Source Distribution and RPM Signing in the gLite Release Process

EGEE07 Conference, Budapest, 1-5 Oct 2007
The gLite Release Process and Porting Session

Juha Herrala, SA3/CERN

Overview

- Releasing gLite 3.1 vs. 3.0
- Source distribution
- Signing RPMs
- Discussion

Releasing gLite 3.1 vs. 3.0

- **ETICS as a build engine for gLite 3.1**
  - instead of the gLite 3.0 build system.
- **Driven by patches, packages are picked from the ETICS repository**
  - and inserted in the gLite repository.
- **Releases and updates per node type**
  - no longer releases and updates for the whole middleware.
- **Releases for multiple platforms**
  - Currently the ETICS project provides the required system architecture (32/64bits) and operating system variants (slc3/slc4/etc).
- **Main challenges**
  - Growing number of architecture/os combinations
    - Consequences to development, testing and certification activities
    - How to best manage patch information and repositories
  - How to use the ETICS system in an optimal way
    - And how to move to developer-driven builds with ETICS

gLite Source Distribution

- **Currently only binaries are distributed**
  - Approach for the source distribution depends of the requirements received from the community
  - For example, software reviews require lighter-weight system in place than installation from source.
- **The mapping information between binary rpm packages and source CVS tags is already available in the build system (ETICS).**
  - But that information is not easy to extract
- **Three basic options for enhancement**
  - Distributing CVS tags with the middleware
  - Distributing source tarballs
  - Distributing source RPMs (installation?)
- **Distributing CVS tags with the middleware**
  - Tags could be embedded in the RPM packages.
  - Alternatively mapping between RPM versions and CVS tags could be made available outside of the build system.
- **Distributing source tarballs**
  - Tarballs could be created during ETICS build and copied in the ETICS repository with the binary RPM.

gLite Source Distribution

- **Distributing source RPMs**
  - ETICS supports building source RPMs by activating another switch in the build config: `--createsource`
  - But, ETICS does not guarantee installable source RPMs.
  - Package managers and developers should provide correct configs/specs/requirements for proper build of source RPMs.
- **Testing and certifying responsibilities of the source distribution remain with the integration team.**
  - Limited resources
- **Generally it should be agreed:**
  - Method to put in place: CVS tags, tarballs, srpms?
  - Should installation from source be supported..
    - Should the tarballs contain installation instructions?
    - Should all the released srpms install?
    - Which are the supported architecture/os combinations?
  - Defining the process and sharing the effort and responsibilities

Signing RPMs

- **Currently the RPMs in gLite releases have not been signed.**
  - Lately there has been more frequent requests for signed packages.
- **An RPM package can be signed in different ways:**
  - Signing a package at build-time.
  - Replacing the signature on an already-existing package.
  - Adding a signature to an already-existing package.
- **In addition, within the gLite build and release process, there are several possible phases to sign the packages, for example,**
  - artifacts could be signed in the end of the build process
  - ETICS could sign when the artifacts are copied in the ETICS repository
  - “mirrored” ETICS repository containing signed rpms
  - gLite release management tools could sign when the packages are taken from the ETICS repository and inserted into the gLite (certification) repository
  - when moved to pre-production service
  - when released to production service
- **Signing in the end of the build process**
  - Change in build scripts
  - (Private) key management challenge
  - There would only be signed rpms available in all repositories

Signing RPMs

- **ETICS signs artifacts when copied in the repository**
  - Would require effort from the ETICS team to implement
  - There would only be signed rpms available
  - Does not concern developers, release process
- **“Mirrored” ETICS rpm repository with signed RPMs**
  - Source for gLite release management tools
  - Both signed and unsigned (ETICS) rpms available
- **Signing when packages are inserted into the gLite certification repository**
  - Central effort
  - Both signed and unsigned (ETICS) rpms available
- **Signing when released to pre-production or production**
  - A centralized effort not involving developers and ETICS
  - Both signed and unsigned (ETICS) rpms available
- **Which packages to sign**
  - Packages released “from now on”
  - Already released packages as well
  - External packages will not be signed
- **Key distribution**
  - Public key of the signing entity should be available and inserted in the RPM database of all gLite nodes requiring signed packages.

Discussion

- **Should gLite source distribution be enhanced?**
- **Three basic options for enhancement:**
  - Distributing CVS tags with the middleware
  - Distributing source tarballs
  - Distributing source RPMs (installation?)

- **Should gLite RPM packages be signed?**
- **Several strategies for signing, for example:**
  - Artifacts could be signed in the end of the build process
  - ETICS could sign when the artifacts are copied in the ETICS repository
  - Implementing a “mirrored” ETICS repository containing signed rpms
  - When the packages are taken from the ETICS repository and inserted into the gLite (certification) repository
  - When moved to pre-production service
  - When released to production service