STATUS AND FUTURE PLANS

DDS

Alice Offline week,
2017-03-31

Anar Manafov, Andrey Lebedev,
GSI, Germany

During past 3 Months we were mostly working on:

- house keeping: build system, project environment, external tools (improved code formatting rules).
- Automated tests (DDS Octopus).
- Multiple major improvements of the the Core code (Some of them are still in development but release is imminent).

  This part of our work is not highly visible to end-users.

But there are changes, which users will notice…

TOPOLOGY: Task/Collection Requirements

- Requirements is a way to tell DDS that
  a given task/collection has to be deployed on a specific computing node.

Old Format:

```
<declrequirement id="requirement1">
   <hostPattern type="hostname" value="*.gsi.de"/>
</declrequirement>
```

New Format:

```
<declrequirement id="requirement1" type="hostname" value="*.gsi.de"/>
<declrequirement id="requirement2" type="gpu.available" value="true"/>
Usage:
<decltask id="task1">
   <requirements>
      <id>requirement1</id>
      <id>requirement2</id>
   </requirements>
</decltask>
```

More info: http://dds.gsi.de/doc/nightly/topology.html

TOPOLOGY: Task Triggers

- A task trigger defines a certain action which is performed whenever a specified condition is fulfilled.
- There will be a list of predefined conditions and actions, which users can combine to achieve a desired result.

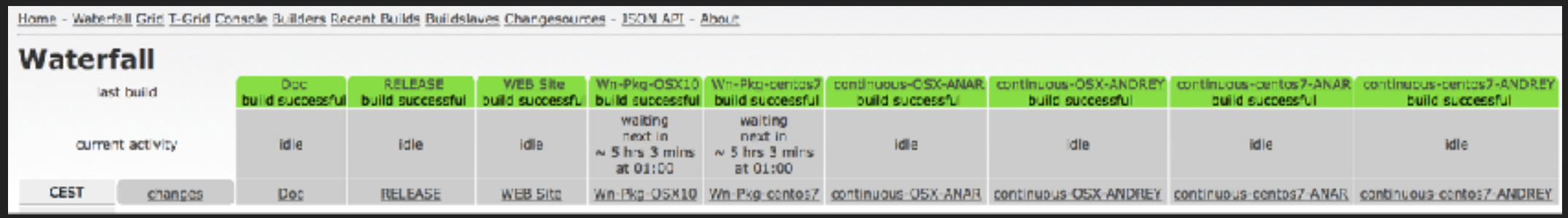
```
<decltrigger id="trigger1" condition="OnTaskStop" action=“RestartTask“ arg =“5“/>
<decltrigger id="trigger2" condition=“OnTaskClean“ action=“StartTaskOnce“ arg =“–clean“/>
```

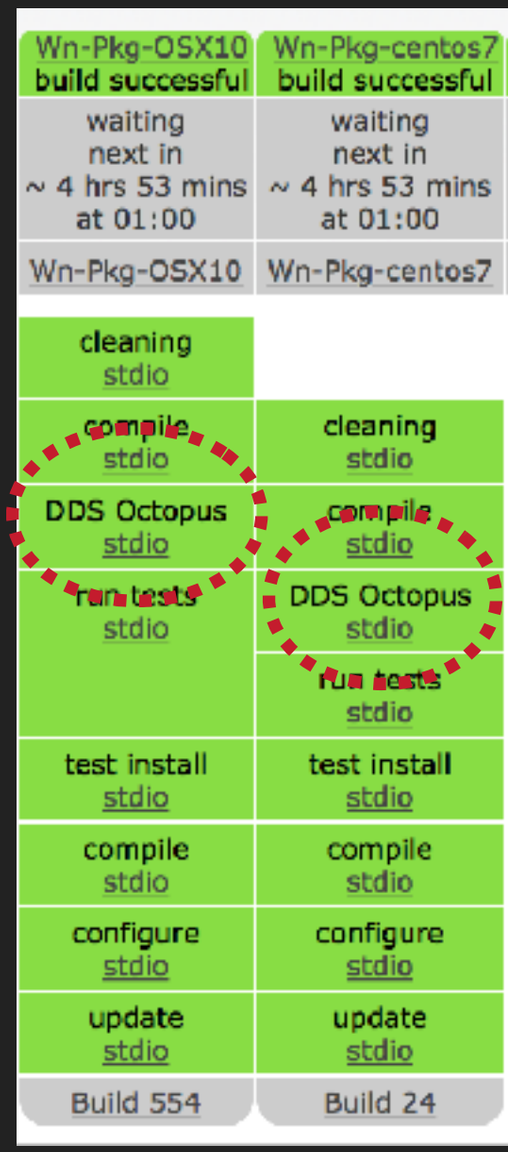
Usage:

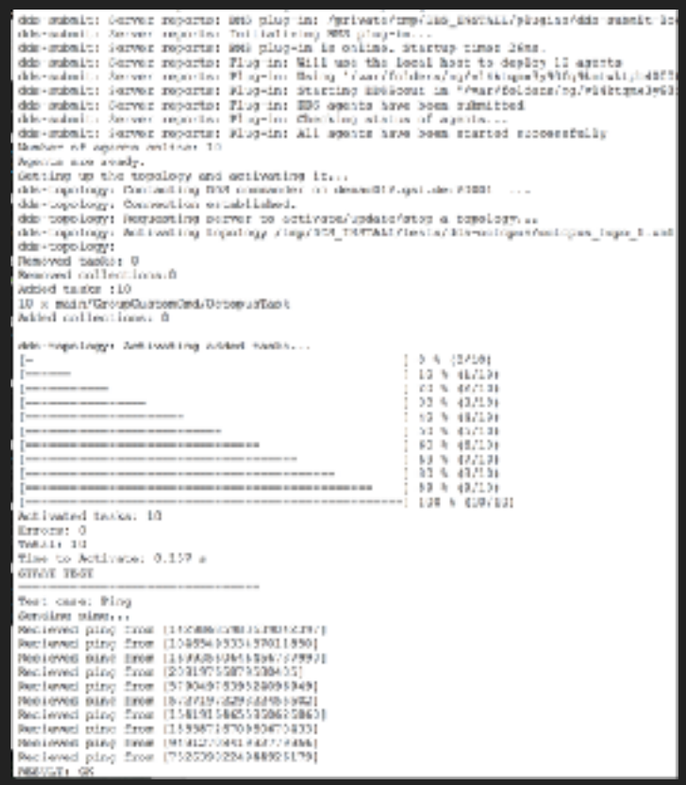
```
<decltask id=“task1">
   <requirements>
      <id>trigger1</id>
      <id>trigger2</id>
   </requirements>
</decltask>
```

More info: http://dds.gsi.de/doc/nightly/topology.html

BUILD SYSTEM







- Improved Buildbot configuration.
- Updated target OS:
  - CentOS 7 (instead of SLC 6),
  - macOS 10.12.3
- 32bit Linux target is dropped.
- **DDS Octopus is the part of builds on all platforms.**

Bundle-like installation & New DDS WN-package dependency look up

Motivation:

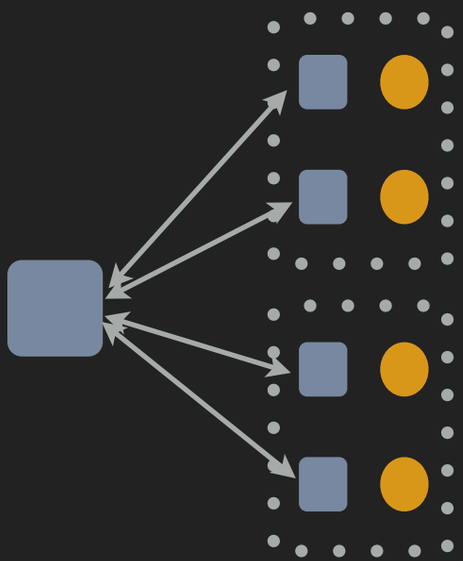
- Users forget to export BOOST libs path,
- Users export wrong BOOST libs path at run-time,
- macOS doesn't export DYLD_LIBRARY_PATH in sub-shell environment.

Solution:

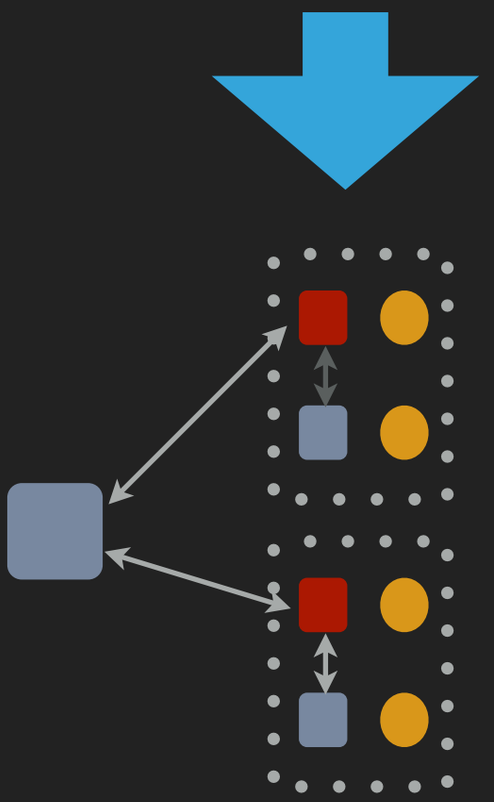
- Our improved cmake configuration automatically enumerates all 3rd-party dependent libs and creates soft links to found ones in the install folder.
- It makes DDS installation self-sufficient and movable.
- It also helps to automate binary distributions (in this case dependent libs will be copied into the dist. package).
- It also helped to simplify building of DDS WN-pakages.

IN DEVELOPMENT: lobby based deployment

- On a given host a random agent will be assigned to act as a lobby leader.
- DDS commander will have one connection per host.



- A lobby leader will act as a dummy proxy services, no special logic will be put on it except key-value propagation inside collections.
- key-value propagation will be either global or local for a collection.



FUTURE PLANS

- Extend RMS plug-in API with task requirements support,
- open topology API,
- add more predefined task triggers,
- finish up lobby based deployment implementation.

- Release - DDS v1.6 (http://dds.gsi.de/download.html),
- DDS Home site: http://dds.gsi.de
- User's Manual & API doc.: http://dds.gsi.de/documentation.html
- Continuous integration: http://demac012.gsi.de:22001/waterfall
- Source Code: https://github.com/FairRootGroup/DDS https://github.com/FairRootGroup/DDS-user-manual https://github.com/FairRootGroup/DDS-web-site