國立臺灣海洋大學

AI 模型訓練平台使用說明

2025.3.25 更新

目錄

服務說明

◎AI 模型訓練平台服務說明

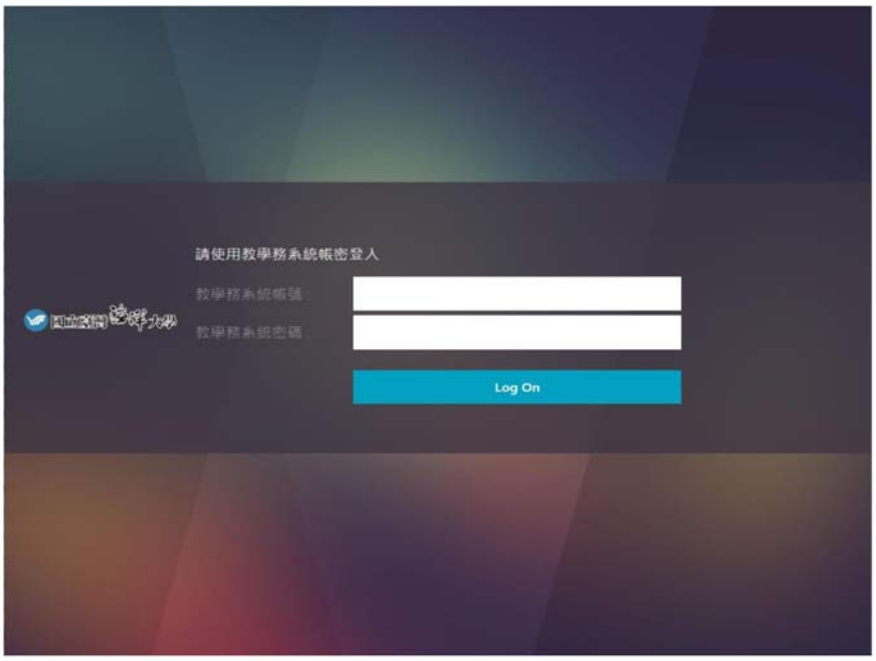
1. 本服務用於支援深度學習、機器學習等相關領域之教學及研究。
2. 使用資格：限本校教師及學生使用，以教學務系統帳密登入立即使用。
3. 因主機資源有限，使用方式一律透過排程管理分配計算資源。每個帳號可以使用的算力資源以當前伺服器的所有資源為限，目前最多可分配使用 CPU 72 核心、記憶體 240GB、4 張 GPU 顯示卡記憶體共 96GB，儲存空間 1GB(若有需要擴充請洽圖系組賴先生，電子郵件 tedlai@mail.ntou.edu.tw 或校內分機 2110)，限制使用時長為 7 日(168 小時)，之後容器作業即自動停止，運算資源釋放讓下一位等待者使用，儲存空間保留至學期末刪除。
4. 為有效使用系統之儲存空間與安全性，使用者應自行下載重要資料，本處不承擔資料毀損之責任。
5. 使用者可依本平台提供的映像檔、樣板或自訂方式建立容器(Container)運算環境，容器提供使用者完整權限可自行安裝套件，使用者須具備 Linux、Container 及 Machine Learning 等操作使用相關知識。
6. 禁止使用本系統進行虛擬貨幣挖礦行為，違者經發現立即取消使用資格。

前置作業

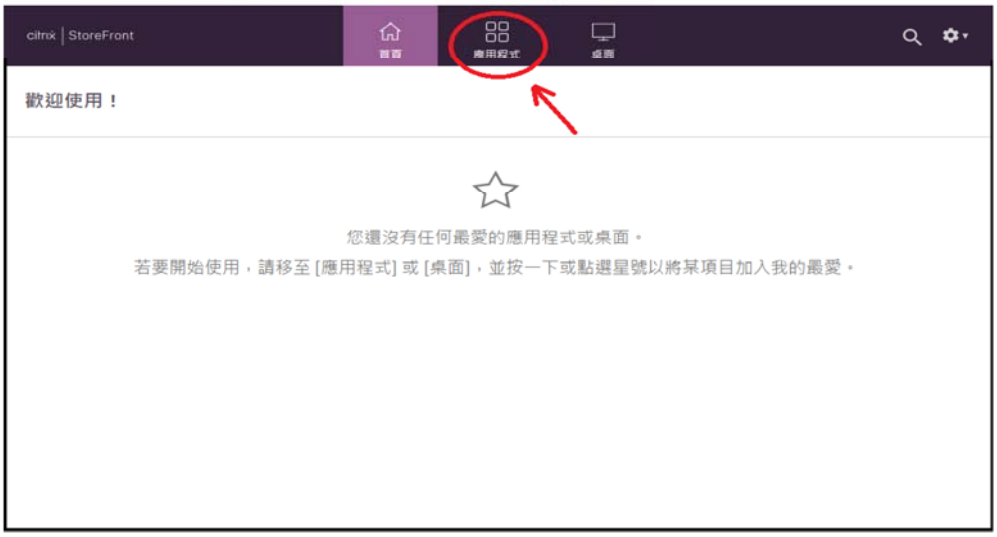
◎請使用雲端電腦教室進入 AI 模型訓練平台

若您是第一次使用雲端電腦教室，請務必詳閱使用說明(電腦版 PDF 檔)(電腦版影片檔)(行動裝置版 PDF 檔)，依照正確步驟安裝軟體。**雲端電腦教室安裝完成後再依照以下步驟操作。**

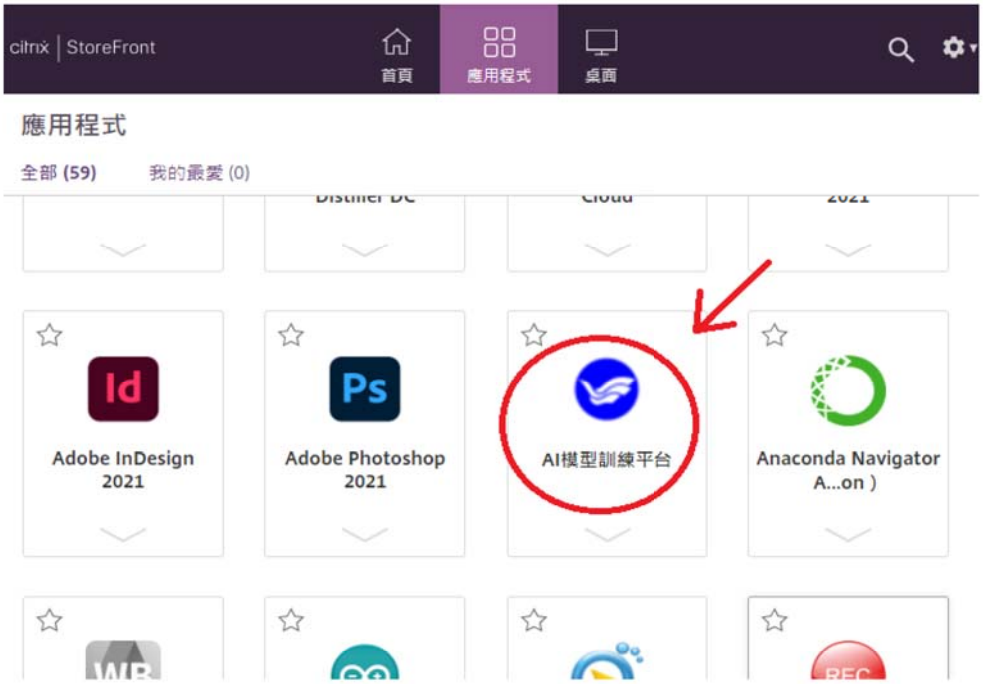
1. 開啟雲端電腦教室網頁 https://vpc.ntou.edu.tw/，輸入教學務系統帳密登入。



2. 登入後，滑鼠點擊紅圈處「應用程式」。



3. 滑鼠點擊「AI 模型訓練平台」。



4. 出現正在啟動畫面，請靜待片刻。

5. 等待開啟 AI 模型訓練平台登入頁面之後，輸入您的教學務系統帳號及密碼登入，若您的帳號第一個字元為英文字母時請輸入小寫的英文字母。

6. 成功進入 AI 模型訓練平台的畫面如下。

開始使用 AI 模型訓練平台

◎上傳檔案或資料夾

集中式儲存提供您可以將開發端編輯好的程式碼及資料掛載進容器中進行計算與回收運算成果的空間，預設每人 1GB 的儲存空間，若有需要擴充請洽圖系組賴先生(電子郵件 tedlai@mail.ntou.edu.tw 或校內分機 2110)，空間保留至學期末刪除。

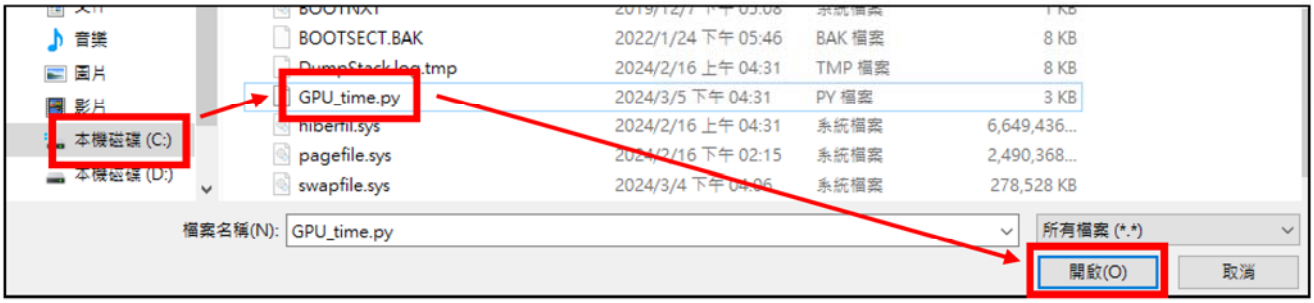
以下示範上傳我們已經在本機寫好的程式碼檔(假設檔案放在 C:\底下的 GPU_time.py)。

1. 資料管理 -> 集中式儲存 -> 點選帳號同名資料夾。

2. 進入資料夾後點選〔上傳檔案〕。

3. 點選〔選擇檔案〕。這邊示範〔選擇檔案〕的方式，您也可以直接拖曳檔案到中間框框裡。

4. 點選要上傳的檔案，再點開啟。



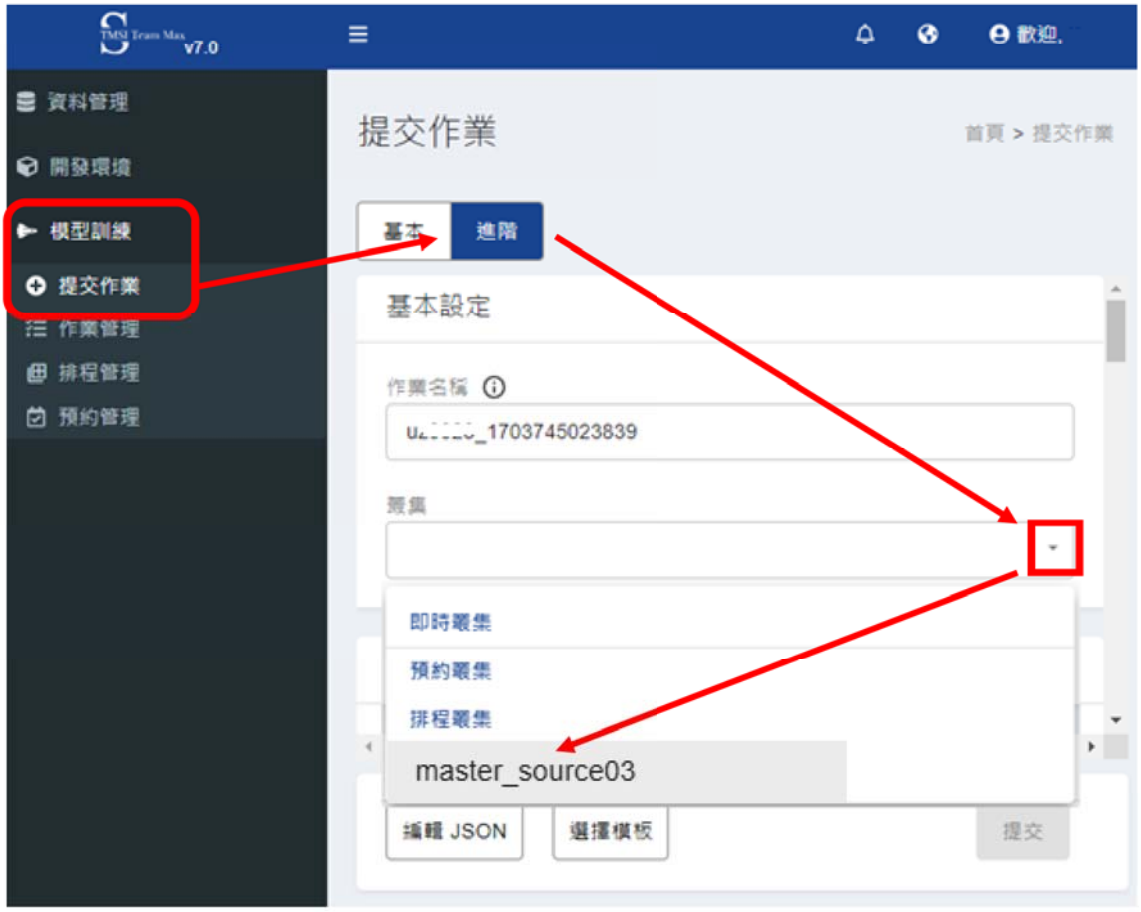
5. 中間框框出現上傳的檔案，再點上傳。

6. 出現上傳成功，按確定。

7. 資料夾底下出現上傳成功的檔案。

◎提交作業(建立容器)

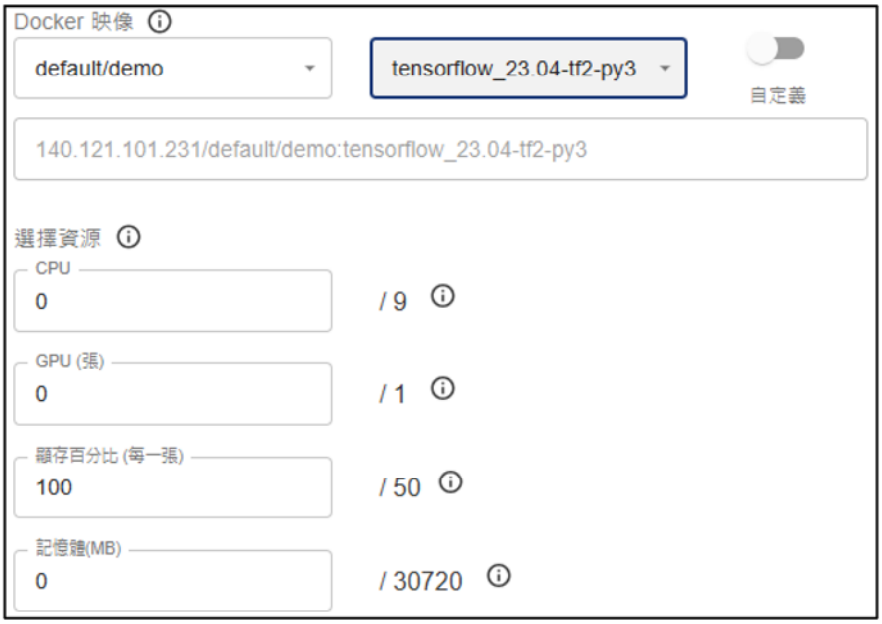
1.模型訓練 -> 提交作業 -> 切換到進階設置頁面 -> 叢集下拉選單 -> 選擇 master_source03 -> 下一個步驟繼續設定



2.往下繼續設定:

Docker 映像:選擇 default/demo，右邊 tag 擇一選取。

選擇資源:斜線右邊的數字為個人最多可使用的資源，例如 CPU 為 72 核心、GPU 為 4 張、顯存百分比 400(換算約為 96GB)、記憶體 245760MB(約為 240GB)。 -> 下一個步驟繼續設定



3. 往下繼續設定:

儲存設定預設已自動選好帳號同名空間，掛載點為容器內的路徑。

以上皆確認無誤之後，點擊畫面最下方的提交按鈕。

4. 若排程內無任何作業在等待時則剛剛建立的作業會立刻運行。

模型訓練 -> 作業管理 在此可找到運行的作業

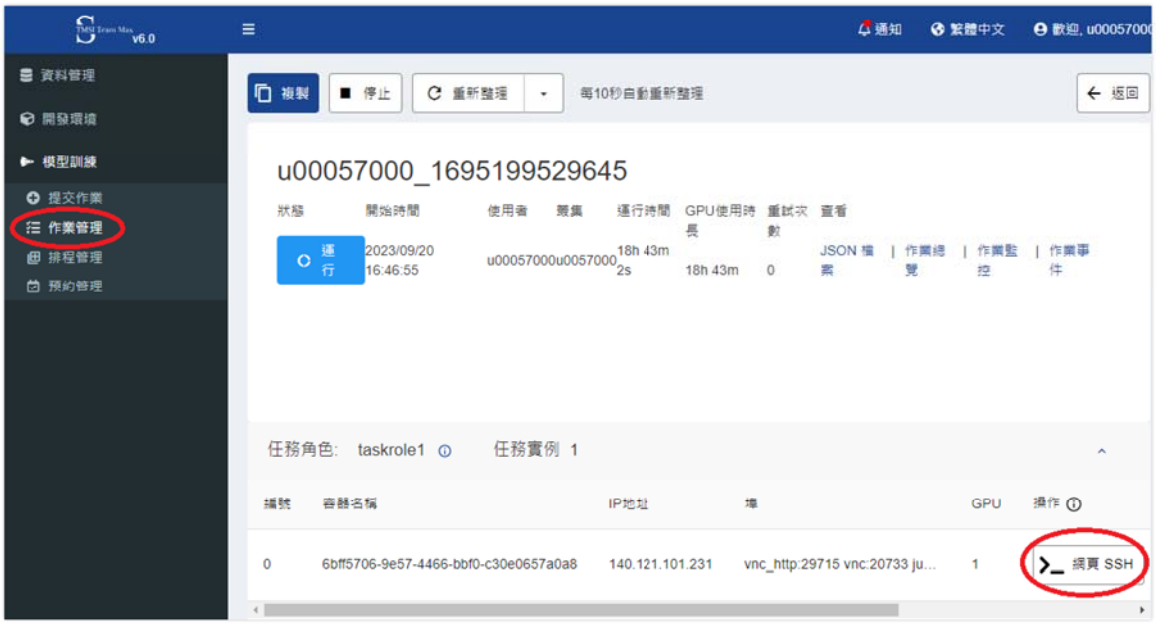
當使用 7 日(168 小時)後，容器作業即自動停止，算力資源釋放讓下一位等待者使用，儲存空間保留至學期末帳號清除時一併刪除。

5. 若系統資源用罄時，則須排隊等待。

模型訓練 -> 排程管理 在此可找到等待中的作業

◎進入容器及操作

1. 模型訓練 -> 作業管理 -> 點擊作業 -> 點擊>_網頁 SSH -> 即可進入容器開始使用。
若>_網頁 SSH為灰色無法點擊時，請耐心等待幾分鐘。



2. 以下範例為執行第 6 頁上傳到集中式儲存裡的 GPU_time.py 檔(模型訓練)。

```
root@ubuntu22node231:~# ls
data  setup.bash
root@ubuntu22node231:~# cd data/
root@ubuntu22node231:~/data# ls
GPU_time.py  clean_GPU_memory.py  print_GPU_info.py  test_torch.py
root@ubuntu22node231:~/data# python GPU_time.py
2024-02-23 12:00:59.919424: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE3 SSE4.1 SSE4.2 AVX, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-02-23 12:01:03.220581: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1638] Created device /job:localhost/replica:0/task:0/device:GPU:0 with 4120 MB memory:  -> device: 0, name: NVIDIA A30, pci bus id: 0000:90:00.0, compute capability: 8.0
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 1s 0us/step
Epoch 1/20
2024-02-23 12:01:06.418724: I tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:655] TensorFloat-32 will be used for the matrix multiplication. This will only be logged once.
2024-02-23 12:01:06.510214: I tensorflow/compiler/xla/service/service.cc:169] XLA service 0x7efe5c790720 initialized for platform CUDA (this does not guarantee that XLA will be used). Devices:
2024-02-23 12:01:06.510279: I tensorflow/compiler/xla/service/service.cc:177]   StreamExecutor device (0): NVIDIA A30, Compute Capability 8.0
2024-02-23 12:01:06.517780: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:269] disabling MLIR crash reproducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTORY` to enable.
2024-02-23 12:01:06.543853: I tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:424] Loaded cuDNN version 8900
2024-02-23 12:01:06.631291: I tensorflow/tsl/platform/default/subprocess.cc:304] Start cannot spawn child process: No such file or directory
2024-02-23 12:01:06.743462: I ./tensorflow/compiler/jit/device_compiler.h:180] Compiled cluster using XLA!  This line is logged at most once for the lifetime of the process.
1875/1875 [==============================] - 13s 6ms/step - loss: 0.2956 - accuracy: 0.9146
Epoch 2/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.1429 - accuracy: 0.9578
Epoch 3/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.1067 - accuracy: 0.9683
Epoch 4/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0878 - accuracy: 0.9722
Epoch 5/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0747 - accuracy: 0.9764
Epoch 6/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0658 - accuracy: 0.9787
Epoch 7/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0572 - accuracy: 0.9812
Epoch 8/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0520 - accuracy: 0.9835
Epoch 9/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0453 - accuracy: 0.9855
Epoch 10/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0439 - accuracy: 0.9857
Epoch 11/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0411 - accuracy: 0.9864
Epoch 12/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0370 - accuracy: 0.9876
Epoch 13/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0353 - accuracy: 0.9883
Epoch 14/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0330 - accuracy: 0.9887
Epoch 15/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0338 - accuracy: 0.9890
Epoch 16/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0304 - accuracy: 0.9897
Epoch 17/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0286 - accuracy: 0.9900
Epoch 18/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0283 - accuracy: 0.9905
Epoch 19/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0251 - accuracy: 0.9914
Epoch 20/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0249 - accuracy: 0.9913

Training time on GPU: 222.60849022865295 seconds
```

以上的操作說明為簡易版，亦建立一般容器的最低設定要求，若您需要進階說明，例如上傳映像檔、獲得開源映像檔、封裝映像檔等等，可參考以下兩個連結。

使用者功能(左側工具列)

Quick Start with User

進階使用

◎透過 VNC、JupyterLab 輕鬆存取容器內部內容

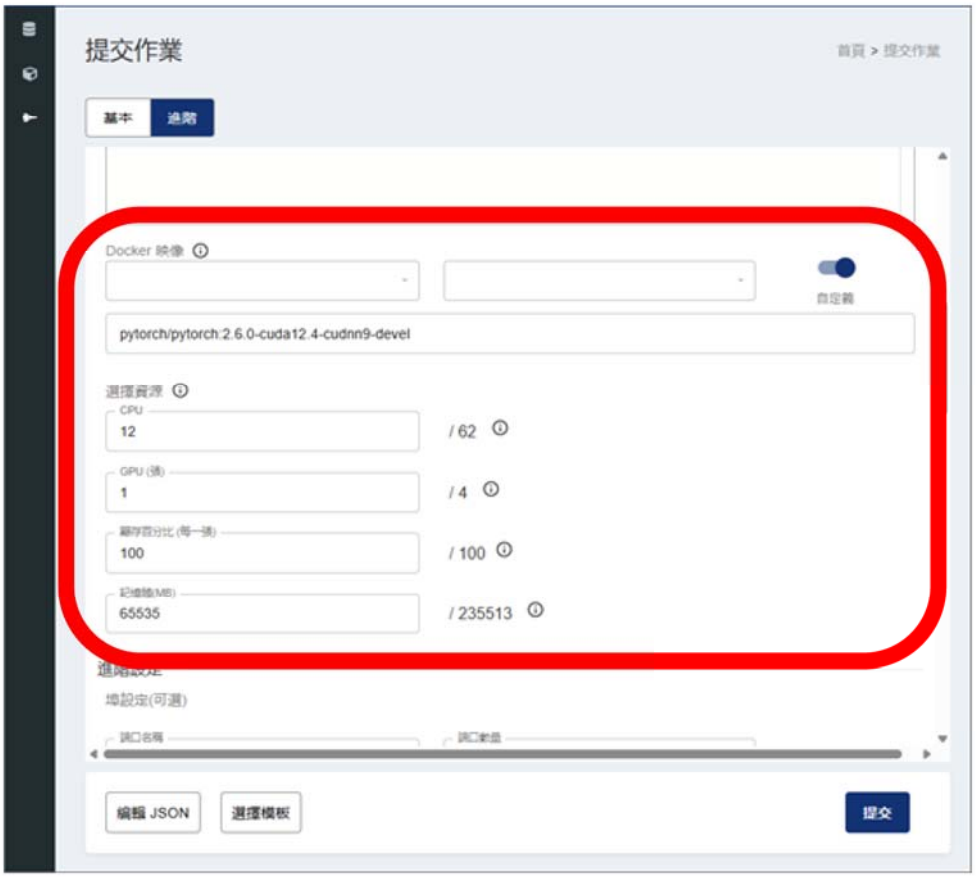
1. 模型訓練 -> 提交作業

2. 使用「進階」模式，選擇可使用「叢集」，並於命令欄輸入:

```
cd
bash setup.bash
sleep infinity
```

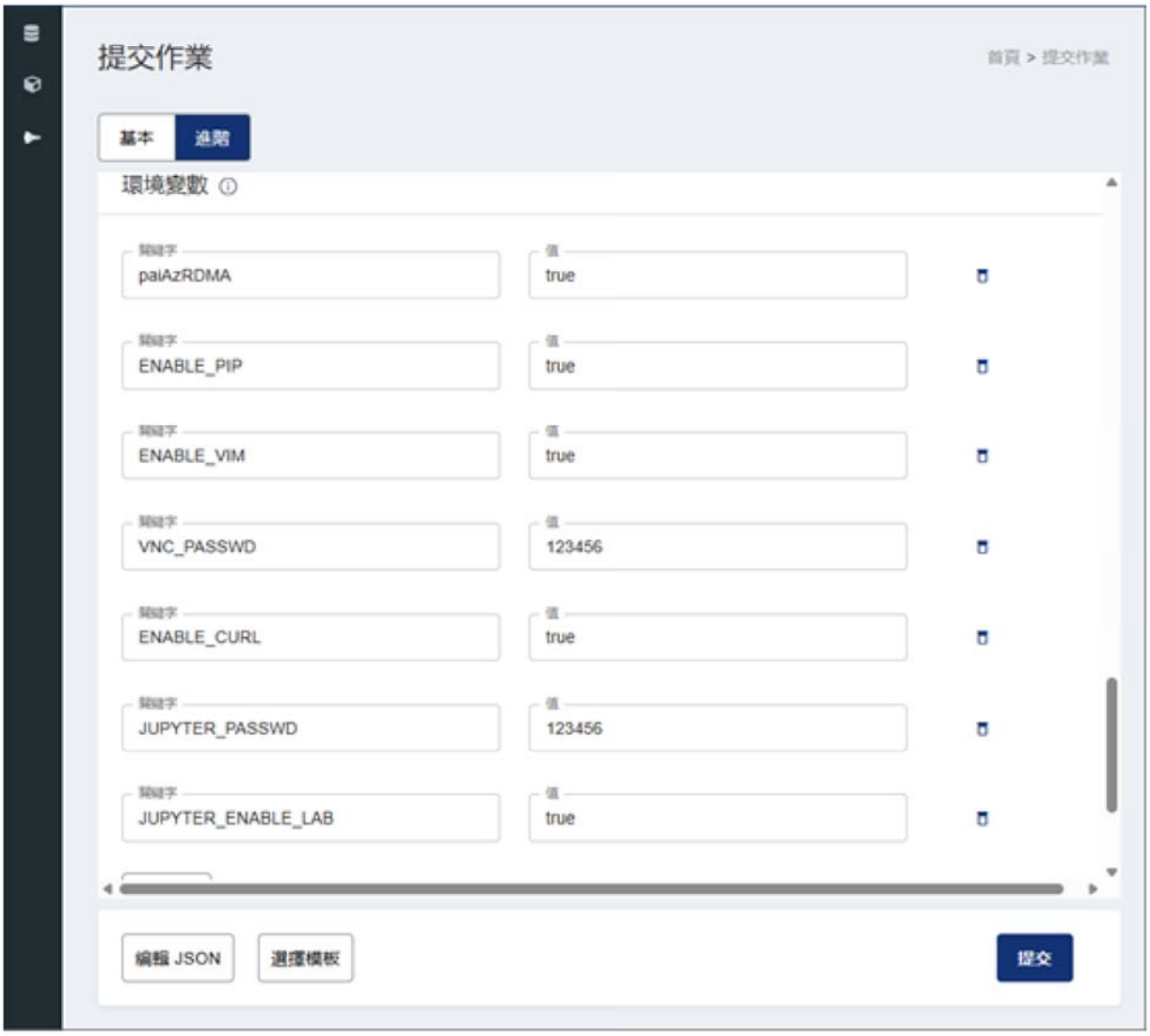
3. 可根據任務自定義 Docker 映像，並輸入合理範圍的資源請求

Docker Hub 搜尋連結: https://hub.docker.com/

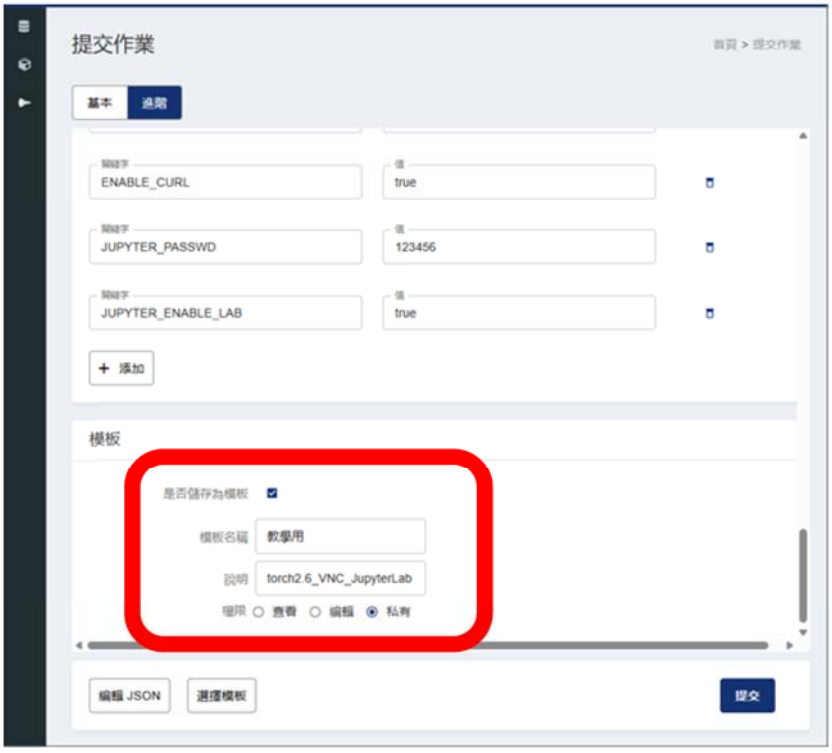


4. 添加「埠設定」，開啟 vnc、vnc_http、jupyter_http、jupyter_lab_http 端口

5. 設定環境變數，密碼可自行更改(長度不可小於 6)

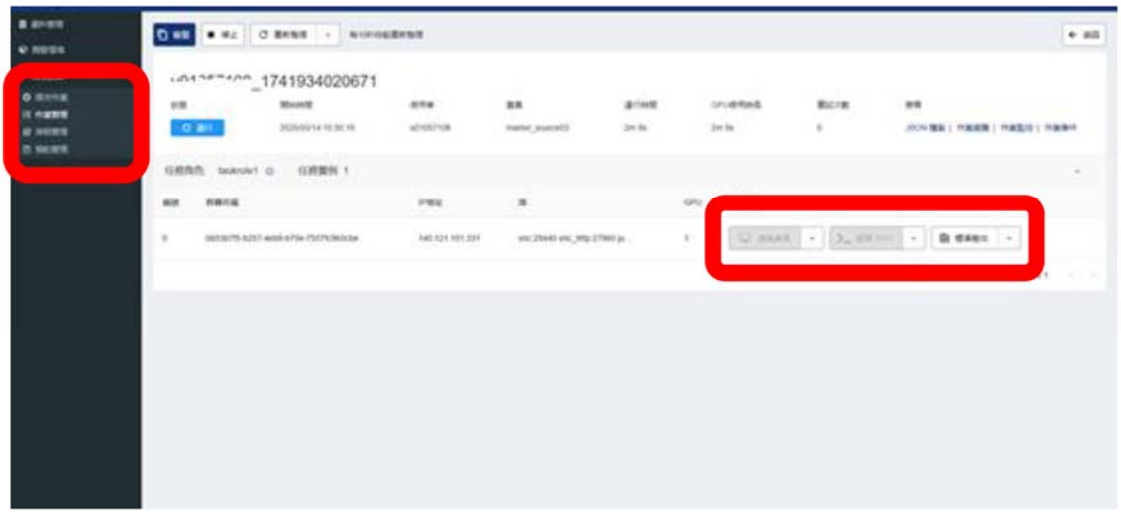


6. 保存為模板，方便以後調用



7.「提交」並「確定」

8. 到「作業管理」找到剛剛提交的作業，等待「遠端桌面」、「網頁 SSH」按鈕亮起，這可能需要 10 分鐘的等待時間

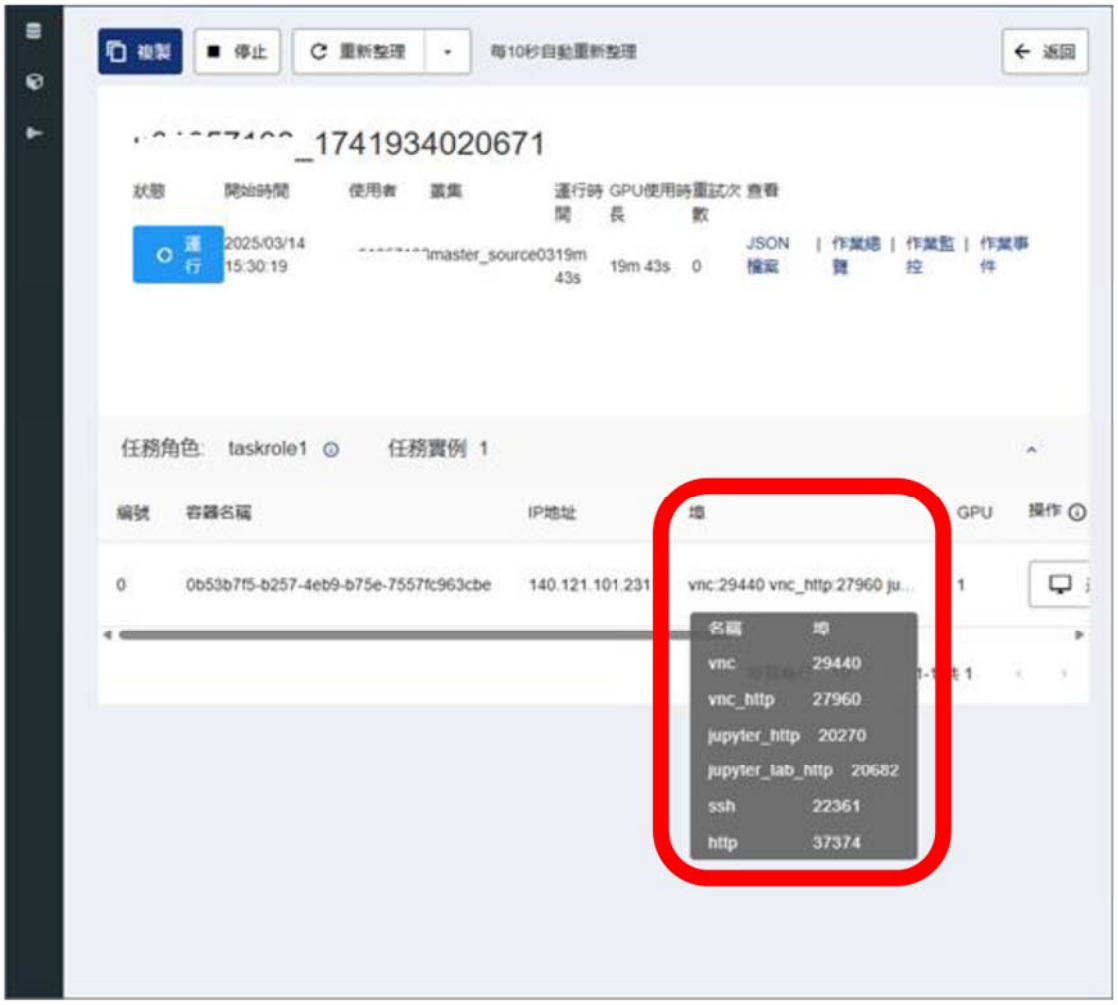


9. 待系統完成安裝，我們成功開啟 SSH 與 VNC 功能，進入「網頁 SSH」依序輸入底下指令安裝 JupyterLab:

apt-get update -y

pip install jupyterlab

10. 將鼠標移置「埠」下方，紀錄「jupyter_lab_http」的端口碼(由隨機產生，本次是 20682)



11. 回到「網頁 SSH」輸入底下指令，port 後面的數字為步驟 10 紀錄的數字

```
jupyter lab --ip=0.0.0.0 --port=20682 --no-browser --allow-root
```

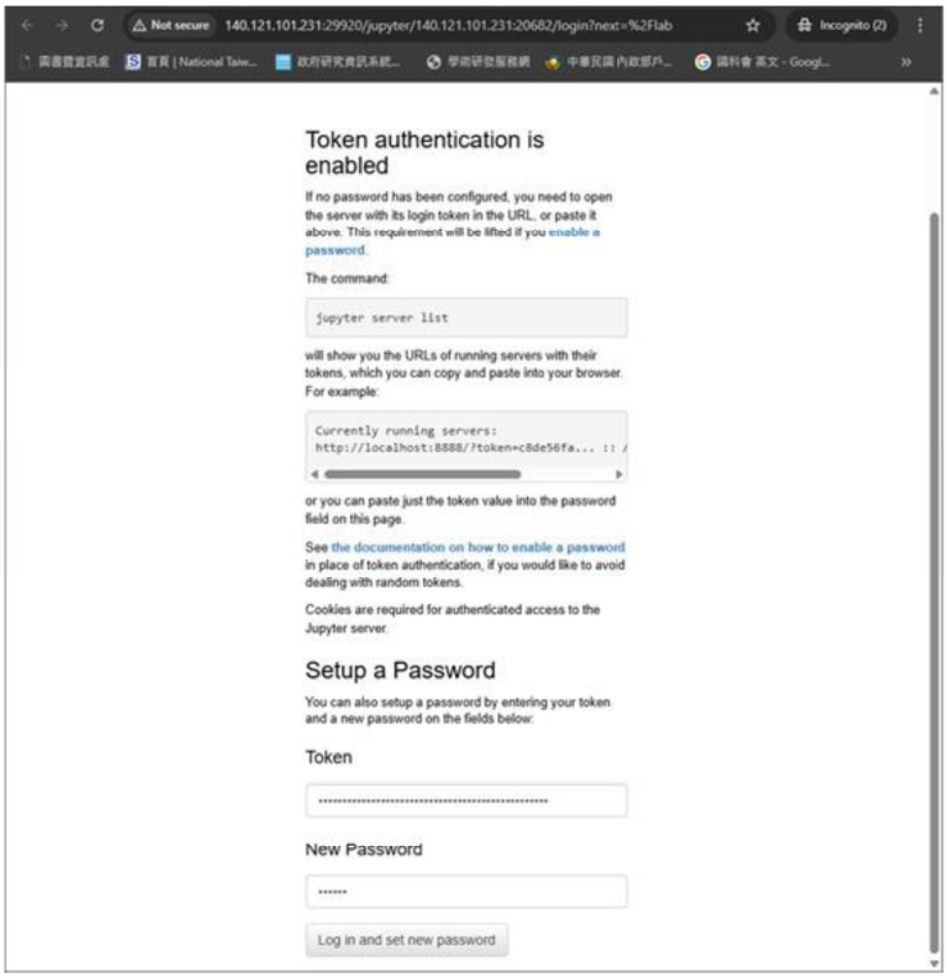
12. 複製紅色框框處的連結，保留 token 後面的代碼

（由隨機產生，本次是 6ea37835a66dffc348ccfdb7306bad5ed2b55b26408b8fd2）

注意：使用時不可關閉此頁面

13. 回到 AI Cloud 平台點選「遠端桌面」旁的按鈕，如果有設定成功，「Jupyter」按鈕會亮起來

14. 進入「Jupyter」，填入步驟 12 紀錄的 token 與設定的密碼登錄



15. 你已成功透過 SSH 設定 JupyterLab，現在就可以開始編寫與執行你的程式了！