Package 'ManagedCloudProvider'

January 20, 2025

Type Package

Title Providing the Kubernetes-Like Functions for the Non-Kubernetes Cloud Service

Version 1.0.0

Description Providing the kubernetes-like class 'ManagedCloudProvider' as a child class of the 'CloudProvider' class in the 'DockerParallel' package. The class is able to manage the cloud instance made by the non-kubernetes cloud service. For creating a provider for the non-kubernetes cloud service, the developer needs to define a reference class inherited from 'ManagedCloudProvider' and define the method for the generics runDockerWorkerContainers(), getDockerWorkerStatus() and killDockerWorkerContainers(). For more information, please see the vignette in this package and <https://CRAN.R-project.org/package=DockerParallel>.

Imports DockerParallel(>= 1.0.3), adagio, methods, utils, jsonlite

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.1

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

VignetteBuilder knitr

URL https://github.com/Jiefei-Wang/ManagedCloudProvider

BugReports https://github.com/Jiefei-Wang/ManagedCloudProvider/issues

NeedsCompilation no

Author Jiefei Wang [aut, cre]

Maintainer Jiefei Wang <szwjf08@gmail.com>

Repository CRAN

Date/Publication 2021-06-14 07:30:18 UTC

Contents

`addManagedWorkerHandles` *Add or get the worker container handles to the managed cloud provider*

Description

Add or get the worker container handles to the managed cloud provider. The handles can be duplicated if multiple workers share the same container

Usage

```
addManagedWorkerHandles(provider, handles)

getManagedWorkerHandles(provider)
```

Arguments

`provider`	A `ManagedCloudProvider` object
`handles`	the worker container handles

Value

addManagedWorkerHandles: No return value getManagedWorkerHandles: A character vector of the worker handles

Examples

```
## make a dummy provider
DummyProvider <- setRefClass("DummyProvider", contains = "ManagedCloudProvider")
provider <- DummyProvider()

## No worker handle in the provider
getManagedWorkerHandles(provider)

## Add worker handles
addManagedWorkerHandles(provider, c("a", "b"))
getManagedWorkerHandles(provider)

## It is possible to add the same handle
## if multiple workers share the same container
```

```
addManagedWorkerHandles(provider, c("a"))
getManagedWorkerHandles(provider)
```

`getDockerWorkerNumbers,ManagedCloudProvider-method` *get the worker number in the cluster*

Description

The developer should define `runDockerWorkerContainers`, `getDockerWorkerStatus` and `killDockerWorkerContainers` instead.

Usage

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

Arguments

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

Value

A list with `initializing` and `running` integer elements

`getDockerWorkerStatus` *Get the worker status*

Description

Get the worker status. Unless you have a faster implementation, you only need to define `getDockerWorkerStatus`. The function should return a character vector with each element corresponding to a worker in `workerHandles`. Each element must be one of three possible characters `"initializing"`, `"running"` or `"stopped"`. There is no default method for `getDockerWorkerStatus`.

Usage

```
getDockerWorkerStatus(provider, cluster, workerHandles, verbose)

IsDockerWorkerInitializing(provider, cluster, workerHandles, verbose)

IsDockerWorkerRunning(provider, cluster, workerHandles, verbose)

IsDockerWorkerStopped(provider, cluster, workerHandles, verbose)

## S4 method for signature 'DummyManagedProvider'
getDockerWorkerStatus(provider, cluster, workerHandles, verbose)

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

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

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

Arguments

provider	S4 CloudProvider object. The service provider.
cluster	S4 DockerCluster object.
workerHandles	Character(n). A character vector of **unique** instance handles.
verbose	Integer. The verbose level, default 1.

Value

getDockerWorkerStatus : A character vector with each element corresponding to an instance in workerHandles. Each element must be one of three possible characters "initializing", "running" or "stopped"

IsDockerWorkerInitializing, IsDockerWorkerRunning, IsDockerWorkerStopped: A logical vector with each element corresponding to the status of each instance

Functions

- getDockerWorkerStatus,DummyManagedProvider-method: The method for the dummy managed provider

killDockerWorkerContainers *Kill the worker container*

Description

Kill the worker container. The worker handles are unique. If multiple workers share the same instance, all workers in the same container should be killed. There is no default method for this generic.

Usage

```
killDockerWorkerContainers(provider, cluster, workerHandles, verbose)

## S4 method for signature 'DummyManagedProvider'
killDockerWorkerContainers(provider, cluster, workerHandles, verbose)
```

Arguments

provider	S4 CloudProvider object. The service provider.
cluster	S4 DockerCluster object.
workerHandles	Character(n). A character vector of **unique** instance handles.
verbose	Integer. The verbose level, default 1.

Value

A logical vector indicating whether the killing operation is successful for each instance

Functions

- killDockerWorkerContainers,DummyManagedProvider-method: The method for the dummy managed provider

ManagedCloudProvider-class *The root class for the managed cloud provider*

Description

The root class for the managed cloud provider

Fields

`serverHandle` Character(1), the server handle that can be recognized by the cloud provider.

`workerHandles` Character(n), a list object for internal use only. Please call `.getWorkerHandles(cluster)` to access the worker handles

`workerPerHandle` An internal counter. Please call `.getWorkerHandles(cluster)` to access the worker handles

---

`runDockerWorkerContainers` *Run the worker container*

Description

Run the workers and return a character vector of the worker handles. Each handle must correspond to a container. The handle can be duplicated if multiple workers share the same container. There is no default method for this generic.

Usage

```
runDockerWorkerContainers(
  provider,
  cluster,
  container,
  hardware,
  workerNumber,
  verbose
)

## S4 method for signature 'DummyManagedProvider'
runDockerWorkerContainers(
  provider,
  cluster,
  container,
  hardware,
  workerNumber,
  verbose
)
```

Arguments

`provider`	S4 `CloudProvider` object. The service provider.
`cluster`	S4 `DockerCluster` object.
`container`	S4 `DockerContainer` Object. The worker container.
`hardware`	S4 `DockerHardware` Object. The worker hardware.
`workerNumber`	Integer. The number of workers needs to be run.
`verbose`	Integer. The verbose level, default 1.

Value

A character vector with each element corresponding to a worker container. The length must be equal to `workerNumber`

Functions

- `runDockerWorkerContainers,DummyManagedProvider-method`: The method for the dummy managed provider

`setDockerWorkerNumber,ManagedCloudProvider-method`
Set the worker number in the cluster

Description

The developer should define `runDockerWorkerContainers`, `getDockerWorkerStatus` and `killDockerWorkerContainers` instead.

Usage

```
## S4 method for signature 'ManagedCloudProvider'
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 worker number to be set
`verbose`	Integer. The verbose level, default 1.

Value

No return value

Index