Use of 21st century software paradigms and tools for **EXFOR**

NRDC Workshop 2018, IAEA, Vienna, Austria

22-25 October 2018

Michael Fleming
OECD Nuclear Energy Agency

michael.fleming@oecd-nea.org

Agenda

Prelude

Three ideas:

1. Technology in collaboration pays large dividends
2. Automation can eliminate the majority of our workload and free up time for us to do more impactful work
3. We can leverage technologies that represent 100s+ person-**years** of development to aid our work

Multiple groups have similar requirements:

- Provide collaborative space to add/update/correct files
- Selectively control access to content
- Version control content
- QA content
- Generate outputs/results
  - Documents
  - Simulation outputs
  - QA process results
- Record discussions and tie these to decisions reflected in content
- Make official releases of content
- Selectively distribute releases

Traditional methods
Methods

- Send data by emails
- Individuals do work based on their own schedules, asynchronously delivering and reporting within meetings
- Store working data on a webserver, in a folder system or compressed file
- Repeat until deliverable reached

Traditional methods (II)
Solutions

Advantages:

- Easy to implement
- Users don't need to use any new systems

Disadvantages:

- Emails are not ideal for data transfer
  - Spam or other culprits
  - Size limits
  - 'Point-in-time' mailing lists
  - Outlook is not a database management system
- Version control done by many copies on some drive, sometimes inaccessible to participants
- Technical points are typically lost, unless they are put within (official) records/reports
- Processing/using data takes time
- (Valuable) Meeting time spent on mundane tasks

Repository systems
Methods

- GitLab offers:
  - Unlimited private repositories with hierarchical spaces
  - Issue tracking and other project management tools built-in
  - Built-in 'continuous integration/deployment'
  - Natural integration with many tools, e.g. Docker, Kubernetes, others
  - **Many** other features
- GitLab facts:
  - MIT licensed open-source service
  - Choice of RDBs technologies as base
  - Easy to manage, although paid support available
  - Very large user base, including IBM, SpaceX, US DOE, Alibaba, many more
- GitLab misconceptions:
  - It is **not** just for source code (text is more powerful)
  - Any binary large objects (BLOBs) can be managed and BOBs are easier - large file storage could also be utilised

Continuous Integration
Methods

- Every commit to a repository needs to be reviewed - otherwise errors can be introduced
- We could just test before release, but then we have to *find those bugs after the fact*
  - In practice, this is what people do, since testing costs time
- Instead, we can automate the process and we are alerted within seconds/minutes
- This is performed by a ‘Runner’, which performs any task and reports the findings
- We refer to this automatic process as ‘Continuous Integration’
- Outputs, or ‘artifacts’ may be retained as required, including logs, results, programs, documents, etc.

Example: EXFOR - background

Case studies

EXFOR compilations provided by the OECD-NEA Data Bank have utilised legacy methods up to 2018

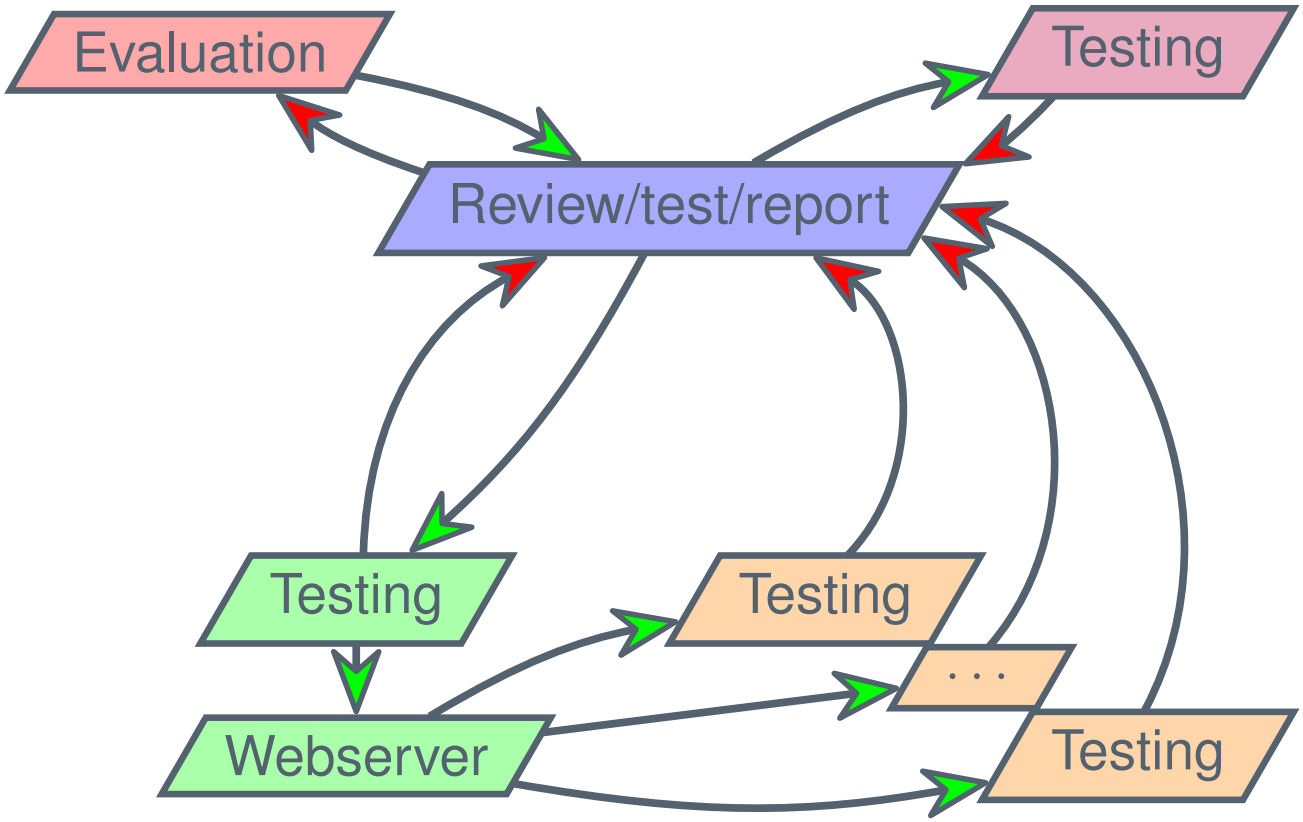
- Many data stored on folders on some network drive(s)
- Little/no description/documentation on the versions (if versions are stored)
- Impractical to trawl emails of previous staff to find information
- ED to OC transition resulted in differences in approach
- Transition from OC to MF with months of interim caused additional burdens

This is a **real, unembellished** case

Example: EXFOR - previous practice

Case studies

The NEA delivers through contractors, with IT support, submitting via the IAEA, reviewed by ~30 officials in the NRDC



Example: EXFOR - real consequences

Case studies

Humans are routinely performing tasks that can be automated:

- Download file and run application on it
- Read output
- Write a summary
- Email to colleagues
- Administrators trawl through Outlook trying to ensure all feedback is responded to

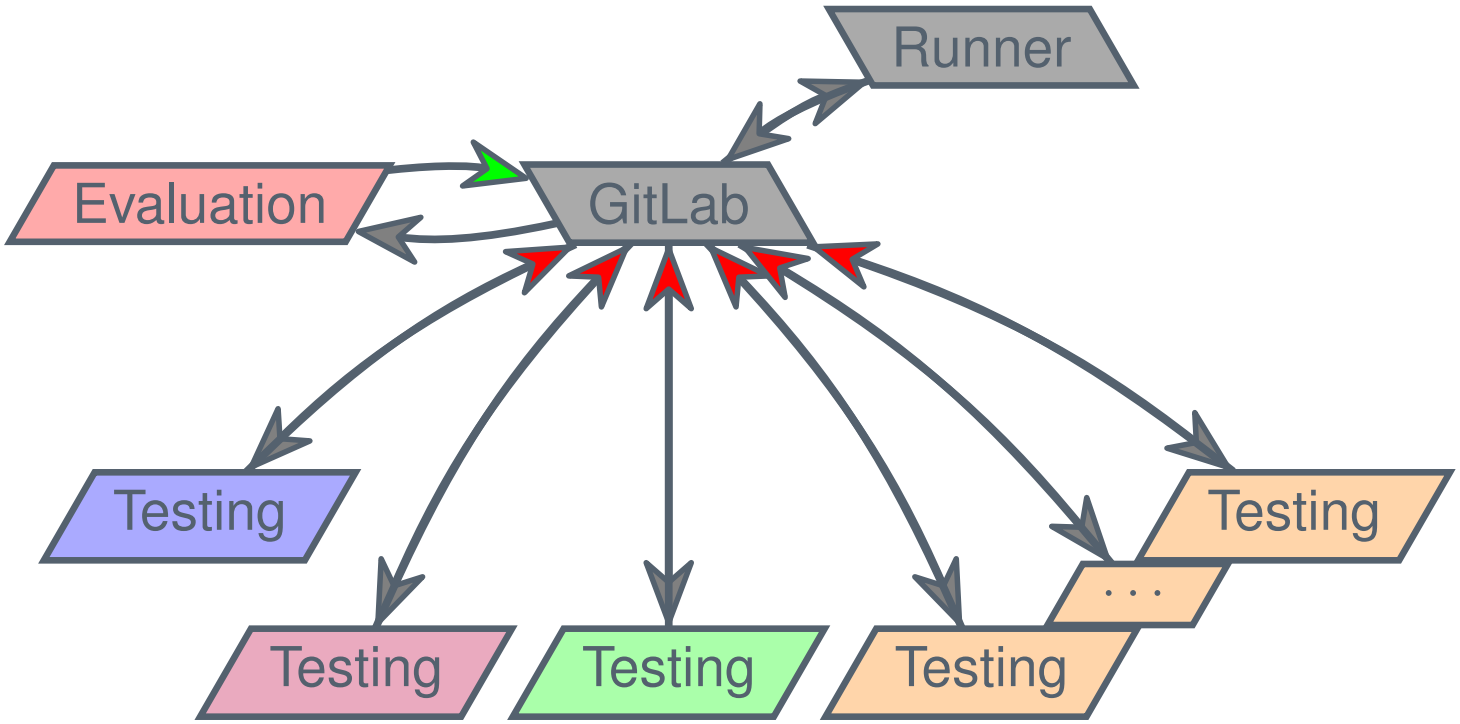
As a result:

- Mistakes are all too common
- Many iterations performed, often with repeated feedback
- Verification is time-consuming and error-prone
- **Significant time and resources spent**

Example: EXFOR - solution

Case studies

Use a **repository system** to collect, version control and test the data, providing immediate, organised and indefinite access to all relevant material to all relevant participants



Example: EXFOR - implementation
Case studies

This has been implemented for the NEA contributions to EXFOR: `https://git.oecd-nea.org/databank/nds/exfor/`

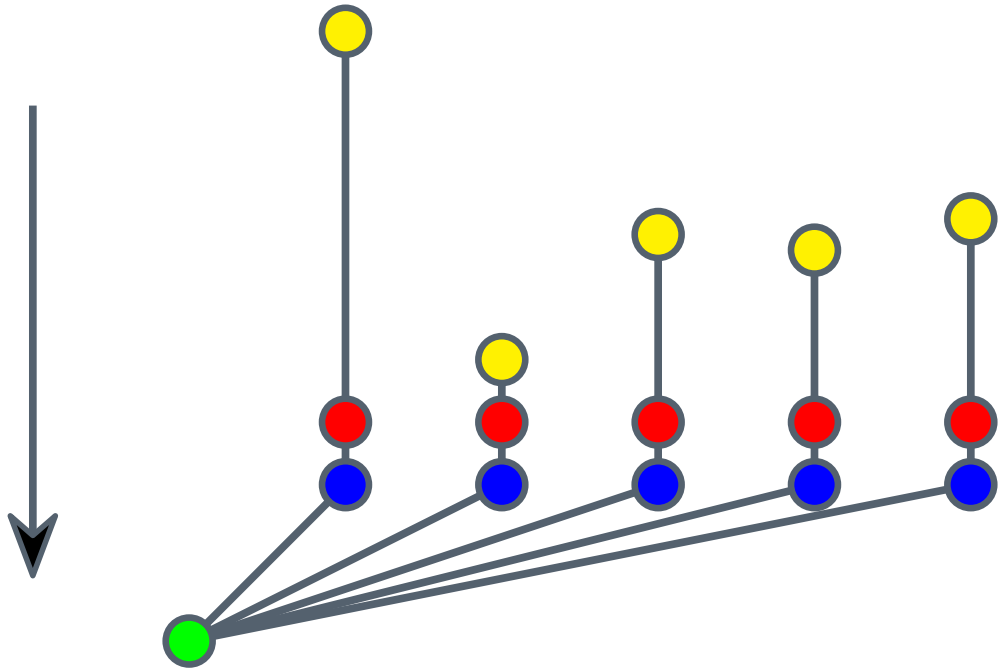
Lessons learnt:

- Very easy to setup structures, accounts
- Some are happy to adopt, others are reluctant
- Simple training can help (i.e. at your next meeting)
- Once they really engage, people appreciate it
- **Processes may be moved onto the automated runner** over time to remove the requirements for humans to perform various tasks

Example: Collaborative Output

Case studies

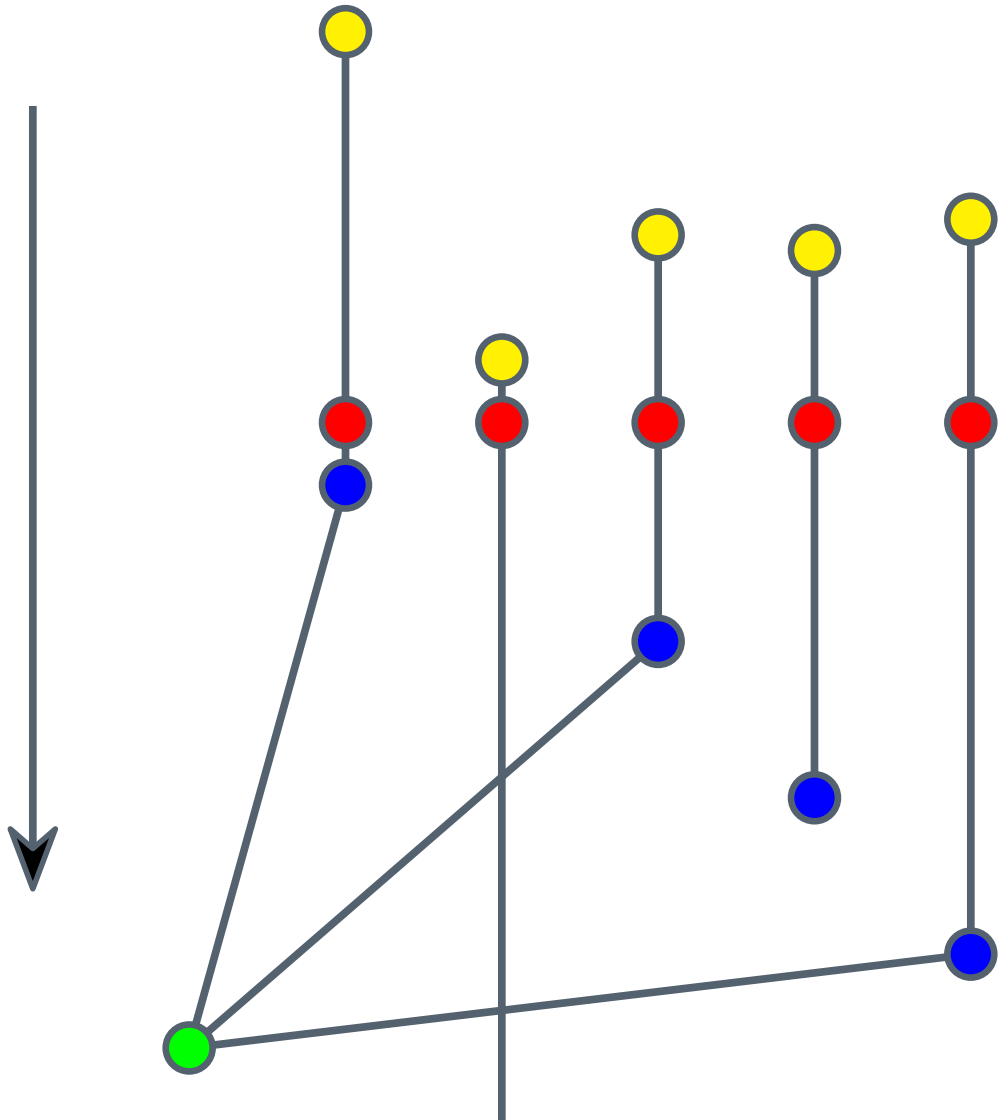
- Typical activities involve work, meetings, contributions, and an output that synthesises the activity
- In theory, it works like this:



Example: Collaborative Output

Case studies

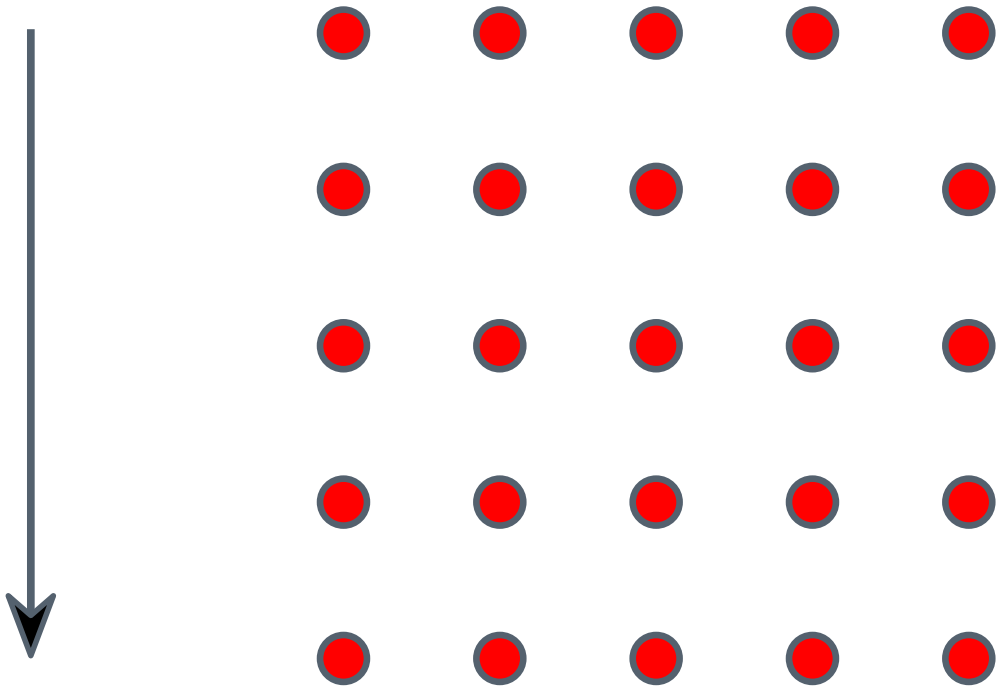
- Sometimes we get this:



Example: Collaborative Output

Case studies

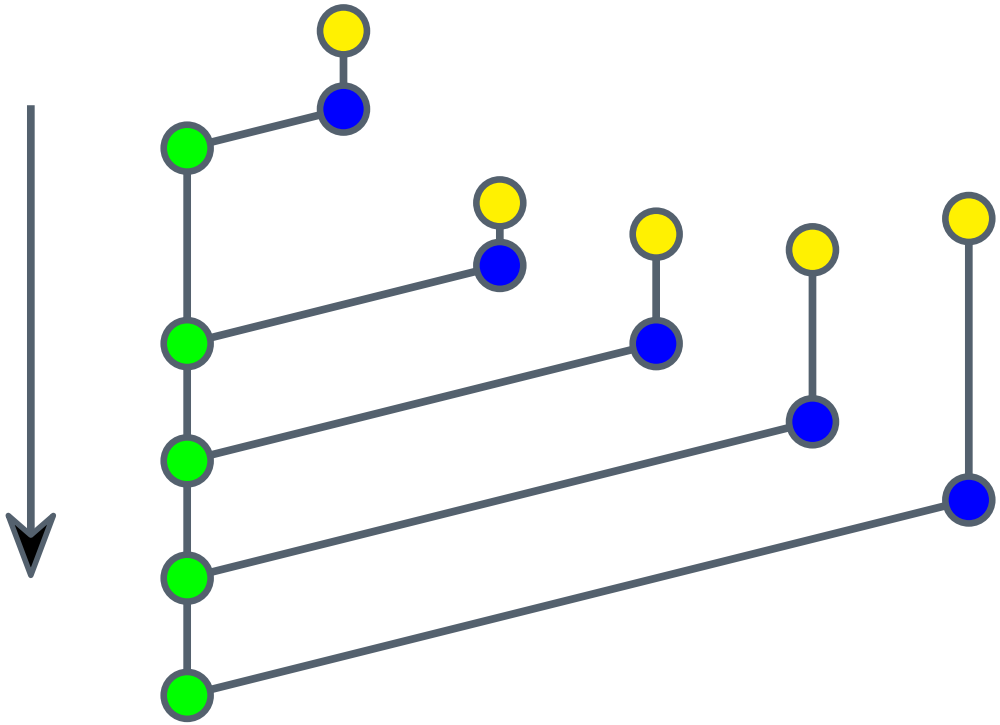
- By focusing on integrating the work and contributions into outputs, and empowering participants to engage directly with them, we avoid:



Example: Collaborative Output

Case studies

- By providing a collaborative space, we can focus instead on the contributions to the output:



Examples: Publications
Collaborative output

- This isn't a theoretical example, there are collaborations that have already used these methods to deliver outputs, with:
  - increased efficiency;
  - transparency between participants;
  - automatic build/testing to limit admin;
  - ability go back to any version or contribution to re-draft; and many more advantages
- Example (private password-protected):
  `https://git.oecd-nea.org/databank/nds/jeff/3-3/epj-publication`

Future projects
Collaborative output

There are many NRDC projects that are ideal for transition to such a system

- Dictionaries
- Joint documents/manuals (e.g. LEXFOR)
- Specifications/proposals (e.g. new formats)
- Data transmissions
- NRDC summary reports ??

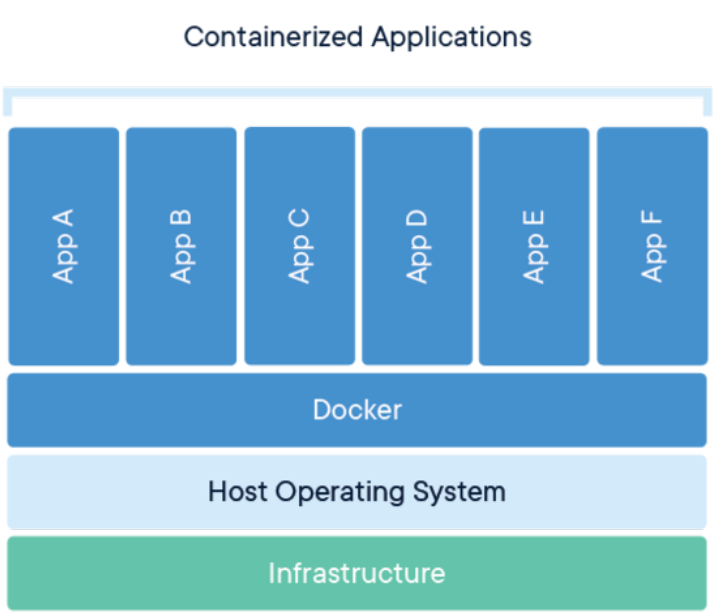
GitLab Runners
Autmated testing

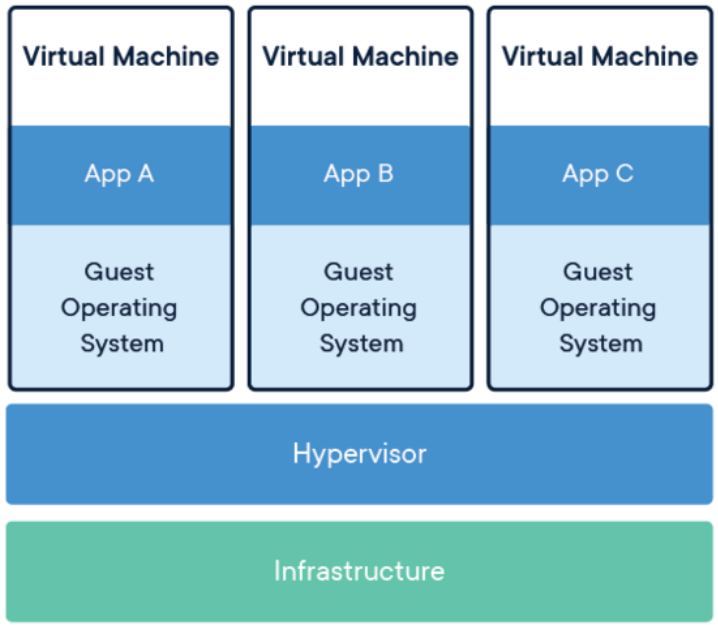
- Basic idea:
  - Any commit to a repository must be checked
  - Send jobs to a machine, which performs the job and reports back
- With any project, **we must have the tests defined**
- Very easy to add the 'Runner' onto a machine and register it to the system
- However, each machine or VM requires administration

Source: GitLab

Containers
Automated testing

- Containers offer a more flexible and compact solution
- Configuration directly controlled (e.g. by 'Dockerfile') and built directly into images
- Images highly portable, reproducible, traceable





Source: Docker

All codes can be put in containers

- We can select different containers for different codes - i.e. different OS, version, compilers, etc.
- These containers can be transferred to (virtually) any machine/system
- We can run different containers for the same CI
- Any process can be executed and outputs saved
- We can download on-the-fly (e.g. dictionaries) to ensure processes integrate up-to-date dependencies

However, we must consider licences, export controls and any other restrictions – part of the reason to provide on-site (non-cloud) services

Final thoughts
Discussion

- NEA has started with new GitLab, with multiple use cases
- In some cases, can revolutionise our processes
- Allows participants to directly engage with outputs
- Project management tools to track progress, discussions, milestones, etc. and tie these to contributions
- Runners perform tests in a much more rigorous (and less time consuming) way than humans, ensuring superior QA
- With web interface, **requires very little expertise!**

How can we start using it for NRDC activities?

Thank you for your attention