Tools for software development
Module introduction

Dr Peadar Grant

January 24, 2024

Contents

1 Aims

- This module introduces students to modern development tools, environments and processes.
- The industry is evolving rapidly and so are the tools.
- Understanding how to use these tools requires more knowledge and skills than just development, mathematics or theory.

1.1 Changing environment

Some of the drivers for these changes are:

- Complex systems with large data requirements are increasingly run on multiple computers so they can complete tasks quickly.
- Companies are increasingly responsible for running software platforms for their clients and so have become far more focused on operating quality.
- In order to have many teams work on larger systems, many companies are moving to platforms that distribute system functionality across different computers.

2 Challenges of Data Science Computing

2.1 Complexity

- Data Science Applications can be complex and required to process very large quantities of data.
- Algorithms can be mathematical and processor intensive.
- Some datasets we wish to analyze can be very large and require a lot of resources (disk, memory, processing).
- In modern environments we address these challenges by creating environments where we can run code on multiple computers at the same time (in parallel).

2.2 Consistent environment

- Complex programs uses many libraries and interacts with 3rd party software (like databases)
- A challenge of all areas of computing is ensuring that environments are consistent.
- It is difficult to ensure the environment where software is developed and tested is the same as the environment it runs in production:
  - Libraries can be different versions
  - 3rd party software can be different versions
  - Operating system can be different versions
  - Different hardware environments
- **“It works on my laptop!” is no excuse!**

2.3 Working in teams

How do does a team:

1. Edit the same files while tracking changes
2. Show ownership / responsibility for code
3. Collaborate in on-site, fully remote and hybrid work environments
4. Automate routine tasks (like flagging code updates in MS Teams / Slack)
5. Centrally build, test and deploy code to production environment

3 Key themes

3.1 Source control

- Need to track revisions to files (primarily text-based).
- Need ability to snapshot changes, or roll back.
- Maintain a log of changes.
- Show changes with each version.
- Distributed source control:
  - Developer(s) want to work on different systems.
  - Permit collaboration by sharing changes amongst developers in a team.
- **We will use git in conjunction with GitLab for source control.**

3.2 Automation

- Your own **time is valuable**!
- Don’t waste it on tasks, processes that take longer than they should.
- Routine tasks should be automated to save time, improve consistency.
- Familiarity with command-line environments required:
  - Windows — PowerShell
  - Mac / Linux — Bash, zsh
  - Cross platform — Python (re-use your programming knowledge!)

  They may appear scary but are likely to be your most useful tool!
- **We will use some PowerShell and Bash to automate tasks.**

3.3 Continuous integration

1. Over the years the speed of releasing software has become more important (referred to as Velocity).
2. Businesses need software faster so they can be more competitive and not have business strategy held back by IT.
3. Businesses have also become far more dependent on software to run day to day business (Digital Transformation). Any outages can have serious impact to business operation.
4. **Continuous integration:**
    - automatically build, test and deploy code
    - automatically run analysis, generating result artifacts

3.4 Containers

- By building all dependencies into a container, they can be run anywhere and will perform very consistently.
- Multiple containers can be put onto their own private network so they can talk to each other.
- This allows for micro service based systems to be built.
- The most common container technology is called Docker.
- **Docker** predominantly uses Linux Operating System.

4 Assessment

Table 1: Assessment breakdown

Component	Marks
Weekly lab exercises	40
End-of-module project	40
Class test	20
Total CA	100

4.1 Weekly lab work

- You will be required to do some tasks each week during class in the lab.
- You will use a source-control system (`git`) to keep these in a repository (on GitLab) which the lecturer will have access to.
- You will be graded for completing the lab tasks throughout the semester.
- You can (and should!) help each other with lab work.
- **Your submitted work must be entirely your own**
- If you miss a week, you should catch up as best you can.
- Follow repository, folder naming and formatting requirements.

  If you don't you won't get any marks... Will be strict on this!