量子计算实验室产品手册

概述

量子计算

量子计算是一门前沿的计算领域，它利用量子力学原理来处理信息和进行计算，与经典计算相比具有独特的优势和潜力。在量子计算中，信息以量子比特的形式存储和处理，这些量子比特可以处于叠加态，即同时具有多种可能的状态。这种叠加态的特性使得量子计算具有并行计算的能力，可以在某种程度上同时处理多个计算任务，从而大幅提高计算效率。

除了叠加态，量子计算还利用了量子纠缠的特性。量子纠缠是一种量子系统中量子态之间的特殊关联，其中一个量子系统的状态无法独立描述，必须考虑整体系统的状态。这种量子纠缠的现象在量子计算中被广泛应用，可以实现量子比特之间的信息传输和量子操作，为量子计算提供了更为强大的计算能力。

另一个量子计算的重要特征是可逆性。与经典计算中的信息丢失不同，量子计算过程中的操作，除了测量之外都需要是可逆的，即可以通过逆操作将系统恢复到初始状态。

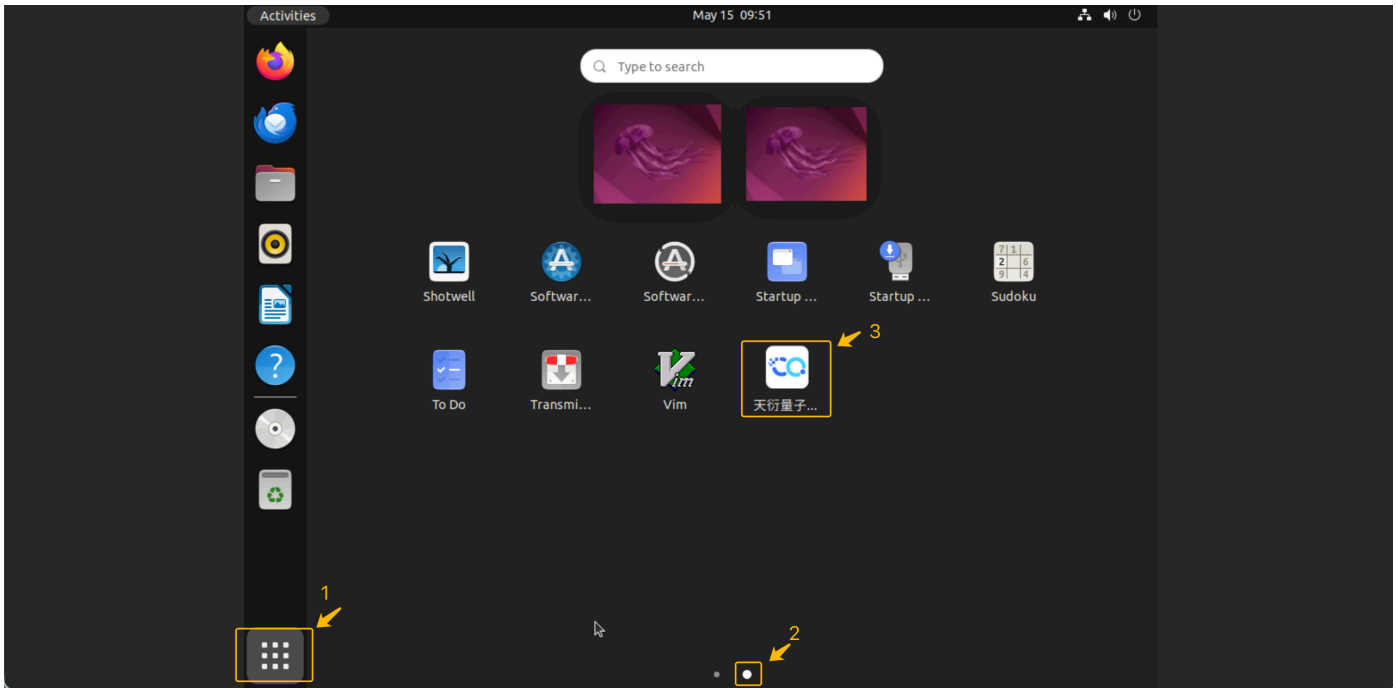
量子计算云实验室

量子计算云平台实验室面向量子计算开发者提供计算资源展示、图形化编程、在线编程、超量融合任务创建、任务管理等功能，支持通过拖拽搭建量子线路，实现量子编程可视化，大大降低量子编程门槛。同时，在线编程工具引入Cqlib编程框架，为专业开发者提供一个灵活便捷的开发环境。

启动开发机并登录VNC远程桌面

步骤1： 完成购买后，待实例状态为“可用”时，点击“登录”进入登录云主机图形化界面

步骤2： 点击【打开】跳转到开发机，成功进入到VNC远程桌面页面中，如下图所示，可看见在桌面中有"天衍量子计算实验室"应用图标；



步骤3： 点击"天衍量子计算实验室"应用图标后，进入天衍量子计算实验室功能。

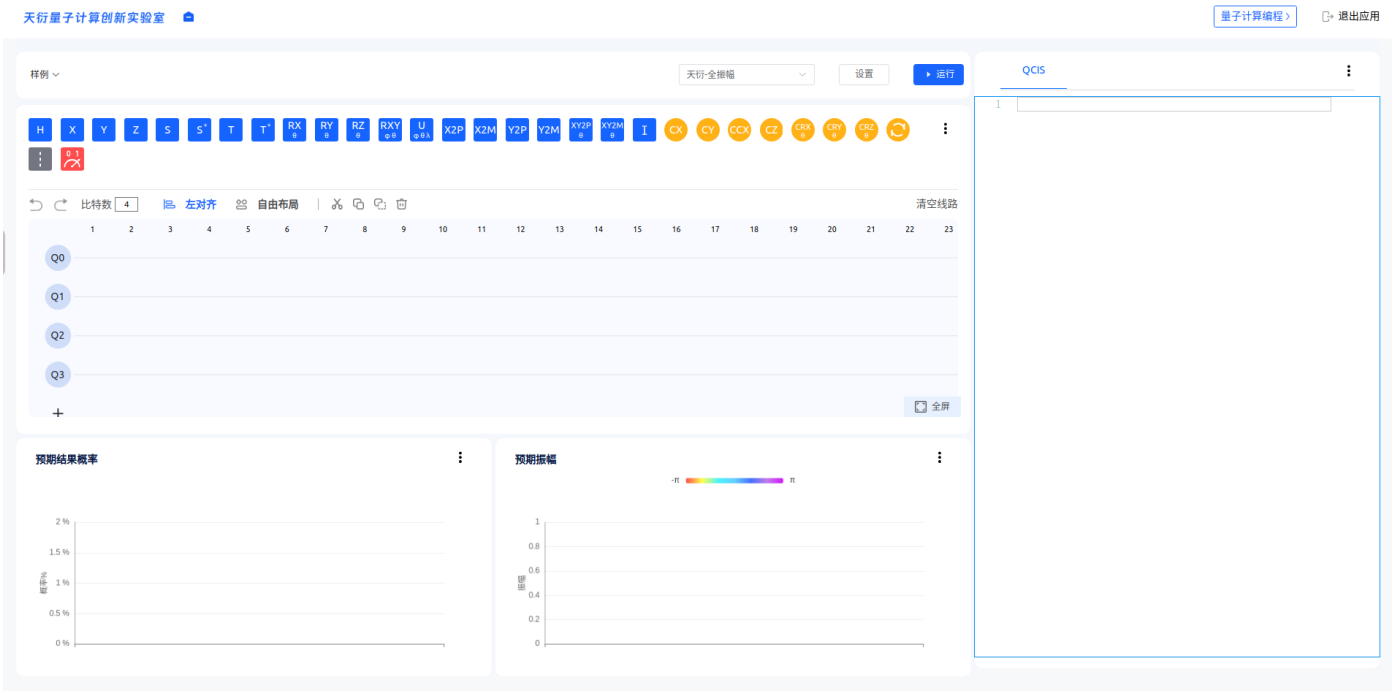
如需返回VNC桌面、点击右上角"退出应用"按钮、即可退出应用返回桌面。

开发机中使用量子开发机进行图形化编程

图形化编程，依托天衍量子计算技术积淀，以可视化拖拽方式开发，降低编码门槛，适配多场景快速开发，融合可视化交互与底层算力，赋能开发者高效构建各类应用。

在天衍量子计算实验室功能首页、点击"图形化编程"链接，

进入图形化编程。

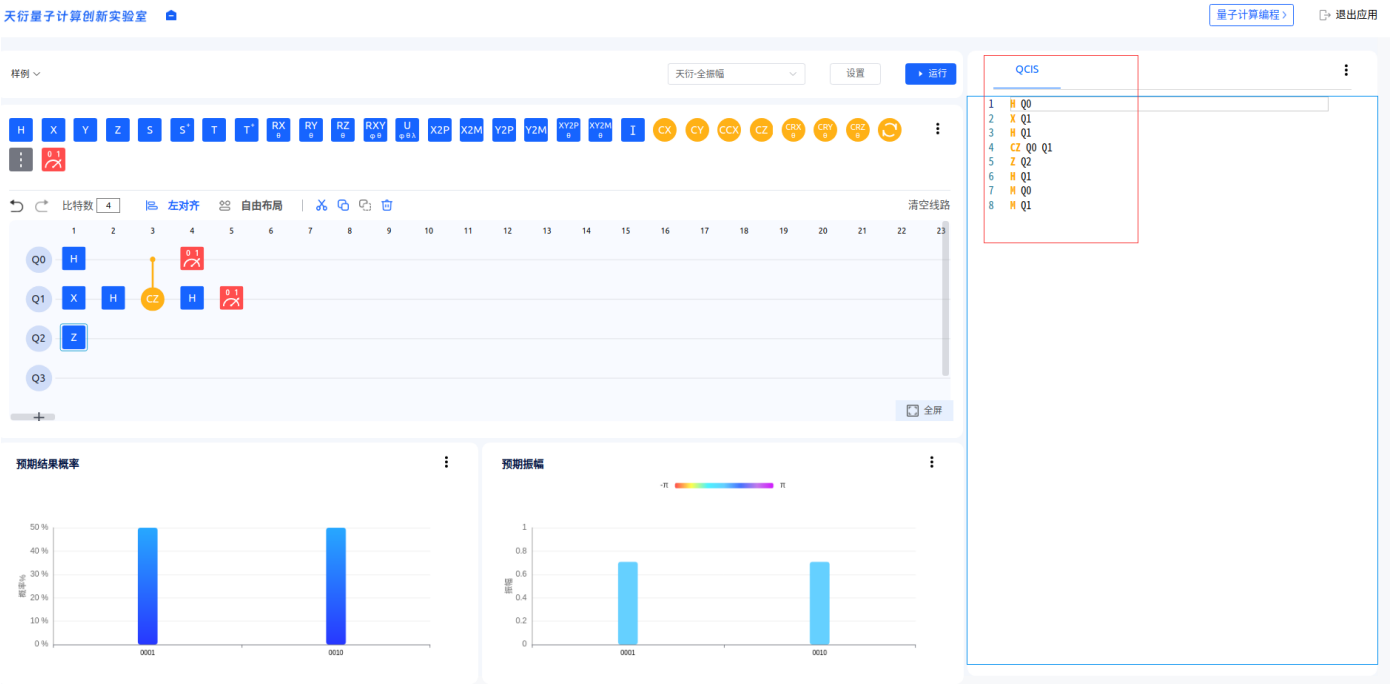


可以在"样例"下拉框中、选择不同类型的样例进行实验。

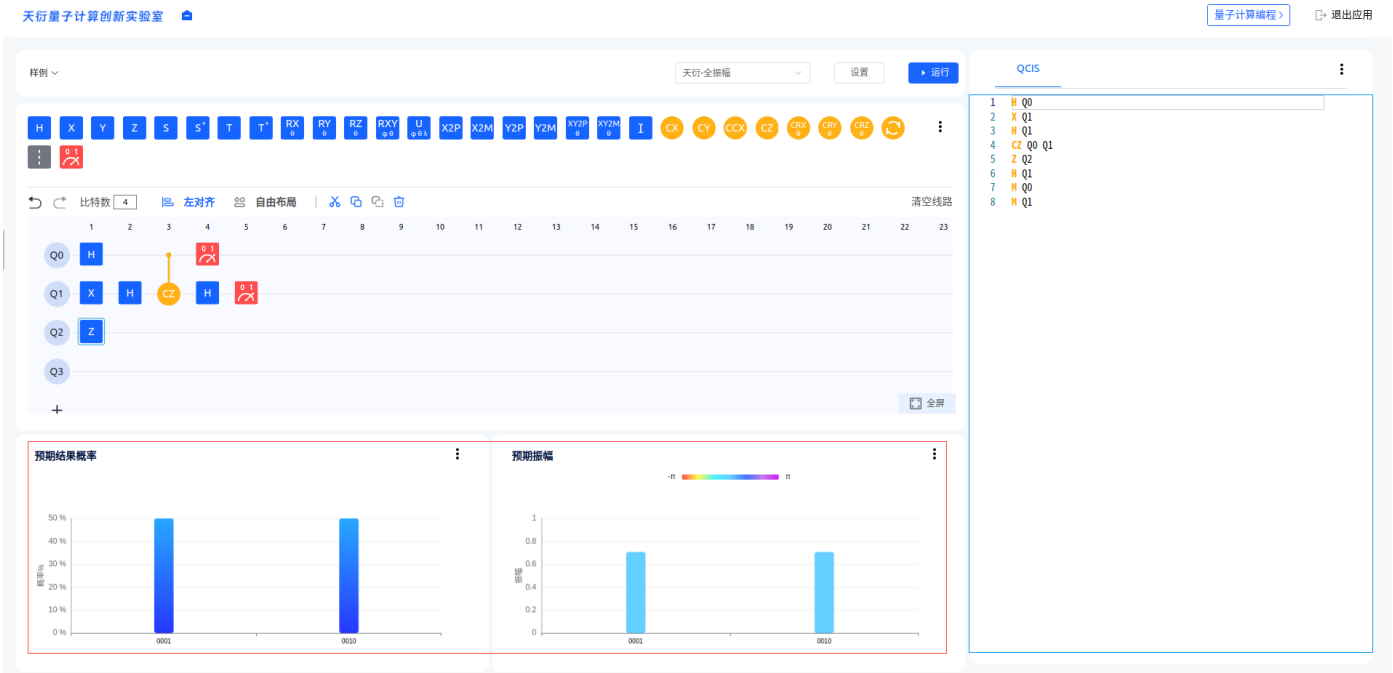
新建的线路默认为4个量子比特，点击左侧量子比特下方加号可增加量子比特；鼠标悬浮在量子比特上，出现删除图标可点击减少量子比特。

拖拽量子门到对应比特的画布线路上。

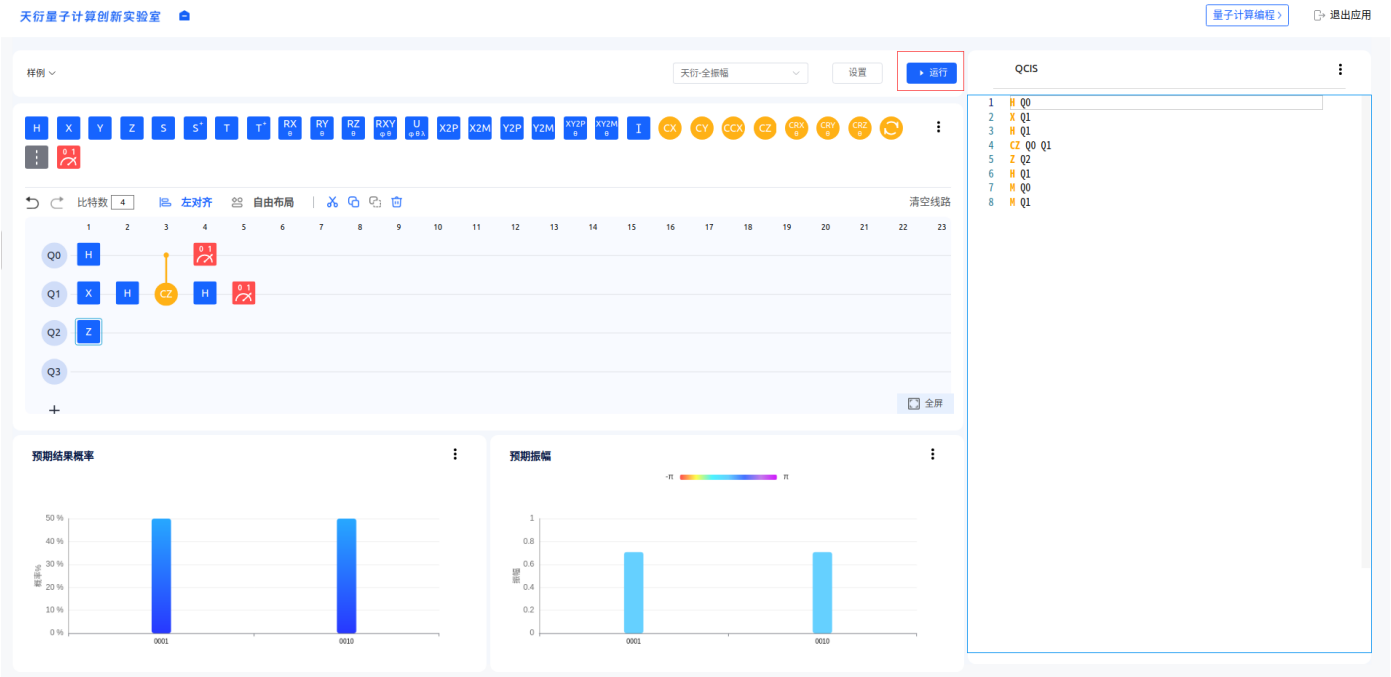
左边图形拖拽完成，右边动态实时生成对应的QCIS指令集。



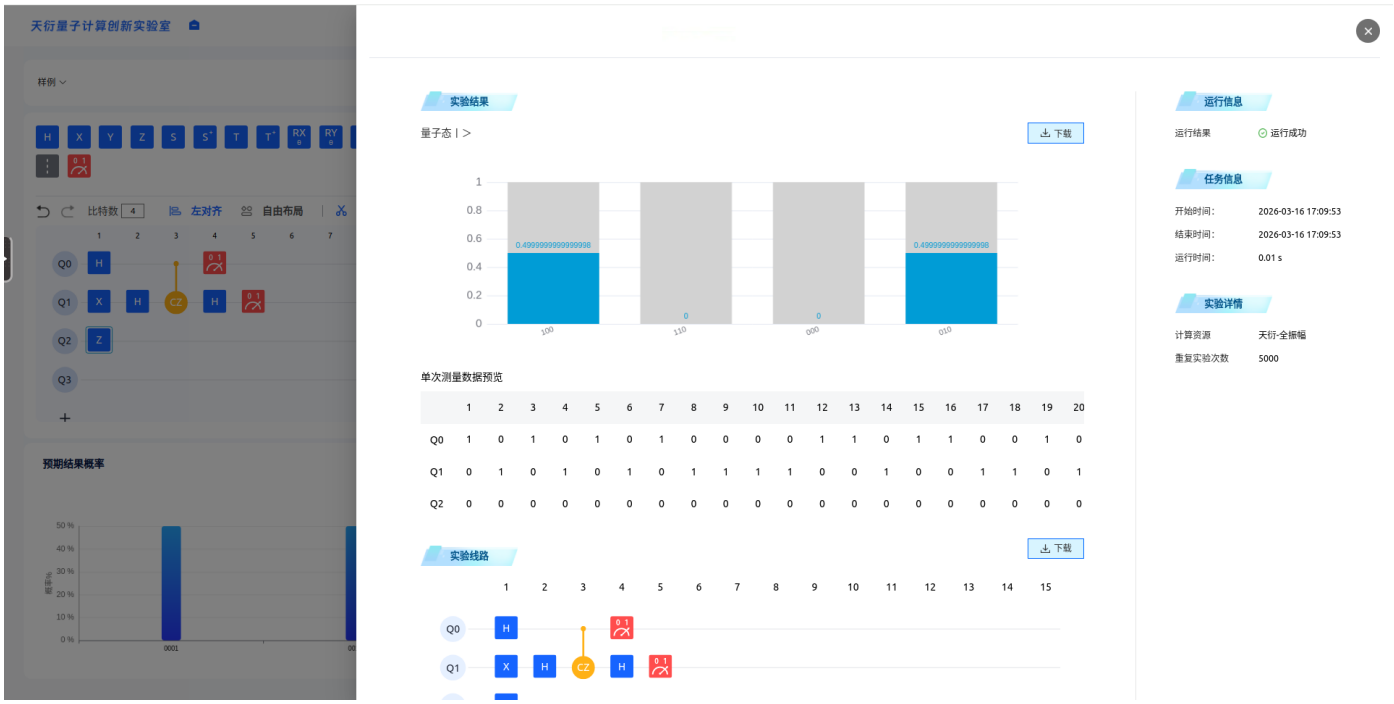
预期结果概率和振幅展示。



搭建完量子线路后，点击【运行】按钮，



可在任务结果查看任务运行状态，运行状态有运行中、运行失败、运行成功。



最后，点击右上角"退出应用"按钮、退出图形化编程。

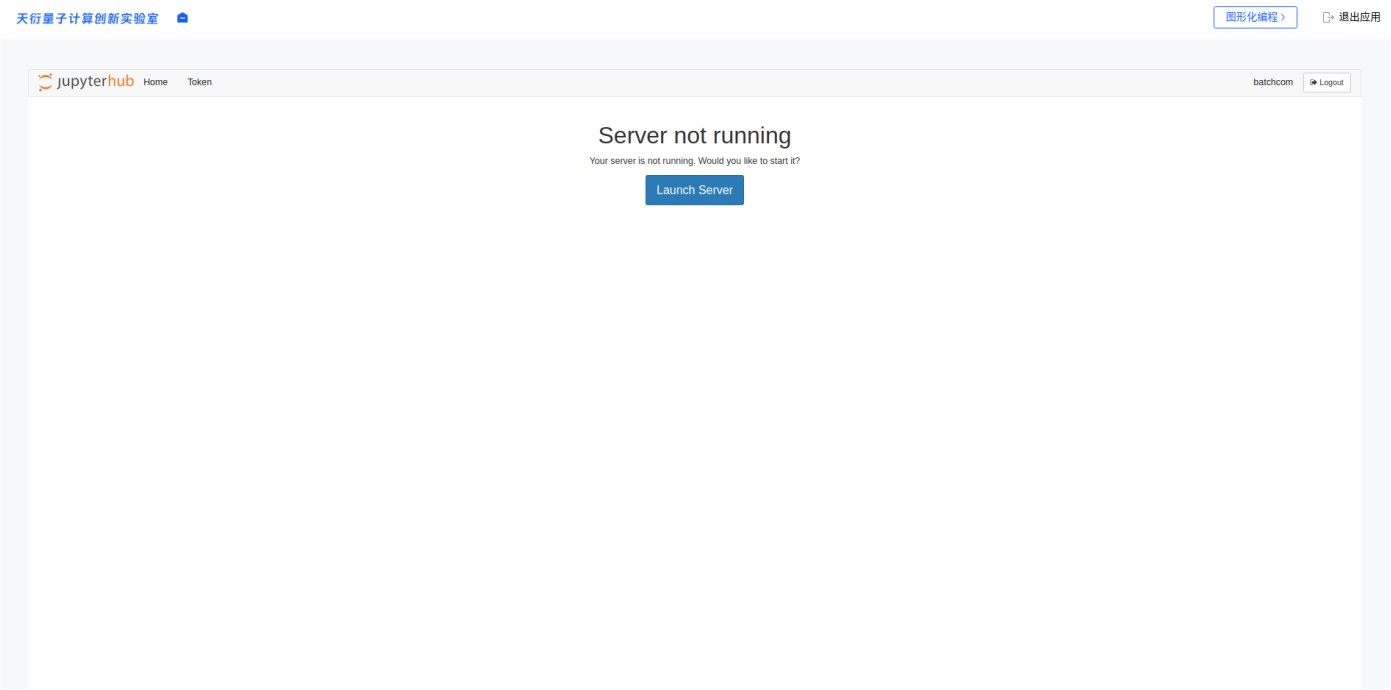
在开发机中使用量子开发机进行量子计算编程

量子计算实验室提供了量子计算编程环境。支持交互式集成开发工具 Jupyter ，已预置Cqlib，开发者可直接进行编程体验，开启量子编程之旅。

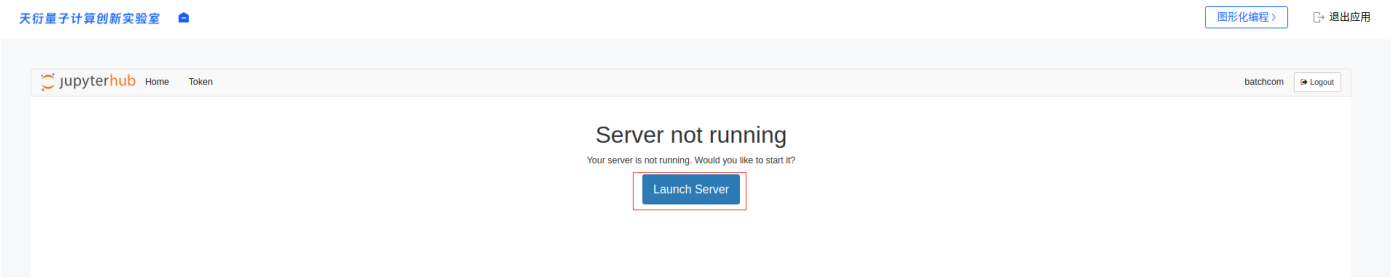
Cqlib 是量子计算软件开发工具包。基于国产指令集 QCIS 打造，提供了简单高效的量子编程工具。

在天衍量子计算实验室功能首页、点击"量子计算编程"按钮

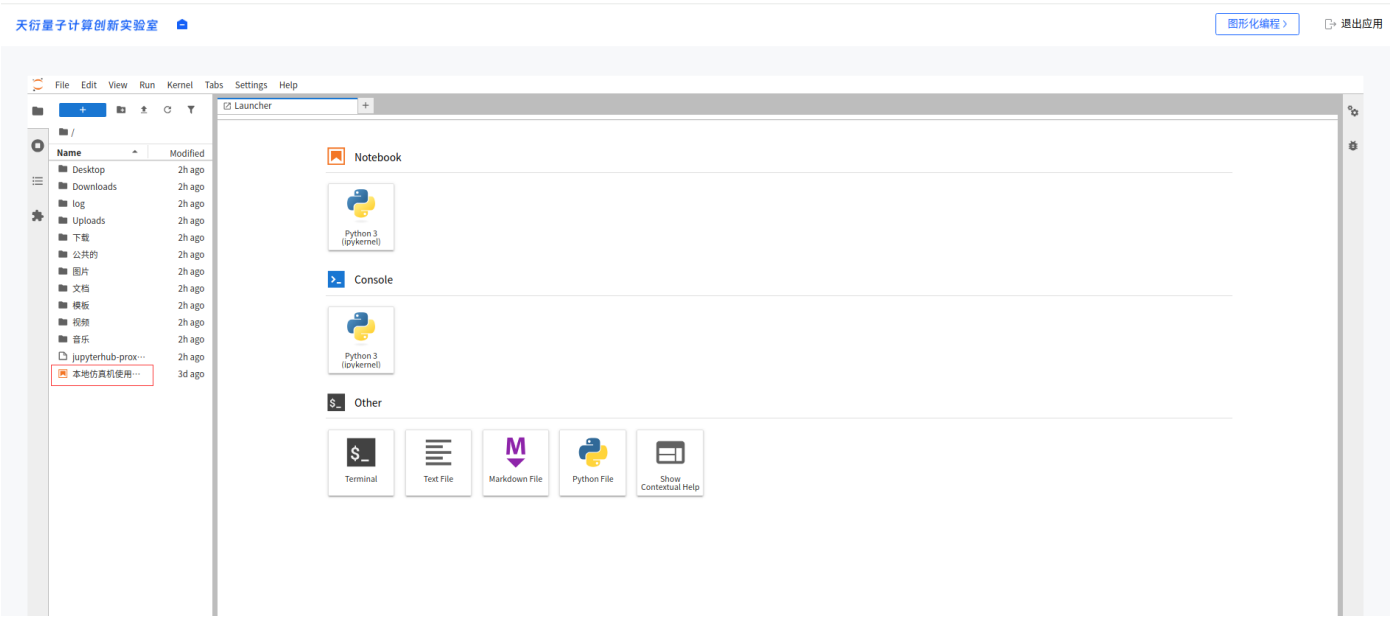
进入量子计算编程。



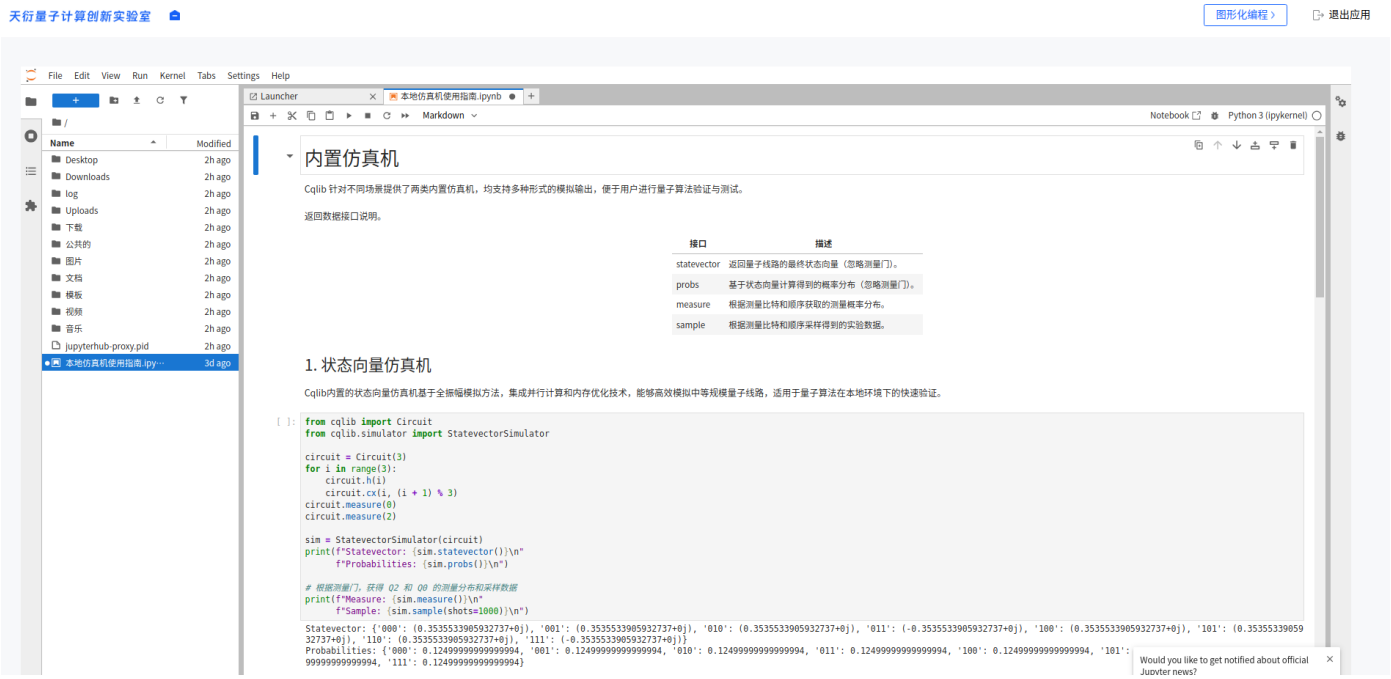
点击"Launch Server"按钮、初始化服务。



初始化完成后、显示jupyter页面，双击左侧"本地仿真机使用指南.ipynb"链接，

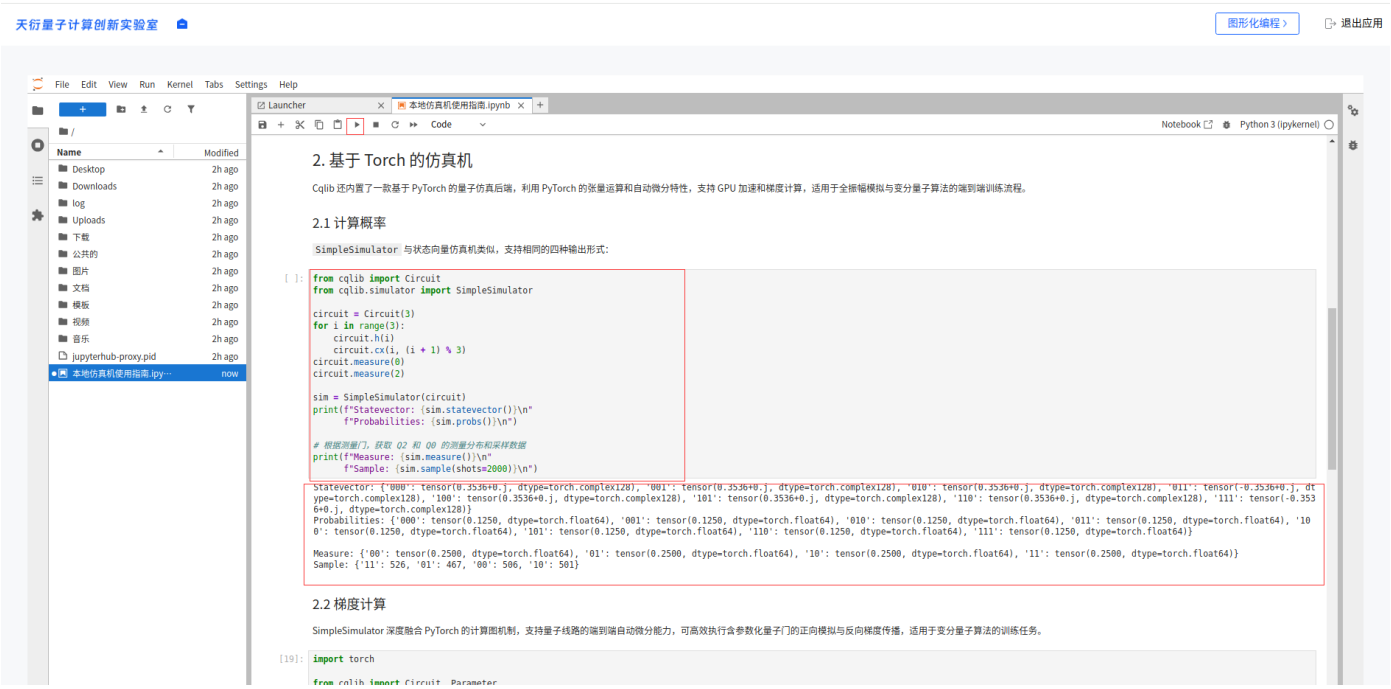


打开本地仿真机页面。

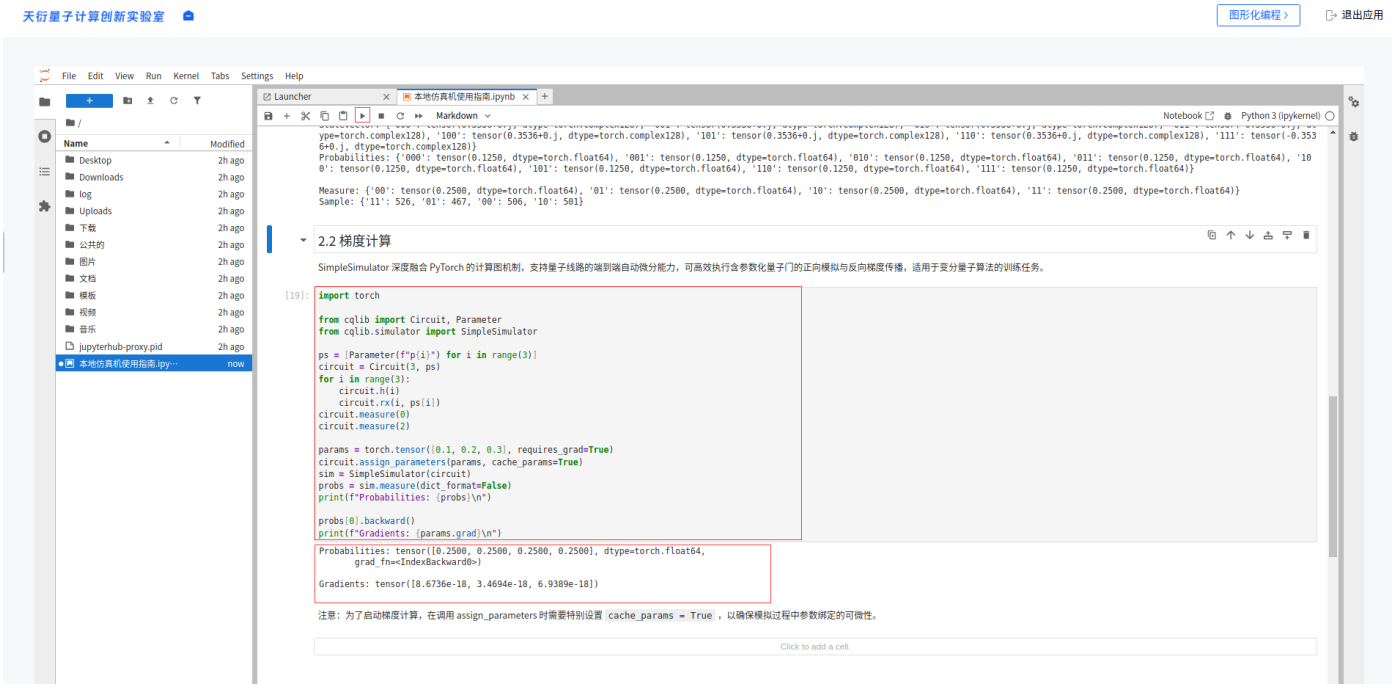


按照本地仿真机页面中段落，依次进行状态向量仿真机、基于 Torch 的仿真机实验。

状态向量仿真机基于全振幅模拟方法，集成并行计算和内存优化技术，能够高效模拟中等规模量子线路，适用于量子算法在本地环境下的快速验证。



基于 PyTorch 的量子仿真后端，利用 PyTorch 的张量运算和自动微分特性，支持 GPU 加速和梯度计算，适用于全振幅模拟与变分量子算法的端到端训练流程。



调用真机流程

登陆天衍量子计算云平台 https://qc.zdxlz.com/home?lang=zh

点击个人中心，复制连接秘钥。每个新注册用户，每个月会赠送10min 天衍-176机时。

编写代码

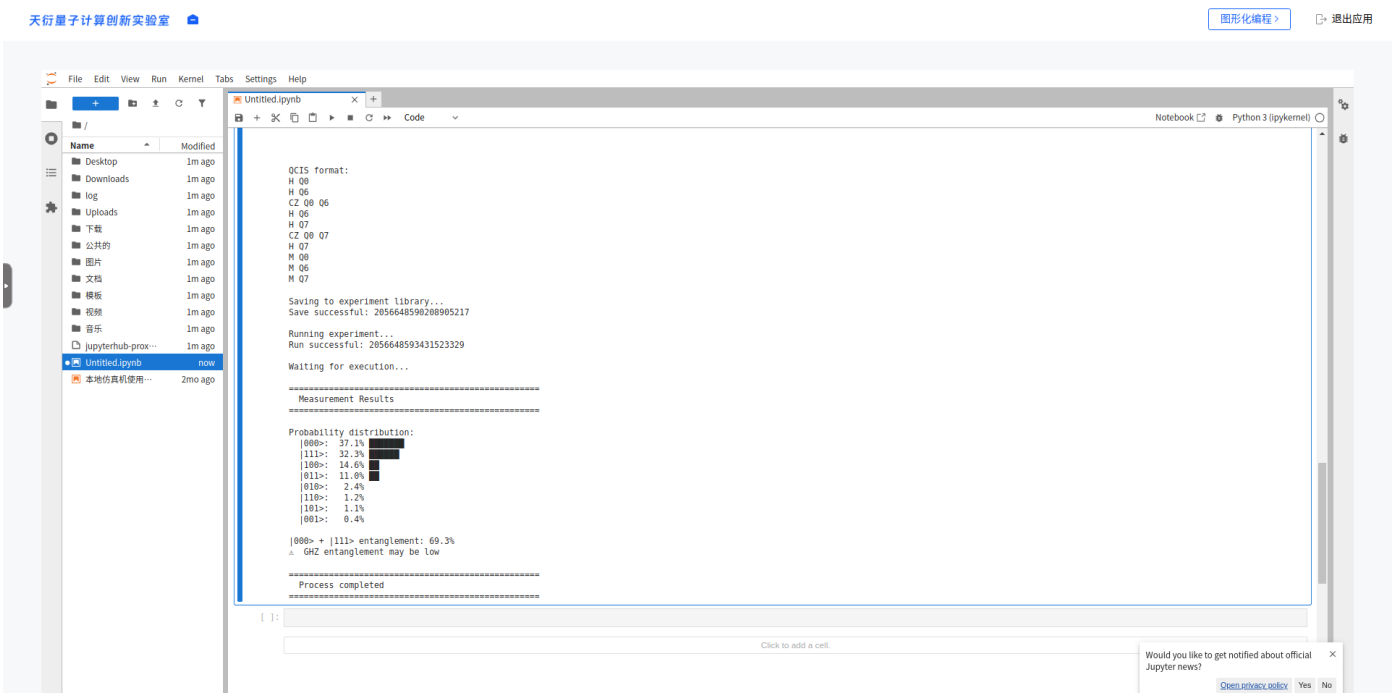
此处以用调用天衍-176完成 3比特 GHZ 纠缠态为样例

```
import time
import json
from cqlib import TianYanPlatform
from cqlib.circuits import Circuit

print("=" * 50)
print("  3-bit GHZ Entanglement - Tianyan-176 (拓扑适配版)")
```

```
print("=" * 50)

# ========== 1. Login ==========

login_key = "替换为刚刚copy的连接秘钥"
platform = TianYanPlatform(login_key=login_key)
print("Login successful")

# ========== 2. Create Lab ==========

lab_name = "GHZ-Test-" + str(int(time.time()))
lab_remark = "Temporary lab for GHZ test"
try:
    lab_id = platform.create_lab(name=lab_name, remark=lab_remark)
    print(f"Lab created: {lab_id}")
except Exception as e:
    print(f"Failed to create lab: {e}")
    lab_id = input("Enter existing lab id (or press enter to exit): ").strip()
    if not lab_id:
        print("No lab id, exiting")
        exit()

# ========== 3. Set Machine ==========

platform.set_machine("tianyan176")
print("Device set: tianyan176")

# ========== 4. Build Circuit - 使用真实拓扑!  ==========

# 使用 0, 6, 7 这三个可连接的量子比特 注意需要使用可用拓扑

circuit = Circuit([0, 6, 7])
circuit.h(0)
circuit.cx(0, 6)
circuit.cx(0, 7)
circuit.measure_all()

print("\nCircuit preview:")
print(circuit.draw())

print("\nQCIS format:")
print(circuit.qcis)

# ========== 5. Save Experiment ==========

print("\nSaving to experiment library...")
try:
    exp_id = platform.save_experiment(
        lab_id=lab_id,
        circuit=circuit.qcis,
        exp_name="GHZ-3-qubit-topo",
        shots=1024
    )
    print(f"Save successful: {exp_id}")
except Exception as e:
    print(f"Save failed: {e}")
    exit()
```

```
# ========== 6. Run Experiment ==========

print("\nRunning experiment...")
try:
    query_id = platform.run_experiment(
        exp_id=exp_id,
        num_shots=1024,
        is_verify=True
    )
    print(f"Run successful: {query_id}")
except Exception as e:
    print(f"Run failed: {e}")
    exit()

# ========== 7. Query Result ==========

print("\nWaiting for execution...")
try:
    exp_result = platform.query_experiment(
        query_id=query_id,
        max_wait_time=180,
        sleep_time=5
    )print()
print("=" * 50)
print("  Measurement Results")
print("=" * 50)
print()

probability = exp_result[0]["probability"]
if isinstance(probability, str):
    probability = json.loads(probability)

print("Probability distribution:")
for state, prob in sorted(probability.items(), key=lambda x: x[1], reverse=True):
    pct = prob * 100
    bar = "█" * int(pct / 5)
    print(f"  |{state}>: {pct:5.1f}% {bar}")

# 验证 GHZ 态
# 注意：现在是 0,6,7 三个比特，所以状态顺序是 0(7号),1(6号),2(0号)
# 二进制表示可能是 000, 001, 010, 011, 100, 101, 110, 111
# 对应的比特分别是 q7, q6, q0

prob_000 = probability.get("000", 0)
prob_111 = probability.get("111", 0)
total = prob_000 + prob_111
print(f"\n|000> + |111> entanglement: {total * 100:.1f}%")
if total > 0.9:
    print("✅ GHZ state verification passed")
else:
    print("⚠️  GHZ entanglement may be low")
    except Exception as e:
    print(f"Query failed: {e}")

print("\n" + "=" * 50)
print("  Process completed")
print("=" * 50)
```

点击运行，可以查看到运行结果。结果显示已经成功制备了 3 个量子比特的 GHZ 纠缠态。



至此，我们实现了在科研助手上使用量子开发机进行量子编程。