Package ‘DockerParallel’

June 23, 2021

Title Using the Docker Container to Create R Workers on Local or Cloud Platform

Version 1.0.4

Description This is the core package that provides both the user API and developer API to deploy the parallel cluster on the cloud using the container service. The user can call clusterPreset() to define the cloud service provider and container and makeDockerCluster() to create the cluster. The developer should see ``developer's cookbook” on how to define the cloud provider and container.

Imports methods, utils, jsonlite

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.1

Suggests markdown, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr


NeedsCompilation no

Author Jiefei Wang [aut, cre]

Maintainer Jiefei Wang <szwjf08@gmail.com>

Repository CRAN

Date/Publication 2021-06-23 13:00:02 UTC

R topics documented:

  .getCloudProvider .................................................. 3
  cleanupDockerCluster ............................................. 5
  cleanupDockerCluster,DummyProvider-method .................... 6
  CloudConfig-class .................................................. 6
  CloudPrivateServer ............................................... 7
  CloudProvider-class .............................................. 8
### R topics documented:

- `CloudRuntime-class` .................................................. 8
- `ClusterMethodGetter-class` ........................................ 8
- `clusterPreset` ......................................................... 9
- `configServerContainerEnv` ......................................... 9
- `configWorkerContainerEnv` ......................................... 10
- `DockerCluster-class` ................................................ 11
- `DockerCluster-common-parameters` ............................... 11
- `dockerClusterExists` ............................................... 12
- `dockerClusterExists,DummyProvider-method` ..................... 12
- `DockerContainer-class` .............................................. 13
- `DockerHardware` ....................................................... 13
- `DockerHardware-class` .............................................. 14
- `DummyProvider` ....................................................... 14
- `DummyWorkerContainer` ............................................. 15
- `generalDockerClusterTest` ......................................... 15
- `generics-commonParams` ............................................ 16
- `getDockerServerIp` .................................................. 16
- `getDockerServerIp,DummyProvider-method` ....................... 17
- `getDockerStaticData` ............................................... 18
- `getDockerWorkerNumbers` .......................................... 19
- `getDockerWorkerNumbers,DummyProvider-method` ............... 20
- `getExportedNames` .................................................. 20
- `getServerContainer` ................................................ 21
- `getServerStatus,DummyProvider-method` .......................... 22
- `getServerStatus,DummyProvider-method` .......................... 22
- `getSSHPubKeyPath` .................................................. 23
- `initializeCloudProvider,DummyProvider-method` ............... 24
- `initializeCloudProvider,DummyProvider-method` ............... 25
- `makeDockerCluster` .................................................. 25
- `names,ClusterMethodGetter-method` ................................ 27
- `names,DockerCluster-method` ..................................... 28
- `reconnectDockerCluster` ............................................ 28
- `reconnectDockerCluster,DummyProvider-method` ............... 29
- `registerParallelBackend` .......................................... 30
- `resetDummyProvider` ................................................ 31
- `runDockerServer,DummyProvider-method` .......................... 31
- `runDockerServer,DummyProvider-method` .......................... 31
- `setDockerWorkerNumber,DummyProvider-method` ............... 32
- `setDockerWorkerNumber,DummyProvider-method` ............... 32
- `setSSHPubKeyPath,DummyProvider-method` ....................... 33
- `show,CloudConfig-method` ......................................... 34
- `show,CloudRuntime-method` ....................................... 34
- `show,ClusterMethodGetter-method` ................................ 35
- `show,DockerCluster-method` ...................................... 35
- `show,DockerContainer-method` .................................... 36
- `show,DockerHardware-method` ..................................... 36
- `stopDockerServer,DummyProvider-method` ....................... 37
.getCloudProvider

Description

Accessor functions for the developer.

Usage

.getCloudProvider(cluster)
.getCloudConfig(cluster)
.getServerContainer(cluster)
.getWorkerContainer(cluster)
.getCloudRuntime(cluster)
.getClusterSettings(cluster)
.getVerbose(cluster)
.setCloudProvider(cluster, value)
.setCloudConfig(cluster, value)
.setServerContainer(cluster, value)
.setWorkerContainer(cluster, value)
.setCloudRuntime(cluster, value)
.setClusterSettings(cluster, value)
.setVerbose(cluster, value)
.setStopClusterOnExit(cluster, value)
.getJobQueueName(cluster)
.getExpectedWorkerNumber(cluster)
.getWorkerHardware(cluster)
.getServerHardware(cluster)
.getServerWorkerSameLAN(cluster)
.getServerClientSameLAN(cluster)
.getServerPassword(cluster)
.getServerPort(cluster)
.setJobQueueName(cluster, value)
.setExpectedWorkerNumber(cluster, value)
.setWorkerHardware(cluster, value)
.setServerHardware(cluster, value)
.getServerWorkerSameLAN(cluster, value)
.getServerClientSameLAN(cluster, value)
.setServerPassword(cluster, value)
.getServerPort(cluster, value)
.getServerFromOtherSource(cluster)
.getServerPrivateIp(cluster)
.getServerPrivatePort(cluster)
.getServerPublicIp(cluster)
.getServerPublicPort(cluster)
.getInitializingWorkerNumber(cluster)
.getRunningWorkerNumber(cluster)
.setServerPrivateIp(cluster, value)
.setServerPublicIp(cluster, value)
### cleanupDockerCluster

```
.setServerPrivatePort(cluster, value)
.setServerPublicPort(cluster, value)
.setInitializingWorkerNumber(cluster, value)
.setRunningWorkerNumber(cluster, value)
.setServerFromOtherSource(cluster, value)
```

**Arguments**

- **cluster**: A `DockerCluster` object
- **value**: The value you want to set/add/remove

**Value**

No return value for the setter. The getter will get the object from the cluster.

---

### cleanupDockerCluster

**Description**

Cleanup the resources after the cluster has been stopped. After this function is called, all the non-free resources should be stopped. The cloud provider can still preserve some resources if they are free. This generic might be called multiple times. The default method does nothing.

**Usage**

```
cleanupDockerCluster(provider, cluster, deep, verbose)
```

```r
## S4 method for signature 'ANY'
cleanupDockerCluster(provider, cluster, verbose)
```

**Arguments**

- **provider**: S4 `CloudProvider` object. The service provider.
- **cluster**: S4 `DockerCluster` object.
- **deep**: Logical(1), whether all the associated resources should be removed.
- **verbose**: Integer. The verbose level, default 1.

**Value**

No return value
CloudConfig-class

Functions


cleanupDockerCluster,DummyProvider-method

Create a Dummy provider for testing the container

Description

This function will set the slot cleanup to TRUE

Usage

```r
## S4 method for signature 'DummyProvider'
cleanupDockerCluster(provider, cluster, verbose)
```

Arguments

- provider: S4 CloudProvider object. The service provider.
- cluster: S4 DockerCluster object.
- verbose: Integer. The verbose level, default 1.

Value

No return value

CloudConfig-class

The cloud configuration

Description

The cloud configuration. It is a class purely for storing the information for the cloud. The values in CloudConfig in a cluster can be accessed by the getter function which starts with the prefix `.get(e.g. .getJobQueueName(cluster))`. 

Fields

- jobQueueName: Character(1), the name of the job queue.
- expectedWorkerNumber: Integer(1), the expected number of workers that should be run on the cloud.
- serverHardware: DockerHardware, the server hardware.
- workerHardware: DockerHardware, the worker hardware.
- serverPort: Integer(1) or integer(0), the port that will be used by the worker to connect with the server.
serverPassword Character(1) or character(0), the server password.
serverWorkerSameLAN Logical(1), whether the server and workers are behind the same router.
serverClientSameLAN Logical(1), whether the server and client are behind the same router.

CloudPrivateServer

**Description**

Define the data object for a cloud private server. The data object can be passed to `makeDockerCluster` and let the cluster use the private server instead of the server from the cloud provider.

**Usage**

```r
CloudPrivateServer(
  publicIp = character(0),
  publicPort = integer(0),
  privateIp = character(0),
  privatePort = integer(0),
  password = "",
  serverWorkerSameLAN = FALSE,
  serverClientSameLAN = FALSE
)
```

**Arguments**

- publicIp Character(0) or Character(1), the public Ip of the server
- publicPort Integer(0) or Integer(1), the public port of the server
- privateIp Character(0) or Character(1), the private Ip of the server
- privatePort Integer(0) or Integer(1), the private port of the server
- password Character(1), the password for the server
- serverWorkerSameLAN Logical(1), whether the server and works are in the same LAN
- serverClientSameLAN Logical(1), whether the server and client are in the same LAN

**Examples**

```r
CloudPrivateServer(publicIp = "192.168.1.1", publicPort = 1234)
```
ClusterProvider-class  *The root class of the cloud provider*

Description

The root class of the cloud provider

CloudRuntime-class  *The cloud runtime*

Description

The cloud runtime. It is a class purely for storing the runtime information for the cloud. The values in CloudRuntime in a cluster can be accessed by the getter function which starts with the prefix .get(e.g. .getServerPublicIp(cluster)).

Fields

- **serverFromOtherSource**  Logical(1), whether the server is provided outside of cluster. If TRUE, the cluster will not try to stop the server when it is stopped.
- **serverPublicIp**  Character(1) or character(0), the server public IP.
- **serverPublicPort**  Integer(1) or integer(0), the server public port.
- **serverPrivateIp**  Character(1) or character(0), the server private IP.
- **serverPrivatePort**  Integer(1) or integer(0), the server private port.
- **runningWorkerNumber**  Integer(1), the current initializing workers.
- **runningWorkerNumber**  Integer(1), the current running workers.

ClusterMethodGetter-class  *An utility class*

Description

An utility class for exporting the APIs from the cloud provider and container.
Description

Set the default cloud provider and container. You must install the provider and container packages before using them.

Usage

```r
clusterPreset(
  cloudProvider = c("", "ECSFargateProvider"),
  container = c("", "rbaseDoRedis", "rbaseRedisParam", "biocDoRedis", "biocRedisParam")
)
```

Arguments

- `cloudProvider`: The default cloud provider name, can be abbreviated
- `container`: The default container name, can be abbreviated

Value

No return value

Examples

```r
## Not run:
clusterPreset(cloudProvider = "ECSFargateProvider", container = "rbaseDoRedis")
cluster <- makeDockerCluster()
cluster
## End(Not run)
```

---

configServerContainerEnv

Configure the server container environment

Description

Configure the server container environment. Developers can use this function to set the server password, port number and etc. via the container environment variable. The server info can be found by the getter function with the prefix `.getServer` (e.g. `.getServerPassword(cluster)`). The developer must calls `container$copy()` before setting the server environment. The user provided environment variables should be respected and overwritten only when necessary. There is no default method for this generic.
Usage

configServerContainerEnv(container, cluster, verbose)

## S4 method for signature 'DummyContainer'
configServerContainerEnv(container, cluster, verbose = FALSE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>Reference Container Object. The server container.</td>
</tr>
<tr>
<td>cluster</td>
<td>S4 DockerCluster object.</td>
</tr>
<tr>
<td>verbose</td>
<td>Integer. The verbose level, default 1.</td>
</tr>
</tbody>
</table>

Value

An object which has the same class as container

Functions

- configServerContainerEnv,DummyContainer-method: method for the dummy container

---

configWorkerContainerEnv

Configure the worker container environment

Description

Configure the worker container environment. Developers can use this function to set the server IP, password and etc. via the container environment variable. The server info can be found by the getter function with the prefix .getServer (e.g. .getServerPassword(cluster)). Depending on the network status, the worker can use the server private IP to connect with the server. The developer must calls container$copy() before setting the server environment. The user provided environment variables should be respected and overwritten only when necessary. There is no default method for this generic.

Usage

configWorkerContainerEnv(container, cluster, workerNumber, verbose)

## S4 method for signature 'DummyContainer'
configWorkerContainerEnv(container, cluster, workerNumber, verbose = FALSE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>Reference Container Object. The worker container.</td>
</tr>
<tr>
<td>cluster</td>
<td>S4 DockerCluster object.</td>
</tr>
<tr>
<td>workerNumber</td>
<td>Integer. The number of workers in a container.</td>
</tr>
<tr>
<td>verbose</td>
<td>Integer. The verbose level, default 1.</td>
</tr>
</tbody>
</table>
DockerCluster-class

Value
An object which has the same class as container

Functions
- `configWorkerContainerEnv,DummyContainer-method`: method for the dummy container

DockerCluster-class
The docker cluster class

Description
The docker cluster class. The values in the cluster can be accessed by the getter or setter function which starts with the prefix `.get` or `.set` (e.g. `.getJobQueueName(cluster)`).

Slots
- `cloudProvider`: CloudProvider
- `cloudConfig`: CloudConfig
- `serverContainer`: The container definition for the server.
- `workerContainer`: The container definition for the worker
- `cloudRuntime`: CloudRuntime
- `settings`: Environment, the cluster settings

DockerCluster-common-parameters
Common DockerCluster parameter

Description
Common DockerCluster parameter

Arguments
- `x`: The DockerCluster object
- `name`: Character, the name of the exported object
- `object`: The DockerCluster object

Value
No return value
dockerClusterExists

Whether the cluster is running on the cloud?

Description

The function checks whether the cluster is running on the cloud. It returns TRUE if the cluster specific to the value from .getJobQueueName(cluster) exists. The default method always returns FALSE.

Usage

dockerClusterExists(provider, cluster, verbose)

## S4 method for signature 'ANY'
dockerClusterExists(provider, cluster, verbose)

Arguments

provider S4 CloudProvider object. The service provider.
cluster S4 DockerCluster object.
verbose Integer. The verbose level, default 1.

Value

A logical value

Functions

• dockerClusterExists,ANY-method: The default method, it always returns FALSE.

dockerClusterExists,DummyProvider-method

Create a Dummy provider for testing the container

Description

This function returns TRUE only when the environment variable dummyProvider is equal to the job queue name.

Usage

## S4 method for signature 'DummyProvider'
dockerClusterExists(provider, cluster, verbose)
DockerContainer-class

Arguments

provider S4 CloudProvider object. The service provider.
cluster S4 DockerCluster object.
verbose Integer. The verbose level, default 1.

Value

No return value

DockerContainer-class  The root class of the container

Description

The root class of the container

Fields

name Character(1) or character(0), the optional name of a container.
backend Character(1), the backend used by the parallel package
maxWorkerNum Integer(1), the maximum worker number in a container.
environment List, the environment variables in the container.
image Character(1), the container image.

DockerHardware  Make a DockerHardware object

Description

Make a DockerHardware object

Usage

DockerHardware(cpu = 256, memory = 512, id = character(0))

Arguments

cpu Numeric(1), the CPU limitation for the docker. 1024 CPU unit corresponds to 1 core.
memory Numeric(1), the memory limitation for the docker, the unit is MB.
id character(1) or character(0), the id of the hardware, the meaning of id depends on the cloud provider.
### DummyProvider

**Value**

A DockerHardware object

**Examples**

DockerHardware()

---

**DockerHardware-class**  *The hardware for running the docker*

**Description**

The hardware for running the docker

**Slots**

- **cpu**  Numeric(1), the CPU limitation for the docker. 1024 CPU unit corresponds to 1 core.
- **memory**  Numeric(1), the memory limitation for the docker, the unit is MB.
- **id**  character(1) or character(0), the id of the hardware, the meaning of id depends on the cloud provider.

---

**DummyProvider**  *Create a Dummy provider for testing the container*

**Description**

Create a Dummy provider for testing the container

**Usage**

DummyProvider(initialized = FALSE, isServerRunning = FALSE, cleanup = FALSE)

**Arguments**

- **initialized**, **isServerRunning**, **cleanup**
  
  logical(1), the flags

**Value**

A DummyProvider object

**Examples**

DummyProvider()
**DummyWorkerContainer**

* A dummy container

---

**Description**

A dummy container. It is for purely testing purpose.

**Usage**

```r
dummyWorkerContainer(
    image = "workerImage",
    backend = "testBackend",
    maxWorkerNum = 123L
)
```

```r
dummyServerContainer(image = "serverImage", backend = "testBackend")
```

**Arguments**

- **image**: The image for the container
- **backend**: The parallel backend for the container
- **maxWorkerNum**: The maximum worker number

**Examples**

```r
dummyWorkerContainer()
```

---

**generalDockerClusterTest**

*The general testthat function for testing the cluster*

---

**Description**

The general `testthat` function for testing the cluster. The function should be called by the cloud provider to test the functions in the provider. If `testReconnect` is `TRUE`, the provider must define `reconnectDockerCluster` for making the test function work.

**Usage**

```r
generalDockerClusterTest(
    cloudProvider,
    workerContainer,
    workerNumber = 5L,
    testReconnect = TRUE,
    ...
)
```
**Arguments**

- `cloudProvider` The CloudProvider
- `workerContainer` The workerContainer
- `workerNumber` Integer(1), The number of workers used in the unit test
- `testReconnect` Logical(1), whether to test the reconnect feature
- ... Additional parameters passed to `makeDockerCluster`

**Value**

No return value

---

**getDockerServerIp**

*Get the server IP and port*

**Description**

Get the server public/private IPs. The IPs will be used by the cluster to make connections between server and worker, server and client. If the server does not have the public or private IP, its value can be set to character(0) and port can be set to integer(0). If the IP has not been assigned yet, this function should wait until the IP is available. If the server is not provided by the cloud provider, this function will not be called. There is no default method for this generic. The return value should be a name list with four elements `publicIp`, `publicPort`, `privateIp` and `privatePort`. If the server does not have the public endpoint, public IP and port can be NULL.
Usage

getDockerServerIp(provider, cluster, verbose)

Arguments

provider S4 CloudProvider object. The service provider.
cluster S4 DockerCluster object.
verbose Integer. The verbose level, default 1.

Value

a name list with four elements publicIp, publicPort, privateIp and privatePort.

Description

This function always returns list(publicIp = "8.8.8.8", publicPort = 123, privateIp = "192.168.1.1", privatePort = 456)

Usage

## S4 method for signature 'DummyProvider'
getDockerServerIp(provider, cluster, verbose)

Arguments

provider S4 CloudProvider object. The service provider.
cluster S4 DockerCluster object.
verbose Integer. The verbose level, default 1.

Value

No return value
**getDescription**

get/set docker cluster static data. These functions are designed for the reconnect function for DockerCluster. The return value can be serialized and used by the cloud provider to recover the DockerCluster object. The default method for DockerCluster will use `getDockerStaticData` to get the static data in `cloudConfig`, `ServerContainer`, and `WorkerContainer`.

**Usage**

```r
getDockerStaticData(x)
setDockerStaticData(x, staticData)
```

## S4 method for signature 'CloudConfig'

```r
getDockerStaticData(x)
setDockerStaticData(x, staticData)
```

## S4 method for signature 'DockerCluster'

```r
getDockerStaticData(x)
setDockerStaticData(x, staticData)
```

## S4 method for signature 'DockerContainer'

```r
getDockerStaticData(x)
setDockerStaticData(x, staticData)
```

**Arguments**

- **x**
  - The object which the static data will be extracted from or the object that will hold the unserialized data.

- **staticData**
  - The data returned by `getDockerStaticData`

**Value**

- `getDockerStaticData`: Any data that is serializable
- `setDockerStaticData`: No return value should be expected, the object that is passed to the function will be updated.
**getDockerWorkerNumbers**

**Functions**

- `getDockerStaticData,CloudConfig-method`: The method for `CloudConfig`
- `setDockerStaticData,CloudConfig-method`: The method for `CloudConfig`
- `getDockerStaticData,DockerCluster-method`: The method for `DockerCluster`
- `setDockerStaticData,DockerCluster-method`: The method for `DockerCluster`
- `getDockerStaticData,DockerContainer-method`: The method for `DockerContainer`
- `setDockerStaticData,DockerContainer-method`: The method for `DockerContainer`

---

**getDockerWorkerNumbers**

*Get the worker number on the cloud*

**Description**

Get the worker number on the cloud. Return a list with two elements, which are the number of initializing and running workers. The names must be "initializing" and "running". The default method will return `list(initializing = 0L, running = .getExpectedWorkerNumber(cluster))`.

**Usage**

```r
getDockerWorkerNumbers(provider, cluster, verbose)
## S4 method for signature 'ANY'
getDockerWorkerNumbers(provider, cluster, verbose = 0L)
```

**Arguments**

- `provider` S4 `CloudProvider` object. The service provider.
- `cluster` S4 `DockerCluster` object.
- `verbose` Integer. The verbose level, default 1.

**Value**

`list(initializing = ?, running = ?)`.

**Functions**

- `getDockerWorkerNumbers,ANY-method`: The default `getDockerWorkerNumbers` method. Return `c(0L,.getExpectedWorkerNumber(cluster))`
getDockerWorkerNumbers, DummyProvider-method

Create a Dummy provider for testing the container

Description

This function returns value defined by the environment variable dummyProviderWorkerNumber

Usage

## S4 method for signature 'DummyProvider'
getDockerWorkerNumbers(provider, cluster, verbose)

Arguments

provider S4 CloudProvider object. The service provider.
cluster S4 DockerCluster object.
verbose Integer. The verbose level, default 1.

Value

No return value

getExportedNames Get the exported method and variable from the provider or container

Description

Get the exported method and variable from the provider or container. These methods should be used by the developer to export their APIs to the user. The DockerCluster object will call getExportedNames and getExportedObject and export them to the user.

Usage

getExportedNames(x)
getExportedObject(x, name)

## S4 method for signature 'ANY'
getExportedNames(x)

## S4 method for signature 'ANY'
getExportedObject(x, name)
**getServerContainer**

**Arguments**

- `x`
  - A cloud provider or container object
- `name`
  - The name of the exported object

**Details**

If the exported object is a function, the exported function will be defined in an environment such that the `DockerCluster` object is assigned to the variable `cluster`. In other words, the exported function can use the variable `cluster` without define it. This can be useful if the developer needs to change anything in the cluster without asking the user to provide the `DockerCluster` object. The best practice is to define `cluster` as the function argument, the argument will be removed when the function is exported to the user. The user would not be bothered with the redundant `cluster` argument.

**Value**

- `getExportedNames`: The names of the exported functions or variables
- `getExportedObject`: The exported functions or variable

---

**getServerContainer** *Get the server container from the worker container*

**Description**

Get the server container from the worker container. This function will be called by the `DockerCluster` object when the user only provides a worker container to its constructor. There is no default method defined for this generic.

**Usage**

```r
getServerContainer(workerContainer)
```

## S4 method for signature 'DummyContainer'

```r
getAddress(workerContainer)
```

## S4 method for signature 'ANY'

```r
getAddress(workerContainer)
```

**Arguments**

- `workerContainer`
  - The worker container.

**Value**

A server container
Functions

- `getServerContainer,DummyContainer-method`: method for the dummy container
- `getServerContainer,ANY-method`: The default method throws an error

```
getServerStatus
Get the server status
```

Description

Get the server status, return a character value which must be in one of three values "initializing", "running" or "stopped". The default method always returns "running"

Usage

```
getServerStatus(provider, cluster, verbose)
```

Arguments

- `provider` S4 CloudProvider object. The service provider.
- `cluster` S4 DockerCluster object.
- `verbose` Integer. The verbose level, default 1.

Value

Character(1)

```
getServerStatus,DummyProvider-method
Create a Dummy provider for testing the container
```

Description

This function will return either "running" or "stopped" depending on the slot `isServerRunning`

Usage

```
## S4 method for signature 'DummyProvider'
getServerStatus(provider, cluster, verbose)
```

Arguments

- `provider` S4 CloudProvider object. The service provider.
- `cluster` S4 DockerCluster object.
- `verbose` Integer. The verbose level, default 1.
**getSSHPubKeyPath**

**Value**

No return value

---

**getSSHPubKeyPath** *Get the path to the public ssh key*

**Description**

Get the path to the public ssh key

**Usage**

getSSHPubKeyPath()

**Value**

The path to the public ssh key

**Examples**

getSSHPubKeyPath()

---

**getSSHPubKeyValue** *Get the public ssh key*

**Description**

Get the public ssh key

**Usage**

getSSHPubKeyValue()

**Value**

The public ssh key

**Examples**

getSSHPubKeyValue()
initializeCloudProvider

Initialize the service provider

Description

Initialize the service provider. This function will be called prior to runDockerServer and runDockerWorkers. It is used to initialize the cloud-specific settings (e.g., Initialize the cloud network). The function might be called many times. Developers can cache the cloud status and speed up the initialization process.

Usage

initializeCloudProvider(provider, cluster, verbose)

## S4 method for signature 'ANY'
initializeCloudProvider(provider, cluster, verbose = 0L)

Arguments

- **provider** S4 CloudProvider object. The service provider.
- **cluster** S4 DockerCluster object.
- **verbose** Integer. The verbose level, default 1.

Details

Based on the cloud nature, an initialization process might be required before deploying the container on the cloud. This function will be called by the DockerCluster object before running the server and workers. The default method will do nothing.

Besides initializing the cloud settings, if the server container will be deployed by the cloud provider. The function should call .setServerWorkerSameLAN to inform the DockerCluster object whether the server and the workers are under the same router. If .getServerWorkerSameLAN returns TRUE (default), the worker will connect to the server using the server’s private IP. Otherwise, the server’s public IP will be used.

Although it is possible to change any settings in the cluster object in this function, the best practice is to only initialize provider and the value serverWorkerSameLAN.

Value

No return value

Functions

- initializeCloudProvider, ANY-method: The default cloud initialization method, do nothing.
initializeCloudProvider, DummyProvider-method

Create a Dummy provider for testing the container

Description
This function will set the slot initialized to TRUE

Usage
## S4 method for signature 'DummyProvider'
initializeCloudProvider(provider, cluster, verbose)

Arguments
- provider: S4 CloudProvider object. The service provider.
- cluster: S4 DockerCluster object.
- verbose: Integer. The verbose level, default 1.

Value
No return value

makeDockerCluster Create a docker cluster

Description
Create a docker cluster. The user needs to provide a cloud provider and a worker container to make it work.

Usage
makeDockerCluster(
  cloudProvider = NULL,
  workerContainer = NULL,
  workerNumber = 1,
  workerCpu = 1024,
  workerMemory = 2048,
  workerHardwareId = character(0),
  serverCpu = 256,
  serverMemory = 2048,
  serverHardwareId = character(0),
  jobQueueName = "DockerParallelQueue",
  privateServerData = NULL,
serverContainer = getServerContainer(workerContainer),
stopClusterOnExit = TRUE,
verbose = 1
)

Arguments

cloudProvider A CloudProvider object, the cloud that the container will be deployed
workerContainer A DockerContainer object, the object that defines the worker container
workerNumber Integer, the number of workers in the cluster
serverCpu, workerCpu Integer, the CPU unit used by the server or each worker. 1024 CPU unit corresponds to a physical CPU core.
serverMemory, workerMemory Integer, the memory used by the server or each worker in MB
serverHardwareId, workerHardwareId Character, the ID of the hardware, this argument might be ignored by some cloud providers.
jobQueueName Character, the job queue name used by the cluster to send the job.
privateServerData A data object made from CloudPrivateServer. If this object is provided, the cluster server should be from another source and the cloud provider will not deploy the server container.
serverContainer A DockerContainer object, the object that defines the server container.
stopClusterOnExit Logical, whether to stop the cluster when the cluster has been removed from the R session. The default value is TRUE.
verbose Integer, the verbose level

Details

This is the core function of the DockerParallel package which defines the cluster object. To use the function, you need to at least provide the cloud provider and worker container. Currently we have ECSFargateProvider and BiocFERNContainer, see example.

Value

A DockerCluster object

Examples

## Not run:
## Load the ECS fargate provider
library(ECSFargateProvider)
provider <- ECSFargateProvider()
```r
## Load the bioconductor foreach redis container
container <- BiocFERWorkerContainer()

## Define a cluster with 2 workers,
## each worker use one fourth CPU core and 512 MB memory
cluster <- makeDockerCluster(
  cloudProvider = provider,
  workerContainer = container,
  workerNumber = 2,
  workerCpu = 256, workerMemory = 512)

## Start the cluster
cluster$startCluster()

## rescale the worker number
cluster$setWorkerNumber(4)

## Use foreach to do the parallel computing
library(foreach)
getDoParWorkers()
foreach(x= 1:4)%dopar%{
  Sys.info()
}

## End(Not run)
```

---

**names,** `ClusterMethodGetter-method`  
*Get the exported object names*

**Description**  
Get the exported object names

**Usage**  
```r
## S4 method for signature 'ClusterMethodGetter'
names(x)
```

**Arguments**  
- `x` ClusterMethodGetter object

**Value**  
A vector of object names
names,DockerCluster-method

Show the exported object names

Description
Show the exported object names

Usage
## S4 method for signature 'DockerCluster'
names(x)

Arguments
x The DockerCluster object

Value
A character vector

reconnectDockerCluster

Reconnect to the cluster

Description
Reconnect to the cluster with the same job queue name. It is provider's responsibility to recover the data in the cluster, see details. The default method will do nothing.

Usage
reconnectDockerCluster(provider, cluster, verbose)

## S4 method for signature 'ANY'
reconnectDockerCluster(provider, cluster, verbose)

Arguments
provider S4 CloudProvider object. The service provider.
cluster S4 DockerCluster object.
verbose Integer. The verbose level, default 1.
Details

This function is designed for reconnecting to the same cluster on the cloud from a new DockerCluster object. Since the new object does not have the data used by the old DockerCluster object, it is provider's responsibility to obtain them from the cloud (mostly from the server container).

The data for a DockerCluster object can be extracted by getDockerStaticData() and set by setDockerStaticData(). It is recommended can extract and store the data in the server container during the deployment process and recover the cluster data from the server container when this function is called.

Value

No return value

Functions

**registerParallelBackend**

Register/deregister the parallel backend

**Description**

Register/deregister the parallel backend. These methods will be dispatched based on the *worker* container. The parallel framework depends on the container image. If the container uses the foreach framework, there is no need to define `deregisterParallelBackend` as its default method will deregister the foreach backend. There is no default method defined for `registerParallelBackend`.

**Usage**

```r
registerParallelBackend(container, cluster, verbose, ...)

deregisterParallelBackend(container, cluster, verbose, ...)
```

## S4 method for signature 'DummyContainer'

```r
registerParallelBackend(container, cluster, verbose, ...)

deregisterParallelBackend(container, cluster, verbose, ...)
```

**Arguments**

- `container`: The worker container.
- `cluster`: S4 DockerCluster object.
- `verbose`: Integer. The verbose level, default 1.
- `...`: The additional parameter that will be passed to the registration function.

**Value**

No return value

**Functions**

- `registerParallelBackend,DummyContainer-method`: method for the dummy container
- `deregisterParallelBackend,DummyContainer-method`: method for the dummy container
resetDummyProvider  

reset the dummy provider

Description
reset the dummy provider and remove all the environment variables it defined.

Usage
resetDummyProvider()

Value
No return value

Examples
resetDummyProvider()

runDockerServer  

Run or stop the server container

Description
Run or stop the server. These functions will not be called if the server is not managed by the provider. There is no default method for these generics.

Usage
runDockerServer(provider, cluster, container, hardware, verbose)

stopDockerServer(provider, cluster, verbose)

Arguments

provider  S4 CloudProvider object. The service provider.
cluster    S4 DockerCluster object.
container  S4 DockerContainer Object. The server container.
hardware  S4 DockerHardware Object. The server hardware.
verbose   Integer. The verbose level, default 1.

Value
No return value, if error occurs, the function can throw an error.
runDockerServer,DummyProvider-method

Create a Dummy provider for testing the container

Description

This function will set the slot isServerRunning to TRUE and cleanup to FALSE. It also adds the environment variable dummyProvider and dummyProviderClusterData.

Usage

```r
## S4 method for signature 'DummyProvider'
runDockerServer(provider, cluster, container, hardware, verbose)
```

Arguments

- `provider`: S4 CloudProvider object. The service provider.
- `cluster`: S4 DockerCluster object.
- `container`: S4 DockerContainer Object. The server container.
- `hardware`: S4 DockerHardware Object. The server hardware.
- `verbose`: Integer. The verbose level, default 1.

Value

No return value

setDockerWorkerNumber

Set the worker number on the cloud. There is no default method for this generic.

Description

Set the worker number on the cloud. The provider needs to scale the worker number up and down accordingly.

Usage

```r
setDockerWorkerNumber(
  provider,
  cluster,
  container,
  hardware,
  workerNumber,
  verbose
)
```
Arguments

provider  S4 CloudProvider object. The service provider.
cluster   S4 DockerCluster object.
container S4 DockerContainer Object.
hardware  S4 DockerHardware Object.
workerNumber  Integer(1), the number of the workers.
verbose   Integer. The verbose level, default 1.

Value

No return value

Description

This function will set the environment variable dummyProviderWorkerNumber and stores its container in the slot workerContainer.

Usage

```r
## S4 method for signature 'DummyProvider'
setDockerWorkerNumber(
  provider, 
  cluster, 
  container, 
  hardware, 
  workerNumber, 
  verbose 
)
```

Arguments

provider  S4 CloudProvider object. The service provider.
cluster   S4 DockerCluster object.
container S4 DockerContainer Object.
hardware  S4 DockerHardware Object.
workerNumber  Integer(1), the number of the workers.
verbose   Integer. The verbose level, default 1.

Value

No return value
**setSSHPubKeyPath**  
*Set the ssh key file*

**Description**

Set the ssh key file. This function will be called when the package is loaded. If no argument is provided and the current stored path is NULL, it will look at the environment variables DockerParallelSSHPublicKeypath.

**Usage**

```r
setSSHPubKeyPath(publicKey = NULL)
```

**Arguments**

- **publicKey**  
  path to the public key

**Value**

The path to the public key

**Examples**

```r
## Getting the path from the environment variable "DockerParallelSSHPublicKey"
setSSHPubKeyPath()
```

---

**show,CloudConfig-method**

*Print the CloudConfig*

**Description**

Print the CloudConfig

**Usage**

```r
## S4 method for signature 'CloudConfig'
show(object)
```

**Arguments**

- **object**  
  The CloudConfig object

**Value**

No return value
print, CloudRuntime-method

Print the cloudRuntime

Description
Print the cloudRuntime

Usage
## S4 method for signature 'CloudRuntime'
show(object)

Arguments
object The cloudRuntime object

Value
No return value

print, ClusterMethodGetter-method

print method

Description
print method

Usage
## S4 method for signature 'ClusterMethodGetter'
show(object)

Arguments
object ClusterMethodGetter object

Value
No return value
show,DockerCluster-method

Print the DockerCluster object

Description

Print the DockerCluster object

Usage

```r
## S4 method for signature 'DockerCluster'
show(object)
```

Arguments

- `object`: The DockerCluster object

Value

No return value

show,DockerContainer-method

Show the docker container

Description

Show the docker container

Usage

```r
## S4 method for signature 'DockerContainer'
show(object)
```

Arguments

- `object`: The DockerContainer object

Value

No return value
**show,DockerHardware-method**

*Print the docker hardware*

**Description**

Print the docker hardware

**Usage**

```r
## S4 method for signature 'DockerHardware'
show(object)
```

**Arguments**

- `object` The DockerHardware object

**Value**

No return value

**Examples**

```r
hardware <- DockerHardware()
show(hardware)
```

**stopDockerServer,DummyProvider-method**

*Create a Dummy provider for testing the container*

**Description**

This function will set the slot isServerRunning to FALSE

**Usage**

```r
## S4 method for signature 'DummyProvider'
stopDockerServer(provider, cluster, verbose)
```

**Arguments**

- `provider` S4 CloudProvider object. The service provider.
- `cluster` S4 DockerCluster object.
- `verbose` Integer. The verbose level, default 1.

**Value**

No return value
$\text{ClusterMethodGetter-method}$

*Get the exported object by the name*

**Description**

Get the exported object by the name

**Usage**

```r
## S4 method for signature 'ClusterMethodGetter'
x$name
```

**Arguments**

- `x`: ClusterMethodGetter object
- `name`: Character name

**Value**

The exported object

$\text{DockerCluster-method}$

*Get the exported object*

**Description**

Get the exported object

**Usage**

```r
## S4 method for signature 'DockerCluster'
x$name
```

**Arguments**

- `x`: The DockerCluster object
- `name`: Character, the name of the exported object

**Value**

The object in the cluster
\$<-\,\texttt{DockerCluster-method}

*Set the value of the exported object*

---

**Description**

Set the value of the exported object

**Usage**

```r
## S4 replacement method for signature 'DockerCluster'
x$name <- value
```

**Arguments**

- `x`: The DockerCluster object
- `name`: Character, the name of the exported object
- `value`: The value of the exported object

**Value**

The DockerCluster object
Index

.CloudConfig (CloudConfig-class), 6
.CloudProvider (CloudProvider-class), 8
.CloudRuntime (CloudRuntime-class), 8
.ClusterMethodGetter (ClusterMethodGetter-class), 8
.DockerCluster (DockerCluster-class), 11
.DockerContainer (DockerContainer-class), 13
.DockerHardware (DockerHardware-class), 14

.getCloudConfig (.getCloudProvider), 3
.getCloudProvider, 3
.getCloudRuntime (.getCloudProvider), 3
.getClusterSettings (.getCloudProvider), 3
.getExpectedWorkerNumber (.getCloudProvider), 3
.getInitializingWorkerNumber (.getCloudProvider), 3
.getJobQueueName (.getCloudProvider), 3
.getRunningWorkerNumber (.getCloudProvider), 3
.getServerClientSameLAN (.getCloudProvider), 3
.getServerContainer (.getCloudProvider), 3
.getServerFromOtherSource (.getCloudProvider), 3
.getServerHardware (.getCloudProvider), 3
.getServerPassword (.getCloudProvider), 3
.getServerPort (.getCloudProvider), 3
.getServerPrivateIp (.getCloudProvider), 3
.getServerPrivatePort (.getCloudProvider), 3
.getServerPublicIp (.getCloudProvider), 3
.getServerPublicPort (.getCloudProvider), 3
.getServerWorkerSameLAN (.getCloudProvider), 3
.getStopClusterOnExit (.getCloudProvider), 3
.getVerbose (.getCloudProvider), 3
.setCloudConfig (.getCloudProvider), 3
.setCloudProvider (.getCloudProvider), 3
.setCloudRuntime (.getCloudProvider), 3
.setClusterSettings (.getCloudProvider), 3
.setExpectedWorkerNumber (.getCloudProvider), 3
.setInitializingWorkerNumber (.getCloudProvider), 3
.setJobQueueName (.getCloudProvider), 3
.setRunningWorkerNumber (.getCloudProvider), 3
.setServerClientSameLAN (.getCloudProvider), 3
.setServerContainer (.getCloudProvider), 3
.setServerFromOtherSource (.getCloudProvider), 3
.setServerHardware (.getCloudProvider), 3
.setServerPassword (.getCloudProvider), 3
.setServerPort (.getCloudProvider), 3
.setServerPrivateIp (.getCloudProvider), 3
.setServerPrivatePort (.getCloudProvider), 3
.setServerPublicIp (.getCloudProvider), 3
.setServerPublicPort (.getCloudProvider), 3

3
 overly_server_public_port
.setServerPublicPort
 (.getCloudProvider), 3

 overly_server_worker_same_lan
.setServerWorkerSameLAN
 (.getCloudProvider), 3

 overly_stop_cluster_on_exit
.setStopClusterOnExit
 (.getCloudProvider), 3

 overly_verbose
.setVerbose
 (.getCloudProvider), 3

 overly_worker_container
.setWorkerContainer
 (.getCloudProvider), 3

 overly_worker_hardware
.setWorkerHardware
 (.getCloudProvider), 3

 $, ClusterMethodGetter-method, 38
 $, DockerCluster-method, 38
 $<-, DockerCluster-method, 39

 cleanupDockerCluster, 5
 cleanupDockerCluster, ANY-method
 (cleanupDockerCluster), 5

 cleanupDockerCluster, DummyProvider-method, 6

 CloudConfig-class, 6
 CloudPrivateServer, 7
 CloudProvider-class, 8
 CloudRuntime-class, 8
 ClusterMethodGetter-class, 8
 clusterPreset, 9
 configServerContainerEnv, 9
 configServerContainerEnv, DummyContainer-method
 (configServerContainerEnv), 9
 configWorkerContainerEnv, 10
 configWorkerContainerEnv, DummyContainer-method
 (configWorkerContainerEnv), 10

deregisterParallelBackend
 (deregisterParallelBackend), 30

deregisterParallelBackend, DummyContainer-method
 (deregisterParallelBackend), 30

 DockerCluster-class, 11
 DockerCluster-common-parameters, 11
dockerClusterExists, 12
dockerClusterExists, ANY-method
 (dockerClusterExists), 12
dockerClusterExists, DummyProvider-method, 12

 DockerContainer-class, 13
 DockerHardware, 13
 DockerHardware-class, 14

 DummyProvider, 14

 DummyServerContainer
 (DummyServerContainer), 15
 DummyWorkerContainer, 15

 generalDockerClusterTest, 15
generics-commonParams, 16

getDockerServerIp, 16
getDockerServerIp, DummyProvider-method, 17

 getDockerStaticData, 18
getDockerStaticData, CloudConfig-method
 (getDockerStaticData), 18

 getDockerStaticData, DockerCluster-method
 (getDockerStaticData), 18
getDockerStaticData, DockerContainer-method
 (getDockerStaticData), 18

 getDockerWorkerNumbers, 19
getDockerWorkerNumbers, ANY-method
 (getDockerWorkerNumbers), 19
getDockerWorkerNumbers, DummyProvider-method, 20

 getExportedNames, 20
getExportedNames, ANY-method
 (getExportedNames), 20
getExportedObject (getExportedNames), 20
getExportedObject, ANY-method
 (getExportedNames), 20

 getServerContainer, 21
getServerContainer, ANY-method
 (getServerContainer), 21
getServerContainer, DummyContainer-method
 (getServerContainer), 21
getServerStatus, 22
getServerStatus, DummyProvider-method, 22

 getSSHPubKeyPath, 23
getSSHPubKeyValue, 23

 initializeCloudProvider, 24
initializeCloudProvider, ANY-method
 (initializeCloudProvider), 24
initializeCloudProvider, DummyProvider-method, 25

 makeDockerCluster, 25

 names, ClusterMethodGetter-method, 27
 names, DockerCluster-method, 28

 reconnectDockerCluster, 28
reconnectDockerCluster, ANY-method
  (reconnectDockerCluster), 28
reconnectDockerCluster, DummyProvider-method, 29
registerParallelBackend, 30
registerParallelBackend, DummyContainer-method
  (registerParallelBackend), 30
resetDummyProvider, 31
runDockerServer, 31
runDockerServer, DummyProvider-method, 32

setDockerStaticData
  (getDockerStaticData), 18
setDockerStaticData, CloudConfig-method
  (getDockerStaticData), 18
setDockerStaticData, DockerCluster-method
  (getDockerStaticData), 18
setDockerStaticData, DockerContainer-method
  (getDockerStaticData), 18
setDockerWorkerNumber, 32
setDockerWorkerNumber, DummyProvider-method, 33
setSSHPubKeyPath, 34
show, CloudConfig-method, 34
show, CloudRuntime-method, 35
show, ClusterMethodGetter-method, 35
show, DockerCluster-method, 36
show, DockerContainer-method, 36
show, DockerHardware-method, 37
stopDockerServer (runDockerServer), 31
stopDockerServer, DummyProvider-method, 37