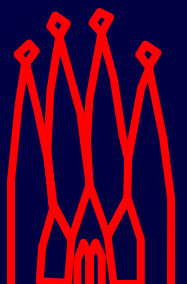
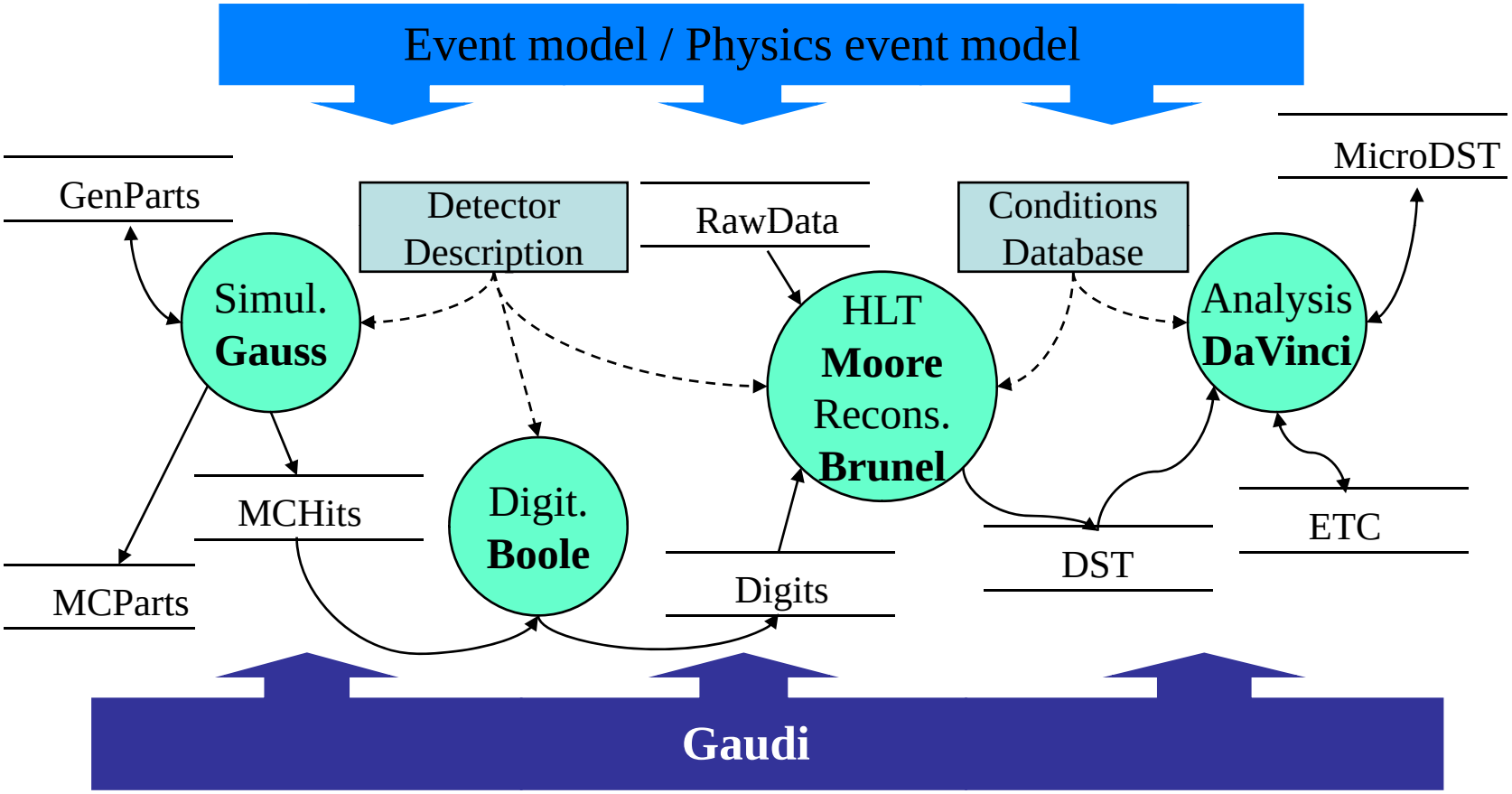
1

Overview of LHCb applications and software environment

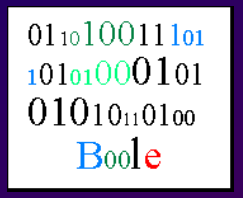


LHCb applications



Main LHCb applications

- **Gauss**
  - Event generation and GEANT4 simulation



- **Boole**
  - Detector response and digitization
  - Output in same format as real data

- **Moore**
  - Trigger reconstruction and HLT selection
  - Runs both online (in trigger farm) and offline

- **Brunel**
  - Event reconstruction
  - Output Tracks, Particle ID, “ProtoParticles”

More main LHCb applications

- **DaVinci**
  - Physics analysis framework
  - Manipulate particles and vertices to identify and measure physics processes

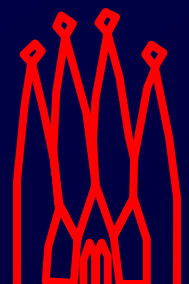
- **Panoramix**
  - Event and geometry display
  - Scripting based on Python

- **Ganga**
  - User interface for handling job preparation, submission and retrieval (e.g. on the grid)

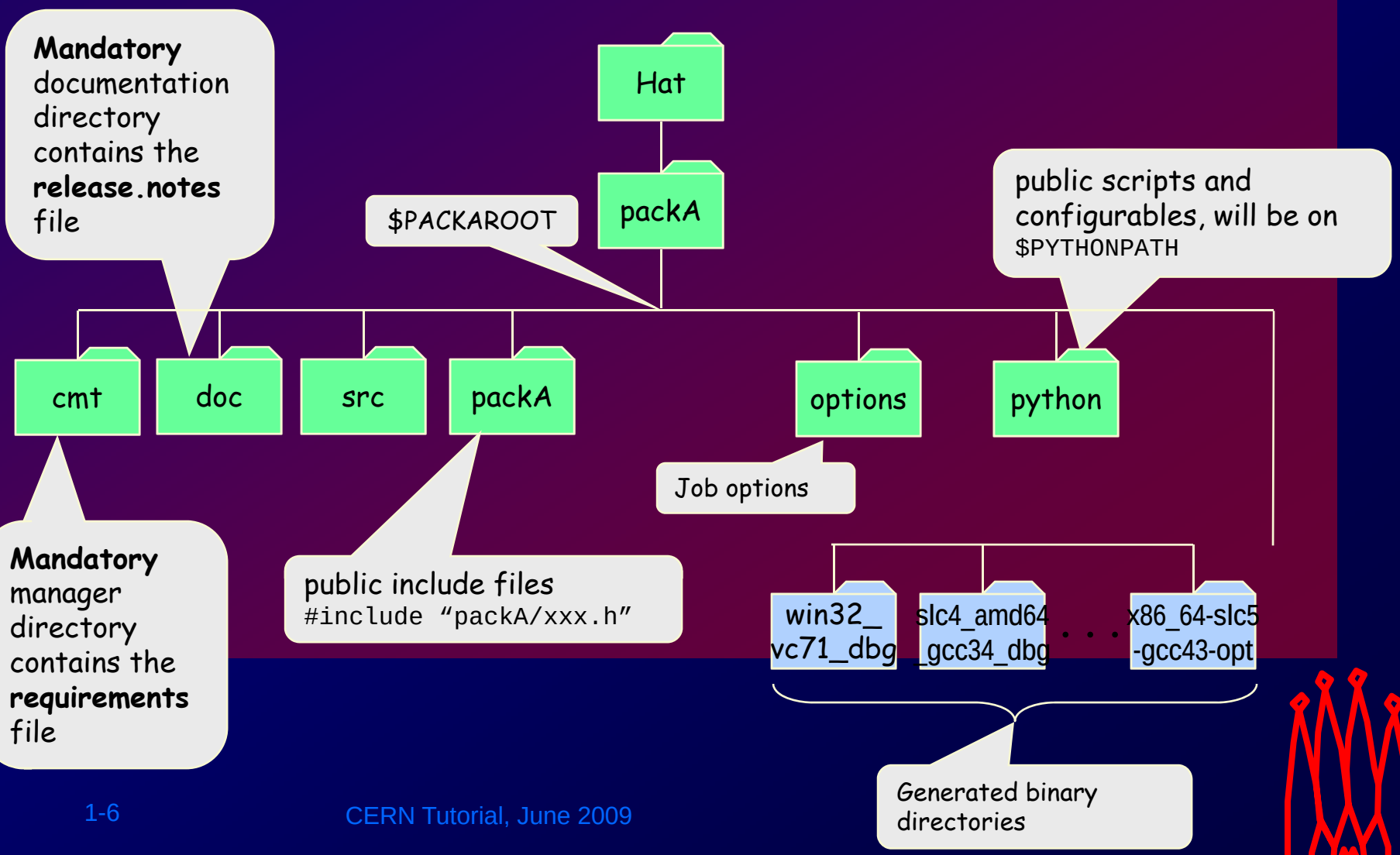
+BENDER
+VETRA
+PANOPTES
+ORWELL
+…

Applications are assembled from packages

- Package Definition:
  - Collection of related classes in a logically cohesive physical unit
  - Minimal entity that can be versioned
- Reflects on
  - Logical structure of the application
  - Organizational structure of the development team



Package: Structure



Package versions

Packages have several versions

- Defined in cmt requirements file
- Version number formatted according to convention:

Major version

- Indicates a change in the interface: all packages that use it may have to change

Minor version

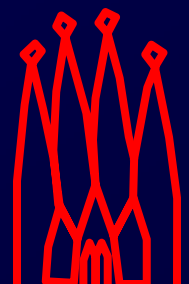
- Indicates an internal only change

Patch version

- Not usually present. A minor bug fix to an existing release

Project

- Projects are a collection of packages that are released together
  - One project per application (e.g. Brunel, DaVinci)
  - Several independent projects for components (e.g. Lbcom, Rec, Hlt, Phys, Analysis, Online)
  - Two projects for the framework (Gaudi, LHCb)
- Users work in the environment defined for a given version of the chosen project
  - e.g. SetupProject DaVinci v23r1

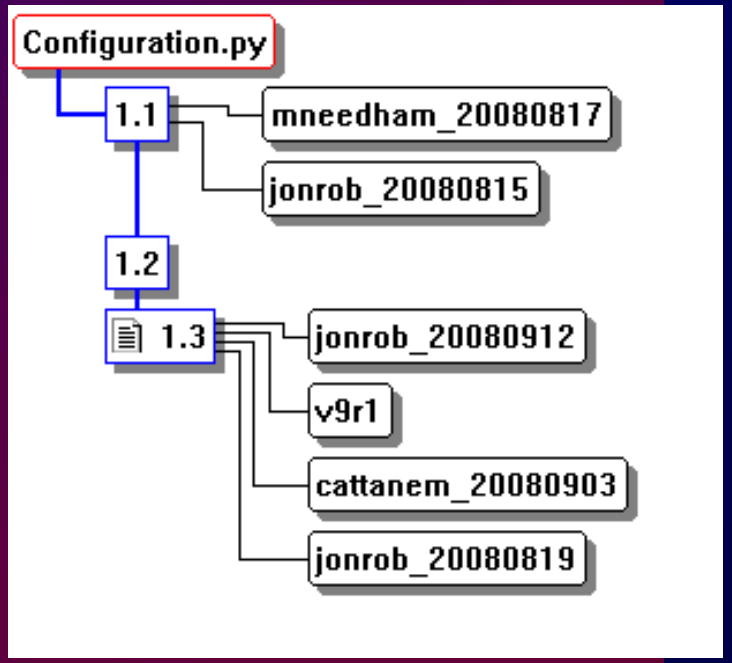


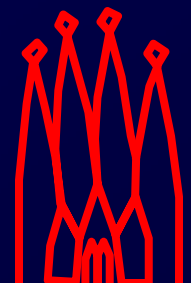
CVS

Version Control System

- **Record the history of your source files**
- **Helps you if you are part of a group of people working on the same project.**

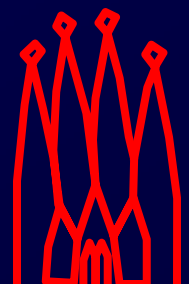
(Repository, Module, File, Version, Tag)





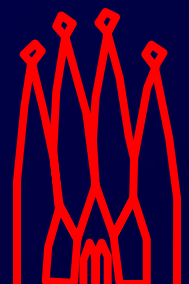
CVS: Common Repository

- **LHCb Repository on CERN-IT CVS server**
  - **Web browsable**
    - http://isscvs.cern.ch/cgi-bin/cvsweb.cgi/?cvsroot=lhcb
  - **World readable if authenticated**
    - SSH authentication
      - Automatic if authenticated in CERN AFS cell (lxplus)
      - Detailed instructions at http://cvs.web.cern.ch/cvs/howto.php#accessing-ssh
  - **For write access**
    - register with Hubert.Degaudenzi@cern.ch



SVN

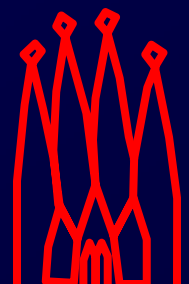
- **SVN is a replacement for CVS**
  - **Gaudi repository has migrated to IT SVN service**
    - Web access: https://svnweb.cern.ch/world/wsvn/gaudi
    - Usage instructions at https://twiki.cern.ch/twiki/bin/view/Gaudi/GaudiSVNRepository
  - **LHCb will migrate before end 2010**



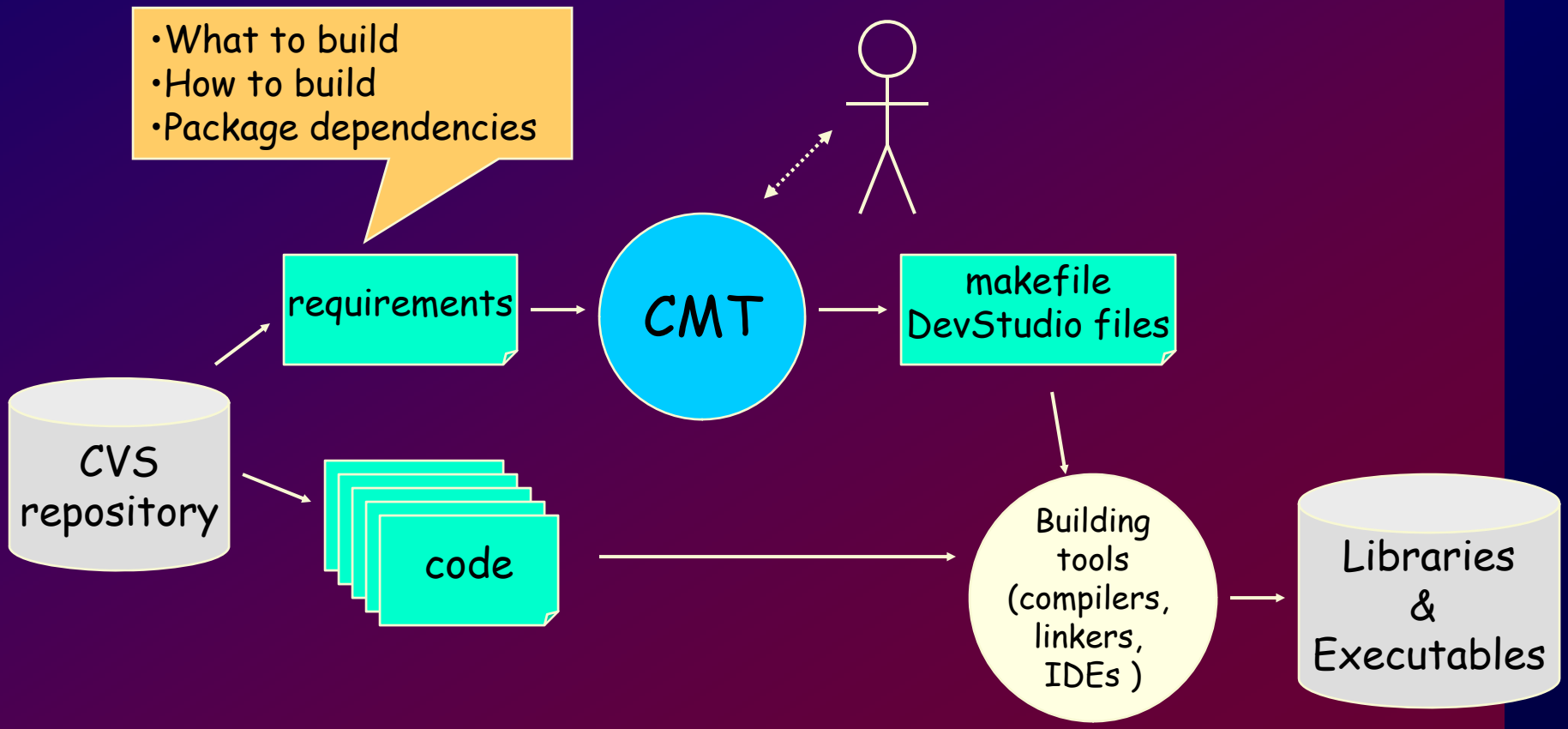
CMT

Configuration Management Tool written by C. Arnault (LAL, Orsay)

- **It is based around the notion of *Package***
- **Provides a set of *tools for automating* the configuration and building packages**
- **It has been adopted by LHCb (other experiments are also using it)**

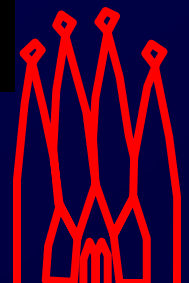


How we use CMT



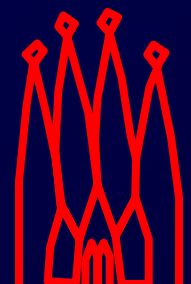
CMT: Requirements file

```
package           MyPackage
version           v1r0

# Structure, i.e. directories to process.
branches          cmt doc src

# Used packages.
use GaudiAlg      v*

# Component library building rule
library          MyPackage    ../src/*.cpp

# define component library link options
apply_pattern component_library library=MyPackage
```



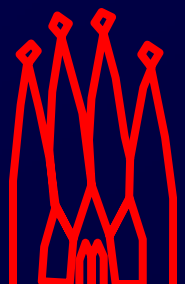
CMT: Basic Commands

- **cmt config**
  - Configures the package (creates setup and make files)
  - Invoked automatically by getpack
- **cmt show uses**
  - Show dependencies and actual versions used
- **cmt show macro <macro>**
  - Show the value of a macro for the current configuration
- **cmt broadcast <command>**
  - Recursive CMT command in all used packages found in the current CMT project
  - e.g. cmt broadcast gmake
- **cmt run <command>**
  - Executes <command> in current package environment



Package Categories

- ***Application***: is a package that sets up the environment and contains the job options needed for running a program.
- ***Library***: contains a list of classes and the list of dependent packages needed to compile it.
- ***Package group***: contains a list of other packages with their version number (e.g. LHCbSys)
- ***Interface package***: interfacing to packages not managed with CMT (e.g. Python, GSL, ROOT,…)

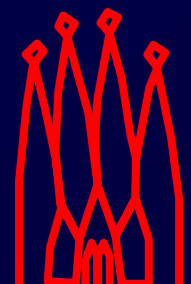


Link vs. *Component* Libraries

- Link libraries are need for linking the program (static or dynamic)
  - Traditional libraries.
- Component libraries are loaded at run-time
  - Collection of components (Algorithms, Tools, Services, etc.)
  - Plug-in

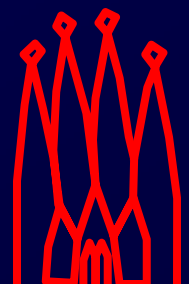
Components_dll.cpp

```
#include “GaudiKernel/LoadFactoryEntries.h”
LOAD_FACTORY_ENTRIES ( Components )
```



Setting the CMT environment

- **Create a working directory for a given project**
  - **setenv<Project> [<version>]**
  - **Creates working directory and cd to it: ~/cmtuser/<Project>_<version>**
- **Packages are searched for in Projects that are on CMTPROJECTPATH**
  - **Default is ${User_release_area}:${LHCBPROJECTPATH}**
    - Add additional paths with --nightly and --dev-dir switches of SetupProject

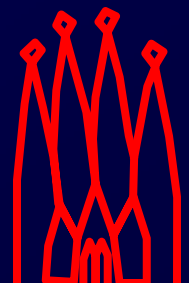


Getting a package

- The “getpack” command
  - Script combining “cvs checkout” + “cmt config”

```
> getpack [hat/]<package> [<version>] [head]
```

  - If no version given, it suggests the latest version of a package
    - N.B. Suggested version is not necessarily consistent with current environment; especially if you are not using the latest environment
- Includes transparent support for Gaudi SVN

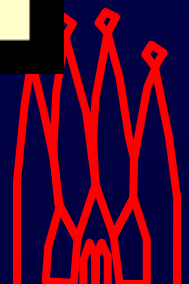


Building a package

- **Work in the /cmt directory**
  - <hat>/<package>/cmt
- **Set the CMT configuration ($CMTCONFIG)**
  - Defines platform, compiler, directories where to find binaries
    - slc4 default: slc4_amd64_gcc34
    - slc5 default: x86_64-slc5-gcc43-opt
    - setenv CMTCONFIG slc4_ia32_gcc34
    - setenv CMTCONFIG $CMTDEB
- **Invoke the make command, via cmt**

```
> cmt make [-j][target][clean][binclean]
```

command is executed in a shell that has defined all the environment described by the requirements file



Running the application

- **Set the run time environment**

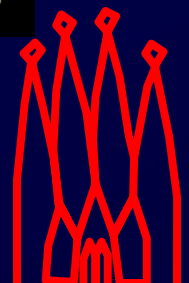
```
> SetupProject <project> <version>
```

  - Takes into account value of CMTCONFIG
  - Needed once at beginning of session, then only if environment changes (new packages checked out, requirements file changed, CMTCONFIG changed, etc.)

- **Execute the program**

```
➢gaudirun.py job.py
```

- **But see also `cmt run` command in hands on**

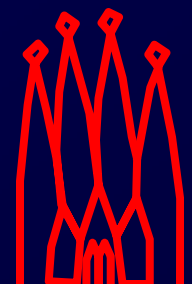


Emacs customisation

- **A customisation of emacs for LHCb:**
  - **Templates for creation of files**
    - E.g. MyAlgorithm.h, .cpp, <Components>_dll.cpp, requirements, release.notes etc.
  - **Various shortcuts for code insertions**
  - **Optionally, load an EDT keypad emulation**
- **Add following lines to ~/.emacs:**

```
(load (expand-file-name "$EMACSDIR/edt"))
(load (expand-file-name "$EMACSDIR/lhcb"))
```

  - **Or copy from $EMACSDIR/.emacs**



Exercise

Now read the web page attached to this lesson in the agenda and work through the exercises