An Update on LCGCMAKE:
Using LCG releases without CVMFS

Pere Mato

HSF Packaging Working Group, 7 November 2018

Recap LCGCMake

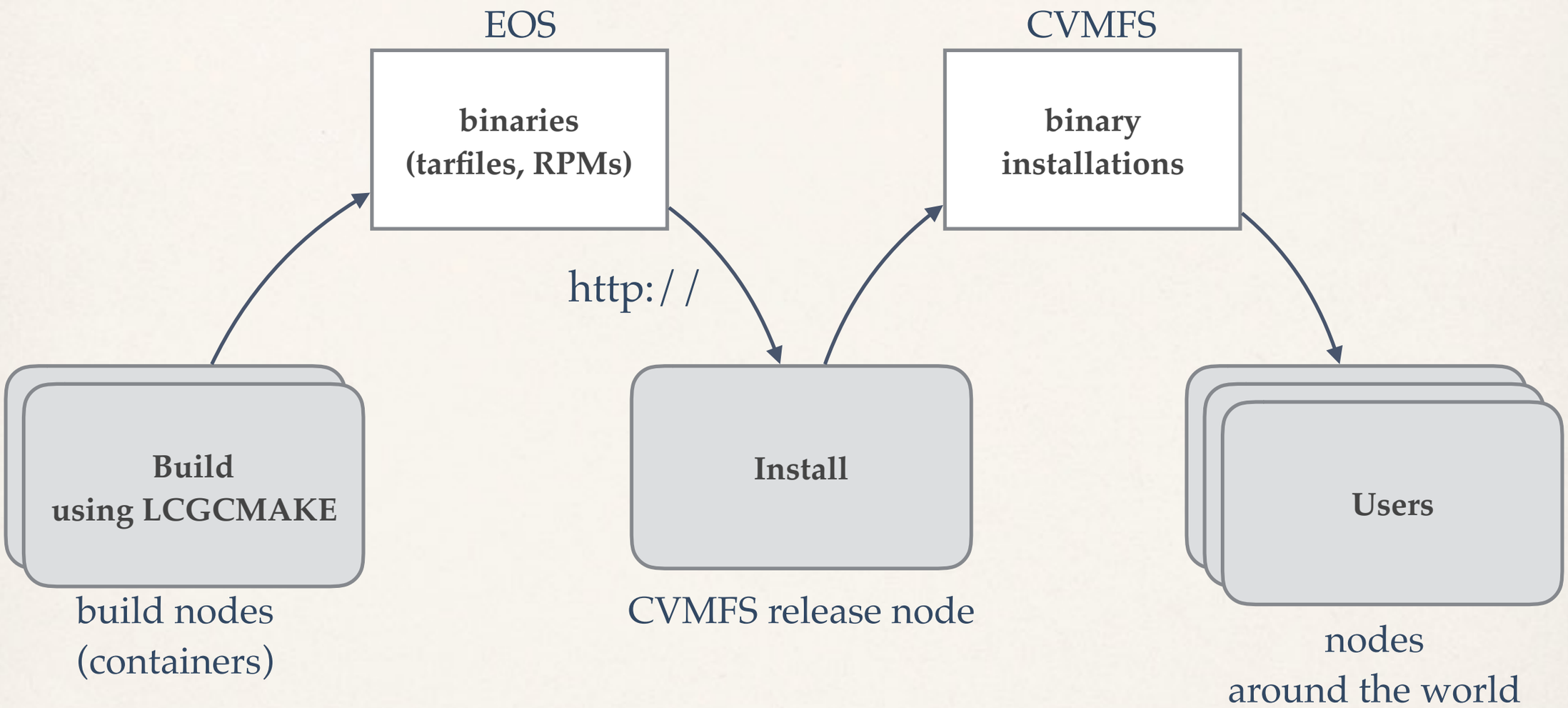
- The LCGCMake is based on the CMake standard module *ExternalProject* that creates custom targets to drive download, update/patch, configure, build, install and test steps of an external package
- A single file lists all the packages and their required versions for a given configuration of the software stack

```
#---agile-----------------------------------------------------------------------

LCGPackage_Add(
  agile
  URL http://www.hepforge.org/archive/agile/AGILe-${agile_native_version}.tar.bz2
  CONFIGURE_COMMAND ./configure --prefix=<INSTALL_DIR>
                                --with-hepmc=${HepMC_home}
                                --with-boost-incpath=${Boost_home_include}
                                --with-lcgtag=${LCG_platform}
                    PYTHON=${Python_home}/bin/python
                    LD_LIBRARY_PATH=${Python_home}/lib:$ENV{LD_LIBRARY_PATH}
                    SWIG=${swig_home}/bin/swig
  BUILD_COMMAND make all LD_LIBRARY_PATH=${Python_home}/lib:$ENV{LD_LIBRARY_PATH}
  INSTALL_COMMAND make install
                  LD_LIBRARY_PATH=${Python_home}/lib:$ENV{LD_LIBRARY_PATH}
  BUILD_IN_SOURCE 1
  DEPENDS HepMC Boost Python swig
)
```

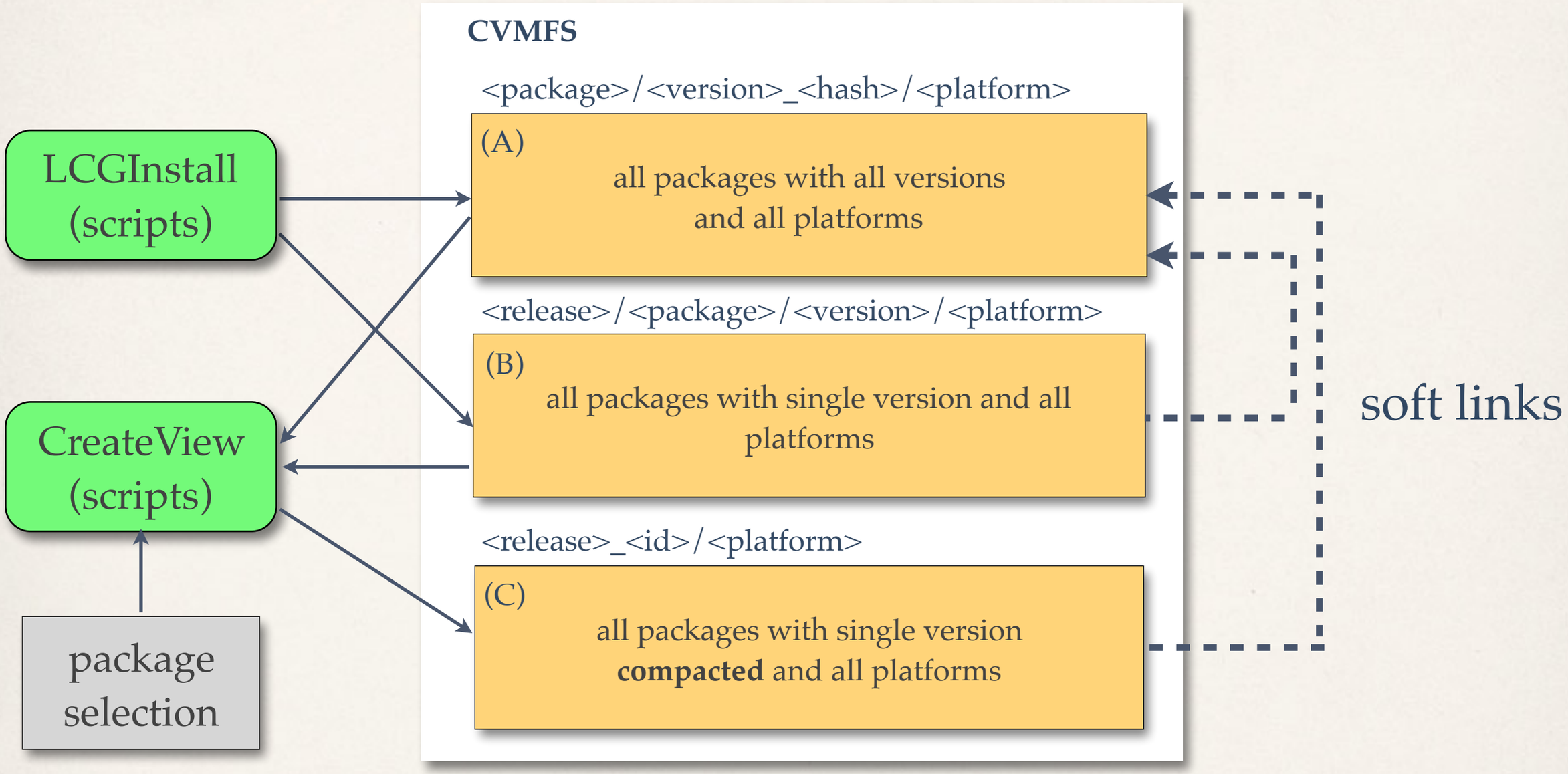
```
# Application Area Projects
LCG_AA_project(COOL  COOL_2_8_17)
LCG_AA_project(CORAL CORAL_2_3_26)
LCG_AA_project(RELAX RELAX_1_3_0k)
LCG_AA_project(ROOT  5.34.05)
LCG_AA_project(LCGCMT LCGCMT_${heptools_version})

# Externals
LCG_external_package(4suite            1.0.2p1                                 )
LCG_external_package(AIDA              3.2.1                                   )
LCG_external_package(blas              20110419                                )
LCG_external_package(Boost             1.50.0                                  )

...

# Generators
LCG_external_package(starlight         r43           MCGenerators/starlight    )
LCG_external_package(herwig            6.520         MCGenerators/herwig       )
LCG_external_package(herwig            6.520.2       MCGenerators/herwig       )
LCG_external_package(crmc              v3400         MCGenerators/crmc         )
LCG_external_package(cython            0.19          MCGenerators/cython       )
LCG_external_package(yaml_cpp          0.3.0         MCGenerators/yaml_cpp     )
LCG_external_package(yoda              1.0.0         MCGenerators/yoda         )
```

https://gitlab.cern.ch/sft/lcgcmake

Current Process



Recap LCG Views




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

Environment Setup

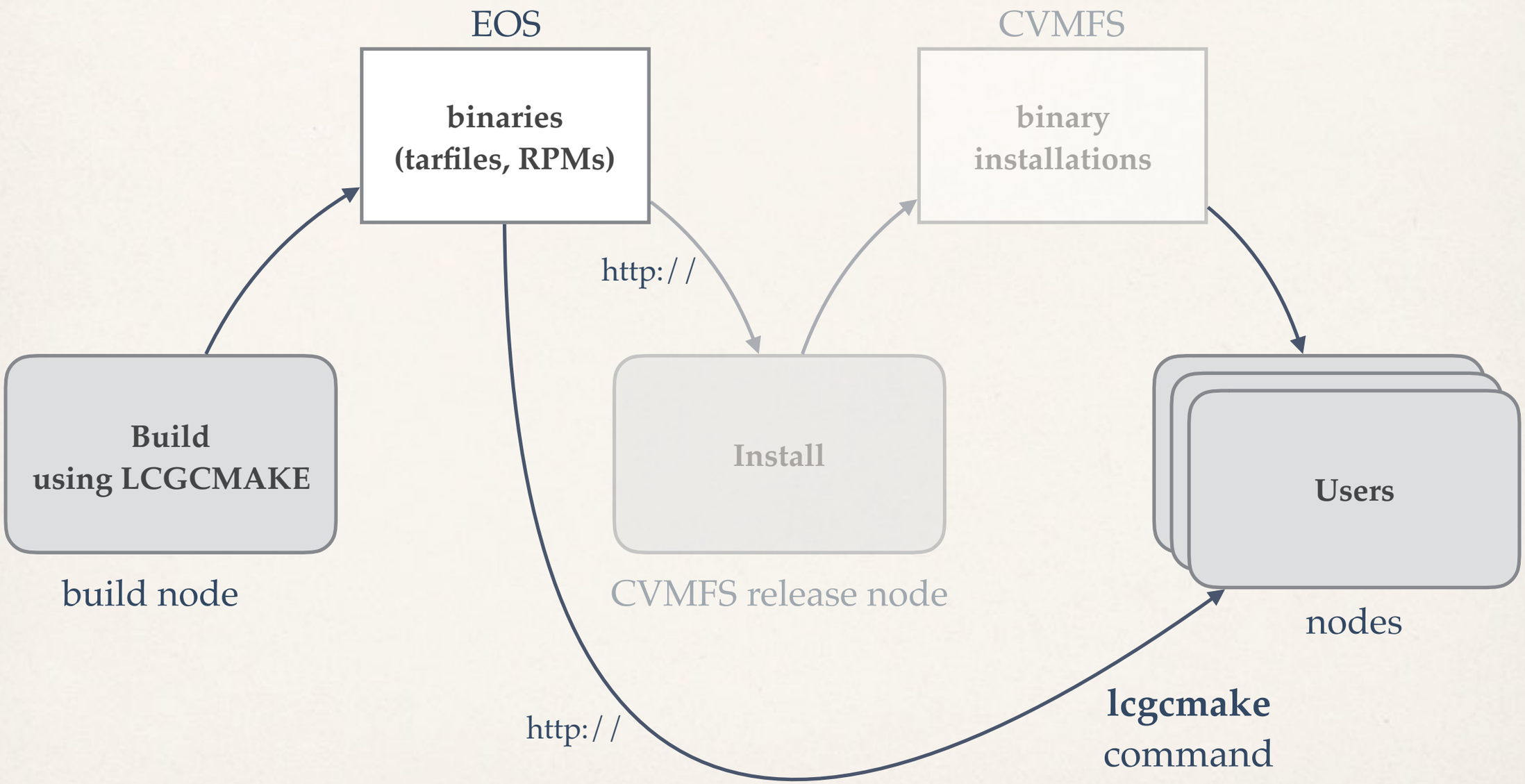
```
source /cvmfs/sft.cern.ch/lcg/views/<release>/<platform>/setup.[c]sh
```

- 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 have been added as needed…

Non-CVMFS Releases

- In some cases you would like to have local installations of the externals, e.g.
  - No CMVFS installed, working in a network-less location, did not find the exact match of arch/os/compiler, etc.
- For this use case we have developed an utility command (‘**lcgcmake**’) to provide local installations
  - downloading existing binaries if matches arch/os/compiler, and exact version of package and depends (hash value)
  - building from sources
  - a combination of both

Non-CVMFS Releases



Installation of 'lcgcmake'

- Download package and set PATH

```
$ git clone https://gitlab.cern.ch/sft/lcgcmake.git
$ export PATH=$PWD/lcgcmake/bin:$PATH
```

- The command **lcgcmake** is now available

```
$ lcgcmake --help

This command drives the full process of building a LCG release using the lcgcmake tool.

positional arguments:
    {version,configure,config,conf,install,show,sh,run}
                        available sub-commands
    version             print the version of lcgcmake
    configure (config,conf)
                        configure the lcgcmake session by selecting the
                        software stack version, etc.
    install             Install or build a set of given targets with all their
                        dependencies
    show (sh)           get information from the lcgcmake session
    run                 run a command with the just installed software stack

optional arguments:
  -h, --help            show this help message and exit
  --verbose             Increase logging verbosity
  -q, --quiet           Decrease logging verbosity
  -a ARCH, --arch ARCH  processor architecture
  -o OSVERS, --os OSVERS
                        operating system keyword (e.g. slc6, mac1013, etc.)
```

Installing externals locally

- Basic commands: configure, install, run

```
$ lcgcmake configure --prefix <installprefix> [--compiler <compilertag>]
$ lcgcmake install <package> [<package> ...]
$ lcgcmake run [<command>]
```

- In addition to the basic command there are a number of useful introspection commands

```
$ lcgcmake show compilers  # to list the available compliers for current platform
$ lcgcmake show versions   # to list the software stack [LCG] versions
$ lcgcmake show config     # current configuration parameters
```

Example: External libraries for GeantV

- To develop GeantV requires to setup an environment with a large number of externals packages:
  - ROOT, Python, Boost, fftw, graphviz, GSL, mysql, xrootd, R, numpy, tbb, blas, pythia6
  - Geant4, XercesC, CLHEP, expat, hepmc3
  - VecGeom, umesimd, veccore,Vc, veccorelib
  - benchmark, ninja, CMake

```
$ lcgcmake configure --prefix <installprefix>
$ lcgcmake install GeantV-externals  # equivalent to 'ROOT Geant4 VecGeom ...'
$ lcgcmake run
```

Example: Developing GeantV

- Once the installation of all the externals is finished, execute the standard commands to build GeantV

```
$ git clone https://gitlab.cern.ch/GeantV/geant.git
$ mkdir build; cd build
$ cmake ../geant/ -DCMAKE_INSTALL_PREFIX=<...>  -DCTEST=ON
...
$ make -j10
...
$ ctest
Test project /build/geantv/build
    Start 1: TestEm3_GV
1/2 Test #1: TestEm3_GV .......................   Passed    8.55 sec
    Start 2: TestEm5_GV
2/2 Test #2: TestEm5_GV .......................   Passed    7.61 sec
```

Conclusions

- The command ‘lcgcmake’ have been tested on:
  - MacOS 10.13 (‘lcgcmake run’ need to be replaced with ‘source …’)
  - Ubuntu 18.04 (require a number of prerequisites; see DockerFile)
  - SLC 6 (gcc 6.2) (with HepOSLibs)
  - centos 7 (gcc 7.3) (with HepOSLibs)
- Note that the use of the binaries will only work for exact match of platform identifier (os, compiler, build type)
- Give it a try. Your feedback and complains will be appreciated!!

Additional Slides

Dockerfile for ubuntu18

```
FROM ubuntu:18.04
MAINTAINER Pere Mato <pere.mato@cern.ch>

ENV LCGCMAKE_ROOT=/usr/local/lcgcmake    \
    ARCH=x86_64 \
    DEBIAN_FRONTEND=noninteractive

RUN apt-get update && apt-get upgrade -y
# Tools necessary for installing and configuring Ubuntu
RUN apt-get install -y \
    python git cmake nano \
    g++ gcc gfortran binutils uuid-dev libssl-dev \
    libx11-dev libxpm-dev libxft-dev libxext-dev libmotif-dev \
    mesa-common-dev libglu1-mesa libxi-dev libxmu-dev libglu1-mesa-dev \
    libncurses5-dev zlib1g-dev libcurl4-openssl-dev libreadline6-dev \
    libgdbm-dev tk-dev libgl2ps-dev liblzma-dev
# Setup LCGCMake
RUN git clone https://gitlab.cern.ch/sft/lcgcmake.git $LCGCMAKE_ROOT
RUN useradd -m hsf
ENV USER=hsf
ENV HOME=/home/hsf
WORKDIR /build

RUN ln -s $LCGCMAKE_ROOT/bin/lcgcmake /usr/local/bin/lcgcmake
RUN chown hsf:hsf -R $LCGCMAKE_ROOT
RUN chown hsf:hsf /usr/local/bin/lcgcmake
RUN chmod +x /usr/local/bin/lcgcmake
RUN chown hsf:hsf /opt
RUN chown hsf -R /build

USER hsf
```

FAQ

- What about debug builds of the externals?
  - configure with `--options LCG_BUILD_TYPE=dbg`
- What about the sources for debugging
  - `install <package>-sources` will install the sources in the installation prefix