

SaaS Computing Service

DiCOSApp

2025/4/10

Jingya You

Interactive Jobs - SaaS Computing Service

System Spec

- Kubernetes and Openstack
 - High extensible and reliable virtual environment
 - Customized Application Deployment
 - Images Repository
 - JupyterLab and various scientific applications
 - Built by user's requirements
 - Software-on-demand Web UI
 - No installation and easy to adopt
 - Working Space
 - Ceph Filesystem
- Node Spec : NVIDIA GPU A100 、 RTX3090 、 RTX4090
 - Service Web Portal :
 - dicos.grid.sinica.edu.tw
 - Mount Space :
 - User Space : /dicos_ui_home/{user}
 - Group Space : /ceph/work/{group}
 - Backup Space : /ceph/project/{group} will be available in 2025

SaaS for Virtualized Computing Service

Scientific Software Repository

- Interactive : Ovito(Molecular Dynamics) 、 cisTEM 、 RELION(Medical Image Reconstruction)
- BioMedical : Cryosparc (* License required from users)
- Anaconda Python packages for ML : JupyterLab 、 TensorFlow 、 PyTorch 、 PyRoot 、 DeepMD(Molecular Dynamics)···etc.

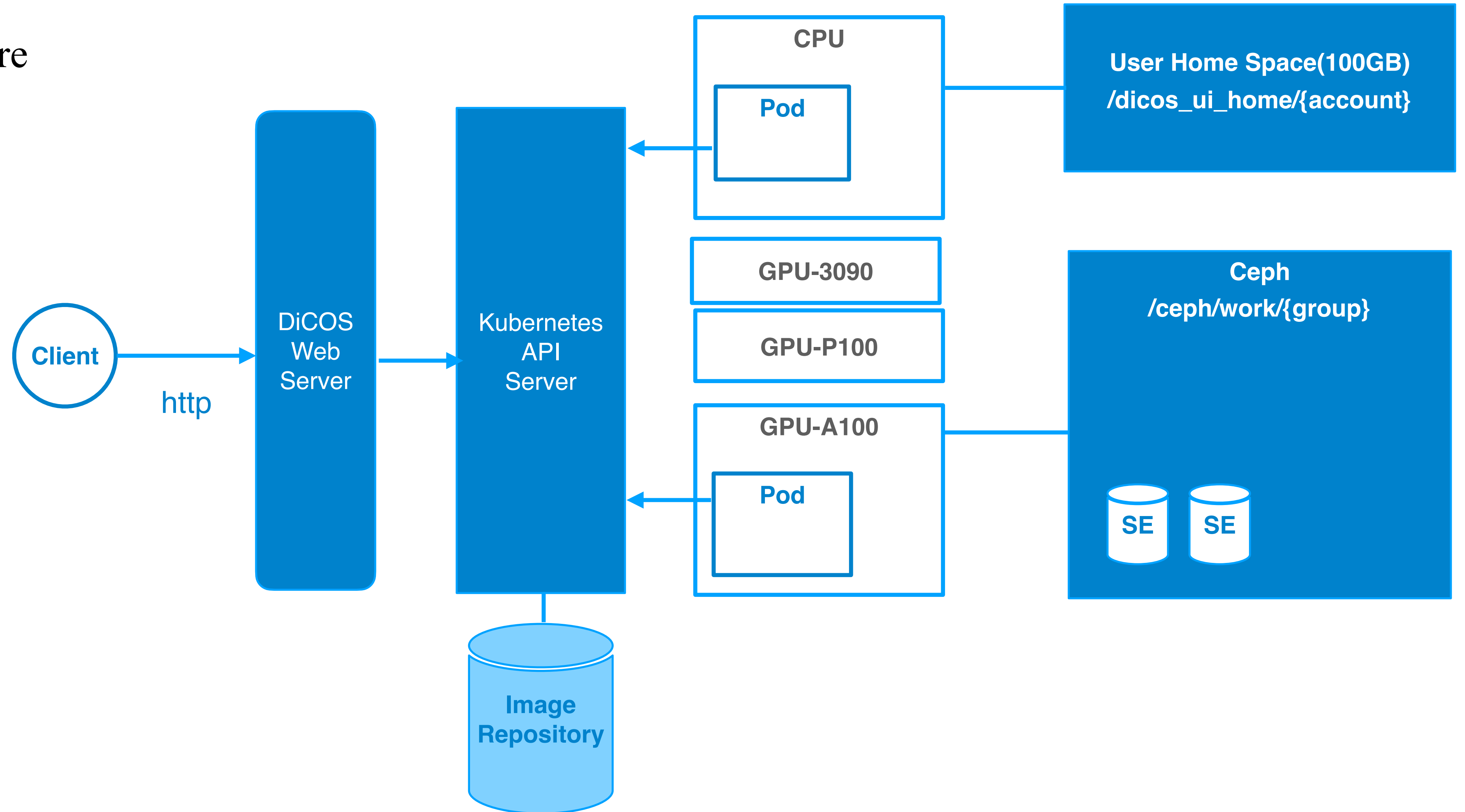


When your job needs

- Interactive UI
- Specific OS or Application required
- Dedicate node for rapid development for multi-core or GPU to develop and testing your task


DiCOSApp for SaaS Computing Service

- System Architecture




SaaS for Virtualized Computing Service - Service Portal

<https://dicos.grid.sinica.edu.tw/dockerapps/>




CryoSPARC 1080ti
Version: 3.3.2
Resources: 52%

Launch




CryoSPARC RTX3090
Version: 4.0.2
Resources: 44%

Launch




PyRoot
Version: GPU with 1080ti
Resources: 52%

Launch ▾




RELION 4 beta
Version: V4
Resources: 52%

Launch ▾



Triton
Version: 22.01-py3 (GPU P100)
Resources: 50%

Launch ▾



Jupyter Lab GPU A100
Version: GPU with Tensorflow A100
Resources: 62%

Launch ▾

- 2 hours
- 3 days
- 7 days
- 10 days

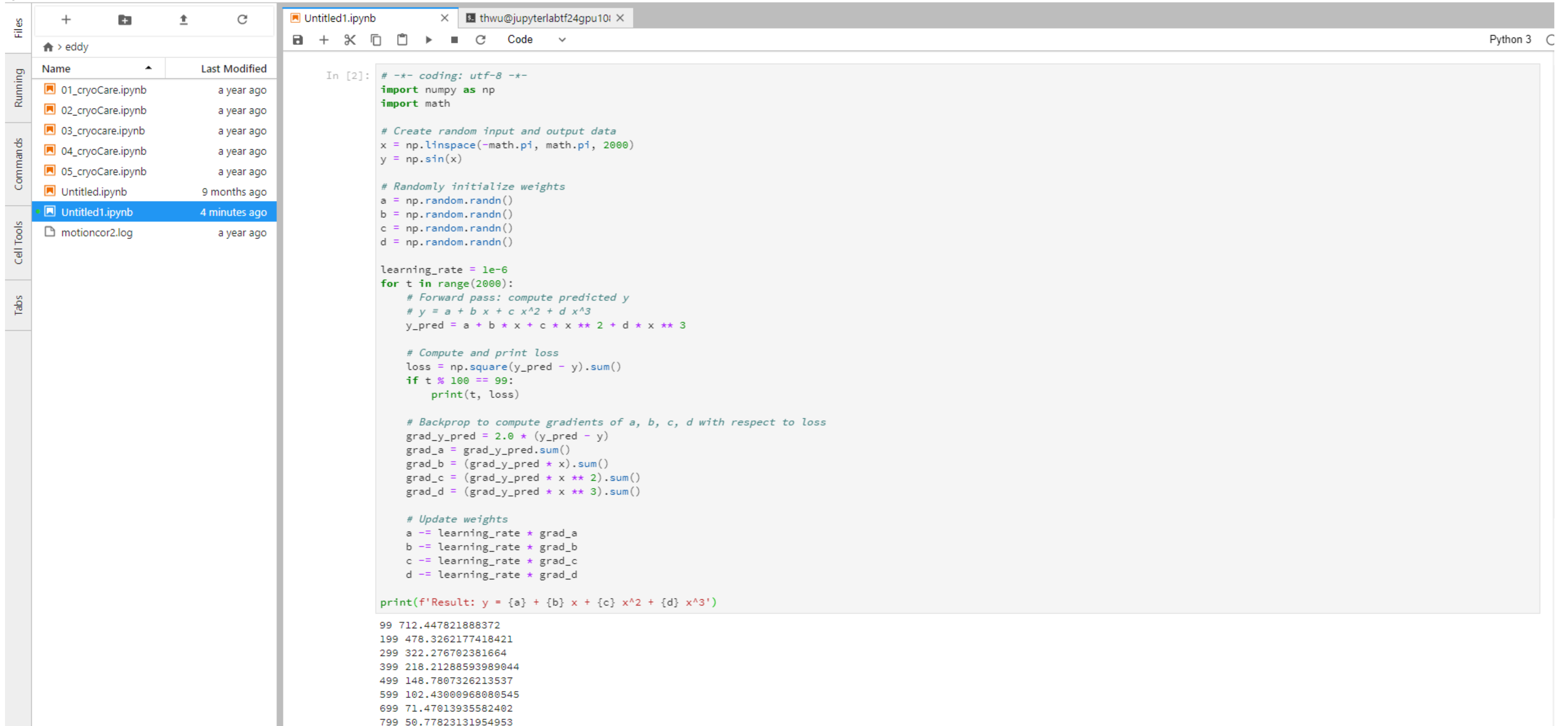
The path of Disk Space

- All user:
/dicos_ui_home/{user} (UI)
- cryoEM group:
/activeEM/data/{group}/{user}
- NSTCCore group:
/ceph/work/{group}/{user}

Example 1.1 (Open a Jupyter RTX 3090)

The screenshot displays the JupyterLab interface. The top menu bar includes File, Edit, View, Run, Kernel, Tabs, Settings, and Help. The left sidebar contains a 'Files' panel with a '+' button and a file browser for the 'eddy' directory. Below this are sections for 'Running' (listing notebooks like 01_cryoCare.ipynb), 'Commands', 'Cell Tools', and 'Tabs'. The main area is the 'Launcher' view, which shows a grid of application icons: Python 2 and Python 3 notebooks (the Python 3 icon is highlighted with a blue border), a Console, another Python 2 and Python 3 notebook, and 'Other' options like Terminal and Text File. The browser tabs at the top show 'Launcher', 'Untitled1.ipynb', and 'thwu@jupyterlabtf24gpu10i'.

Example 1.2 (Open a Jupyter RTX 3090)



The screenshot displays the JupyterLab interface. On the left, the 'Files' sidebar shows a directory named 'eddy' containing several IPYNB files and a log file. The 'Untitled1.ipynb' file is selected. The main area shows the code editor for 'Untitled1.ipynb' with the following Python code:

```
In [2]: # -*- coding: utf-8 -*-
import numpy as np
import math

# Create random input and output data
x = np.linspace(-math.pi, math.pi, 2000)
y = np.sin(x)

# Randomly initialize weights
a = np.random.randn()
b = np.random.randn()
c = np.random.randn()
d = np.random.randn()

learning_rate = 1e-6
for t in range(2000):
    # Forward pass: compute predicted y
    # y = a + b x + c x^2 + d x^3
    y_pred = a + b * x + c * x ** 2 + d * x ** 3

    # Compute and print loss
    loss = np.square(y_pred - y).sum()
    if t % 100 == 99:
        print(t, loss)

    # Backprop to compute gradients of a, b, c, d with respect to loss
    grad_y_pred = 2.0 * (y_pred - y)
    grad_a = grad_y_pred.sum()
    grad_b = (grad_y_pred * x).sum()
    grad_c = (grad_y_pred * x ** 2).sum()
    grad_d = (grad_y_pred * x ** 3).sum()

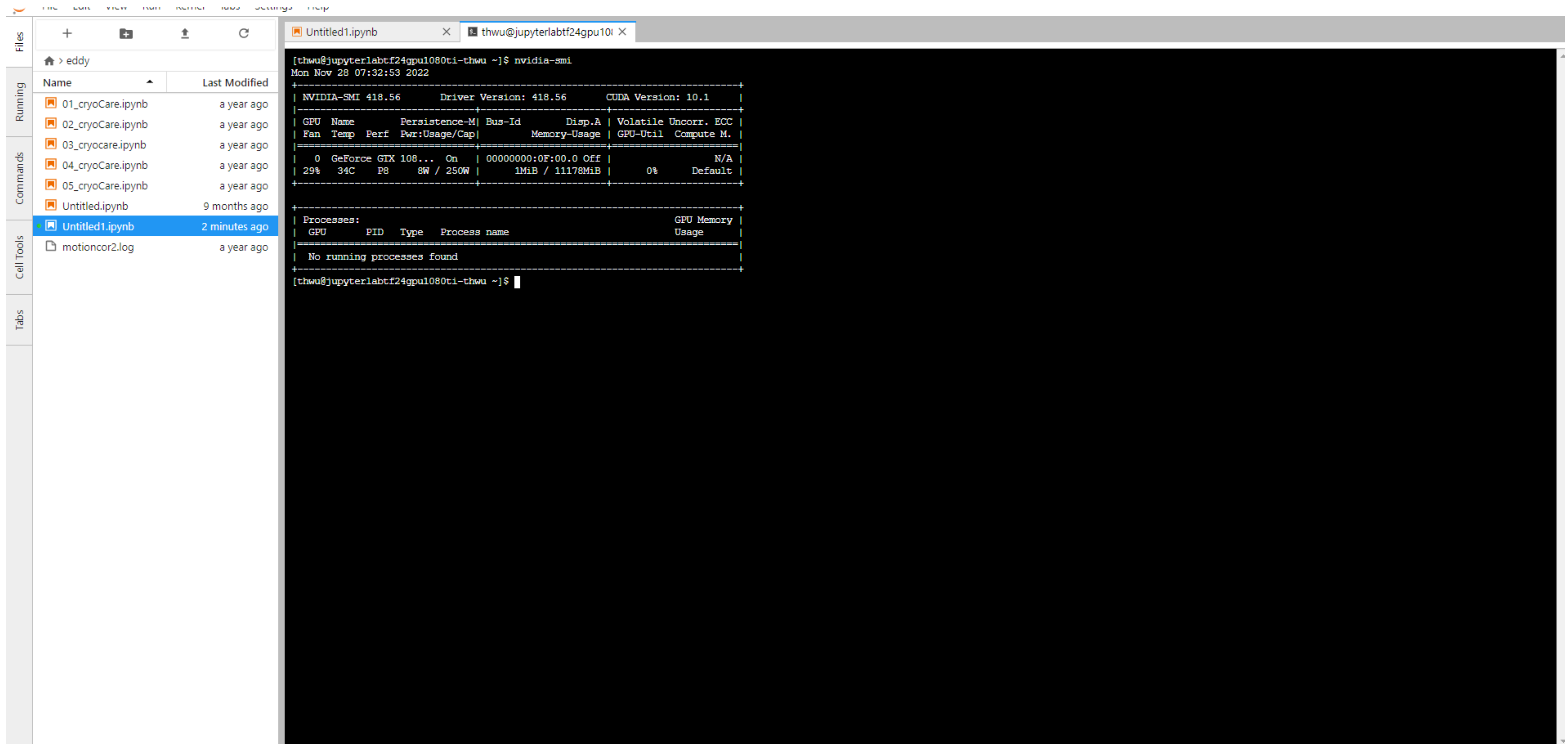
    # Update weights
    a -= learning_rate * grad_a
    b -= learning_rate * grad_b
    c -= learning_rate * grad_c
    d -= learning_rate * grad_d

print(f'Result: y = {a} + {b} x + {c} x^2 + {d} x^3')
```

The output of the code is displayed below the editor:

```
99 712.447821888372
199 478.3262177418421
299 322.276702381664
399 218.21288593989044
499 148.7807326213537
599 102.43000968080545
699 71.47013935582402
799 50.77823131954953
```



Example 1.3 (Open a Jupyter RTX 3090 with terminal)




The screenshot displays the JupyterLab interface. On the left, the 'Files' sidebar shows a directory named 'eddy' containing several IPython notebooks (01_cryoCare.ipynb to 05_cryoCare.ipynb) and 'Untitled1.ipynb' (modified 2 minutes ago). The main area is a terminal window with the following output:

```
[thwu@jupyterlabtf24gpu1080ti-thwu ~]$ nvidia-smi
Mon Nov 28 07:32:53 2022
+-----+
| NVIDIA-SMI 418.56      Driver Version: 418.56      CUDA Version: 10.1     |
+-----+-----+
| GPU   Name           Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|  Memory-Usage | GPU-Util  Compute M. |
+-----+-----+
|  0   GeForce GTX 108...  On          | 00000000:0F:00.0 Off  |          N/A         |
| 29%   34C   P8      8W / 250W | 1MiB / 11178MiB |    0%      Default   |
+-----+-----+
+-----+
| Processes:                                                       GPU Memory |
|  GPU       PID    Type   Process name                               Usage      |
+-----+-----+
| No running processes found                                     |
+-----+
[thwu@jupyterlabtf24gpu1080ti-thwu ~]$
```


Example 2. (Open a QisKit app)


 **qiskit**
Version:
Resources: 100%



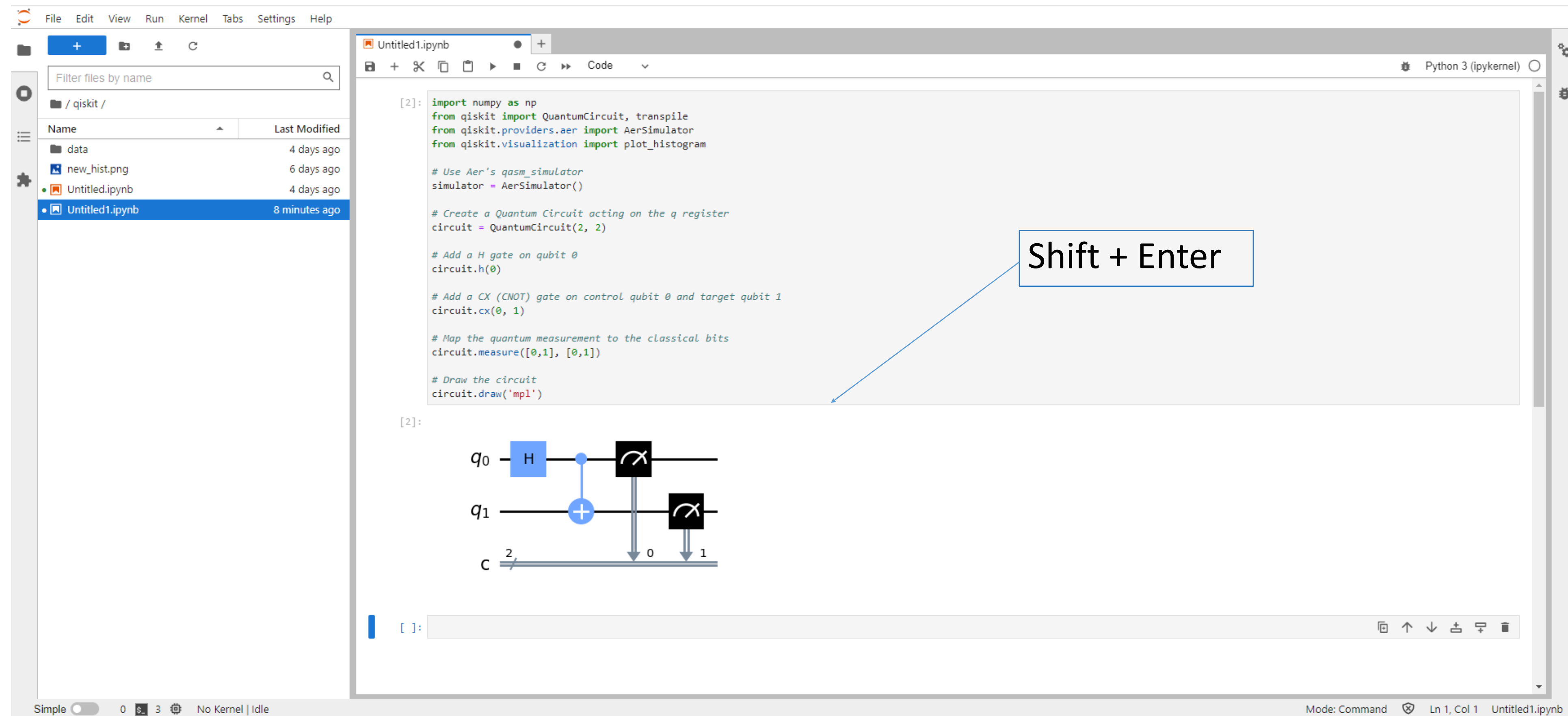
Launch ▾

3 days

 **qiskit**
Version:
Resources: 100%



Open Delete



File Edit View Run Kernel Tabs Settings Help

Filter files by name

Name	Last Modified
data	4 days ago
new_hist.png	6 days ago
Untitled.ipynb	4 days ago
Untitled1.ipynb	8 minutes ago

```
[2]: import numpy as np
from qiskit import QuantumCircuit, transpile
from qiskit.providers.aer import AerSimulator
from qiskit.visualization import plot_histogram

# Use Aer's qasm_simulator
simulator = AerSimulator()

# Create a Quantum Circuit acting on the q register
circuit = QuantumCircuit(2, 2)

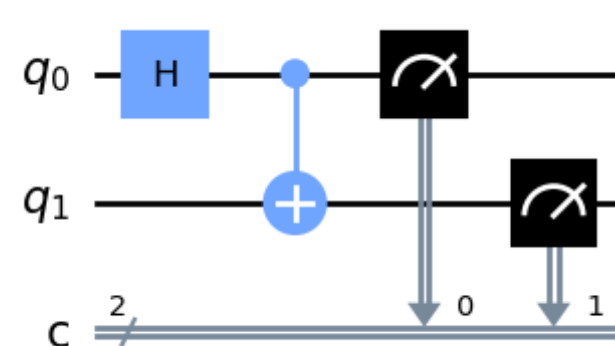
# Add a H gate on qubit 0
circuit.h(0)

# Add a CX (CNOT) gate on control qubit 0 and target qubit 1
circuit.cx(0, 1)

# Map the quantum measurement to the classical bits
circuit.measure([0,1], [0,1])

# Draw the circuit
circuit.draw('mpl')
```


[2]:



[]:

Simple 0 s 3 No Kernel | Idle Mode: Command Ln 1, Col 1 Untitled1.ipynb


Example 3. (Open a Relion 4-beta app)



RELION 4 beta
Version: V4
Resources: 50%

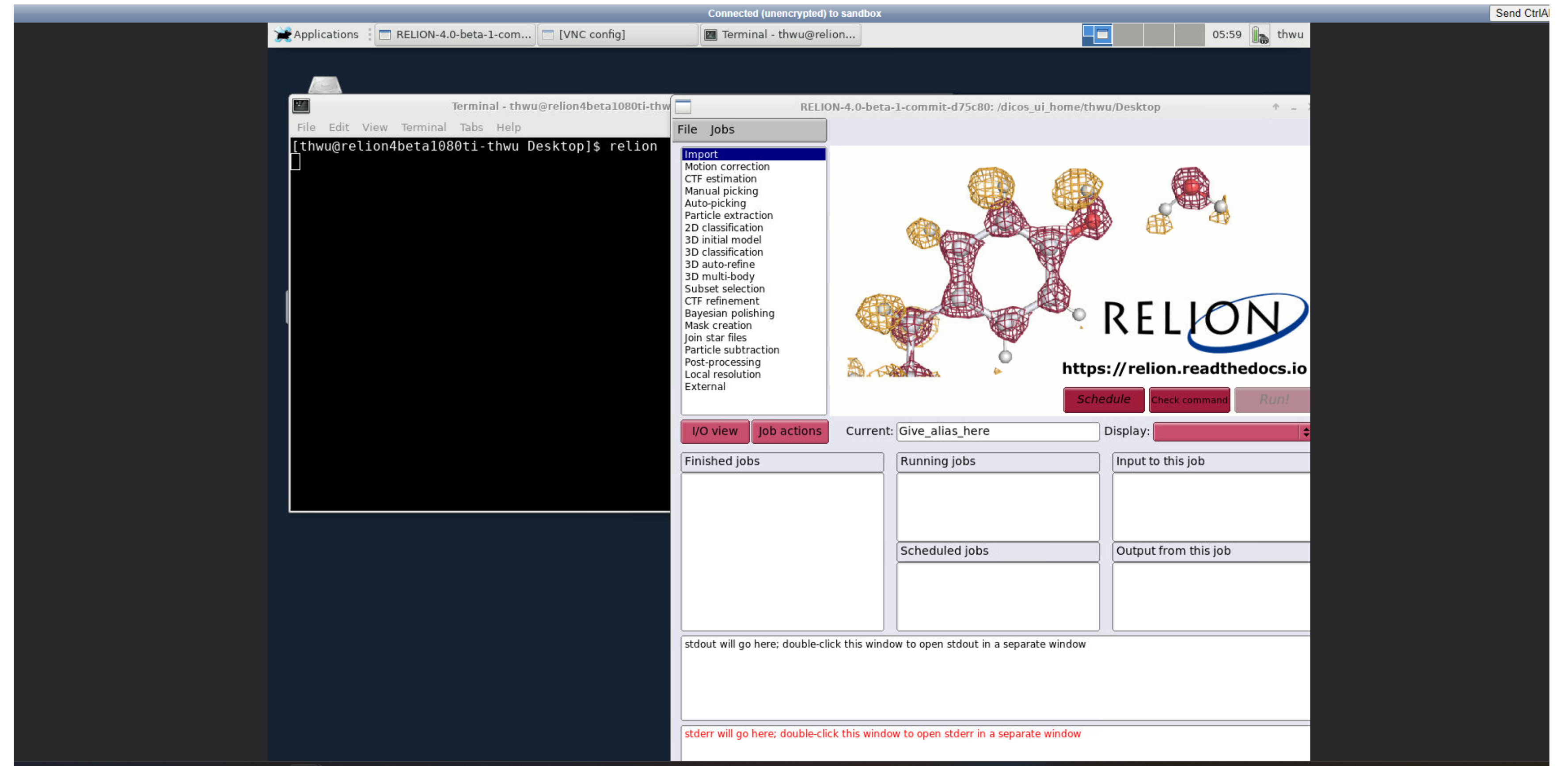
Launch ▾

- 3 days
- 7 days
- 10 days**



RELION 4 beta
Version: V4
Resources: 43%

Open



The screenshot shows a VNC desktop environment. At the top, a status bar indicates "Connected (unencrypted) to sandbox" and the time "05:59" with the user "thwu". The desktop contains several windows:

- Terminal - thwu@relion4beta1080ti-thwu Desktop**: Shows the command `[thwu@relion4beta1080ti-thwu Desktop]$ relion` being executed.
- RELION-4.0-beta-1-commit-d75c80: /dicos_ui_home/thwu/Desktop**: The main application window. It features a sidebar menu with options like "Import", "Motion correction", "CTF estimation", "Manual picking", "Auto-picking", "Particle extraction", "2D classification", "3D initial model", "3D classification", "3D auto-refine", "3D multi-body", "Subset selection", "CTF refinement", "Bayesian polishing", "Mask creation", "Join star files", "Particle subtraction", "Post-processing", "Local resolution", and "External". The main area displays a 3D molecular model and the RELION logo with the URL <https://relion.readthedocs.io>. Buttons for "Schedule", "Check command", and "Run!" are visible.
- Job Management Panels**: Below the main interface, there are panels for "Finished jobs", "Running jobs", "Input to this job", "Scheduled jobs", and "Output from this job".
- Output Windows**: At the bottom, there are instructions for "stdout will go here; double-click this window to open stdout in a separate window" and "stderr will go here; double-click this window to open stderr in a separate window".

Upload and Download your data

1. Download [FileZilla Client](#)
2. Host (主機): `dicos-ui.grid.sinica.edu.tw`
3. Account (使用者名稱): `account`
4. Password (密碼): `password+2FA`
5. Port (連接埠): `22`

