Software development and deployment of the FCC Software

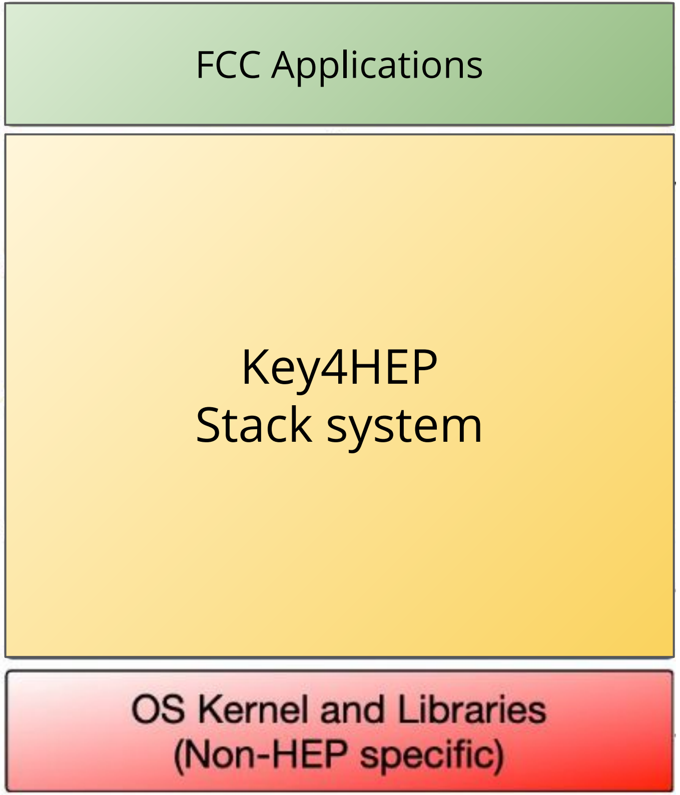
Javier Cervantes (CERN) for the FCC Software Team

FCC Week 2019, Brussels

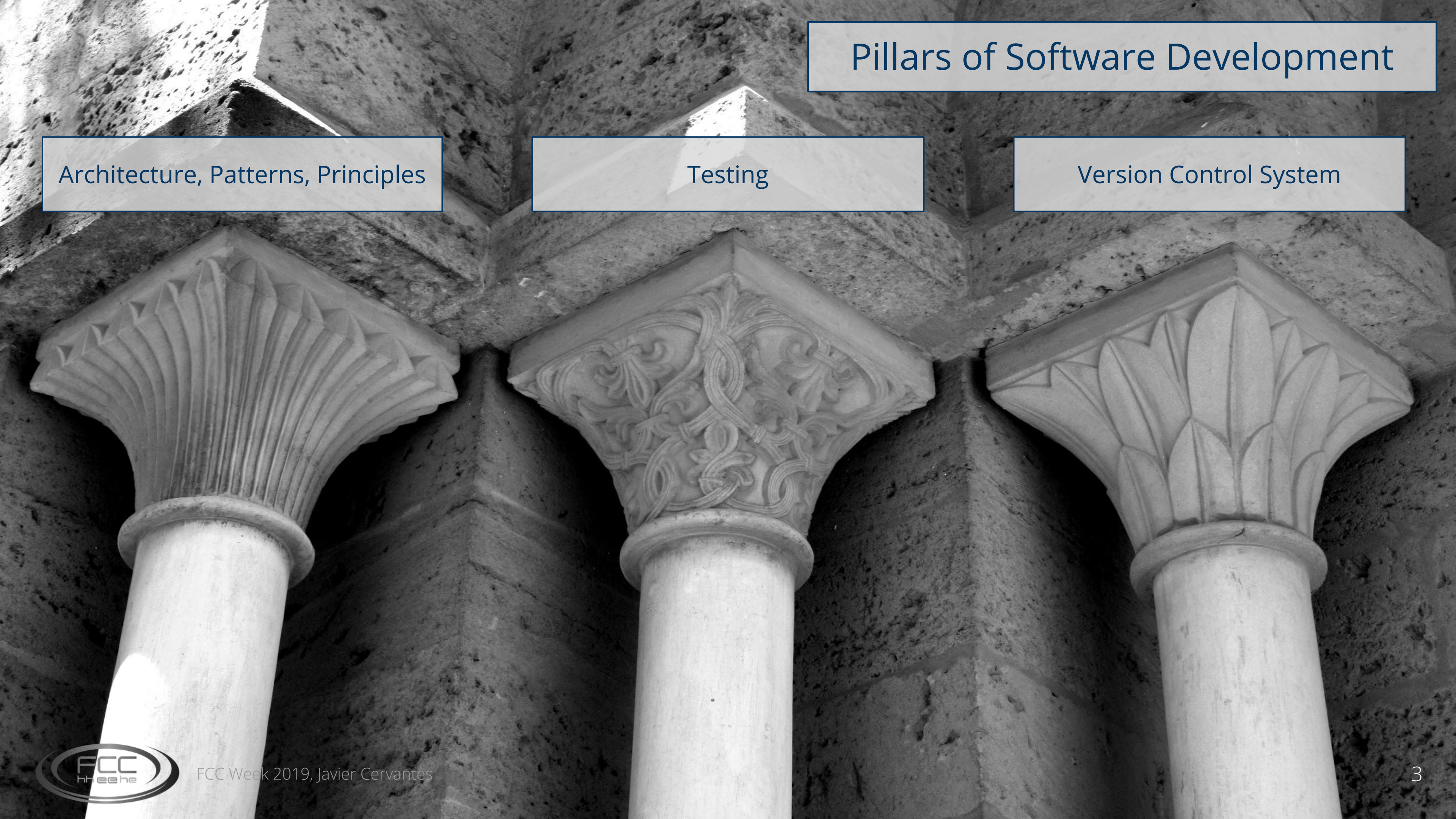
26 June 2019

FCC Software in Context

- Built on top of Key4HEP stack
  - *Common building blocks* for future experiments
- Provide FCC-specific applications
- Integrate solutions into common stack system
- Sharing methodologies and tools



Pillars of Software Development

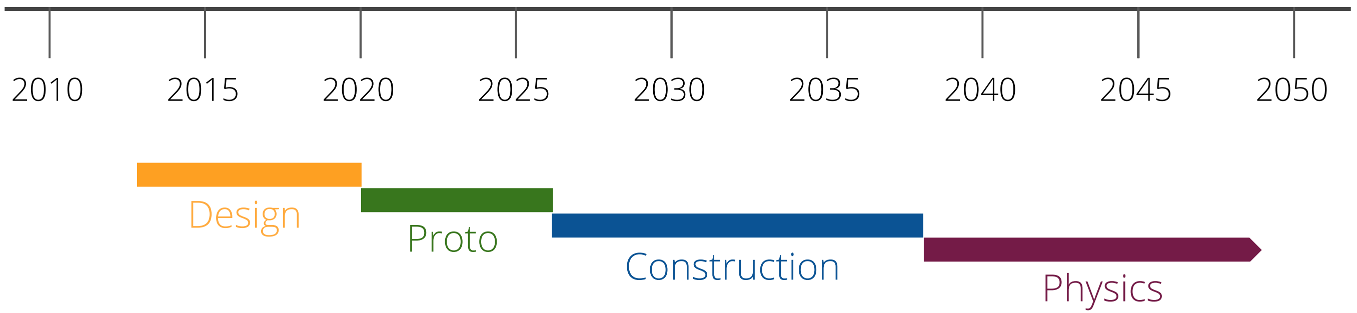


Architecture, Patterns, Principles

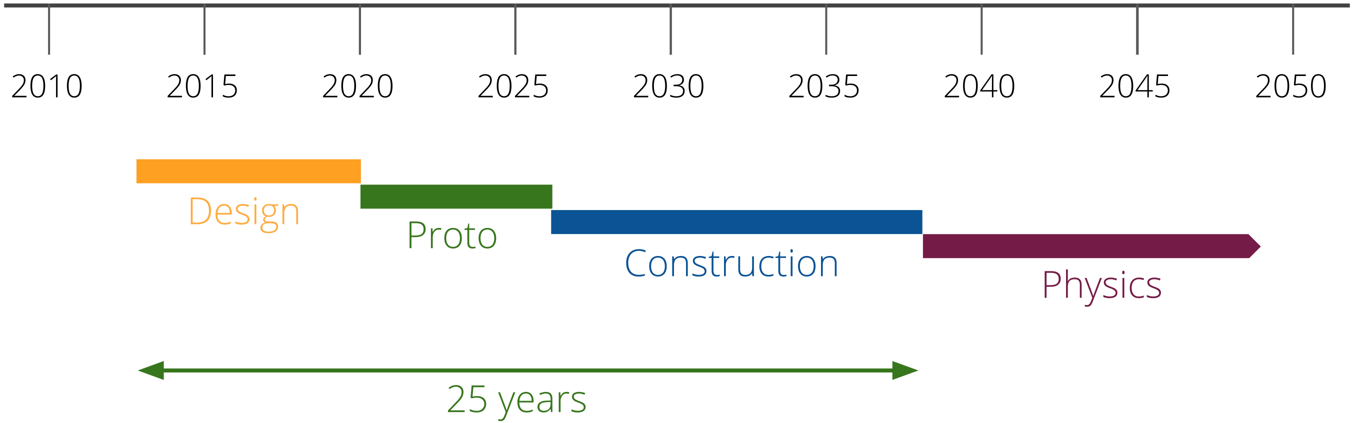
Testing

Version Control System

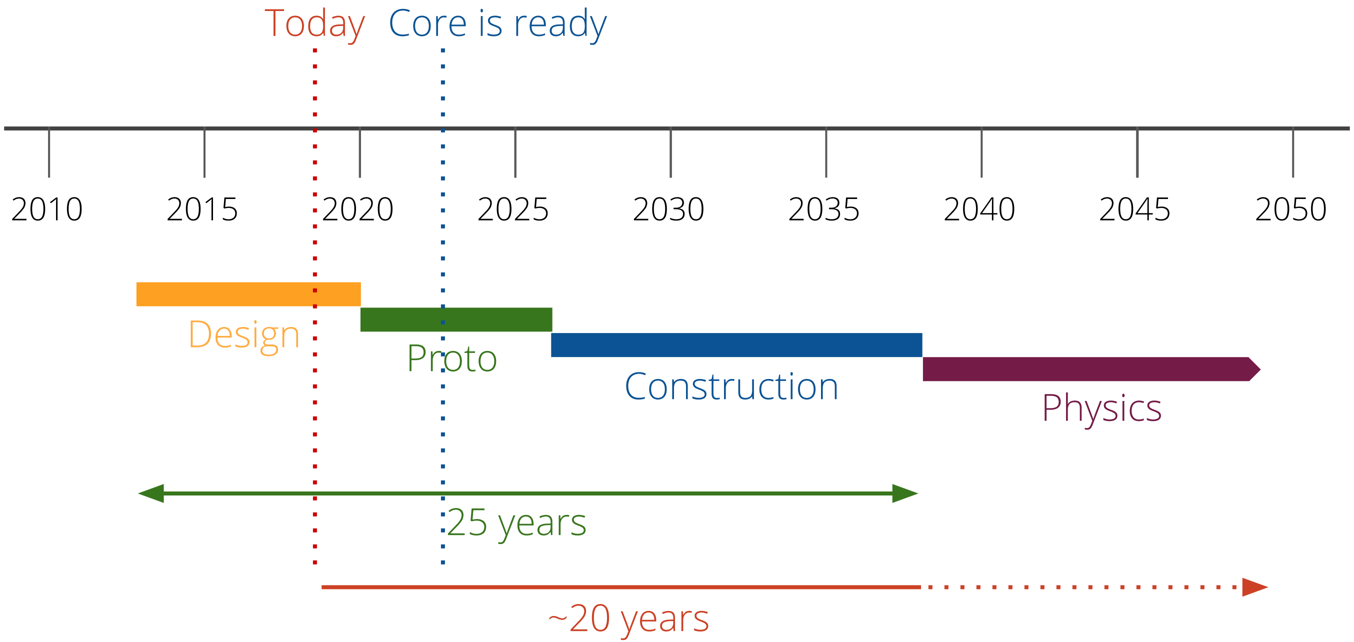
FCC Estimated Timescale



FCC Estimated Timescale



FCC Estimated Timescale



Challenges of writing SW for more than 20 years

Challenges of writing SW for more than 20 years

- Large number of people from different fields

Challenges of writing SW for more than 20 years

- Large number of people from different fields
- Wide range of skills and abilities

Challenges of writing SW for more than 20 years

- Large number of people from different fields
- Wide range of skills and abilities
- Large rotation of code authors (other projects, labs, industry...)

Challenges of writing SW for more than 20 years

- Large number of people from different fields
- Wide range of skills and abilities
- Large rotation of code authors (other projects, labs, industry...)
- General solutions for single-experiment scopes

Challenges of writing SW for more than 20 years

- Large number of people from different fields
- Wide range of skills and abilities
- Large rotation of code authors (other projects, labs, industry...)
- General solutions for single-experiment scopes

Future experiments' software needs stable, robust and efficient supporting infrastructure

Building a project for decades

Software easy to use

> *Make interfaces easy to use correctly and hard to use incorrectly*
>
> Scott Meyers - The Most Important Design Guideline?

Software easy to use

> *Make interfaces easy to use correctly and hard to use incorrectly*
>
> Scott Meyers - The Most Important Design Guideline?

Simple to install

Software easy to use

> *Make interfaces easy to use correctly and hard to use incorrectly*
>
> Scott Meyers - The Most Important Design Guideline?

Simple to install

CONDA Spack Homebrew

Software easy to use

> *Make interfaces easy to use correctly and hard to use incorrectly*
>
> Scott Meyers - The Most Important Design Guideline?

Simple to install

Easy to update / maintain

Software easy to use

> *Make interfaces easy to use correctly and hard to use incorrectly*
>
> Scott Meyers - The Most Important Design Guideline?

Simple to install

Easy to update / maintain

Tutorials, documentation, training

Software easy to use

> *Make interfaces easy to use correctly and hard to use incorrectly*
>
> Scott Meyers - The Most Important Design Guideline?

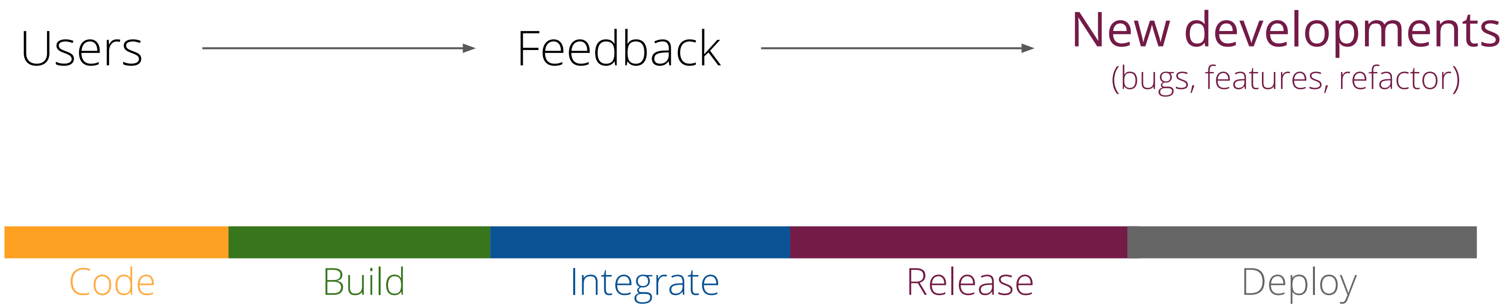
Simple to install

Easy to update / maintain

Tutorials, documentation, training

Effective and clear rules for contributions

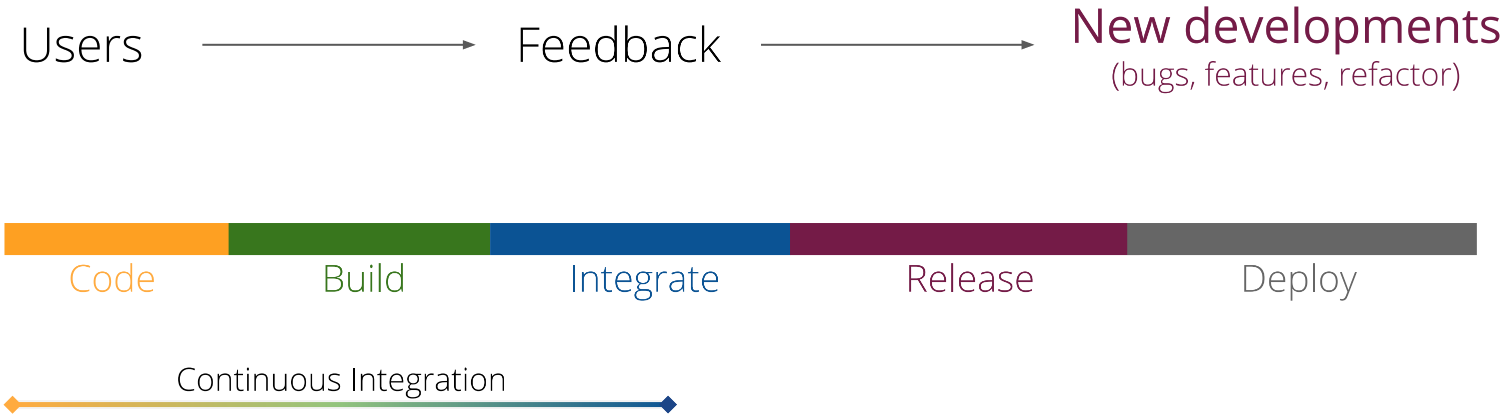
Continuous Integration and Deployment processes



Users → Feedback → New developments (bugs, features, refactor)

Code | Build | Integrate | Release | Deploy

Continuous Integration and Deployment processes



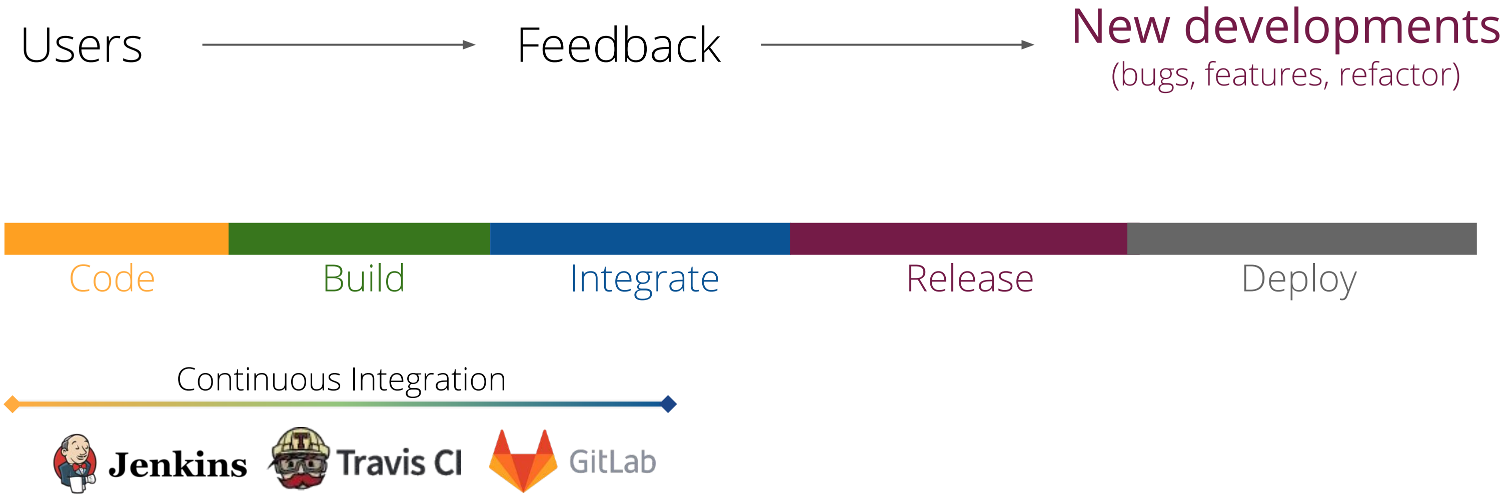
Users → Feedback → New developments

(bugs, features, refactor)

Code Build Integrate Release Deploy

Continuous Integration

Continuous Integration and Deployment processes



Users → Feedback → New developments

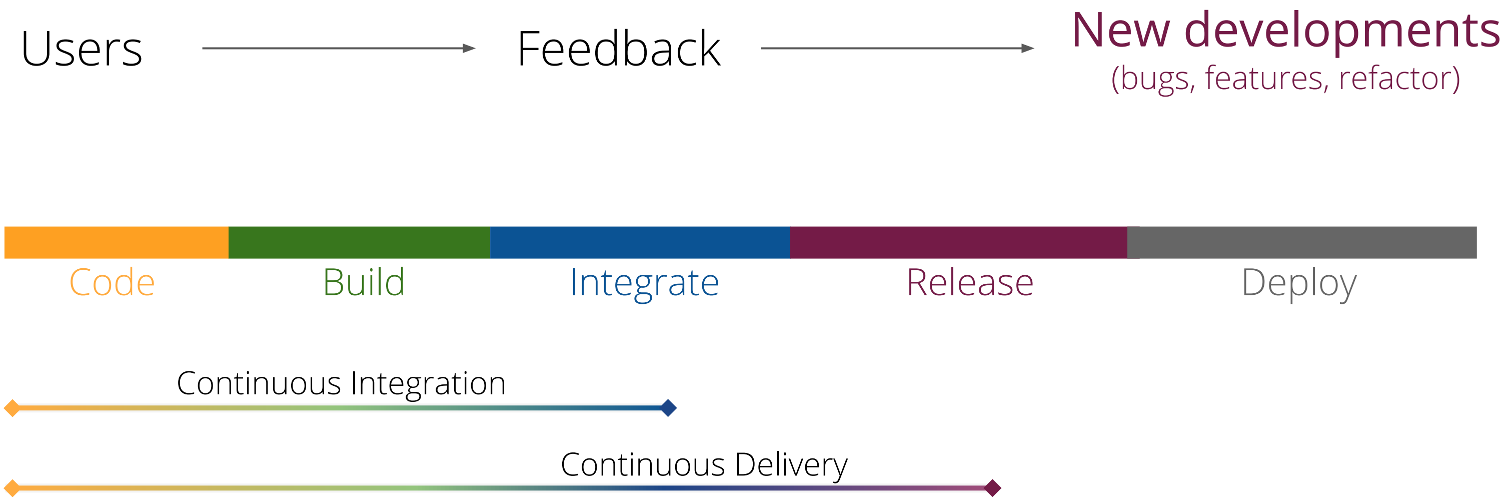
(bugs, features, refactor)

Code | Build | Integrate | Release | Deploy

Continuous Integration

Jenkins Travis CI GitLab

Continuous Integration and Deployment processes



Users → Feedback → New developments (bugs, features, refactor)

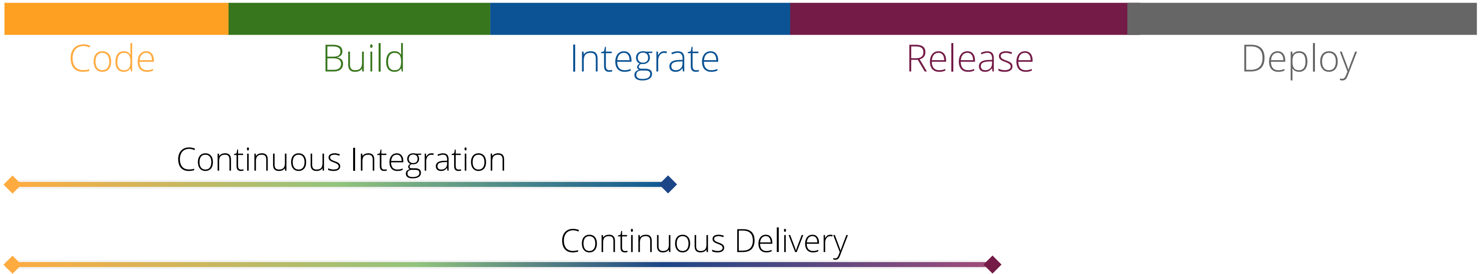
Code | Build | Integrate | Release | Deploy

Continuous Integration

Continuous Delivery

Continuous Integration and Deployment processes

Users → Feedback → New developments
(bugs, features, refactor)



Code

Build

Integrate

Release

Deploy

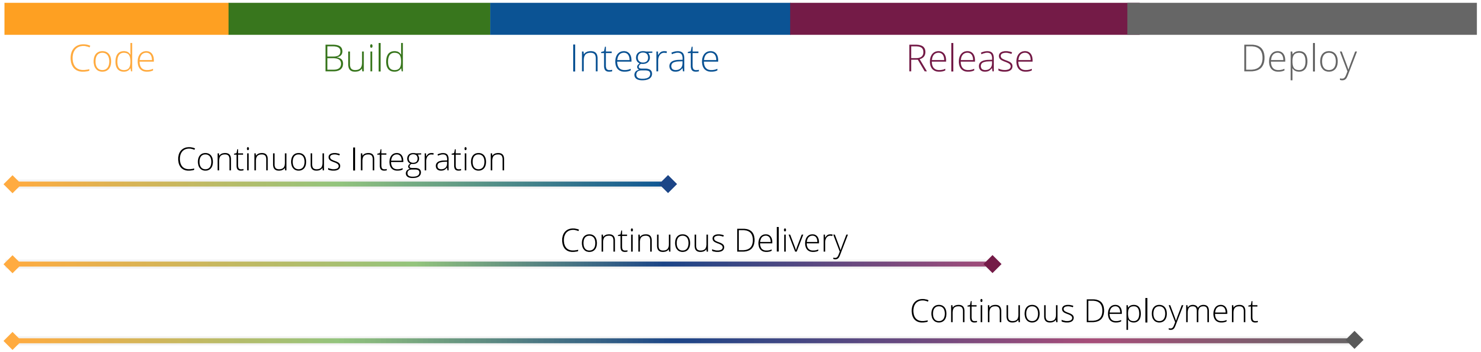
Continuous Integration

Continuous Delivery

GitHub Jenkins Travis CI GitLab docker

Continuous Integration and Deployment processes

Users → Feedback → **New developments**
(bugs, features, refactor)



Code

Build

Integrate

Release

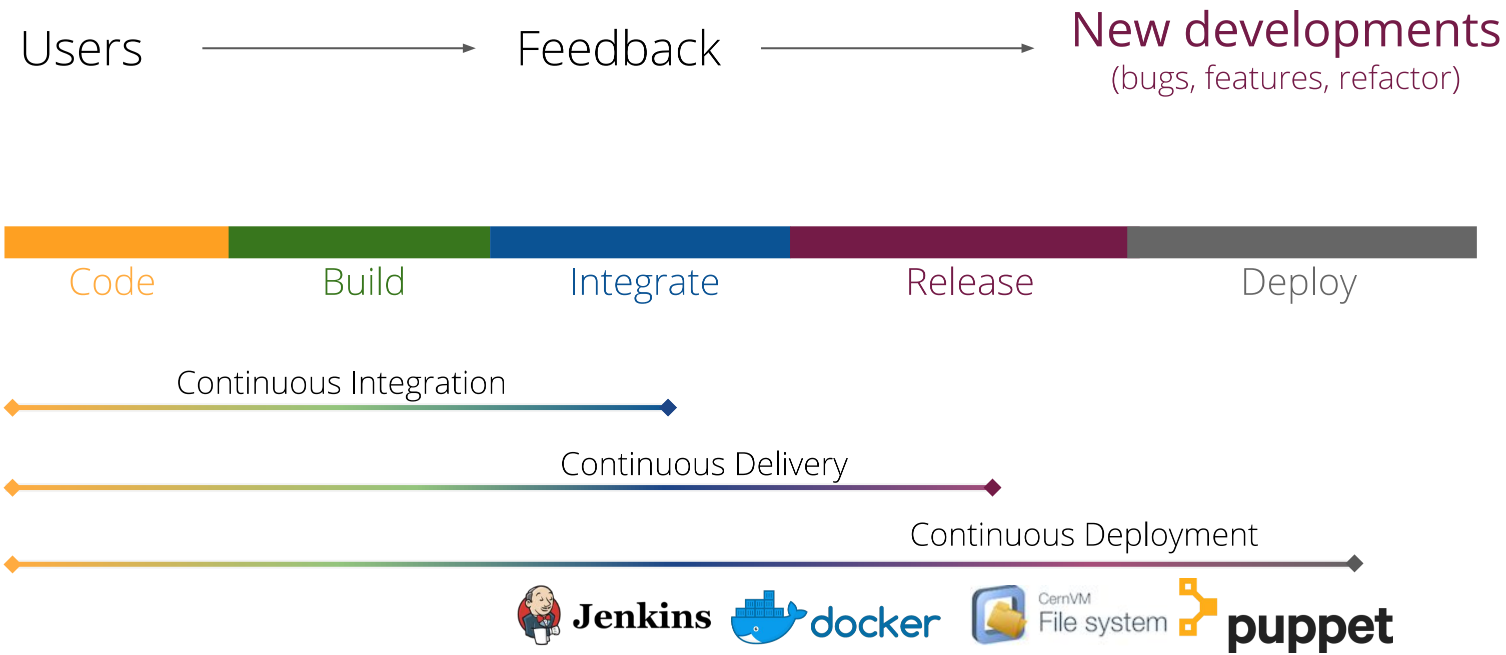
Deploy

Continuous Integration

Continuous Delivery

Continuous Deployment

Continuous Integration and Deployment processes



Users → Feedback → New developments (bugs, features, refactor)

Testing

Testing

What? Evaluate the quality of software

Testing

What?	Evaluate the quality of software
How?	Checking functional requirements Forcing errors Incrementally Assessing usability and **performance**

Testing

What?	Evaluate the quality of software
How?	Checking functional requirements Forcing errors Incrementally Assessing usability and **performance**
When?	**Always**, it is part of the process New contributions, bugs, features

Testing

What?	Evaluate the quality of software	
How?	Checking functional requirements Forcing errors Incrementally Assessing usability and **performance**	
When?	**Always**, it is part of the process New contributions, bugs, features	
Why?	Quality Faster Feedback Efficiency	Earlier Error detection User experience Longer project life

Good Practices

Good Practices

- Effective coding techniques
  - Evolution and maintainability of the code
  - Adopt new programming models and technologies without breaking API's

Good Practices

- Effective coding techniques
  - Evolution and maintainability of the code
  - Adopt new programming models and technologies without breaking API's
- Version control systems
  - Mandatory for reproducibility and deployments

Good Practices

- Effective coding techniques
  - Evolution and maintainability of the code
  - Adopt new programming models and technologies without breaking API's
- Version control systems
  - Mandatory for reproducibility and deployments
- Community engagement
  - Help users and **developers** (Forums, documentation, ...)
  - Examples, tutorials, support

Good Practices

- Effective coding techniques
  - Evolution and maintainability of the code
  - Adopt new programming models and technologies without breaking API’s
- Version control systems
  - Mandatory for reproducibility and deployments
- Community engagement
  - Help users and **developers** (Forums, documentation, ...)
  - Examples, tutorials, support
- Sharing the codebase between programmers
  - Code reviews, design discussions

GitHub GitLab

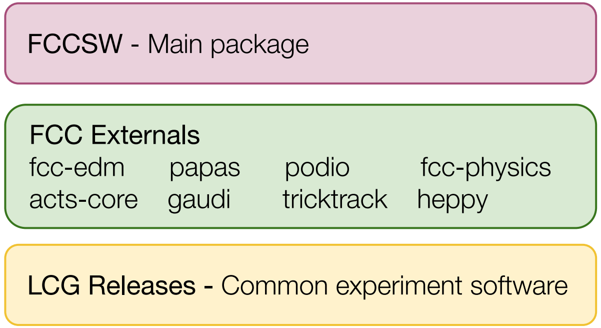
Discourse GitHub Pages

JIRA

Current status

FCC Software today

- 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
  - CVMFS as main software repository for distribution
- Build services based on Spack
  - Automated with Jenkins



FCCSW - Main package

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

LCG Releases - Common experiment software

Setup the FCC environment

```
source /cvmfs/fcc.cern.ch/sw/views/releases/externals/94.2.0/x86_64-centos7-gcc62-opt/setup.sh
```

Common conventions

Adopted community guidelines

Consistency

Interoperability

Efficiency

HEP Software Foundation Project templates

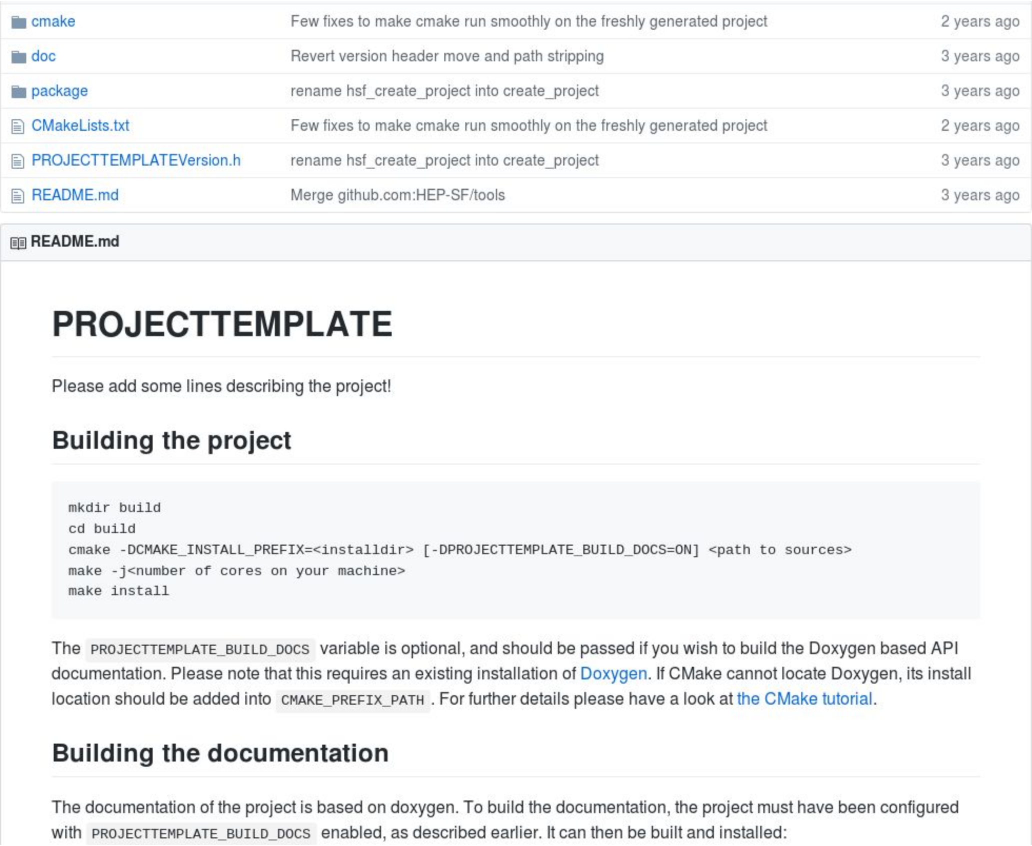
Documentation

Code structure

Similar packaging

Extensible to python modules or code techniques

[Common HSF Tools]



cmake	Few fixes to make cmake run smoothly on the freshly generated project	2 years ago
doc	Revert version header move and path stripping	3 years ago
package	rename hsf_create_project into create_project	3 years ago
CMakeLists.txt	Few fixes to make cmake run smoothly on the freshly generated project	2 years ago
PROJECTTEMPLATEVersion.h	rename hsf_create_project into create_project	3 years ago
README.md	Merge github.com:HEP-SF/tools	3 years ago

README.md

PROJECTTEMPLATE

Please add some lines describing the project!

Building the project

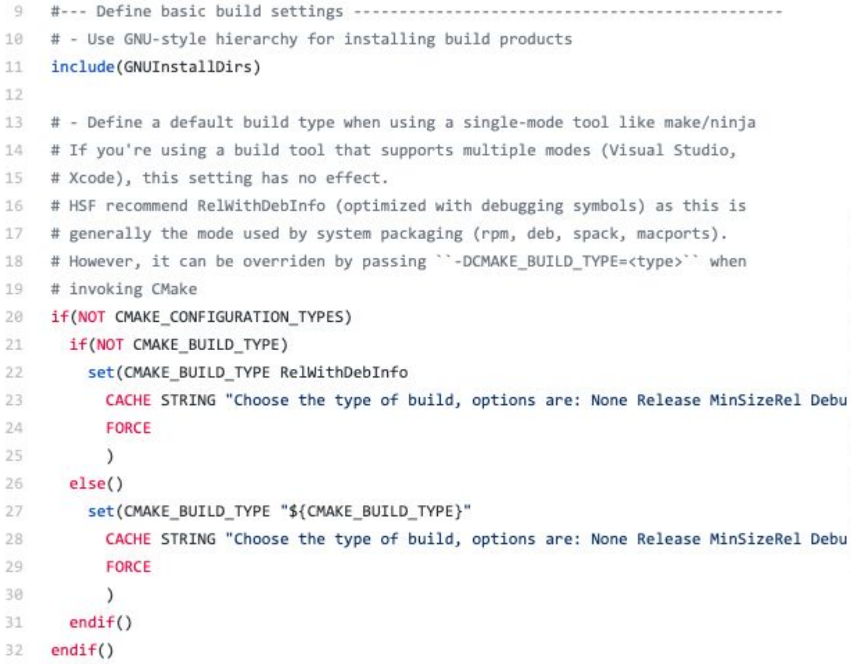
```
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=<installdir> [-DPROJECTTEMPLATE_BUILD_DOCS=ON] <path to sources>
make -j<number of cores on your machine>
make install
```

The `PROJECTTEMPLATE_BUILD_DOCS` variable is optional, and should be passed if you wish to build the Doxygen based API documentation. Please note that this requires an existing installation of Doxygen. If CMake cannot locate Doxygen, its install location should be added into `CMAKE_PREFIX_PATH`. For further details please have a look at the CMake tutorial.

Building the documentation

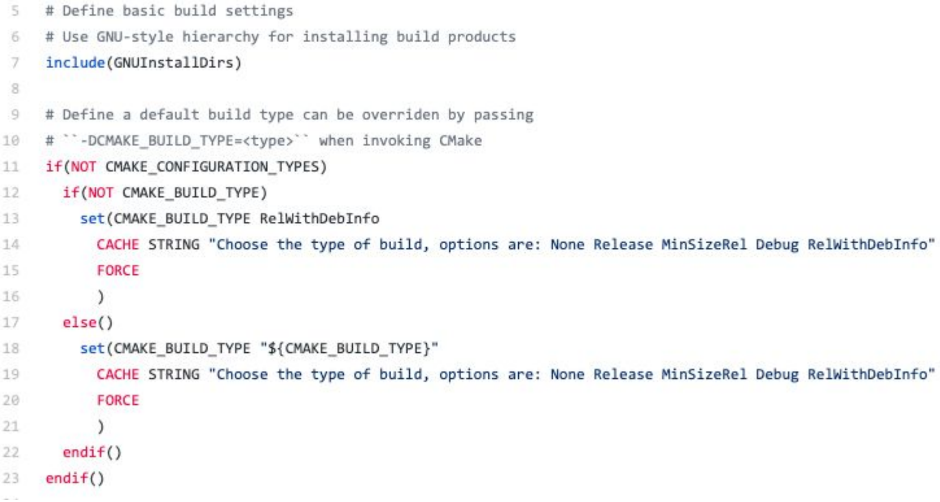
The documentation of the project is based on doxygen. To build the documentation, the project must have been configured with `PROJECTTEMPLATE_BUILD_DOCS` enabled, as described earlier. It can then be built and installed:

HSF Project templates - *Example*



```
#--- Define basic build settings -----------------------------------------------
# - Use GNU-style hierarchy for installing build products
include(GNUInstallDirs)

# - Define a default build type when using a single-mode tool like make/ninja
# If you're using a build tool that supports multiple modes (Visual Studio,
# Xcode), this setting has no effect.
# HSF recommend RelWithDebInfo (optimized with debugging symbols) as this is
# generally the mode used by system packaging (rpm, deb, spack, macports).
# However, it can be overriden by passing ``-DCMAKE_BUILD_TYPE=<type>`` when
# invoking CMake
if(NOT CMAKE_CONFIGURATION_TYPES)
  if(NOT CMAKE_BUILD_TYPE)
    set(CMAKE_BUILD_TYPE RelWithDebInfo
      CACHE STRING "Choose the type of build, options are: None Release MinSizeRel Debu
      FORCE
      )
  else()
    set(CMAKE_BUILD_TYPE "${CMAKE_BUILD_TYPE}"
      CACHE STRING "Choose the type of build, options are: None Release MinSizeRel Debu
      FORCE
      )
  endif()
endif()
```



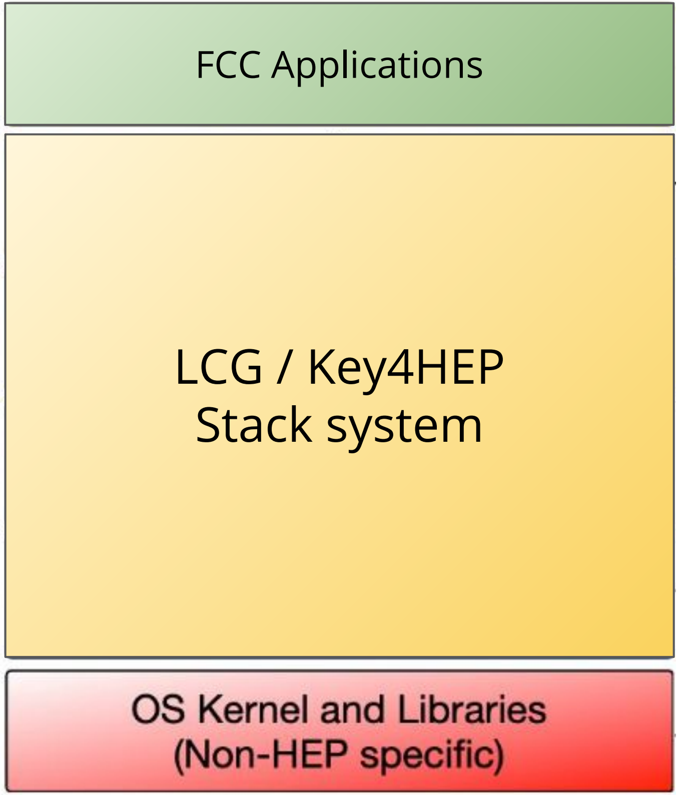
```
# Define basic build settings
# Use GNU-style hierarchy for installing build products
include(GNUInstallDirs)

# Define a default build type can be overriden by passing
# ``-DCMAKE_BUILD_TYPE=<type>`` when invoking CMake
if(NOT CMAKE_CONFIGURATION_TYPES)
  if(NOT CMAKE_BUILD_TYPE)
    set(CMAKE_BUILD_TYPE RelWithDebInfo
      CACHE STRING "Choose the type of build, options are: None Release MinSizeRel Debug RelWithDebInfo"
      FORCE
      )
  else()
    set(CMAKE_BUILD_TYPE "${CMAKE_BUILD_TYPE}"
      CACHE STRING "Choose the type of build, options are: None Release MinSizeRel Debug RelWithDebInfo"
      FORCE
      )
  endif()
endif()
```

CMake Project Template

FCC-EDM Package

Community-oriented mindset



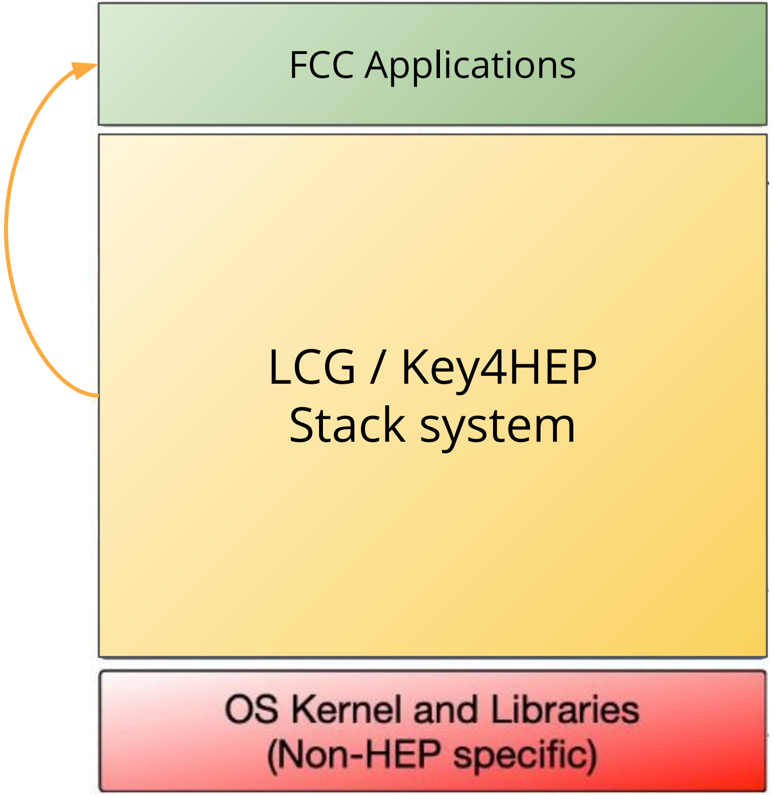
FCC Applications

LCG / Key4HEP
Stack system

OS Kernel and Libraries
(Non-HEP specific)

Community-oriented mindset

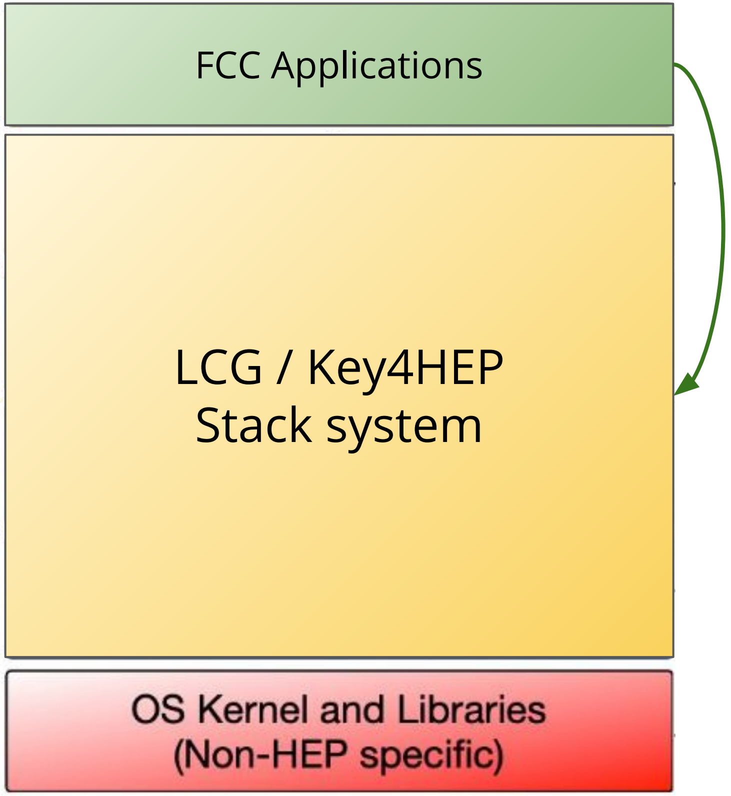
Benefit from the common efforts



Community-oriented mindset

Benefit from the common efforts

Provide feedback to the community



FCC Applications

LCG / Key4HEP
Stack system

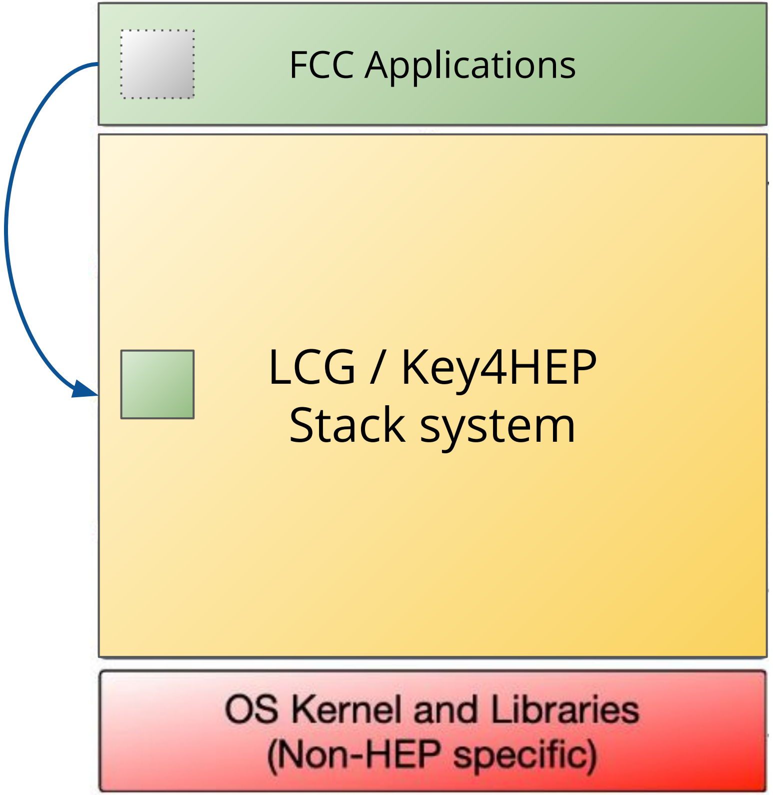
OS Kernel and Libraries
(Non-HEP specific)

Community-oriented mindset

Benefit from the common efforts

Provide feedback to the community

Build software aiming to contribute



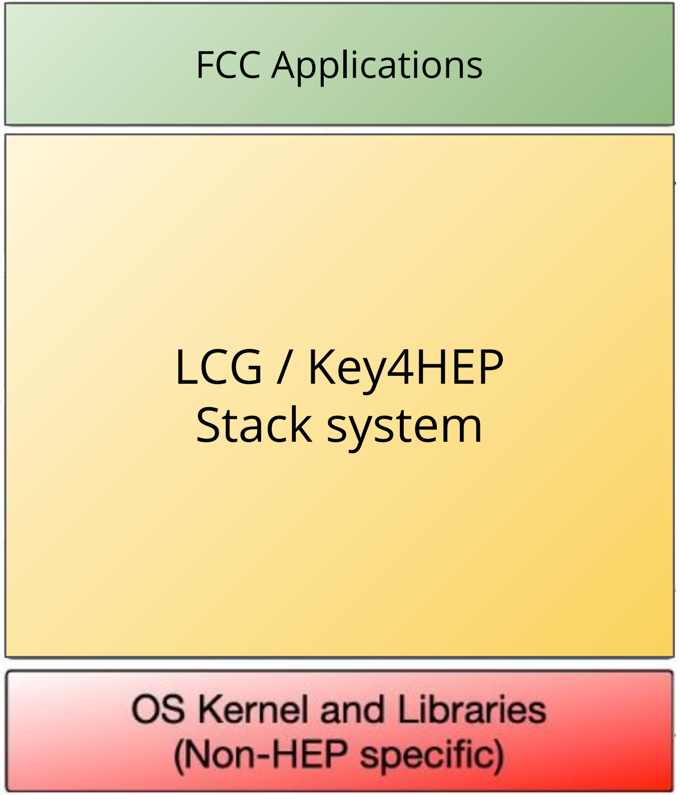
Community-oriented mindset

Benefit from the common efforts

Provide feedback to the community

Build software aiming to contribute

Documentation, forums, support



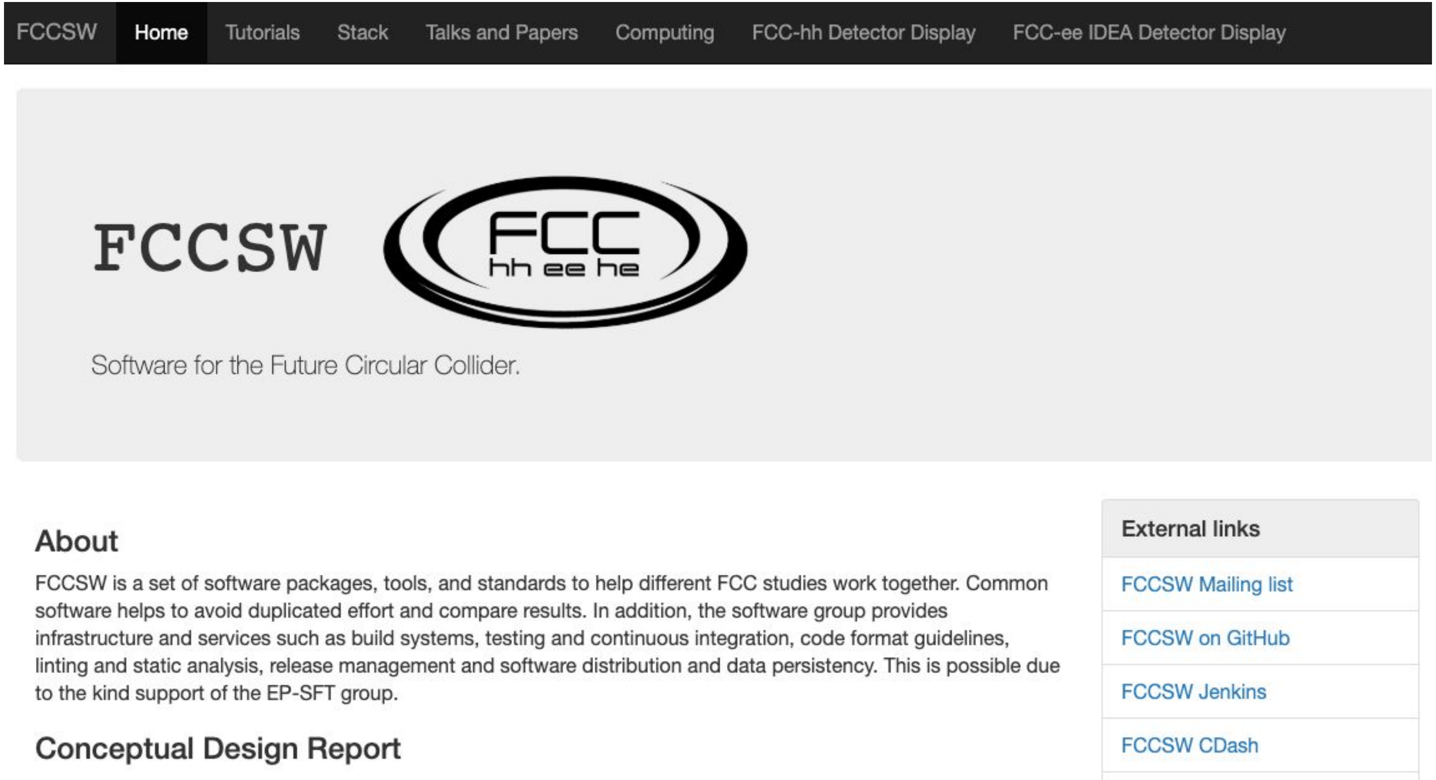
FCC Applications

LCG / Key4HEP
Stack system

OS Kernel and Libraries
(Non-HEP specific)

FCC Software Website

Documentation, tutorials, infrastructure



Software for the Future Circular Collider.

About

FCCSW is a set of software packages, tools, and standards to help different FCC studies work together. Common software helps to avoid duplicated effort and compare results. In addition, the software group provides infrastructure and services such as build systems, testing and continuous integration, code format guidelines, linting and static analysis, release management and software distribution and data persistency. This is possible due to the kind support of the EP-SFT group.

Conceptual Design Report

External links

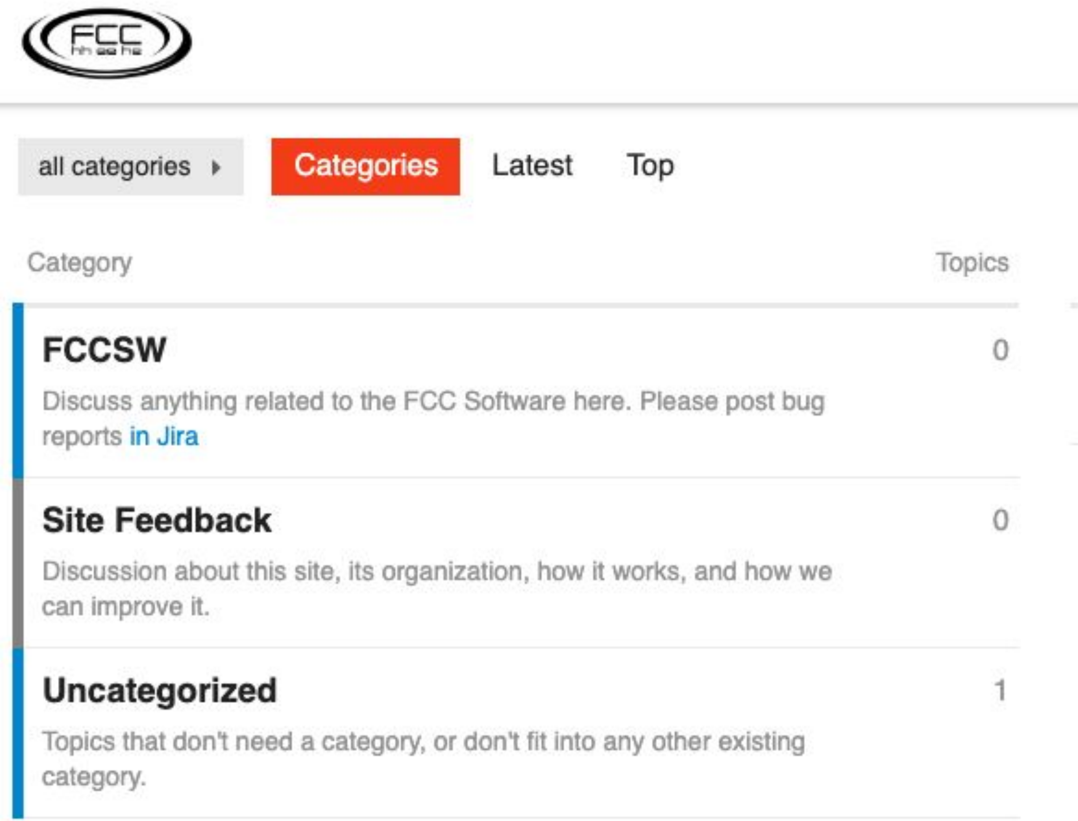
- FCCSW Mailing list
- FCCSW on GitHub
- FCCSW Jenkins
- FCCSW CDash

http://hep-fcc.github.io/FCCSW/

FCC Software Forum

Users support

https://fccsw-forum.web.cern.ch

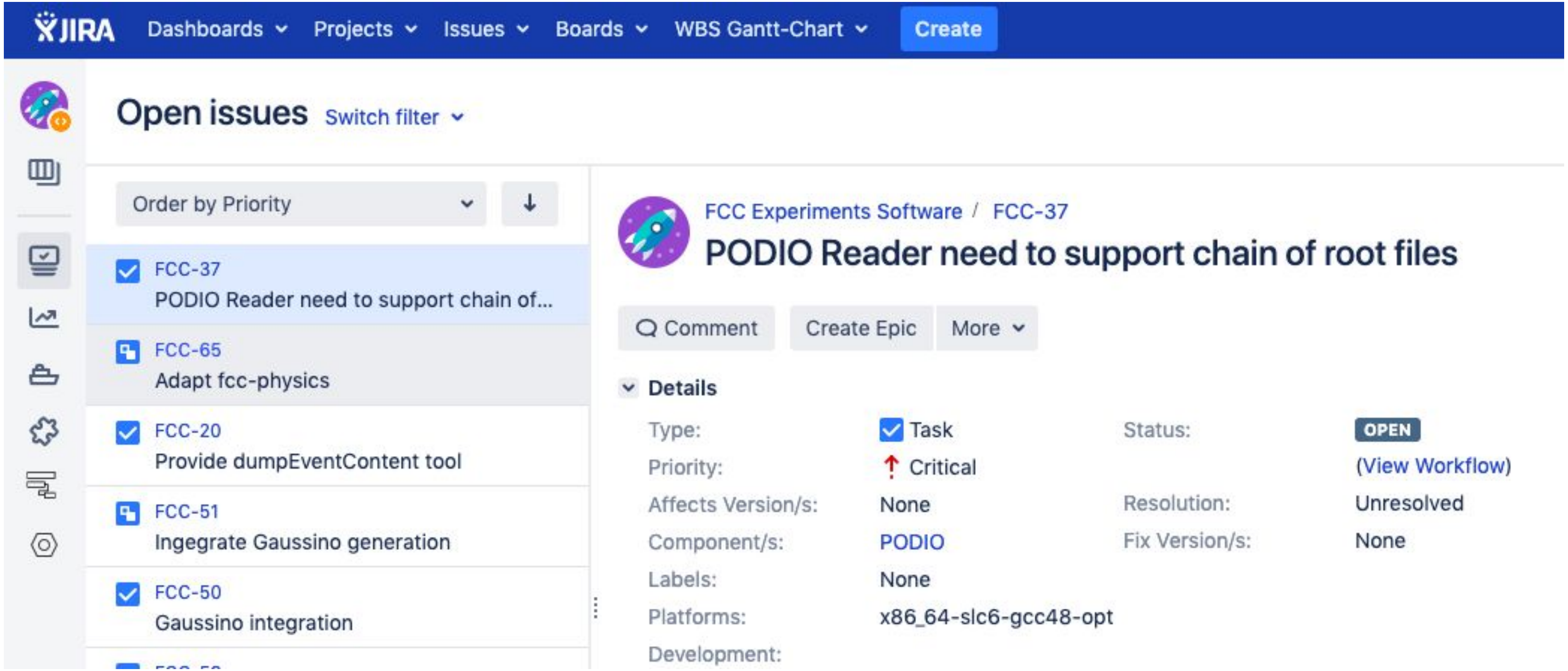


Category	Topics
FCCSW Discuss anything related to the FCC Software here. Please post bug reports in Jira	0
Site Feedback Discussion about this site, its organization, how it works, and how we can improve it.	0
Uncategorized Topics that don't need a category, or don't fit into any other existing category.	1

FCC Software Jira

https://sft.its.cern.ch/jira/projects/FCC/issues

Issue tracker



Open issues

Switch filter

Order by Priority

- ☑ FCC-37 PODIO Reader need to support chain of...
- FCC-65 Adapt fcc-physics
- ☑ FCC-20 Provide dumpEventContent tool
- FCC-51 Ingegrate Gaussino generation
- ☑ FCC-50 Gaussino integration

FCC Experiments Software / FCC-37

PODIO Reader need to support chain of root files

Comment | Create Epic | More

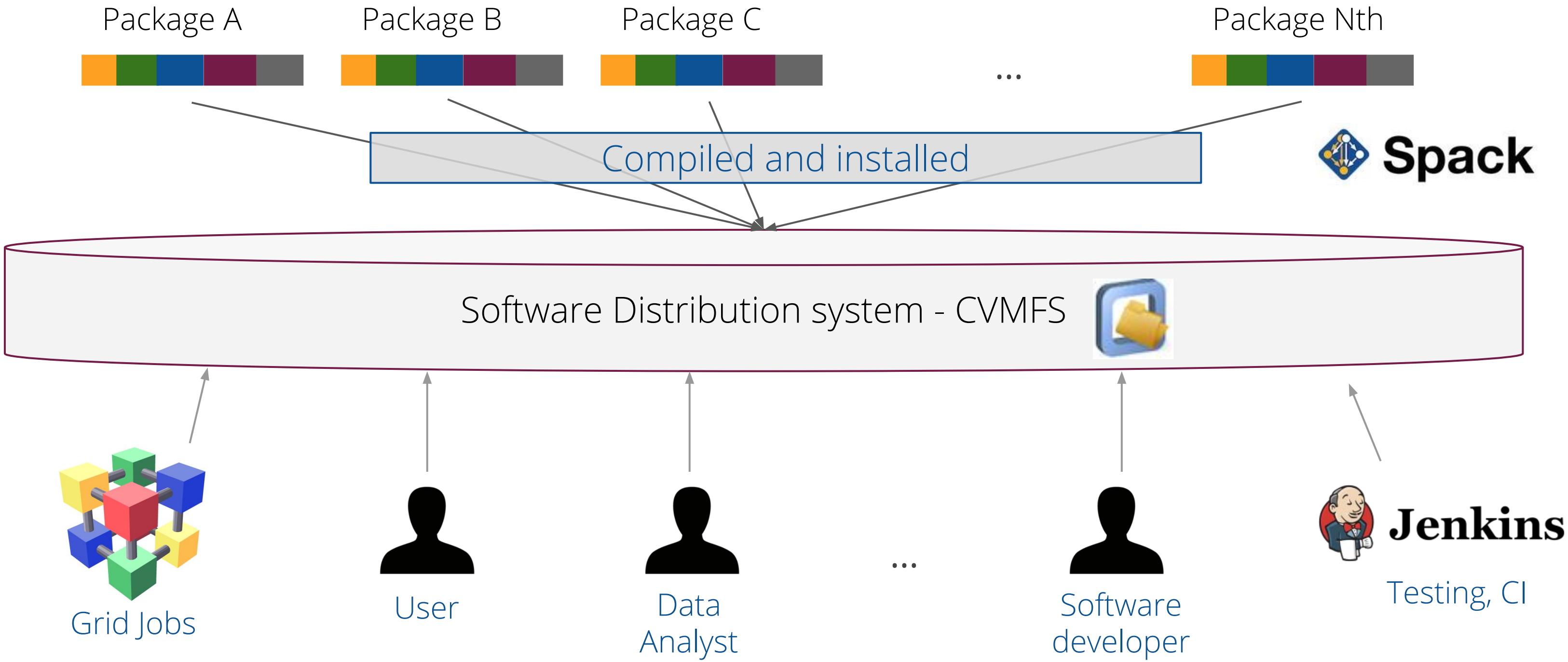
Details

Type:	☑ Task	Status:	OPEN
Priority:	↑ Critical		(View Workflow)
Affects Version/s:	None	Resolution:	Unresolved
Component/s:	PODIO	Fix Version/s:	None
Labels:	None		
Platforms:	x86_64-slc6-gcc48-opt		
Development:			

Providing software to users

- Software stacks need to be made available to users
  - In parallel to the development process
- Covering different configurations
  - Compilers, platforms, architectures, stack versions
- Stable and bleeding edge versions
  - Releases (static), Nightlies (ephemeral, likely unstable)
- Flexible to cover different use-cases
  - Production, grid jobs, developments, testing

Providing software - *Current approach*



Package A

Package B

Package C

...

Package Nth

Compiled and installed

Spack

Software Distribution system - CVMFS

Grid Jobs

User

Data Analyst

...

Software developer

Jenkins

Testing, CI

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

Tooling for users

Tooling for users

- Prepare full or partial environment

Tooling for users

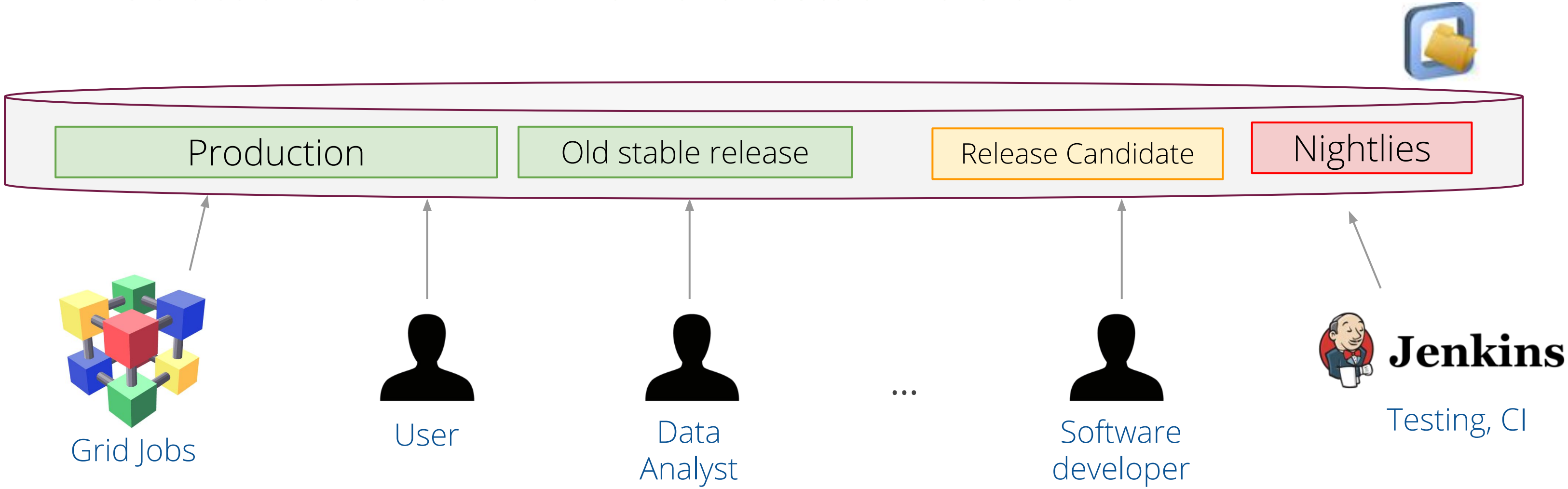
- Prepare full or partial environment
  - All the packages to run data analysis
  - Set up all the dependencies to develop a package

Tooling for users

- Prepare full or partial environment
- Select and switch from different stack versions

Tooling for users

- Prepare full or partial environment
- Select and switch from different stack versions



Tooling for users

- Prepare full or partial environment
- Select and switch from different stack versions
- Hide complexity

Tooling for users

- Prepare full or partial environment
- Select and switch from different stack versions
- Hide complexity
- Reproducibility of environment

Tooling for users

Not yet there, looking into existing experiments workflows (e.g. LHCb)

- Prepare full or partial environment
- Select and switch from different stack versions
- Hide complexity
- Reproducibility of environment

Conclusions

- Common base for future the experiments: *Key4HEP stack system*
- Community-oriented mindset:
  - Build generic tools useful for similar experiments / technologies
  - Contribute to the common layer
- Rely on stable, robust, maintained and efficient software
- Developers
  - Follow good practices: testing, coding techniques, agile development cycles
  - Follow community guidelines
- Users
  - Provide them with tooling to effectively handle software stacks

Thank you for your attention