TUFLOW

aws Australian Water School

Hands-on model running with automation

Jaap van der Velde

BMT

Hands-on Model Running with Automation

Jaap van der Velde
TUFLOW (BMT)

As a software engineer at heart, I approach every problem as a software problem. In a time where AI is finally becoming mainstream, and the capacity to write software, at least at a small scale is becoming universal, I am more motivated than ever to help people make it work.

“Software is the Language of Automation“ – Jensen Huang

Overview

Motivation

- Why another webinar on automation?
- The impact of AI

A brief recap

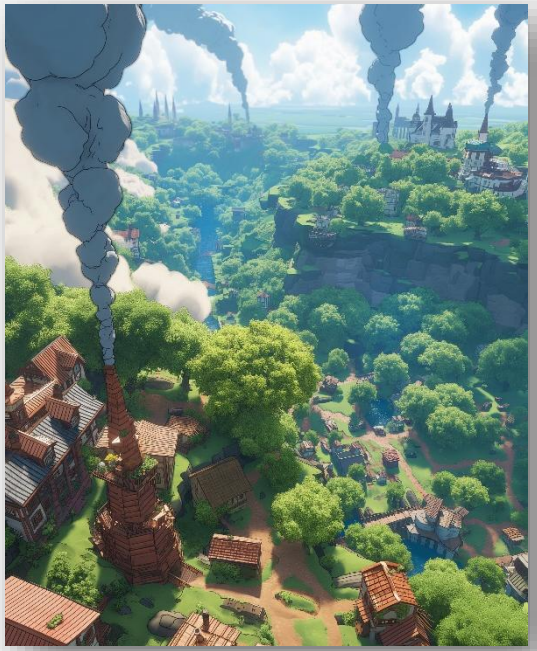
What role AI can play

Diving into Console Applications and the CLI, with examples

Applying Automation to Model Running

Tools & Tips

Your Questions & Answers.



Motivation for Automation

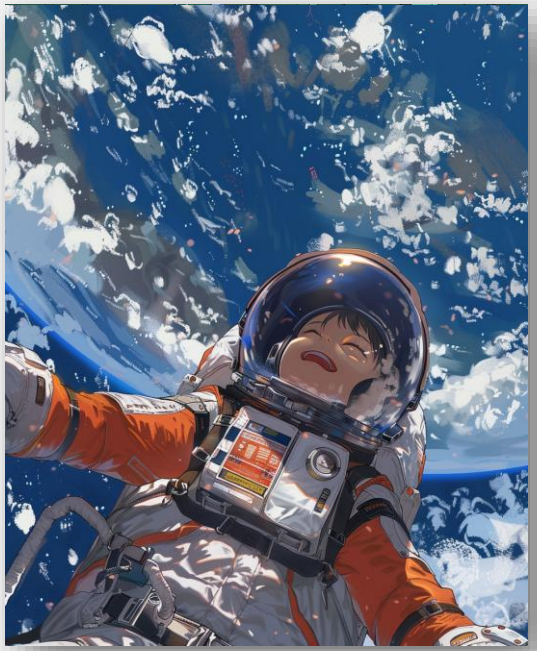
When done well, automation is:

- **Efficient**
  Optimises resource use over time: people, hardware, and licences.
- **Consistent, Repeatable & Reproducible**
  Minimises human error and personal variation.
  It works the same way every time, no matter who runs it, or where.
- **Scalable**
  Can handle one task or a thousand, with little extra effort.
- **Documented & Versioned**
  You know what it does, how it does it, and when it changed.

Efficient Model Runing with Automation - Summary

Key points:

- **Engage with IT, understand your problem**
  Know enough to talk to them about your problem and solution
- **Balance what you can handle, are allowed to use, and expect it to do.**
- **Get to know your environment and the possibilities**
  Test your assumptions and solutions.
- **There is too much to show in examples, or is there?**
  The aim is to give a sense of what is possible, not complete recipes.
- **Configure and properly develop your model and automation**
  Automation is part of you model and workflow, not just an add-on.



AI & Automation

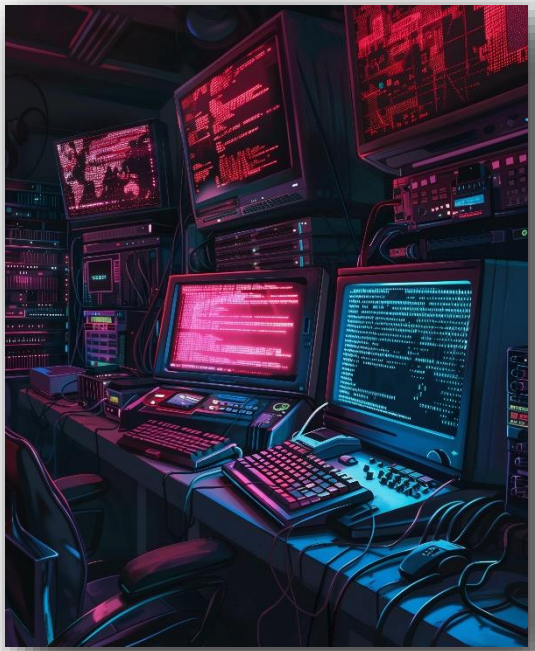
Use AI (LLM) as a collaborator, not as a worker or a crutch.

- **Design with AI**
  Use it to design and refine your automated process, not replace it.
- **Generate code, not output**
  Create artefacts you can review, vary, save, extend, and hand over.
- **Play to its strengths**
  Code generation and analysis suit LLM capabilities best.
- **Stay accountable**
  Responsibility lies with you – validate everything. Ask the AI to explain anything you don't fully understand.

AI results can be wrong, incomplete, or misleading. Output produced directly by AI lacks documentation of how it was created and cannot be reliably reproduced. Models can change outside your control.

The Command Line Interface

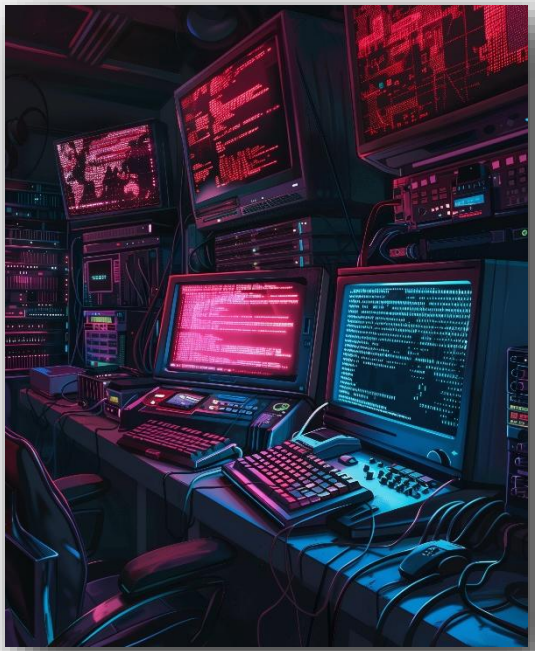
- **The right tool for the job**
  Command Prompt? Windows PowerShell? PowerShell Core? Bash? Python? R?
- **Whatever you pick, use it yourself**
  Your scripts are like a transcript of your work, at least at first.
- **Don't learn by rote**
  Tools are often self-documenting, understand what you run.
  https://github.com/Grismar/gpt
- **Most commands are literally self-explanatory**
  Not all of them are though, notably TUFLOW isn't.



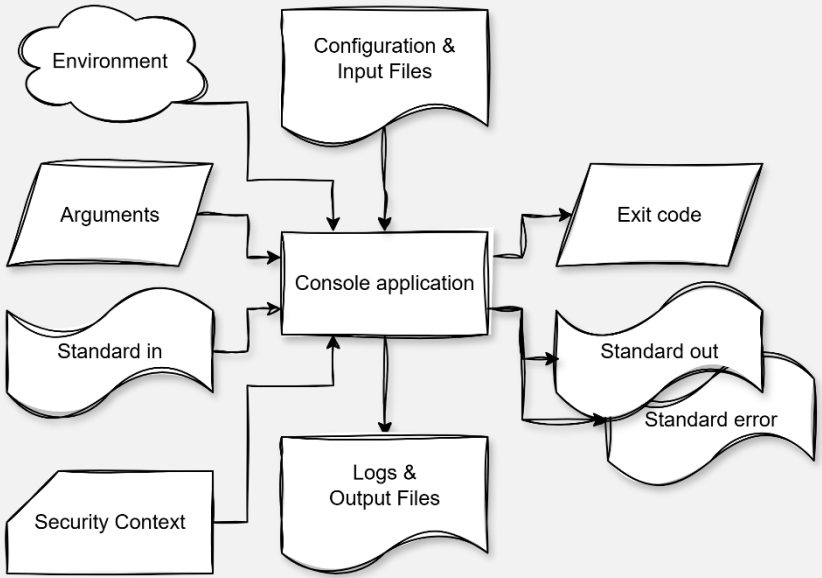
The Command Line Interface

```
PowerShell
PS C:\Users\jaap.vandervelde> gpt --replay 3

 3> "how do I get help on other commands?"
Use the Get-Help cmdlet followed by the command name to get help on PowerShell commands:

 Get-Help <CommandName>

PS C:\Users\jaap.vandervelde> get-help get-childitem -online
```

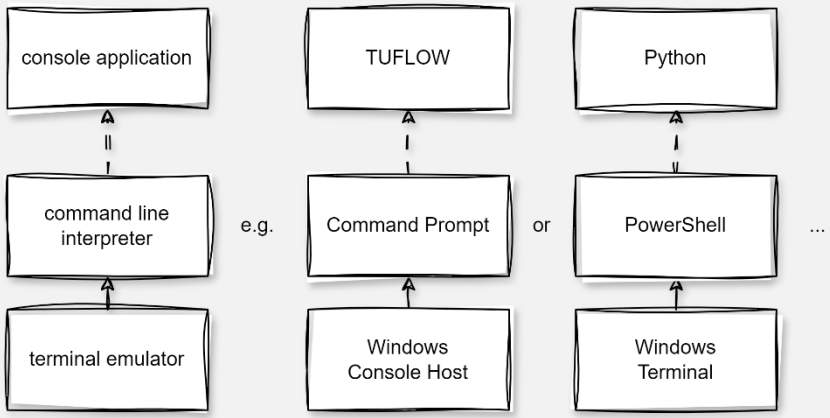
https://github.com/Grismar/gpt *(personal project, free and open, no official support)*



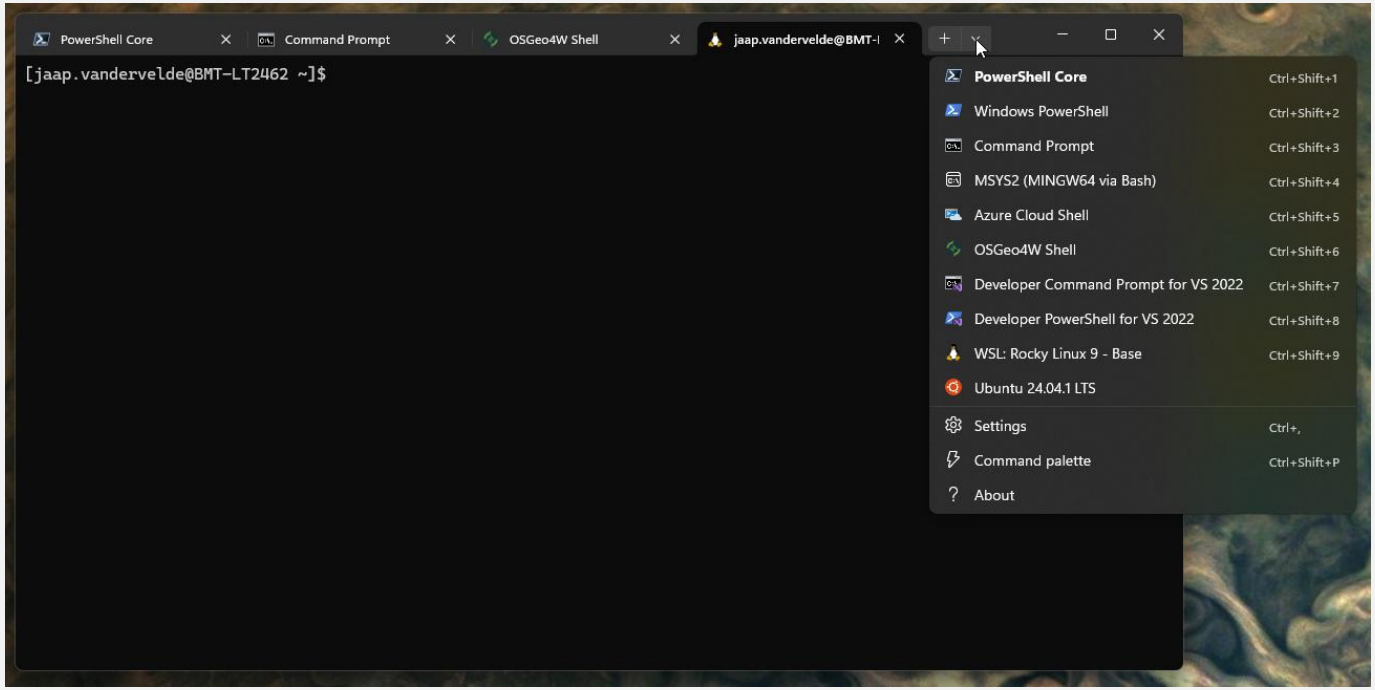
Basics of command line automation



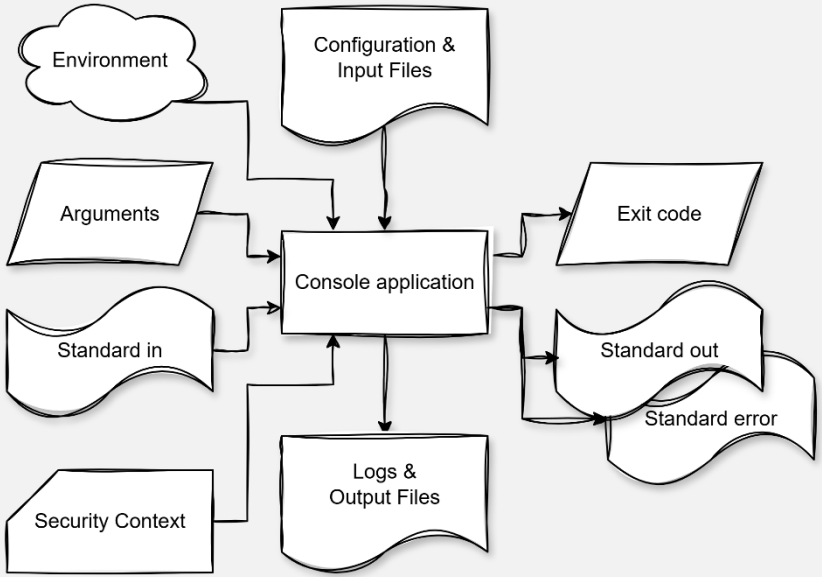
Basics of command line automation



Basics of command line automation

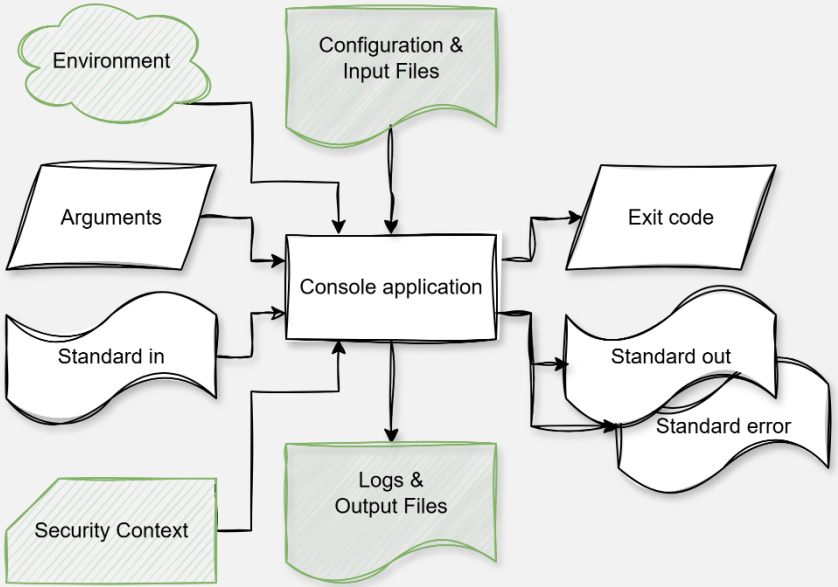


Basics of command line automation

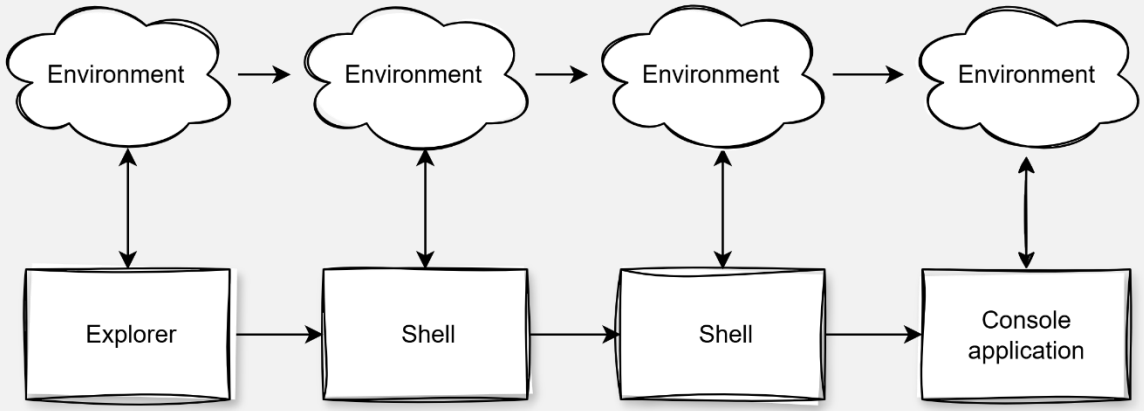


CLI – Environment and Security Context

- **The environment is a set of inherited variables**
  The 'working directory' is often included, as it typically comes along with the environment.
- **Files & Folders are relative to the working directory**
  Both are easily accessible and changeable from within the application.
- **The security context has identity and permissions**
  By default, application run with your user identity, group memberships, privileges, etc.

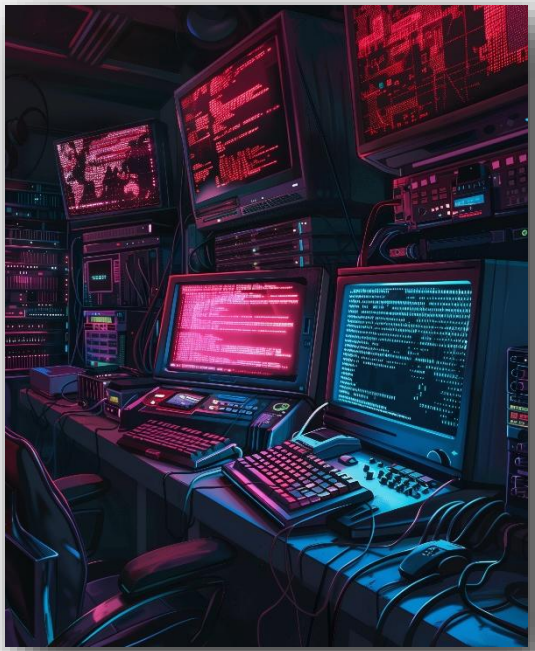


CLI – Environment and Security Context

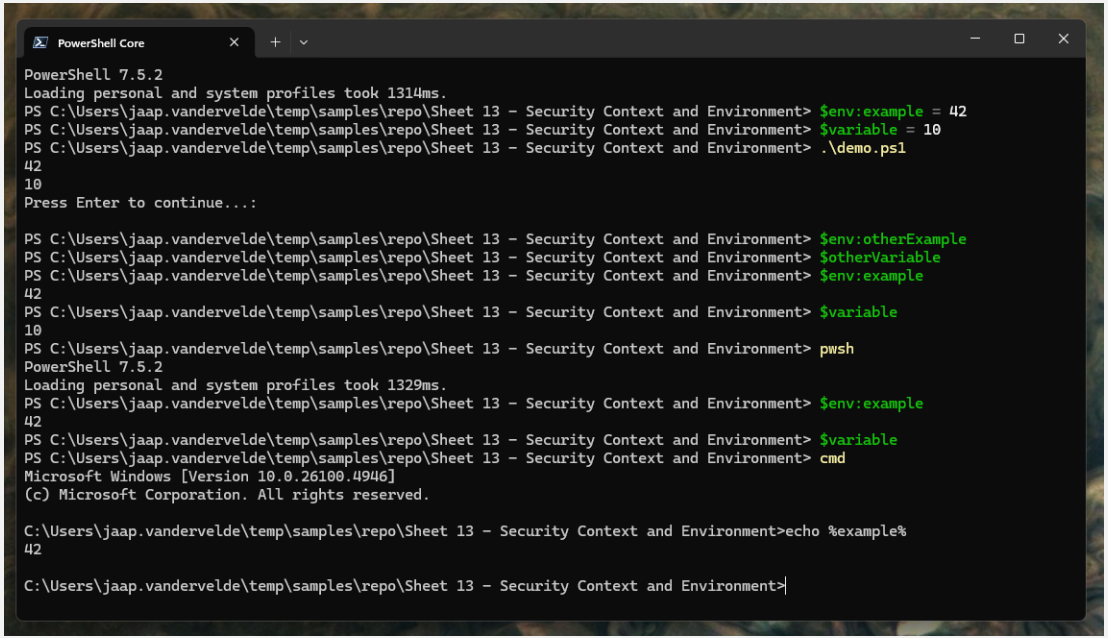


CLI – Environment and Security context

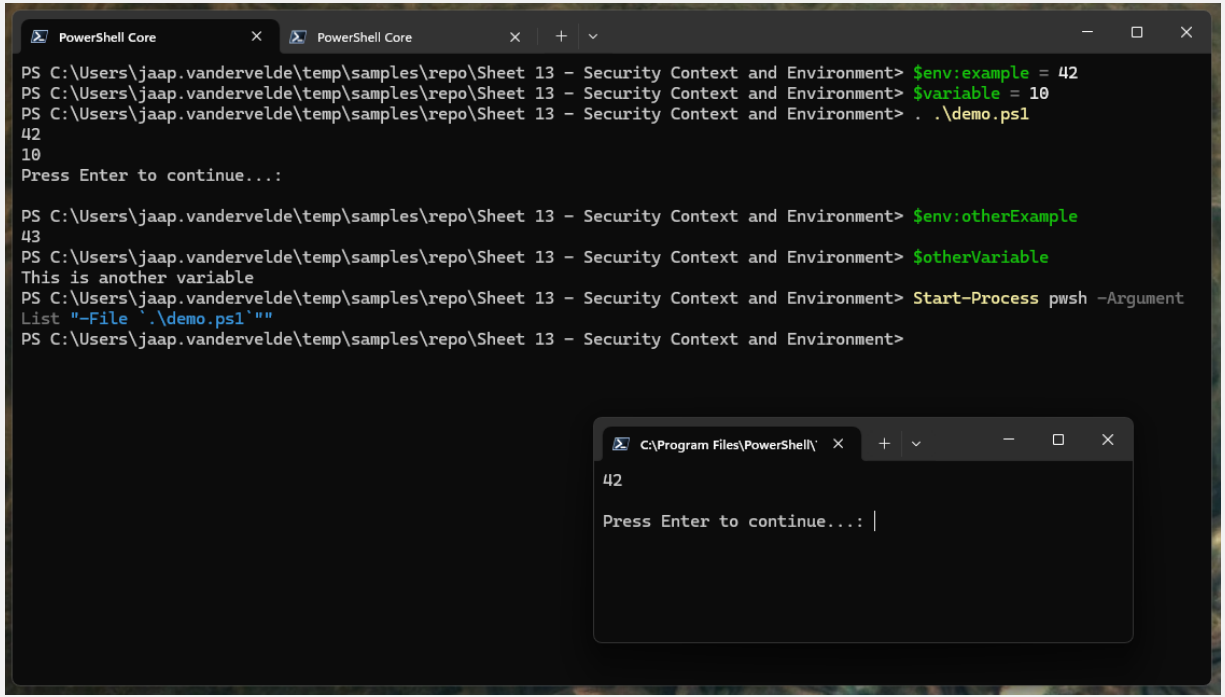
- If variables seem to disappear, think about ‘scope’
- It depends on the type of shell you use
- It depends on whether you run a:
  - command, directly or using source (.)
  - script e.g.
    ```
    .\script.ps1
    ```
  - process e.g.
    ```
    Start-Process pwsh -ArgumentList '-File', 'script.ps1'
    ```
- It depends on the kind of variable, environment or local



CLI – Environment and Security Context

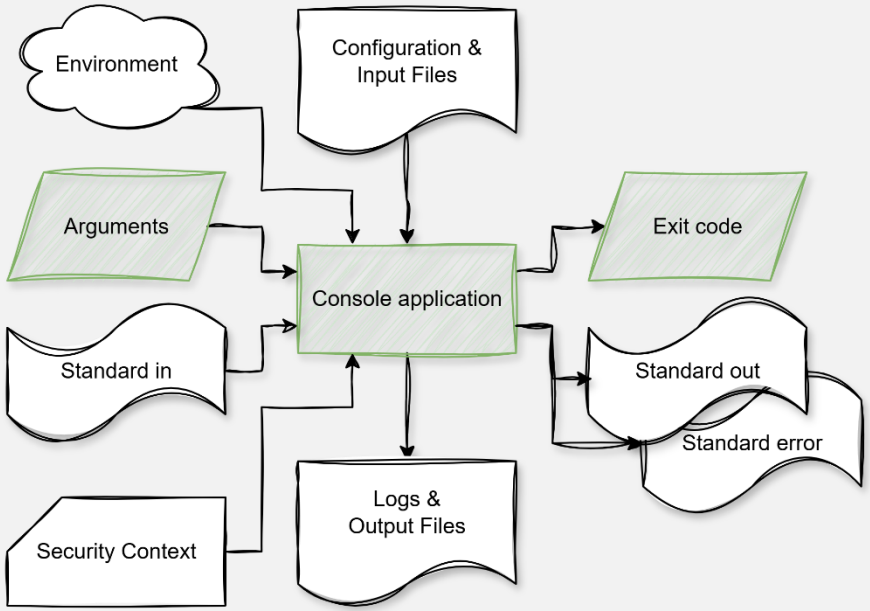


CLI – Environment and Security Context



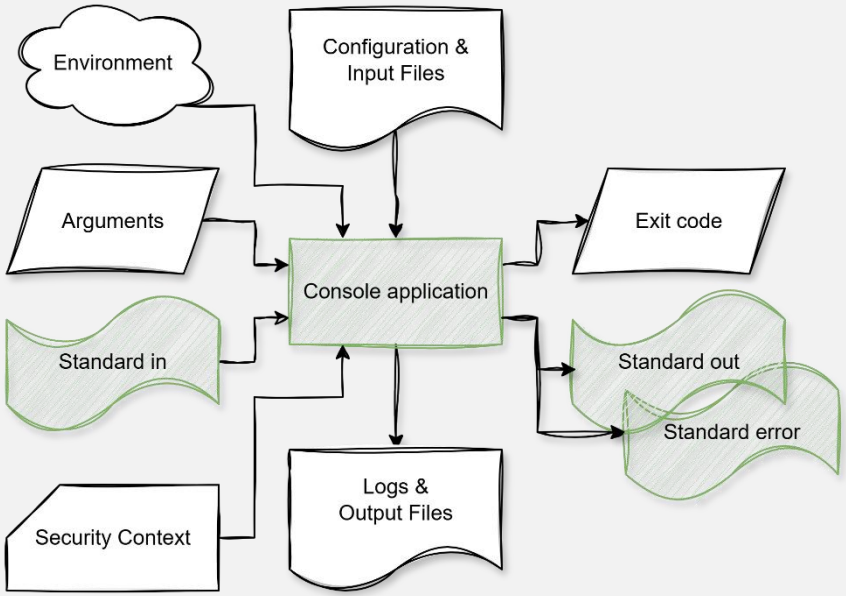
CLI – Arguments and Exit codes

- **Online help and help online**
  Get-Help, /?, -h, --help
  TUFLOW.exe Options (Switches)
- **What are you really running?**
  `Get-Command, where, which, whereis`
- **On exit, zero is good**
  In PowerShell, commands set `$?`, to `$True` or `$False`, executables have `$lastexitcode`



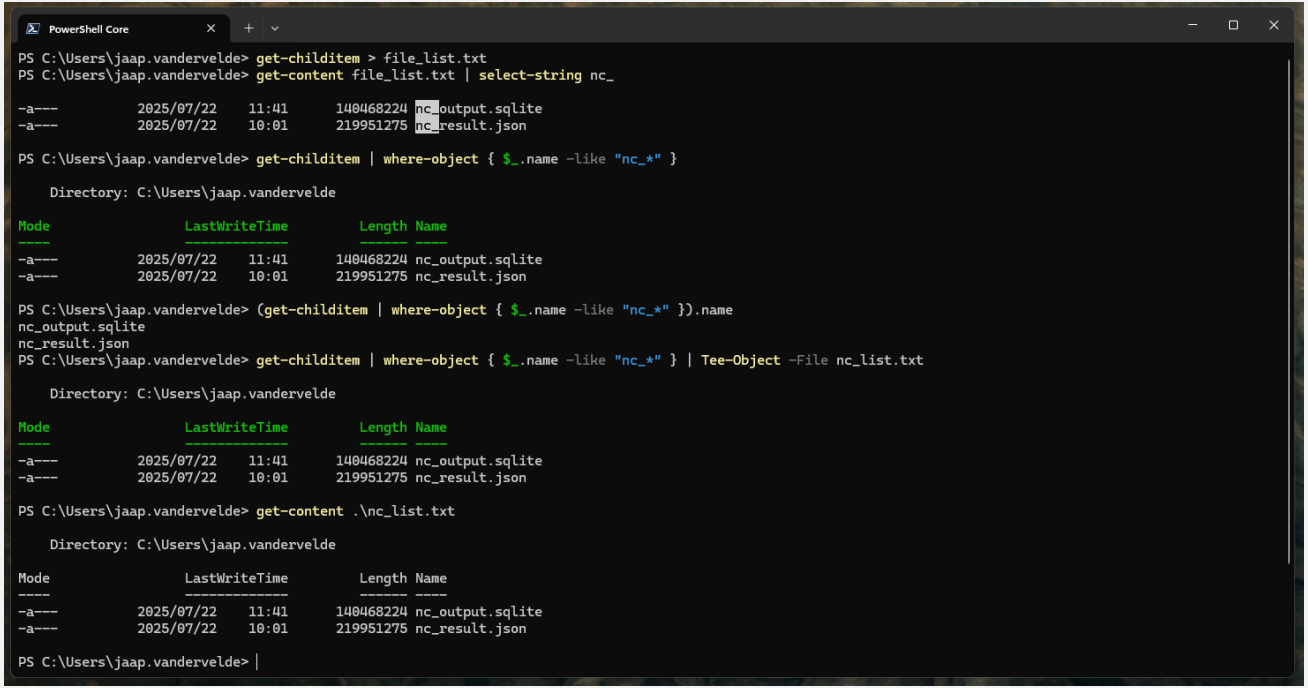
CLI – Streams, buffers, pipes, and redirection but not as you know them

- Connect applications with pipes: `A | B`
- Redirect outgoing streams: `A > file.txt`
- For redirecting the error stream: `2>`
- Streams are live, but some applications may buffer if they can tell they are streaming into a pipe (TUFLOW does)
- You can split a stream with `Tee-Object`
- PowerShell streams objects,
  Command Prompt streams text.
- Think of a text file as a frozen stream.



CLI – Streams, buffers, pipes, and redirection

but not as you know them



Aside: Regular Expressions, a superpower

A brief more advanced aside:

- Worthy of a webinar all by itself: regular expressions.
  https://regex101.com
- A single example:

```
$regex = 'HPC:.*(\d{2}:\d{2}:\d{2})\s+[\d\.]+\s+[\d\.]+\s+[\d\.]+\s+([\d\.]+)'
Get-Content .\buffered.log |
    Where-Object { $_ -match $regex} |
    ForEach-Object { $matches[1], $matches[2] }
```

- Nearly everything supports regular expressions

  PowerShell, Command Prompt, Bash, but also Notepad++, VS Code, Python, some databases, etc.

Applied Automation

Before you get started…

- **Model Design**

Applied Automation – Model Design

- **Set up a new model with automation in mind**
  You can add automation to an existing workflow, but if you're serious about automation, consider optimising for it
- **Version your model**
  Always a good idea, but more so when automation is part of your setup.
  Naming Convention - Tuflow
- **AI *can* be a great help**
  It is at its best when you help it help you.
- **Consider automating your model configuration**
  But use what is there before doing it yourself.
  4.1 Control Files | TUFLOW Classic/HPC User Manual Release Candidate 2025.2

Applied Automation – Model Running

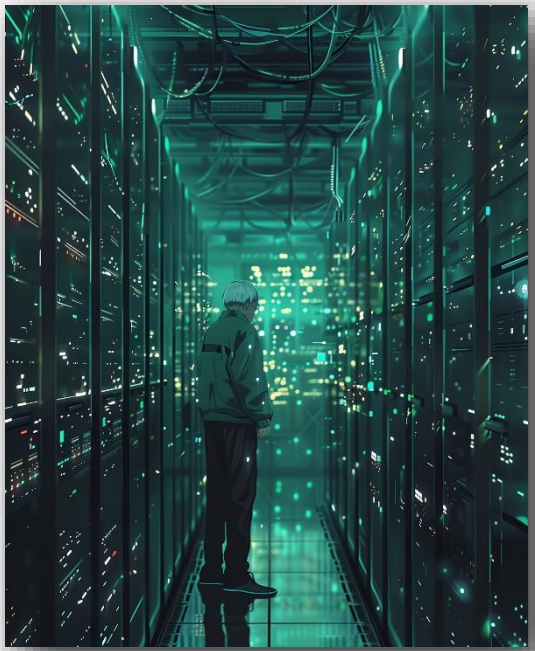
- Model Design
- Model Running

Applied Automation – Model Running

- **Most common, scripted model runs**
  Mock models, a decent editor, and collecting output – on any platform.
- **Batch Files**
  Right for some use cases, but not for serious automation.
- **PowerShell or Python?**
  From scripting to development.
- **The loop, a classic**
  From beginner to advanced to …

Running TUFLOW – wiki

TUFLOW Runner - Tuflow



Applied Automation

- Model Design
- Model Running
- Output Processing

TUFLOW User Group - Gitlab

Applied Automation

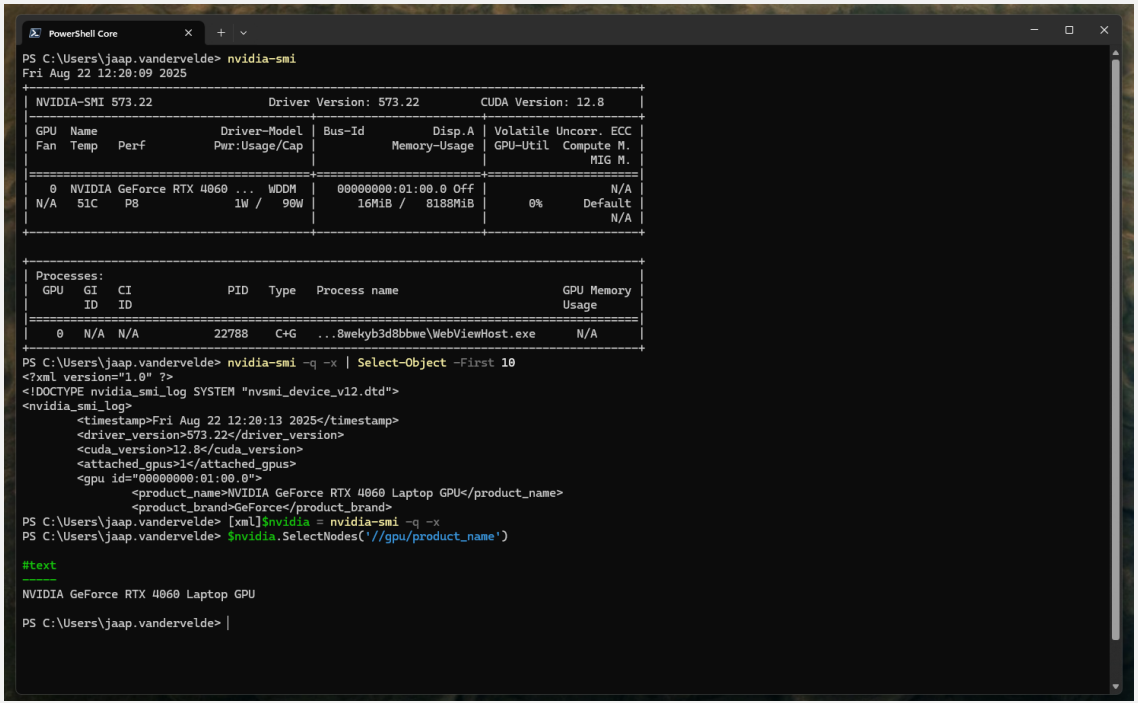
- Model Design
- Model Running
- Output Processing
- Resource Management

Applied Automation – Resource Management

- **Some examples:**
  - **Available licences**
  - **Available hardware**
  - **Available computers**

- **This is where IT can really shine**
  Talk to them about Zabbix, Grafana, Nagios, etc.

Applied Automation – Resource Management



Applied Automation

- Model Design
- Model Running
- Output Processing
- Resource Management
- Data Management

Applied Automation – Data Management

- **Consider getting your data where and when you need it**
  The convenience of copying whole models is one for humans.

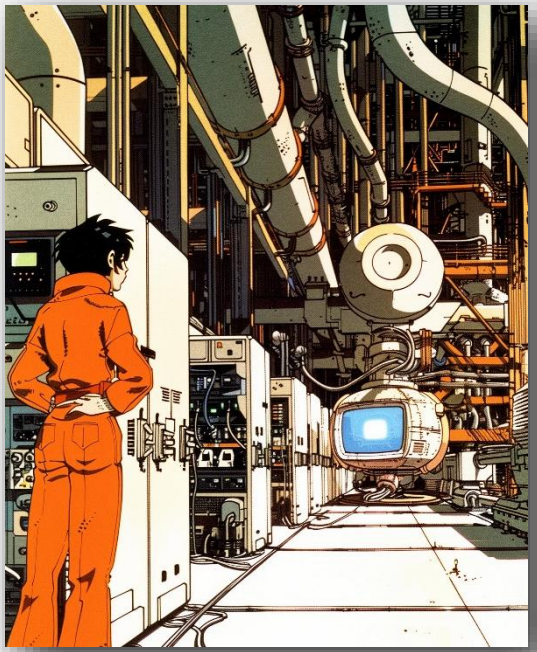
  TUFLOW FV Get Tools

- **Automation makes it easier to regenerate**
  Consider more carefully what you really need to move or keep

- **Compression**
  You can compress what you keep, even without losing access

  py7za · PyPI

- **Moving data around**
  - **Avoid A > B > A**
  - **Your convenience vs. the efficiency of tools**



Tools & Tips

- **Windows Terminal**
- **VS Code** (TUFLOW extension coming soon)
- **Git** (possibly GitLab or similar)
- **WSL** (TUFLOW is coming to Linux)
- **Py7za**
- **TUFLOW Runner**
- **Sandbox**
  Maintain a sandbox, work on sample data, be agile.

Questions?