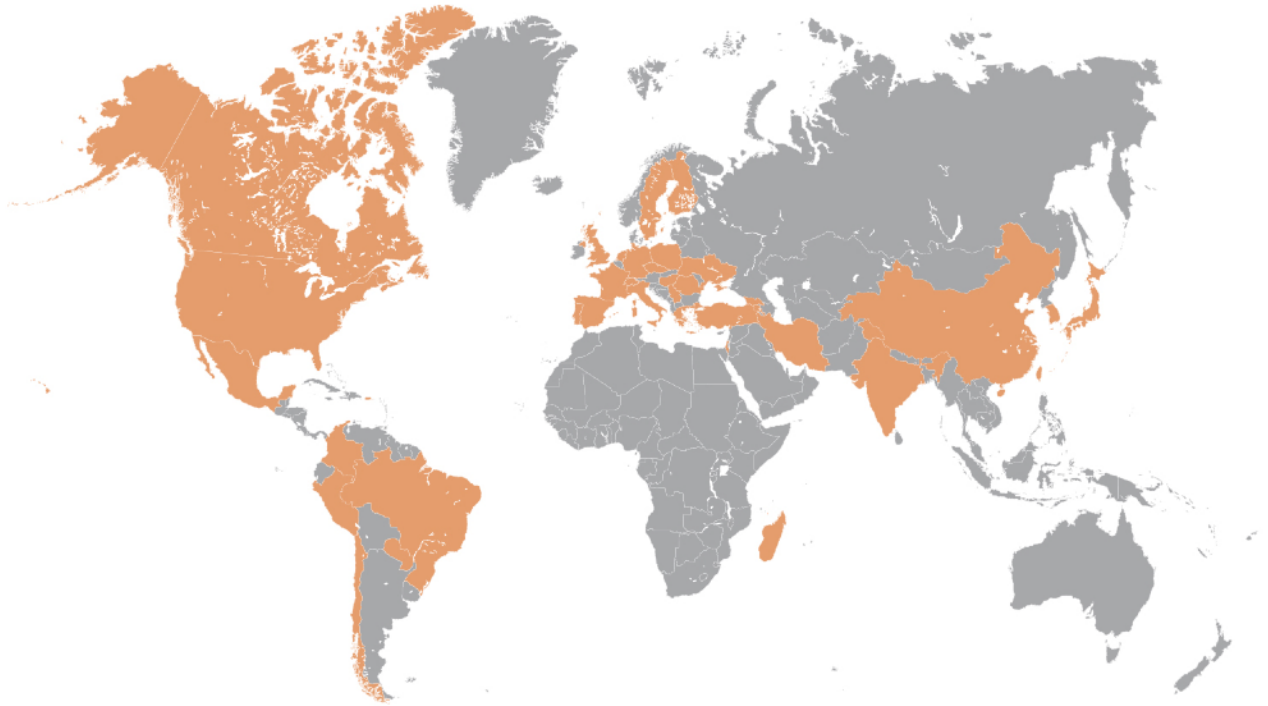
Introduction to justIN



Andrew McNab
University of Manchester

MANCHESTER 1824
The University of Manchester

DUNE

Context: MetaCat and Rucio

MetaCat for DUNE

- MetaCat is a Fermilab product, developed with DUNE's requirements in mind
- It provides us with a file catalogue, with each file associated with its metadata
  - This replaces the metadata aspect of SAM's functionality
  - So we can find the metadata about a file
  - Or find lists of files that have particular metadata values using MetaCat Query Language (MQL)
  - Metadata is usually visible as JSON dictionaries
- MetaCat also allows us to put files into datasets, and apply metadata to datasets
- See https://metacat.readthedocs.io/en/latest/
  - and https://metacat.fnal.gov:9443/dune_meta_prod/app/gui/namespaces?all=yes

MetaCat for DUNE


```
‘hd-protodune:np04hd_raw_run027304_0018_dataflow2_datawriter_0_20240619T165956.hdf5’ :
{‘core.data_stream’: 'physics', 'core.data_tier': 'raw', 'core.end_time': 1718816504.0, 'core.event_count': 35, 'core.events': [2522, 2526, 2530, 2534, 2538, 2542, 2546, 2550, 2554, 2558, 2562, 2566, 2570, 2574, 2578, 2582, 2586, 2590, 2594, 2598, 2602, 2606, 2610, 2614, 2618, 2622, 2626, 2630, 2634, 2638, 2642, 2646, 2650, 2654, 2658], 'core.file_content_status': 'good', 'core.file_format': 'hdf5', 'core.file_type': 'detector', 'core.first_event_number': 2522, 'core.last_event_number': 2658, 'core.run_type': 'hd-protodune', 'core.runs': [27304], 'core.runs_subruns': [2730400001], 'core.start_time': 1718816396.0, 'dune.daq_test': False, 'retention.class': 'physics', 'retention.status': 'active'}
```

Rucio for DUNE

- Where MetaCat is a *file* catalogue, Rucio is a *replica* catalogue
  - Rucio keeps track of all the replicas (copies) of each file
  - As with MetaCat, files can be in datasets
- Rucio is used by the data management team to manage storage
  - Can apply rules to files to control how Rucio creates/deletes replicas at different sites
  - Very large movements of data can be triggered to create free space, archive whole datasets to tape etc
- From a user's point of view, Rucio is what you need to use to grab a single file
  - Rucio will find a suitable replica and download it for you from there

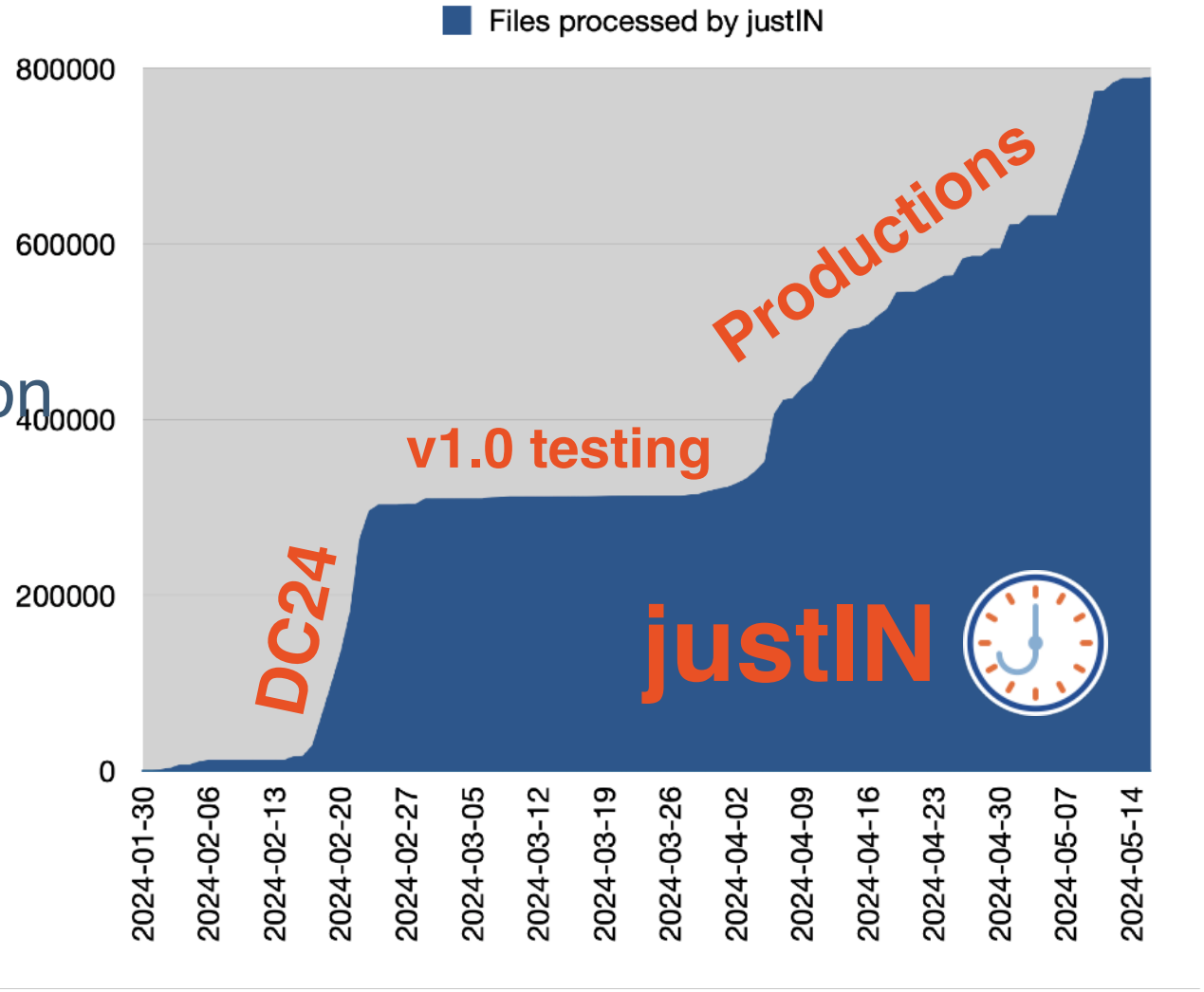
justIN itself

justIN for DUNE

- justIN manages workflows to process data, using info from MetaCat/Rucio
- In short
  - You write a shell jobscript that processes input file(s)
  - You create an MQL MetaCat query that will find the list of input files
  - You pass jobscript and query to justIN to define a workflow, along with your memory, processors, wall clock time requirements
- justIN then
  - Creates jobs to run your jobscript
  - Runs those jobs “near” the input files not yet processed
  - Uploads the outputs of your jobscript to storage

justIN in 2024 so far

- Workflows have been running since Data Challenge 22 in December 2022
- Between the last two collaboration meetings, 790k files had been processed with justIN
- v1.1 is now the basis for central DUNE productions



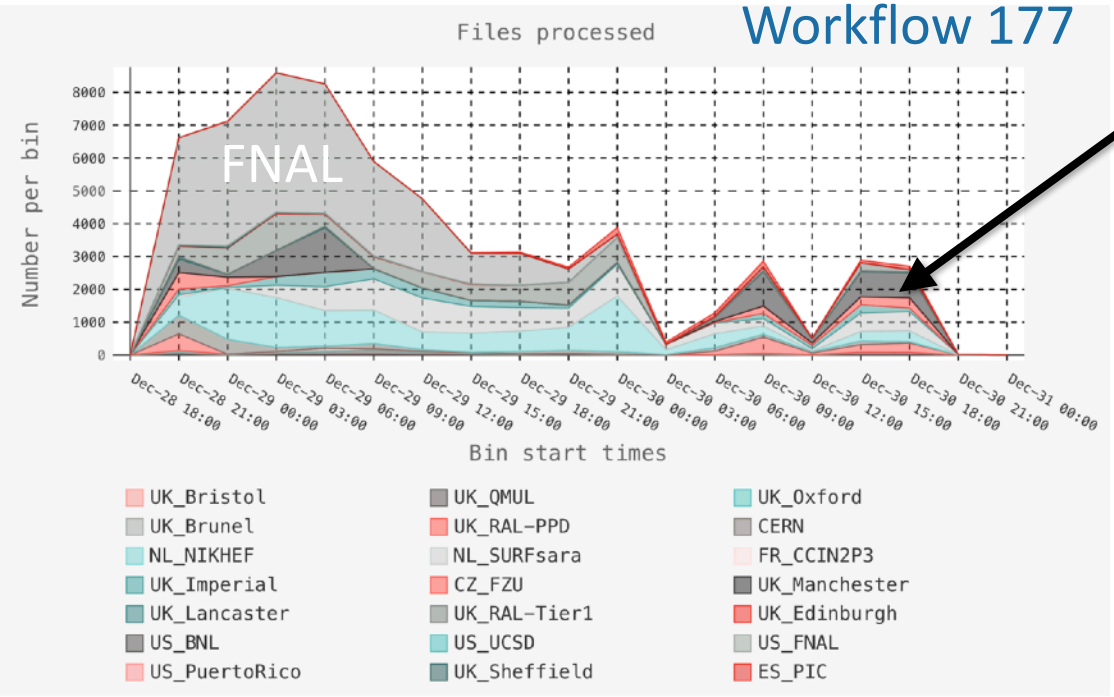
justIN workflows, stages, and jobs

- Workflows consist of one or more stages, with output files from one stage available to the next stage via Rucio-managed storage
- A MetaCat Query Language expression defines the inputs of the first (only?) stage
  - `"files from dune:all where core.run_type='dc4-vd-coldbox-bottom' and dune.campaign='dc4' limit 10"`
- justIN creates HTCondor wrapper jobs for each stage in each workflow
  - They match sites "near" the input files, ranked by suitability by HTCondor ClassAds
- When the jobs start, they run the jobscript for that stage
- The jobscript asks what input files to work on, and leaves outputs on local disk
- The wrapper job does the outputs for you, trying multiple storages if necessary
- If the jobscript or uploads fail, justIN reallocates the inputs to other job(s) up to 6 times

justIN and Apptainer

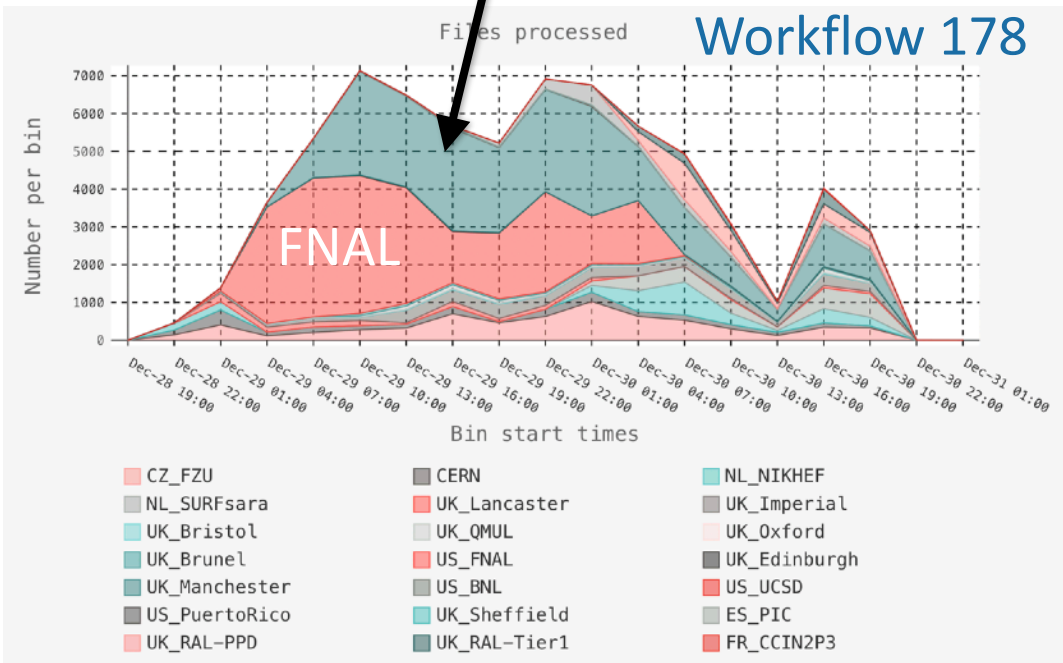
- justIN runs the user's jobscript inside an Apptainer container
- This makes the environment predictable and reproducible
  - There is a `justin-test-jobscript` command which will run your jobscript interactively in the same containers for testing
- justIN uses an operating system image from cvmfs
  - Currently this is set to CentOS 7 for all jobs
  - In a future version it will be possible to choose what OS version at submission time
  - Potentially different OS versions for different stages!
- All the usual cvmfs software repositories are available
  - So at the moment you do UPS setups in your jobscripts; Spack in the future

justIN in DUNE Data Challenge DC4



Not much Manchester

Lots of Manchester!



Job allocations follow the data to suitable sites, using replica locations originally from Rucio

justIN and Monte Carlo workflows

- For Monte Carlo workflows, justIN creates a series of virtual “counter files” in its own database
  - They don’t appear in MetaCat or Rucio or on disk anywhere
- These are used as the input data files for the jobscripts
  - They are numbered sequentially so they could be used for the MC run numbers etc
- These counter files get “used up” each time a job succeeds
  - If a job fails, the counter file is allocated to another job and justIN tries again
- This mechanism could be used for workflows other than MC that also don’t have a series of input data files

GPU support in justIN

- In the next production release of justIN (01.02), basic GPU support is provided
- You can include a `--gpu` option when creating a workflow/stage
- justIN will then include a requirement that **a** GPU is provided to the job by HTCondor/GlideInWMS
- When the wrapper job runs, it uses the Apptainer `--nv` option to make the proprietary NVIDIA libraries and executables available inside the jobscript's container
  - So you can run nvidia-smi for instance
- It's not yet possible to request particular models, GPU memory etc
  - We want some experience with how it's used and what people really need
  - So far only three grid sites are advertising GPUs for us: Manchester, QMUL, Nebraska

justIN at HPC sites, including NERSC

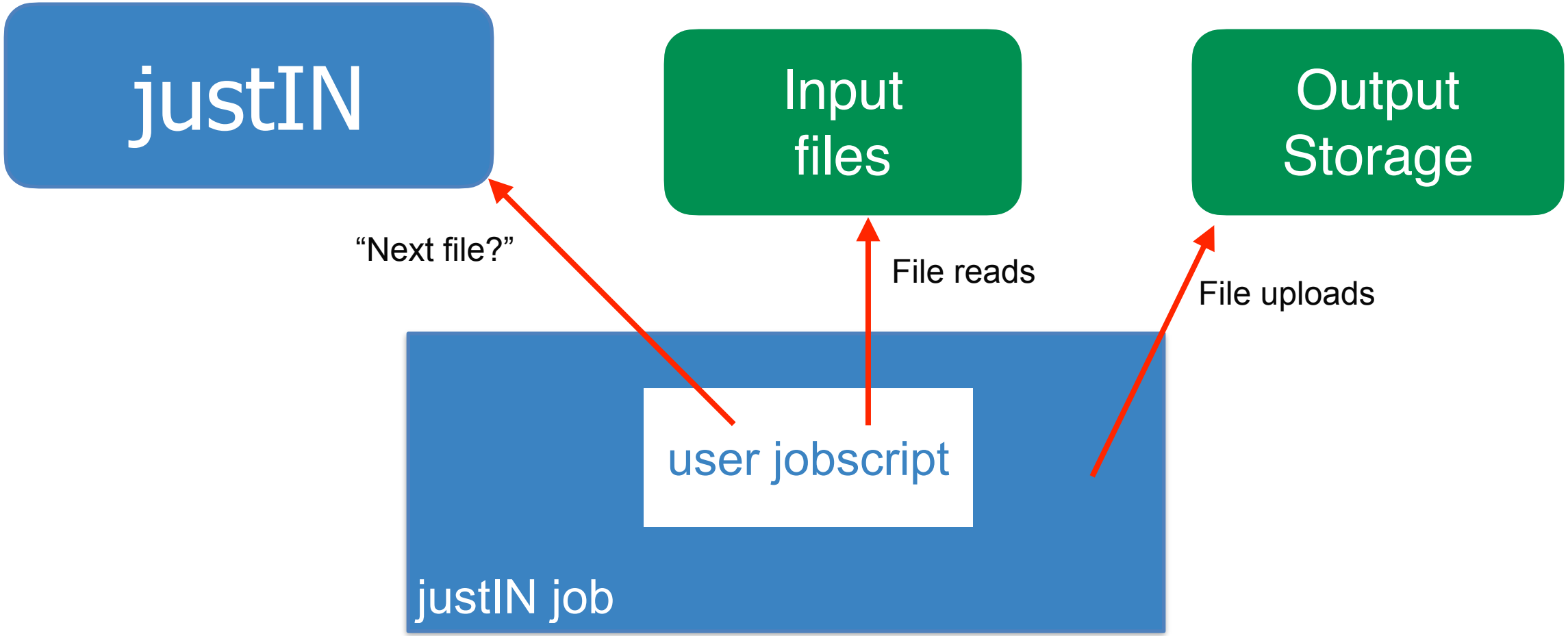
- Support for HPC sites with varying amounts of network connectivity on their workers nodes is part of the long term justIN plan and design
- However, some sites, like NERSC, will be much easier because they can already be added to GlideInWMS/HTCondor pools like the DUNE Global Pool
  - This relies on HEPCloud to extend the DUNE Global Pool there
  - Work is already underway to do this
  - Once this is done, it will be possible to create workflows which require files on NERSC's Rucio storage element and NERSC specific features, and they will match automatically
  - In fact, most the work on justIN's side is in **stopping** this from happening accidentally or unnecessarily

justIN documentation and tutorials

- General docs
  - https://justin.dune.hep.ac.uk/docs/
- Getting started tutorial
  - https://justin.dune.hep.ac.uk/docs/tutorials.dune.md
- Guidance about writing jobscripts, including a checklist of essential points
  - https://justin.dune.hep.ac.uk/docs/jobscripts.md
- Aaron Higuera's Brief Introduction to MetaCat, Rucio, and justIN
  - https://docs.dunescience.org/cgi-bin/sso/ShowDocument?docid=30145

Backup

justIN wrapper jobs



justIN with just-in-time ClassAds