HSF Build and Packaging Working Group

Graeme Stewart & Ben Morgan

WLCG/HSF Workshop Naples, 2018-03-28

The potted history

- One of the first and most active HSF groups (Liz Sexton-Kennedy and Benedikt Hegner driving things)
- Prepared an HSF Technical Note looking at the main build tools
    - Community and FOSS - main advantages and disadvantages summarised
- Handed over group to new coordinators last Autumn (Graeme Stewart and Ben Morgan)
    - Tracked updates on tools that were promising candidates, e.g., Spack
    - Looked again at some FOSS solutions: portage and nix
    - Asked what we learned from successful community tools: AliBuild
- Broadened a little the scope - not just building and packaging:
    - Challenges of deployment (CMVFS, containers)
    - Development environment (for ‘end user’ developers)
- Recently we released v1.0 of the Use Cases and worked on a test stack for re-evaluating solutions (‘test driving’)

Why this matters... and matters a lot!

- We have a real problem to solve with build, packaging, deployment and development
  - This has been the work of the group so far
  - Best practice and common tools would help a lot with today’s workflows
- Now hope to have more community software projects
  - Prototypes that need to be tested by a wider user group
  - Also projects from FOSS world that pop up and might be useful
- This makes having advice on how to structure projects, build them and integrate them very important
  - Project Guidelines and Project Template (to be covered later)
  - Integrate a standard build tool recipe

Use Cases

- Group worked a lot on what the use cases actually are
  - The idea being to properly motivate all of the requirements that we derive for the packaging tools
- Settled last week on Use Cases v1.0
  - Note that these are not very formal use cases, and some requirements are derived in the same document
    - We felt this provided the most useful document
  - Not ruling out evolution from here as we actually go to testing, but it’s a solid beginning
- Modus operandi
  - Find the largest set of common needs
    - Not every experiment needs to use all features
  - Knowledge sharing (e.g., build recipes) is one of the most useful features we strive for
    - Sharing inside HEP is good, with a wider science ecosystem better, full FOSS best
  - May not end up with ‘one tool to rule them all’ - a recommended suite is also an outcome

Use Case Highlights

- Deterministic builds
  - We'd better be confident we can redo what has been done...
- Flexibility
  - Be able to express what we want easily
- Use system packages if desired
  - *caveat emptor*
- Incremental builds
  - Build stack in several chunks, allows for sharing of common layers
- Efficient
- Install time relocation
  - Multiple destinations for one build, e.g., CVMFS, /usr/local/, /my/supercomputer (inc. containers)
- Patching process
  - What if something needs patched (canonical examples - OpenSSL bug; new generator; skim bug)

Next Steps

- We now have a test stack - some basic set of HEP packages that can be useful for some experiments
- Ben has laid out a procedure to test drive the stack with tools we want to evaluate
  - Spack (LLNL tool, HPC origins, wide science base)
  - Portage (Generic FOSS, packager for gentoo linux)
  - Nix (Functional build and packager tool, used in NixOS)
  - AliBuild (ALICE tool, developed and refined ideas from CMSBuild)
  - LCGCMake (Used to build LCG stacks)
  - ...? (This is quite a dynamic area)
- Test stack is just a starting point - then we need to 'stress' against the more challenging use cases
- Aim to have some preliminary conclusions in time for CHEP parallel talk
- **Packaging requirements: what are the contentious points and how do we go forward?**
- **How do we scale to multiple project which have completely different timescales and requirements in terms of software stacks?**

Questions from Giulio and Martin