Using CVMFS with Spack at the FCC study

Javier Cervantes Villanueva
EP-SFT

CernVM Users Workshop 2019

3 June 2018

What's next after the LHC?

- LHC will continue running until 2035
  - Run 3 from end LS2 until 2025
  - HL-LHC from LS3 until 2035
- Two ways to continue the research currently being conducted at LHC:

More energy

Direct observation of smaller particles

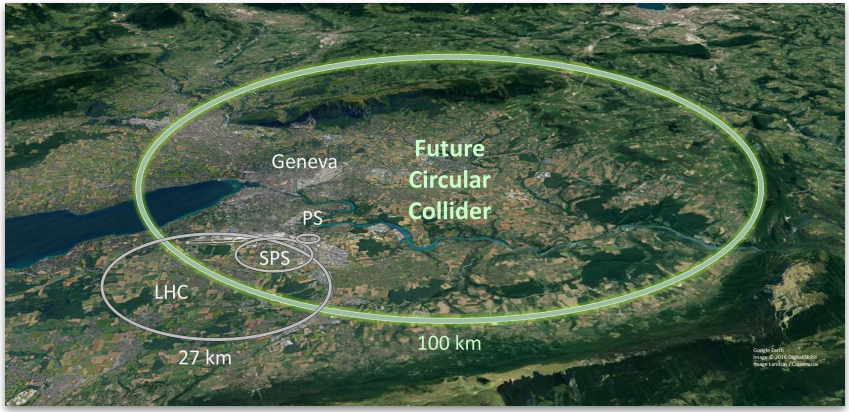
More intensity

High-precision measurements, new interactions

- The FCC program pursues both goals

FCC Study

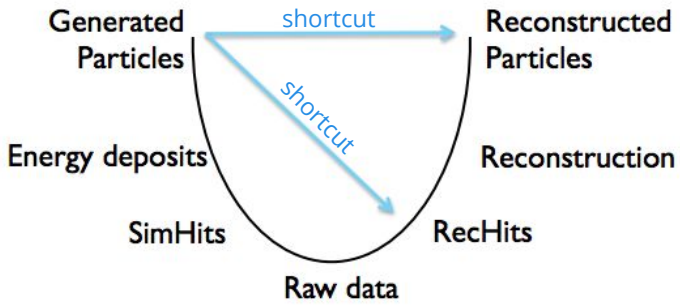
- Future Circular Collider Study (FCC)
  - Designs for a higher performance particle collider to extend the research at the LHC
- International collaboration hosted by CERN
- Proposal for a 100km circumference tunnel with different colliders:
  - FCC-hh: hadron collisions
  - FCC-ee: positron-electron collisions
  - FCC-he: proton-electron collisions
  - *HE-LHC: LHC + FCC-hh magnets



Software for the FCC

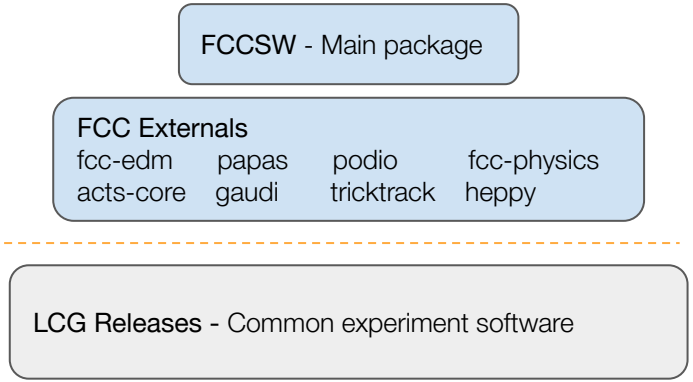
- Support experiments for all collider options: ee, eh & hh
- Support physics and multiple detector studies
- Collaborative approach:
  - Extract and adapt from the LHC experiments if possible
  - Invest into new solutions where necessary
- One software stack:

  Support all experiments from *event generation* to *physics analysis*



Build infrastructure

- Two main deliverables:
  - FCCSW: Common software framework for FCC-hh, -ee, and -eh
  - Externals: FCC-specific software dependencies
- Computing resources
  - Shared with LCG infrastructure
  - CERN Openstack virtual machines + LCG Physical nodes
  - **CVMFS, main service for software distribution**
- Build operations based on Spack
  - Package manager tool for HPC
  - **Supported by HSF*** and recently adopted by other labs
  - Manages **configuration, build** and **installation** steps
  - Installs new packages reusing LCG installations



* HEP Software Foundation (HSF)

A brief journey through the Basics of Spack

spack.io

- Spack is a package management tool
  - Does not replace CMake / Autotools / …
  - Handles different build systems
  - Manages dependencies at the level of packages
  - Support multiple versions and configurations on a variety of platforms and environments
- How to install Spack:

```
$ git clone https://github.com/spack/spack.git
$ . spack/share/spack/setup-env.sh
```

- How to install a package:

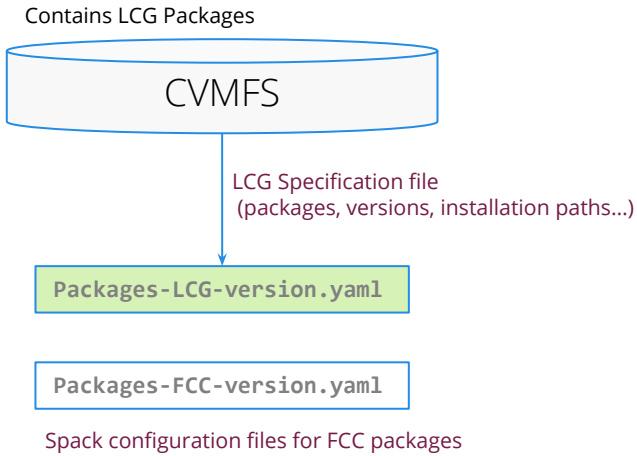
```
$ spack install hdf5
```

- Some features
  - Syntax to describe configuration
  - Full control over the combinatorial build space
  - Many configurations can coexist on the same system (new version does not break existing installations)
  - Support for binaries
  - Incremental builds

Sources - Spack: A Package Manager for HPC
Spack Documentation

Build process

Run on a Build Node



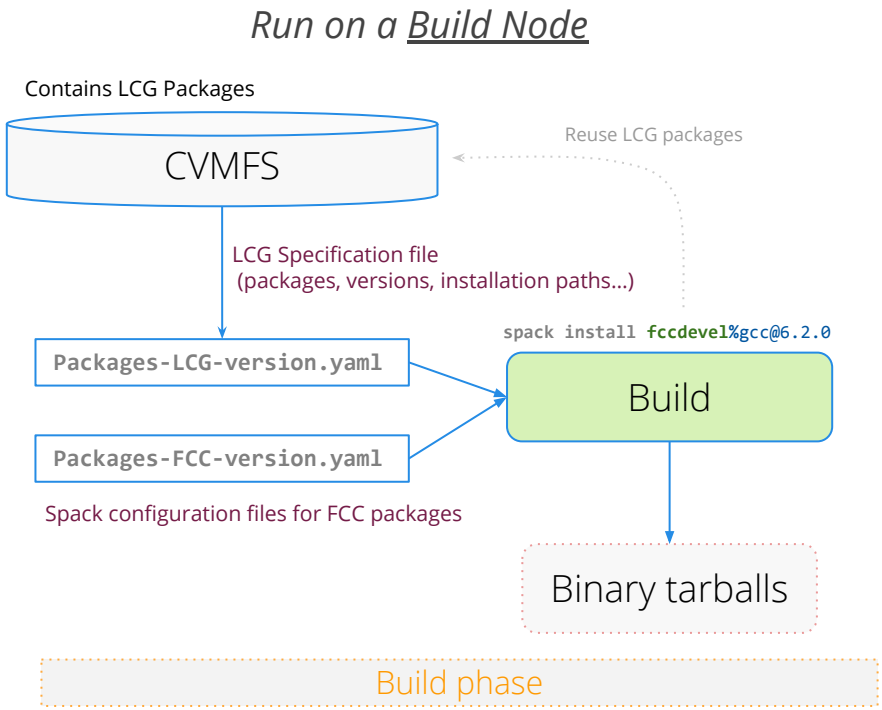
Build reusing LCG Packages

1. Let Spack know where these packages are installed in CVMFS:

```
root:
      buildable: false
      paths:{
              root@6.14.04%gcc@6.2.0
              arch=x86_64-centos7:
              /cvmfs/sft.cern.ch/lcg/releases/LCG_94/
               ROOT/6.14.04/x86_64-centos7-gcc62-opt
             }
```

Build phase

Build process



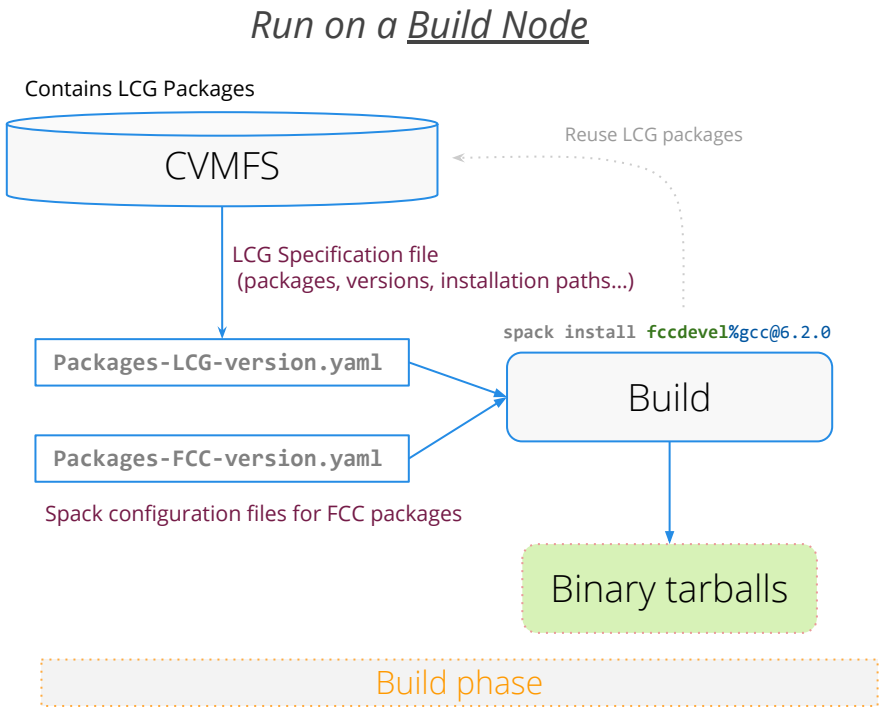
Build reusing LCG Packages

1. Let Spack know where these packages are installed in CVMFS:

```
root:
      buildable: false
      paths:{
              root@6.14.04%gcc@6.2.0
              arch=x86_64-centos7:
              /cvmfs/sft.cern.ch/lcg/releases/LCG_94/
               ROOT/6.14.04/x86_64-centos7-gcc62-opt
             }
```

2. Use the external packages during the build process

Build process



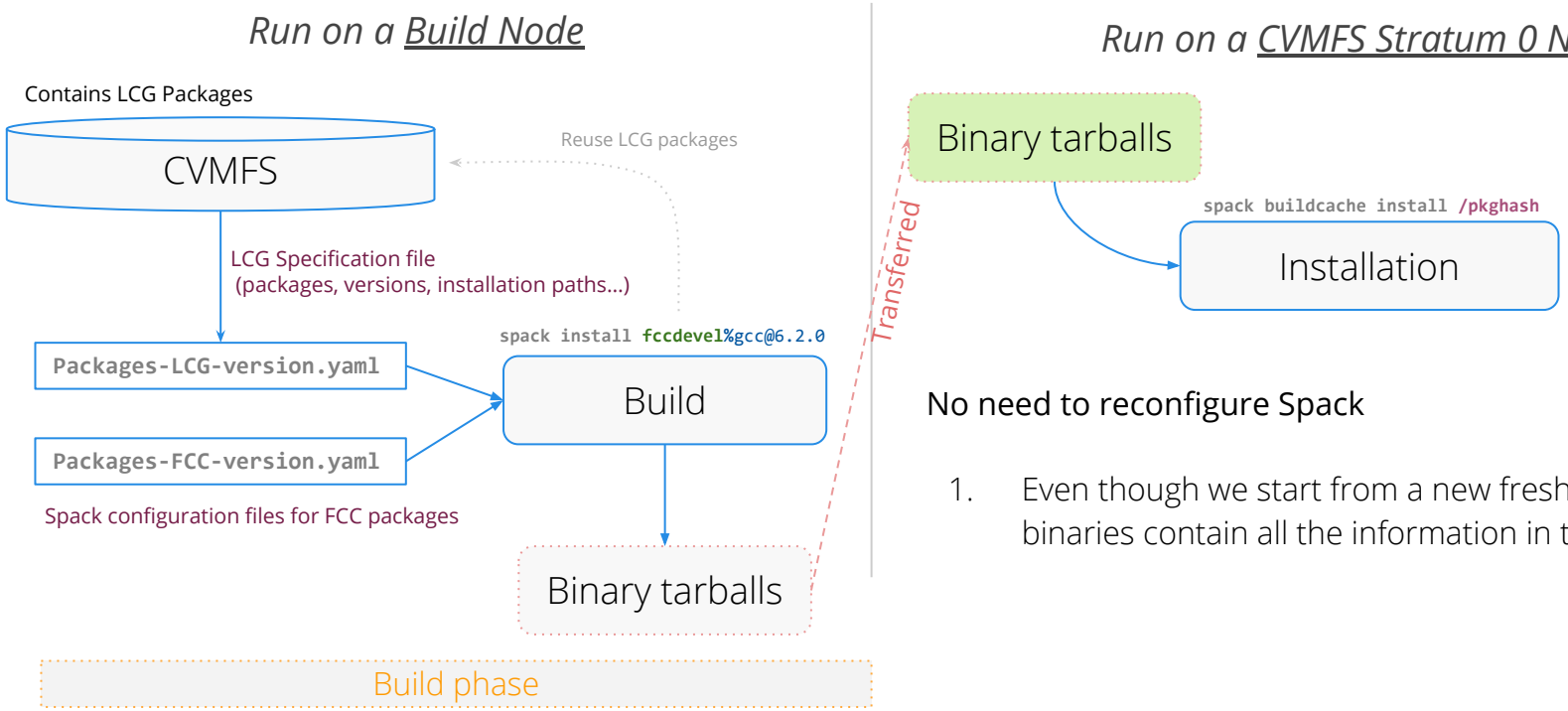
Build reusing LCG Packages

1. Let Spack know where these packages are installed in CVMFS:

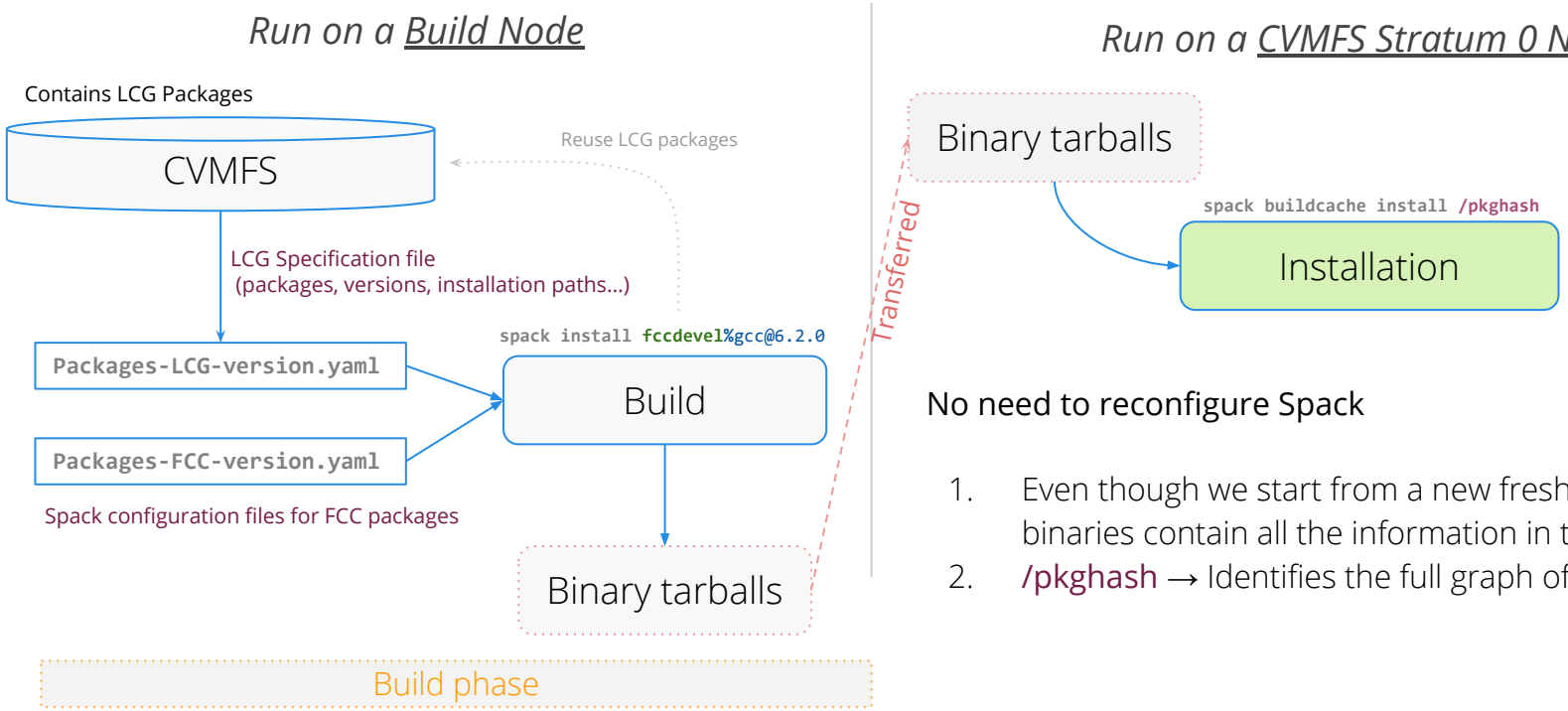
```
root:
      buildable: false
      paths:{
              root@6.14.04%gcc@6.2.0
              arch=x86_64-centos7:
              /cvmfs/sft.cern.ch/lcg/releases/LCG_94/
               ROOT/6.14.04/x86_64-centos7-gcc62-opt
             }
```

2. Use the external packages during the build process
3. Include such info as part of the binaries

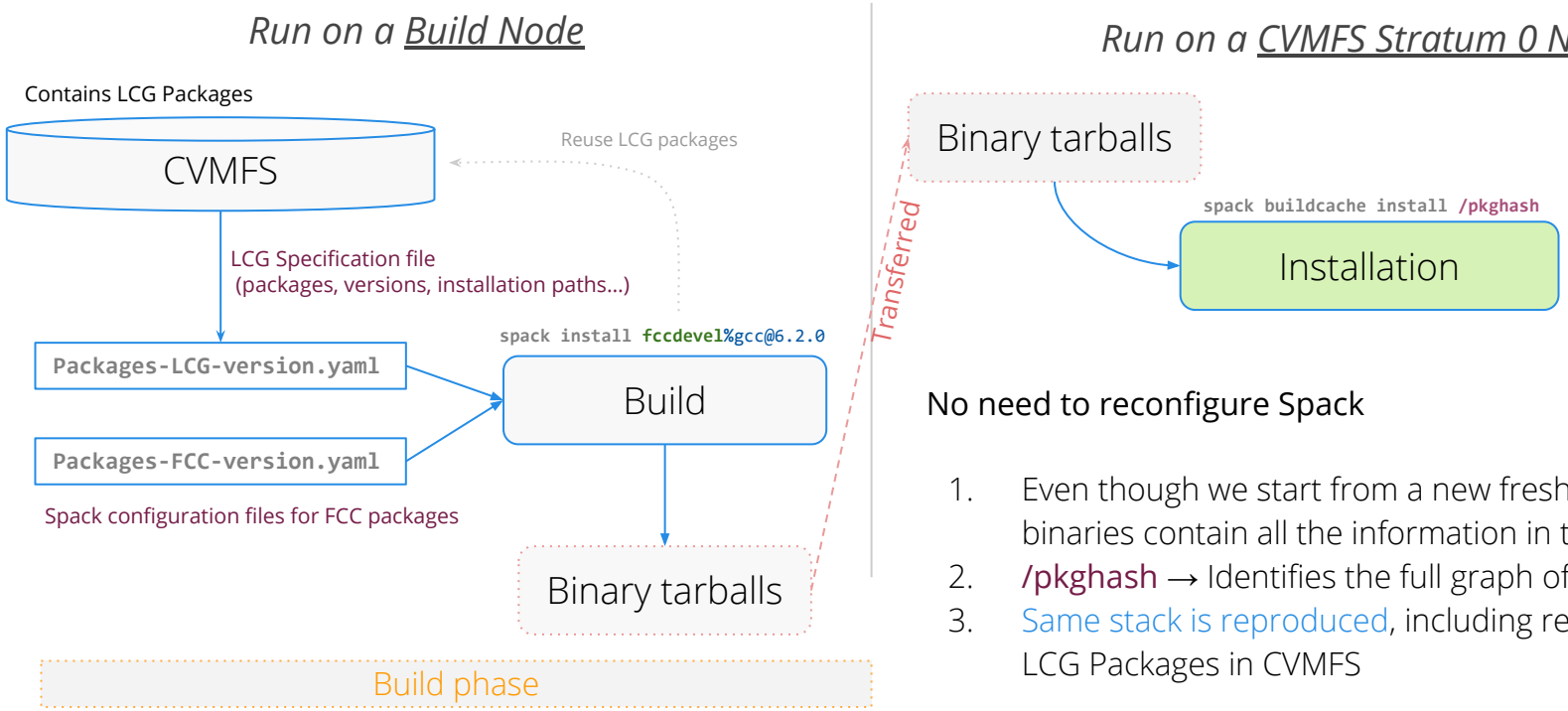
Build process



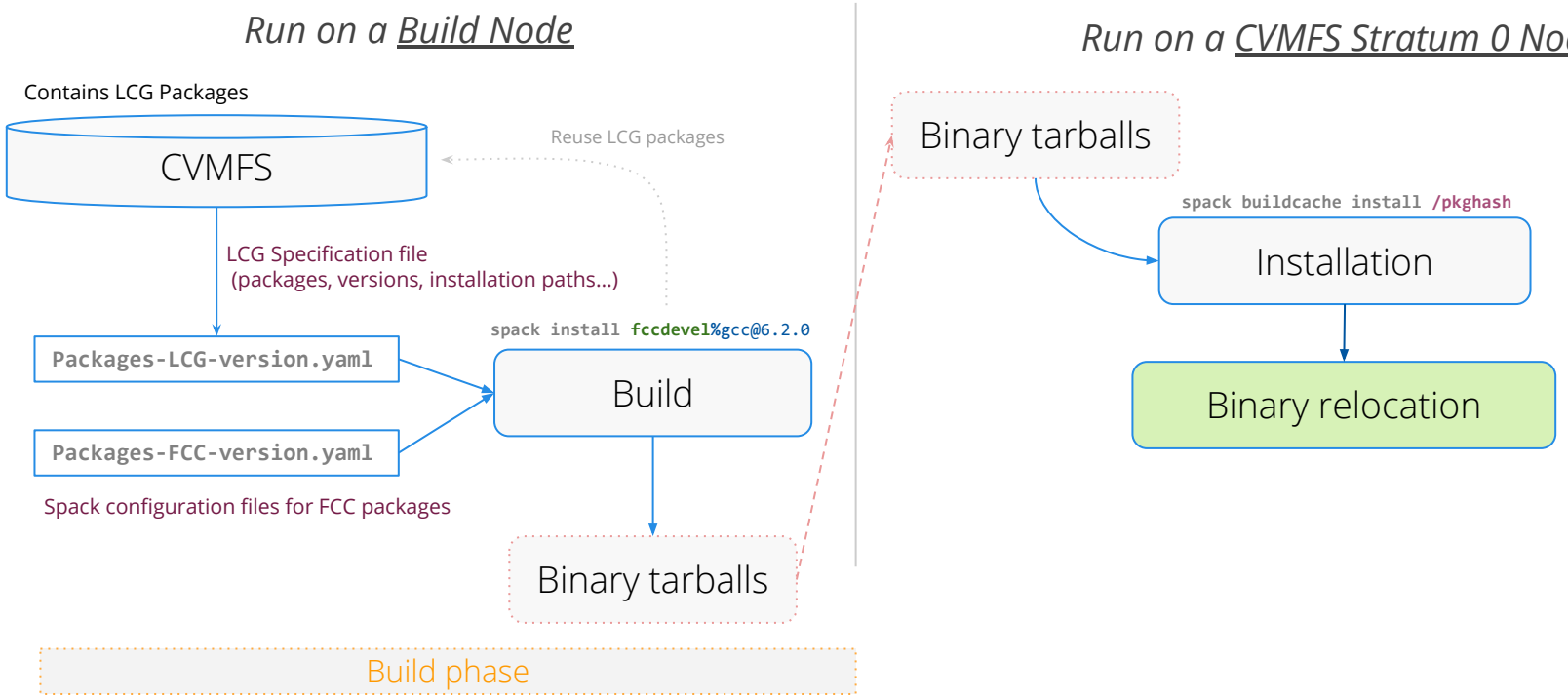
Build process



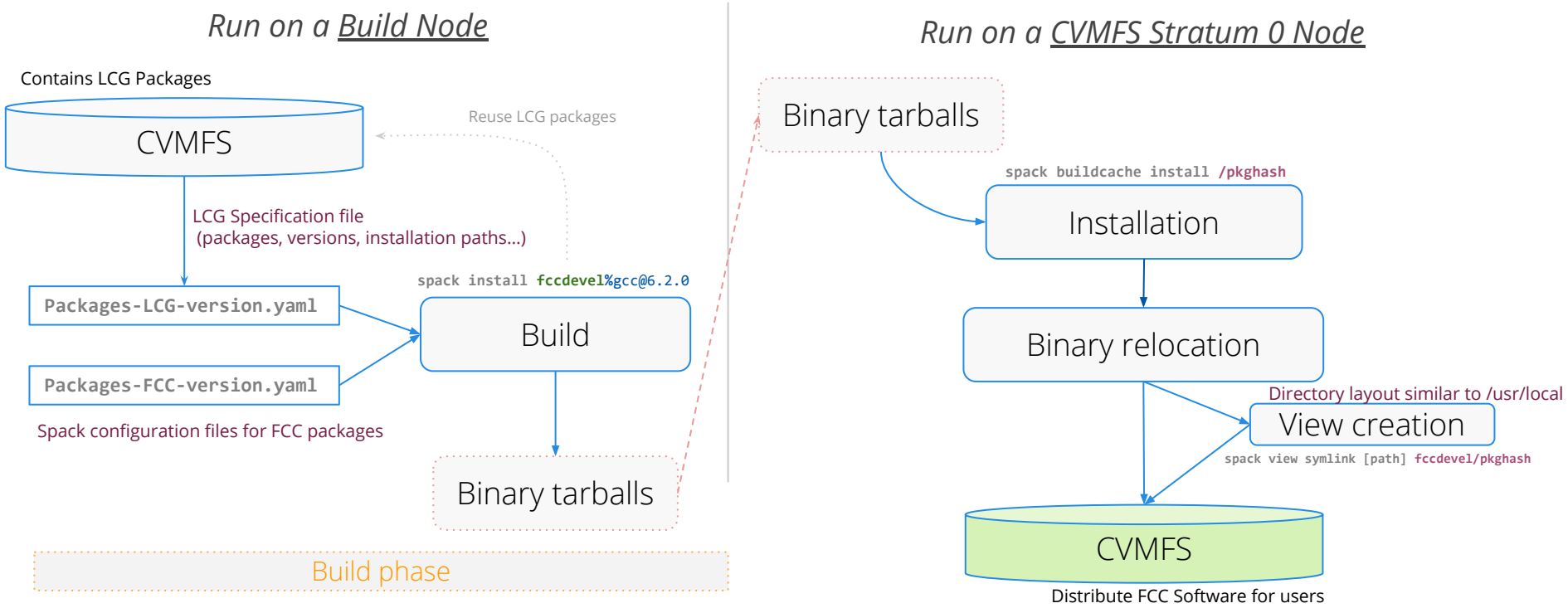
Build process



Build process

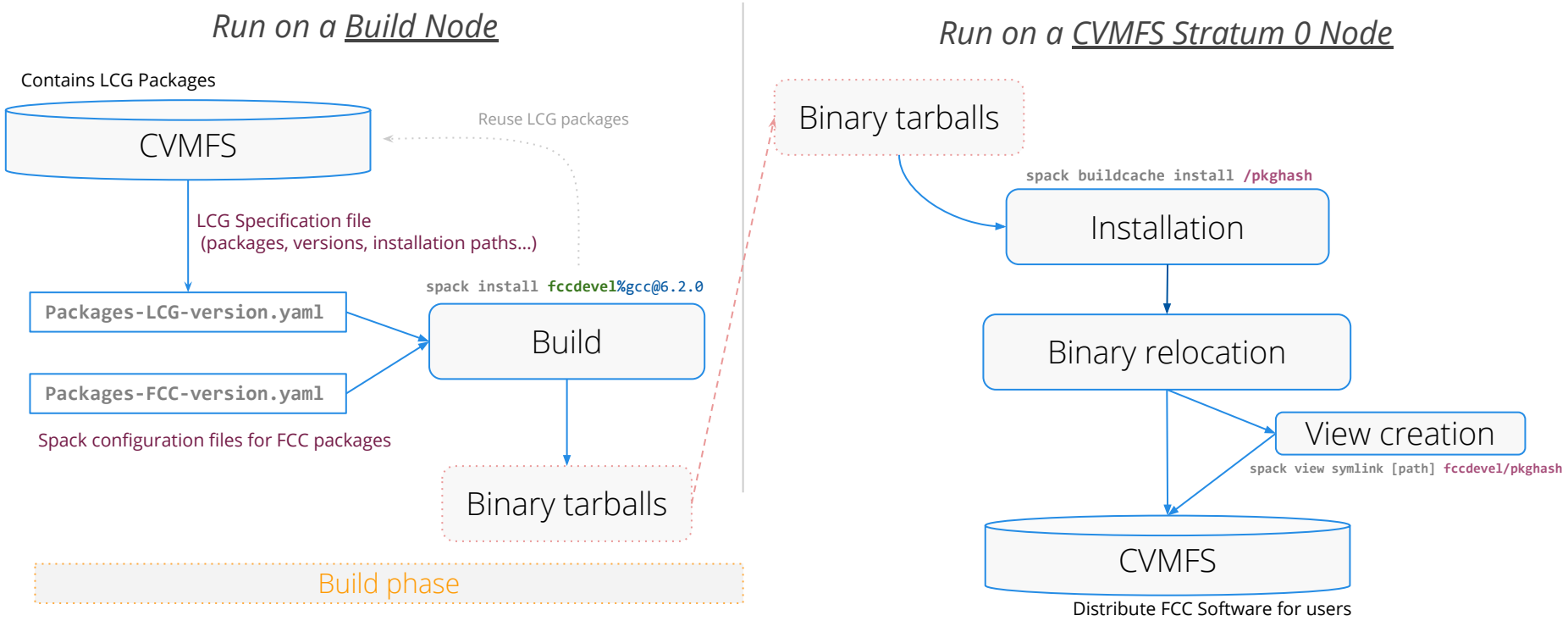


Build process



Build process

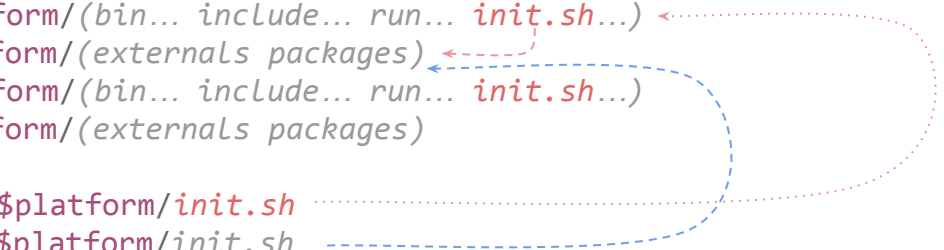
https://github.com/HEP-FCC/fcc-spi



CVMFS Organization

- Two repositories
  - `fcc.cern.ch` → **Stable releases**, exclusive addition of new releases
  - `fcc-nightlies.cern.ch` → **Nightlies and views**, weekly modifications *(remove and add)*

```
/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
```



Each version of `FCCSW` is bound to a specific version of externals:
`init.sh` sources it

Deploying binaries in CVMFS with Spack

- Spack binary files:

  Tarballs + Spack metadata → Packed into a `.spack` file

- Binaries use the Linux binary executable format (ELF)*
  - `Patchelf` is required on the Host system to relocate binaries
- Installation of packages for a OS different than the host OS has some drawbacks:
  - CVMFS Stratum 0 Node are CentOS7
  - Spack needs to be configured with the compiler for the target OS
    - Concretization process requires it
    - Spack installs Patchelf if not present, therefore it needs the compiler
  - Some cases require Spack to be running on the target OS (to get some metadata)
    - Real python home directory is directly asked to python

* Mach-O Format for MacOS

Deploying binaries in CVMFS with Spack

- These issues can be worked around
  - Indeed, the current FCC build infrastructure deploys CentOS and SLC6 releases
  - Authors are aware of them
- Suggested alternative: Docker containers*
  - Open CVMFS transaction
  - Spawn a container with an image of the target OS
  - Bind mount target CVMFS directory in the docker container
  - Run installation as usual
  - Stop container, publish transaction
- Drawback: Not valid for MacOS

* Non-tested on the FCC use-case

Summary

- CVMFS plays a key role in the FCC infrastructure
  - Stores the base software required by the FCC stack through the LCG Releases
  - Distributes the FCC-specific software
    - Releases, nightlies and views
    - Production and testing
    - User laptops, lxplus system, grid jobs
- Workflow Spack + CVMFS could be improved
  - *Option 1:* Adapt Spack → Unlikely since current design fits HPC needs
  - *Option 2:* Adopt Docker containers for software publications
- FCC infrastructure exploits a small number of CVMFS features
  - Enough to fulfill all our current requirements