Customized user metrics

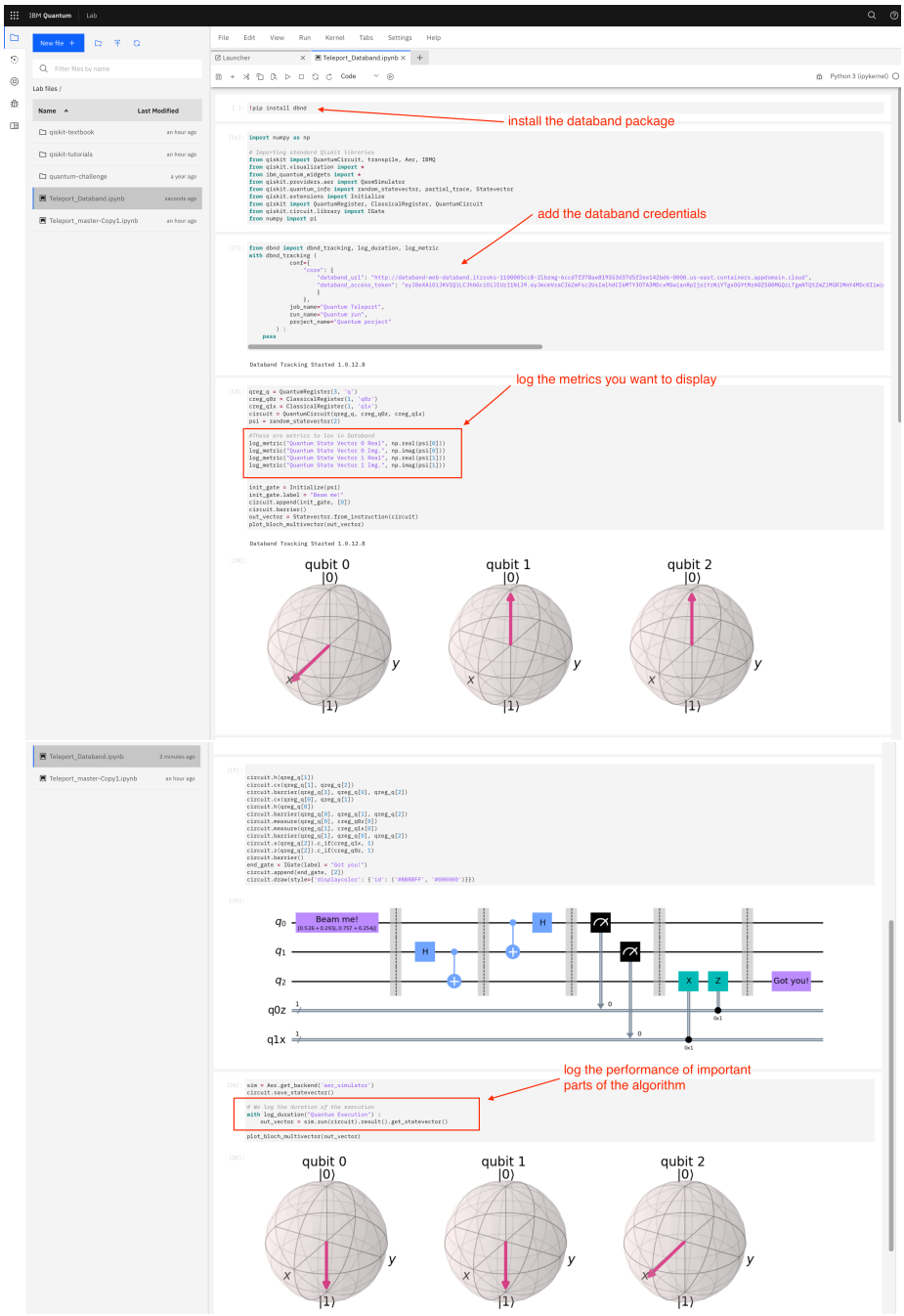
Exemplified with Quantum Computing algorithms

This chapter will show one example of monitoring the performance data of any python program, with independence of it was originally designed as a data pipeline or perhaps for quite different purpose that might need to be analyzed in the context of a data preparation project. It is not necessary to install or setup any of the technical resources we will see below to understand the contents of this chapter. Consequently, this section will not necessarily be a hands-on exercise.

The use case that we will inspect is one of the most basic algorithms in quantum computing: The Quantum Teleportation. For this workshop, the only thing we need to know about it is that working with quantum computing implies, in the vast mayority of the current use cases, to code an algorithm in Python and, optionally for educational purposes, in a Jupyter environment like the IBM Quantum Lab.

1. Enable the monitoring of an algorithm

The only thing that we need to do is to add some lines of code in the quantum algorithm as described in the Databand documentation and more specifically here and here. The following screenshots are taken in the IBM Quantum Lab environment and indicate the sections in the code that we want to track and the databand api calls that we need to embed.



As you can see, we have simply used the functions:

- `dbnd_tracking` to enable the tracking

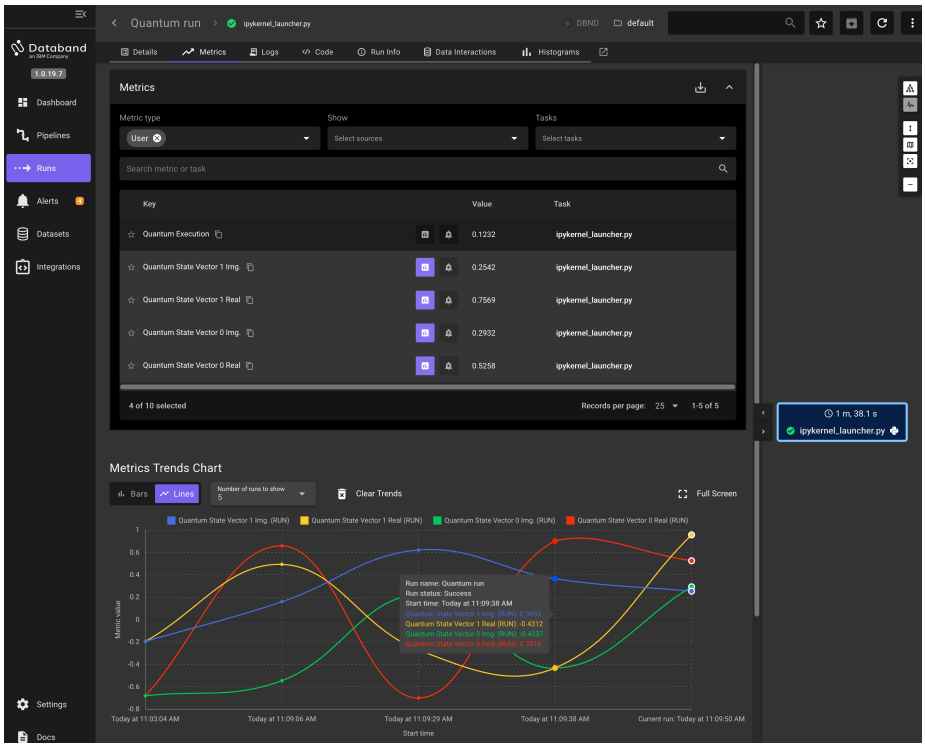
- `log_metric` to track variable parameters of the algorithm
- `log_duration` to track the elapsed time of the algorithm execution

Don't forget to add your credentials in the pipeline definition header, like we did in the chapter 9 Python pipelines

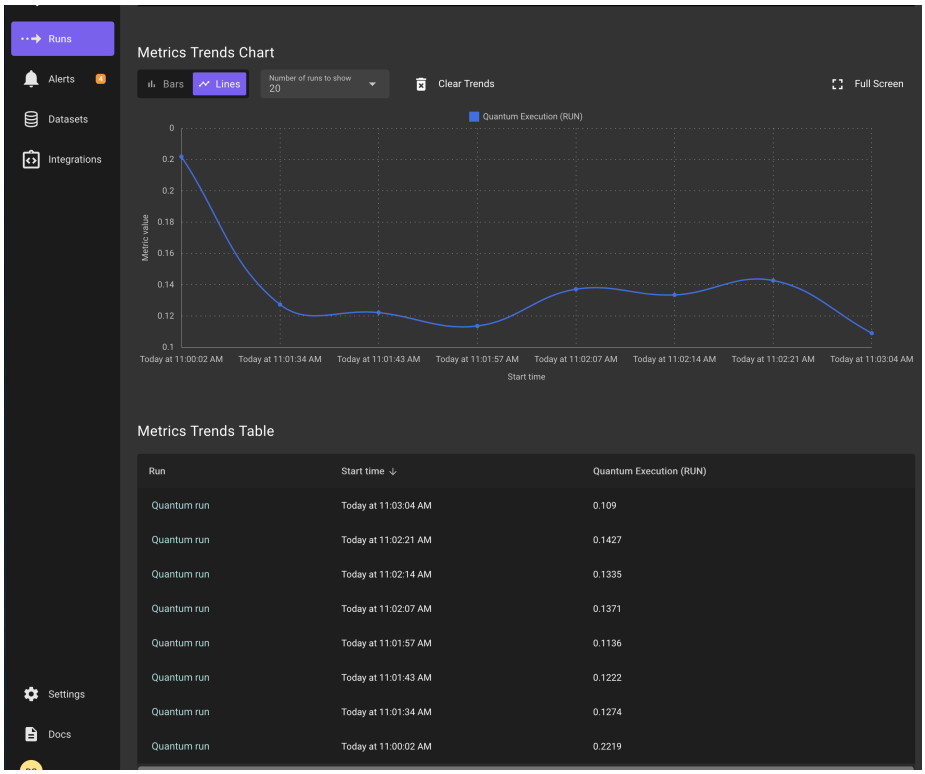
2. Observe the data collected

The metrics of our algorithm will be collected and displayed in the Metrics section of each run. Remember to select user metric types (default choice) and click on the purple icons of the metrics you want to be added to the graph.

The variabiliy of the parameters in the algorithm execution will be displayed this way:



Analogously, the following graph is an indication of the execution time of the algorithm. Note that it is not the full execution time of the python program, but only the elapsed time that the quantum computer (actually, the simulator) was occupied executing the quantum circuit (the materialization of our algorithm)



Previous Section: Alerts and exceptions

Return to main