LCG Releases @ FCC Project

Current model and proposals

1st LIM Workshop - 30/05/2018
Genève, CERN

Javier Cervantes Villanueva - EP-SFT
(for the FCC Software team)

Outline

- LCG Releases and FCC
- FCC Build infrastructure
- LCG Services in use
- FCC Feedback
  - Build tool
  - Customized views
  - Workflows interaction

LCG Releases and FCC

Motivation

- LHC common software
  - We adopt the same set of tools, standards and procedures as the rest of LHC experiments
  - New developments can be integrated from/by existing projects

LCG Releases and FCC

Motivation

- LHC common software
  - We adopt the same set of tools, standards and procedures as the rest of LHC experiments
  - New developments can be integrated from/by existing projects
- Evolving our software stack continuously
  - Based on *user* needs and current state of the art
  - New packages constantly added
  - Version maintenance

LCG Releases and FCC

Motivation

- LHC common software
  - We adopt the same set of tools, standards and procedures as the rest of LHC experiments
  - New developments can be integrated from/by existing projects
- Evolving our software stack continuously
  - Based on *user* needs and current state of the art
  - New packages constantly added
  - Version maintenance
- Reduce complexity
- Speed up FCC-specific builds

LCG Releases and FCC

Motivation

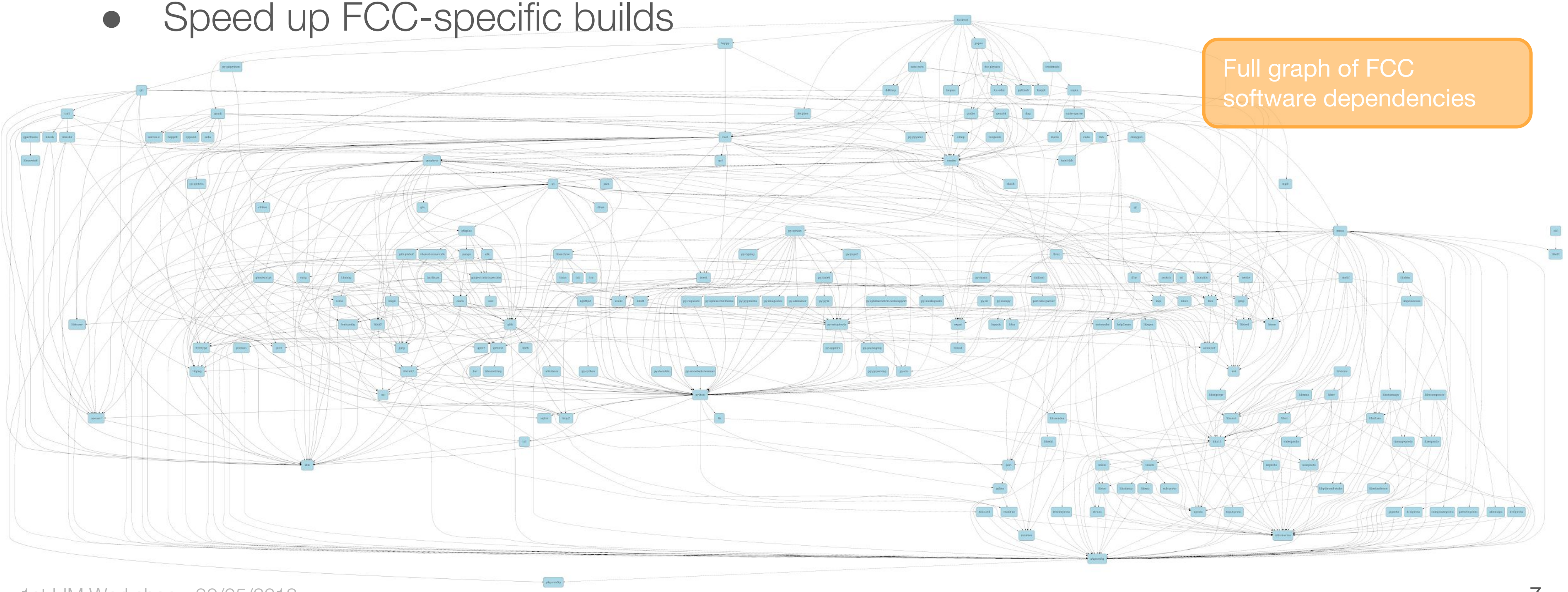
- LHC common software
  - We adopt the same set of tools, standards and procedures as the rest of LHC experiments
  - New developments can be integrated from/by existing projects
- Evolving our software stack continuously
  - Based on *user* needs and current state of the art
  - New packages constantly added
  - Version maintenance
- Reduce complexity
- Speed up FCC-specific builds

Works in general quite well

LCG Releases and FCC

Motivation

- Reduce complexity
- Speed up FCC-specific builds

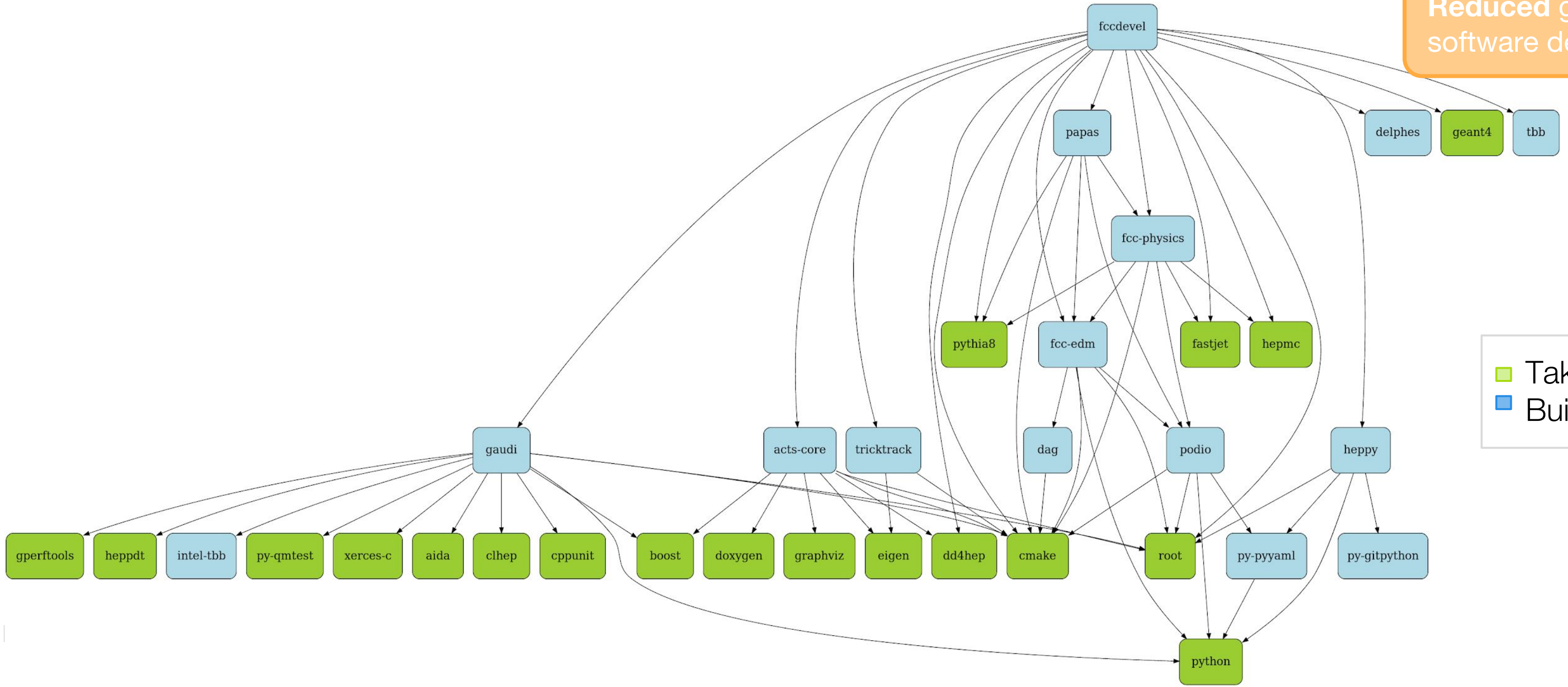


Full graph of FCC software dependencies

LCG Releases and FCC

Motivation

- Reduce complexity
- Speed up FCC-specific builds



Reduced graph of FCC software dependencies

🟩 Taken from LCG
🟦 Built by FCC

LCG Releases and FCC

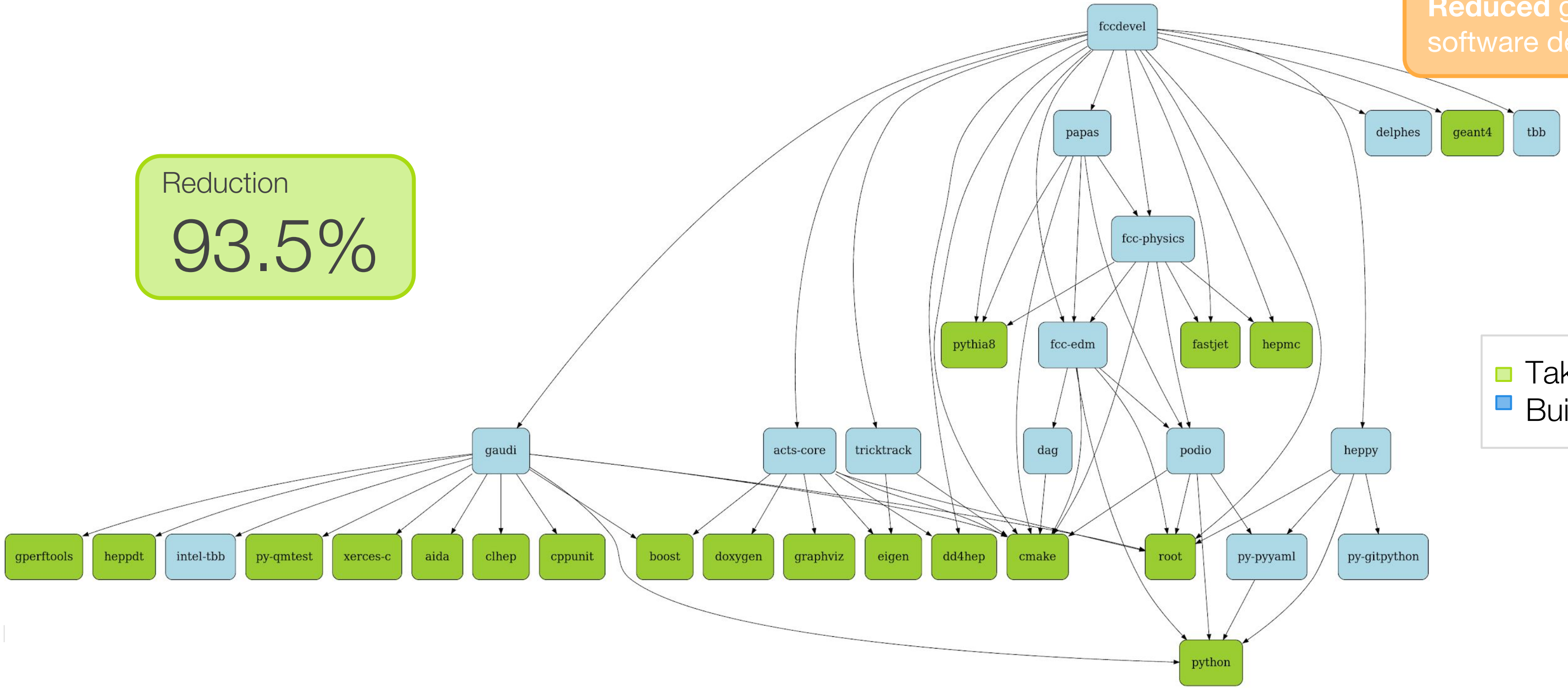
Motivation

- Reduce complexity
- Speed up FCC-specific builds

Reduced graph of FCC software dependencies

Reduction

93.5%



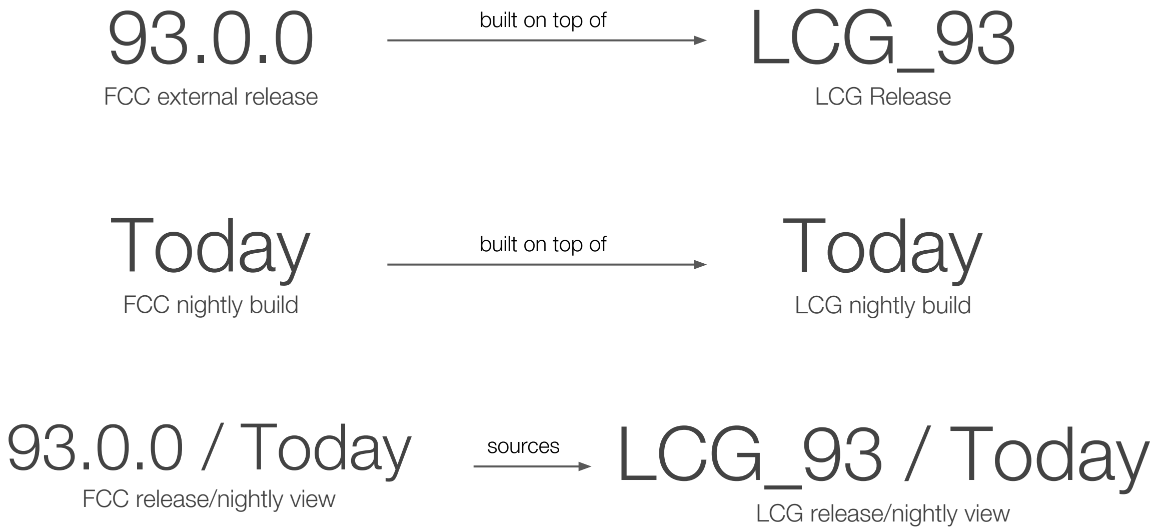
🟩 Taken from LCG
🟦 Built by FCC

LCG Releases and FCC

Where do we use it?

Every stage of the FCC Build process is based on the equivalent LCG product

- Releases
- Nightlies
- Views



93.0.0 FCC external release → built on top of → LCG_93 LCG Release

Today FCC nightly build → built on top of → Today LCG nightly build

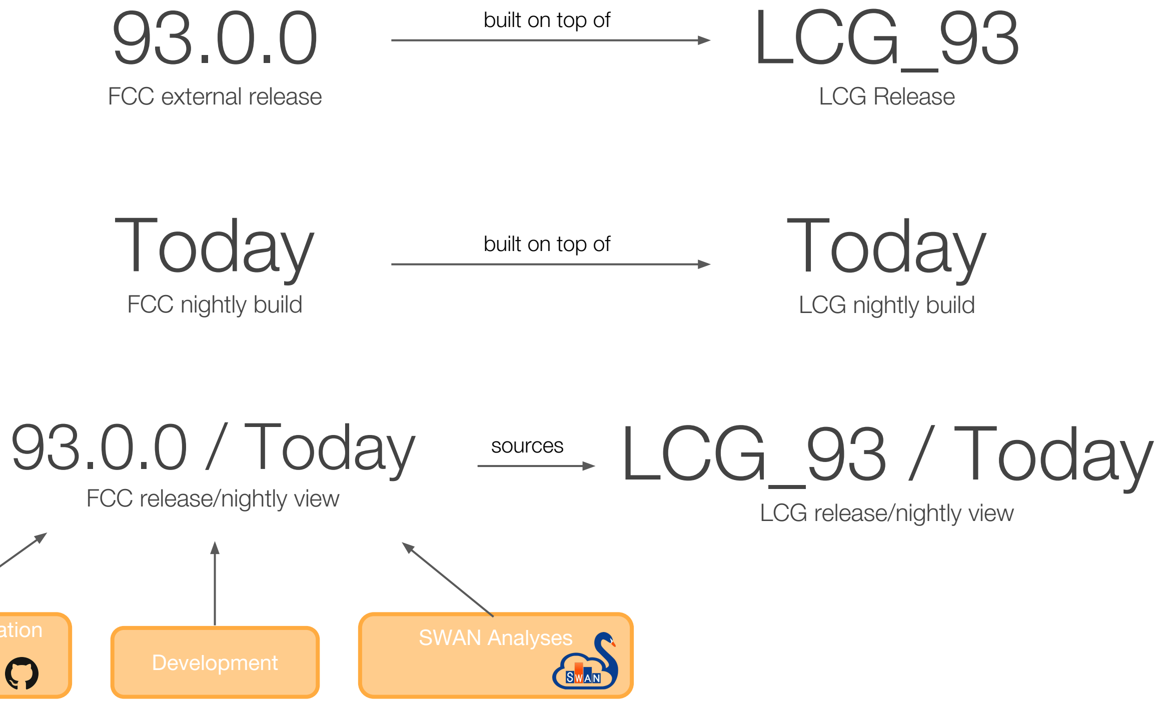
93.0.0 / Today FCC release/nightly view → sources → LCG_93 / Today LCG release/nightly view

LCG Releases and FCC

Where do we use it?

Every stage of the FCC Build process is based on the equivalent LCG product

- Releases
- Nightlies
- Views



93.0.0

FCC external release

built on top of

LCG_93

LCG Release

Today

FCC nightly build

built on top of

Today

LCG nightly build

93.0.0 / Today

FCC release/nightly view

sources

LCG_93 / Today

LCG release/nightly view

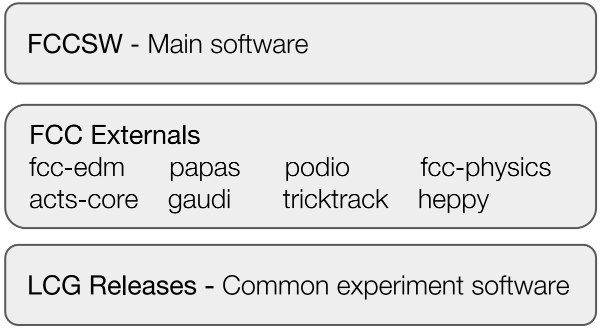
Continuous Integration

Development

SWAN Analyses

FCC Build infrastructure

- Two main deliverables:
  - FCCSW: FCC software, framework common to FCC-hh, -ee, and -eh
  - Externals: FCC-specific software dependencies
- Computing resources
  - Shared with LCG infrastructure
  - CERN Openstack virtual machines + LCG Physical nodes
    - Everything directly runs on the host env (no docker, yet)
  - CVMFS as main software repository for distribution
- Build services based on Spack
  - Package manager tool *user-environment-independent**
  - Installs new packages reusing LCG installations



FCCSW - Main software

FCC Externals
fcc-edm papas podio fcc-physics
acts-core gaudi tricktrack heppy

LCG Releases - Common experiment software

See Graeme's talk
15:40 - 16:10

FCC Build infrastructure

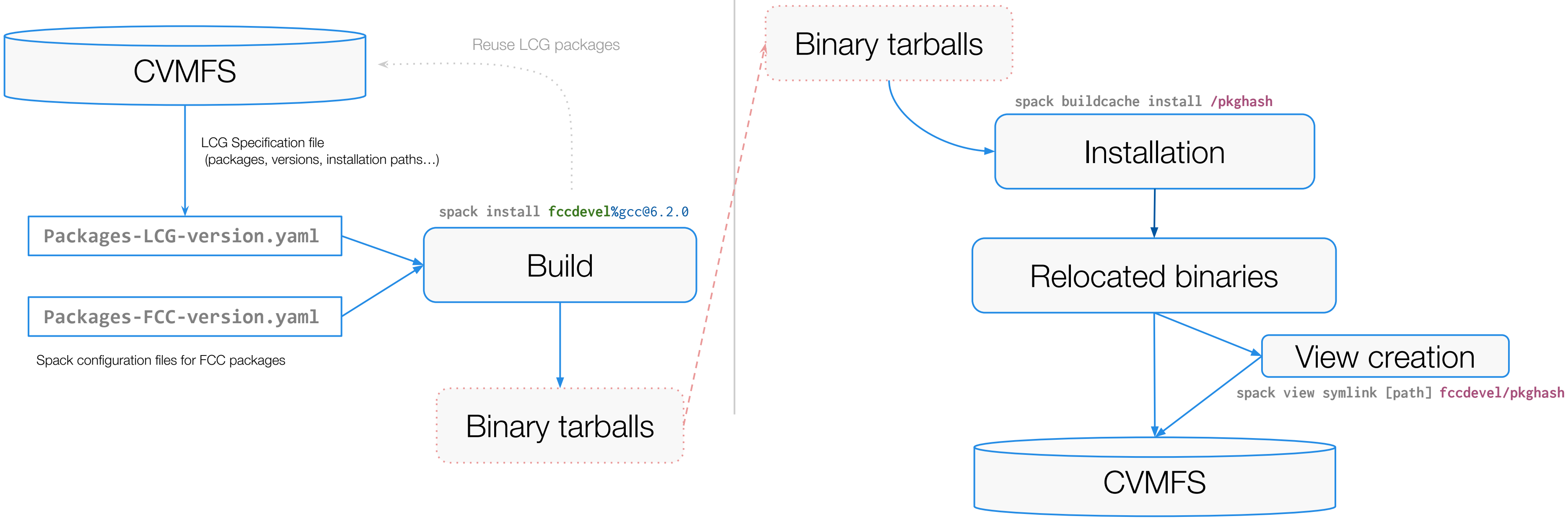
Workflow

CVMFS FCC software repository

```
/cvmfs/fcc.cern.ch/sw/
|-- nightlies/fccsw    /$weekday/$platform/(bin… include… run… init.sh…)
             /externals/$weekday/$platform/(externals packages)
|-- releases /fccsw    /$version/$platform/(bin… include… run… init.sh…)
             /externals/$version/$platform/(externals packages)
`-- views
           |-- nightlies/fccsw    /$weekday/$platform/init.sh
                        /externals/$weekday/$platform/init.sh
           `-- releases /fccsw    /$version/$platform/init.sh
                        /externals/$version/$platform/init.sh
```

Build node

CVMFS Stratum 0 Node


CVMFS
Reuse LCG packages
LCG Specification file
(packages, versions, installation paths…)
Packages-LCG-version.yaml
Packages-FCC-version.yaml
Spack configuration files for FCC packages
Build
Binary tarballs
Binary tarballs
Installation
Relocated binaries
View creation
CVMFS

```
spack install fccdevel%gcc@6.2.0
```

```
spack buildcache install /pkghash
```

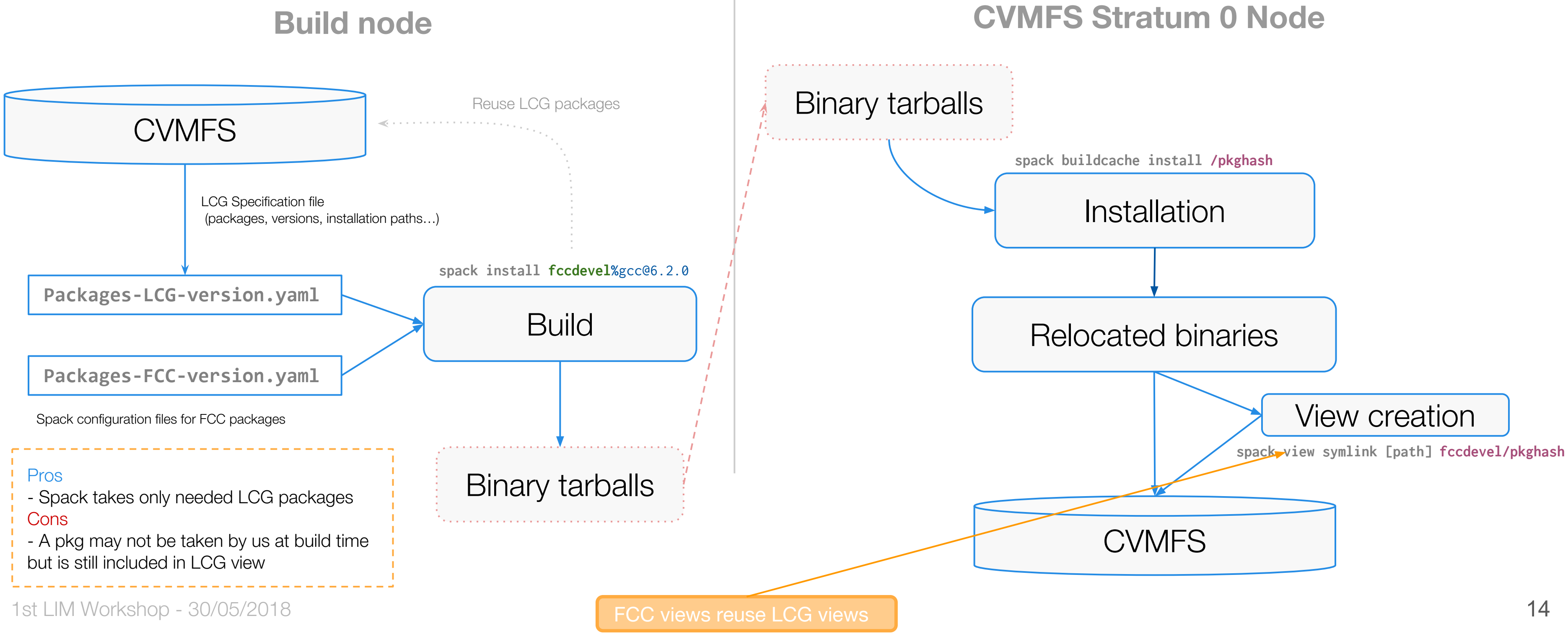
```
spack view symlink [path] fccdevel/pkghash
```

FCC Build infrastructure

Workflow

CVMFS FCC software repository

```
/cvmfs/fcc.cern.ch/sw/
|-- nightlies/fccsw    /$weekday/$platform/(bin… include… run… init.sh…)
            /externals/$weekday/$platform/(externals packages)
|-- releases /fccsw    /$version/$platform/(bin… include… run… init.sh…)
            /externals/$version/$platform/(externals packages)
`-- views
          |-- nightlies/fccsw    /$weekday/$platform/init.sh
                      /externals/$weekday/$platform/init.sh
          `-- releases /fccsw    /$version/$platform/init.sh
                      /externals/$version/$platform/init.sh
```



Build node

CVMFS

Reuse LCG packages

LCG Specification file
(packages, versions, installation paths…)

`Packages-LCG-version.yaml`

`Packages-FCC-version.yaml`

Spack configuration files for FCC packages

`spack install fccdevel%gcc@6.2.0`

Build

Binary tarballs

Pros
- Spack takes only needed LCG packages

Cons
- A pkg may not be taken by us at build time but is still included in LCG view

CVMFS Stratum 0 Node

Binary tarballs

`spack buildcache install /pkghash`

Installation

Relocated binaries

View creation

`spack view symlink [path] fccdevel/pkghash`

CVMFS

FCC views reuse LCG views

Summary of the LCG Services in use

- ✓ Releases
  - FCC Release on top
- ✓ Nightlies
  - FCC Nightlies on top
- ✓ Views
  - FCC Views source it in first place
  - Setup FCC Environment in SWAN
- ✓ lcginfo.cern.ch
  - Check content releases/nightlies
- ✓ LCG contrib compilers
  - Spack / FCC views take compilers from lcg/contrib in cvmfs

- x RPM's / Tarfiles
  - LCG Packages directly taken from CVMFS
  - Spack requires its own binary format (due to metainformation)
- x Basic Docker images
  - Not yet, possibly in a future
- x Docker containers

  (with LCG Releases)

FCC Feedback

Build tool

- Integration of two different package managers spawns tricky problems
    - Common problem to almost every pair of tools
    - No protocol / communication to extend / modify an already existing installation in a consistent manner
        - *What packages should be reused?*
        - *What packages should be reinstalled instead of taken from a LCG Release?*

FCC Feedback

Build tool

- Integration of two different package managers spawns tricky problems
  - Common problem to almost every pair of tools
  - No protocol / communication to extend / modify an already existing installation in a consistent manner
    - *What packages should be reused?*
    - *What packages should be reinstalled instead of taken from a LCG Release?*

One single tool:

- Easier integration of projects and common sw
- No need of DIY solutions

FCC Feedback

Build tool

- Integration of two different package managers spawns tricky problems
  - Common problem to almost every pair of tools
  - No protocol / communication to extend / modify an already existing installation in a consistent manner
    - *What packages should be reused?*
    - *What packages should be reinstalled instead of taken from a LCG Release?*

One single tool:

- Easier integration of projects and common sw
- No need of DIY solutions

Importance of HSF Packaging WG
15:40 - 16:10

FCC Feedback

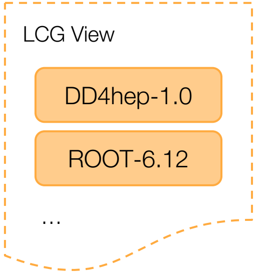
Views

- Extremely useful:
  - Abstraction from the complexity of the release
  - Source and use approach
  - Full installation ready to be used
  - Consistent environment
  - Node-independent (as long as CVMFS is installed)
- How could it be improved even further? Proposals:
  - Dynamic views: experiment-specific views based on a modular approach
  - Current views are common `setup.sh` scripts
    - *What if one could select what packages are added to the current environment?*
  - Means to find out (applicable for releases too):
    - dependency affecting options in a release: *Was ROOT built with PyROOT?*
    - tree of dependencies: *Does Geant4 depend on Qt5 in this concrete release?*

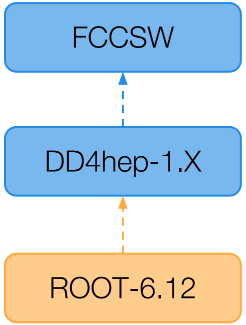
FCC Feedback

Need for customised views

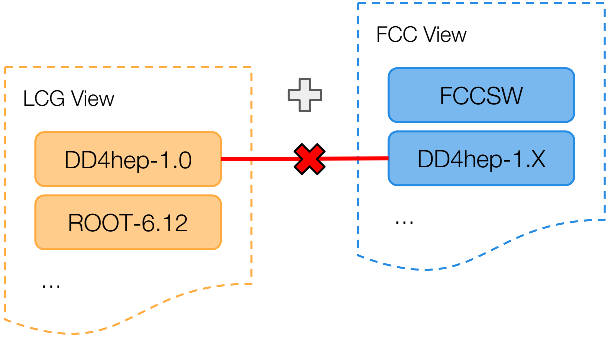
- Example of a problem:



1. LCG view available in CVMFS



2. Install our stack overwriting DD4hep (without picking it from CVMFS)



3. New FCC view sources the LCG View including DD4hep despite of building our own

FCC Feedback

Workflows interaction

- Main requirements for the LCG Nightlies: performance and efficiency
  - Execution of the nightly builds of the experiments heavily depend on LCG results
- Every software build may suffer from issues, code errors, environment problems, service downtimes...
- Given the key role of the LCG Nightlies it is important to have a systematic approach to handle every possible case

FCC Feedback

Workflows interaction

- Main requirements for the LCG Nightlies: performance and efficiency
  - Execution of the nightly builds of the experiments heavily depend on LCG results
- Every software build may suffer from issues, code errors, environment problems, service downtimes...
- Given the key role of the LCG Nightlies it is important to have a systematic approach to handle every possible case
- We just need to define the interaction of our different workflows:

Notification

Status

Action

FCC Feedback

Workflows interaction

Notification

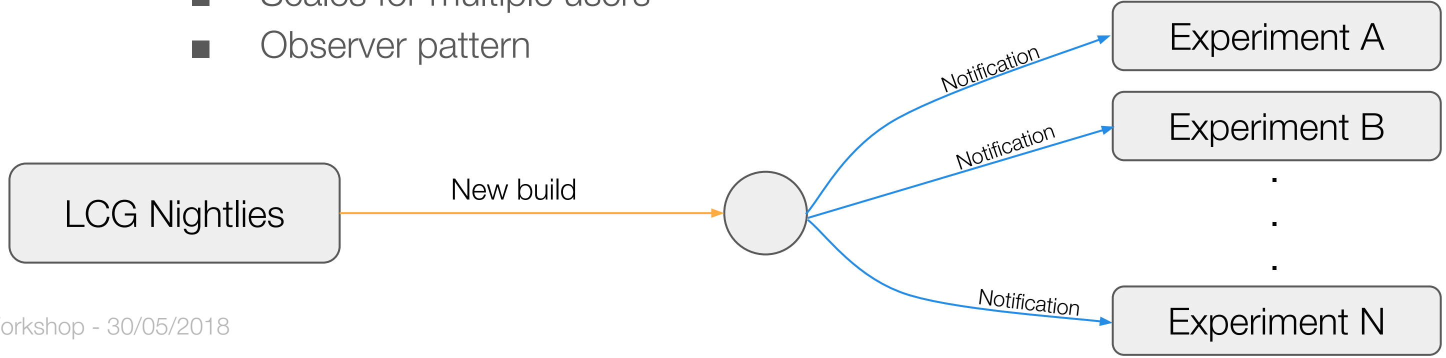
- Experiments builds wait for the LCG nightly builds until they are installed in CVMFS
  - Polling to check if the build is already present in CVMFS
    - Inefficient, waste resources

FCC Feedback

Workflows interaction

Notification

- Experiments builds wait for the LCG nightly builds until they are installed in CVMFS
  - Polling to check if the build is already present in CVMFS
    - Inefficient, waste resources
- Proposal:
  - LCG processes notify to their users when a build is ready:
    - Efficient trigger
    - Scales for multiple users
    - Observer pattern



FCC Feedback

Workflows interaction

Status

- Each build ends in a certain state
- These results should be communicated via the notification to the *users*
- Important to define possible statuses and their meanings.
- Questions:
  - SUCCESSFUL:
    - *Build finished?*
    - *Build finished without compiler errors?*
    - *Build finished without errors and got successfully installed in cvmfs?*
  - FAILURE:
    - *Can I still run my builds against this build?*
    - *Failed but it is on cvmfs?*
  - CANCELLED:
    - *Do not expect a build for today*

FCC Feedback

Workflows interaction

Actions

- Which actions can be done after a build?
  - Notify a SUCCESSFUL build → Experiments builds get trigger as soon as they receive it
  - FAILURE build
    - Try to relaunch → Experiments builds keep waiting for a notification
    - Notify FAILURE → Experiments know they should use the latest successful build
  - CANCELLED build → Experiments know they should use the latest successful build
    - After HH:MM a nightly is automatically considered CANCELLED to avoid time-outs

Conclusions

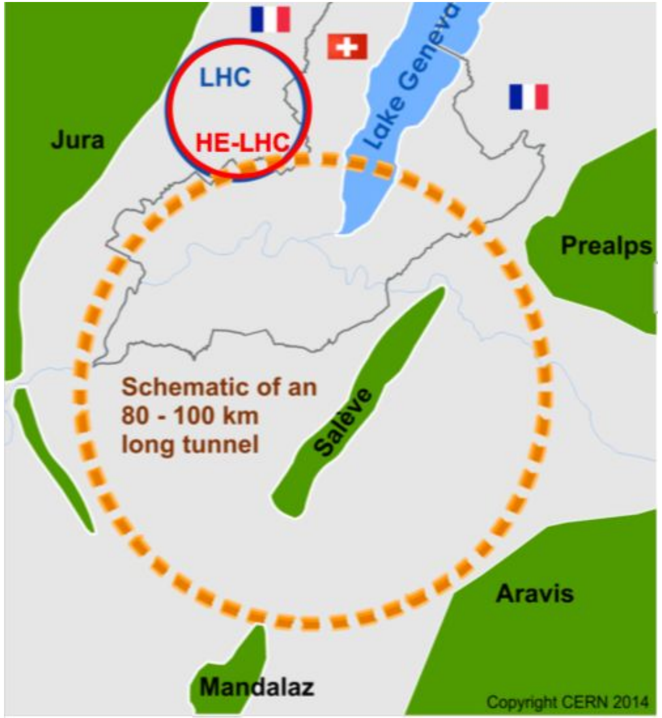
- LCG Releases are crucial
  - not only to speed up builds and reduce complexity
  - but also to establish common practices, standards and tools which allow interoperability.
- FCC Build infrastructure heavily relies on the LCG releases / nightlies / views for most of the deliverables
- Common package manager tools would reduce the amount of time needed to settle new experiment / stacks of software on top of the existing ones
- Modular and customized LCG-hosted views would be desirable
- A communication protocol between workflows would optimize the interaction with the experiments (especially when it comes to chained nightly builds)

Backup

The FCC design study - Overview

International FCC collaboration (CERN as host lab) to study :

- pp-collider (*FCC-hh*)
  - main emphasis, defining infrastructure requirements
  - ~16T magnets → 100TeV pp in 100km
  - 100 km infrastructure in Geneva area
- e^+e^- collider (*FCC-ee*)
  - as potential first step
- *HE-LHC*
  - HL-LHC with FCC-hh magnets
- p-e collider (*FCC-he*) optional


LHC
HE-LHC
Lake Geneva
Jura
Prealps
Schematic of an 80 - 100 km long tunnel
Salève
Aravis
Mandalaz
Copyright CERN 2014