HPC_TU

1. Verbindung zum HPC-Cluster herstellen

- Öffne das Terminal in VS Code / alternativ unten links auf SSH Verbindung klicken
- Stelle die SSH-Verbindung her:

```
ssh <TU-Anmeldename>@gateway.hpc.tu-berlin.de
```

2. Miniconda installieren

- Lade das Miniconda-Installationsskript herunter über das VS-Code Terminal:

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

- Führe das Installationsskript aus:

```
bash Miniconda3-latest-Linux-x86_64.sh
```

- Folge den Anweisungen im Installationsskript. Akzeptiere die Lizenzvereinbarung und wähle das Installationsverzeichnis (z.B. `~/miniconda3`).
- Initialisiere Miniconda:

```
source ~/miniconda3/bin/activate
```

3. Neue Conda-Umgebung erstellen (Alternativ) und Pakete installieren

- Erstelle eine neue Conda-Umgebung mit Python:

```
conda create -n myenv python=3.8
```

- Aktiviere die neue Umgebung:

```
conda activate myenv
```

- Installiere die benötigten Pakete (z.B. `numpy`, `pandas`):

```
conda install numpy pandas
```

4. Einfaches Python-Skript erstellen

- Erstelle eine Datei `simple_calculation.py`:

```
import csv

import os

def main():

    a = 5

    b = 3

    result = a + b

    print(f"The result of {a} + {b} is {result}")

    # Verzeichnis überprüfen und Debugging-Ausgabe hinzufügen

    output_dir = os.getcwd()
```

```
    print(f"Current working directory: {output_dir}")

    # CSV-Datei erstellen und Daten schreiben

    csv_file_path = os.path.join(output_dir, 'calculation_result.csv')

    print(f"CSV file path: {csv_file_path}")

    try:

        with open(csv_file_path, mode='w', newline='') as file:

            writer = csv.writer(file)

            writer.writerow(['a', 'b', 'result'])

            writer.writerow([a, b, result])

        print("CSV file created successfully.")

    except Exception as e:

        print(f"Failed to create CSV file: {e}")

if __name__ == "__main__":

    main()
```

5. SLURM-Skript erstellen

- Erstelle eine Datei `run_python_job.sh`:

```
#!/bin/bash

#SBATCH -o myjob.%j.%N.out   # Output-File im aktuellen Verzeichnis

#SBATCH -D .                 # Working Directory auf das aktuelle Verzeichnis setzen

#SBATCH -J Hello-World_GPU   # Job Name

#SBATCH --ntasks=1           # Anzahl Prozesse P (CPU-Cores)

#SBATCH --cpus-per-task=1    # Anzahl CPU-Cores pro Prozess P

#SBATCH --mem=1G             # 1GiB resident memory pro node

#SBATCH --time=00:02:00      # Erwartete Laufzeit auf 2 Minuten setzen

#SBATCH --partition=gpu_short # Auf GPU-Knoten in der gpu_short Partition rechnen

#SBATCH --mail-type=ALL      # Job-Status per Mail

#SBATCH --mail-user=vorname.nachname@tu-berlin.de

# Miniconda initialisieren

source ~/miniconda3/bin/activate

# Conda-Umgebung aktivieren (base)

conda activate base          # hier die Umgebung eingeben base / myenv etc.

# Python-Skript ausführen
```

```
python simple_calculation.py
```

6. Job einreichen

Python Script und SLURM Skript in einen Ordner legen

- Reiche den Job mit `sbatch` ein:

```
sbatch run_python_job.sh
```

7. Job-Status überprüfen

- Zeige alle laufenden und wartenden Jobs des aktuellen Benutzers an:

```
squeue -u $USER
```

- Zeige Details zu einem bestimmten Job an:

```
scontrol show job <JobID>
```

- Zeige die Job-Historie an:

```
sacct -j <JobID>
```

- Zeige den Output-File an:

```
cat myjob.<JobID>.<NodeName>.out
```

8. Status der Nodes anzeigen

- Zeige den Status aller Nodes im Cluster an:

```
sinfo -N -l
```

- Zeige Details zu den Nodes an:

```
scontrol show nodes
```

9. CSV-Datei überprüfen

- Überprüfe, ob die CSV-Datei im aktuellen Verzeichnis erstellt wurde:

```
ls -l calculation_result.csv

cat calculation_result.csv
```

Zusammenfassung der SLURM-Befehle

- **Job einreichen**: `sbatch run_python_job.sh`
- **Job-Status anzeigen**: `squeue -u $USER`
- **Details zu einem Job anzeigen**: `scontrol show job <JobID>`
- **Job-Historie anzeigen**: `sacct -j <JobID>`
- **Job-Output anzeigen**: `cat myjob.<JobID>.<NodeName>.out`
- **Nodes-Status anzeigen**: `sinfo -N -l`
- **Details zu den Nodes anzeigen**: `scontrol show nodes`