Package 'abba'

April 28, 2026

Title Batch Execution of R Programs on 'Kubernetes', 'SLURM', and 'Posit Workbench'

Version 0.2.0

Description Submit and monitor batch execution of R programs across distributed computing backends including 'Kubernetes', 'SLURM', and 'Posit Workbench'. Provides end-user job submission functions, cluster interface functions using 'kubectl' and 'SLURM' commands, and a 'plumber' API template for secure identity segregation. Supports parallel and sequential batch execution, file-based caching to skip unchanged programs, and 'logrx' integration for execution logging.

License Apache License (>= 2)

URL https://atorus-research.github.io/abba/, https://github.com/atorus-research/abba

BugReports https://github.com/atorus-research/abba/issues

Encoding UTF-8

RoxygenNote 7.3.2

Imports yaml, uuid, stringr, httr2, magrittr, tools, rstudioapi, digest, tidyr

Suggests testthat (>= 3.0.0), mockery, knitr, rmarkdown, logrx

Config/testthat/edition 3

VignetteBuilder knitr

SystemRequirements kubectl (optional, for Kubernetes backend), SLURM (optional, sbatch/scontrol/sacct/squeue for SLURM backend), Posit Workbench (optional, for Workbench backend)

NeedsCompilation no

Author Eli Miller [aut] (ORCID: <https://orcid.org/0000-0002-2127-9456>), Mike Stackhouse [aut, cre] (ORCID: <https://orcid.org/0000-0001-6030-723X>), Ross Didenko [aut], Yevhenii Boiko [aut], Atorus Research, Inc. [cph]

Maintainer Mike Stackhouse <mike.stackhouse@atorusresearch.com>

Repository CRAN

Date/Publication 2026-04-28 18:10:14 UTC

Contents

abba_get_batch_log *Send GET request to get logs of all jobs in a batch*

Description

Send GET request to get logs of all jobs in a batch

Usage

```
abba_get_batch_log(
  batch_id,
  api_address = Sys.getenv("ABBA_API_ADDRESS"),
  api_key = Sys.getenv("ABBA_API_KEY")
)
```

Arguments

batch_id	unique batch identificator
api_address	URL to send requests to, hosted in Posit Connect. Defaults to environment variable ABBA_API_ADDRESS.
api_key	API Key for accessing restricted endpoints. Defaults to environment variable ABBA_API_KEY

Value

body of request‘s response in a list format

Examples

```
## Not run:
response <- abba_get_job_log('1234j-13j4l5k-ajslfd')
## End(Not run)
```

abba_get_batch_status *Send GET request to get batch job statuses*

Description

Send GET request to get batch job statuses

Usage

```
abba_get_batch_status(
  batch_id,
  api_address = Sys.getenv("ABBA_API_ADDRESS"),
  api_key = Sys.getenv("ABBA_API_KEY")
)
```

Arguments

batch_id	batch ID to get status for
api_address	URL to send requests to, hosted in Posit Connect. Defaults to environment variable ABBA_API_ADDRESS.
api_key	API Key for accessing restricted endpoints. Defaults to environment variable ABBA_API_KEY

Value

body of request‘s response in a list format

Examples

```
## Not run:
response <- abba_get_batch_status('1234j-13j4l5k-ajslfd')
## End(Not run)
```

---

abba_get_job_log *Send GET request to get logs of specified Jobs*

Description

Send GET request to get logs of specified Jobs

Usage

```
abba_get_job_log(
  job_ids,
  api_address = Sys.getenv("ABBA_API_ADDRESS"),
  api_key = Sys.getenv("ABBA_API_KEY")
)
```

Arguments

job_ids	A list of job IDs to get logs for
api_address	URL to send requests to, hosted in Posit Connect. Defaults to environment variable ABBA_API_ADDRESS.
api_key	API Key for accessing restricted endpoints. Defaults to environment variable ABBA_API_KEY

Value

body of request‘s response in a list format

Examples

```
## Not run:
response <- abba_get_job_log('1234j-13j4l5k-ajslfd')
## End(Not run)
```

`abba_get_job_status` *Send GET request to get job status*

Description

Send GET request to get job status

Usage

```
abba_get_job_status(
  job_id,
  api_address = Sys.getenv("ABBA_API_ADDRESS"),
  api_key = Sys.getenv("ABBA_API_KEY")
)
```

Arguments

`job_id`	job IDs to get status for
`api_address`	URL to send requests to, hosted in Posit Connect. Defaults to environment variable ABBA_API_ADDRESS.
`api_key`	API Key for accessing restricted endpoints. Defaults to environment variable ABBA_API_KEY.

Value

body of request‘s response in a list format

Examples

```
## Not run:
response <- abba_get_job_status('1234j-13j4l5k-ajslfd')
## End(Not run)
```

`abba_get_k8s_batch_log_local` *Return list of logs for jobs that are marked with a given batch ID*

Description

Return list of logs for jobs that are marked with a given batch ID

Usage

```
abba_get_k8s_batch_log_local(
  batch_id,
  namespace = getOption("abba.k8s.namespace")
)
```

Arguments

`batch_id`	string containing batch ID
`namespace`	Kubernetes namespace to search for batch jobs

Value

Job logs in a from of list consisting of character vectors

Examples

```
## Not run:
logs <- abba_get_k8s_batch_log_local("batch-sdtm-abc123")

## End(Not run)
```

`abba_get_k8s_batch_status_local` *Get status of all jobs in a batch*

Description

Get status of all jobs in a batch

Usage

```
abba_get_k8s_batch_status_local(
  batch_id,
  namespace = getOption("abba.k8s.namespace")
)
```

Arguments

batch_id	unique identifier for a batch
namespace	Kubernetes namespace

Value

list of statuses for every job in a batch

Examples

```
## Not run:
status <- abba_get_k8s_batch_status_local("batch-sdtm-abc123")

## End(Not run)
```

---

abba_get_k8s_job_log_local

Get log for every job specified in an input vector/list

Description

Get log for every job specified in an input vector/list

Usage

```
abba_get_k8s_job_log_local(
  job_ids,
  namespace = getOption("abba.k8s.namespace")
)
```

Arguments

job_ids	A list of job IDs to get logs for
namespace	Kubernetes namespace to search for job

Value

A list of job logs. Each list entry will contain complete log for a job

Examples

```
## Not run:
logs <- abba_get_k8s_job_log_local(c("job-sdtm-abc123", "job-adam-def456"))

## End(Not run)
```

abba_get_k8s_job_status_local

Get status of all pods that belong to a job

Description

Get status of all pods that belong to a job

Usage

```
abba_get_k8s_job_status_local(
  job_id,
  namespace = getOption("abba.k8s.namespace")
)
```

Arguments

job_id	unique identifier for a job
namespace	Kubernetes namespace

Value

list of statuses for every pod in a job(typically just one).

Examples

```
## Not run:
status <- abba_get_k8s_job_status_local("job-sdtm-abc123")

## End(Not run)
```

abba_rslauncher_get_job_status_local

Get Workbench job status for a given vector/list of job IDs

Description

Get Workbench job status for a given vector/list of job IDs

Usage

```
abba_rslauncher_get_job_status_local(job_ids, ...)
```

Arguments

job_ids	A list/vector of job IDs
...	other positional/keyword arguments that will be ignored

Value

a vector of job ID statuses

Examples

```
## Not run:
job_statuses <- abba_rslauncher_get_job_status_local(c('job-id-1', 'job-id-2'))

## End(Not run)
```

`abba_rslauncher_get_job_succeeded_local` *Check whether Workbench jobs have been fully executed.*

Description

Check whether Workbench jobs have been fully executed.

Usage

```
abba_rslauncher_get_job_succeeded_local(job_ids, ...)
```

Arguments

`job_ids`	a list/vector of Workbench job IDs
`...`	other positional/keyword arguments that will be ignored

Value

a named boolean vector. FALSE value indicates that job did not fully execute

Examples

```
## Not run:
job_statuses <- abba_rslauncher_get_job_succeeded_local(c('job-id-1', 'job-id-2'))

## End(Not run)
```

abba_rslauncher_submit_job_local

Create a job for executing an R program

Description

Create a job for executing an R program

Usage

```
abba_rslauncher_submit_job_local(p, log_path, user_tag = "", ...)
```

Arguments

p	path to program
log_path	Path to the directory where the log will be saved. Required; must be supplied explicitly so that logs are never written to an unexpected location in the user's filespace.
user_tag	optional; any user tags user might want to add to the job
...	other arguments that will be passed to internal rslauncher_submit_job function

Value

job id

Examples

```
## Not run:
job_id <- abba_rslauncher_submit_job_local("/path/to/program.R",
                                           log_path = "/path/to/logs")

## End(Not run)
```

abba_rslauncher_submit_logrx_job_local

Execute programs via logrx

Description

Execute programs via logrx

Usage

```
abba_rslauncher_submit_logrx_job_local(p, log_path, user_tag = "", ...)
```

Arguments

p	path to program
log_path	Path to the directory where the log will be saved. Required; must be supplied explicitly so that logs are never written to an unexpected location in the user's filespace.
user_tag	optional; any user tags user might want to add to the job
...	other arguments that will be passed to internal rslauncher_submit_job function

Value

job id

Examples

```
## Not run:
job_id <- abba_rslauncher_submit_logrx_job_local("/path/to/program.R",
                                                 log_path = "/path/to/logs")

## End(Not run)
```

---

abba_rslauncher_watch_job_local

Periodically poll Workbench jobs for status and return their IDs when all job statuses arrive at 'Finished' state

---

Description

Periodically poll Workbench jobs for status and return their IDs when all job statuses arrive at 'Finished' state

Usage

```
abba_rslauncher_watch_job_local(
  job_ids,
  poll_interval_seconds = 1,
  timeout_seconds = 300,
  ...
)
```

Arguments

job_ids	a list/vector of Workbench job IDs
poll_interval_seconds	how often job statuses should be updated
timeout_seconds	maximum amount of time in seconds after which job IDs will be returned regardless of job statuses
...	other positional/keyword arguments that will be ignored

Value

a vector of job IDs

Examples

```
## Not run:
job_statuses <- abba_rslauncher_watch_job_local(c('job-id-1', 'job-id-2'))

## End(Not run)
```

`abba_slurm_get_job_log`

Return job log for each submitted job ID

Description

Return job log for each submitted job ID

Usage

```
abba_slurm_get_job_log(job_ids, ...)
```

Arguments

`job_ids`	valid SLURM job IDs
`...`	other parameters that will be ignored

Value

list of character vectors. each character vector would represent program log.

Examples

```
## Not run:
job_log <- abba_slurm_get_job_log("156")
## End(Not run)
```

abba_slurm_get_job_status

Return descriptive job status for slurm jobs

Description

Return descriptive job status for slurm jobs

Usage

```
abba_slurm_get_job_status(job_ids, ...)
```

Arguments

job_ids	a list/vector of Slurm job IDs
...	other positional/keyword arguments that will be ignored

Value

a named character vector with job statuses as values and job IDs as names.

Examples

```
## Not run:
job_statuses <- slurm_get_job_status(c('job-id-1', 'job-id-2'))

## End(Not run)
```

abba_slurm_get_job_succeeded

Check whether SLURM jobs have been fully executed.

Description

Check whether SLURM jobs have been fully executed.

Usage

```
abba_slurm_get_job_succeeded(job_ids, ...)
```

Arguments

job_ids	a list/vector of valid SLURM job IDs
...	other parameters that will be ignored

Value

a named boolean vector. TRUE if job finished running and has 'COMPLETED' status, FALSE otherwise. If job has not finished running, a NULL will be returned.

Examples

```
## Not run:
job_statuses <- abba_slurm_get_job_succeeded(c('job-id-1', 'job-id-2'))

## End(Not run)
```

`abba_slurm_submit_job` *Submit R program as a SLURM job*

Description

Submit R program as a SLURM job

Usage

```
abba_slurm_submit_job(
  program_path,
  log_path,
  r_version = NULL,
  user_tag = NULL,
  cpu_cores = getOption("abba.slurm.cpu.cores"),
  memory = getOption("abba.slurm.memory"),
  username = NULL,
  working_dir = NULL,
  job_timeout = 3600,
  ...
)
```

Arguments

`program_path`	Full path to the R program file. Must be accessible from the SLURM node
`log_path`	Parent folder for the program's log file. Required; must be supplied explicitly so that logs are never written to an unexpected location in the user's filespace.
`r_version`	Version of R that will be used to run the program. Can be specified as a full path to Rscript executable, or as a label of R version that is displayed in the Workbench GUI.
`user_tag`	custom string that will be added to the job name.
`cpu_cores`	Amount of CPU cores that will be requested for the job.
`memory`	Amount of RAM in megabytes that will be requested for the job.
`username`	user whose permission level is used to execute the script. Defaults to user submitting the job.

working_dir	working directory for the SLURM job. Defaults to parent directory of program_path.
job_timeout	time limit for a job. Must be specified in a format of "days-hours:minutes:seconds" If exceeded, job will be cancelled.
...	additional arguments (currently unused)

Value

job ID

Examples

```
## Not run:
job_id <- abba_slurm_submit_job("/home/user/tfl/t1_dm.sas",
                                log_path = "/home/user/tfl/logs")

## End(Not run)
```

---

abba_slurm_watch_job *Watch SLURM job, periodically polling its execution status.*

Description

Watch SLURM job, periodically polling its execution status.

Usage

```
abba_slurm_watch_job(
  job_id = "",
  poll_interval_seconds = 3,
  timeout_seconds = 300,
  ...
)
```

Arguments

job_id	job ID that was specified when submitting jobs
poll_interval_seconds	Time interval for polling job status in seconds
timeout_seconds	Total time to wait before timeout in seconds
...	other positional/keyword arguments that will be ignored

Value

a list of job ID(s) and status(es)

Examples

```
## Not run:
result <- abba_slurm_watch_job("5195", 10, 3000)

## End(Not run)
```

---

abba_submit_and_get_log *Send job and poll for status. This function sends multiple requests so it won't time out on heavy jobs*

Description

Send job and poll for status. This function sends multiple requests so it won't time out on heavy jobs

Usage

```
abba_submit_and_get_log(
  file_path,
  batch_group_id = "",
  user_tag = "",
  cpu_limit = 1L,
  memory_limit = "512M",
  container = "",
  mounts = "",
  auto_mount_home = FALSE,
  home_nfs_ip_address = getOption("abba.home.nfs.ip.address"),
  poll_interval_seconds = 3,
  timeout_seconds = 600,
  api_address = Sys.getenv("ABBA_API_ADDRESS"),
  api_key = Sys.getenv("ABBA_API_KEY")
)
```

Arguments

file_path	Full path to R file
batch_group_id	Group ID for batch processing
user_tag	Optional; a string that describes what kind of job will be scheduled to run
cpu_limit	Maximum number of cores available for Kubernetes container
memory_limit	Maximum amount of RAM available for Kubernetes container
container	list that contains container name and image name
mounts	Specifically formatted list with information bout volumes that container would have access to during the run

auto_mount_home
: set to TRUE to mount service user home directory

home_nfs_ip_address
: IP address for mounting service user home directory

poll_interval_seconds
: Total time to wait before timeout in seconds

timeout_seconds
: Total time to wait before timeout in seconds

api_address
: URL to send requests to, hosted in Posit Connect. Defaults to environment variable ABBA_API_ADDRESS.

api_key
: API Key for accessing restricted endpoints. Defaults to environment variable ABBA_API_KEY.

Value

list with 2 attributes: job_id for submitted job‘s id, and its logs

Examples

```
## Not run:
response <- abba_submit_and_get_log('/path/to/R/program.R', batch_group_id='SDTM')
## End(Not run)
```

---

abba_submit_batch *Submit programs for execution in order defined by structure of input list. Programs inside sublists will be executed in parallel, and sublists themselves would be submitted sequentially.*

Description

Submit programs for execution in order defined by structure of input list. Programs inside sublists will be executed in parallel, and sublists themselves would be submitted sequentially.

Usage

```
abba_submit_batch(
  prog_list,
  sequential = FALSE,
  submit_func = abba_rslauncher_submit_job_local,
  wait_func = abba_rslauncher_watch_job_local,
  succeed_func = abba_rslauncher_get_job_succeeded_local,
  col_name = "run_group",
  halt_on_error = TRUE,
  rerun_unchanged_programs = TRUE,
  update_cache = FALSE,
  cache_folder = getOption("abba.default_cache_folder"),
  ...
)
```

Arguments

prog_list	A list of R program paths to execute
sequential	when sequential=TRUE, prog_list is flattened and everything is executed sequentially.
submit_func	function that will be used to submit jobs
wait_func	function that checks job status and returns when job finishes executing
succeed_func	function that returns TRUE if job finished running without errors and FALSE otherwise
col_name	Name of the column that contains run group numbers when prog_list is a data frame
halt_on_error	If TRUE: if prog_list contains program inputs/outputs - programs that depend on failed program will not be executed; if prog_list contains only program paths - when program fails, entire batch will stop executing. TRUE by default
rerun_unchanged_programs	If FALSE: will not re-run programs whose code and inputs have not been modified since last batch run with update_cache=TRUE.
update_cache	if TRUE, file hash for programs and their inputs will be calculated and written to cache_folder after the batch run. If FALSE, the cache will not be created or updated. FALSE by default.
cache_folder	Path to the folder where hash-sums of programs and their inputs will be stored. Required when update_cache=TRUE or rerun_unchanged_programs=FALSE. The default value can be set via the abba.default_cache_folder option; abba never falls back to writing caches under the user's filespace.
...	arguments that will be passed to submit_func and wait_func functions

Value

list job IDs associated with executed programs

Examples

```
## Not run:
job_ids <- abba_submit_batch(list(
  c("/mnt/work_drive/proj/comp/prot/task/development/prod/program/sdtm/dm.sas"),
  c("/mnt/work_drive/proj/comp/prot/task/development/prod/program/sdtm/ae.sas"),
  c("/mnt/work_drive/proj/comp/prot/task/development/prod/program/tfl/t1_dm.sas",
    "/mnt/work_drive/proj/comp/prot/task/development/prod/program/tfl/t1_ae.sas")))

## End(Not run)
```

abba_submit_batch_and_get_results *Submit programs for execution in order defined by structure of input list.*

Description

Submit programs for execution in order defined by structure of input list.

Usage

```
abba_submit_batch_and_get_results(
  prog_list,
  sequential = FALSE,
  submit_func = abba_rslauncher_submit_job_local,
  wait_func = abba_rslauncher_watch_job_local,
  succeed_func = abba_rslauncher_get_job_succeeded_local,
  status_func = rslauncher_get_job_display_status,
  col_name = "run_group",
  halt_on_error = TRUE,
  rerun_unchanged_programs = TRUE,
  update_cache = FALSE,
  cache_folder = getOption("abba.default_cache_folder"),
  ...
)
```

Arguments

prog_list	A list of R program paths to execute
sequential	when sequential=TRUE, prog_list is flattened and everything is executed sequentially.
submit_func	function that will be used to submit jobs
wait_func	function that checks job status and returns when job finishes executing
succeed_func	function that returns TRUE if job finished running without errors and FALSE otherwise
status_func	function that returns descriptive job statuses
col_name	Name of the column that contains run group numbers when prog_list is a data frame
halt_on_error	If TRUE: if prog_list contains program inputs/outputs - programs that depend on failed program will not be executed; if prog_list contains only program paths - when program fails, entire batch will stop executing. TRUE by default
rerun_unchanged_programs	If FALSE: will not re-run programs whose code and inputs have not been modified since last batch run with update_cache=TRUE.

update_cache	if TRUE, file hash for programs and their inputs will be calculated and written to cache_folder after the batch run. If FALSE, the cache will not be created or updated. FALSE by default.
cache_folder	Path to the folder where hash-sums of programs and their inputs will be stored. Required when update_cache=TRUE or rerun_unchanged_programs=FALSE. The default value can be set via the abba.default_cache_folder option; abba never falls back to writing caches under the user's filespace.
...	arguments that will be passed to submit_func and wait_func functions

Value

Data frame containing program names, job ids, execution statuses

Examples

```
## Not run:
job_ids <- abba_submit_batch_and_get_results(list(
  c("/mnt/work_drive/proj/comp/prot/task/development/prod/program/sdtm/dm.sas"),
  c("/mnt/work_drive/proj/comp/prot/task/development/prod/program/sdtm/ae.sas"),
  c("/mnt/work_drive/proj/comp/prot/task/development/prod/program/tfl/t1_dm.sas",
    "/mnt/work_drive/proj/comp/prot/task/development/prod/program/tfl/t1_ae.sas")),
  sequential=TRUE)

## End(Not run)
```

---

abba_submit_job *Send POST request to submit-job endpoint*

Description

Send POST request to submit-job endpoint

Usage

```
abba_submit_job(
  file_path,
  batch_group_id = "",
  user_tag = "",
  cpu_limit = 1L,
  memory_limit = "512M",
  container = "",
  mounts = "",
  auto_mount_home = FALSE,
  home_nfs_ip_address = getOption("abba.home.nfs.ip.address"),
  api_address = Sys.getenv("ABBA_API_ADDRESS"),
  api_key = Sys.getenv("ABBA_API_KEY"),
  ...
)
```

Arguments

file_path	Full path to R file
batch_group_id	Group ID for batch processing
user_tag	Optional; a string that describes what kind of job will be scheduled to run
cpu_limit	Maximum number of cores available for Kubernetes container
memory_limit	Maximum amount of RAM available for Kubernetes container
container	A string containing a permitted container name.
mounts	Specifically formatted list with information bout volumes that container would have access to during the run
auto_mount_home	set to TRUE to mount service user home directory
home_nfs_ip_address	IP address for mounting service user home directory
api_address	URL to send requests to, hosted in Posit Connect. Defaults to environment variable ABBA_API_ADDRESS.
api_key	API Key for accessing restricted endpoints. Defaults to environment variable ABBA_API_KEY.
...	Other arguments that will be ignored

Value

body of request‘s response in a list format

Examples

```
## Not run:
response <- abba_submit_job('/path/to/R/program.R')
## End(Not run)
```

---

abba_submit_k8s_job_and_poll_local

Submit a job profile for execution on a Kubernetes cluster and poll for completion

---

Description

Submit a job profile for execution on a Kubernetes cluster and poll for completion

Usage

```
abba_submit_k8s_job_and_poll_local(
  file_path,
  batch_group_id = "",
  user_tag = "",
  cpu_limit = 1L,
  memory_limit = "512M",
  container = getOption("abba.default.container"),
  mounts = "",
  auto_mount_home = FALSE,
  home_nfs_ip_address = getOption("abba.home.nfs.ip.address"),
  namespace = getOption("abba.k8s.namespace"),
  username = NULL,
  poll_interval_seconds = 3,
  timeout_seconds = 600
)
```

Arguments

file_path	Full path to R file
batch_group_id	Group ID for batch processing
user_tag	Optional; a string that describes what kind of job will be scheduled to run
cpu_limit	Maximum number of cores available for Kubernetes container
memory_limit	Maximum amount of RAM available for Kubernetes container
container	A valid container image name provided as a character string. Defaults to the option abba.default.container.
mounts	Specifically formatted list with information bout volumes that container would have access to during the run
auto_mount_home	set to TRUE to mount service user home directory
home_nfs_ip_address	IP address for mounting service user home directory
namespace	Kubernetes namespace to put the job in
username	user whose authority will be used to run the program
poll_interval_seconds	Time interval for polling job status in seconds
timeout_seconds	Total time to wait before timeout in seconds

Value

A "Job completed successfully" message, or a list of failed jobs and their ID's.

Examples

```
## Not run:
result <- abba_submit_k8s_job_and_poll_local("path/to/your/job.yaml")

## End(Not run)
```

`abba_submit_k8s_job_local` *Submit an R program for execution on a Kubernetes cluster*

Description

Submit an R program for execution on a Kubernetes cluster

Usage

```
abba_submit_k8s_job_local(
  file_path,
  batch_group_id = "",
  user_tag = "",
  cpu_limit = 1L,
  memory_limit = "512M",
  container = getOption("abba.default.container"),
  mounts = "",
  auto_mount_home = FALSE,
  home_nfs_ip_address = getOption("abba.home.nfs.ip.address"),
  namespace = getOption("abba.k8s.namespace"),
  username = NULL
)
```

Arguments

`file_path`	Full path to R file
`batch_group_id`	Group ID for batch processing
`user_tag`	Optional; a string that describes what kind of job will be scheduled to run
`cpu_limit`	Maximum number of cores available for Kubernetes container
`memory_limit`	Maximum amount of RAM available for Kubernetes container
`container`	A valid container image name provided as a character string. Defaults to the option abba.default.container.
`mounts`	Specifically formatted list with information bout volumes that container would have access to during the run
`auto_mount_home`	set to TRUE to mount service user home directory

home_nfs_ip_address
: IP address for mounting service user home directory

namespace
: Kubernetes namespace to put the job in

username
: user whose authority will be used to run the program

Value

A list with job_id and batch_id attributes in case of successful submission

Examples

```
## Not run:
job_info <- abba_submit_k8s_job_local("path/to/your/program.R", batch_group_id='SDTM')

## End(Not run)
```

---

abba_wait_for_batch_log

Monitor batch status and retrieve its log when the all jobs in batch finish running

---

Description

Monitor batch status and retrieve its log when the all jobs in batch finish running

Usage

```
abba_wait_for_batch_log(
  batch_id,
  poll_interval_seconds = 3,
  timeout_seconds = 600,
  api_address = Sys.getenv("ABBA_API_ADDRESS"),
  api_key = Sys.getenv("ABBA_API_KEY")
)
```

Arguments

batch_id
: unique batch identificator

poll_interval_seconds
: Total time to wait before timeout in seconds

timeout_seconds
: Total time to wait before timeout in seconds

api_address
: URL to send requests to, hosted in Posit Connect. Defaults to environment variable ABBA_API_ADDRESS.

api_key
: API Key for accessing restricted endpoints. Defaults to environment variable ABBA_API_KEY

Value

list with 2 sublists: job_ids and their logs

Examples

```
## Not run:
response <- abba_wait_for_batch_log('batch-sdtm-sdfj4-asdjlk-bjslk')
## End(Not run)
```

---

abba_wait_for_job_log *Monitor job status and retrieve its log when the job finishes running*

Description

Monitor job status and retrieve its log when the job finishes running

Usage

```
abba_wait_for_job_log(
  job_id,
  poll_interval_seconds = 3,
  timeout_seconds = 600,
  api_address = Sys.getenv("ABBA_API_ADDRESS"),
  api_key = Sys.getenv("ABBA_API_KEY"),
  ...
)
```

Arguments

job_id	unique job identificator
poll_interval_seconds	Total time to wait before timeout in seconds
timeout_seconds	Total time to wait before timeout in seconds
api_address	URL to send requests to, hosted in Posit Connect. Defaults to environment variable ABBA_API_ADDRESS.
api_key	API Key for accessing restricted endpoints. Defaults to environment variable ABBA_API_KEY
...	Other arguments that will be ignored

Value

list with 2 attributes: job_id for submitted job‘s id, and its logs

Examples

```
## Not run:
response <- abba_wait_for_job_log('sdfj4-asdjlk-bjslk')
## End(Not run)
```

`abba_watch_k8s_batch_local` *Watch a K8S batch that has been submitted to Workbench, periodically polling it's execution status.*

Description

Watch a K8S batch that has been submitted to Workbench, periodically polling it's execution status.

Usage

```
abba_watch_k8s_batch_local(
  batch_group_id = "",
  poll_interval_seconds = 3,
  timeout_seconds = 600,
  namespace = getOption("abba.k8s.namespace")
)
```

Arguments

`batch_group_id`	Batch ID that was specified when submitting a group of jobs
`poll_interval_seconds`	Time interval for polling batch status in seconds
`timeout_seconds`	Total time to wait before timeout in seconds
`namespace`	Kubernetes namespace

Value

a list of jobs IDs and statuses that belong to batch named batch_group_id

Examples

```
## Not run:
result <- abba_watch_k8s_batch_local("safety-tfls-f0bf6848-46de-45b8-9fae-0e732b104760", 10, 3000)

## End(Not run)
```

`abba_watch_k8s_job_local` *Watch a K8S job that has been submitted to Workbench, periodically polling it's execution status.*

Description

Watch a K8S job that has been submitted to Workbench, periodically polling it's execution status.

Usage

```
abba_watch_k8s_job_local(
  job_id = "",
  poll_interval_seconds = 3,
  timeout_seconds = 600,
  namespace = getOption("abba.k8s.namespace")
)
```

Arguments

`job_id`	Job ID. Typically obtained as a return value from submit_job and similar functions
`poll_interval_seconds`	Time interval for polling job status in seconds
`timeout_seconds`	Total time to wait before timeout in seconds
`namespace`	Kubernetes namespace

Value

a list of pods, their IDs and execution statuses

Examples

```
## Not run:
result <- abba_watch_k8s_job_local("job-sdtm-f0bf6848-46de-45b8-9fae-0e732b104760", 10, 3000)

## End(Not run)
```

calculate_run_group *Calculate run_group variable using inputs and outputs of programs supplied by user*

Description

Calculate run_group variable using inputs and outputs of programs supplied by user

Usage

```
calculate_run_group(x, col_name = "run_group")
```

Arguments

x	input data frame. Must contain columns 'inputs', 'outputs' that list input/output datasets for each program
col_name	name of newly created variable. Defaults to 'run_group_calculated'

Value

an input data frame with one new column

Examples

```
input_ds <- as.data.frame(list(program_name=c('prog1.R', 'prog2.R'),
                               inputs=c('ds0.xpt', 'ds1.xpt'),
                               outputs=c('ds1.xpt', 'ds2.xpt')))
batch_ready <- calculate_run_group
```

create_batch_api *Create a Batch API or Job file template*

Description

This function create a batch API or job file template at the specified location. The Batch API file template is a plumber API with the necessary REST API server side to interface with the **abba** package. This simplifies the process of setting up the receiver API for which jobs are submitted. The job template file is a markdown file with the necessary function calls to run a batch job.

Usage

```
create_batch_api(path)

create_batch_job(path)
```

Arguments

path	A file path where the target file will be created. Must be supplied explicitly; no default is provided so that files are never written to an unexpected location.

Details

Note that to deploy an R api to Posit Connect, the file must be named plumber.R.

Value

No return value, called for side effects (file creation).

Examples

```
# Write the template to a temporary directory
create_batch_api(tempdir())
create_batch_job(tempdir())

## Not run:
create_batch_api("~/api_directory")
create_batch_api("~/api_directory/plumber.R")

create_batch_job("~/job_directory")
create_batch_job("~/job_directory/my_job.Rmd")

## End(Not run)
```

---

submit_k8s_yaml *Submit a job profile for execution on a Kubernetes cluster*

Description

Submit a job profile for execution on a Kubernetes cluster

Usage

```
submit_k8s_yaml(yaml_full_path)
```

Arguments

yaml_full_path	Path to YAML config file

Value

A string containing Job ID

Examples

```
## Not run:
job_id <- submit_k8s_yaml("/tmp/my_job.yaml")

## End(Not run)
```

Index