Table of Contents

Staged roll-out brainstorming series: repositories.....1
Attendance.....1
Purpose of the meeting.....1
Relevant points of discussion.....1
Decisions.....4

Staged roll-out brainstorming series: repositories

- Date: Thu 28 May 2009
- Agenda: na
- Description: Staged roll-out brainstorming series: repositories
- Chair: none

Attendance

Oliver, Maarten, Nick, Antonio

Purpose of the meeting

General goal: to agree between SA3 and SA1 on a strategy to do a phased roll-out of the middleware to a number of "early adopter" sites. The process has to be compliant to the future middleware distribution model from EGI_DS . We focus our analysis on two real use cases: WMS 3.2.1-4 (now in PPS) and BDII 5 (currently in certification).

Points to discuss:

1. Clarification of SA1 requirements about staged roll-out
2. USE CASES STUDY (brainstorm)
    1. Staged Roll-out for WMS 3.2
    2. Staged roll-out for BDII 5.0

for both examples we need to clarify the following points:

1. Repository/ies
    1. How many repositories should we have?
    2. Who controls the repositories?
    3. Should we have separate unstable/release repositories?
    4. rollback management (production repository): EPOCH
    5. rollback management (production repository): NON-EPOCH PROBLEMS
2. Failure management
3. Sites
    1. How do you identify the sites involved in the early stages?
    2. i.e. How do you know which site to contact in each case?
    3. How to communicate with them?
    4. How to report issues?

During the discussion we constantly try and map our concept assumptions and decisions in the future EGI/UMD world. For the sake of the future reviews I have tried and mark in green the arguments which I identified as specifically pertaining EGI/UMD.

Material:

- Diana's thoughs about staged roll-out

Relevant points of discussion

Clarification about SA1 requirements on staged roll-out

(Antonio) SA1 doesn't want to put technical requirement on the implementation either of the repositories or the procedures to manage them. The only operational requirement is very well stated in Diana's document "_a repository should be relied upon at any given time, without the site administrator having to worry if they need to pin any rpm at a particular version (unless they want to, of course) if they need to (re-)install any given service._" . What we really want to avoid is to have a repository which contain package X and then a message in the release page saying "Please don't install package X". No requirements on the implementation. Specifically the concept of "downgrading by upgrading" introduced at the end of the proposal (second last paragraph) has not to be regarded as an SA1 requirement but as an example. Actually this method, if the component's version number is changed can in our opinion give way to a lot of misunderstandings, apart from being technically difficult to manage in a decentralised context.
(Maarten): It would be nice to have the possibility to do "downgrade by upgrading" in some cases as we have done in the past, it is not impossible to do it, but it is very expensive. Furthermore it doesn't cover all possible cases (for example doesn't work to roll-back changes in the configuration or in the file system)
(Oliver):

- We have to take into account that a bunch of UK sites have expressed their preference for the "downgrading by upgrading" method
- It is clear however that when once a broken update has been rolled-out at a site some sort of action will always be needed locally in order to restore the previous situation. This is independent on the policies and techniques used for the roll-back of the repository/documentation (point taken by everyone)

Repositories for staged roll-out (the brainstorming part starts here)

The use case under study is the roll-out the WMS 3.2 currently in PPS

How many repositories should we have ?

We start from the operational assumption to have one repository per node type. It's not clear whether, when and how this is going to happen in UMD, but the current public interface to the repository is already node-oriented although the back-end is a single one so the assumption doesn't imply any change in the logic for us.

Should we have separate unstable/release repositories ?

As Diana states in her document *we should have several operations repositories, one for production, one for the managed rollout.*
We could call them release and preview or current and next, or stable and unstable, etc.etc.
At that point the EGI MU could control the current+next repositories and most of the roll backs could be done by never moving the rpm from the next to the current repository.
This should make the rollback management easier, provided it has to be done on the unstable/next repository. One will simply do the staged rollout by using an unstable repository and roll back by removing the rpm in the next/unstable repository.

Supposing to have the repository *current(C)* and *next(N)* , early adopters (EA) should configure permanently both C and N, while standard production sites should configure only C . No changes in the configuration of the yum config file should be requested at the early adopters sites for the managed roll-out: all the changes would be managed internally in the repository and the EA sites should be requested only to upgrade or, in the worse case, to downgrade.

The procedures related to the use of the repo N do not necessarily have to be *easy* . It is important instead that the use of the repo C at any given time is as straightforward as possible.

This model only makes sense if the naming convention for packages in both repos C and N are the same. So

already in our case we couldn't use the pps-glite-WMS but we should use glite-WMS .
What do we miss by getting rid of the PPS naming convention?
This aim of this convention is to be able to pick random packages from the PPS glite update release and recompose them in a PROD glite update. It is likely that the approach will be different with the UMD product teams. Probably with the differentiation of the product repositories every single product team will release a whole "blob" including their product plus the dependencies (e.g. voms at a given version) and the dependent component will not necessarily be all at the same version for all the product. Then there will be a list of recommended component versions endorsed by the EGI MU in order to encourage the convergence between product team, but this list will not be constraining for the developers. It is possible that at some point WLCG may want to define their own list of appropriate versions and require their sites to stick to it, but in that case WLCG will necessarily have to manage a repository on their own . In consideration of what said we think that we can get rid of the pps naming convention for this exercise.

Two hypothesis for the content of the repo N :

1. the *delta* (only the new rpms) b. the full repository with the new version (e.g. the full WMS 3.2.1-4 repo)

b) has the pro that the mirroring process in easier to implement, a) is better for visualisation although slightly more difficult to manage. For the time being we give the preference to option a)

Analysis of workflows for the management of the repository N

basic workflow

When the new version is available we put the new rpms in N and ask the sites to update (how? to be detailed) if everything is fine (e.g. some weeks of operations) we move the new rpms in C. If problems are observed at the EA sites that make the new version unusable the rpms are removed (zapped) waiting for a new iteration. If the application of the new update has compromised the functionality of an EA, the EA may be requested to downgrade, otherwise they can simply wait for the next iteration.
If an urgent update (e.g. security update) has to be fast-tracked to all the sites while this process is pending (maybe weeks), this could be done by bumping up the *release number* (-N) of the version existing in production . We assume here that fast-tracked changes do not contain major or incompatible changes. e.g., an urgent update to WMS 3.1.16-0 requested while the staged roll-out of WMS 3.2.1-4 is in progress would be inserted in C as WMS 3.1.16-1 . This practice may represent in some case a little abuse of the release number.

glite update number

the release to N has the full characteristics of a release to production. How about the gLite update number? If the release to N were labeled with an update number we could have funny cases e.g.

- roll-back in N --> skipped gLite update numbers in production
- fast-tracked security update in production --> non-sequential update numbers

The concept of gLite Update has never been abandoned even though we moved to a more and more node-oriented release because of the many inter-dependencies between services. Often a change in one single component (e.g. info provider templates) could cause many node types to be affected and we preferred to release all of them in a single update instead of producing N releases of different nodes all featuring the same change. This operational approach is more convenient and therefore will probably be continued in the future by the MU. On the other hand what is really important is that the update of multiple nodes featuring the same change happen at the same time, while the definition of a progressive gLite Update number has only the "cosmetic" function of re-mapping a tuple of different node versions into a more human-readable integer. So in the UMD perspective we don't see for the time being a functional value of the update number. Therefore we assume that we will not use it (**strong point: alternative presentation for the gLite update has to be**

defined)

documentation

The release to N has the full characteristics of a release to production and as such should be documented (release notes, rpm lists etc.) .
Open point: which version of glite should be proposed in the official glite release pages? the cutting-edge one (used by early adopters) or the previous (used by the vast majority of sites). This brings to the open question on how to consider the release to the EA as a whole, whether as a PPS-like release or as a standard release so that from the documentation point of view the staged roll-out process is transparent to SA3/MU (the management of repositories for the staged roll-out is still for SA3 though)

In both cases a re-organisation of some sort of the release pages seems to be needed. Suggestions were moved to

- provide a node-oriented update page
- document the sw released to N to work as an indipendent "plug-in" to the release notes to be added when the which could be moved be documented

(discussion to be continued in the next meeting)

communication

A similar open point emerges for the communication of the release to the early adopters. We could do it e.g. through mailing list but this way we would create a restricted "club" of friendly sites whereas we would prefer to leave open to everybody the choice to adopt the cutting-edge release. So a general broadcast may be preferable, but it should be clear to whom the release is addressed.
If a double announcement is needed for every release (one for early adopters first and for everybody later), this change in the workflow has to be heavily pre-announced.
this is somehow related to the open point above, whether to consider the release to N as a real release or a pre-production one

(discussion to be continued in the next meeting)

Decisions

pertaining the WMS 3.2 staged roll-out

- use of a "delta" Next repository
- no pps naming convention for meta-packages
- no gLite update number.

This topic: LCG > PPSTechnicalSession2009x05x30
Topic revision: r1 - 2009-05-30 - AntonioRetico