Introduction to DIRAC

Request Management System

Christophe Haen

8th DIRAC User Workshop

23/05/18

Why am I in front of you again?

- **The Request Management System (RMS) has been an increasingly important part of DIRAC**
  - For scalability
  - For fault tolerance
  - For new types of workflows
- **Already great, super easy to make greater in your extension**
  - But do it in DIRAC if it is not too VO specific !

Request Management System

- **Very generic system to handle TODO lists:**
  - A TODO list is created and added to the RMS
  - An Agent takes care of executing them
  - A callback is (sometimes) performed when finished
- **Non mandatory system of DIRAC**
  - But you should definitely use it
- **All the doc is here**

RMS objects

- **Request**
  - TODO list
  - Has a unique ID and a name (not necessarily unique)
  - Has an owner and a group
  - Has an ordered list of Operations

RMS objects

- **Operation**
  - One task on the TODO list
  - Ordered within the Request
  - Has a **Type**
  - Few important attributes:
    - SourceSE, TargetSE
    - Catalog
    - Arguments

RMS objects

- **File**
  - Only for some Operation types
  - No order within the Operation
  - Just an LFN and its metadata
- **Summary:**
  - A Request belongs to someone, and consists of an ordered list of Operations that sometimes apply to Files.

Operation types

- **One python file per Operation Type**
  - Defined in the CS
  - Can be in the extension
- **A fair bunch in DIRAC already**
  - By default in the configuration
  - Mostly DMS related

Operation types

- **ReplicateAndRegister**
  - Dirac or to FTS
- **RemoveFile/RemoveReplica**
- **ForwardDISET**
  - Replay any DISET call
- **…**

Standard use cases for the RMS

- **Failover**
  - Failover for DISET calls (registration of files, job state update, etc)
  - Failover of file transfer if SE not available
  - If Accounting not available, failover to RMS
- **Transformation System**
  - Massive DM operations
- **User Sandbox cleaning**

Exotic LHCb use cases for the RMS

- **Data export from the experiment site**
  - Interface between experiment & the Grid
- **BOINC**
  - Central part of the interaction between the BOINC users and DIRAC services
- **Isolated resources**
  - HPC
- **LFC → DFC migration**

What you need to run the RMS

- **ReqManager**
  - Service interfacing to the DB
- **RequestExecutingAgent (REA)**
  - The Agent executing the Requests
- **CleanReqDBAgent**
  - Cleans the DB from old Requests
- **ReqProxy (optional)**
  - Failover of the Failover

A bit more on the execution

- **REA**
  - Multiprocess
  - Requests from several users in parallel
  - Can be duplicated
- **Objects status**
  - Follow a state machine, driven by File (or Operation if no File)
  - Retry logic
- **Callback when successful**

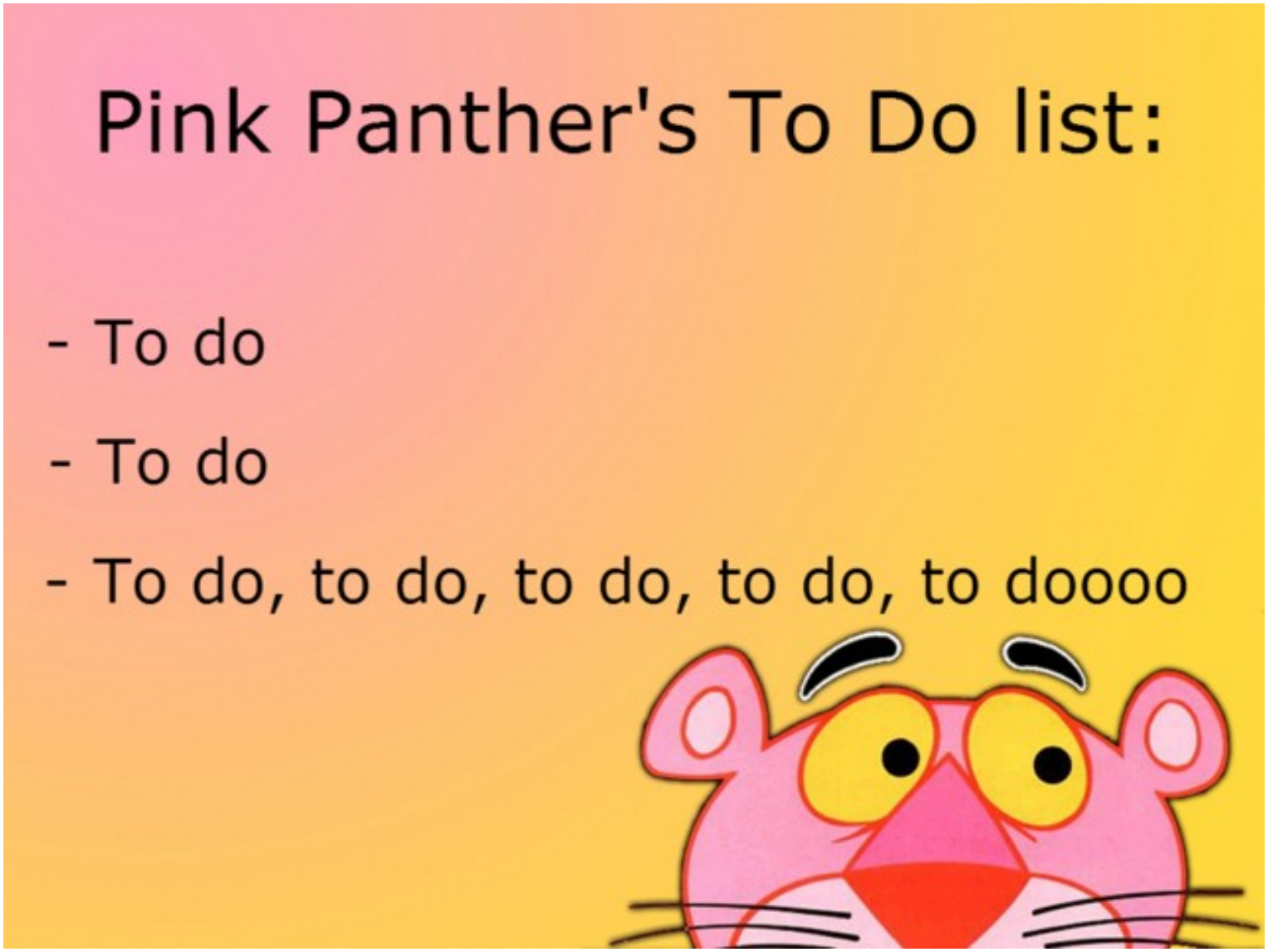
Few more features to come

- **Replace DN with User in the Request**
  - Only exception in DIRAC
- **Possibility to define an ‘Activity’ for a Request**
  - Limit Agent to given activity

Summary

- TODO list & its execution
- Very generic
- Used heavily within DIRAC when available
- Easy to extend
- Exotic use cases made easy
- Key to big scale DataManagement operation

Questions?


Pink Panther's To Do list:
- To do
- To do
- To do, to do, to do, to do, to doooo