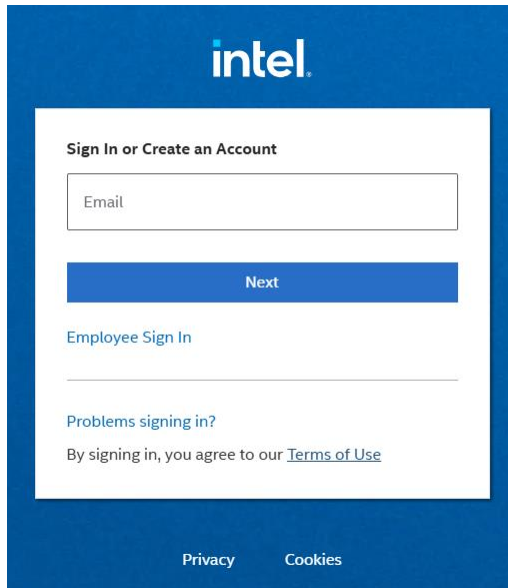


Migrating AI Applications to SYCL with SYCLomatic

How to Login to Intel® Tiber™ AI Cloud

- 1) Sign in to <https://console.cloud.intel.com/> (if you have an existing account) or create a new account

The image shows the Intel sign-in page. It has a blue header with the Intel logo. Below the header is a white box with the title "Sign In or Create an Account". Inside this box is an "Email" input field, a blue "Next" button, and a link for "Employee Sign In". Below the white box, there are links for "Problems signing in?" and "By signing in, you agree to our Terms of Use". At the bottom of the page are links for "Privacy" and "Cookies".

intel

Sign In or Create an Account

Email

Next

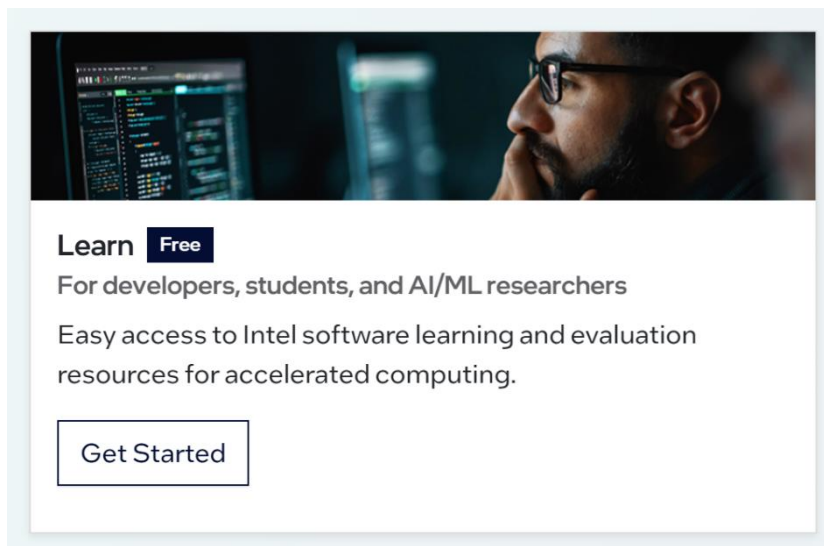
Employee Sign In

Problems signing in?

By signing in, you agree to our [Terms of Use](#)

Privacy Cookies

- 2) Please note that you will need a mobile phone number to complete the registration. Two-factor authentication will be used to log in. Click on Get Started under Learn tab

The image is a banner for the "Intel Learn Free" program. It features a background image of a man with glasses looking at a computer screen. The text on the banner reads: "Learn Free", "For developers, students, and AI/ML researchers", "Easy access to Intel software learning and evaluation resources for accelerated computing.", and a "Get Started" button.

Learn **Free**

For developers, students, and AI/ML researchers

Easy access to Intel software learning and evaluation resources for accelerated computing.

Get Started

- 3) Select 'Connect Now' and GPU

Learn Free

For developers, students, and AI/ML researchers

Gain mastery in AI and accelerated computing with Jupyter notebooks running on Intel GPUs and AI accelerators.

Start now

Use our shared environments to discover what Intel enables you to do. Get access to Intel GPUs or Intel AI accelerators.

Connect now ▾

New

AI Accelerator

GPU

Learning paths

Gain mastery in AI and cutting-edge topics with our Jupyter notebooks running on GPUs and AI accelerators

Learning

4) Launch the Jupyter Notebook:

Launching notebook

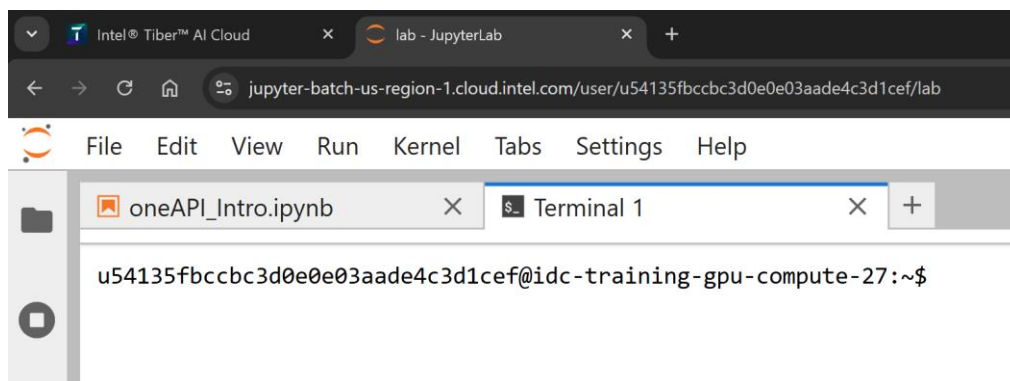


To ensure optimum security, you'll be asked to sign in before the notebook loads.

Launch

5) Note that you will be asked to enter your credentials one more time (including verification via SMS).

The terminal windows look like this:



6) If, for some reason, it is not opened for you, please select File -> New -> Terminal.

CMake-based Project Migration: DL-MNIST

Prepare the environment for the migration

Download the `set_env.tgz` package, it contains scripts to set the env for DL-MNIST migration.

- 1) `wget https://github.com/ivorobts/IWOCL2025/raw/refs/heads/main/set_env.tgz`
- 2) `tar -zxvf set_env.tgz`

The script will download Velocity-Bench (which includes DL-MNIST code) and install the required dependencies via Conda for the original code compilation (~ 5 mins).

- 3) `source ~/set_env/dl-mnist/set_env.sh`

Migrate the project to SYCL

We prepare a Makefile via CMake and then intercept compilation commands via the `intercept-build` tool:

- 4) `mkdir build && cd build`
- 5) `cmake ..`
- 6) `intercept-build make -B`

Once the compilation database file is generated `compile_commands.json`, we may proceed with the project migration to SYCL:

- 7) `cd ~/Velocity-Bench/dl-mnist`
- 8) `dpct --in-root .. -p ./CUDA/build --out-root out --migrate-build-script=CMake`

Build and run the SYCL code project

Need to prepare a data file required to run the DL-MNIST workload

- 9) `cd out/dl-mnist; mkdir datasets; cp ~/set_env/dl-mnist/train-images.idx3-ubyte datasets/`

Build the application via the migrated CMake file:

- 10) `cd CUDA; mkdir build && cd build; cmake -DCMAKE_CXX_COMPILER=icpx ..`
- 11) Some minor code modifications are required before the compilation, e.g.
 - `vi ../CMakeLists.txt` (comment out line 82: `#add_compile_options(-DUSE_CUDA)`)
 - `vi ../main.cudnn.cpp.dp.cpp` (comment out line 68: `//Utility::QueryCUDADevice();`)
- 12) `make -j`

Run the migrated version of the application:

- 13) `SYCL_UR_TRACE=1 ./dl-mnist-cuda -conv_algo CUDNN_FIND_BEST_ALGO`

PyTorch-based Project Migration: Stylegan3

Prepare the environment for the migration

Download the `set_env.tgz` package (if not done earlier). It contains scripts to set the env for stylegan3 migration.

- 1) `wget https://github.com/ivorobts/IWOCL2025/raw/refs/heads/main/set_env.tgz`
- 2) `tar -zxvf set_env.tgz`

The script will download the stylegan3 code and required CUDA header files used during the migration and modify© the compilation database file `compile_commands.json` (~ 3 mins). Usually, this file should be generated/intercepted while compiling the original CUDA code. This file is prepared to be used on Intel® Tiber™ AI Cloud since the environment/hw to build the CUDA version of stylegan3 is not available.

- 3) `source ~/set_env/stylegan3/set_env.sh`

Migrate the project to SYCL

The two custom migration rule files contain required code changes for the Python build scripts and source code migration. Each fix is root-caused after the code migration step and identified during the code build stage. For this tutorial, each change is organized in a YAML file to demonstrate that the manual code changes can be organized in the user-defined migration rules files.

Note that step 1 is optional; you also can fix the migrated SYCL code/Python scripts during the build step.

- 1) `cp ~/set_env/stylegan3/stylegan3_specific_migration_rule_v0.yaml .`
`cp ~/set_env/stylegan3/stylegan3_specific_python_build_script_migration_rule.yaml .`

Run the `SYCLomatic` command to migrate the source code. The CUDA source files listed in `compile_commands.json` will be migrated into SYCL code. After migration, all SYCL code will be available in the **out** folder.

- 2) `dpct -p . --in-root . --out-root out --use-experimental-features=free-function-queries,local-memory-kernel-scope-allocation` *I am running a few minutes late; my previous meeting is running over.*
`--rule-file ~/syclomatic_package/extensions/pytorch_api_rules/pytorch_api.yaml \`
`--analysis-scope-path ~/.conda/envs/stylegan3/lib/python3.9/site-packages/torch/include \`
`--analysis-scope-path . --rule-file ./stylegan3_specific_migration_rule_v0.yaml \`
`--cuda-include-path=$HOME/cudaheaders`

Run the `SYCLomatic` command to migrate Python build scripts. After migration, SYCL-capable Python build scripts will be available in the **out** folder.

- 3) `dpct -p . --in-root . --out-root out \`
`--rule-file ~/syclomatic_package/extensions/python_rules/python_build_script_migration_rule_pytorch.yaml \`
`--migrate-build-script=Python --migrate-build-script-only \`
`--rule-file ./stylegan3_specific_python_build_script_migration_rule.yaml`

Build and run the SYCL code project

Set the environment to build the migrated code. This includes creating a `stylegan3_sycl` Conda environment, removing the default PyTorch with no Intel XPU support, and installing a new version of PyTorch with Intel XPU support (via <https://download.pytorch.org/whl/test/xpu>).

1) `source ~/set_env/stylegan3/env_to_run.sh`

Run the migrated version of the application:

- 2) `python3 gen_images.py --outdir=out_image --trunc=1 --seeds=2 --
network=https://api.ngc.nvidia.com/v2/models/nvidia/research/stylegan3/versions/1/files/stylegan3-r-afhqv2-512x512.pkl`
- 3) `python gen_video.py --output=lerp.mp4 --trunc=1 --seeds=0-31 --grid=4x2 \`
`--network=https://api.ngc.nvidia.com/v2/models/nvidia/research/stylegan3/versions/1/files/stylegan3-r-afhqv2-512x512.pkl`