Package 'FaaSr'

April 5, 2025

Type Package

Title FaaS (Function as a Service) Package

Version 1.4.4

Maintainer Figueiredo Renato <renato.figueiredo@oregonstate.edu>

Description Allows users to create and deploy the workflow with multiple functions in Function-as-a-Service (FaaS) cloud computing platforms.
The 'FaaSr' package makes it simpler for R developers to use FaaS platforms by providing the following functionality:
1) Parsing and validating a JSON-based payload compliant to 'FaaSr' schema supporting multiple FaaS platforms
2) Invoking user functions written in R in a Docker container (derived from rocker), using a list generated from the parser as argument
3) Downloading/uploading of files from/to S3 buckets using simple primitives
4) Logging to files in S3 buckets
5) Triggering downstream actions supporting multiple FaaS platforms
6) Generating FaaS-specific API calls to simplify the registering of a user's workflow with a FaaS platform
Supported FaaS platforms:
Apache OpenWhisk <https://openwhisk.apache.org/>
GitHub Actions <https://github.com/features/actions>
Amazon Web Services (AWS) Lambda <https://aws.amazon.com/lambda/>
Supported cloud data storage for persistent storage:
Amazon Web Services (AWS) Simple Storage Service (S3) <https://aws.amazon.com/s3/>.

License MIT + file LICENSE

URL https://github.com/FaaSr/FaaSr-package

BugReports https://github.com/FaaSr/FaaSr-package/issues

Depends R (>= 3.5.0)

Imports jsonlite, httr, uuid, paws.application.integration, paws.compute, paws.storage, paws.security.identity, cli, jsonvalidate, base64enc, sodium, askpass

Suggests arrow, glue, rmarkdown, paws.common, testthat, knitr, devtools, utils, methods, withr

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Figueiredo Renato [aut, cre, ths, cph]
(<https://orcid.org/0000-0001-9841-6060>),
Park Sungjae [aut] (<https://orcid.org/0009-0000-5357-804X>),
Mu Nan [ctb],
Ku Yun-Jung [ctb],
Daneshmand Vahid [ctb],
Thomas R. Quinn [aut],
Carey Cayelan [ctb]

Repository CRAN

Date/Publication 2025-04-04 23:30:02 UTC

Contents

faasr *faasr*

Description

FaaSr library client-side main function It's generating the instance for the user Users can use the functions with the instance generated by "faasr"

Usage

```
faasr(json_path = NULL, env_path = NULL)
```

Arguments

json_path	a string for the json path
env_path	a string for the env(credentials) path

Value

svc; a set of data consisting of functions and data

Examples

```
if (interactive()){
test <- faasr(json_path="json_path.json", env_path="env_path")
}
```

faasr_arrow_s3_bucket *faasr_arrow_s3_bucket*

Description

'test' Uses "arrow" library to set up the configurations with given json file and provide the object to the users

Arguments

server_name	for string, default value is faasr$DefaultDataStore
faasr_config	optional configuration to use instead of global .faasr

Value

s3 representing object for "arrow"

Examples

```
# this function can be run only inside the container
if (interactive()){
arrow_s3 <- faasr_arrow_s3_bucket()
arrow_s3$ls
}
```

`faasr_delete_file` *faasr_delete_file*

Description

Helper function to delete a file in an S3 bucket

Arguments

`server_name`	string with name of the S3 bucket to use; must match a name declared in the faasr list
`remote_folder`	string with the name of the remote folder where the file is to be deleted from
`remote_file`	string with the name for the file to be deleted
`faasr_config`	optional configuration to use instead of global .faasr

Value

return nothing / delete the file in the bucket

Examples

```
# This function can be run only in the container
if (interactive()){
faasr_delete_file(remote_file="test.txt")
}
```

`faasr_delete_log` *faasr_delete_log*

Description

deletes a FaaSr log file stored in S3 This function deleted a specific FaaSr log file from the user's S3 FaaSrLog bucket

Usage

```
faasr_delete_log(uuid)
```

Arguments

uuid	the UUID of the log to retrieve

Value

nothing is returned

Examples

```
if (interactive()){
test <- faasr("test.json", "env")
logs <- test$faasr_delete_log('SOME-UUID')
}
```

---

faasr_delete_log_date *faasr_delete_log_date*

Description

deletes all FaaSr logs of a given date This function deleted all FaaSr log files in a given date from the user's S3 FaaSrLog bucket

Usage

```
faasr_delete_log_date(target_date)
```

Arguments

target_date	the date (yyyy-mm-dd) of the log data

Value

nothing is returned

Examples

```
if (interactive()){
test <- faasr("test.json", "env")
logs <- test$faasr_delete_log_date('2024-06-05')
}
```

faasr_get_file *faasr_get_file*

Description

Helper function to download a file from an S3 bucket to local Action folder

Arguments

server_name	string with name of the S3 bucket to use; must match a name declared in the faasr list
remote_folder	string with the name of the remote folder where the file is to be downloaded from
remote_file	string with the name for the file to be downloaded from the S3 bucket
local_folder	string with the name of the local folder where the file to be downloaded is stored
local_file	string with the name of the local file once downloaded
faasr_config	optional configuration to use instead of global .faasr

Value

return nothing / delete the file in the bucket

Examples

```
# This function can be run only in the container
if (interactive()){
faasr_get_file("remote_folder", "remote_file", "local_folder", "local_file")
}
```

faasr_get_folder_list *faasr_get_folder_list*

Description

Helper function to get a list of objects in an S3 bucket

Arguments

server_name	string with name of the S3 bucket to use; must match a name declared in the faasr list
faasr_prefix	string with prefix of objects in the S3 bucket
faasr_config	optional configuration to use instead of global .faasr

Value

object_list a list of objects in the bucket

Examples

```
# This function can be run only in the container
if (interactive()){
faasr_get_folder_list(server_name="My_Minio_Bucket")
}
```

---

faasr_get_log *faasr_get_log*

Description

downloads a FaaSr log file stored in S3 This function downloads a specific FaaSr log file from the user's S3 FaaSrLog bucket to local disk This can be useful in debugging by helping the user retrieve the specific log of a FaaSr run

Usage

```
faasr_get_log(uuid)
```

Arguments

uuid	the UUID of the log to retrieve

Value

nothing is returned; log is downloaded

Examples

```
if (interactive()){
test <- faasr("test.json", "env")
logs <- test$faasr_get_log('SOME-UUID')
}
```

faasr_get_log_df *faasr_get_log_df*

Description

returns a data frame with FaaSr logs stored in S3 This function queries and aggregates FaaSr log information configured with the user's S3 bucket This can be useful in debugging by helping the user identify and download a particular log

Usage

```
faasr_get_log_df()
```

Value

this function returns nothing

Examples

```
if (interactive()){
test <- faasr("test.json", "env")
test$faasr_get_log_df()
}
```

faasr_invoke_workflow *faasr_invoke_workflow*

Description

invoke workflow This function aggregates the invoke workflow function for openwhisk, github actions and lambda This can be used as a cross-platform function

Usage

```
faasr_invoke_workflow(FunctionInvoke = NULL, ...)
```

Arguments

FunctionInvoke	a string for invoke function
...	a string for underlying functions

Value

return nothing / invokes the FaaS platform.

Examples

```
if (interactive()){
test <- faasr("test.json", "env")
test$invoke_workflow
}
```

`faasr_log` *faasr_log*

Description

Helper function to append to the log file residing in an S3 bucket the name of the S3 server is implicit from the validated JSON payload, key LoggingServer the log file is a concatenation of folder "logs" and file name "faasr_log_" + InvocationID + ".txt"

Arguments

`faasr`	list with parsed and validated Payload
`log_message`	string message to be appended to the log

Value

return nothing / leave the log to the bucket

Examples

```
# This function can be run only in the container
if (interactive()){
log_message <- "test message"
faasr_log(log_message)
}
```

`faasr_parse` *faasr_parse*

Description

This function uses JSON parsing and validation to ensure the Payload is compliant Two checks are made here: 1) is it a valid JSON format? and 2) does it conform to the FaaSr JSON schema? If both checks pass, return a list with all the parsed key/value pairs Otherwise, abort

Usage

```
faasr_parse(faasr_payload)
```

Arguments

faasr_payload	JSON Payload provided upon Action invocation by the FaaS platform

Value

faasr list with parsed and validated Payload

Examples

```
# This function can be run only in the container
if (interactive()){
faasr <- faasr_parse(faasr_payload)
}
```

---

faasr_put_file *faasr_put_file*

Description

Helper function to upload a file from a local Action folder to an S3 bucket

Arguments

server_name	string with name of the S3 bucket to use; must match a name declared in the faasr list
local_folder	string with the name of the local folder where the file to be uploaded resides
local_file	string with the name of the local file to be uploaded
remote_folder	string with the name of the remote folder where the file is to be uploaded to
remote_file	string with the name for the file once uploaded to the S3 bucket
faasr_config	optional configuration to use instead of global .faasr

Value

return nothing / put the file into the bucket

Examples

```
# This function can be run only in the container
if (interactive()){
faasr_put_file("local_folder", "local_file", "remote_folder", "remote_file")
}
```

faasr_rank *faasr_rank*

Description

Helper function to let users identify the rank and max rank in the function This function is used by users in the container.

Usage

```
faasr_rank()
```

Value

return a list of Rank & MaxRank

Examples

```
# This function can be run only in the container
if (interactive()){
rank_info <- faasr_rank()
MaxRank <- rank_info$MaxRank
Rank <- rank_info$Rank
}
```

faasr_register_workflow *faasr_register_workflow*

Description

Client tools for registering actions This aggregates openwhisk, github actions and lambda's register functions

Usage

```
faasr_register_workflow(...)
```

Arguments

...	inputs for timeout, cron, and memory

Value

return nothing / executes the FaaS

Examples

```
if (interactive()){
test <- faasr("test.json", "env")
test$register_workflow
}
```

`faasr_replace_values` *faasr_replace_values*

Description

replace dummy keys with real keys Dummy key format: Servername + "_" + Key Name with CAPITAL LETTER e.g., My_OW_Account + API.key = My_OW_Account_API_KEY

Usage

```
faasr_replace_values(faasr, cred)
```

Arguments

`faasr`	a list form of the JSON file
`cred`	a list form of the credentials

Value

faasr a list form of the JSON file with real keys

Examples

```
if (interactive()){
faasr_with_cred <- faasr_replace_values(faasr, cred)
}
```

`faasr_run_user_function` *faasr_run_user_function*

Description

Run user functions and leave the state information

Usage

```
faasr_run_user_function(.faasr)
```

Arguments

.faasr	list with parsed and validated Payload

Value

return nothing / executes the given user function

Examples

```
# This function can be run only in the container
if (interactive()){
faasr_run_user_function(.faasr)
}
```

---

faasr_set_workflow_timer

faasr_set_workflow_timer

Description

set cron timer for workflow This function aggregates the set cron timer function for openwhisk, github actions and lambda This can be used as a cross-platform function

Usage

```
faasr_set_workflow_timer(cron, target = NULL, ...)
```

Arguments

cron	a string for cron data e.g., */5 * * * *
target	a string for specific function
...	a string for underlying functions

Value

return nothing / set the workflow timer

Examples

```
if (interactive()){
test <- faasr("test.json", "env")
test$set_workflow_timer("*/5 * * * *")
}
```

faasr_start *faasr_start*

Description

This is the entry-point FaaSr Function that is invoked by a FaaS platform when an Action is instantiated Terminology to clarify the various modules involved:

* Action: an instance of a Docker container instantiated by a FaaS platform * User Function: a single function written in R; at runtime, it is executed by a single Action * FaaSr Function: function implemented by the FaaSr package to implement all the logic necessary to manage the execution of a User Function within an Action. A FaaSr function has a prefix faasr_ * User Workflow: a graph where each vertex represents a single User Functions and each edge represents a trigger * Payload: a JSON-formatted text file that conforms to the FaaSr schema. It is delivered by the FaaS platform for each Action. It describes the entire User Workflow and may contain credentials for FaaS and S3 services

faasr_start calls other FaaSr Functions to go through the following steps:

* Parse the Payload and ensure that it conforms to the FaaSr JSON schema; otherwise, abort * Build the User Workflow graph from Payload and ensure it is cycle-free; otherwise, abort * Initialize the logs folder in an S3 bucket, only if this is the entry point to the User Workflow * Ensure only a single User Function runs if it has multiple predecessors; otherwise, abort * Invoke the User Function, supplying the parsed payload as a list argument * Update the logs folder to assert that this User Function has completed * Generate triggers to start Actions that will run the next User Functions, if there are any in the User Workflow

Arguments

faasr_payload	JSON Payload provided upon Action invocation by the FaaS platform

Value

faasr a list form of JSON payload

Examples

```
# This function can be run only in the container
if (interactive()){
faasr <- faasr_start(faasr_payload)
}
```

faasr_test *faasr_test*

Description

Client tools for local tests Users can use local test by using both for Docker and local file system

Usage

```
faasr_test(
  use_docker = list(use = FALSE, version = docker_default_version, image =
    docker_default_image)
)
```

Arguments

use_docker	a list of docker configuration - use, version and image.

Value

return nothing / executes the FaaS

Examples

```
if (interactive()){
test <- faasr("test.json", "env")
test$faasr_test()
test$faasr_test(use_docker=list(use=TRUE, version="test", image="docker.io/test/test-docker:image"))
}
```

faasr_trigger *faasr_trigger*

Description

Uses FaaS-specific APIs to generate triggers to execute downstream User Function Currently supports: * Apache OpenWhisk * AWS Lambda * GitHub Actions

Usage

```
faasr_trigger(faasr)
```

Arguments

faasr	list with parsed and validated Payload

Value

return nothing / send requests to the FaaS servers.

Examples

```
# This function can be run only in the container
if (interactive()){
faasr_trigger(faasr)
}
```

`faasr_unset_workflow_timer` *faasr_unset_workflow_timer*

Description

unset cron timer for workflow This function aggregates the unset cron timer function for openwhisk, github actions and lambda This can be used as a cross-platform function

Usage

```
faasr_unset_workflow_timer(target = NULL, ...)
```

Arguments

`target`	a string for specific function
`...`	a string for underlying functions

Value

return nothing / unset the workflow timer

Examples

```
if (interactive()){
test <- faasr("test.json", "env")
test$unset_workflow_timer()
}
```

Index