FOSS4GNL 2026

8-9 July 2026, Groningen, The Netherlands

Benchmarking Geospatial Workflows with Geobench: A Hands-on Tutorial

Dr. Ing. Serkan Girgin MSc[1,2]
s.girgin@utwente.nl
https://linkedin.com/in/serkan-girgin/

[1] Center of Expertise on Big Geodata Science
[2] Department of Geoinformatics
Faculty of Geo-information Science and Earth Observation (ITC)

UNIVERSITY OF TWENTE. | ITC

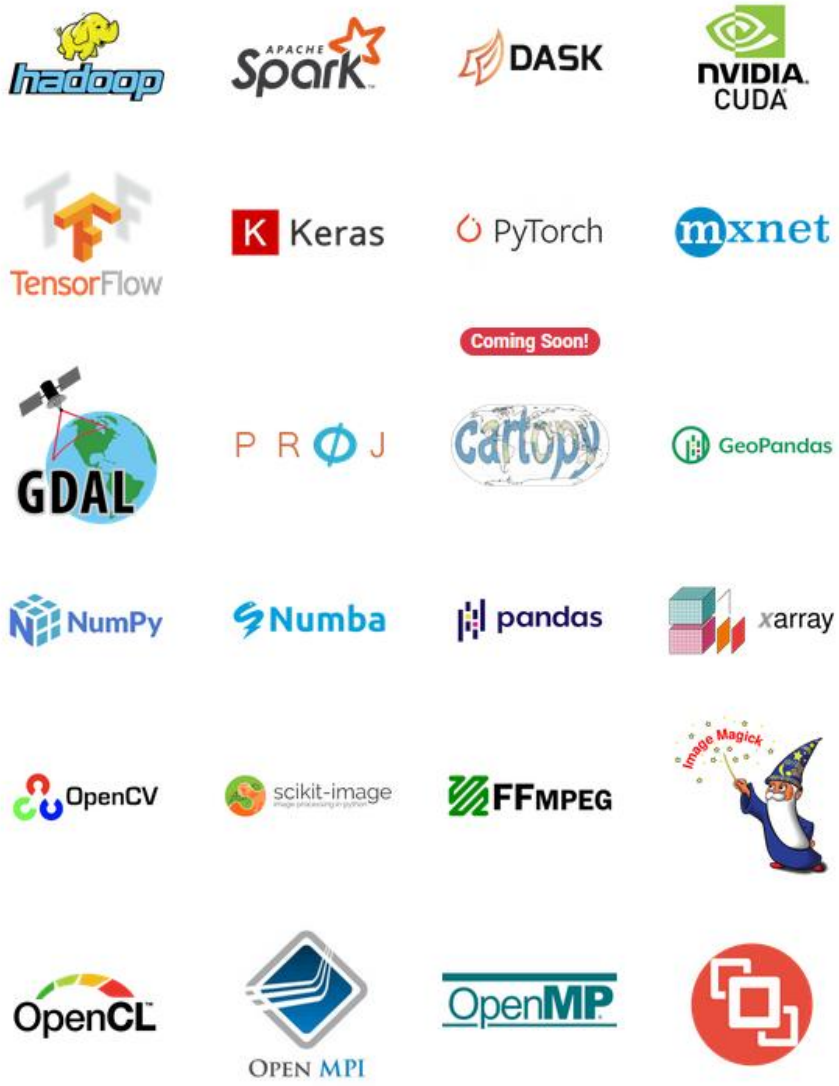
Geospatial data is getting bigger and more difficult to analyse

- Satellites, drones, vehicles, social networks, mobile devices, cameras, etc. **generate vast amount of (open) geospatial data**.
- Large and complex geospatial datasets are increasingly being **analyzed using advanced, computationally intensive methods.**
- Data processing and analysis tasks are **time consuming**, **sometimes even not possible**, depending on the computing infrastructure.

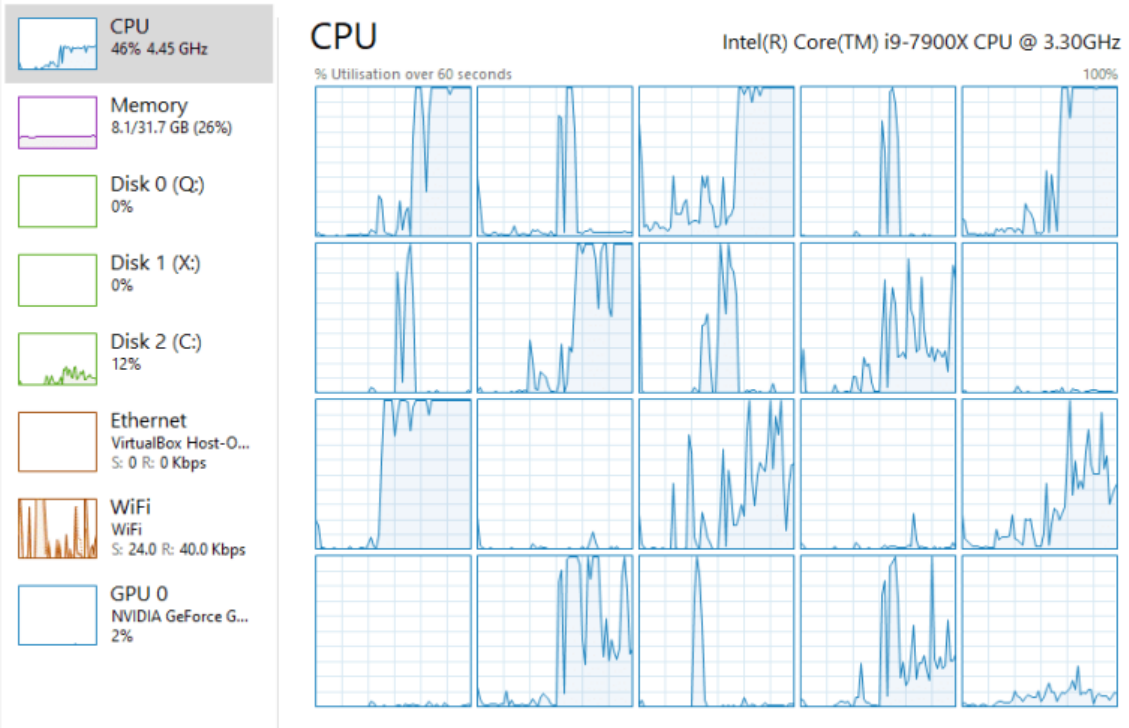
Illustration by Storyset.com

Using the right tools helps to perform tasks efficiently

- The **same computing tasks** can be performed by using **different software tools.**
- **⚠ Not all software components are able to utilize available modern computing capabilities or utilize them efficiently.**
- **⚠ Computing without efficient use of resources is suboptimal and costly.**



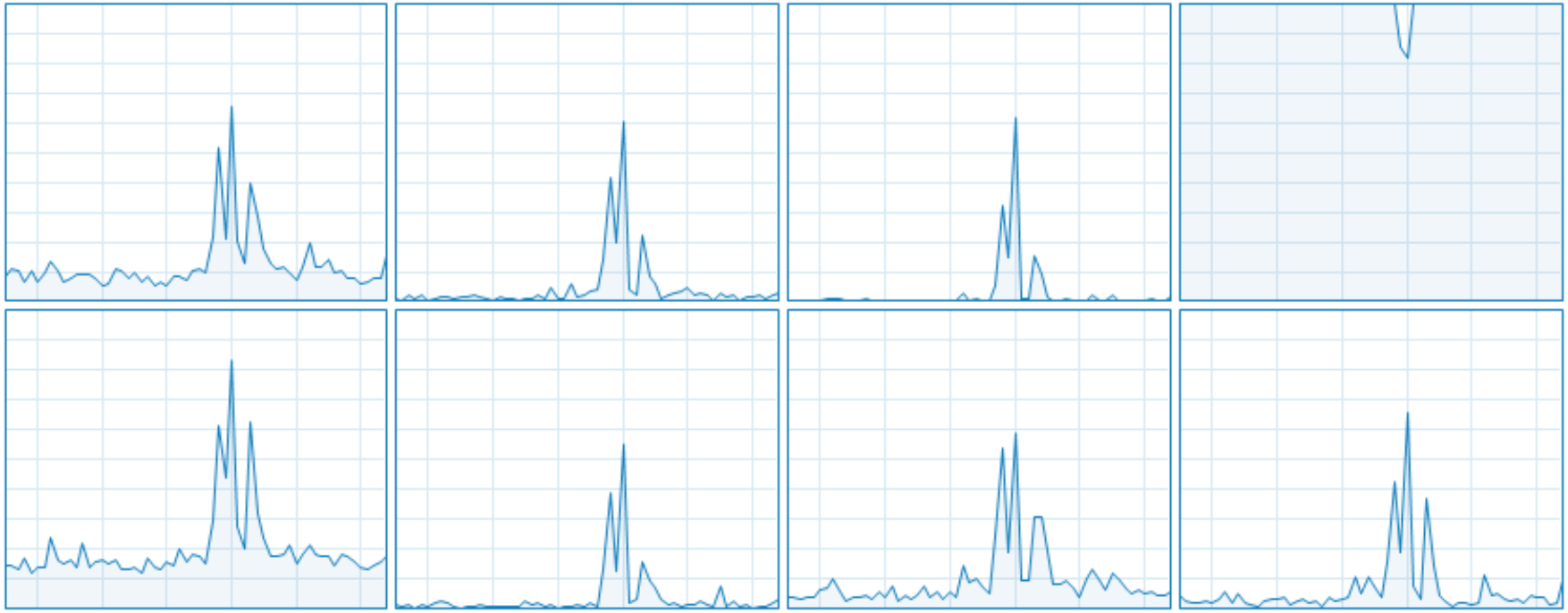
Monitoring of the resource utilization is the crucial first step to understand the situation.



Operation systems provide tools to monitor system resources

OS	GUI Tools	CLI Tools	Metrics Covered
Windows	• Task Manager • Resource Monitor • Performance Monitor	• typeperf (performance counters) • PowerShell: Get-Process, Get-Counter	CPU, memory, disk, network, GPU, custom logs
Linux	• GNOME System Monitor • KDE System Monitor • Other DE-specific system monitors	• top / htop (process, CPU, memory) • vmstat (memory, processes, disk) • iostat (disk) • sar (historical stats) • uptime (load averages) • perf (advanced profiling)	CPU, memory, disk, network, load averages
macOS	• Activity Monitor	• top (process, CPU, memory) • vm_stat (virtual memory) • iostat (disk) • netstat (network) • powermetrics (hardware, energy)	CPU, memory, disk, network, energy usage

"I have an advanced machine learning workflow that processes large number of S2 scenes. The analysis takes very long time. Can you give me more accounts, so that I can run more tasks with multiple accounts?"



Shall we do that?

You can make **informed decisions** about the most **suitable computing strategy** for your workflow by **benchmarking available options**.

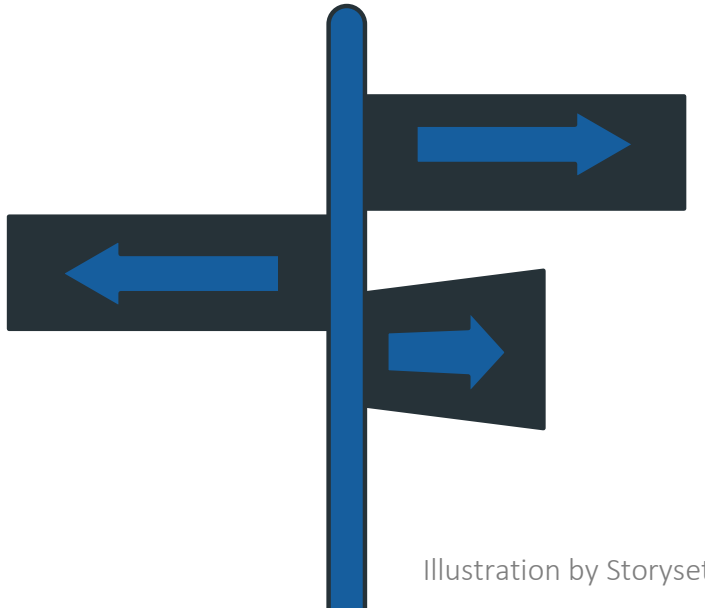


Illustration by Storyset.com

Benchmarking is the **systematic process of measuring and comparing** the performance of a system or method against a standard reference point (a benchmark) or against peers.

Illustration by Storyset.com

Benchmarking geospatial analysis workflows is tricky

To **ensure reliability and reproducibility**, multiple points should be considered with care:

- Experimental design
- Workflow documentation
- Data management
- Performance metrics
- Reporting and transparency

Illustration by Storyset.com

geobench

A package and command-line tool to monitor and benchmark geospatial workflows

https://github.com/ITC-CRIB/geobench
https://geobench.readthedocs.io/en/latest/

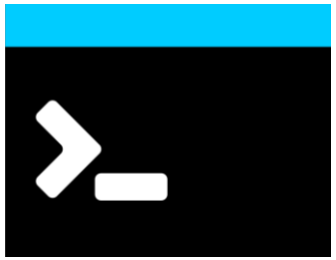
Key features of Geobench facilitate geospatial benchmarking

- A **robust benchmarking framework** ensuring a controlled working environment with idle status monitoring.
- Monitoring of all critical **metrics with minimum overload** on the system with a stage-wise approach.
- Support for **multiple executors** common to geospatial workflows.
- Easy **definition of scenarios** with automated generation of different sets of arguments to test different options.
- **Machine- and human-friendly** benchmarking outputs and reports.
- Automated **storage of input and output (geospatial) data**.

Illustration by Storyset.com

Geobench supports multiple executors to run geospatial workflows

- Shell commands
- Python scripts
- QGIS processes
- QGIS Python scripts
- GDAL commands
- Jupyter Notebooks

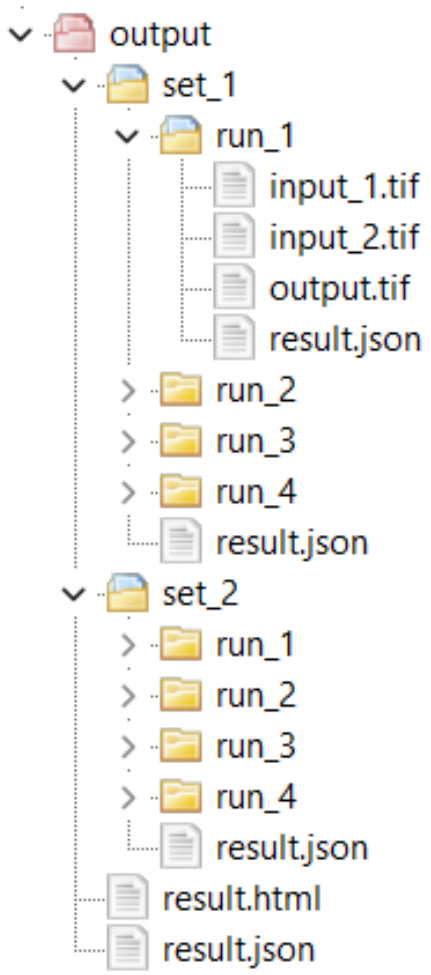


Geobench collects workflow- and system-related metrics for all runs

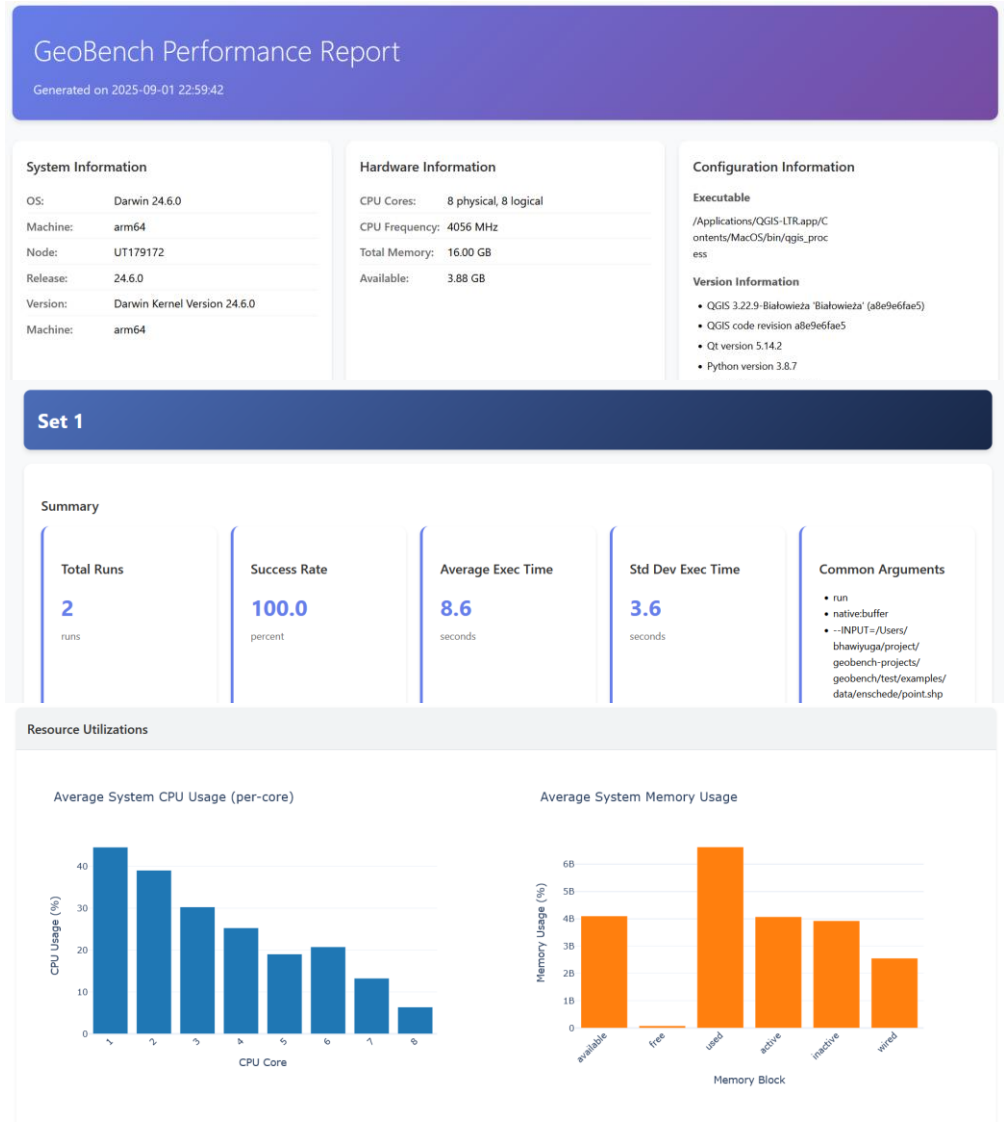
- Baseline and endline monitoring of the system
  - CPU usage, memory usage, disk I/O, network I/O at system level.
  - CPU usage, memory usage, disk I/O of all processes.
- Monitoring of the workflow
  - CPU usage, memory usage, disk I/O of workflow processes.
  - CPU usage, memory usage, disk I/O, network I/O at system level.
  - CPU usage, memory usage, disk I/O of all processes (less frequently).
- Additional metrics will be available soon, such as GPU usage, process-level network I/O and energy consumption.


```
"13512": {
  "pid": 13512,
  "parent_pid": 15092,
  "name": "python.exe",
  "executable": "C:\\Program Fil
  "command": [
    "C:\\Program Files\\Python31
    "-c",
    "from multiprocessing.spawn
    "--multiprocessing-fork"
  ],
  "environment": {},
  "create_time": 1756061986.1379
  "metrics": [
    {
      "step": 2,
      "timestamp": 1756061988.17
      "cpu_percent": 95.6,
      "memory_percent": 0.048161
      "num_threads": 4
    },
    {
      "step": 3,
      "timestamp": 1756061989.20
      "cpu_percent": 103.1,
      "memory_percent": 0.048161
      "num_threads": 4
    },
    {
      "step": 4,
```

Results are provided as detailed and summary JSON and HTML files




```
{
  "set": 1,
  "run": 2,
  "arguments": {},
  "baseline": {
    "duration": 5.0,
    "interval": 1.0,
    "start_time": 1756061980.4074802,
    "end_time": 1756061984.911623,
    "avg_cpu_percent": 0.6,
    "avg_memory_percent": 20.7,
    "process_summary": [
      {
        "pid": 0,
        "name": "System Idle Process",
        "username": "NT AUTHORITY\\SYSTEM",
        "avg_cpu_percent": 1590.94,
        "avg_memory_percent": 2.477468354367659e-05,
        "read_bytes": 0,
        "write_bytes": 0
      },
      {
        "pid": 2840,
        "name": "dwm.exe",
        "username": "Window Manager\\DWM-1",
        "avg_cpu_percent": 2.74,
        "avg_memory_percent": 0.34699669518108867,
        "read_bytes": 5568,
        "write_bytes": 0
      },
      {
        "pid": 5728,
        "name": "python.exe",
        "username": "ThinkPad-Yoga\\Serkan",
        "avg_cpu_percent": 2.06,
        "avg_memory_percent": 0.09839761063042031,
        "read_bytes": 0,
        "write_bytes": 0
      },
```

How to install Geobench?

- Currently Geobench is in alpha version, and it is not available via the Python Package Index.
- (We have also a name conflict with GEO-Bench).

```
Install most up-to-date version from the repository
pip install git+https://github.com/ITC-CRIB/geobench
```

How to use the command line interface?

```
Run a scenario and store results:
geobench <scenario.yaml>

Run a scenario with a custom number of repetitions and store results:
geobench --repeat <n> <scenario.yaml>

Run a scenario with a custom number of repetitions and store results in a custom directory:
geobench --repeat <n> --output <out_dir> <scenario.yaml>
```

Geobench uses YAML files to define scenarios

```yaml
name: Count Primes Python
type: python
command: count_primes.py
arguments:
    cores:
    - 4
    - 8
    num:
    - 1000000
    - 2000000
repeat: 5
system_wait: 5.0
system_monitor: 10.0
run_wait: 5.0
run_monitor: 5.0
```

```yaml
name: Buffer QGIS Process
type: qgis-process
command: native:buffer
repeat: 10
inputs:
    INPUT: data/enschede/point.shp
outputs:
    OUTPUT: output.shp
arguments:
    distance_units: meters
    ellipsoid: EPSG:7030
    SEGMENTS: 19
    END_CAP_STYLE: 0
    JOIN_STYLE: 0
    MITER_LIMIT: 2
    DISSOLVE: false
    SEPARATE_DISJOINT: false
    DISTANCE:
    - 0.001
    - 0.002
```

How to use programmatically, e.g. in Jupyter notebooks?

```python
from geobench import Geobench

# Create a Geobench instance
bench = Geobench(
    name="my-benchmark",
    outdir="results",
    run_monitor=2.0,
    clean=True
)

# Start monitoring
bench.start("my-workflow")

# Run the workflow
err = None
try:
    my_workflow()

except Exception as err:
    raise

finally:
    # Stop monitoring
    bench.stop(True if not err else False)

    # Generate HTML report
    bench.generate_report()
```

```python
from geobench import geobench

# Create a decorated function
@geobench(
    name="my-benchmark",
    outdir="results",
    run_monitor=2.0,
    clean=True
)
def my_function():
    # Your code here
    return result

# Call the decorated function
result = my_function()
```

Let's demonstrate!

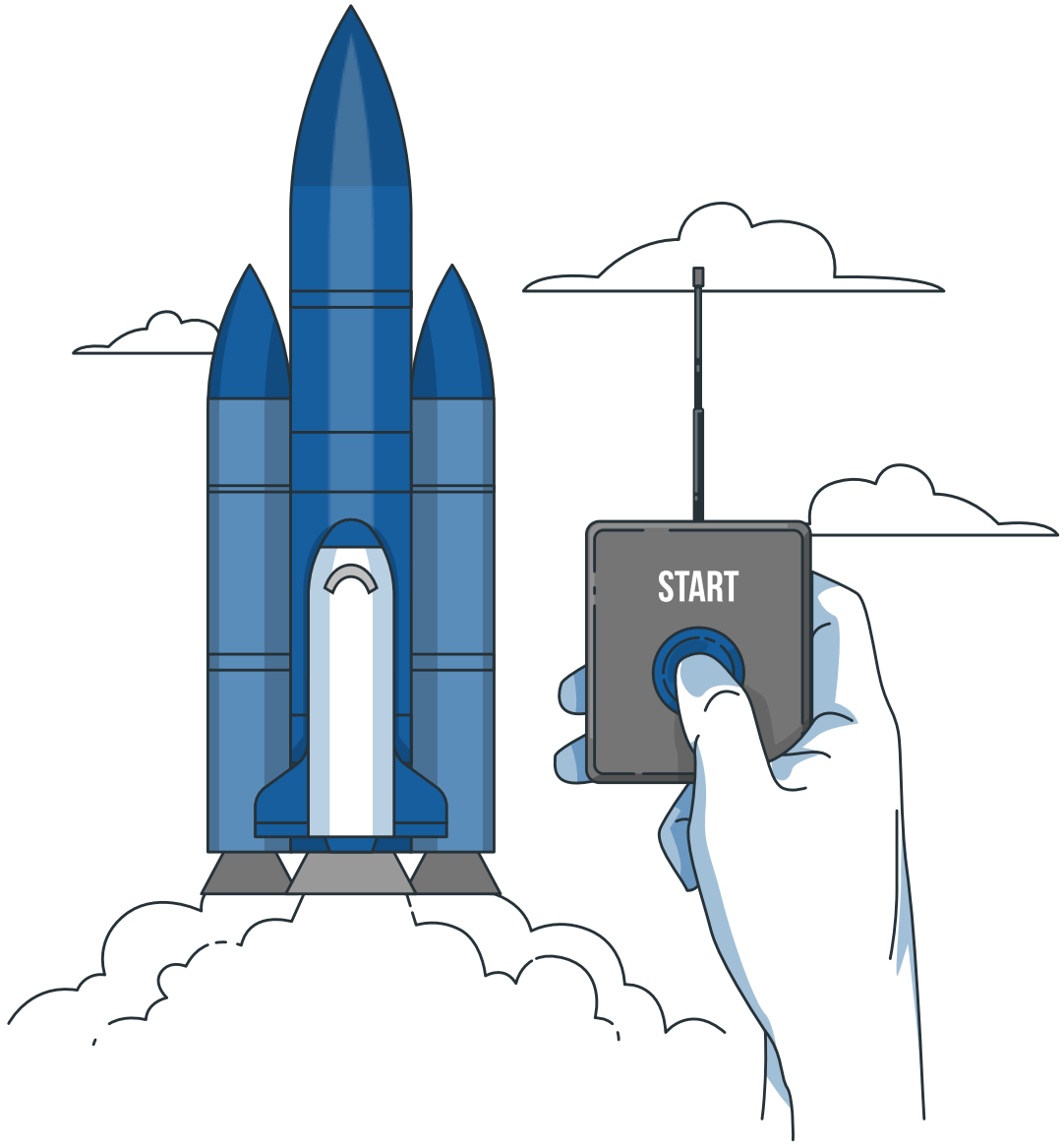


Illustration by Storyset.com

We will be happy to have your further feedback and suggestions!

Dr. Ing. Serkan Girgin MSc

Director
Center of Expertise in Big Geodata Science

Associate Professor
Department of Geoinformatics

Faculty ITC, University of Twente

s.girgin@utwente.nl
https://linkedin.com/in/serkan-girgin/