What SFT is providing

Librarian and Integrators WS
30 May 2018, CERN
Patricia Mendez Lorenzo


We all together provide
common software to
ensure the success of
the LHC experiments…
...the rest are technical
aspects

Outlook of the "*rest*": the technical aspects

1. **Let's see how we contribute to the goal (next 30 min):**
   - What we provide
   - How we do it (including the core elements)
   - Why we do it like this
   - Who and for whom
2. **Let's see how you contribute to this goal (rest of the day)**
3. **Let's ensure we + you = WE**
   - Discussion session at the end of the day

What do I expect from this talk:

-- That you perfectly understand and learn how we work: our procedures, our operations, our tools

-- That we establish single communication points through the LIM with those persons knowing the system

Let’s start with the persons

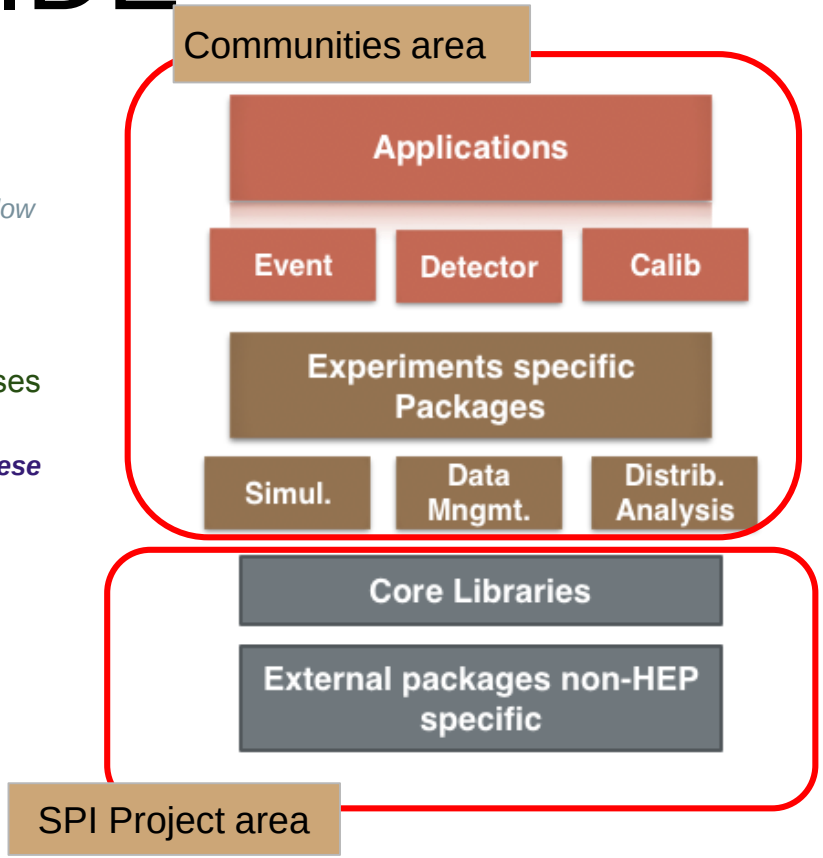
- **SPI manpower:**
  - Patricia Mendez
  - Rafal Pacholek
  - Ilias Goulas
  - Genser team → Dima, Ivan, Gregory
- **Contributors**
  - Pere Mato (GL)
  - Gerardo Ganis (DGL)

Quite a lot of valuable work in collaboration with the CVMFS experts and the rest of members of SFT

- **New members**
  - Emil Obreshkov for ATLAS
  - Shahzad Malik Muzaffar for CMS

WHAT WE PROVIDE

- **~300 packages among:**
  - Common SFT projects: ROOT, Geant4
  - ~270 external packages including python packages
    - *Ecosystem for Machine Learning tools including Tensorflow and Keras*
  - 55 generators and 18 Grid (specific MW) packages
- **Distribution of binaries to CVMFS**
  - For 10x2 platforms for Python2 and Python3 → Releases
  - Extra platforms currently under testing → Nightlies
    - ***LHCb and ATLAS can/should test their SW on top these builds for ROOT validation purposes***
- **Binaries packaging in tar files and rpms formats, distributed to EOS**
  - Recently in docker containers also
- **Common tools in CVMFS**
  - CMake, gcc,clang, etc
- **Common infrastructure for the SFT projects**
  - Jenkins server, build nodes



Supported platforms

```
Platform = x86_64+({architecture})-${OS}-${COMPILER}-${BUILD_TYPE}
```

(Definition of the platform based on the HSF recommendations)

Compilers(*)	Architectures (**)	Operating systems	Build type
gcc62 gcc7 gcc8 clang60 native (ubuntu)	avx2+fma	slc6 Centos7 ubuntu16 ubuntu18 mac	Release Debug

- (*) Previous gcc versions available for generators (on demand)
- (**) Infrastructure included in the system, not provided yet in releases nor nightlies

Our challenges

1. **Build results reproducibility**
   - Enabled via the implementation of HASHES for each package
2. **Scalable approach in:**
   - Package dependencies
   - Number of platforms
   - New/existing packages
   - Builds based on an incremental approach
3. **“Prefix” independent distribution → reallocation**
4. **Limited manpower → automation**
5. **Quality assurance → software validation**
   - Rafal’ presentation
6. **Fast improvement/feedback cycle**

The rest of the talk will explain how we cope with these challenges

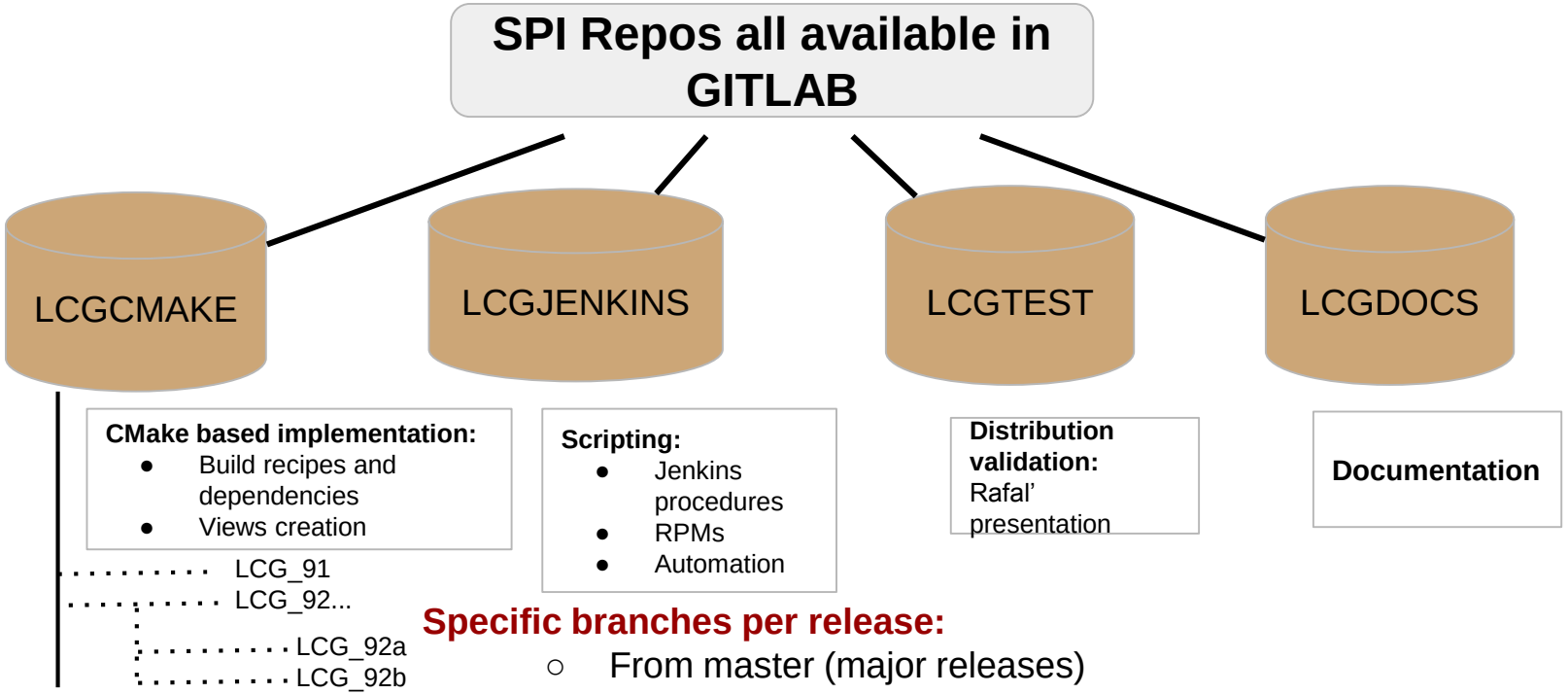
Significant number of challenges common to many projects that can be commonly managed:

HSF Graeme’s presentation

Keywords/ideas for each challenge

- **For reproducibility**
  - Sources stored in EOS independent of changes in code
  - Specific branches per release → modified for new generators only
  - Handling of the “internal” packages dependencies by us
- **For escalation**
  - Build software structure based on specific cmake modules handling dependencies
  - Modular approach for the implementation of new packages and versions
    - *compiler, OS, architecture independent*
- **For reallocation**
  - Packaging installation easily adaptable to any “prefix” and reallocation software executed after each package expansion
- **For automation**
  - Jenkins structure and own scripting
- **For continuous improvement**
  - Operations based on a fast feedback implementation

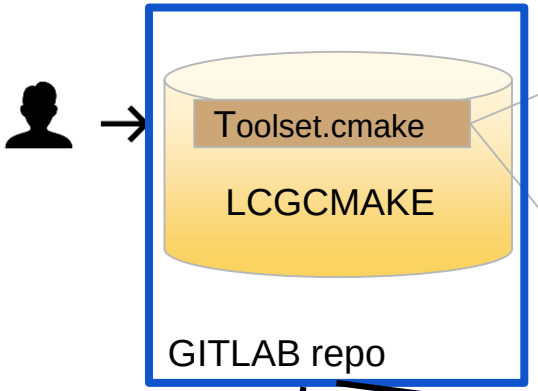
Summary of our projects



Specific branches per release:

- From master (major releases)
- From existing releases (minor releases)

Build procedures: core elements (I)

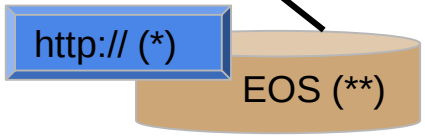



```
# Application Area Projects
set(heptools_version  dev3)
[...]
LCG_external_package(Jinja2  2.10)
LCG_external_package(jpype   0.6.2)
LCG_external_package(jupyter 1.0.0)
```

1) Sources download

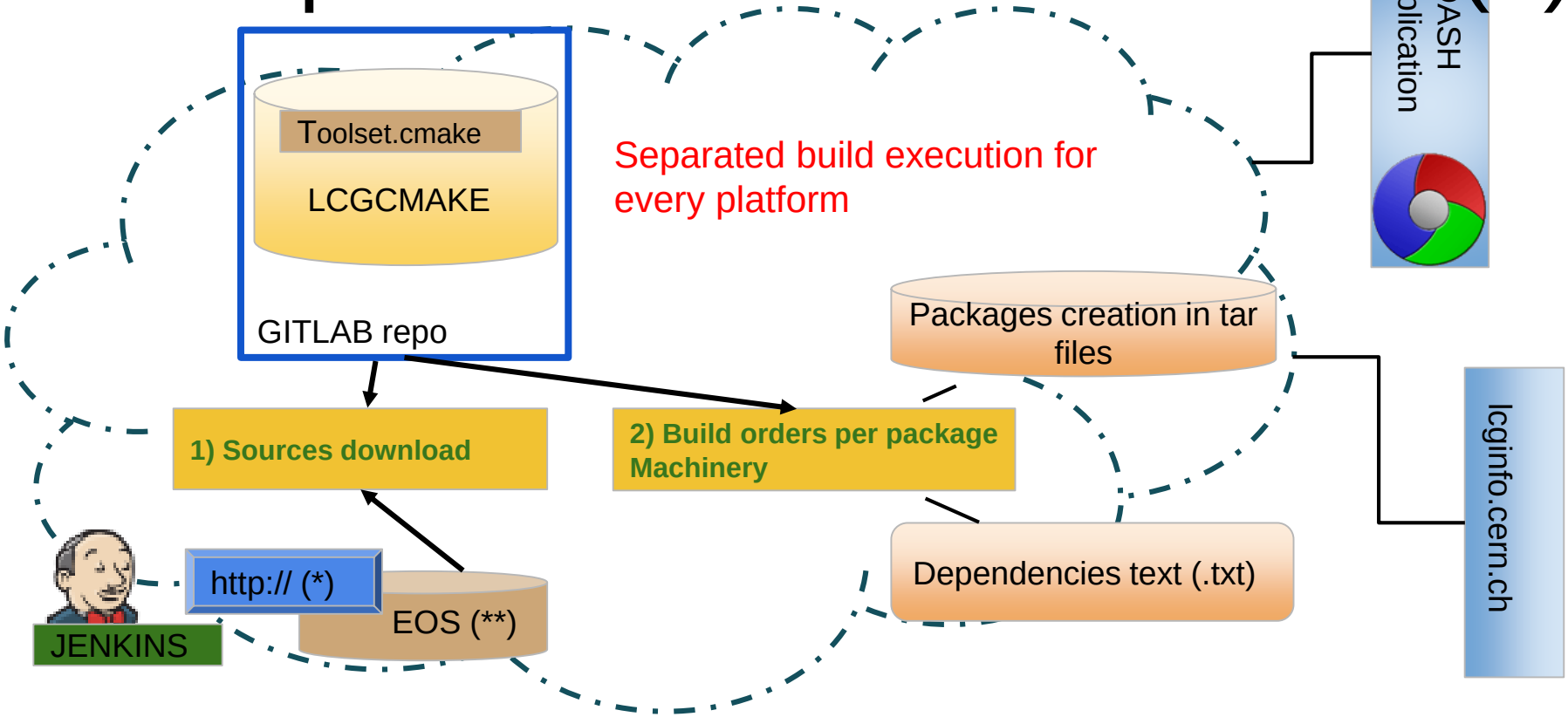
2) Build orders per package Machinery

- Branch (release/nightly)
- Platform

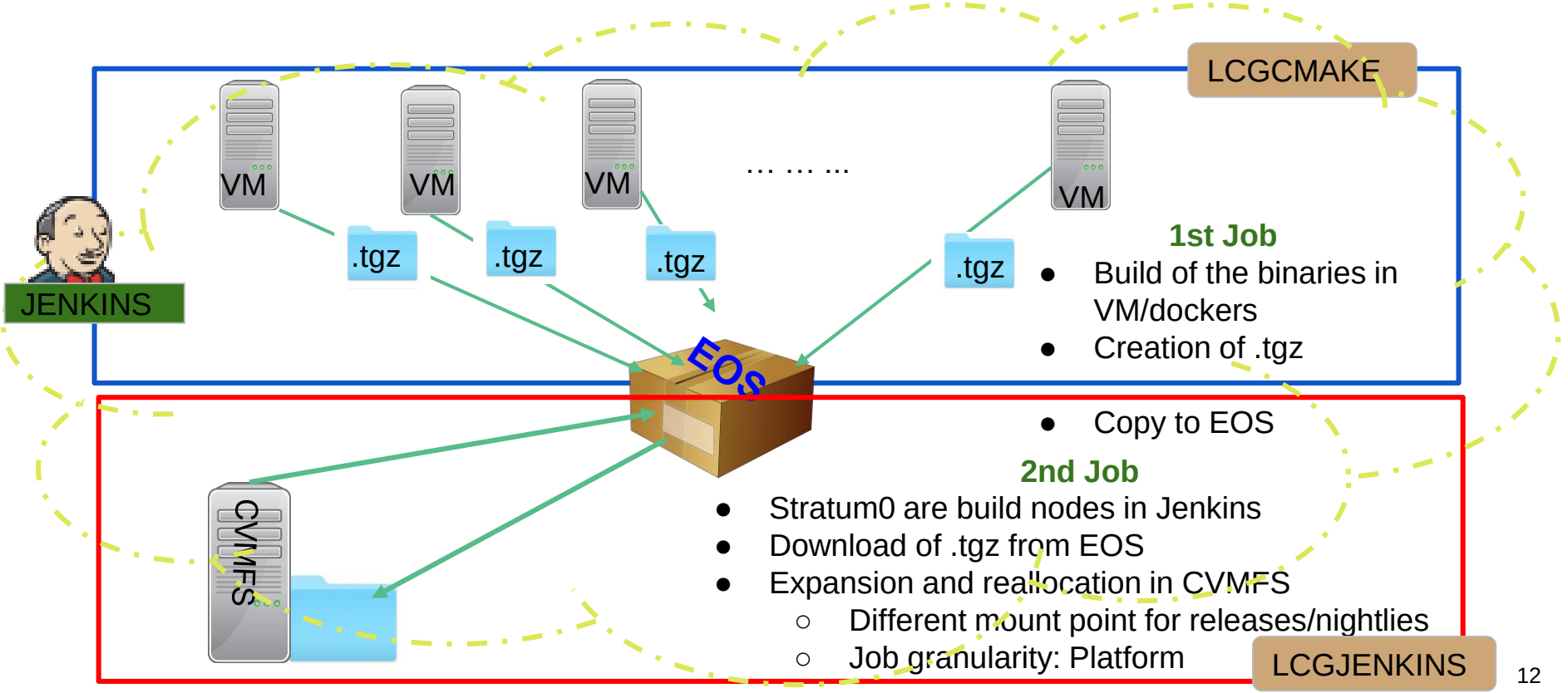


(*) http://lcgpackages.web.cern.ch/lcgpackages
(**) /eos/project/l/lcg/www/lcgpackages

Build procedures: core elements (II)



Software distribution workflow



Building from sources → LCGCMAKE Repo

- **Build and testing infrastructure CMAKE based:**
  - Build specifications included in platform-independent list files
  - Generation of CMakeFile driving the full build
  - Modular and scalable system
- **SFT toolkit infrastructure based on cmake: LCGCMAKE**
  - Based on “ExternalProject” CMake module implementation
    - *Enable builds from external software sources*
    - *Easy way to control package dependencies*
  - Repository available in **GITLAB** → *Specific branch for every release*
  - It includes:
    - *Specific tests per package if needed*
    - *Individual packaging in tar files*
    - *Creation of views for both nightlies and releases*

Default packaging

- **lcgcmake packs the binaries in .tgz files**
  - One summary .txt file including packages dependencies available per platform

```
# Concatenation of all the .buildinfo files for a given LCG version
 COMPILER: GNU 6.2.0, HOSTNAME: lcgapp-slc6-physical1.cern.ch, GITHASH: 'd35450d', HASH:
edbe7, DESTINATION: lcgenv, DIRECTORY: lcgenv, NAME: lcgenv, VERSION: 1.3.5, REVISION: ,
DEPENDS:
 COMPILER: GNU 6.2.0, HOSTNAME: lcgapp-slc6-physical1.cern.ch, GITHASH: 'd35450d', HASH:
dc723, DESTINATION: absl_py, DIRECTORY: absl_py, NAME: absl_py, VERSION: 0.2.0, REVISION: 1,
DEPENDS: Python-2.7.13-b163d,setuptools-36.0.1-49883,
```

- **At the end of the build tar files are copied to from each individual build node to EOS → LCGJENKINS repo**
  - **Incremental approach being the HASH (determined by the version-revision-dependencies) a key parameter**
  - Separated areas for releases and nightlies in EOS
  - Dependencies .txt file is also copied for installation purposes

Binaries Packaging

Three packaging infrastructures supported

1. **Nightlies/Releases: .tgz files created in default after the individual build of any package**
2. **For Releases ONLY: RPMS created after the build in each individual VM through the infrastructure created by LHCb**
   - Code/repository managed by LCGJENKINS and adapted to our needs
   - RPMS repository migrated from AFS to EOS
3. **For Releases only: Provision of docker containers in a flexible way**
   - Total or partial number of packages per release
   - slc6/centos7 OS images downloaded from the official CERN repository
   - Created containers uploaded to EOS

Binaries Distribution

NIGHTLIES/RELEASES ARE JUST FLAVOURS OF THE SAME DISTRIBUTION PROCEDURES

- **Dependencies handling for tar files installation managed by a summary .txt file**
  - Created for each build, i.e., per each platform and uploaded to EOS
- **CVMFS default end-system**
  - AFS no longer use/maintained unless required by experiments for generators ONLY → Jan' talk in the afternoon
  - We do provide the possibility to distribute to any other system → Interesting for the BE team
- **Binaries installation method automatically handles the following main steps:**
  - CVMFS transaction handling
  - Connection with EOS to download the tar files and the dependencies checking
  - reallocation procedures after installation
- **Creation of views afterwards distributed to CVMFS**
  - Compatible installation of software packages belonging to a LCG release under a single $PREFIX
  - A stable view always available under: `/cvmfs/sft.cern.ch/lcg/nightlies/dev3(4)/latest` → Used by SWAN
    - *Week day unaware: Link to the latest successful week day view*

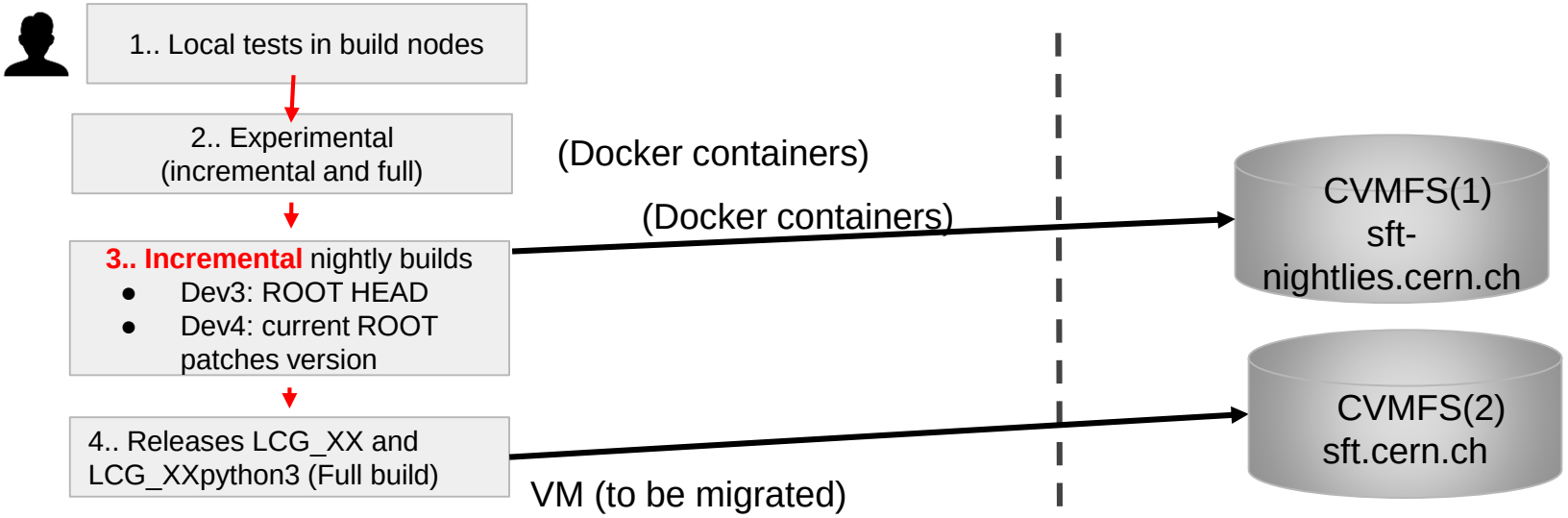
Our CVMFS setups

- **Two Stratum0 separated for releases and nighlies**
  - Releases: /cvmfs/sft.cern.ch/lcg → cvmfs-sft.cern.ch
    - *General tools such as gcc, clang, CMake, git also provided here*
  - Nighlies: /cvmfs/sft-nightlies.cern.ch/lcg → cvmfs-sft-nightlies.cern.ch
    - *Linked from the release area to have a single entry point*
- **New HW infrastructures recently provided by IT**
  - New machine configuration based on local SSDs for publishing purposes
    - *30GB of RAM with temporal cvmfs transactions storage at the local SSDs*
  - Spectacular publication time improvement; factor 6
    - *Effort from our side is still needed due to the high volumes we handle:*

26096 dirs	317822 files
3296 symlinks	17GB space

Release volumes per platform

Managing changes: workflow



Particular cases in the case of Releases

- `Limited` → Reduced number of packages selected by the user
- `Latest` → "Internally" triggered on demand to improve CVMFS speed performance.
  - WARNING-1 : SFT INTERNAL USE ONLY
  - WARNING-2: Do not confuse it with the "latest" views distribution already mentioned

Aspect of the installations in CVMFS

/cvmfs/sft.cern.ch/lcg/releases/LCG_93

```
LCG_externals_<platform>.txt     pytools
LCG_generators_<platform>.txt    MCGenerators
ROOT                             Grid
CORAL                            png
Qt5                              R
[...]                            [...]
```

/<version>/<platforms>/binaries

link to:
../../../png/<version-HASH>/<platform>/binaries
REAL AREA OF INSTALLATION

/cvmfs/sft.cern.ch/lcg/nightlies

```
dev3
dev4
python3
```

Mon Tue Wed Thu Fri Sat Sun

Same tree structure as in releases also linking to the same real area installation to ensure the incremental installation approach

/cvmfs/sft.cern.ch/lcg/views

```
devX
LCG_XX
[....]
```

Mon Tue Wed Thu Fri Sat Sun latest

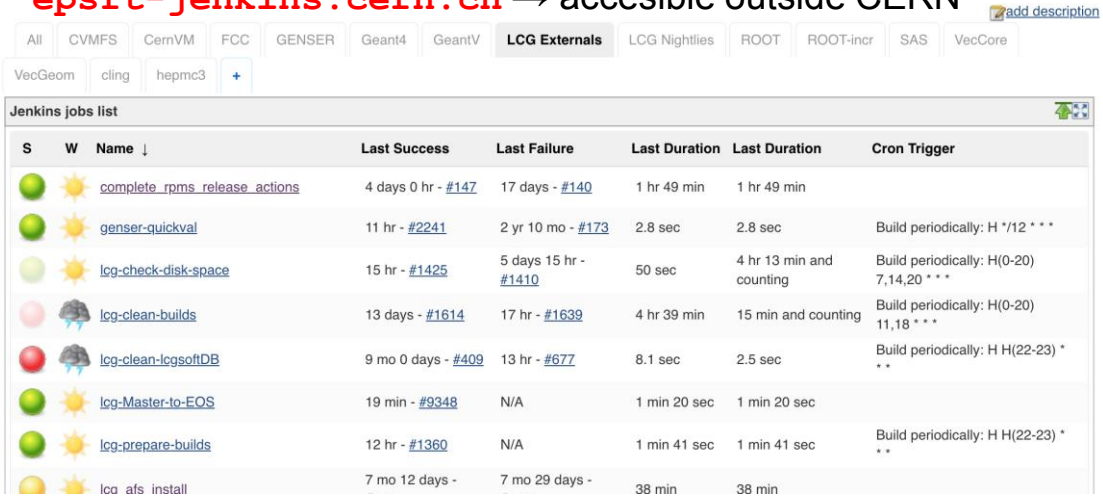
/platform/<common_prefix>/binaries AND setup.(c)sh files

Automation procedures: Jenkins

Jenkins 2.46.3 version:

- Puppet Service handled by us for all SFT projects
- VM/Centos7 with 32 CPUs and 1.8 TB of external disk (highest I/O provided by IT)
- Big server/service migration in summer 2017
- In general quite stable service BUT
  - No standard service restarts —> bottlenecks with AFS based slaves

epsft-jenkins.cern.ch → accesible outside CERN



SLAVES

- Around 500 CPUs distributed among ubuntu/mac/slc6/centos6/fedora systems
- Docker containers (Centos7 VM) available for SLC6/Centos7/Ubuntu16/Ubuntu18/Fedora
- Entering 2 Techlab ARM64 machines with the latest HW

Information for users/developers: CDASH

cdash.cern.ch → accesible outside CERN

Login All Dashboards

LCGSoft

Dashboard Calendar Previous Current Next Project

No file changed as of **Saturday, May 26 2018 - 22:00 UTC**

5 minutes ago: 2 tests failed on dev4-x86_64-slc6-gcc62-dbg
22 minutes ago: 1 warning introduced on 93-x86_64-slc6-gcc62-opt
22 minutes ago: 5 errors introduced on 93-x86_64-slc6-gcc62-opt
25 minutes ago: 1 warning introduced on 93-x86_64-slc6-gcc62-opt
42 minutes ago: 144 tests failed on experimental-x86_64-ubuntu18-gcc7-dbg

Experimental

Site	Build Name	Update	Configure		Build					
		Files	Error	Warn	Error					
lcgapp-centos7-x86-64-40.cern.ch-docker	experimental-x86_64-ubuntu18-gcc7-opt	0	0	1	86					
lcgapp-centos7-x86-64-38.cern.ch-docker	experimental-x86_64-centos7-gcc8-opt	0	0	1	25					
lcgapp-centos7-x86-64-33.cern.ch-docker	experimental-x86_64-centos7-gcc62-opt	0	0	1	0					
lcgapp-centos7-x86-64-31.cern.ch-docker	experimental-x86_64-centos7-gcc7-dbg	0	0	1	4	1	0	119	563 -460	May 27, 2018 - 09:51 UTC
lcgapp-centos7-x86-64-39.cern.ch-docker	experimental-x86_64-centos7-gcc7-opt	0	0	1	4 -9	1	0	123 +10 -178	559 +46 -55	May 27, 2018 - 09:46 UTC

LCGSoft

Dashboard Calendar Previous Current Next

Site: lcgapp-centos7-x86-64-40.cern.ch-docker
Build name: experimental-x86_64-ubuntu18-gcc7-opt
Build start time: 2018-05-27 11:03:56

URLs or Files submitted with this build

File	Size	SHA-1
Frontier_Client-2.8.19-build.log	1 Kb	1e375b16ec6d82006473b732c4901f9b6f450dd8
Qt5-5.9.2-build.log	22 Kb	a0fda25a8cf8d08405c8cf556a0ff7d21b064373
automake-1.14-build.log	571 b	46567a3743d52d4cd0dd7e3798ae364c922a3599
clang-3.9.0-install.log	409 Kb	6c490a5843aa41fb9aa3da48ede2e90c2e6f7c9c
hadoop-2.7.4-build.log	1 Mb	39877c9d0d268660d7ae9a7d4d40390193daa7d9
hdf5-1.8.18-build.log	933 Kb	c73749d348b50a41b049308d2baa6b81e5a7172b
igprof-5.9.16-build.log	12 Kb	038d72ab6a38612ac93e471f42d29af0982169e9
valgrind-3.11.0-configure.log	2 Kb	f39350eb2a06abb1fa0deae8313e22cfb8a49ea8
herwig++-2.7.1-build.log	919 Kb	b653c51be3779a0f5d7d4fc9f7635e6375b51e06

```
                (void)write(1,log_msg,ret+1)
                ^~~~~~~~~~~~~~~~~~~~~~~~~~~~
frontier_log.c:93:3: warning: ignoring return value of 'write', declared with attribute warn_unused_result [-Wunused-result]
   (void)write(log_fd,log_msg,ret+1)
   ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/usr/bin/x86_64-linux-gnu-ld: cannot find -lfrontier_client
collect2: error: ld returned 1 exit status
Makefile:289: recipe for target 'fn-req' failed
make[3]: *** [fn-req] Error 1
/usr/bin/x86_64-linux-gnu-ld: cannot find -lfrontier_client
collect2: error: ld returned 1 exit status
Makefile:299: recipe for target 'fn-fileget' failed
make[3]: *** [fn-fileget] Error 1
make[3]: Target 'dist' not remade because of errors.
```

Information for users/developers: lcginfo

`lcginfo.cern.ch` → **accesible outside CERN**

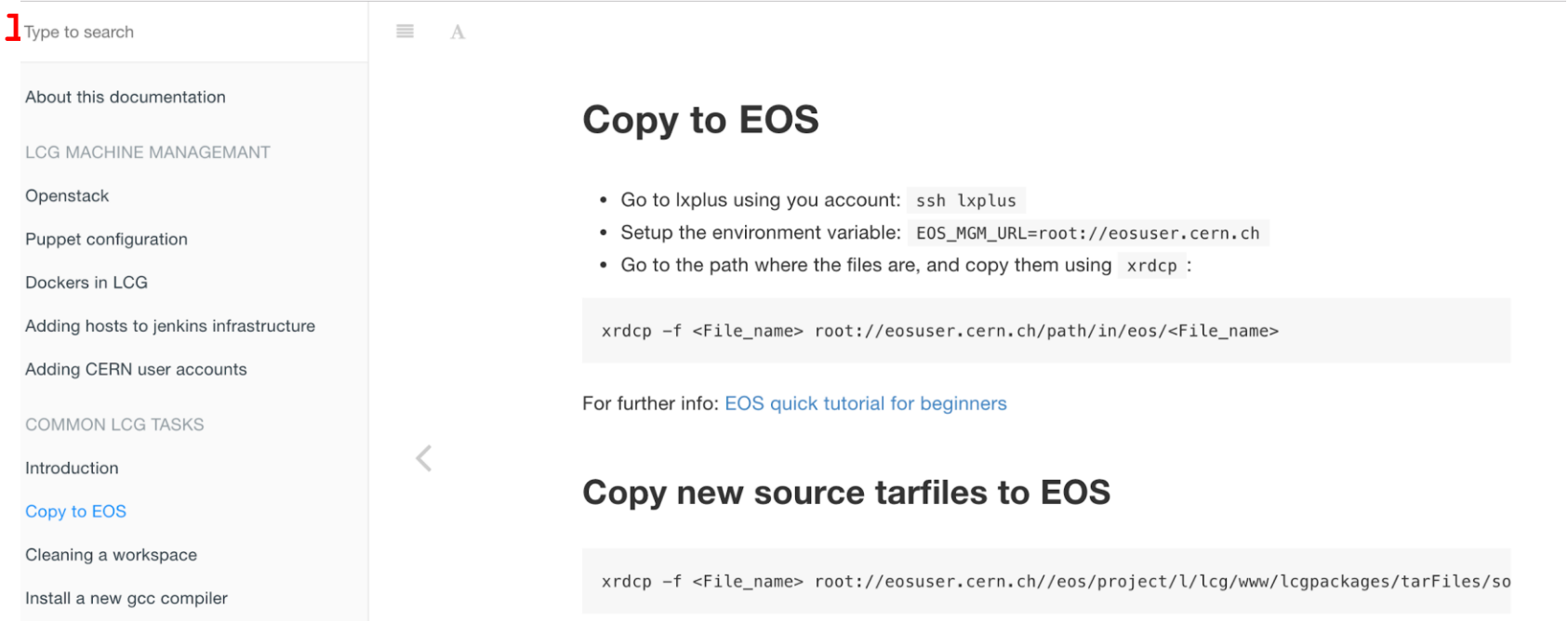
LCG Software Elements

Main / LCG Configurations Diff (93 / 92)

	LCG Configuration 93	LCG Configuration 92
IO		
COOL	3_2_0	3_1_9
CORAL	3_2_0	3_1_9
Simulation		
Geant4	10.04	10.03.p01

Documentation

Procedures information and documentation available at:



(Some) Topics to discuss at the end of the day

- **AFS common deprecation strategy**
  - Where are we at this moment?
- **Evolution of build nodes**
  - Role of HepOSlibs and its evolution
- **Releases/software distribution**
  - evolution and current status
- **arm64 strategy**
- **Future of LIM**
  - How would you like to focus it
- **Anything you want to bring to the table**