EUROPEAN
SPALLATION
SOURCE

Apples to Oranges

A Comparison of EPICS Build and Deployment Systems

PRESENTED BY SIMON ROSE

2023-10-09

Agenda

1 History e3

2 NFS e3

3 Conda e3

4 Conclusions

South africa is beautiful

This has been a dream trip

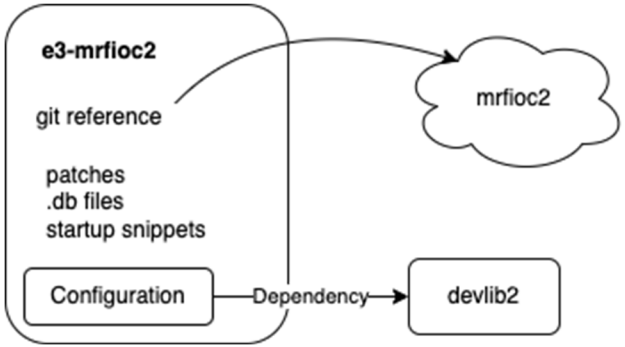
1

History

Introduction and History

Who or what is e3?

- e3 (ESS EPICS Environment) is based on PSI's require module
- Uses dynamic loading instead of static builds or dynamic linking
- Big idea is use of wrappers: repositories that *link* to community source code but contain site-specific configuration and modifications
- This is in contrast to maintaining site-specific forks



2

NFS e3

Current state

We've worked hard to get this far

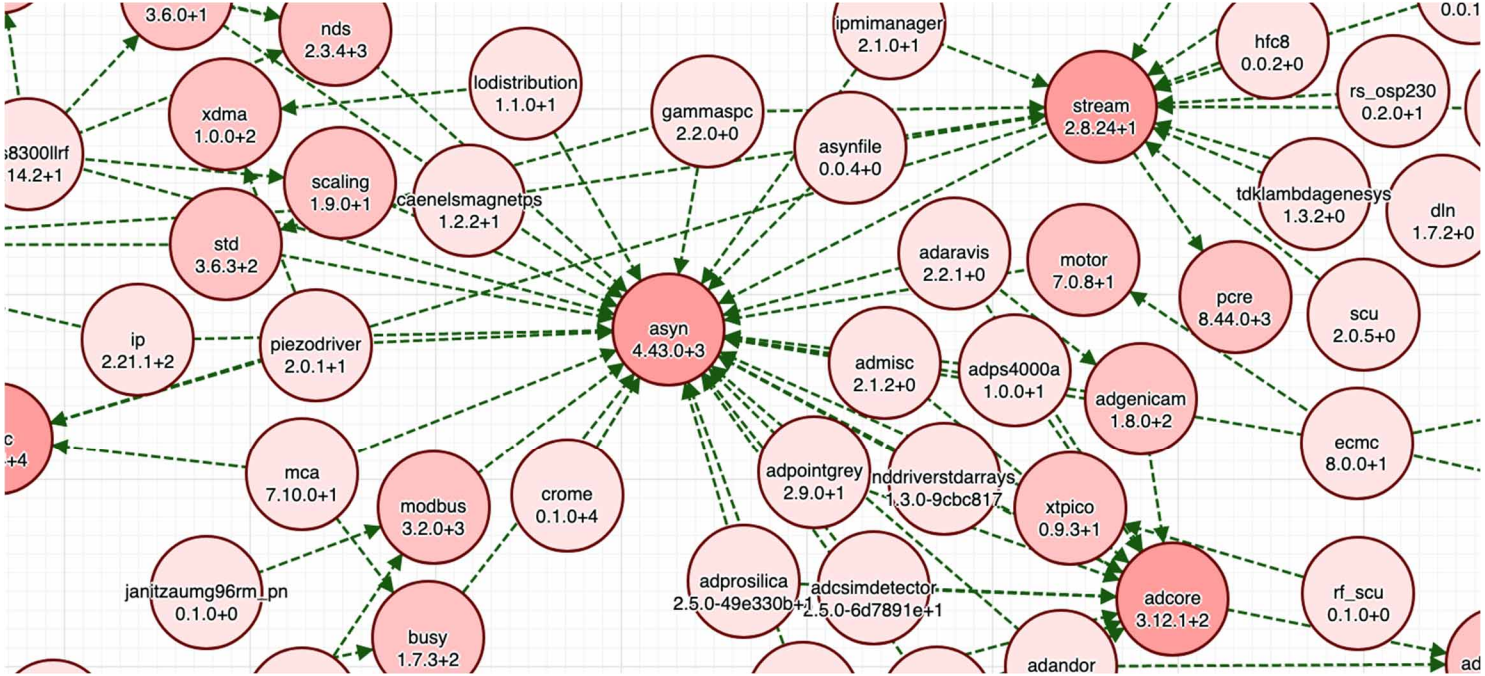
- Many updates to build backend (require)
- Custom build frontend (e3-build) which is used to manage global NFS-deployed EPICS environments using specification files
- GitLab-CI is used to both test changes to modules as well as handle releases to production
- EPICS environment is managed by small team to avoid combinatorial explosion

Specification file

```
config:
  base: 7.0.7-NA/7.0.7-37d472c
  require: 7.0.7-5.0.0/5.0.0-6a40805
metadata:
  type: specification
  version: 1
modules:
  adandor:
    versions:
    - 7.0.7-5.0.0/2.8.0-6f3a1f4+1
    - 7.0.7-5.0.0/2.8.0-6f3a1f4+2
  adcore:
    versions:
    - 7.0.7-5.0.0/3.12.1+2-50b90f0
  adsupport:
    versions:
    - 7.0.7-5.0.0/1.10.0+2-205ab18
  asyn:
    versions:
    - 7.0.7-5.0.0/4.43.0+3-871c171
  autosave:
    versions:
    - 7.0.7-5.0.0/5.10.2+2-45dc1de
  busy:
    versions:
    - 7.0.7-5.0.0/1.7.3+2-1ea0f0a
```

Dependency graph

That's a lot of arrows…



Challenges

…But there are a lot of problems still

- Dependencies are hard
  - EPICS/non-EPICS dependencies
  - Build/runtime dependencies
  - Extremely simple dependency resolution (because this is hard!)
  - Why are we solving this ourselves?

- Using a custom build system is hard
  - EPICS base does a lot, why can't we just use that?
  - It's… complicated

3

Conda e3

Current state

Plus ça change

- Basically uses the workflow and tools defined by B. Bertrand in doi:10.18429/JACoW-ICALEPCS2019-MOPHA014
  - Continuation of that proof-of-concept
- Used by Neutron Instrumentation group
- In short, uses open-source python-based environment and dependency manager Conda to manage IOCs
- Stores EPICS modules in self-hosted artifactory server instead of on NFS

Environment file

```
dependencies:
- epics-base=7.0.7
- require=5.0.0
- essioc
- ethercatmc=4.4.2
```

Challenges

Double plus ça change

- Initial challenges are still present
  - Some automation not in place to deal with certain special systems
  - No clearly defined workflow to manage production environment module releases that will ensure oversight and a lack of combinatorial explosion

4

Conclusions

Conclusions

A recipe for wrappers

- Both NFS- and Conda-based e3 use the same rough concepts
  - Dynamic runtime loading of support modules
  - Use of wrappers to separate site configuration from community code
  - Custom build backend used to package support modules
- Big difference is environment management
  - Conda is designed for this purpose
    - But needs some care to ensure proper control
  - NFS e3 takes a lot of custom work and an entirely new build frontend
    - Lots of different dependency-related issues
    - Many have been solved to the degree that “it works” but not ideal; lots of technical debt still exists

Conclusions

It's dangerous to go alone

- Instead of starting from scratch, why not use community tools?
  - This is true both for environment management but also for custom build rules
- IOC deployment is designed to be (mostly) agnostic to NFS/Conda
- Maybe revisit Conda in time for SCL commissioning?

5

Questions?

Finish presentation