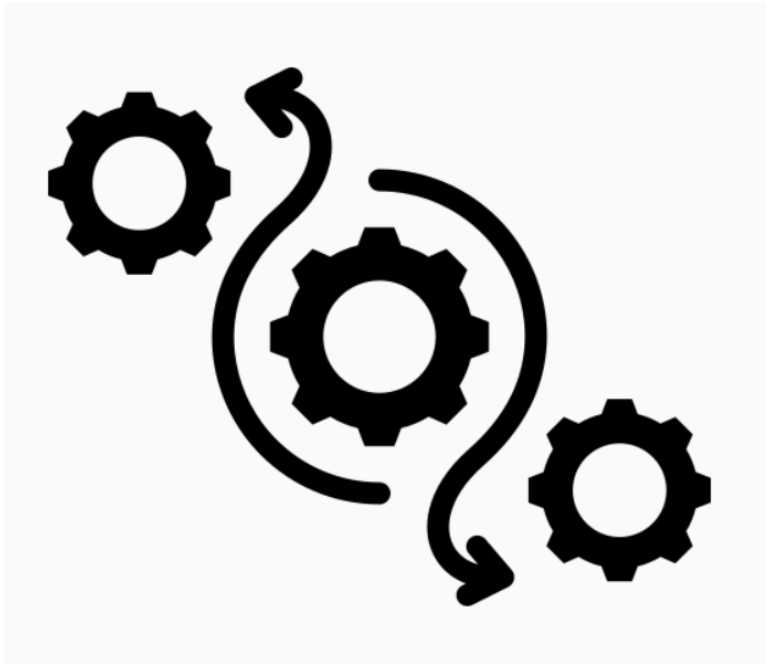
Pretty Raw Preview Visualize HTML

```
1  <html>
2  <h1 href='http://quantumservicesdeployment.spilab.es:8088/ui'>http://quantumservicesdeployment.spilab.es:8088/ui<\h1>
3      <\html>
```

Hands-on!

Workflow for the continuous deployment

Why not bring **generation** and **deployment** together and **automate** it?



WORKFLOW FOR THE CONTINUOUS DEPLOYMENT

GitHub Actions

1.Automation: GitHub Actions allows you to automate workflows in software development projects.

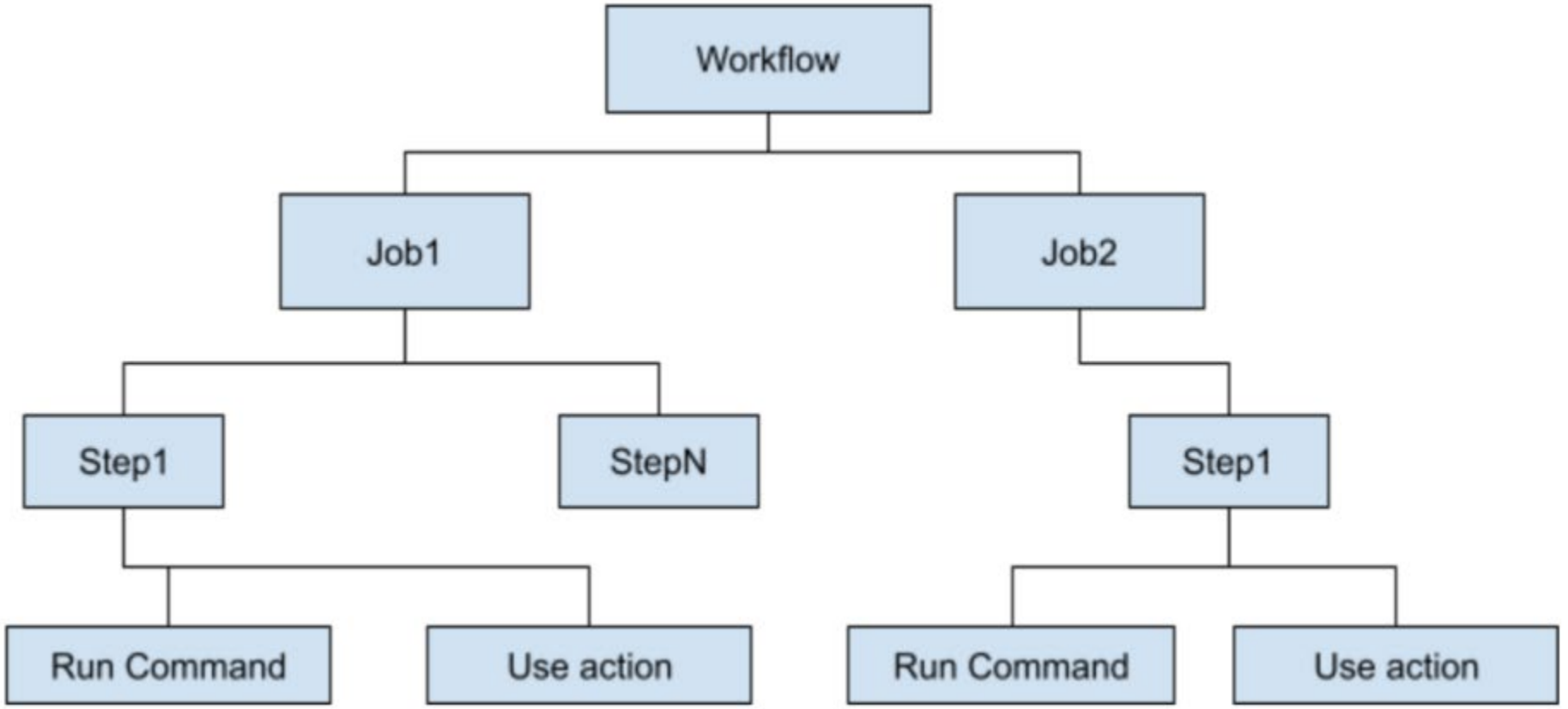
2.YAML-based Configuration: Workflows are defined using YAML configuration files stored in the repository.

3.Event-Driven: Workflows can be triggered by events like code pushes, pull requests, or scheduled intervals.

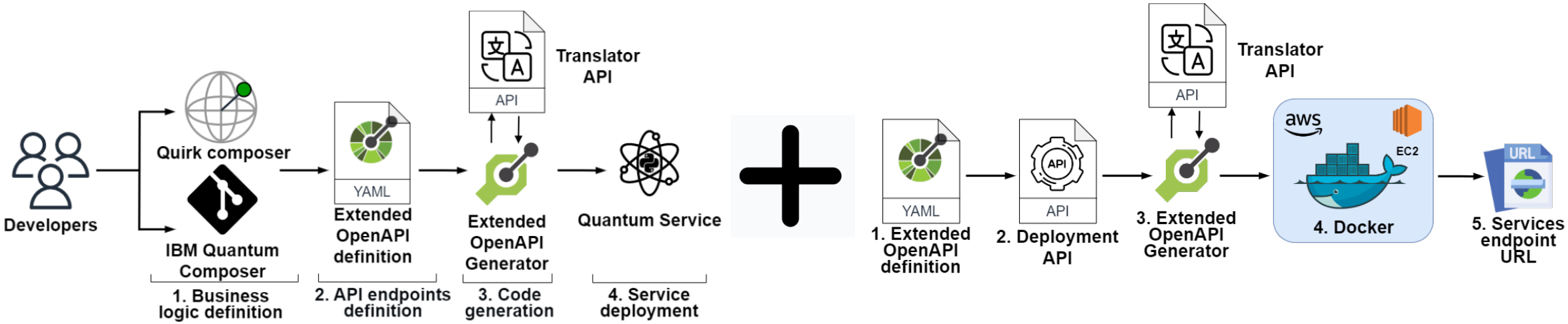
4.Integrations: GitHub Actions seamlessly integrates with other GitHub features, enabling collaboration and streamlining the development process.

Workflow for the continuous deployment

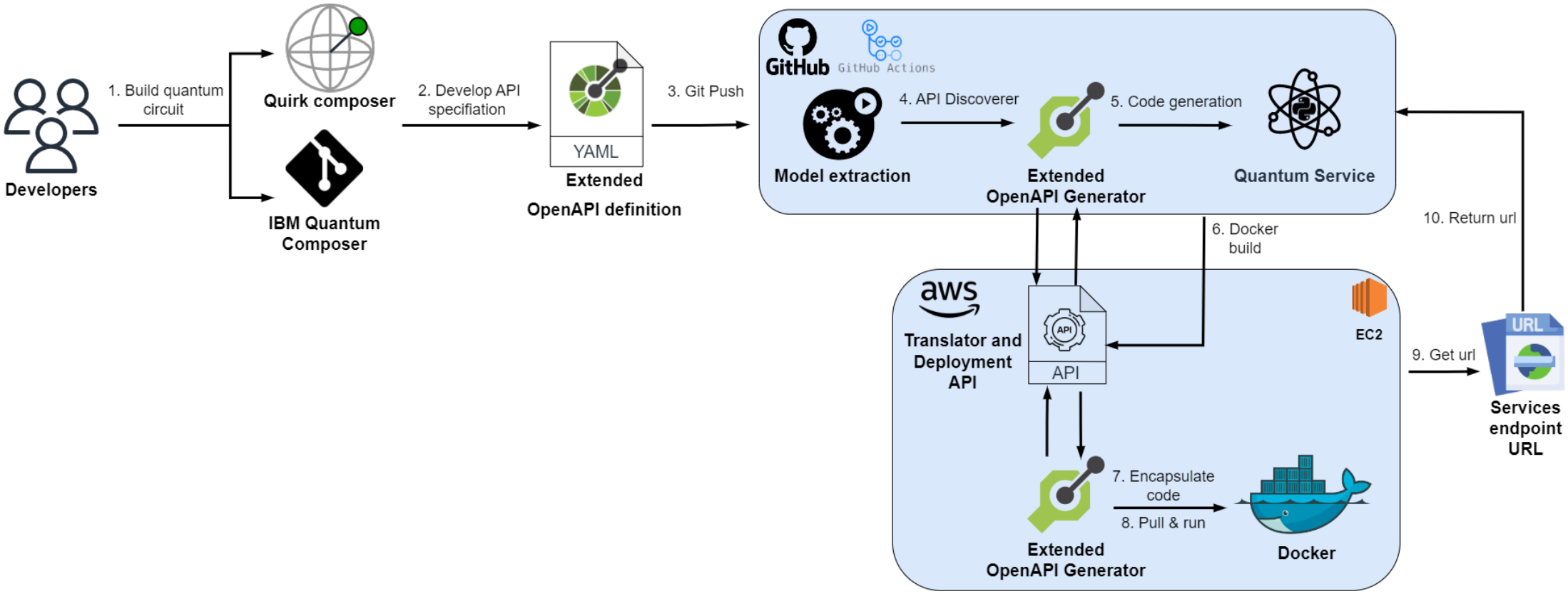
GitHub Actions



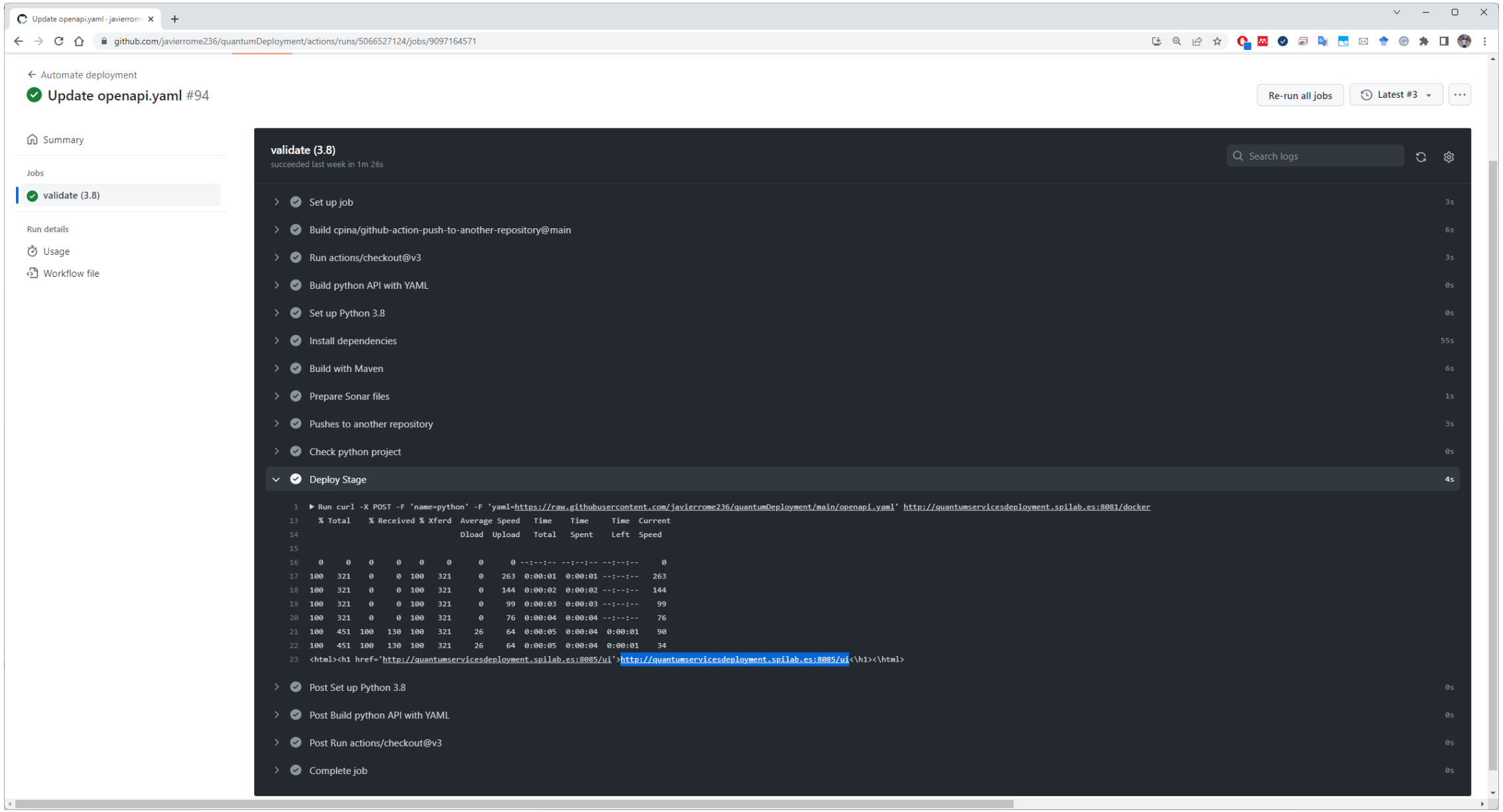
WORKFLOW FOR THE CONTINUOUS DEPLOYMENT



Workflow for the continuous deployment



WORKFLOW FOR THE CONTINUOUS DEPLOYMENT



Hands-on!

Please fill out the following survey

It will be of great help to us!!

https://forms.office.com/e/sQgqTi1LQd

THANKS FOR ATTENTION!

23rd International Conference on Web Engineering - ICWE

Alicante (Spain)

Javier Romero-Álvarez
jromero@unex.es

Jaime Alvarado-Valiente
jaimeav@unex.es

MINISTERIO DE CIENCIA E INNOVACIÓN
UNIÓN EUROPEA
FONDO EUROPEO DE DESARROLLO REGIONAL
AGENCIA ESTATAL DE INVESTIGACIÓN
JUNTA DE EXTREMADURA
Consejería de Economía, Ciencia y Agenda Digital
SPILab by Quercus SEG
Software Engineering Group
QUERCUS
UNIVERSIDAD DE EXTREMADURA