

TACC totutorial

TACC group

Songrun Xie: songrun.xie@connect.ust.hk

Yijun Sun: ysune@connect.ust.hk

Xinyu Yang: xinyu.yang@connect.ust.hk

CSIT6000V

Fall 2026

TACC Brief

What does the platform provide?

- Shared compute nodes for CPU and GPU workloads.
- A scheduler allocates resources and runs submitted jobs.
- Course environments and model weights are already available.
- Your code, data and logs remain in your own workspace.

Today's Tutorial

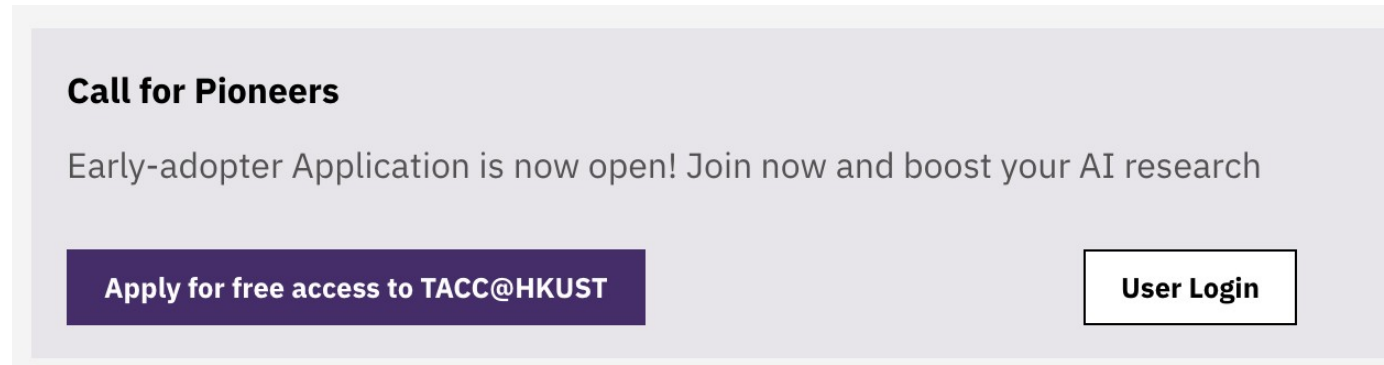
Assignment 1 and the course Project

- Assignment 1: compile MPI code and run the supplied tests.
- Project: serve Qwen on GPUs and call it from your program.
- Keep code and results in your own task workspace.
- Both Assignment and Project use the original tcloud SDK.

TACC Quick Start

Step 1: apply for course access

<https://tacc.ust.hk/#hkust>



1. Open the registration page.

2. Click “Apply for free access”.

3. Complete your application.

4. Mention CSIT6000V in your intended use.

Apply early and wait for your account confirmation email.

TACC Quick Start

Step 2: receive your login credentials

Your TACC Login Credential

TACC Universal Username: `dstking@ust.hk`

The universal username is for both tcloud command line tool and TACC Web Console. To submit tasks to TACC, you need to use [tcloud](#).

Web Console Password: `01mqy0y0d!`

The password is only used for [\[console.tacc.ust.hk/login\]](#)TACC Web Console, which shows your job submission history and status from your web browser.

Universal Username

Use it for tcloud and the web console.

Web Console Password

Use it only for the browser console.

SDK access

Use your matching private key.

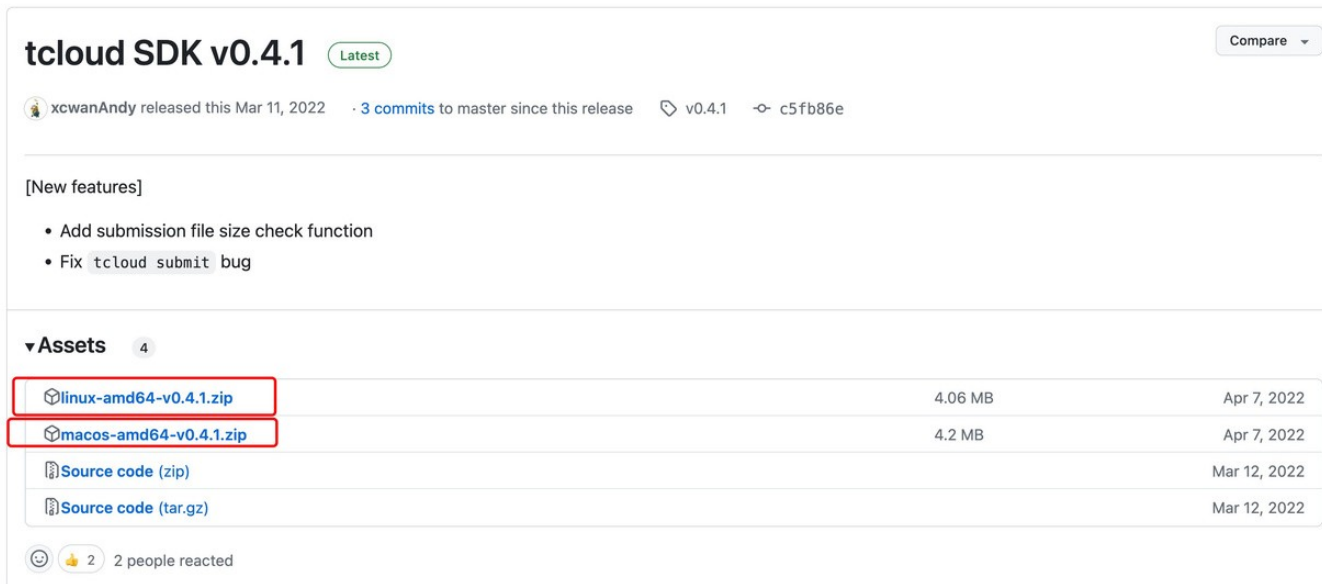
TACC Quick Start

Keep the matching SSH key pair

- Submit the public key; keep the private key on your computer.
- Do not overwrite an existing private key.
- A separate key filename can be used for this platform.
- Configure tcloud with the private key matching your registration.

TACC Quick Start

Download the original SDK



Choose the package for your OS.

Use the official SDK v0.4.1.

No source edits or recompilation are needed.

<https://github.com/turingaicloud/quickstart/releases/tag/v0.4.1>

Original tutorial screenshot; choose your own OS package.

TACC Quick Start

Install and check the local command

Run setup.sh; use your own SDK directory.

```
bash setup.sh
export PATH=/path/to/sdk:$PATH
tcloud
```

- (base) **liyuxuan@gpu09:~/TACC_tutorial/linux-amd64-v0.4.1**\$ bash setup.sh
Remember to execute the following command:
export PATH=/home/liyuxuan/TACC_tutorial/linux-amd64-v0.4.1:/home/liyuxuan/TACC_tutorial/linux-amd64-v0.4.1:/home/liyuxuan/TACC_tutorial/linux-amd64-v0.4.1:/home/liyuxuan/tools/gcc-5.5.0/bin:/home/liyuxuan/anaconda3/envs/ns3/bin:/home/liyuxuan/anaconda3/envs/alpa39/bin:/home/liyuxuan/.vscode-server/bin/af28b32d7e553898b2a91af498b1fb666fdebe0c/bin/remote-cli:/home/liyuxuan/tools/gcc-5.5.0/bin:/home/liyuxuan/anaconda3/envs/ns3/bin:/home/liyuxuan/anaconda3/envs/alpa39/bin:/home/liyuxuan/anaconda3/bin:/home/liyuxuan/anaconda3/condabin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/usr/local/cuda-11.3/bin
- (base) **liyuxuan@gpu09:~/TACC_tutorial/linux-amd64-v0.4.1**\$ export PATH=/home/liyuxuan/TACC_tutorial/linux-amd64-v0.4.1:/home/liyuxuan/TACC_tutorial/linux-amd64-v0.4.1:/home/liyuxuan/TACC_tutorial/linux-amd64-v0.4.1:/home/liyuxuan/tools/gcc-5.5.0/bin:/home/liyuxuan/anaconda3/envs/ns3/bin:/home/liyuxuan/anaconda3/envs/alpa39/bin:/home/liyuxuan/.vscode-server/bin/af28b32d7e553898b2a91af498b1fb666fdebe0c/bin/remote-cli:/home/liyuxuan/tools/gcc-5.5.0/bin:/home/liyuxuan/anaconda3/envs/ns3/bin:/home/liyuxuan/anaconda3/envs/alpa39/bin:/home/liyuxuan/anaconda3/bin:/home/liyuxuan/anaconda3/condabin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/usr/local/cuda-11.3/bin:/usr/local/cuda-11.3/bin

Original example output; replace the old paths with yours.

TACC Quick Start

Check the SDK help output

Run tcloud to confirm that the local command executes.

```
● (base) liyuxuan@gpu09:~/TACC_tutorial/linux-amd64-v0.4.1$ tcloud
TACC Command-line Interface 0.4.1

Usage:
  tcloud [command]

Available Commands:
  add          Add dependency to tuxiv.conf file
  cancel       Cancel job
  cat          Concatenate FILE(s) to standard output.
  config       Configure user's account in tcloud CLI
  download     Download file from TACC
  env          Check environment
  help        Help about any command
```

Original SDK help example; initialization comes next.

TACC Quick Start

Two different “Permission denied” problems

```
⊗ (base) liyuxuan@gpu09:~/TACC_tutorial/linux-amd64-v0.4.1$ tcloud
bash: /home/liyuxuan/TACC_tutorial/linux-amd64-v0.4.1/tcloud: Permission denied

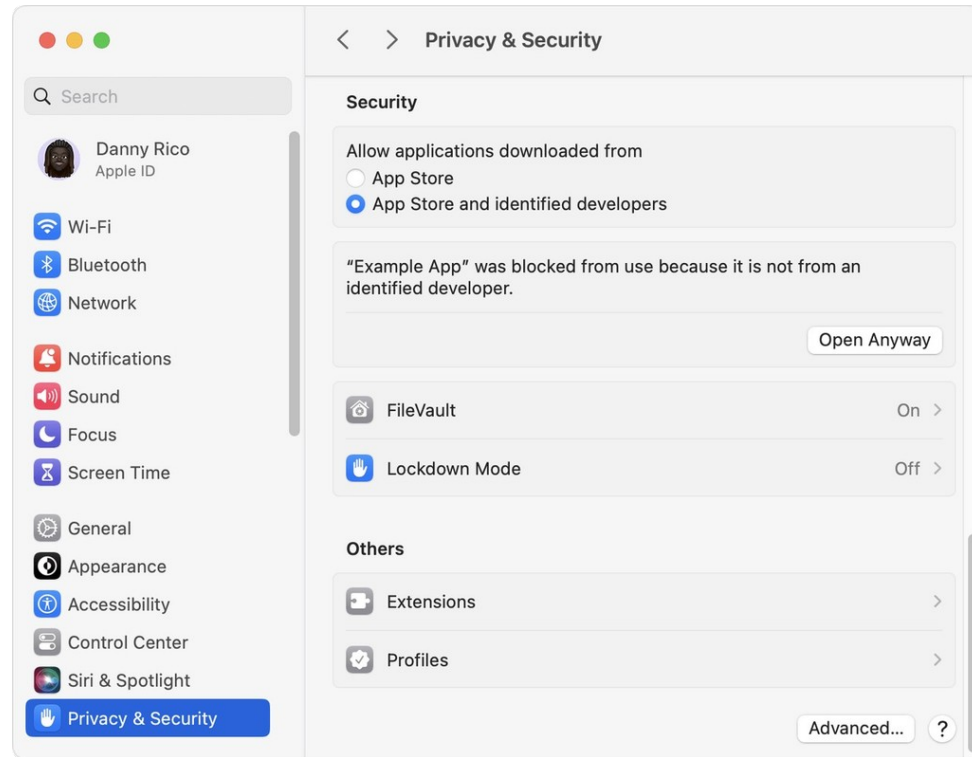
● (base) liyuxuan@gpu09:~/TACC_tutorial/linux-amd64-v0.4.1$ ll
total 8120
drwx----- 2 liyuxuan liyuxuan  4096 Mar  3 11:46 ./
drwxrwxr-x 3 liyuxuan liyuxuan  4096 Mar  3 11:46 ../
-rw-rw-r-- 1 liyuxuan liyuxuan   171 Mar  3 11:46 setup.sh
-rw-rw-r-- 1 liyuxuan liyuxuan 8301941 Mar  3 11:46 tcloud
● (base) liyuxuan@gpu09:~/TACC_tutorial/linux-amd64-v0.4.1$ chmod +x tcloud
● (base) liyuxuan@gpu09:~/TACC_tutorial/linux-amd64-v0.4.1$ tcloud
TACC Command-line Interface 0.4.1
```

chmod +x repairs local executable permission.

For authentication errors, check your username and key.

TACC Quick Start

macOS security prompt



Open Privacy & Security.

Find the blocked tcloud app.

Choose Open Anyway after checking its official source.

Run tcloud again and check the help output.

TACC Quick Start

Configure your credentials

```
tcloud config -u YOUR_USERNAME \  
  -f /path/to/your/private_key  
tcloud init
```

```
● (base) liyuxuan@gpu09:~/TACC_tutorial$ tcloud init  
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST  
tacc*      up    infinite    2    idle 10-0-1-[18-19]
```

Example cluster output; node names can differ.

Local help alone does not confirm platform authentication.

Assignment 1: Shared Environment

Use the prepared MPI toolchain

- GCC, MPICH and Make are in the public environment.
- Use the platform-provided personal Conda manager.
- The actual course environment stays in the public directory.
- Students read and execute it; maintainers update it.

Public environment:

```
/mnt/home/public/course-envs/CSIT6000V/  
assignment1/mpi-gemm-v1
```

Assignment 1: Shared Environment

Register the shared proxy environment

- The platform already provides personal Miniconda.
- Register the empty proxy environment using the helper.
- Each task selects its own prepared runtime environment.

```
bash setup-sdk-account.sh  
tcloud env ls -n proxy
```

Assignment 1: Submit Your Code

Files in your submission directory

- Your `gemm_mpi.c` and the supplied sequential reference.
- `Makefile`, `test.sh`, `dataset/` and `expected/`.
- `tuxiv.conf` selects the entrypoint, resources and environment.
- `sdk-test.sh` activates the public toolchain and runs the tests.

Assignment 1: Submit Your Code

Submit and record the Job ID

- Enter the directory containing your code and tuxiv.conf.
- Keep the returned Job ID for status and log commands.

```
cd /path/to/student-submission  
tcloud submit  
tcloud ps -j JOB_ID
```

Assignment 1: Check the Results

Read the test summary

- Check the job log and the full test output.
- The TA reference run passed 9 tests with 0 failures.
- Your submission must pass the tests with your own implementation.

```
tcloud cat slurm_log/assignment1-sdk-JOB_ID.log
```

Project: Serve Qwen

A complete public model-serving environment

- Public Python, PyTorch, vLLM and model weights are ready.
- Submit a GPU service and a CPU client through the SDK.
- The client sends HTTP requests to the running service.
- Keep the service running until your client finishes.



Project: SDK Starter Kit

Two task folders on your SDK client

- qwen-public-serve/: GPU service configuration and entrypoint.
- qwen-public-client/: your program and server-url.txt.
- setup-sdk-account.sh registers the shared proxy environment.
- No model download or dependency installation is needed.

```
bash setup-sdk-account.sh
```

Project: GPU Resources

Resource requests belong in tuxiv.conf

job:

 general:

- partition=tacc
- nodes=1
- ntasks=1
- cpus-per-task=8
- gres=gpu:2
- time=02:00:00

Project: Submit the Service

Use the original SDK from the service folder

First use: submit from the service task folder.

Today: reuse job 27391; keep it running.

- Use the second slot for CPU demonstrations.

```
SERVICE_JOB_ID=27391
```

```
tcloud ps -j "$SERVICE_JOB_ID"
```

Project: Wait for Readiness

A running job is not yet a ready model

- PD: waiting for resources; R: the job is running.
- Read the service log through tcloud cat.
- Wait for “Application startup complete”.

```
tcloud cat \  
    slurm_log/qwen-public-serve-SERVICE_JOB_ID.log
```

Project: Service Ready

Actual SDK log excerpt

- Service job 27391: one node, two GPUs, TP=2.
- The startup-complete line confirms that the API is ready.

```
$ tcloud cat slurm_log/qwen-public-serve-27391.log  
SERVICE_BASE_URL=http://10-0-1-10:8000/v1  
(APIServer pid=34753) INFO:      Application startup complete.
```

Project: Record the Service Address

Use the address from this job's log

- The log prints SERVICE_BASE_URL=http://NODE:8000/v1.
- Copy the URL after the model is ready.
- Write it into the CPU client task's server-url.txt.

```
printf "%s\n" "http://NODE:8000/v1" \  
> "$CLIENT_DIR/server-url.txt"
```

Project: Your Own Program

Put your logic in qwen-public-client/main.py

```
import os
from qwen_client import chat

result = chat(
    [{"role": "user", "content": "Your prompt"}],
    base_url=os.environ["QWEN_BASE_URL"],
    max_tokens=256, temperature=0.7)
print(result["choices"][0]["message"]["content"])
```

Project: Submit the Client

A separate CPU SDK job calls the service

- The client requests 1 CPU and no GPUs.
- Keep the GPU service running.

```
cd qwen-public-client  
tcloud submit  
tcloud ps -j CLIENT_JOB_ID
```

Project: Read and Save the Results

Look for answers, token usage and elapsed time

- The default examples should answer 95 and 北京 .
- Read the CPU client log, then download it.

```
tcloud cat \  
    slurm_log/qwen-public-client-CLIENT_JOB_ID.log  
tcloud download \  
    slurm_log/qwen-public-client-CLIENT_JOB_ID.log \  
    ./client-result.log
```

Project: Client Results

Actual CPU SDK job output

```
$ tcloud cat slurm_log/qwen-public-client-27393.log
```

```
Question: 37 加 58 等于多少? 只输出数字。
```

```
Answer: 95
```

```
Elapsed seconds: 0.475
```

```
Question: 中国的首都是哪里? 只输出城市名称。
```

```
Answer: 北京
```

```
Elapsed seconds: 0.126
```

Token usage and complete output are saved in the client log.

Project: Model and Request Settings

Use the tested public configuration first

- Qwen3.8-27B: INT4 weights, BF16 computation; text only.
- Two GPUs with TP=2; a 32,768-token context limit.
- The template permits up to 4 concurrent requests.
- max_tokens limits output; temperature controls sampling.

Project: Observe Performance

Separate request time, TTFT and throughput

- The client log reports each complete request's elapsed time.
- TTFT: request sent to first generated output received.
- Non-streaming examples cannot measure TTFT.
- Record input/output tokens, concurrency and cache state.

Project: Troubleshooting

Check the two Job IDs and the service log

- Use `tcloud ps` and `cat` for both service and client.
- Check readiness and the latest `SERVICE_BASE_URL`.
- Check `server-url.txt` and the client error message.
- Do not reinstall shared packages to repair an address error.

```
tcloud ps -j SERVICE_JOB_ID  
tcloud ps -j CLIENT_JOB_ID
```

Project: Stop the Service

Release GPU resources after class

- Keep job 27391 running during this class.
- After class, cancel the recorded service Job ID.
- Keep your local source and downloaded results.

```
tcloud cancel -j SERVICE_JOB_ID  
tcloud ps -j SERVICE_JOB_ID
```

Project: Run Again

Reuse the environment; start a new service job

- Future runs reuse the public weights and runtime.
- Submit a new service; wait and refresh the address.
- Valid per-user compile caches can be reused.
- Today: keep using job 27391; do not resubmit.

```
cd ../qwen-public-serve  
tcloud submit
```

Q & A

Songrun Xie: songrun.xie@connect.ust.hk

Yijun Sun: ysune@connect.ust.hk

Xinyu Yang: xinyu.yang@connect.ust.hk