HSF Packaging Group Meeting - Introduction and Use Cases

Graeme Stewart and Ben Morgan

2017-11-29

Meetings

- Our meeting today will cover 3 topics
  - Progress and discussion of the use cases we have written down (this talk!)
  - ATLAS build and deployment presentation
  - Portage as a build and packaging solution
- We plan to have one more meeting this year
  - 13 December
  - This will focus on deployment issues, with CVMFS and Containers covered
    - Jakob already agreed to give us a talk on this
  - We can probably have one more presentation too

CHEP Paper

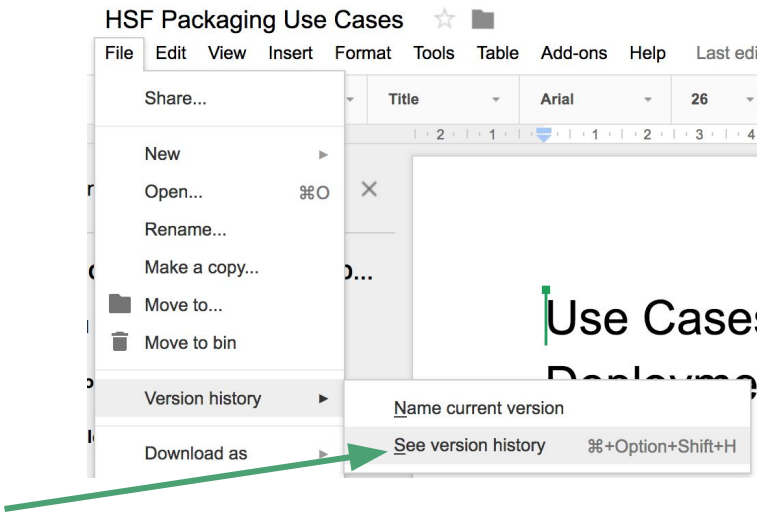
- The CHEP 2018 call for papers is out
  - Deadline for submission 31 December
- Ben and I considered that the work of this group is an interesting subject that merits a paper
  - This is not a paper on a specific packaging tool
    - We are sure that people in this group will probably do those themselves
  - Should describe the process of the group
    - Use cases and desiderata
    - An assessment of the available tools
    - Our conclusions about recommended solutions
  - This will be updated in light of the work we do next year
    - CHEP itself is in July
- We can write an abstract and share it with the group, approving in the next meeting

Bonus Item

- The HSF Community White Paper second draft was released on 17 November
- Comments will close very soon
  - Monday 4 December
- We are encouraging the widest possible readership from the HEP community, so please take a look
  - The section on Software Development is the most directly interesting for this group
- You are also invited to sign your support for the roadmap
  - This is open to all, not just the people who wrote it

Use Cases Document

- After the last meeting we got a lot of comments on the use cases document
  - Many thanks for all that input
- We present here the main updates
  - Where there is consensus we just update the document
  - Where it's less clear we present the points here for more discussion
  - Reminder:
    - You can always use the Google Docs version history to compare with previous iterations
    - The exact version from the last meeting is named "v0.1 presented 2017-11-15"



The Easy Things

- Build 1.a - clarify that build options for dependencies have to be able to be specifiable (for variants)
- Build 4.a - the reuse of binary artefacts when building is considered a **requirement** now (upgrade from *highly desirable*)
- Development 1.a - setting up runtime “views” is considered a **requirement** now (upgrade from *highly desirable*)

Clarifications

- There are a few nice clarifications and pieces of extra information now
  - Defining “global” build options, e.g., for a particular micro-architecture
  - Recipe hierarchy implementations
  - How to manage install time relocation, with a link to the AliBuild code that does this
  - Examples of development environment setups
- These comments don’t affect the text too much, but we should capture this knowledge somewhere
  - Relying on the Google Docs comments history is not really sufficient

Other Points

- Giulio made a mega-comment at the top of the document
  - *The system is not only for building and packaging externals. It needs to encapsulate and take into account the whole process, including the ability to develop one or more of the externals.*
    - Agreed - this is a definite usecase, e.g., integrating a new external
  - *The system must take into account how sources are fetched, backed up and visualised in a coherent manner. Just pointing to a tarball is not enough.*
    - Not sure exactly what this means...
  - *The system must allow people to pick all the "non-physics" related software from their system installation on their laptop. People already have their brew / macports / redhat development tools installation and definitely do not want yet another copy of gcc just because whatever experiment specific tool does not know that GCC 6.2.0 in /usr/local/homebrew is there.*
    - This is requirement Build 3.a

Continued

- ...
  - *The number of command line options must be reduced to a minimum if they trigger different flavours of builds.*
    - I am not sure what the second part means (first part is fine)
  - *Recipes and tool should be separate and the tool should be backward compatible as much as possible with old sets of recipes.*
    - This doesn't look very possible to me, unless you have very well defined old recipes
  - *...package recipes should not require knowledge of a scripting language other than bash and should limit custom templating as much as possible. Conventions (for environment variables) over complex language features is a must, IMHO.*
    - First part could mean a lot of repetition. Second part fine IMO.
  - *Target audience is not only the librarian, but the whole collaboration, including laptop users.*
    - Absolutely - we do try to identify the "actor" in the document

Summary

- I think we converged quite nicely and it looks to me like we can finalise this soon
- We should try and advertise a draft more widely
- And capture the other interesting discussions in some way