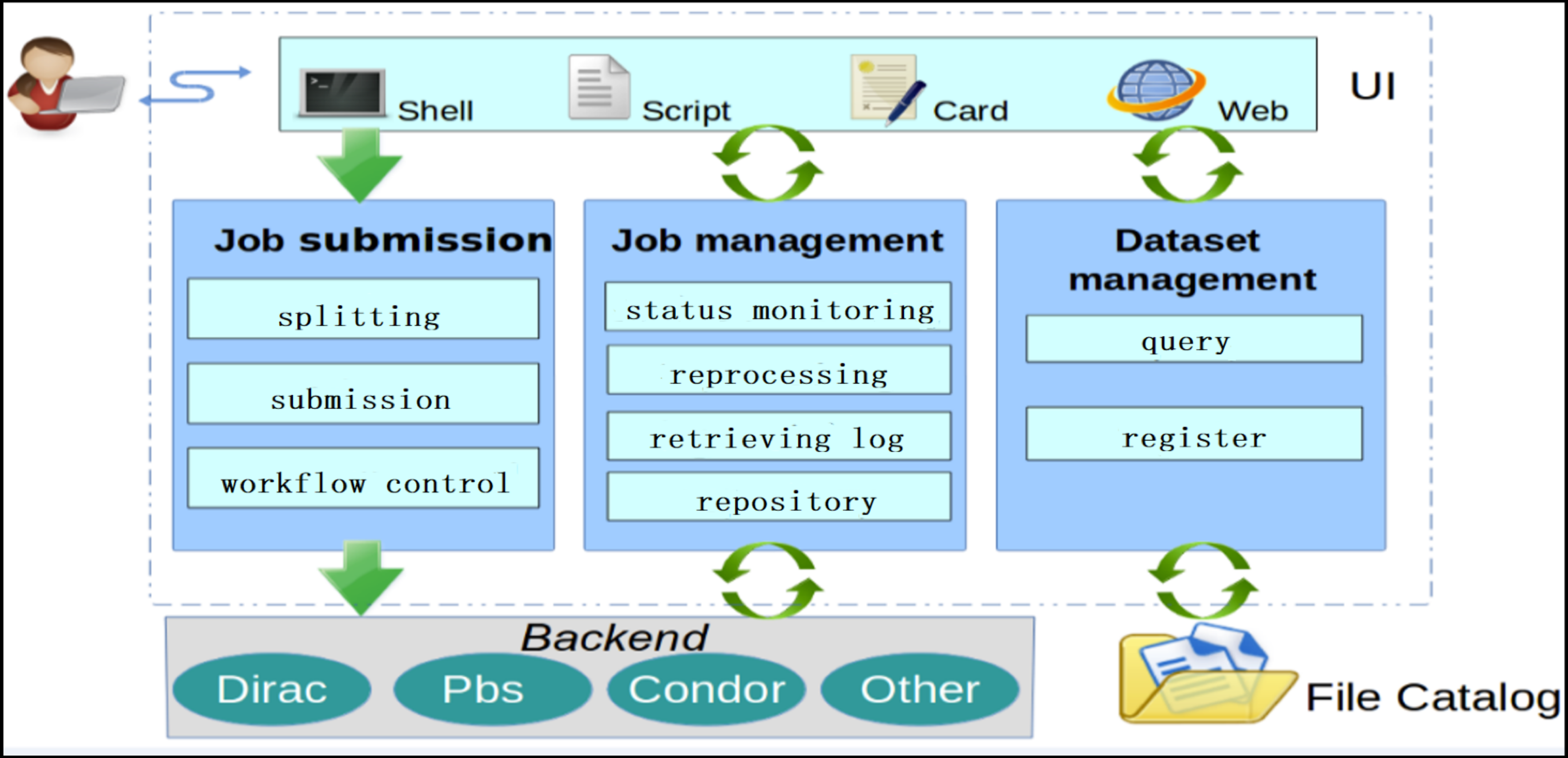
JSUB - A Tool for Job Submission and Management

Yang Yifan, IHEP

JSUB - Job submission utility bundle

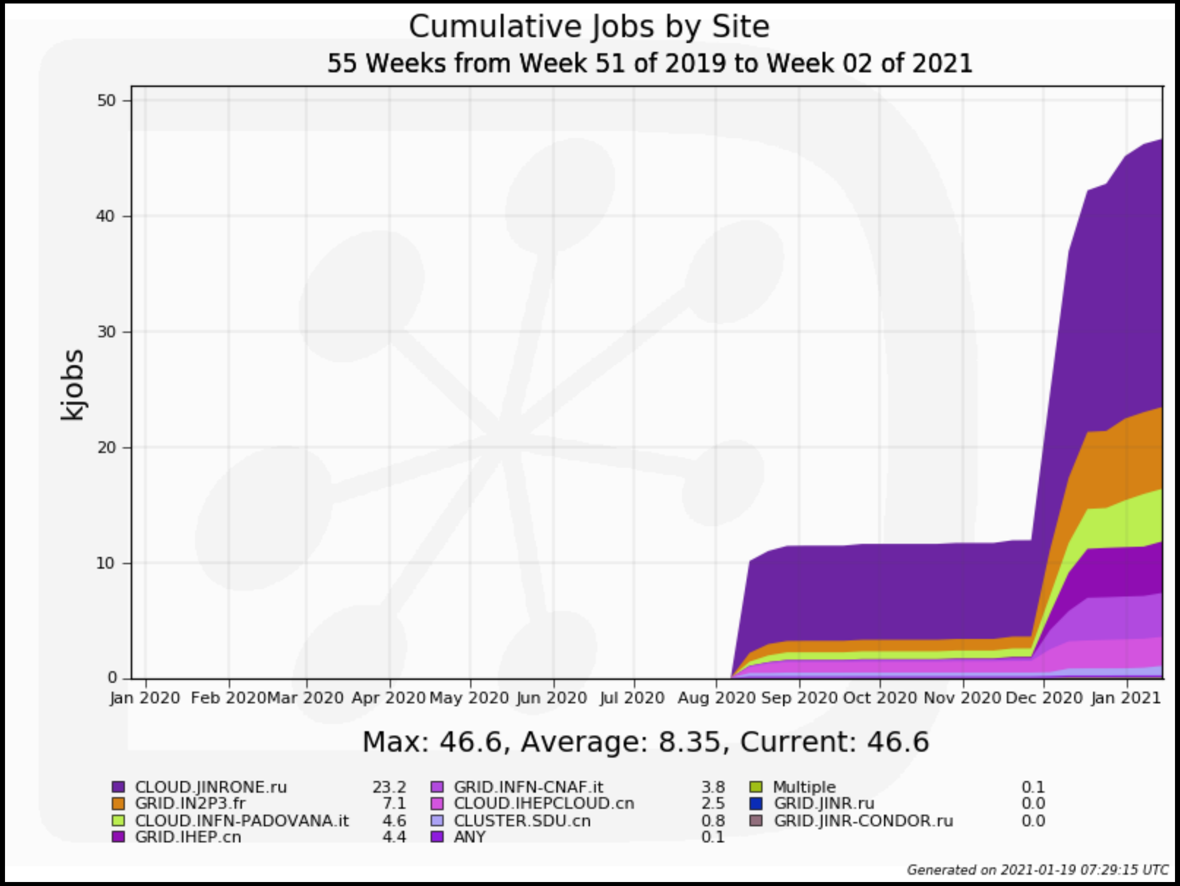
- A frontend software to make user's lives easier
  - Ease the procedure of using DIRAC, and potentially other heterogeneous resources.
  - Automatically manage massive jobs.
  - Extensible for other experiments.
- What JSUB can support for JUNO?
  - User simulation, reconstruction, and analysis.
  - Executing user customized scripts
  - Custom workflow combining multi-steps above.
  - Task-based monitoring and operations

Functionality design of JSUB



Development status

- Basic functionalities has been fully implemented and tested
  - parse of JDL, submission, monitoring, data management and log retrieval
- Performance tests done
  - Fast job management operations
  - Small overhead in terms of memory usage and job running time
- Took part in actual JUNO tasks
  - Tens of thousands of jobs executed.
  - Took part in production of radioactive background and positron samples.
- Inviting more tryings.



A Tutorial on JSUB

- A brief going-through
  - Activation and configuration
  - Defining tasks
  - Job management operations
- Some real task examples

Activation or installation

- Currently, JSUB has been installed on CVMFS. To use the software, a pythonic virtual environment needs to be activated with the following command:

```
source /cvmfs/dcomputing.ihep.ac.cn/frontend/jsub/activate.sh
source .../activate.sh -e juno
source .../activate.sh -v pre1
```

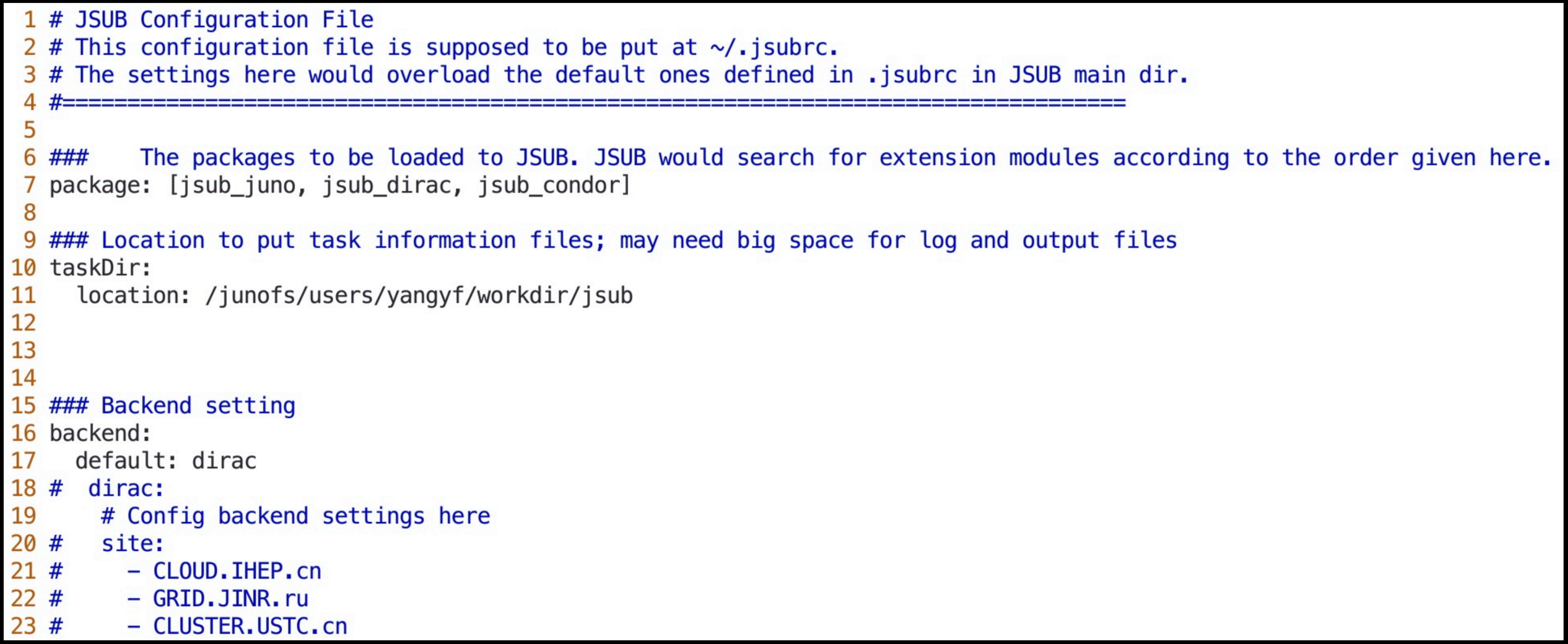
- And the environment can be deactivated with:

```
deactivate
```

- Also, the source code is available on GitHub (https://github.com/jsubpy). The python packages can be installed with pip.

General configuration

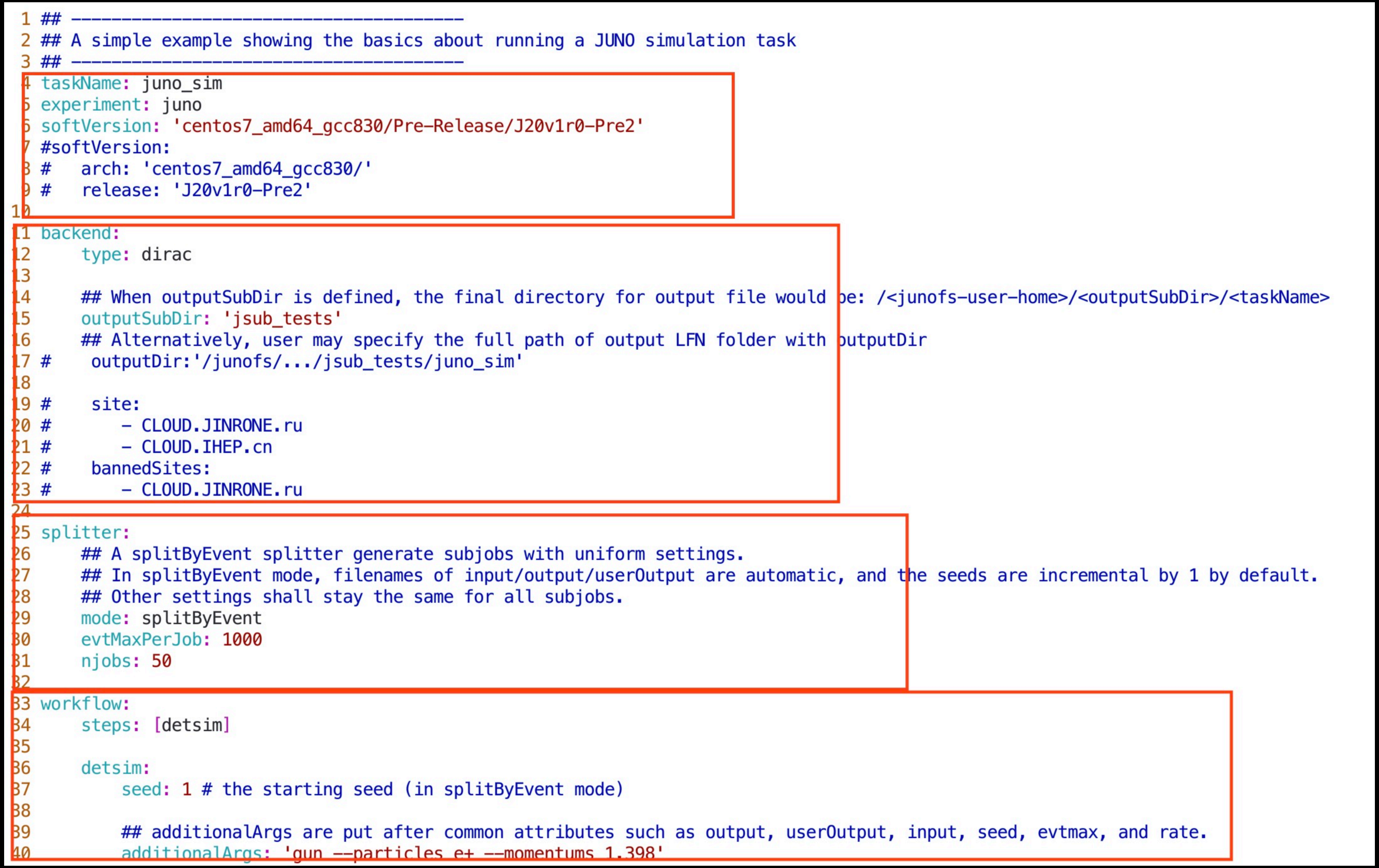
- By default, JSUB loads the configuration file ~/.jsubrc, or from anywhere specified by command line options.
- An example configuration file can be found in /cvmfs/dcomputing.ihep.ac.cn/frontend/jsub/pre1/install/jsub/jsub/support/



```
# JSUB Configuration File
# This configuration file is supposed to be put at ~/.jsubrc.
# The settings here would overload the default ones defined in .jsubrc in JSUB main dir.
#==============================================================================

###    The packages to be loaded to JSUB. JSUB would search for extension modules according to the order given here.
package: [jsub_juno, jsub_dirac, jsub_condor]

### Location to put task information files; may need big space for log and output files
taskDir:
  location: /junofs/users/yangyf/workdir/jsub



### Backend setting
backend:
  default: dirac
#  dirac:
    # Config backend settings here
#    site:
#      - CLOUD.IHEP.cn
#      - GRID.JINR.ru
#      - CLUSTER.USTC.cn
```

Defining tasks

- To define tasks, users need to write task definition files in yaml.
- The settings covered in TDF consist of four parts: general, backend, splitter, and workflow



```yaml
## ---------------------------------------
## A simple example showing the basics about running a JUNO simulation task
## ---------------------------------------
taskName: juno_sim
experiment: juno
softVersion: 'centos7_amd64_gcc830/Pre-Release/J20v1r0-Pre2'
#softVersion:
#   arch: 'centos7_amd64_gcc830/'
#   release: 'J20v1r0-Pre2'

backend:
    type: dirac

    ## When outputSubDir is defined, the final directory for output file would be: /<junofs-user-home>/<outputSubDir>/<taskName>
    outputSubDir: 'jsub_tests'
    ## Alternatively, user may specify the full path of output LFN folder with outputDir
#    outputDir:'/junofs/.../jsub_tests/juno_sim'

#    site:
#       - CLOUD.JINRONE.ru
#       - CLOUD.IHEP.cn
#    bannedSites:
#       - CLOUD.JINRONE.ru

splitter:
    ## A splitByEvent splitter generate subjobs with uniform settings.
    ## In splitByEvent mode, filenames of input/output/userOutput are automatic, and the seeds are incremental by 1 by default.
    ## Other settings shall stay the same for all subjobs.
    mode: splitByEvent
    evtMaxPerJob: 1000
    njobs: 50

workflow:
    steps: [detsim]

    detsim:
        seed: 1 # the starting seed (in splitByEvent mode)

        ## additionalArgs are put after common attributes such as output, userOutput, input, seed, evtmax, and rate.
        additionalArgs: 'gun --particles e+ --momentums 1.398'
```

Defining tasks - general settings

- Basic informations such as task name, experiment, and software version.

```
taskName: juno_sim
experiment: juno
softVersion: 'centos7_amd64_gcc830/Pre-Release/J20v1r0-Pre2'
#softVersion:
#   arch: 'centos7_amd64_gcc830/'
#   release: 'J20v1r0-Pre2'
```

Defining tasks - backend setting

- site, banned sites, job group, output location…

```
backend:
    type: dirac

    ## When outputSubDir is defined, the final directory for output file would be:
    ## /<junofs-user-home>/<outputSubDir>/<taskName>
    outputSubDir: 'jsub_tests'

    ## Alternatively, user may specify the full path of output LFN folder
#    outputDir:'/junofs/.../jsub_tests/juno_sim'

#    jobGroup: ‘jsub.yangyf.test0’
#    site:
#       - CLOUD.JINRONE.ru
#       - CLOUD.IHEP.cn
#    bannedSites:
#       - CLOUD.JINRONE.ru
```

Defining tasks - splitter setting

- A splitter decides how a JSUB task can be splitted into multiple subjobs that each can run on a single backend working node, and how the subjob parameters should be assigned.
- splitByEvent is a splitter suitable for simple tasks with uniform setting other than filenames and random seeds.

```
splitter:
    ## In splitByEvent mode, filenames of input/output/userOutput are automatic,
    ## and the seeds are incremental by 1 by default.
    ## Other settings shall stay the same for all subjobs.
    mode: splitByEvent

    evtMaxPerJob: 1000
    njobs: 50
```

Defining tasks - workflow setting

- Workflow settings describe the list of steps and their parameters

```
workflow:
    steps: [step0]

    step0:
        type: detsim
        seed: 1 # the starting seed (in splitByEvent mode)

        ## additionalArgs are put after common attributes such as
        ## output, userOutput, input, seed, evtmax, and rate.
        additionalArgs: 'gun --particles e+ --momentums 1.398'
```

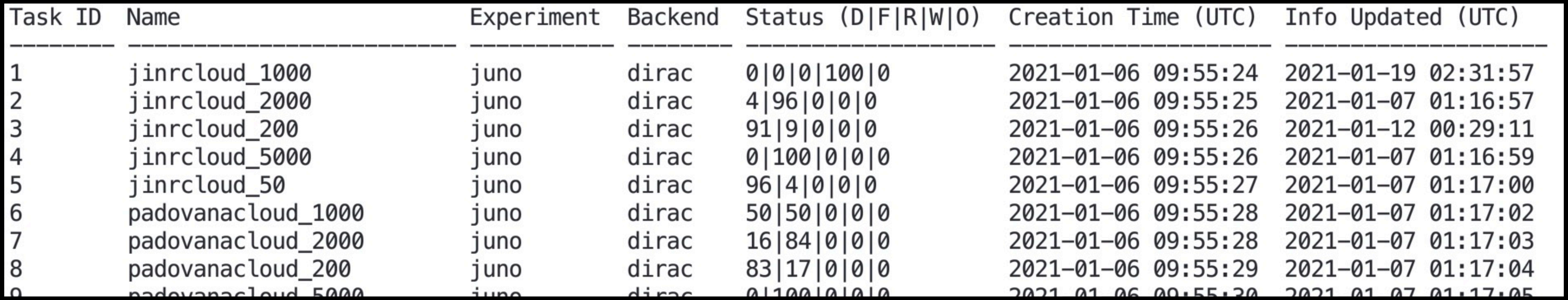
Job management

- Given a TDF, users can create a JSUB task with create command:

jsub create <TDF-filename>

- After successful creation, tasks would be registered into a database saved in configurated path. Users may look up the information of these tasks with ls command:

jsub ls



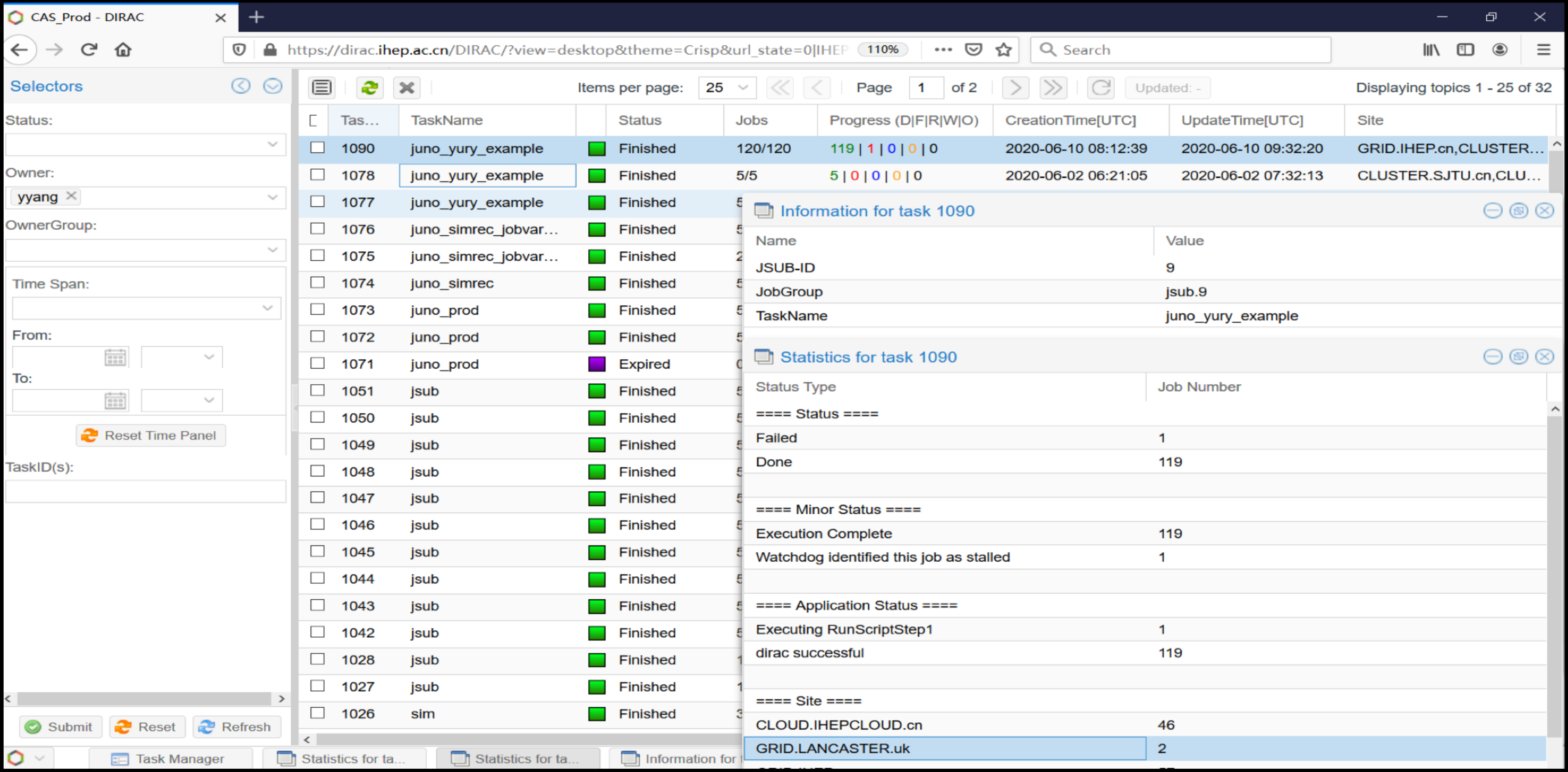
```
Task ID  Name                       Experiment  Backend  Status (D|F|R|W|O)  Creation Time (UTC)  Info Updated (UTC)
-------- -------------------------  ----------  -------- ------------------  -------------------  -------------------
1        jinrcloud_1000             juno        dirac    0|0|0|100|0         2021-01-06 09:55:24  2021-01-19 02:31:57
2        jinrcloud_2000             juno        dirac    4|96|0|0|0          2021-01-06 09:55:25  2021-01-07 01:16:57
3        jinrcloud_200              juno        dirac    91|9|0|0|0          2021-01-06 09:55:26  2021-01-12 00:29:11
4        jinrcloud_5000             juno        dirac    0|100|0|0|0         2021-01-06 09:55:26  2021-01-07 01:16:59
5        jinrcloud_50               juno        dirac    96|4|0|0|0          2021-01-06 09:55:27  2021-01-07 01:17:00
6        padovanacloud_1000         juno        dirac    50|50|0|0|0         2021-01-06 09:55:28  2021-01-07 01:17:02
7        padovanacloud_2000         juno        dirac    16|84|0|0|0         2021-01-06 09:55:28  2021-01-07 01:17:03
8        padovanacloud_200          juno        dirac    83|17|0|0|0         2021-01-06 09:55:29  2021-01-07 01:17:04
9        padovanacloud_5000         juno        dirac    0|100|0|0|0         2021-01-06 09:55:30  2021-01-07 01:17:05
```

- Tasks need to be submitted to backend for running.

jsub submit <task-ID>

Job management

- dirac.ihep.ac.cn can be used for status monitoring of submitted jobs.



Items per page: 25 Page 1 of 2 Updated: - Displaying topics 1 - 25 of 32

Tas...	TaskName		Status	Jobs	Progress (D|F|R|W|O)	CreationTime[UTC]	UpdateTime[UTC]	Site
1090	juno_yury_example	🟩	Finished	120/120	119 | 1 | 0 | 0 | 0	2020-06-10 08:12:39	2020-06-10 09:32:20	GRID.IHEP.cn,CLUSTER...
1078	juno_yury_example	🟩	Finished	5/5	5 | 0 | 0 | 0 | 0	2020-06-02 06:21:05	2020-06-02 07:32:13	CLUSTER.SJTU.cn,CLU...
1077	juno_yury_example	🟩	Finished	5				
1076	juno_simrec_jobvar...	🟩	Finished	5				
1075	juno_simrec_jobvar...	🟩	Finished	2				
1074	juno_simrec	🟩	Finished	5				
1073	juno_prod	🟩	Finished	5				
1072	juno_prod	🟩	Finished	5				
1071	juno_prod	🟪	Expired	0				
1051	jsub	🟩	Finished	5				
1050	jsub	🟩	Finished	5				
1049	jsub	🟩	Finished	5				
1048	jsub	🟩	Finished	5				
1047	jsub	🟩	Finished	5				
1046	jsub	🟩	Finished	5				
1045	jsub	🟩	Finished	5				
1044	jsub	🟩	Finished	5				
1043	jsub	🟩	Finished	5				
1042	jsub	🟩	Finished	5				
1028	jsub	🟩	Finished	1				
1027	jsub	🟩	Finished	1				
1026	sim	🟩	Finished	3				

Information for task 1090

Name	Value
JSUB-ID	9
JobGroup	jsub.9
TaskName	juno_yury_example

Statistics for task 1090

Status Type	Job Number
==== Status ====	
Failed	1
Done	119
==== Minor Status ====	
Execution Complete	119
Watchdog identified this job as stalled	1
==== Application Status ====	
Executing RunScriptStep1	1
dirac successful	119
==== Site ====	
CLOUD.IHEPCLOUD.cn	46
GRID.LANCASTER.uk	2

Job management

- As the final step of successful job execution, the output data would be transferred to target SE. These files can be found with dirac file catalog, or in some certain folders for specific SEs.

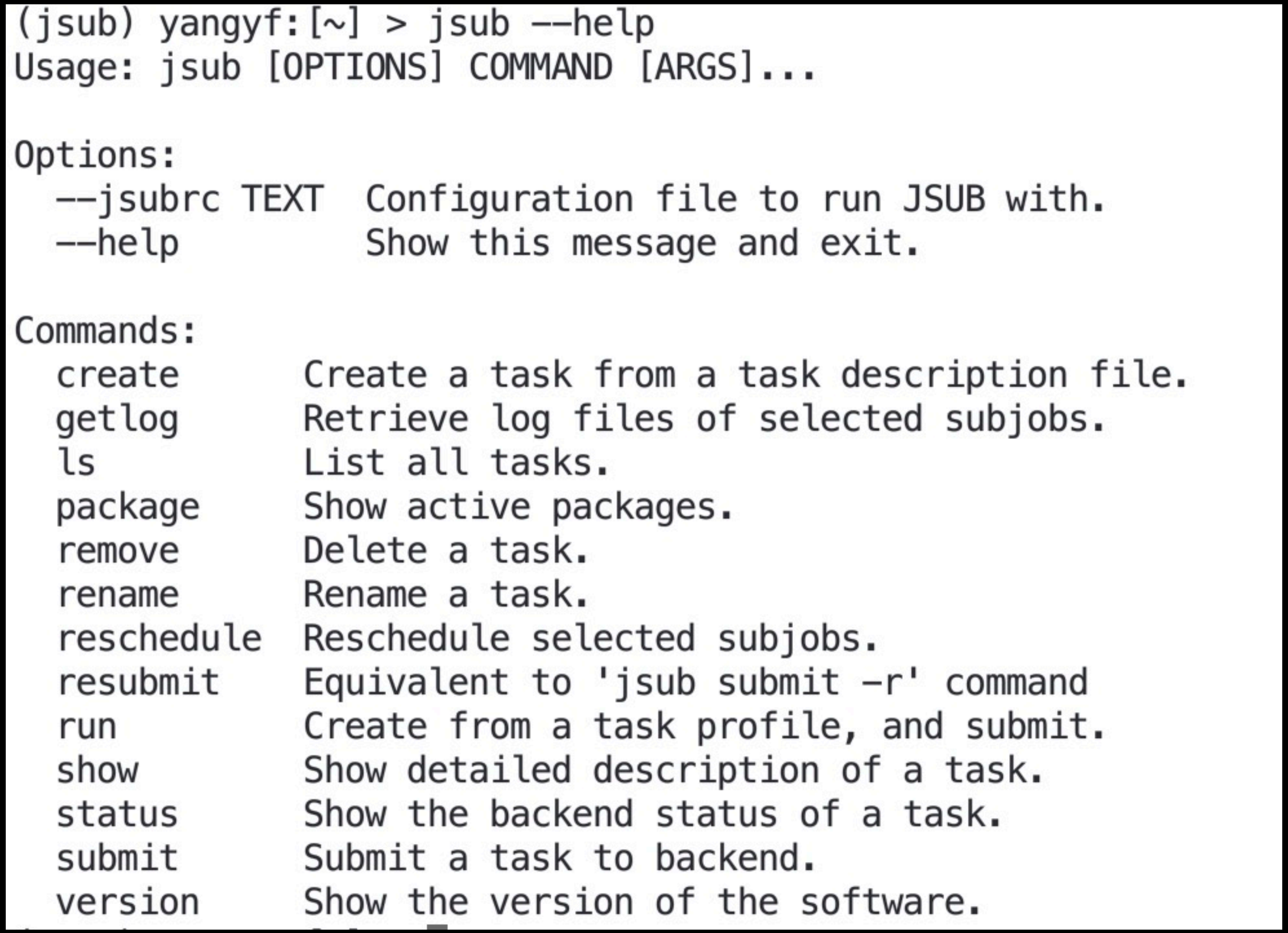
- The log files can be retrieved with getlog command:

```
jsub getlog <task-id> [-s STATUS] [-i subjob_ids]
```

And the downloaded log files can be found in JSUB task dir defined in configuration.

Job Management

- Use command helps to check the list of commands and their usage.

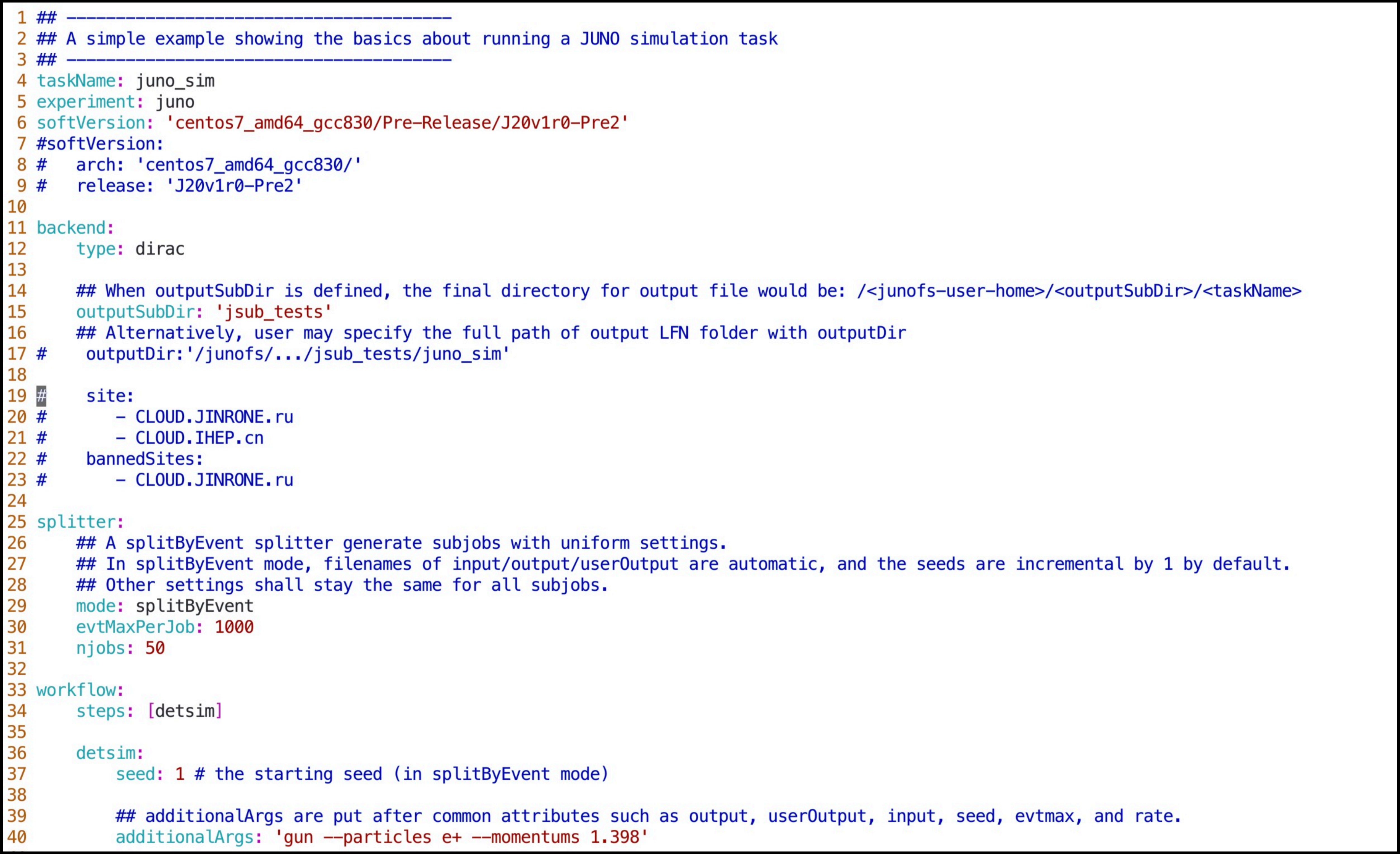


```
(jsub) yangyf:[~] > jsub --help
Usage: jsub [OPTIONS] COMMAND [ARGS]...

Options:
  --jsubrc TEXT  Configuration file to run JSUB with.
  --help         Show this message and exit.

Commands:
  create      Create a task from a task description file.
  getlog      Retrieve log files of selected subjobs.
  ls          List all tasks.
  package     Show active packages.
  remove      Delete a task.
  rename      Rename a task.
  reschedule  Reschedule selected subjobs.
  resubmit    Equivalent to 'jsub submit -r' command
  run         Create from a task profile, and submit.
  show        Show detailed description of a task.
  status      Show the backend status of a task.
  submit      Submit a task to backend.
  version     Show the version of the software.
```

- Also, check http://jsubpy.github.io for user guide.

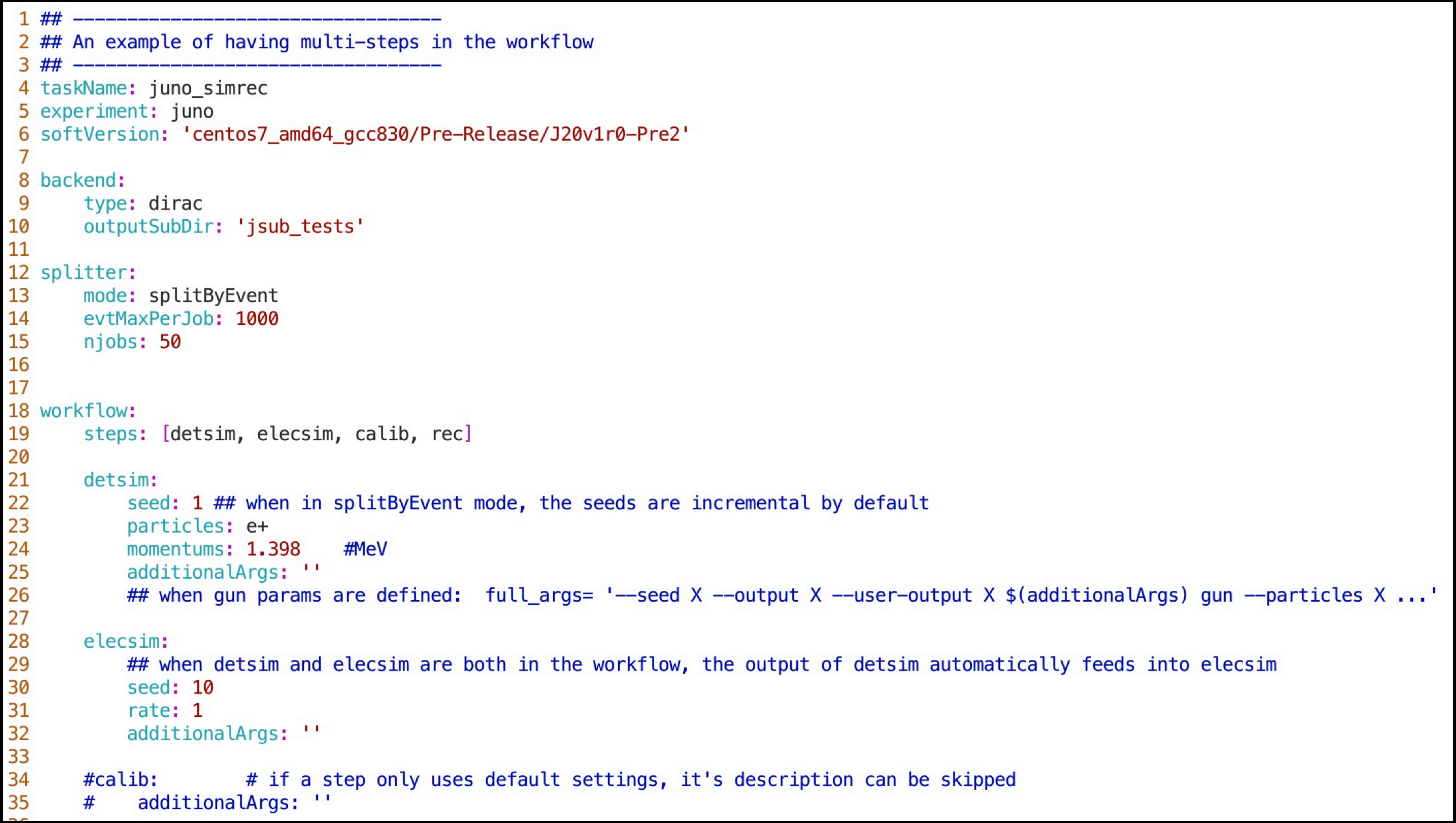
TDF examples

Simple detsim task



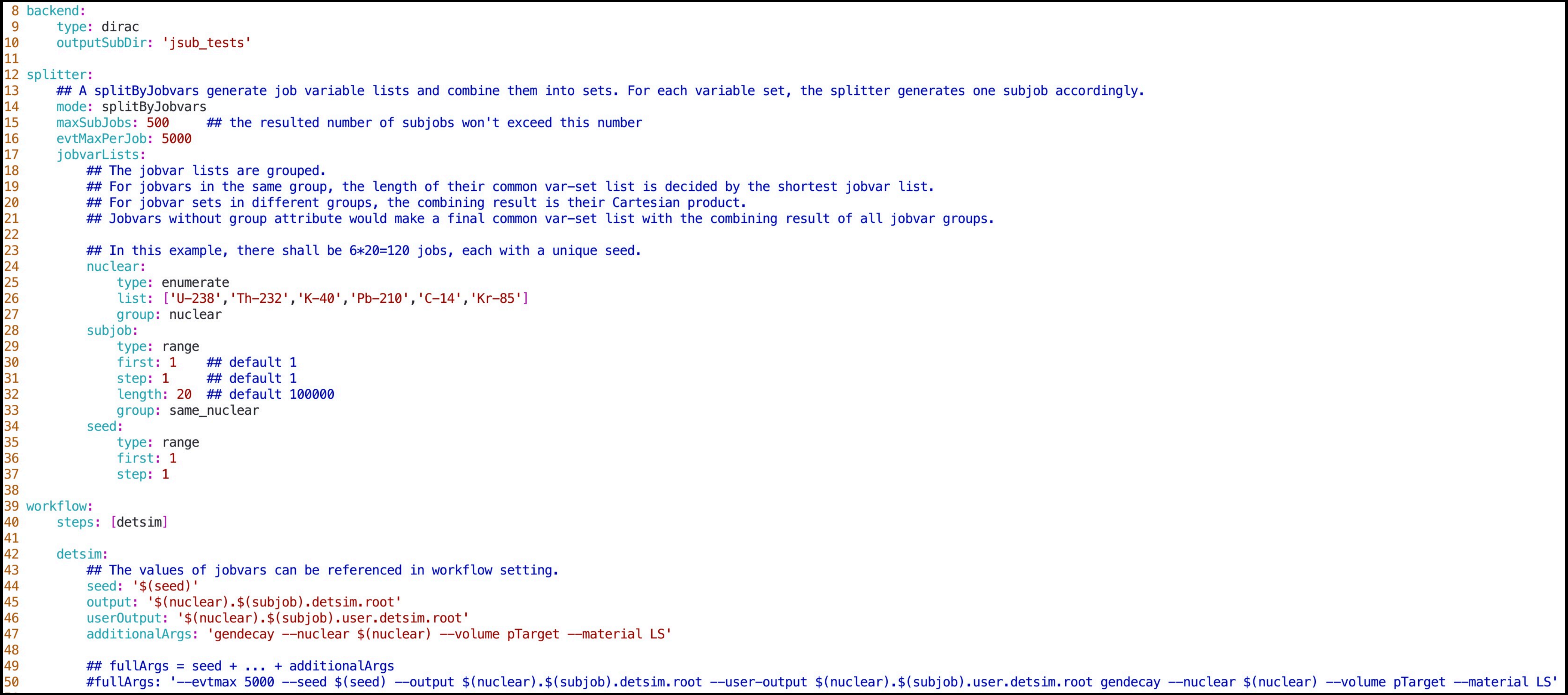
```yaml
## ---------------------------------------
## A simple example showing the basics about running a JUNO simulation task
## ---------------------------------------
taskName: juno_sim
experiment: juno
softVersion: 'centos7_amd64_gcc830/Pre-Release/J20v1r0-Pre2'
#softVersion:
#   arch: 'centos7_amd64_gcc830/'
#   release: 'J20v1r0-Pre2'

backend:
    type: dirac

    ## When outputSubDir is defined, the final directory for output file would be: /<junofs-user-home>/<outputSubDir>/<taskName>
    outputSubDir: 'jsub_tests'
    ## Alternatively, user may specify the full path of output LFN folder with outputDir
#    outputDir:'/junofs/.../jsub_tests/juno_sim'

#    site:
#       - CLOUD.JINRONE.ru
#       - CLOUD.IHEP.cn
#    bannedSites:
#       - CLOUD.JINRONE.ru

splitter:
    ## A splitByEvent splitter generate subjobs with uniform settings.
    ## In splitByEvent mode, filenames of input/output/userOutput are automatic, and the seeds are incremental by 1 by default.
    ## Other settings shall stay the same for all subjobs.
    mode: splitByEvent
    evtMaxPerJob: 1000
    njobs: 50

workflow:
    steps: [detsim]

    detsim:
        seed: 1 # the starting seed (in splitByEvent mode)

        ## additionalArgs are put after common attributes such as output, userOutput, input, seed, evtmax, and rate.
        additionalArgs: 'gun --particles e+ --momentums 1.398'
```

sim/rec multi-step task



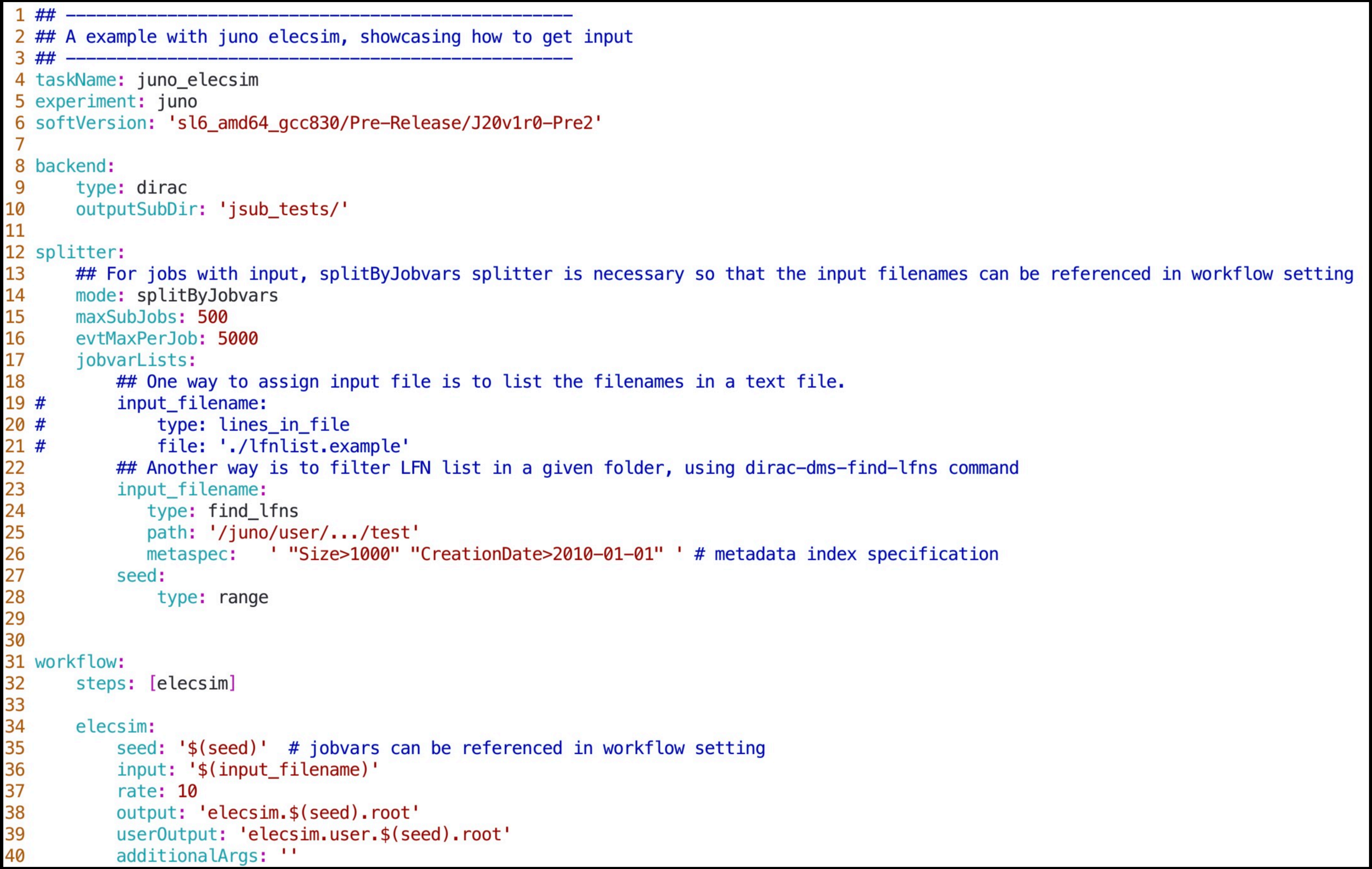
```
## -----------------------------------
## An example of having multi-steps in the workflow
## -----------------------------------
taskName: juno_simrec
experiment: juno
softVersion: 'centos7_amd64_gcc830/Pre-Release/J20v1r0-Pre2'

backend:
    type: dirac
    outputSubDir: 'jsub_tests'

splitter:
    mode: splitByEvent
    evtMaxPerJob: 1000
    njobs: 50


workflow:
    steps: [detsim, elecsim, calib, rec]

    detsim:
        seed: 1 ## when in splitByEvent mode, the seeds are incremental by default
        particles: e+
        momentums: 1.398    #MeV
        additionalArgs: ''
        ## when gun params are defined:  full_args= '--seed X --output X --user-output X $(additionalArgs) gun --particles X ...'

    elecsim:
        ## when detsim and elecsim are both in the workflow, the output of detsim automatically feeds into elecsim
        seed: 10
        rate: 1
        additionalArgs: ''

    #calib:        # if a step only uses default settings, it's description can be skipped
    #    additionalArgs: ''
```

splitByJobvar splitter — a splitter with better handling of subjob details



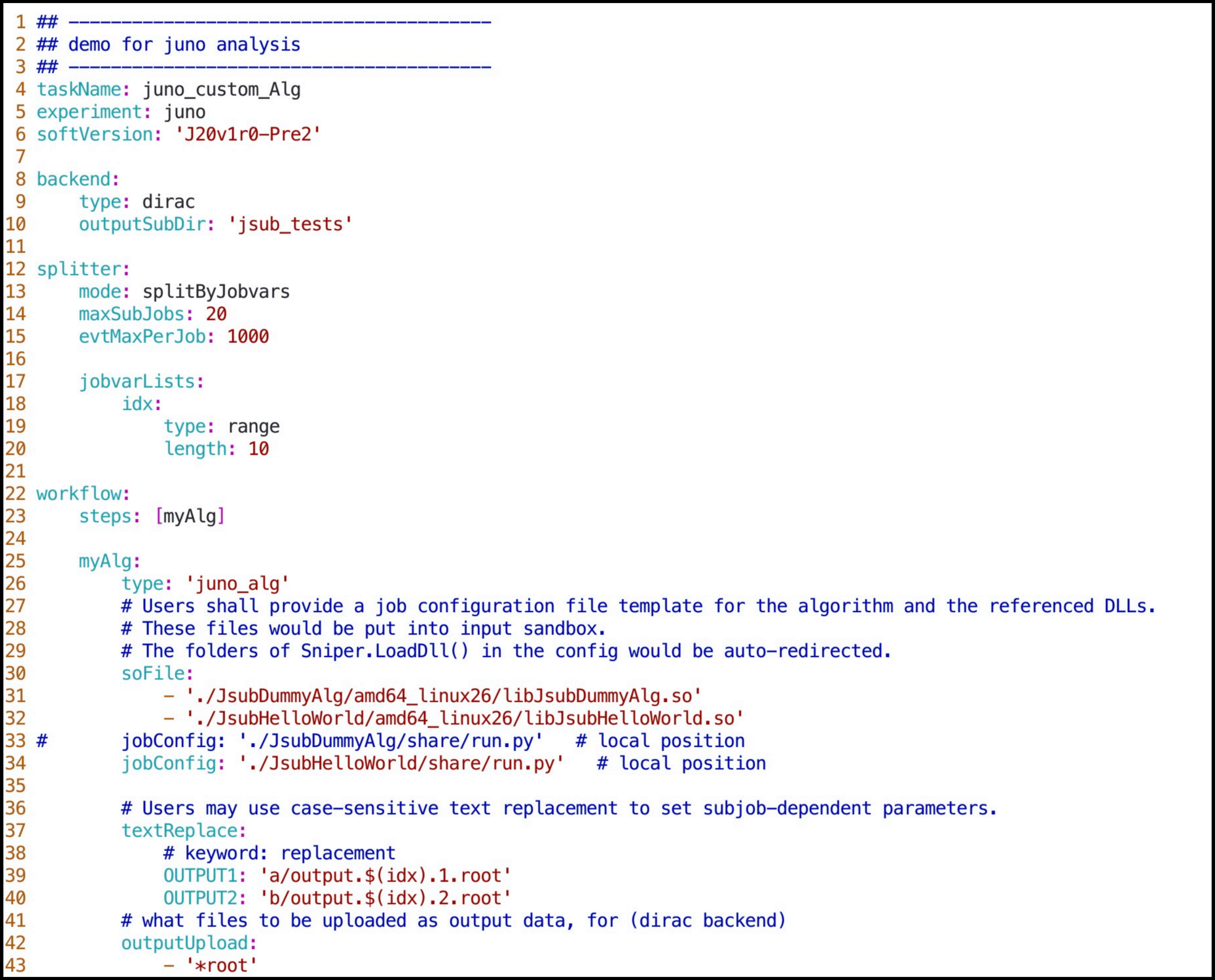
```yaml
backend:
    type: dirac
    outputSubDir: 'jsub_tests'

splitter:
    ## A splitByJobvars generate job variable lists and combine them into sets. For each variable set, the splitter generates one subjob accordingly.
    mode: splitByJobvars
    maxSubJobs: 500     ## the resulted number of subjobs won't exceed this number
    evtMaxPerJob: 5000
    jobvarLists:
        ## The jobvar lists are grouped.
        ## For jobvars in the same group, the length of their common var-set list is decided by the shortest jobvar list.
        ## For jobvar sets in different groups, the combining result is their Cartesian product.
        ## Jobvars without group attribute would make a final common var-set list with the combining result of all jobvar groups.

        ## In this example, there shall be 6*20=120 jobs, each with a unique seed.
        nuclear:
            type: enumerate
            list: ['U-238','Th-232','K-40','Pb-210','C-14','Kr-85']
            group: nuclear
        subjob:
            type: range
            first: 1    ## default 1
            step: 1     ## default 1
            length: 20  ## default 100000
            group: same_nuclear
        seed:
            type: range
            first: 1
            step: 1

workflow:
    steps: [detsim]

    detsim:
        ## The values of jobvars can be referenced in workflow setting.
        seed: '$(seed)'
        output: '$(nuclear).$(subjob).detsim.root'
        userOutput: '$(nuclear).$(subjob).user.detsim.root'
        additionalArgs: 'gendecay --nuclear $(nuclear) --volume pTarget --material LS'

        ## fullArgs = seed + ... + additionalArgs
        #fullArgs: '--evtmax 5000 --seed $(seed) --output $(nuclear).$(subjob).detsim.root --user-output $(nuclear).$(subjob).user.detsim.root gendecay --nuclear $(nuclear) --volume pTarget --material LS'
```

elecsim task — handling input data



```yaml
## --------------------------------------------------
## A example with juno elecsim, showcasing how to get input
## --------------------------------------------------
taskName: juno_elecsim
experiment: juno
softVersion: 'sl6_amd64_gcc830/Pre-Release/J20v1r0-Pre2'

backend:
    type: dirac
    outputSubDir: 'jsub_tests/'

splitter:
    ## For jobs with input, splitByJobvars splitter is necessary so that the input filenames can be referenced in workflow setting
    mode: splitByJobvars
    maxSubJobs: 500
    evtMaxPerJob: 5000
    jobvarLists:
        ## One way to assign input file is to list the filenames in a text file.
#       input_filename:
#           type: lines_in_file
#           file: './lfnlist.example'
        ## Another way is to filter LFN list in a given folder, using dirac-dms-find-lfns command
        input_filename:
           type: find_lfns
           path: '/juno/user/.../test'
           metaspec:   ' "Size>1000" "CreationDate>2010-01-01" ' # metadata index specification
        seed:
            type: range


workflow:
    steps: [elecsim]

    elecsim:
        seed: '$(seed)'  # jobvars can be referenced in workflow setting
        input: '$(input_filename)'
        rate: 10
        output: 'elecsim.$(seed).root'
        userOutput: 'elecsim.user.$(seed).root'
        additionalArgs: ''
```

Custom JUNO analysis



```yaml
## ----------------------------------------
## demo for juno analysis
## ----------------------------------------
taskName: juno_custom_Alg
experiment: juno
softVersion: 'J20v1r0-Pre2'

backend:
    type: dirac
    outputSubDir: 'jsub_tests'

splitter:
    mode: splitByJobvars
    maxSubJobs: 20
    evtMaxPerJob: 1000

    jobvarLists:
        idx:
            type: range
            length: 10

workflow:
    steps: [myAlg]

    myAlg:
        type: 'juno_alg'
        # Users shall provide a job configuration file template for the algorithm and the referenced DLLs.
        # These files would be put into input sandbox.
        # The folders of Sniper.LoadDll() in the config would be auto-redirected.
        soFile:
            - './JsubDummyAlg/amd64_linux26/libJsubDummyAlg.so'
            - './JsubHelloWorld/amd64_linux26/libJsubHelloWorld.so'
#       jobConfig: './JsubDummyAlg/share/run.py'   # local position
        jobConfig: './JsubHelloWorld/share/run.py'   # local position

        # Users may use case-sensitive text replacement to set subjob-dependent parameters.
        textReplace:
            # keyword: replacement
            OUTPUT1: 'a/output.$(idx).1.root'
            OUTPUT2: 'b/output.$(idx).2.root'
        # what files to be uploaded as output data, for (dirac backend)
        outputUpload:
            - '*root'
```

Summary

- JSUB is ready for more trying.
- Feedbacks are welcome!

Thanks!