

# The service of DiCOS Apps

ASGC

Tsung-Hsun Wu

# The Category of DiCOS app

- DiCOS Apps:

1. Bio-Apps:

cryoSPARC version, 2, 3, 4 (1080 ti, p100, rtx 3090) .

Relion version, 3, 3.1, 4-beta (1080 ti)

AlphaFold (v100, A100) and RoseTTAFold. (1080 ti)

Dynamo, IMOD, EMAN2, cisTEM, Jupyter CryoCare.. Etc.


2. Phys-Apps

Deepmd-kit, MAML, PyRoot, Paraview, Ovito, Qiskit.

3. Machine-learning-Apps:


Jupyter-Lab (PyTorch, Tensorflow) (cpu, 1080ti, v100, A100, rtx3090), Triton

# <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 ▾

- 3 days
- 7 days
- 10 days**

# The path of Disk Space

- All user:  
  /dicos\_ui\_home/{user} (UI) (100 GB only)
- cryoEM group:  
  /activeEM/data/{group}/{user} (15 TB )
- NSTCCore group:  
  /ceph/work/{group}/{user} (3 TB)

# Example 1.1 (Open a Jupyter RTX 3090)

The screenshot displays the JupyterLab interface. On the left, the file browser shows a directory named 'eddy' containing several files. The file 'Untitled1.ipynb' is selected and highlighted in blue. A blue arrow points to the '+' icon in the file browser's toolbar. The main area of the interface is the 'Launcher', which is currently empty. It features a grid of icons for creating new resources: two Python notebooks (Python 2 and Python 3), a console, two more Python notebooks (Python 2 and Python 3), and two other resources (Terminal and Text File). The Python 3 notebook icon in the 'Notebook' section is highlighted with a blue border.

File browser contents:

Name	Last Modified
01_cryoCare.ipynb	a year ago
02_cryoCare.ipynb	a year ago
03_cryocare.ipynb	a year ago
04_cryoCare.ipynb	a year ago
05_cryoCare.ipynb	a year ago
Untitled.ipynb	9 months ago
Untitled1.ipynb	6 minutes ago
motioncor2.log	a year ago

Launcher contents:

- Notebook
  - Python 2
  - Python 3
- Console
  - Python 2
  - Python 3
- Other
  - Terminal
  - Text File

# Example 1.2 (Open a Jupyter RTX 3090)

The screenshot displays a Jupyter Notebook environment. On the left, a sidebar shows a file browser with a list of files including 'Untitled1.ipynb' (4 minutes ago) and 'motioncor2.log' (a year ago). The main area shows a code cell 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()

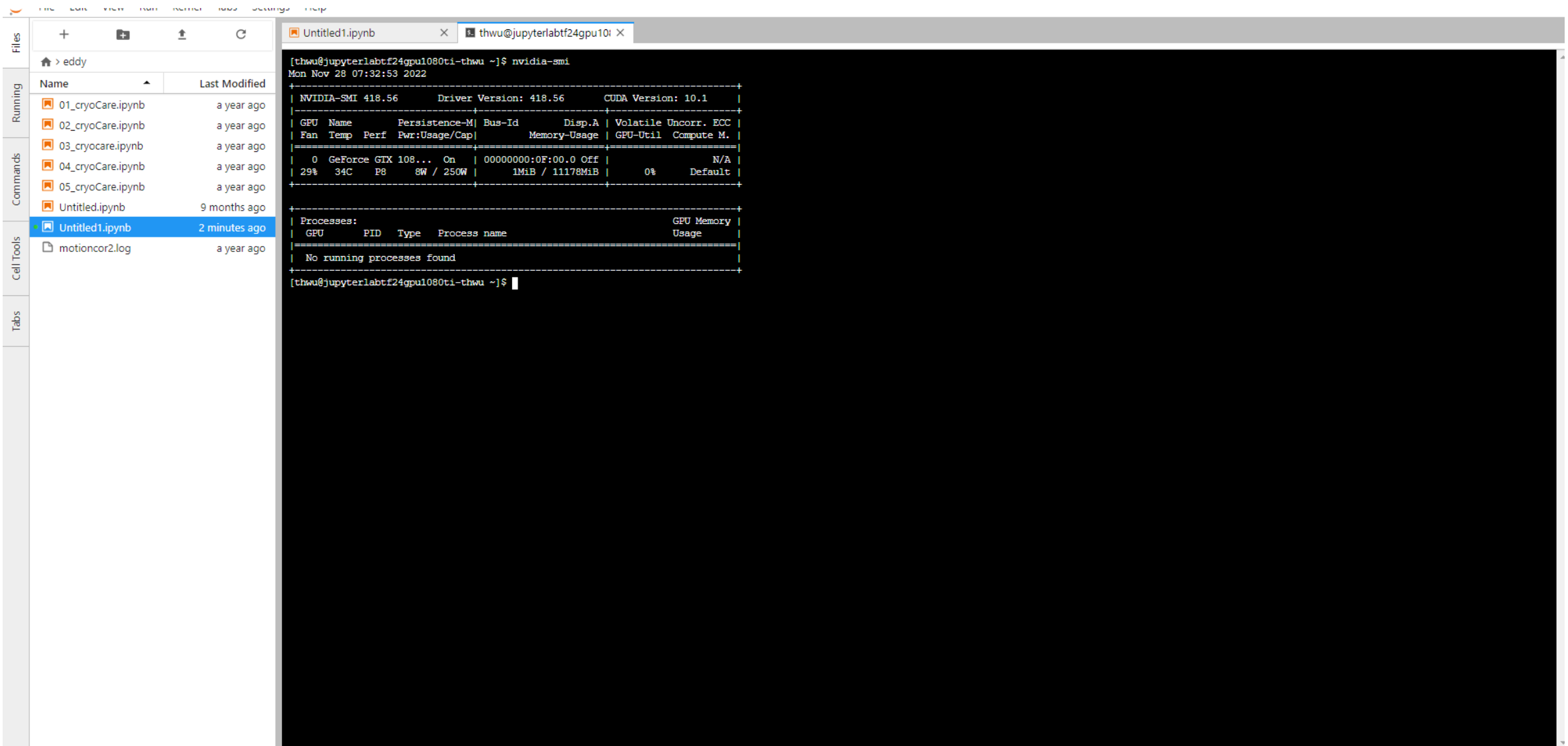
    # Update weights
    a -= learning_rate * grad_a
    b -= learning_rate * grad_b
    c -= learning_rate * grad_c
    d -= learning_rate * grad_d

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

The output of the code cell shows the loss values at intervals of 100 iterations:

```
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 shows the JupyterLab interface. On the left, the 'Files' panel displays a directory structure with files like '01\_cryoCare.ipynb' through '05\_cryoCare.ipynb', 'Untitled.ipynb', and 'motioncor2.log'. The 'Untitled.ipynb' file is selected. The main area shows 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      PID   Type   Process name                      GPU Memory
| Usage
+-----+
| No running processes found
+-----+
[thwu@jupyterlabtf24gpu1080ti-thwu ~]$
```


# Example 2. (Open a QisKit app)



**qiskit**  
Version:  
Resources: 100%

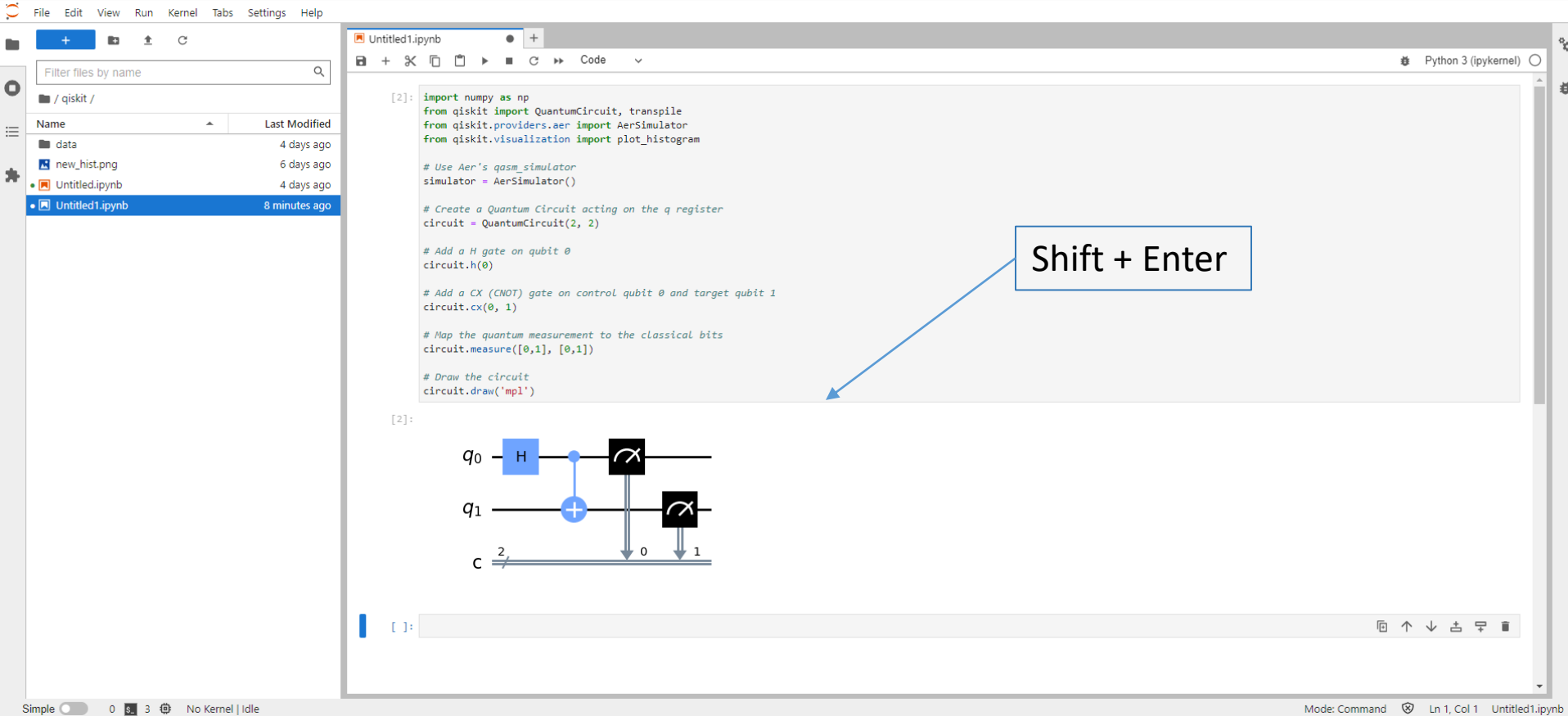
Launch ▾

3 days



**qiskit**  
Version:  
Resources: 100%

Open Delete



The screenshot shows a JupyterLab environment. On the left is a file browser for the directory `/qiskit/` containing files like `data`, `new_hist.png`, `Untitled1.ipynb`, and `Untitled1.ipynb`. The main area is a code editor for `Untitled1.ipynb` with the following Python code:

```
[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')
```


Below the code, the rendered quantum circuit diagram is shown. It features two qubits, `q0` and `q1`, and two classical bits, `c0` and `c1`. Qubit `q0` starts with an H gate, followed by a CNOT gate with `q1` as the target. Both qubits are then measured. The measurement results are stored in classical bits `c0` and `c1`.


A blue box with the text "Shift + Enter" and an arrow points to the code editor, indicating the keyboard shortcut to execute the code cell.

At the bottom of the interface, the status bar shows "Simple" mode, "0" errors, "3" warnings, and "No Kernel | Idle". The bottom right corner indicates "Mode: Command", "Ln 1, Col 1", and "Untitled1.ipynb".




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

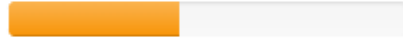
 **RELION 4 beta**  
Version: V4  
Resources: 50%



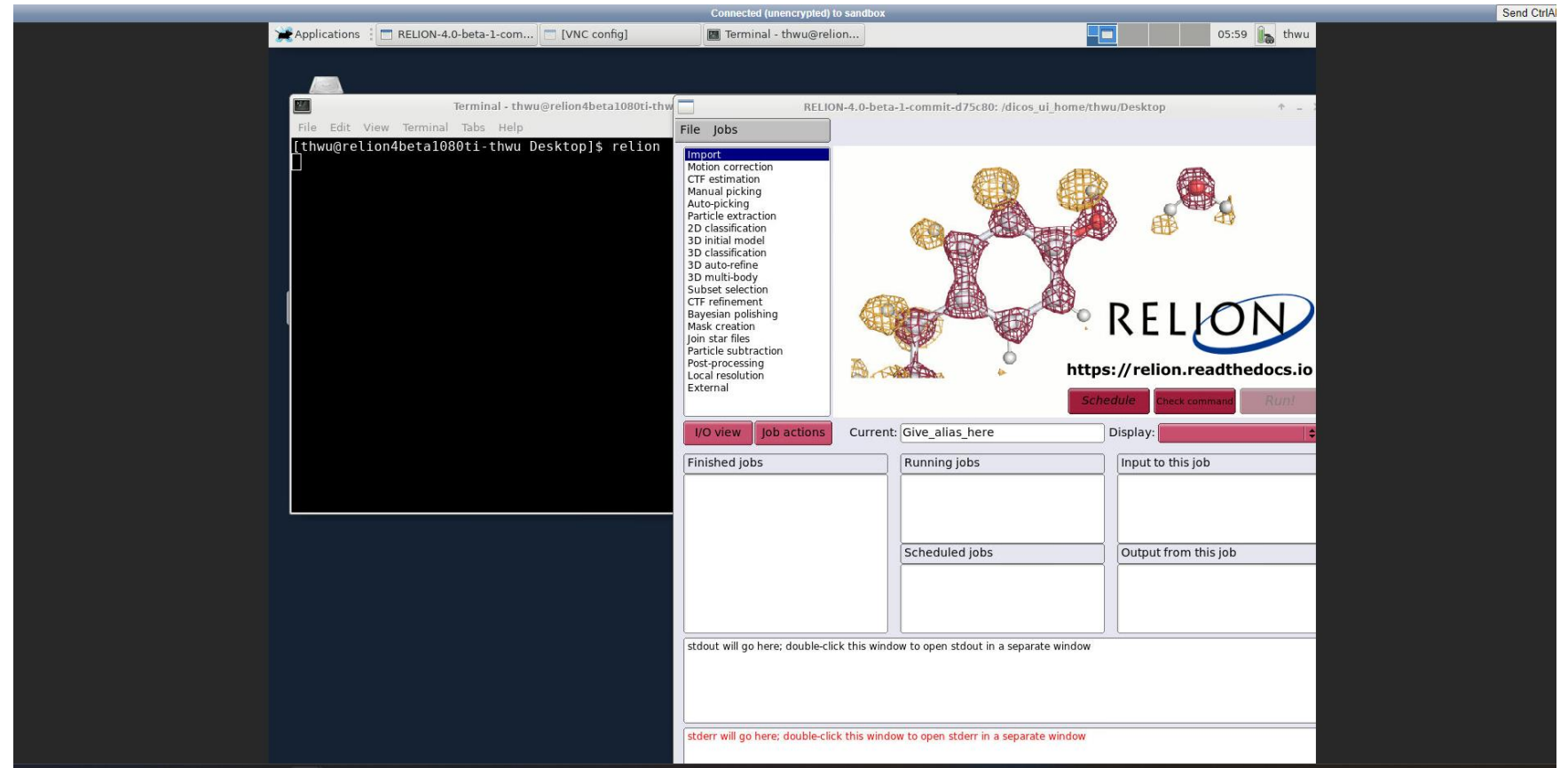
Launch ▾

- 3 days
- 7 days
- 10 days

 **RELION 4 beta**  
Version: V4  
Resources: 43%



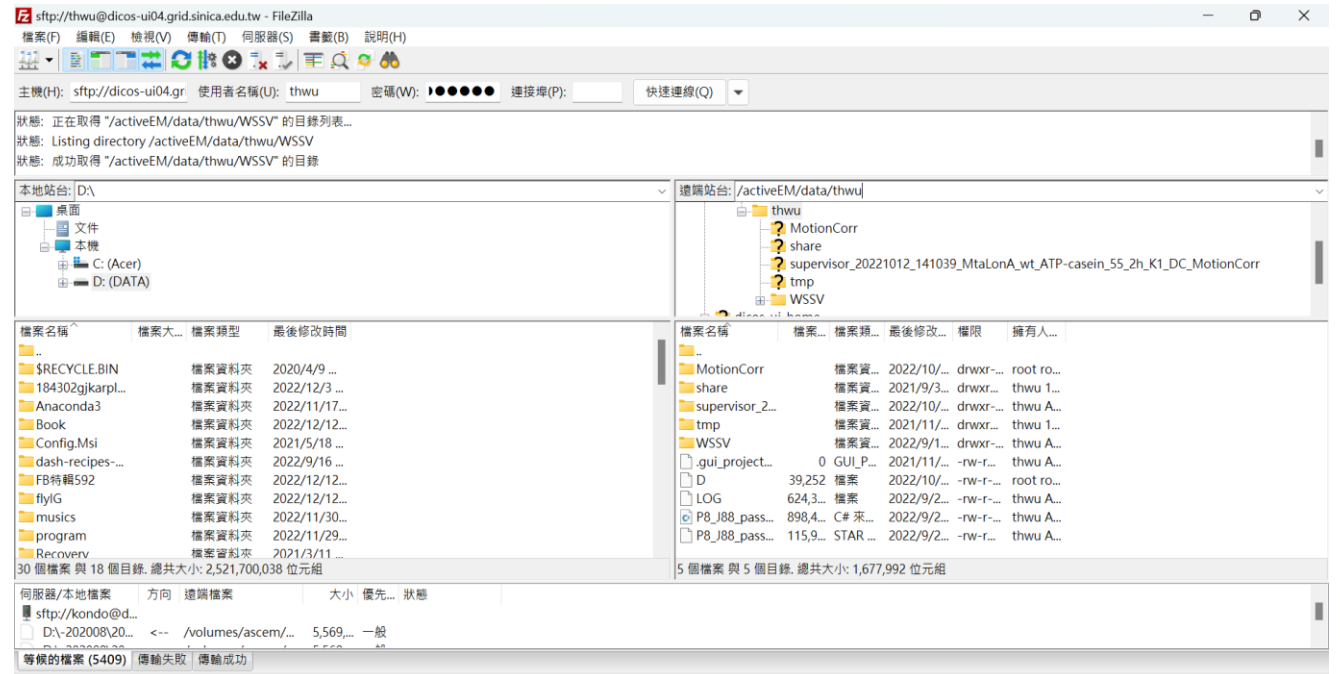
Open Delete



The screenshot shows a VNC desktop environment. At the top, a terminal window is open with the command `relion` entered. To the right, a web browser window displays the RELION interface. The interface features a molecular structure visualization, the RELION logo, and the URL <https://relion.readthedocs.io>. Below the logo, there are buttons for `Schedule`, `Check command`, and `Run!`. The interface also includes a `File Jobs` 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`. At the bottom, there are sections for `Finished jobs`, `Running jobs`, `Scheduled jobs`, `Input to this job`, and `Output from this job`. A note at the bottom states: `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`
5. Port (連接埠): `22`



Thanks & Questions and Answers.

DiCOS Support E-Mail: [DiCOS-Support@twgrid.org](mailto:DiCOS-Support@twgrid.org)