1. Login
   http://account.cscs.ch/

   Follow the instructions in it. Will have to create an authentication key every 24 hours to login.
   Ssh todi

2. Git clone the swiss AI nanotron and pretrain packages at the path:- "/iopsstor/scratch/cscs/$USER"
3. Use the nanotron/examples/llama/convert_hf_to_nanotron.py to convert HF weights to nanotron format. SBATCH script:-

```
#!/bin/bash
#SBATCH --job-name=Chemllama-convert
#SBATCH --account=a05
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:29:59
#SBATCH --partition=normal
#SBATCH --no-requeue
#SBATCH --gpus-per-task=4

export MASTER_ADDR=$(hostname)
export MASTER_PORT=25678
export CUDA_DEVICE_MAX_CONNECTIONS=1
set -eo pipefail

export ENROOT_LIBRARY_PATH=/capstor/scratch/cscs/fmohamed/enrootlibn

srun -ul --container-writable --environment=/store/swissai/a06/containers/nanotron_pretrain/nanotron_pretrain.toml bash -c "
cd /iopsstor/scratch/cscs/ssenthil/nanotron
pip install -e '.[nanosets]' --no-dependencies
torchrun --nproc_per_node=1 ./examples/llama/convert_hf_to_nanotron.py
--checkpoint_path=/store/swissai/a06/models/Meta-Llama-3.1-8B  --save_path=/home/ssenthil/nanotron_weights
"
```

4. And also have the dataset to train in the right directory in the hugging face csv format, with proper README.

5. Add this following nanotron training in the folder config_dir/nanotron

```
checkpoints:
  checkpoint_interval: 25
  checkpoints_path: /work/liac/shai/checkpoints/
  checkpoints_path_is_shared_file_system: false
  resume_checkpoint_path: null
  save_initial_state: false
data_stages:
- data:
    dataset:
      hf_dataset_or_datasets: /work/liac/shai/chemrxiv
      hf_dataset_splits: train
      text_column_name: text
```

```
    num_loading_workers: 10
    seed: 42
  name: Chemrxiv dataset (Single dataset)
  start_training_step: 1
general:
  benchmark_csv_path: null
  consumed_train_samples: null
  ignore_sanity_checks: true
  project: Chemllama
  run: Llama3.1_8B_Chemmllama
  seed: 42
  step: null
lighteval: null
logging:
  iteration_step_info_interval: 1
  log_level: info
  log_level_replica: info
model:
  ddp_bucket_cap_mb: 25
  dtype: bfloat16
  init_method:
    path: /work/liac/shai/checkpoints/0
  make_vocab_size_divisible_by: 1
  model_config:
    bos_token_id: 1
    eos_token_id: 2
    hidden_act: silu
    hidden_size: 4096
    initializer_range: 0.02
    intermediate_size: 14336
    is_llama_config: true
    max_position_embeddings: 8192
    num_hidden_layers: 32
    num_attention_heads: 32
    num_key_value_heads: 8
    pad_token_id: null
    pretraining_tp: 1
    rope_interleaved: false
    rope_theta: 500000.0
    rms_norm_eps: 1.0e-06
    rope_scaling: null
    tie_word_embeddings: false
    use_cache: true
    vocab_size: 128256
optimizer:
  optimizer_factory:
    name: adamW
    adam_eps: 1.0e-08
    adam_beta1: 0.9
    adam_beta2: 0.95
    torch_adam_is_fused: true
  learning_rate_scheduler:
    learning_rate: 3.0e-05
    lr_decay_starting_step: null
    lr_decay_steps: 400
    lr_decay_style: cosine
    lr_warmup_steps: 70
```

```
    lr_warmup_style: linear
    min_decay_lr: 3.0e-06
  zero_stage: 0
  clip_grad: 1.0
  weight_decay: 0.01
  accumulate_grad_in_fp32: true
parallelism:
  dp: 1
  expert_parallel_size: 1
  pp: 1
  tp: 8
  tp_linear_async_communication: true
  tp_mode: REDUCE_SCATTER
profiler: null
tokenizer:
  tokenizer_max_length: null
  tokenizer_name_or_path: /work/liac/shai/Meta-Llama-3.1-8B
  tokenizer_revision: null
tokens:
  batch_accumulation_per_replica: 4
  limit_test_batches: 0
  limit_val_batches: 0
  micro_batch_size: 4
  sequence_length: 8192
  train_steps: 3725
  val_check_interval: -1
```

Make sure total GPUs (world size) = TP*DP*PP

And for calculating the train_steps (limited to one epoch only, otherwise nanotron will throw error. Should load the dataset twice or more for larger than 1 epoch.)

```
# number of sequences: mbsz * baccum * dp = 8 * 4 * 1 = 32
# full bsz: seqlen * mbsz * baccum * dp = 8192 * 32 = 262144
# one epoch in steps: 1000688526(total tokens) / 262144 = 3817
```

6. For benchmarking we use lighteval framework for automatic benchmarking.

We will have to add a python script to decide how to evaluate the model:-

Chemllm_tasks.py :

```
import re
from typing import List, Tuple
from lighteval.metrics.metrics import Metrics
from lighteval.tasks.lighteval_task import LightevalTaskConfig
from lighteval.tasks.requests import Doc
from lighteval.tasks.tasks_prompt_formatting import LETTER_INDICES
_TASKS_STRINGS: List[Tuple[LightevalTaskConfig, str]] = []
_TASKS: List[LightevalTaskConfig] = []
CHEMICAL_REASONING_TASKS = [
LightevalTaskConfig(
        name="organic_qa",
        prompt_function="organic_qa_prompt",
```

```
        hf_repo="shaipranesh/chembench",
        hf_subset="",
        trust_dataset=True,
        metric=["loglikelihood_acc", "loglikelihood_acc_norm_nospace"],
    ),]
def organic_qa_prompt(line, task_name: str = None):
    query = "MCQ Question: "
    query+=line['question']
    query+="\n"
    j=0
    for i in LETTER_INDICES[:len(line["options"])]:
        query+=i
        query+=". "
        query+=line["options"][j]
        query+="\n"
        j+=1
    query+="\n"
    query+="Answer (Give only the correction option letter, no need any other text):"
    return Doc(
        task_name=task_name,
        query=query,
        choices=LETTER_INDICES[:len(line["options"])],
        gold_index=line["answer"][0],
    )
```

config_dir/light_eval

```
batch_size: 4
checkpoints_path: null
generation: null
logging:
  hub_repo_details: null
  hub_repo_results: null
  hub_repo_tensorboard: null
  local_output_path: lighteval_chemllm
  push_details_to_hub: null
  push_results_to_hub: null
  push_results_to_tensorboard: null
  tensorboard_metric_prefix: null
parallelism:
  dp: 4
  expert_parallel_size: 1
  pp: 1
  pp_engine: 1f1b
  tp: 1
  tp_linear_async_communication: true
  tp_mode: REDUCE_SCATTER
slurm_script_dir: null
slurm_template: null
tasks:
  # custom_tasks: /capstor/scratch/cscs/${oc.env:USER}/lighteval/smollm_tasks.py
  # custom_tasks: /workspace/nanotron/lighteval/smollm_tasks.py
  custom_tasks: lighteval/chemllm_tasks.py
  dataset_loading_processes: 16
  max_samples: 1000
  multichoice_continuations_start_space: null
```

```
  no_multichoice_continuations_start_space: null
  num_fewshot_seeds: null
  tasks: custom|organic_qa|0|1
wandb: null
```

7. Change other job parameters in config_dir/run accordingly.

8. Change the config_dir/confi.yaml accordingly:-

```
defaults:
  - nanotron: llama3.1_chemrxiv
  - run: todi_normal
  - lighteval: chemllama
```

9. Follow the steps on swiss-ai pretrain and launch ur job!

10. Otherwise to run only pretraining u can directly launch a job like this:-

```
#!/bin/bash
#SBATCH --job-name=Chemllama
#SBATCH --account=a05
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=5:59:00
#SBATCH
--output=/iopsstor/scratch/cscs/ssenthil/my_nanotron_runs/Chemllama/Llama3.1_8B_DummyRun_20241009172135334677
_7b7ead9_clean/output.log
#SBATCH
--error=/iopsstor/scratch/cscs/ssenthil/my_nanotron_runs/Chemllama/Llama3.1_8B_DummyRun_20241009172135334677_
7b7ead9_clean/output.err
#SBATCH --partition=normal
#SBATCH --no-requeue
#SBATCH --gpus-per-task=4
#SBATCH --reservation=sai-a05
export MASTER_ADDR=$(hostname)
export WANDB_DIR=/iopsstor/scratch/cscs/ssenthil/my_nanotron_runs/wandb_dir
export WANDB_API_KEY=d25f82312adf534313b41219af6dfb51175458da
export HF_TOKEN=None
export MASTER_PORT=25678
export CUDA_DEVICE_MAX_CONNECTIONS=1
set -eo pipefail
export ENROOT_LIBRARY_PATH=/capstor/scratch/cscs/fmohamed/enrootlibn
srun -ul --container-writable --environment=/store/swissai/a06/containers/nanotron_pretrain/nanotron_pretrain.toml bash -c "
cd /iopsstor/scratch/cscs/ssenthil/nanotron
export
WANDB_DIR=/iopsstor/scratch/cscs/ssenthil/my_nanotron_runs/Chemllama/Llama3.1_8B_DummyRun_202410091721353
34677_7b7ead9_clean/wandb_logs
export HF_TOKEN=None
pip install -e '.[nanosets]' --no-dependencies
cd /iopsstor/scratch/cscs/ssenthil/nanotron
TORCHRUN_ARGS=\"
--node-rank=\$SLURM_PROCID \
--master-addr=\$MASTER_ADDR \
--master-port=\$MASTER_PORT \
--nnodes=\$SLURM_NNODES \
```

```
--nproc-per-node=\$SLURM_GPUS_PER_TASK \
\"
torchrun \$TORCHRUN_ARGS run_train.py --config-file
/iopsstor/scratch/cscs/ssenthil/my_nanotron_runs/Chemllama/Llama3.1_8B_DummyRun_20241009172135334677_7b7ead
9_clean/nanotron_config.yaml"
```