Introducing LCG Views

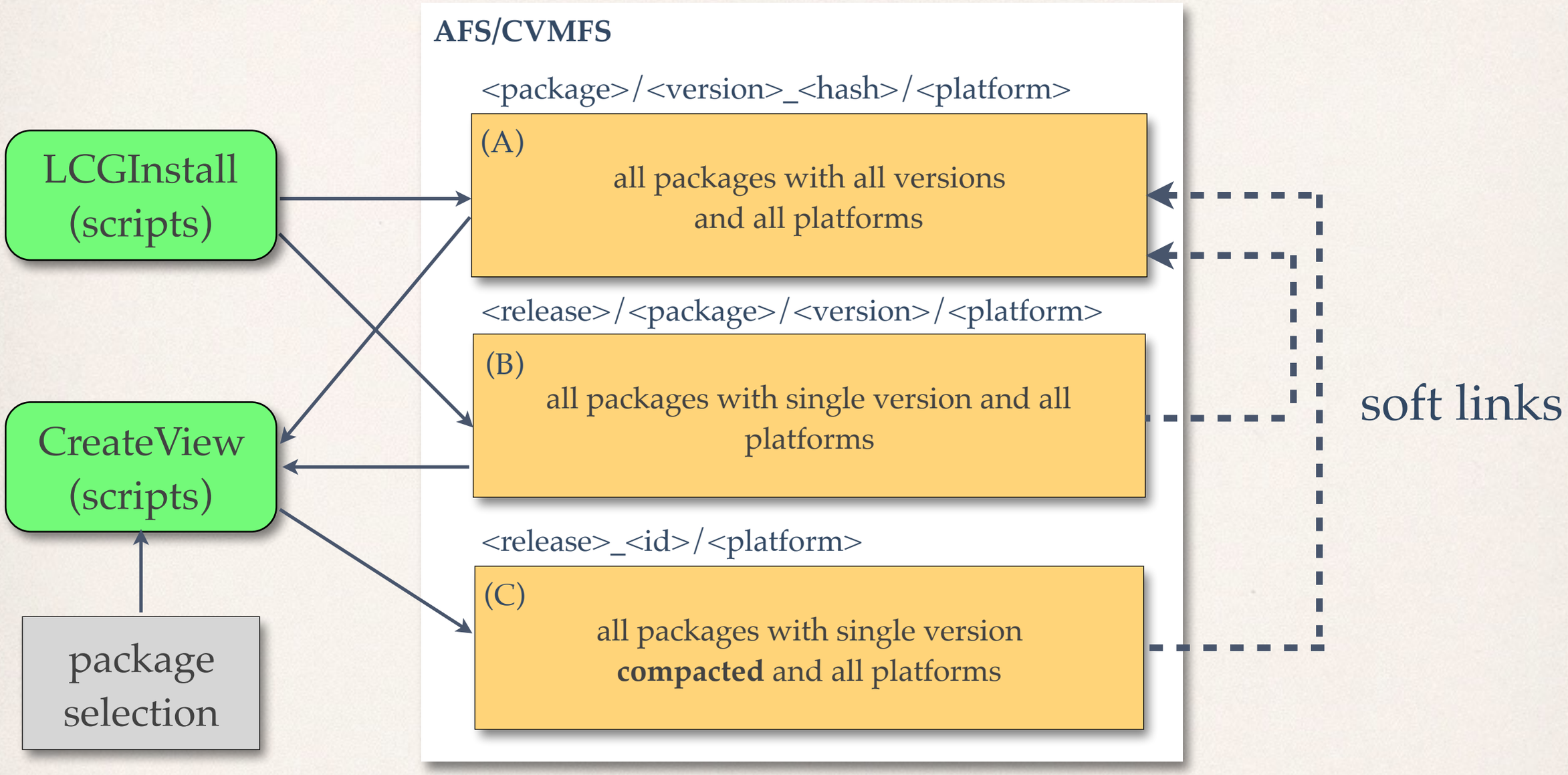
Pere Mato

LIM Meeting, 16th January 2016

Motivations

- Easy runtime environment setup
  - Current methods allow to setup a running environment starting from a top level package/application (following dependencies)
  - Setting-up the environment for a full LCG release is not obvious
- Fast runtime environment setup
  - Scripts to setup the environment are slow (need to scan AFS or CVMFS)
- Faster runtime
  - Reducing the PATH and LD_LIBRARY_PATH can speedup runtime
- Generic Docker images
  - Docker images making use of the software in CVMFS
  - Time to setup environment >> time to start the Docker container

What's a View




```
/cvmfs/sft.cern.ch/lcg/views/<release>/<platform>/bin
                                                 lib
                                                 include
                                                 ...
```

Selection

- Defining the set of packages
  - The default is to take all packages in a LCG release
  - Sub-setting is possible
- Only one version per package
  - Need a rule for multi-version packages (i.e. MC generators)
  - Packages with major versions (i.e. Qt vs. Qt5)
- Selection of top level directories
  - Many unneeded files with big chances of clash (README, …)
  - Current list: [‘aclocal', 'cmake', 'emacs', 'fonts', 'include', 'macros', 'test', ‘tests', 'bin', 'config', 'etc', 'icons', 'lib', 'lib64', 'man', 'tutorials', 'share']

Building an LCG View

```
lcgcmake/cmake/scripts/create_lcg_view.py \
   -r <release>                           \
   -p <platform>                          \
   -l /cvmfs/sft.cern.ch/lcg/releases     \
   -d -B                                  \
   /cvmfs/sft.cern.ch/lcg/views/<release>/<platform>
```

- Script developed by Ivan Razumov
- 1-2 minutes to produce the view
- File clashes while creating the soft-link are properly reported
  - Possibility to fix them by fine-tuning the package ‘backlist’, top level directories, etc.

Environment Setup

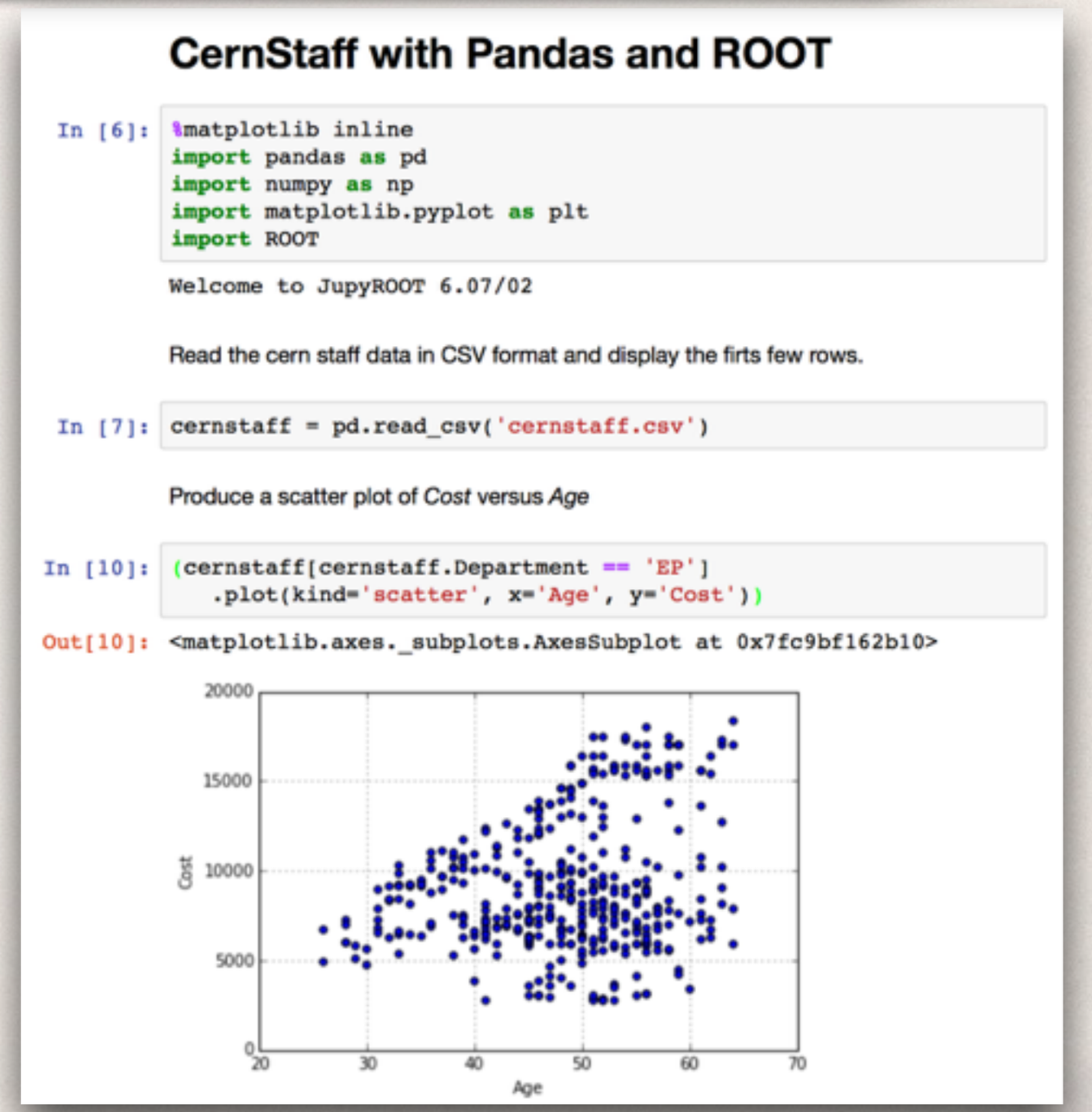
```
source /cvmfs/sft.cern.ch/lcg/views/<release>/<platform>/setup.[c]sh
```

- Sourcing a single and simple file sets the full environment for the complete view. It defines trivially:
  - PATH, LD_LIBRARY_PATH
  - PYTHONPATH
  - CMAKE_PREFIX_PATH
  - ROOTSYS, ROOT_INCLUDE_PATH
- Other variables can be added if needed…

Use Cases: ROOTbooks

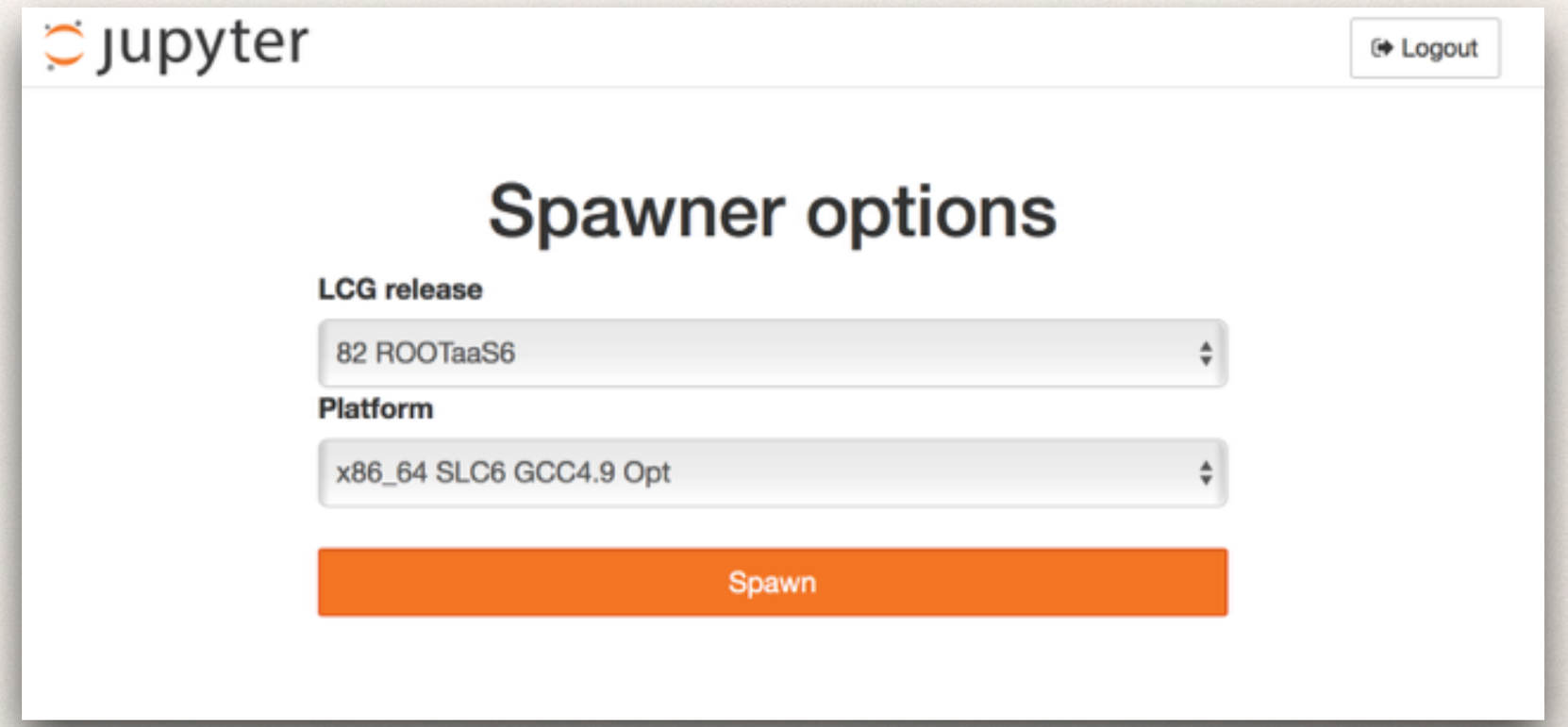
```
source /cvmfs/sft.cern.ch/lcg/views/LCG_82rootaas6/x86_64-slc6-gcc49-opt/setup.csh
root --notebook
```

- The command will open a web browser in which the use can create his/her notebooks
- The notebook will inherit the full environment
  - All Python modules available
  - All HEP libraries available
    - E.g. ROOT, Fastjet, Geant4, …



Use Case: ROOT as a Service

- ROOT as a Service ongoing development
  - Based on CERN SSO, EOS, CERNBox, CVMFS, Dockers, OpenStack, etc.
- The user is able to select the LCG software release at the time of spawning it own server



Use Case:Building Software

```
#-Build and Run the Event ROOT example-------------------------------
wget http://root.cern.ch/download/event.tar.gz
tar -zxf event.tar.gz
cd event/build
cmake ..
make -j10
./Run
```

ROOT

```
#-Build B1 basic Geant4 example--------------------------------------
mkdir build; cd build
cmake <lcg-view>/share/Geant4-10.1.2/examples/basic/B1
make -j10

#-Run B1 basic Geant4 example (needs data files)----------------------
setenv G4DATA /cvmfs/geant4.cern.ch/share/data
setenv G4NEUTRONHPDATA $G4DATA/G4NDL4.5
...
./exampleB1 run1.mac
```

Geant4

Conclusions

- LCG views has been demonstrated
  - Thorough validation is still required
  - Already useful for a number of use cases
- A number of problems needs to be tackled
  - Delivery of package data (e.g. Geant4, LHAPDF, etc.)
  - Installation of *cmaketools* package needs some tuning
- Project integration and testing builds on top of LCG Views?
- Experiment software on top of LCG Views?

Back-up Slides

Building CORAL

- Specially for Andrea

```
wget http://service-spi.web.cern.ch/service-spi/external/tarFiles/CORAL-3_1_1.tar.gz
tar -zxf CORAL-3_1_1.tar.gz

cd CORAL-3_1_1
mkdir build;cd build

cmake .. -DBoost_NO_BOOST_CMAKE=ON -DBINARY_TAG=x86_64-slc6-gcc49-opt
make -j10
```