Enabling Grids for E-science in Europe

www.eu-egee.org

Biomed Application Developer's Course
30 November 2004

Experiment Software Installation in LCG-2

Roberto Santinelli

EGEE is a project funded by the European Union under contract IST-2003-508833

Contents

- The problem
- Requirements from the site administrators
- Requirements from the experiments
- The current mechanism
- Lcg-ManageSoftware and lcg-ManageVOTag

- Limits of the current mechanism
- New tool for Software propagation: Tank&Spark
- Hands on

The problem

How I can run my private application in a world-wide Grid?

How is a user guaranteed that the software he/she needs is in fact installed on a remote resource where the user might not even have an account?

The problem

- The middleware software is often installed and configured by system administrators at sites via customized tools like ***LCFGng*** or ***Quattor*** that provide also for centralized management of the entire computing facility.

while

- Gigabytes of Virtual Organization (VO) specific software need too to be installed and managed at each site. As of today a flexible tool to reliably do so is not available.

The problem

- The difficulty here is mainly connected to the existence of disparate ways of installing, configuring and validating experiment software.
- Each VO (LHC Experiment and not only) wants to maintain its own proprietary method for distributing the software on LCG.
- Furthermore the system for accomplishing this task can change with time even inside the same VO.
- Each single user could require to have his own software installed everywhere.

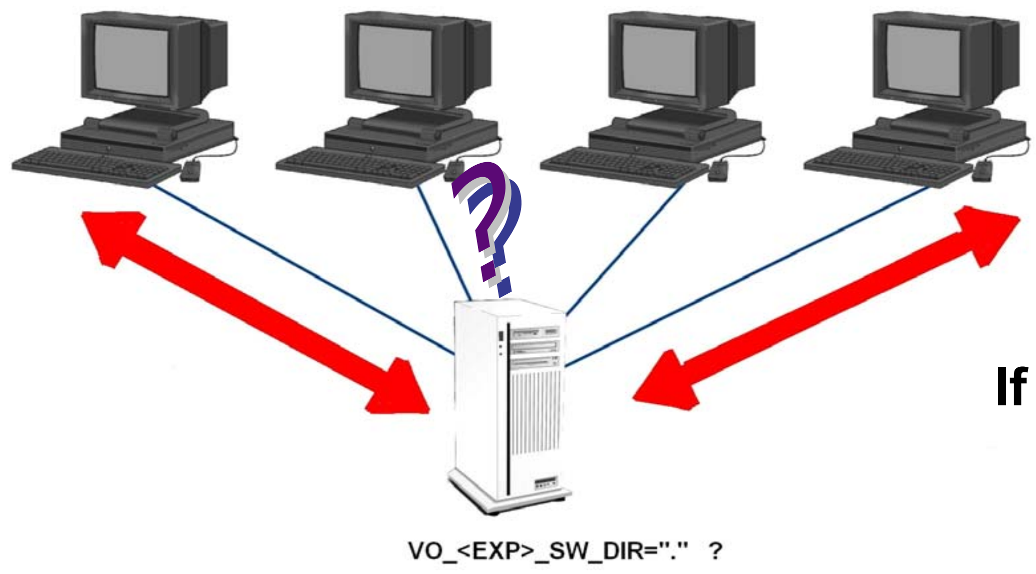
Requirements from site administrators

- No daemon running on the WNs: the WN belongs to the site and not to the GRID
- No root privileges for VO software installation on WN.
- Traceable access to the site (no pool accounts).
- In/Outbound connectivity to be avoided.
- Site policies applied for external action triggering.
- Strong authentication required
- No shared file system area should be assumed to serve the experiment software area at the site.

Requirements from the Experiment

- Only special users are allowed to install software at a site.
- The experiment could need to install very frequently its own software: it should be free to install whenever it wants the software in a site.
- Management of simultaneous and concurrent installations should be in place
- Installation/validation/removal process failure recovery
- Publication of relevant information for the software.
- Installation/validation/removal of software should be done not only through a Grid Job but also via a direct service request in order to assure the right precedence with respect to “normal” jobs.
- Validation of the software installed could be done in different steps and at different times with respect to the installation.
- The software must be “relocatable”
- It is up to the experiment to provide scripts for installation/validation/removal of software (freedom).

Current Mechanism: The VO_<EXP>SW_AREA

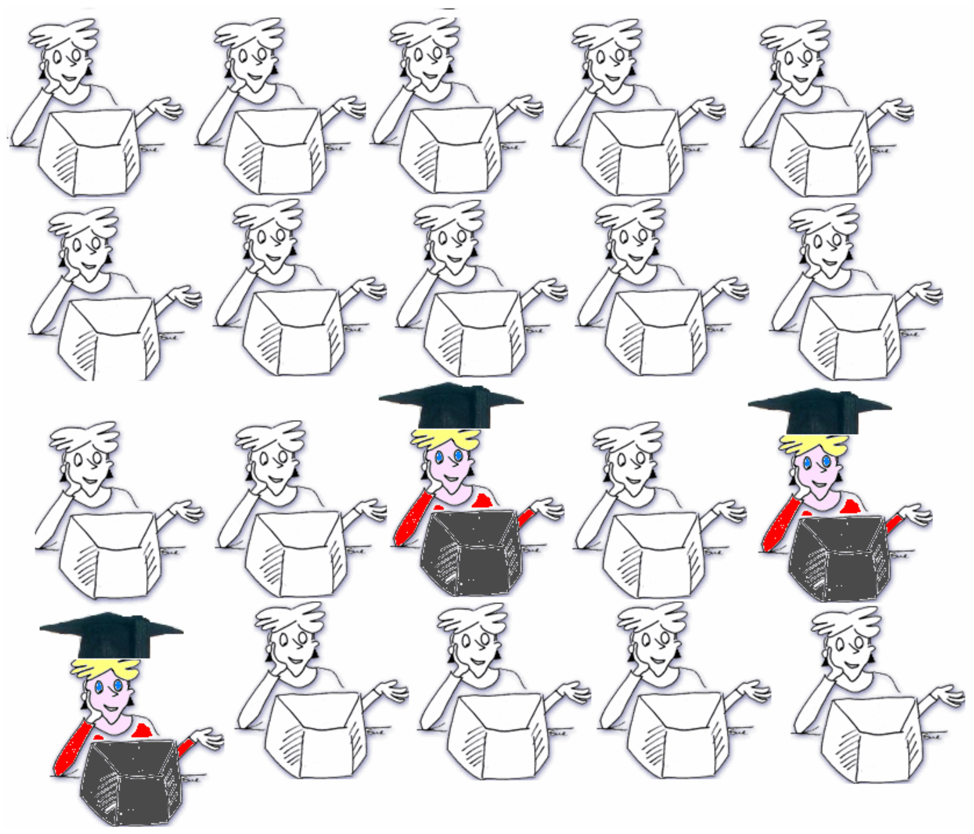


Sites can choose between providing VO space in a shared file system or allow for local software installation on a WN. This is indicated **through** ***the local environment variable VO_<EXP>_SW_DIR*****.**

If it is equal to "." (*meaning in linux the current path*) it means there is not file system shared among WNs and everything is installed locally, on the working directory, and later on scratched by the local batch system.

Vice versa the software can be installed permanently on the shared area and made always accessible by all WNs.

Current Mechanism: The ESM



Only few people in a given collaboration are eligible to install software on behalf of the VO.
These special users have write access to the VO space (in case of shared file system) and privileges to publish the tag in the Information System.
The VO Manager creates the software administrator Group. People belonging to this group are mapped to a special Administrative VO account in the local grid-mapfile at all sites.

Current Mechanism: The workflow

1. The ESM checks the resources available and decides where the new release must be installed.
2. The ESM creates a tarball file containing scripts to install (and validate later on) and bundles of experiment software.
3. The ESM runs the “special lcg-tool” to install software at a site by submitting “special jobs” explicitly on these sites.
4. The ESM runs verification steps by submitting again other “special jobs”
5. The ESM publishes tag by using “special tool”. These tags indicate the software is available to subsequent running jobs .

Current Mechanism: Step 0- Decide where install

The first step foreseen in this chain of actions is decide where the software must be installed.. For that the ESM must know what are the resources available for his VO.

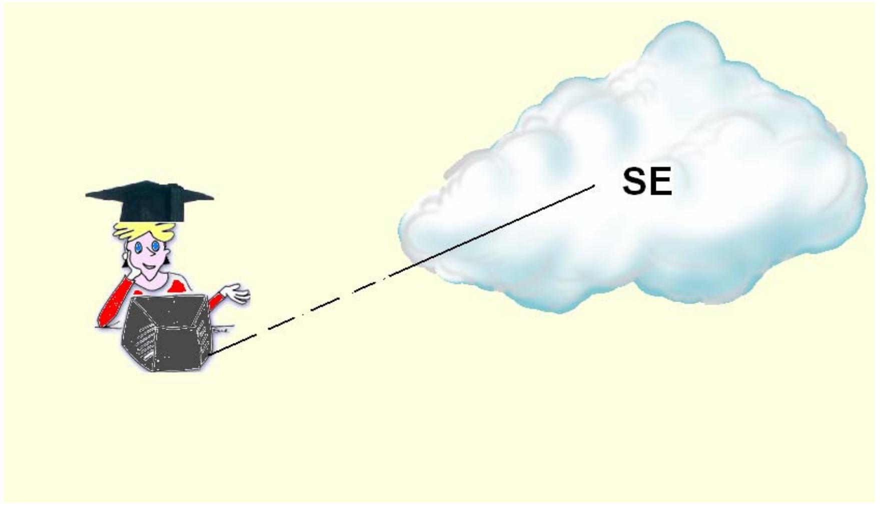
```
% lcg-infosites --vo gilda ce
```

One by one the Computing Elements shown by this command must be considered for subsequent steps. The ESM must also know which Storage Elements are available.

```
% lcg-infosites --vo gilda se
```

The ESM will chose one of these SEs which will be the target of the next command.

Current Mechanism: Step 1-Prepare your tarball



The ESM prepares a stand-alone self-contained distribution of the VO software (all dependencies are solved and needed packages included). The software bundles are packaged together with scripts used to perform the installation and validation.
Such a bundle is then uploaded to a SE in GRID through the Replica Manager (or the lcg-utils) with the –l option.

The name convention for the tarball file is the following:
<vo>-<name_of_SW>-<version>-<patch_level>.tar.gz

```
% tar czvf <vo>-<nameSW>-<Version>-<Release>.tar.gz install_tool\
       install_sw validation_sw <all_packages_constituting_exp_soft >

% lcg-cr --vo gilda file://`pwd`/<vo>-<nameSW>-<Version>-<Release>.tar.gz
       -d <SE> -l lfn:<vo>-<nameSW>-<Version>-<Release>.tar.gz
```

Current Mechanism: Step 1-Prepare your tarball

During the preparation for the installation the ESM must create 5 scripts and name them in the following way:

1. **install_tool:** script used for the installation of ancillary experiment related tool (ex. Pacman, DAR, SCRAM, RPM). These tools will be used for the subsequent software installation step
2. **install_sw:** script used for the experiment software installation. The script could reliy on some experiment-specific tools installed previously
3. **validation_sw:** script used for validating software previously installed.
4. **run_sw:** script used to set the right environment and then to run the exp. Software
5. **uninstall_sw** script used to uninstall the software.

The toolkit provided by LCG will complain if these scripts are not supplied.

Current Mechanism: Step 2: create your jdl for installation and/or validation

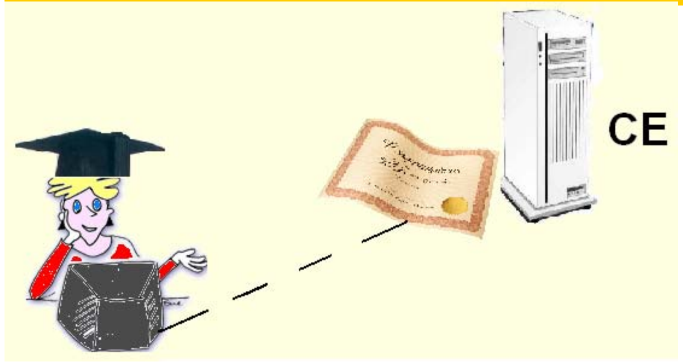
The ESM makes a JDL file in order to trigger an installation/validation process on a given site. Two scenarios:

1. *WNs sharing software installation directory : the software is installed permanently and always accessible from any WN for subsequent jobs.*
2. *Standalone disk space:*
   - *Software is installed and verified in the job working directory in the same job. All dependencies are verified.*
   - *Tag published means that the software can be successfully installed on a WN*
   - *Working dir scratched when job ends*

mandatory* : *The fields between <> in the following JDL must be the same as the ones used for naming the tarball

```
Executable = "/opt/lcg/bin/lcg-ManageSoftware";
OutputSandbox = {"stdout", "stderr"};
stdoutput = "stdout";
stderror = "stderr";
Arguments ="—validate –vo <vo> -V <version> -S <name_of_SW> -N 1 –R\
                    <release>"
Requirements = other.GlueCEUniqueID == "<CE-uniqueID>";
```

Current Mechanism: Step 3 Run the job onto the "certified" CE



For sites on which the installation/certification has been successfully run, the ESM adds (through the appropriate tool) in the IS a Tag that indicates the version of the VO software that has successfully passed the tests.

Note. The flag published in the IS of the CE is always composed as follows: <flag>==VO-<vo>-<name_of_SW>-<version>-<release>. If the user specifies explicitly the flag with –flag option in the validation JDL , VO-<vo> will be in any case added at the head of the string. (<flag>== VO-<vo>*-<flag user provided>*)

```
Executable = "/opt/lcg/bin/lcg-ManageSoftware";
InputSandbox = "<name_of_run_script>"
OutputSandbox = {"stdout", "stderr"};
stdoutput = "stdout";
stderror = "stderr";
Arguments ="—run –vo <vo> -V <version> -S <name_of_SW> -N 1 –R\
          <release> --run_script <name_of_run_script>"
Requirements = \
Member("<flag>",other.GlueHostApplicationSoftwareRunTimeEnvironment);
```

Lcg-ManageSoftware and lcg-ManageVOTag

lcg-ManageSoftware allows the ESM to automatically install/validate/run and de-install a version of experiment software. At the end of the process, if everything went OK it publishes (removes) from the IS the corresponding flag using lcg-ManageVOTag command.

lcg-ManageVOTag: is the command which adds/list/removes the flags published on the Information System of a given CE.
It could be also used as a standalone application.
It is also invoked by lcg-ManageSoftware

Lcg-ManageSoftware

```
% man lcg-ManageSoftware
```

Mandatory to provide:

1. The action to be chosen among: --install –validate –run –uninstall
2. The –S option (the name of the software)
3. The –V option (a string describing the version number of the software going to be installed)
4. The –vo option (the VO you belong)
5. The –R option (the number of release or patch-level, D=1)

Optionally:

-N (the number of tarballs that constitute this release)

--flag : allows the user to define a customized flag (remember that in any case the prefix VO-<vo>- will be added to the head of this string)

--CE the computing element (mandatory if the user submits its job with –R option)

--SE the storage element where the user wants download the tarball(s)

--run_script that allow the user to specify a different name than the default (run_sw) for the script to be used in running a given version of software

--ov (D=1) option allows to perform the validation with (=1) or without (=0) a preinstallation process on the same job. This is useful for shared file system topology sites.

Lcg-ManageVOTag

```
% man lcg-ManageVOtag
```

Mandatory to provide:

The action

–add to add a new tag in the IS of a given CE

--remove to remove a given tag in the IS of a given CE

--list to list the tags published in a given CE

--clean to remolve all tags in a given CE

1. -vo option (the VO you belong)
2. -host the CE where you want perform the action
3. -tag (for –add and –remove actions) : the flag to remove/added in the IS of the specified CE

Limits: from site administrators concerns point of view

- No daemon running on the WNs: the WN belongs to the site and not to the GRID
- No root privileges for VO software installation on WN.
- In/Outbound connectivity to be avoided.
- Strong authentication required
- Site policies applied for external action triggering.
- No shared file system area should be assumed to serve the experiment software area at the site.
- Traceable access to the site (no pool accounts).

Limits: from Experiments requirements point of view

- Only special users are allowed to install software at a site
- Installation/validation/removal process failure recovery
- Publication of relevant information for the software.
- It is up to the experiment to provide scripts for installation/validation/removal of software. (freedom)
- The experiment needs to install very frequently its own software: it should be free to install whenever it wants the software in a site.
- Management of simultaneous and concurrent installations should be in place
- Installation/validation/removal of software should be done not only through a Grid Job but also via a direct service request in order to assure the right precedence in respect to “normal” jobs (special priority)
- Validation of the software installed could be done in different steps and at different times with respect to the installation.
  - THE SOFTWARE MUST BE PROPAGATED IN ALL WNs WHENEVER THERE IS NOT SHARED FILE SYSTEM

Tank & Spark

In order to comply with these requirements we passed from a simple set of simple scripts provided by LCG …….to a structured service : Tank and Spark.

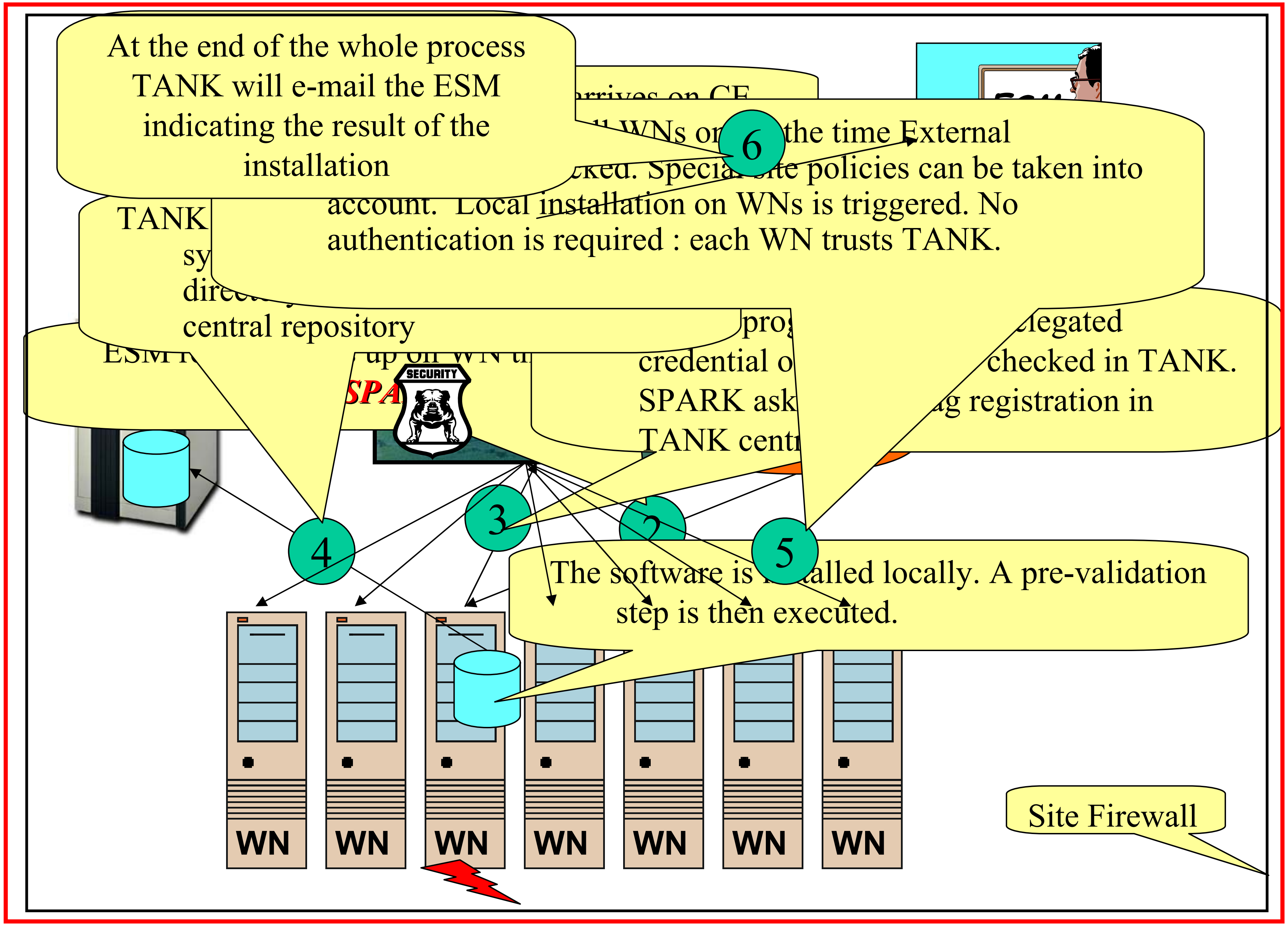
Tank & Spark

It consists of three different component:

Tank : =multithread (gSOAP based) service (running on a dedicated machine for instance CE) waiting for GSI-authenticated (and non) connection

Spark *:*=client application running on each WN (through a cronjob and/or through a normal "grid-job") and contacting tank.

Rsync server running on another machine (a SE for instance) and acting as central repository of the software


At the end of the whole process TANK will e-mail the ESM indicating the result of the installation
arrives on CE
ll WNs or
the time External
cked. Special
te policies can be taken into account. Local installation on WNs is triggered. No authentication is required : each WN trusts TANK.
6
TANK
sy
dire
central repository
ESM
up on WN
SPA
SECURITY
prog
elegated
credential o
checked in TANK.
SPARK ask
g registration in
TANK centr
3
2
4
5
The software is
alled locally. A pre-validation step is then executed.
WN
WN
WN
WN
WN
WN
WN
Site Firewall

Bibliography:

1. F.Donno et al.: "Experiment Software Installation on LCG-1" (https://edms.cern.ch/file/412781//SoftwareInstallation.pdf)
2. R.Santinelli et al."The LCG-2 Software Installation Tool "(https://edms.cern.ch/file/412781/SoftwareInstallation.ps)
3. R. Santinelli et al. "Experience Software Installation Experience in LCG-2"

   (http://indico.cern.ch/getFile.py/access?contribId=116&sessionId=23&resId=0&materialId=paper&confId=0)
4. GridApplicationGroup "Requirements on software installation" (http://cern.ch/fca/SoftInst.pdf)
5. P.Buncic "gLITE Package Manager" (http://agenda.cern.ch/askArchive.php?base=agenda&categ=a043837&id=a043837s1t0/transparencies)

Hands-on

Hands on

Exercise 1.

The package of the software is a empty file (“example_file”)
The **install_tool** script does nothing.

Under the root installation directory the installation script (**install_sw**) creates a directory named <NameOfTheDayOfWeek>-date and copies on this directory the file “*example_file*”.

The **validation_sw** tool checks for the existence of this file and otherwise performs the installation.

The run script **run_sw** must be included in the InputSandbox. It checks for the existences of these files and returns the name of the host. Otherwise it returns a non-zero code.

The **uninstall_sw** script removes the <NameOfTheDayOfWeek>-date directory.

Hands on

We invite you to:

1. Create a tarball taking into account these values:

   name of software =tutorial

   version =1.0.0

   release =1

   number_of_tarball =1

   It must contain the install_sw and install_tool and validation_sw scripts and the software package.
2. Build a JDL (tutorial.jdl) which installs and validates at the same time this package (--validate option) for the VO you belong (gilda) on the CE you choose
3. See the list of flags published for your VO on this site through lcg-ManageVOTag after this operation.
4. Create a JDL (tutorial_run.jdl) which runs exactly the software previously installed. The run_sw must be included in the InputSandBox field
5. Create a JDL (tutorial_uninstall.jdl) which removes this package from the CE. It includes in the InputSandbox the script uninstall_sw
6. Check (after 5) the list of flags published by the CE’s Information System.

Hands on

Exercise 2.

The software package is a tarball (user.tar.gz) containing several files with some C++ code and a Makefile to compile them.
The **install_tool** script does nothing**.**

The installation script (**install_sw**) creates under the root installation directory a subdirectory named <NameOfTheDayOfWeek>-date, untars the tarball, runs ‘make’ and copies the executable under the root directory.

The **validation_sw** tool checks for the existence of this file and -otherwise- performs the installation.

The run script **run_my_test,** which must be included in the InputSandbox, checks for the existence of the executable and runs it.

Hands on

We invite you to:

1. Create a tarball taking into account these values :

   name of software =tutorial

   version =2.0.0

   release =1

   number_of_tarball =1

   It must contain the scripts: install_sw, install_tool and validation_sw and the package of the software (user.tar.gz).
2. Build a JDL (tutorial.jdl) which installs and validates at the same time this package (--validate option). The installation is performed on a CE that you choose and the job must publish a **user-defined** flag.
3. See the list of flags published for your VO on this site through lcg-ManageVOTag after this operation.
4. Create a JDL (tutorial_run.jdl) which runs the software installed at the previous step. A **different name** for the run script must be supplied and shipped with the sandbox.