

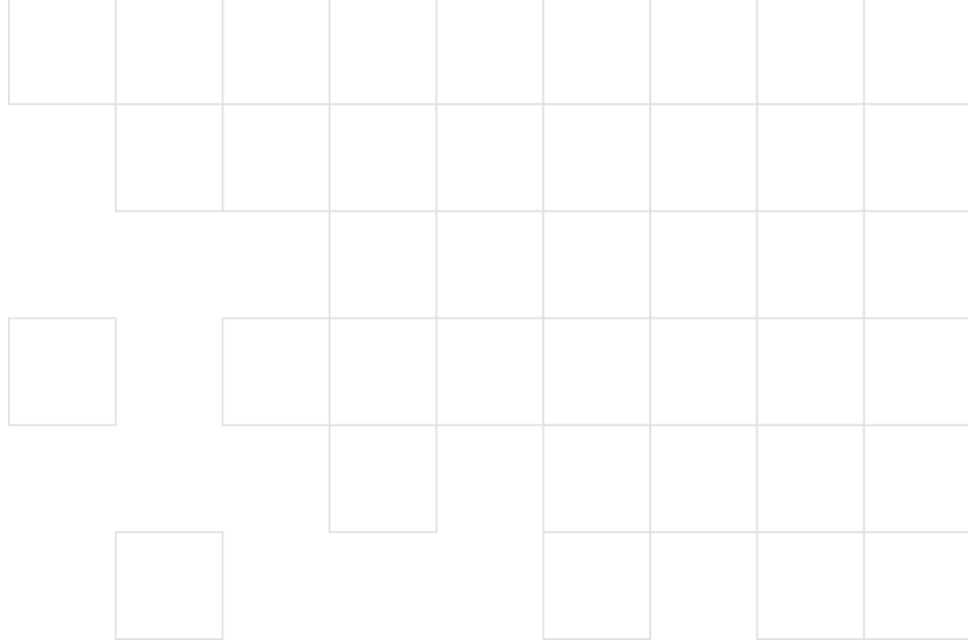
Qiskit Metal to cloud-ready

Marco Facchini @marcolincs

Scott Wyman Neagle @scottwn

Dayeong Kang @tula3and

qamp-fall-21 #16



The goal:
provide
the qiskit-metal tool
as a service

1. Execute qiskit-metal in back-end
2. Visualize the results from back-end to front-end (with #15)

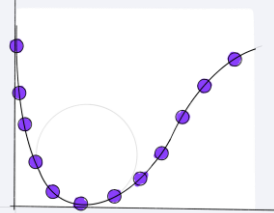
In back-end:

1. Get an input file from FE
2. Execute the input and make a result file
3. Send the result to FE

In front-end:

1. Make an input file with the user-interactive tool
2. Get a result file from BE and show in the webpage

In front-end:



Between FE and BE:



In back-end:

```

from qiskit import QuantumCircuit, execute
from qiskit import Aer, IBMQ
from qiskit.providers.aer.noise import NoiseModel

# Choose a real device to simulate from IBMQ provider
backend = IBMQ.get_backend('ibmq_16melbourne')
noise_model = NoiseModel.from_backend(backend)
coupling_map = backend.configuration().coupling_map

# Generate an Aer noise model for device
noise_model = NoiseModel.from_backend(backend)
backend.noise_model = noise_model

# Generate 3-qubit QMC state
num_qubits = 3
circuit = QuantumCircuit(num_qubits, 3)
circuit.h(0)
circuit.cnot(0, 1)
circuit.cnot(1, 2)
circuit.measure([0, 1, 2], [0, 1, 2])

# Perform noisy simulation
backend = Aer.get_backend('aer_simulator')
job = execute(circuit, backend,
              optimization_level=0,
              noise_model=noise_model,
              result_callback=print)
result = job.result()
print(result.get_counts(0))

```

Scott Wyman Neagle @scottwn

Dayeong Kang @tula3and



Thank you

Marco Facchini @marcolincs
Scott Wyman Neagle @scottwn
Dayeong Kang @tula3and

