Table of Contents

Software configuration DB Prototype

Objectives

In order to ease the management of the software released by LHCb, it was decided to put in place a database of the various applications and projects used by the experiment. This database gathers information about the projects, the versions released (and in which configuration). This information is already available from the release area or from the version control repositories, but it is not easily usable for all types of queries (e.g. which projects in depend on Gaudi vXrY...). It was therefore decided to duplicate this information in a database. At the same time, the Ariadne project started using the Neo4j graph database, which seemed to be a good way to store the graph of dependencies between projects. A prototype was therefore (quickly) prepared to validate this approach.

Design

In the Software configuration DB, there is a node for each couple (Project, Version). All the dependencies between couples (Project, Version) are specified as "REQUIRES" relationships. This dependency graph is the core of the database and allows navigating the dependencies in the various stacks.

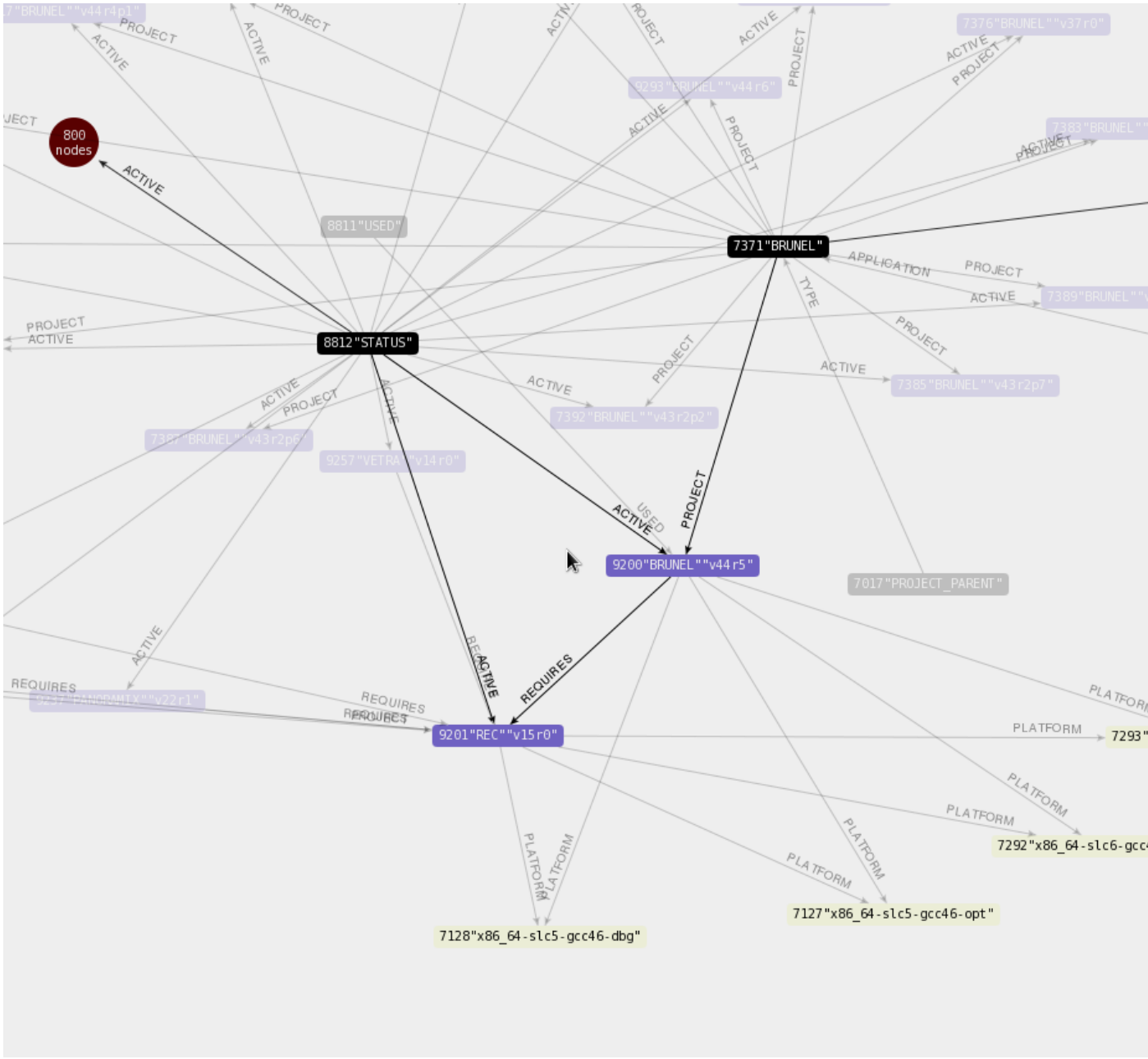
A number of other nodes/relationships have been created in order to ease the look up of nodes in the DB: * A node is created for each project (Brunel, DaVinci etc), which is linked to all the (project, version) for this project by a relationship of type "PROJECT". The project nodes themselves being related to a project_parent node. This allows to quickly list all projects, and all versions existing for a specific project.

* platform nodes, one per platform, to which the (Project,version) are related ("PLATFORM" relationship) if the project was released for this platform.

* An application node, to which all projects that are specifically marked as applications are related (relationship of type APPLICATION).

* A "STATUS" node, to which all projects present on disk are related with a relationship of type "ACTIVE"

* A "USED" node, to which each used application can be related (relationship type: "USED") if they are effectively used. This is used to perform a check of the install area, where the projects that can be archived are the ones that are on disk and that are not used by an application that is used.



Commands and recipes

Command list

In order to populate and keep the database up-to-date, a number of command to the **LbRelease** package in LbScripts:

Command	arguments	*Description
lb-sdb-import	project version	queries a project from SVN and inserts it in the DB. The -r option allows traversing recursively all dependencies of the project and creates them if not already done
lb-sdb-clear	-s a or -s u	-s a clears the active flag for all projects, -s u resets all the used flags
lb-sdb-addplatform	project version platform	Adds a platform for a project/version
lb-sdb-setapp	project	Specifies that the project is an application (e.g. Brunel, DV, but not Phys or Rec)

lb-sdb-setallappsused		Sets the USED flag for all project/versions related to Applications
lb-sdb-query	-	Main DB query command see below
lb-sdb-setprojectprop	project name value	Sets the property name=value to a specific project

Creating a DB from scratch

This is possible for tests, one just needs to setup the project URLs:

```
python -c 'from LbRelease.SoftConfDB.SoftConfDB import SoftConfDB; SoftConfDB().setupDB()'
lb-sdb-setprojectprop Gaudi sourceuri gitlab-cern:gaudi/Gaudi
lb-sdb-setprojectprop Gauss sourceuri "gitlab-cern:lhcb/Gauss"
lb-sdb-setprojectprop LHCb sourceuri "gitlab-cern:lhcb/LHCb"
lb-sdb-setprojectprop GEANT4 sourceuri "gitlab-cern:lhcb/Geant4"
```

DB Queries

The lb-sdb-query command can be used for multiple queries:

Command[shortcut]	Description
listProjects[l]	List all projects known
listActive[a]	List ACTIVE project (i.e on disk in the install area)
listApplications[a]	List all applications known
listUsed[u]	List all projects with the USED flag set
listVersions[v]	List all versions known for a project
listStackPlatforms[sp]	List the platforms existing for the projects the specified projects depends on. (Could be used to know which project to compile for at build time)
listPlatforms[p]	List the platforms deployed for the project/version
listDependencies[d]	List the dependencies of the project/version
listReferences[r]	list all project/versions depending on this one
listActiveReferences[s]	list all project/versions depending on this one that are active
checkUnused	Check for unused projects, defined as (all active projects) - (all project used by active applications)

Updating the DB and checking for unused projects

To add the new projects created and their platforms, the following command can be run in a cron, adapting the nbdays to the frequency of the cron...

```
lb-sdb-importinstallarea -t <nbdays>
```

To check for unused projects, after deleting and application the best is to run:

```
lb-sdb-importinstallarea -c
lb-sdb-setallappsused
lb-sdb-query checkUnused
```

The **lb-sdb-importinstallarea -c** and **lb-sdb-setallappsused** are needed to resynchonize the USED and ACTIVE flags in the DB before the check.

This topic: LHCb > SoftwareConfigurationDB
Topic revision: r3 - 2017-09-26 - BenjaminCouturier