PanDA Rubin Orchestration User Guide

**Editing of this document has been closed. Comments can still be added if necessary. The documentation is being migrated to** [**https://panda.lsst.io/**](https://panda.lsst.io/)

This guide describes basic operations for running Rubin workflow with PanDA on the Google Cloud deployment prepared for the DP0.2 exercise. This setup continuously evolves, this is why the current manual may not always precisely reflect the transitional state. In case of founding any discrepancies please inform the authors using provided Support Channels.

[Setup overview](#_d5yfbc2m8ypy)

[GKE queues](#_k1xp12kt1otp)

[Users authentication](#_tlwjar3842xa)

[How to submit a workflow](#_mxzepqgk8sz8)

[YAML configuration](#_y1cp8984trmk)

[Submission node](#_nbd2uevocaqu)

[IAM user authentication](#_gxv4f0ucy9rg)

[How to monitor workflow](#_vluqqzg8jfvg)

[Workflow progress](#_76bapxxftg8)

[Tasks progress](#_wt95nejaqr4d)

[Jobs progress](#_jup42y32tctg)

[Logs access](#_p8s4rc5oy5w2)

[Real-time logs access](#_t221tr7ksp64)

[Monitor of job resource utilization](#_2a4isn6xgqrb)

[How to debug a workflow](#_ts608i5vat5k)

[Workflow points of inspection](#_i4ytbiytmmmv)

[Tasks retry](#_k69x2x1l31z2)

[Workflow cancel/retry](#_qrvwsbydjadt)

[Support channels](#_7aiq5lihgslt)

# Setup overview

A PanDA setup for the Rubin DP0.2 exercise consists of several components including a test PanDA server instance located at CERN, two Google Kubernetes Engine (GKE) clusters deployed in the Google based Interim Data Facility (IDF), submission node in the Google Cloud and other components. The components overview is presented [here](https://brookhavenlab-my.sharepoint.com/:p:/g/personal/spadolski_bnl_gov/ERnBzu8NO0lHi57ZcS_ESkUBJGl_8qdpKVr4VvG2TICp0A?e=VlXbP2). In this document we will describe only several components from the whole setup (Fig. 1) facing end users.

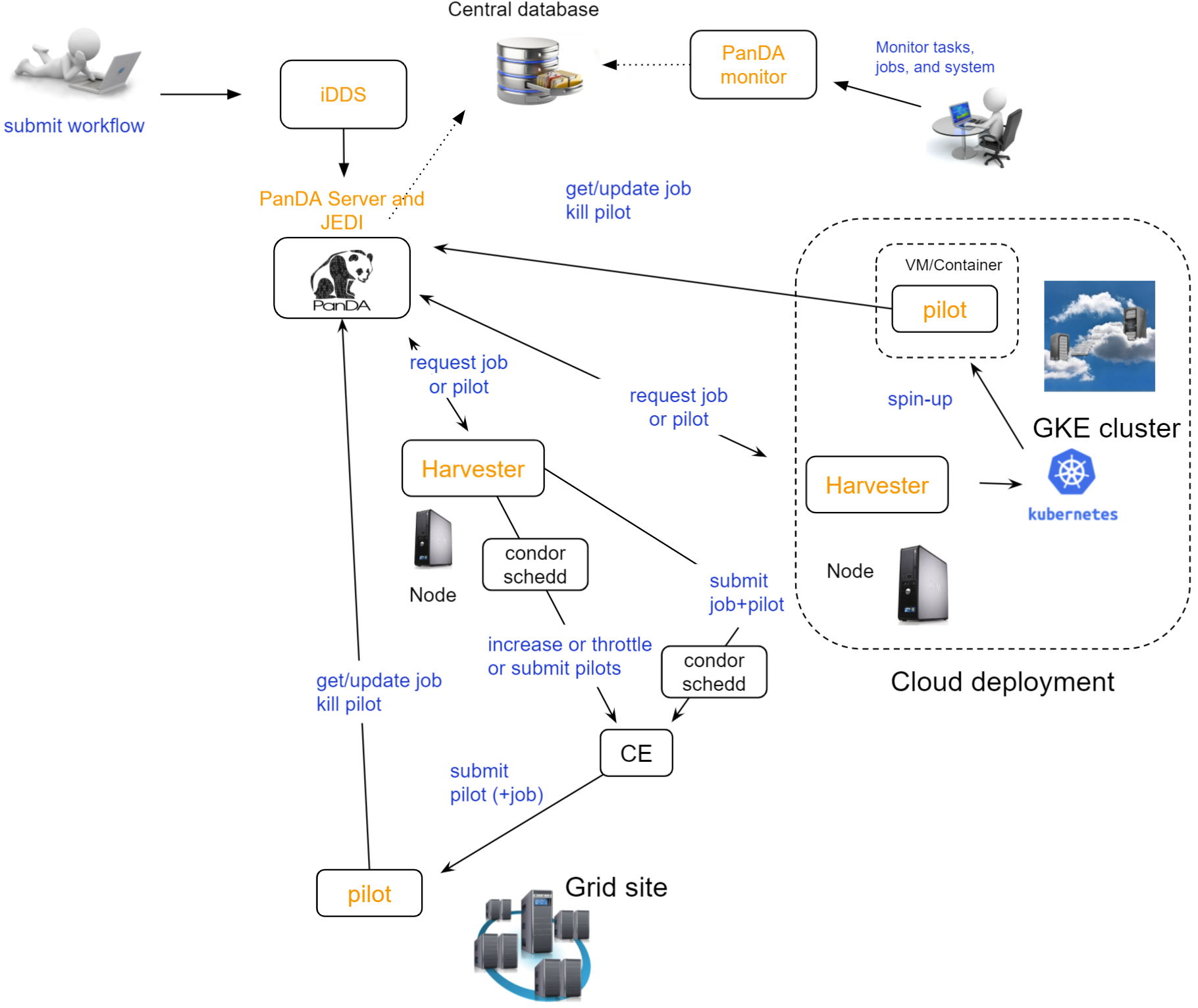


Fig 1. System component diagram

## GKE queues

There are 2 production queues pre configured in the IDF GKE: DOMA\_LSST\_GOOGLE\_TEST and DOMA\_LSST\_GOOGLE\_TEST\_HIMEM. They are dedicated for landing jobs with significantly different requirements to memory. The node used for the DOMA\_LSST\_GOOGLE\_TEST allocates about 4GB of RAM per job and the DOMA\_LSST\_GOOGLE\_TEST\_HIMEM allocates about 14GB of RAM per job.

The PanDA server performs automatic landing tasks in the appropriate queue using the memory requirements information. There are few associated configuration parameters should be defined in the YAML:

computing\_cloud: LSST

pipetask:

measure:

requestMemory: 8129

mergeMeasurements:

requestMemory: 4096

...

The first parameter (computing\_cloud) defines the PanDA cloud associated with the IDF. The requestMemory setting defines the RAM request per task type.

## Users authentication

During the PanDA evaluation procedure we are using the Indigo-IAM (<https://github.com/indigo-iam/iam> ) system to provide users authentication. We set up a dedicated instance of this system available here:

<https://panda-iam-doma.cern.ch/login>

WIth this system a user can create a new PanDA user profile for submission tasks to PanDA. The registration process is starting from the link provided above. Once a registration is approved by the administrator, the user can start submitting tasks. It is up to the user which credential provider to use during registration. It could be an institutional account or general purpose services like Google or Github. The only requirement is that the administrator should know user email used in registration to match a person with a newly created account during approval.

# How to submit a workflow

## YAML configuration

As any other Rubin workflow submitted with BPS commands, PanDA based data processing requires a YAML configuration file. The YAML settings, common for different BPS plugins provided here: <https://pipelines.lsst.io/v/w_2021_24/modules/lsst.ctrl.bps/quickstart.html#defining-a-submission>

Later in this section we focus on PanDA specific and minimal set of the common settings supplied in the YAML with

bps submit <config>.yaml

command. They are:

* maxwalltime: 90000 maximum wall time on the execution node allowed to run a single job in seconds
* maxattempt: 1 number of attempts to successfully execute a job. It is recommended to set this parameter at least to 5 due to preemptions of machines used in the GKE cluster
* whenSaveJobQgraph: "NEVER" this parameter is mandatory because PanDA plugin is currently supports only a single quantum graph file distribution model
* idds\_server: "<https://aipanda015.cern.ch:443/idds>" this is the URL of the iDDS server used for the workflow orchestration
* sw\_image: "spodolsky/centos:7-stack-lsst\_distrib-d\_2021\_08\_11" defines the Docker image with the SW distribution to use on the computation nodes
* fileDistributionEndPoint: "s3://butler-us-central1-panda-dev/hsc/{payload\_folder}/{uniqProcName}/" this is bucket name and path to the data used in the workflow
* s3\_endpoint\_url: "<https://storage.googleapis.com>" the address of the object storage server
* payload\_folder: payload name of the folder where the quantum graph file will be stored
* runner\_command. This is the command will be executed in container by the Pilot instance. The ${{IN/L}} expression is the PanDA substitution rule to be used during jobs generation.
* createQuantumGraph: '${CTRL\_MPEXEC\_DIR}/bin/pipetask qgraph -d "{dataQuery}" -b {butlerConfig} -i {inCollection} -p {pipelineYaml} -q {qgraphFile} {pipelineOptions}' this command does not contain any PanDA specific parameters and executes at the submission node on the local installation
* runQuantumCommand: '${CTRL\_MPEXEC\_DIR}/bin/pipetask --long-log run -b {butlerConfig} --output-run {outCollection} --qgraph {fileDistributionEndPoint}/{qgraphFile} --qgraph-id {qgraphId} --qgraph-node-id {qgraphNodeId} --skip-init-writes --extend-run --clobber-outputs --skip-existing' in this command we replace the CTRL\_MPEXEC\_DIR on container\_CTRL\_MPEXEC\_DIR because it will be executed on the computation node in container

After implementing lazy variables there is not container release specific variables in the YAML file.

## Submission node

Due to the network protection rules implemented in IDF, access to the Butler repository and data files located in object storage is allowed only for machines located inside the IDF network perimeter. Therefore workflow generation can not be proceeded on the local machines and require execution of the bps commands on the dedicated submission machine available for remote ssh access as

ssh <username>@35.239.245.173

Currently this access is limited to a small number of users with lsst.cloud accounts.Before attempting to login to this machine one should receive proper access permission writing in the Rubin slack channel #rubinobs-panda.

The current stack of the Rubin SW is installed there under this tree:

/opt/lsst/software/stack/stack\_d\_2021\_08\_11

To initialize all needed environment variables one should call:

source /opt/lsst/software/stack/stack\_d\_2021\_08\_11/loadLSST.bash

setup lsst\_distrib

source /opt/lsst/software/panda\_env.sh

The last line activates PanDA specific variables such as server addresses and authentication pipeline.

Once the environment is activated the workflow could be submitted into the system:

bps submit <configuration.yaml>

In the case of successful workflow generation, users will get a link to authenticate in the system as described in the next section.

## IAM user authentication

After workflow generation the Rubin middleware BPS system will provide the following authentication prompt:

INFO : Please go to https://panda-iam-doma.cern.ch/device?user\_code=OXIIWM and sign in. Waiting until authentication is completed

INFO : Ready to get ID token?

[y/n]

A user should proceed with the provided URL, login into the IAM system with identity provider used for registration in the https://panda-iam-doma.cern.ch and after confirm the payload:

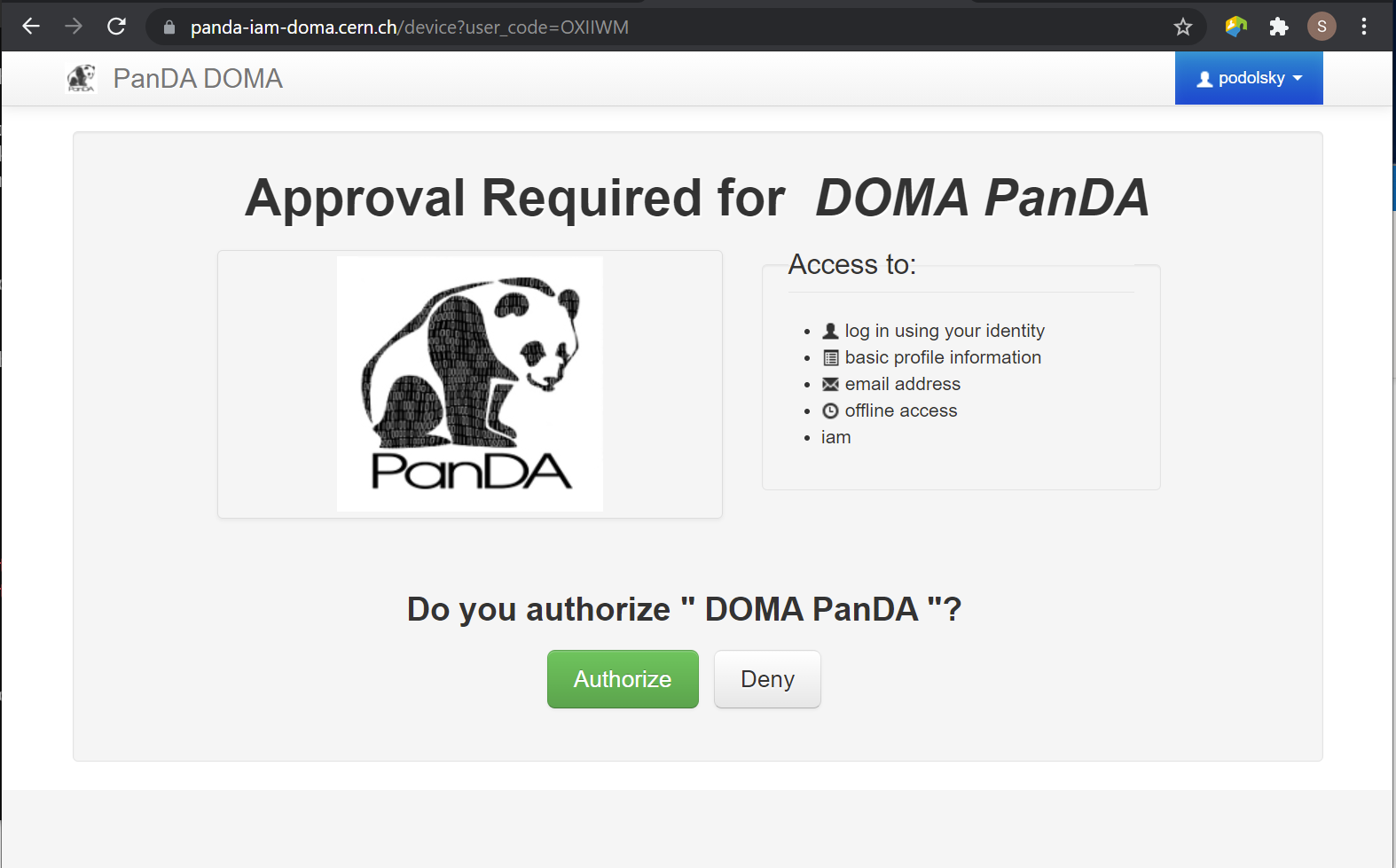


Fig 2. Payload approve screen

After approval, the PanDA client leaves a token in the user home folder and its used for future submissions unless the timeout has expired.

# How to monitor workflow

There are different views provided by PanDA monitor to navigate over the workflow computation progress. The most general view is the workflow progress which shows the processing state for the entire execution graph. The whole workflow is split into tasks that perform the unique kind of data processing against a range of data. This is the example of some tasks in the Rubin workflow: measure, forcedPhotCcd, mergeMeasurements, writeObjectTable, consolidateObjectTable, etc. The smallest current granularity of processing work is the job associated with a particular task which performs processing of a single graph node. One task may hold one of the thousands of jobs doing the same algorithmic operations against different input data. To define the exact location of the data being processed by a job, pseudo input files are used. One pseudo-file name encodes the quantum graph file and the data node id to be processed by a particular job.

The primary monitoring tool used with the test PanDA setup is available on this address:

<https://panda-doma.cern.ch/>

First-time access may require adding this site to the secure exception list, this happens because the site SSL certificate has been signed by the CERN Certification Authority. The inner views of this website require authentication, then Google or GitHub authentication is the easiest way to do this.

## Workflow progress

The workflow summary is available on this address: <https://panda-doma.cern.ch/idds/wfprogress/> .

(Follow instructions on <https://cafiles.cern.ch/cafiles/certificates/list.aspx?ca=grid> and install CERN Grid certification Authority in the browser)

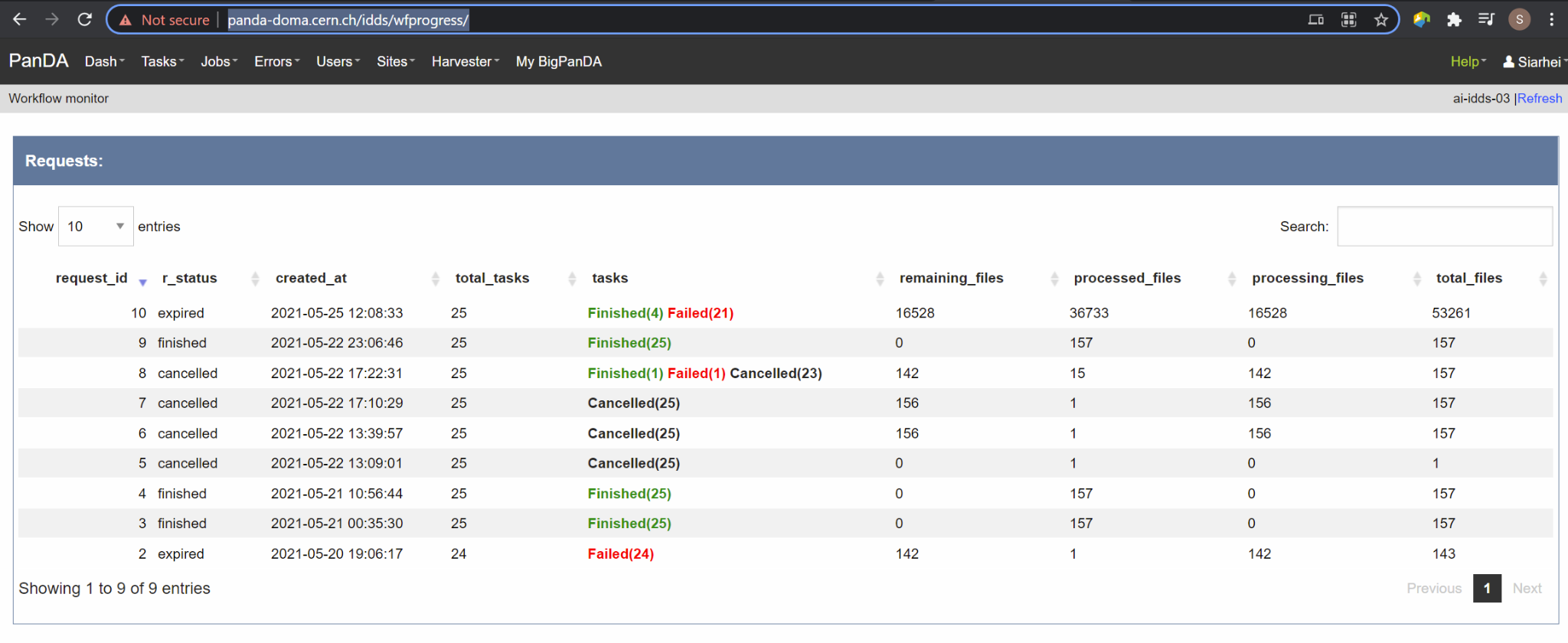


Fig 3. Screenshot of the Workflow progress view

This page provides an overview of the workflow progress:

* requst\_id is the number of the workflow in the iDDS server
* created\_at is the time when the workflow was submitted in the iDDS server. Time provided in the UTC time zone.
* total\_tasks is the number of tasks used for grouping jobs of the same functional role
* tasks column provides link to tasks in different status
* all rest columns provides count of input files in different statuses

Once a new workflow has submitted it can take about 20 minutes to appear in the workflow monitoring

## Tasks progress

Tasks view provides more detailed information about statuses of tasks in the workflow. There are different ways how such a list of tasks could be retrieved. One of the ways is to drill down using the link provided in the WorkFlow progress view described earlier. Another way is to use the workflow name, e.g.:

[https://panda-doma.cern.ch/tasks/?name=shared\_pipecheck\_20210525T115157Z\*](https://panda-doma.cern.ch/tasks/?name=shared_pipecheck_20210525T115157Z*)

This view displays a short summary of tasks, its statuses and progress. For example, a line of the summary table shown in the fig 4.

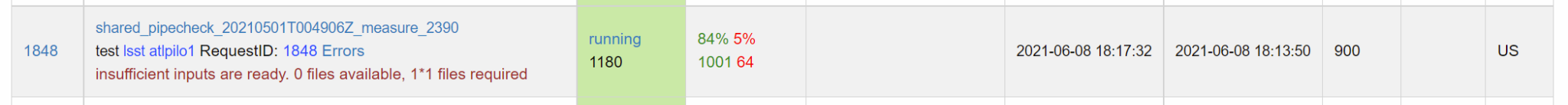


Fig 4. Example of the task summary on the tasks view

In this line the first column is the task id in the PanDA system linked to a task detailed view. The second column provides the task name. There is a message displayed here: “insufficient inputs are ready. 0 files available, 1\*1 files required” this means that not all pseudo inputs (data ids) for this task are released because the previous steps are not yet finished and currently this task has no unprocessed inputs. The third column shows the task status and number of pseudo inputs (data ids) registered for this task. Each data input corresponds to a unique job to be submitted in the computation cluster. In this case the task unites 1180 jobs. The third column shows the overall completion progress (84% or 1001 jobs) and the failure rate (9% or 64 jobs).

Following columns used for the system debug.

## Jobs progress

Clicking on the task id or its name on the tasks view the detailed information is loaded, as shown on the fig. 5:

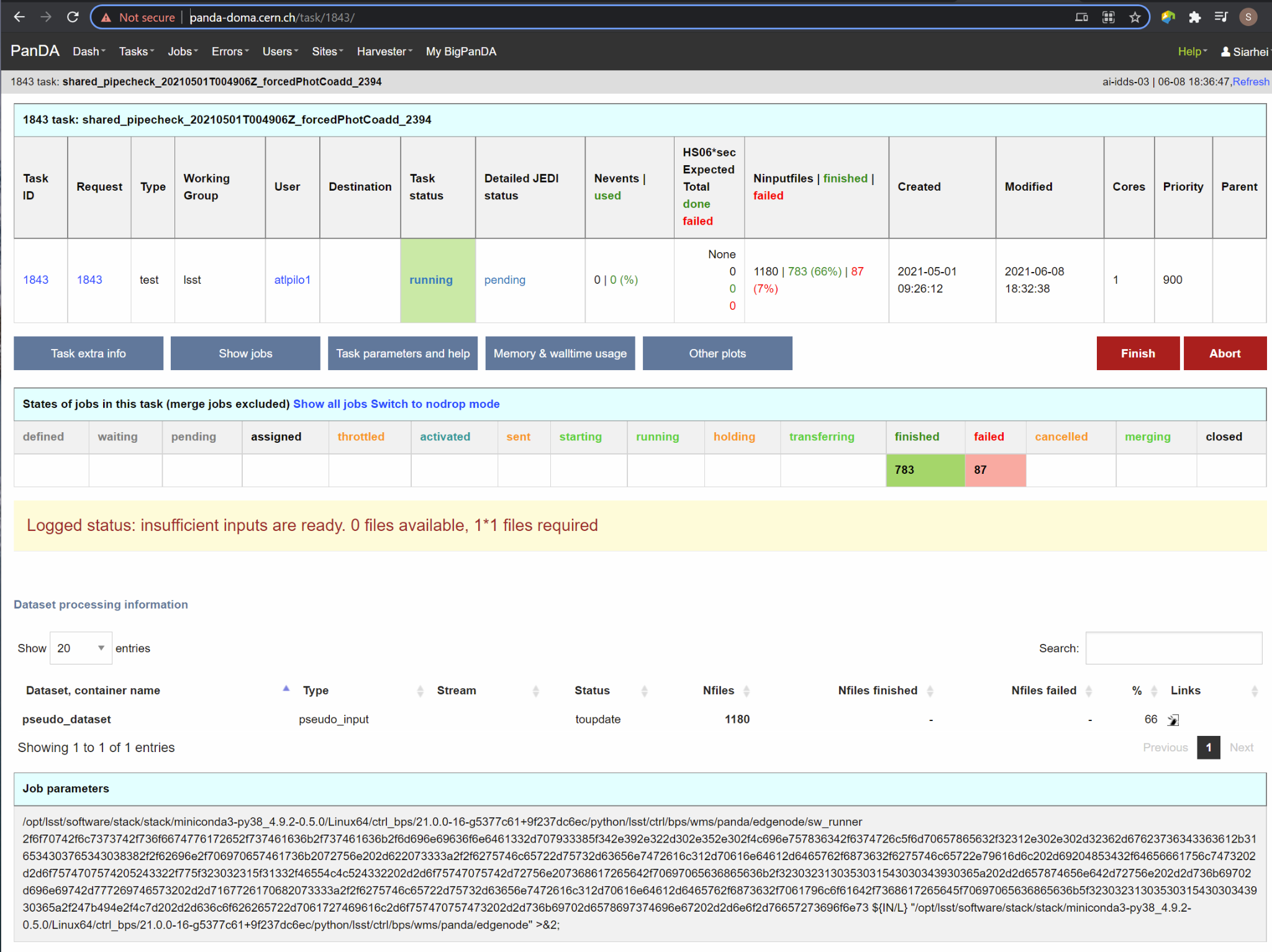


Fig 5. Task details

Here one can see several tables, one of the most important is the jobs summary. In this table all jobs of the task are counted and groped by their statuses. Since PanDA uses late jobs generation, a job is generated only when the next available input is released.

There are two retry filtration modes supported: drop and non drop. They could be switched by clicking the correspondent link in the table head. The drop mode hides all failed jobs which were successfully retried and shows only failures which are hopeless or not yet addressed by the retry module. The drop mode is the default one. The non drop mode shows every failure regardless if they were retried. It could be directly specified in the query URL as follows:

https://panda-doma.cern.ch/task/<taskid>/?mode=nodrop

## Logs access

PanDA monitor provides central access to logs generated by running jobs. A log becomes accessible when a job is in the final state - e.g. finished or failed. In the IDF deployment every log is transferred to the object store and then available for download from there. There are 2 kinds of job logs available: the Rubin software output and the Pilot log which arrange the job run on the computation node.

To access the job log one should load the job details page first. It is accessible as:

https://panda-doma.cern.ch/job/<jobid>/

The job page could be also navigated starting from the task page:

task - > list of jobs in particular state -> job

Once a job page has landed a user should click: Logs -> Pilot job stderr. This will download the Rubin SW output.

## Real-time logs access

The Rubin jobs on the PanDA queues are also provided with (near)real-time logging on Google Cloud Logging. Once the jobs have been running on the PandDA queues, users can check the json format job logs on [the Google Logs Explorer](https://console.cloud.google.com/logs). To access it, you need to login with your Google account of **lsst.cloud**, and select the project of "**panda-dev**" (the full name is panda-dev-1a74).

On the Google Logs Explorer, you make the query. Please include the logName **Panda-RubinLog** in the query:



For specific panda task jobs, you can add one field condition on **jsonPayload.TaskID** in the query, such as:



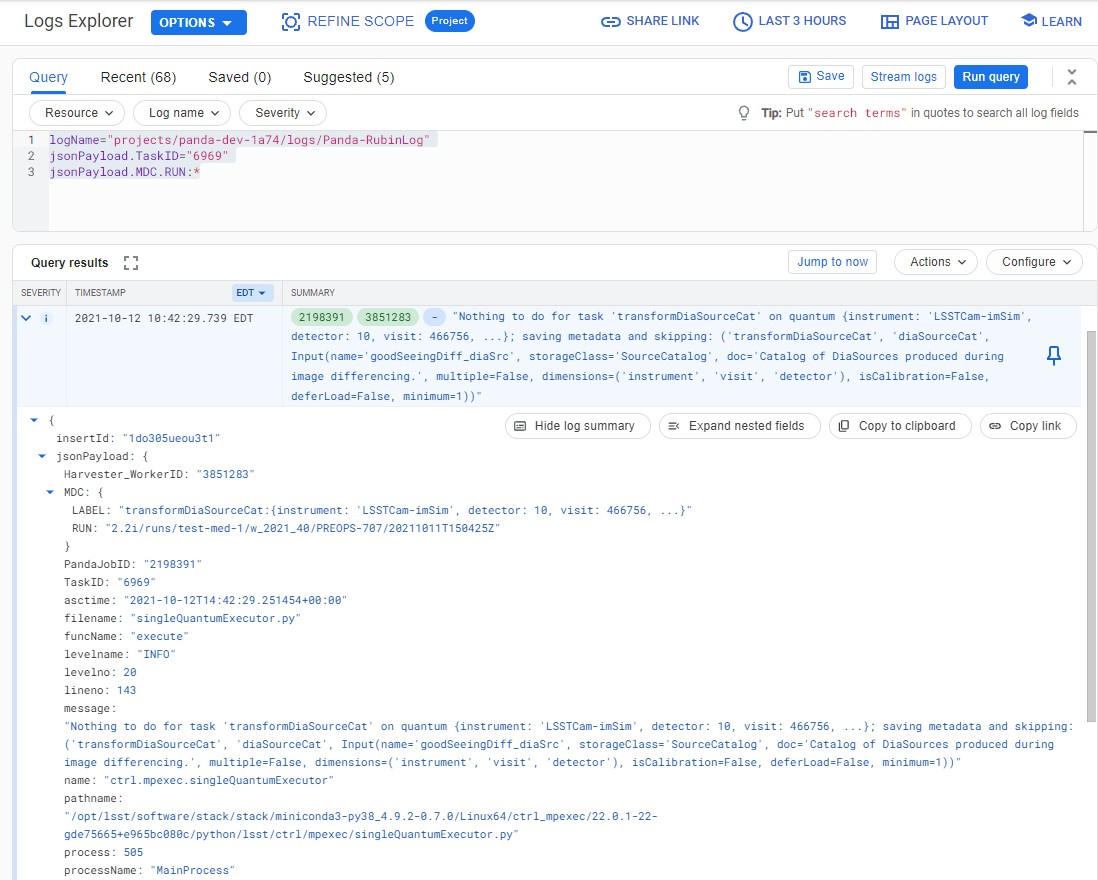
For a specific individual panda job, you can include the field **jsonPayload.PandaJobID**. Or search for a substring "Importing" in the log message:



Or ask for logs containing the field "**MDC.RUN**":



You will get something like:



You can change the time period from the top panel. The default is the last hour. And you can also pull down the **Configure** menu (on the middle right) to change what to be displayed on the Summary column of the query result.

There are more fields available in the query. As you are typing in the query window, it will show up autocomplete field options for you.

You can visit [the page of Advanced logs queries](https://cloud.google.com/logging/docs/view/advanced-queries) for more details on the query syntax.

## Monitor of job resource utilization

For finished and some failed jobs PanDA monitor offers a set of plots with various job metrics collected by the [prmon](https://github.com/HSF/prmon) tool embedded to the middleware container used on IDF. To open that plots user should click on the “Memory and IO plots” button placed on a job view like shown on the fig. 7 and open the popup link.

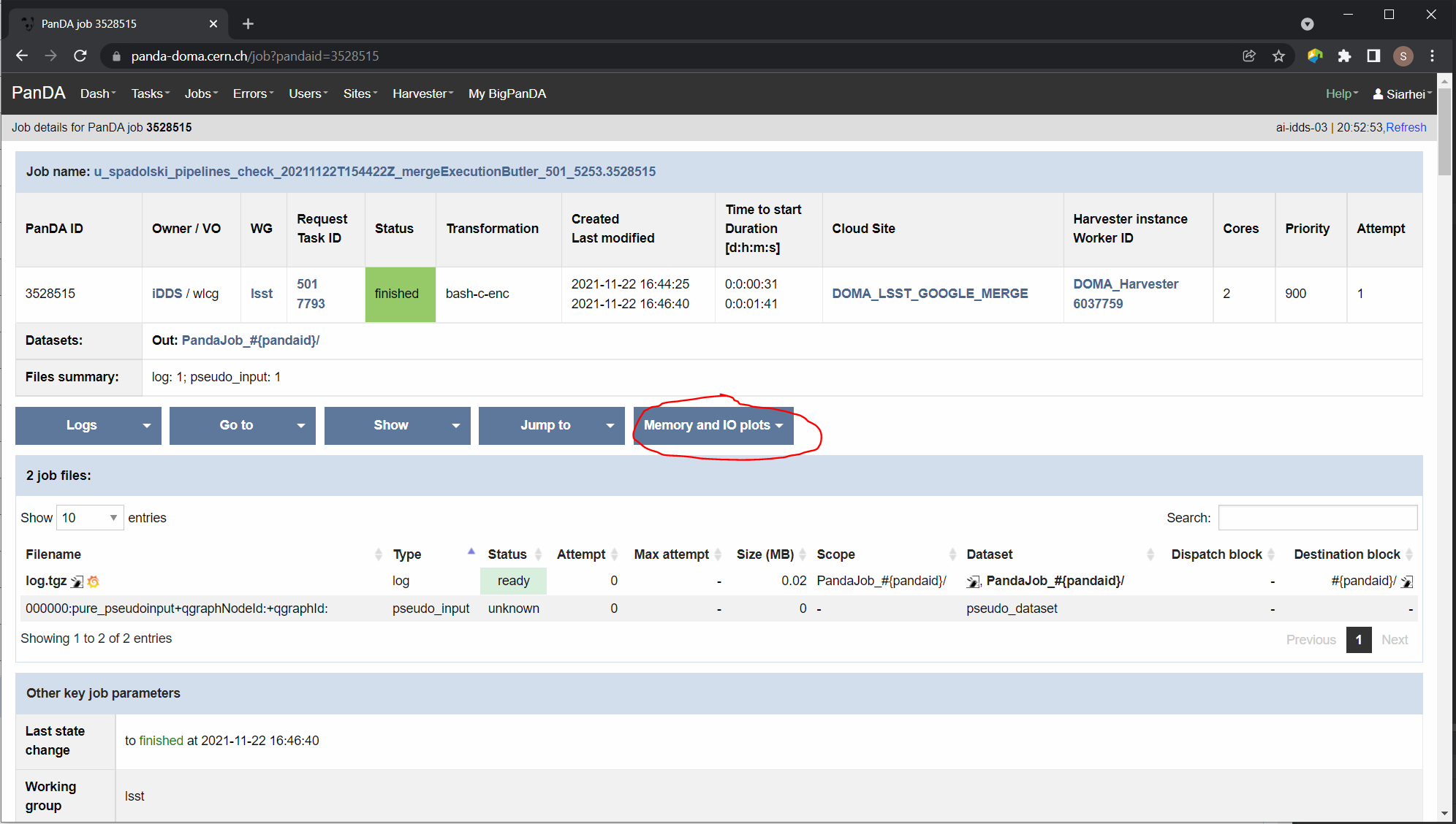


Fig 7. “Memory and IO plots” button

Prmon logs are also available in the textual form. Correspondent links are available in the “Logs” block of the menu.

# How to debug a workflow

## Workflow points of inspection

Different metrics could be inspected to check workflow progress and identify possible issues. There are few of them:

* Is the workflow properly submitted? This could be checked looking into the <https://panda-doma.cern.ch/idds/wfprogress/> table. If the workflow with id provided during submission is in the table, then it went into the iDDS/PanDA systems.
* Are there any failures not related to node preemption? To check this user should list failed jobs and check type of occurred errors:

<https://panda-doma.cern.ch/jobs/?jeditaskid=><task>&jobstatus=failed

## Tasks retry

If a particular task is exhausted in attempts to complete all its jobs, it could be retried. Retrial operation will reinforce to run all uncompleted payload.

The development of PanDA plugin to provide tight integration between BPS and the workflow management system is still in progress, for the DP0.2 period we provide a script which can issue the task retry command:

/opt/lsst/software/retry\_task.py --taskid <taskid>

## Workflow cancel/retry

To abort the entire workflow the following script could be used:

/opt/lsst/software/kill\_workflow.py --workflowid <workflowid>

If there are many tasks in the exhausted state the retry command could be applied to the whole workflow:

/opt/lsst/software/retry\_workflow.py --workflowid <workflowid>

# Support channels

The primary source of support is the Slack channel: #rubinobs-panda.