Package 'AzureContainers'

February 14, 2019

Title Interface to 'Container Instances', 'Docker Registry' and 'Kubernetes' in 'Azure'

Version 1.0.1

Description An interface to container functionality in Microsoft's 'Azure' cloud: <https://azure.microsoft.com/en-us/overview/containers/>. Manage 'Azure Container Instance' (ACI), 'Azure Container Registry' (ACR) and 'Azure Kubernetes Service' (AKS) resources, push and pull images, and deploy services. On the client side, lightweight shells to the 'docker', 'kubectl' and 'helm' command-line tools are provided.

URL https://github.com/cloudyr/AzureContainers

BugReports https://github.com/cloudyr/AzureContainers/issues

License MIT + file LICENSE

VignetteBuilder knitr

Depends R (>= 3.3)

Imports AzureRMR, openssl, httr, R6

Suggests knitr, testthat

RoxygenNote 6.1.0.9000

NeedsCompilation no

Author Hong Ooi [aut, cre],
Bill Liang [ctb] (Assistance debugging MMLS on Kubernetes),
Ramkumar Chandrasekaran [ctb] (Original blog article on Dockerising MMLS),
Microsoft [cph]

Maintainer Hong Ooi <hongooi@microsoft.com>

Repository CRAN

Date/Publication 2019-02-14 05:10:03 UTC

R topics documented:

`aci` *Azure Container Instance class*

Description

Class representing an Azure Container Instance (ACI) resource.

Usage

```
aci
```

Format

An object of class `R6ClassGenerator` of length 24.

Methods

The following methods are available, in addition to those provided by the AzureRMR::az_resource class:

- `new(...)`: Initialize a new ACI object.
- `restart()`, `start()`: Start a stopped container. These methods are synonyms for each other.
- `stop()`: Stop a container.

Details

Initializing a new object of this class can either retrieve an existing ACI resource, or create a new resource on the host. Generally, the best way to initialize an object is via the `get_aci`, `create_aci` or `list_acis` methods of the az_resource_group class, which handle the details automatically.

See Also

acr, aks

ACI documentation and API reference

Docker commandline reference

Examples

```
## Not run:

# recommended way of retrieving a container: via a resource group object
rg <- AzureRMR::az_rm$
    new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")$
    get_subscription("subscription_id")$
    get_resource_group("rgname")

myaci <- rg$get_aci("mycontainer")

myaci$stop()
myaci$restart()


## End(Not run)
```

`aci_ports` *Utilities for specifying ACI configuration information*

Description

Utilities for specifying ACI configuration information

Usage

```
aci_ports(port = c(80L, 443L), protocol = "TCP")

aci_creds(server, username, password)

get_aci_credentials_list(lst)
```

Arguments

`port, protocol`	For `aci_ports`, vectors of the port numbers and protocols to open for the instance.
`server, username, password`	For `aci_creds`, the authentication details for a Docker registry.
`lst`	for `get_aci_credentials_list`, a list of objects.

Details

These are helper functions to be used in specifying the configuration for a container instance. Only `aci_ports` and `aci_creds` are meant to be called by the user; `get_aci_credentials_list` is exported to workaround namespacing issues on startup.

`acr` *Azure Container Registry class*

Description

Class representing an Azure Container Registry (ACR) resource. For working with the registry endpoint itself, including uploading and downloading images etc, see docker_registry.

Usage

```
acr
```

Format

An object of class `R6ClassGenerator` of length 24.

Methods

The following methods are available, in addition to those provided by the AzureRMR::az_resource class:

- `new(...)`: Initialize a new ACR object. See 'Details'.
- `list_credentials`: Return the username and passwords for this registry. Only valid if the Admin user for the registry has been enabled.
- `list_policies`: Return the policies for this registry.
- `list_usages`: Return the usage for this registry.
- `get_docker_registry(username, password)`: Return an object representing the Docker registry endpoint.

Details

Initializing a new object of this class can either retrieve an existing registry resource, or create a new registry on the host. Generally, the best way to initialize an object is via the `get_acr`, `create_acr` or `list_acrs` methods of the az_resource_group class, which handle the details automatically.

Note that this class is separate from the Docker registry itself. This class exposes methods for working with the Azure resource: listing credentials, updating resource tags, updating and deleting the resource, and so on.

For working with the registry, including uploading and downloading images, updating tags, deleting layers and images etc, use the endpoint object generated with `get_docker_registry`. This method takes two optional arguments:

- `username`: The username that Docker will use to authenticate with the registry.
- `password`: The password that Docker will use to authenticate with the registry.

By default, these arguments will be retrieved from the ACR resource. They will only exist if the resource was created with `admin_user_enabled=TRUE`. Currently AzureContainers does not support authentication methods other than a username/password combination.

See Also

create_acr, get_acr, delete_acr, list_acrs

docker_registry for interacting with the Docker registry endpoint

Azure Container Registry and API reference

Examples

```
## Not run:

# recommended way of retrieving a registry: via a resource group object
rg <- AzureRMR::az_rm$
    new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")$
    get_subscription("subscription_id")$
    get_resource_group("rgname")

myacr <- rg$get_acr("myregistry")

myacr$list_credentials()
myacr$list_policies()

# get the registry endpoint
dockerreg <- myacr$get_docker_registry()


## End(Not run)
```

aks *Azure Kubernetes Service class*

Description

Class representing an Azure Kubernetes Service (AKS) resource. For working with the cluster endpoint itself, including deploying images, creating services etc, see kubernetes_cluster.

Usage

```
aks
```

Format

An object of class `R6ClassGenerator` of length 24.

Methods

The following methods are available, in addition to those provided by the AzureRMR::az_resource class:

- `new(...)`: Initialize a new AKS object.
- `get_cluster(config, role)`: Return an object representing the Docker registry endpoint.

Details

Initializing a new object of this class can either retrieve an existing AKS resource, or create a new resource on the host. Generally, the best way to initialize an object is via the `get_aks`, `create_aks` or `list_aks` methods of the az_resource_group class, which handle the details automatically.

Note that this class is separate from the Kubernetes cluster itself. This class exposes methods for working with the Azure resource: updating resource tags, updating and deleting the resource (including updating the Kubernetes version), and so on.

For working with the cluster, including deploying images, services, etc use the object generated with the `get_cluster` method. This method takes two optional arguments:

- `config`: The file in which to store the cluster configuration details. By default, this will be located in the R temporary directory. To use the Kubernetes default `~/.kube/config` file, set this argument to NULL. Note that any existing file in the given location will be overwritten.
- `role`: This can be `"User"` (the default) or `"Admin"`.

See Also

create_aks, get_aks, delete_aks, list_aks

kubernetes_cluster for interacting with the cluster endpoint

AKS documentation and API reference

Examples

```
## Not run:

# recommended way of retrieving a cluster: via a resource group object
rg <- AzureRMR::az_rm$
    new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")$
    get_subscription("subscription_id")$
    get_resource_group("rgname")

myaks <- rg$get_aks("mycluster")

# sync with Azure: AKS resource creation can take a long time, use this to track status
myaks$sync_fields()

# get the cluster endpoint
kubclus <- myaks$get_cluster()


## End(Not run)
```

`aks_pools` *Utility function for specifying Kubernetes agent pools*

Description

Utility function for specifying Kubernetes agent pools

Usage

```
aks_pools(name, count, size = "Standard_DS2_v2", os = "Linux")
```

Arguments

`name`	The name(s) of the pool(s).
`count`	The number of nodes per pool.
`size`	The VM type (size) to use for the pool. To see a list of available VM sizes, use the list_vm_sizes method for the resource group or subscription classes.
`os`	The operating system to use for the pool. Can be "Linux" or "Windows".

Details

This is a convenience function to simplify the task of specifying the agent pool for a Kubernetes cluster. You can specify multiple pools by providing vectors as input arguments; any scalar inputs will be replicated to match.

Value

A list of lists, suitable for passing to the `create_aks` constructor method.

See Also

list_vm_sizes

Examples

```
# 1 pool of 5 Linux VMs
aks_pools("pool1", 5)

# 1 pool of 3 Windows Server VMs
aks_pools("pool1", 3, os="Windows")

# 2 pools with different VM sizes per pool
aks_pools(c("pool1", "pool2"), count=c(3, 3), size=c("Standard_DS2_v2", "Standard_DS3_v2"))
```

---

call_docker *Call the docker commandline tool*

Description

Call the docker commandline tool

Usage

```
call_docker(cmd = "", ...)
```

Arguments

cmd	The docker command line to execute.
...	Other arguments to pass to system2.

Details

This function calls the `docker` binary, which must be located in your search path. AzureContainers will search for the binary at package startup, and print a warning if it is not found.

Value

By default, the return code from the `docker` binary. The return value will have an added attribute `cmdline` that contains the command line. This makes it easier to construct scripts that can be run outside R.

See Also

system2, call_kubectl for the equivalent interface to the `kubectl` Kubernetes tool

docker_registry

Docker command line reference

Examples

```
## Not run:

# without any args, prints the docker help screen
call_docker()

# build an image
call_docker("build -t myimage .")

# list running containers
call_docker("container ls")

# prune unused containers and images
call_docker("container prune -f")
call_docker("image prune -f")


## End(Not run)
```

`call_helm` *Call the Helm commandline tool*

Description

Call the Helm commandline tool

Usage

```
call_helm(cmd = "", ...)
```

Arguments

`cmd`	The Helm command line to execute.
`...`	Other arguments to pass to system2.

Details

This function calls the `helm` binary, which must be located in your search path. AzureContainers will search for the binary at package startup, and print a warning if it is not found.

Value

By default, the return code from the `helm` binary. The return value will have an added attribute `cmdline` that contains the command line. This makes it easier to construct scripts that can be run outside R.

See Also

system2, call_docker, call_kubectl

kubernetes_cluster

Kubectl command line reference

`call_kubectl` *Call the Kubernetes commandline tool, kubectl*

Description

Call the Kubernetes commandline tool, kubectl

Usage

```
call_kubectl(cmd = "", ...)
```

Arguments

`cmd`	The kubectl command line to execute.
`...`	Other arguments to pass to system2.

Details

This function calls the `kubectl` binary, which must be located in your search path. AzureContainers will search for the binary at package startup, and print a warning if it is not found.

Value

By default, the return code from the `kubectl` binary. The return value will have an added attribute `cmdline` that contains the command line. This makes it easier to construct scripts that can be run outside R.

See Also

system2, call_docker, call_helm

kubernetes_cluster

Kubectl command line reference

Examples

```
## Not run:

# without any args, prints the kubectl help screen
call_kubectl()

# append "--help" to get help for a command
call_kubectl("create --help")

# deploy a service from a yaml file
call_kubectl("create -f deployment.yaml")

# get deployment and service status
call_kubectl("get deployment")
call_kubectl("get service")


## End(Not run)
```

`create_aci` *Create Azure Container Instance (ACI)*

Description

Method for the AzureRMR::az_resource_group class.

Usage

```
create_aci(name, location = self$location,
           container = name, image,
           registry_creds = list(),
           cores = 1, memory = 8,
           os = c("Linux", "Windows"),
           command = list(), env_vars = list(),
           ports = aci_ports(), dns_name = name, public_ip = TRUE,
           restart = c("Always", "OnFailure", "Never"), ...)
```

Arguments

- `name`: The name of the ACI service.
- `location`: The location/region in which to create the ACI service. Defaults to this resource group's location.
- `container`: The name of the running container.
- `image`: The name of the image to run.
- `registry_creds`: Docker registry authentication credentials, if the image is stored in a private registry. See 'Details'.
- `cores`: The number of CPU cores for the instance.

- `memory`: The memory size in GB for the instance.
- `os`: The operating system to run in the instance.
- `command`: A list of commands to run in the instance.
- `env_vars`: A list of name-value pairs to set as environment variables in the instance.
- `ports`: The network ports to open. By default, opens ports 80 and 443. See 'Details'.
- `dns_name`: The domain name prefix for the instance. Only takes effect if `public_ip=TRUE`.
- `public_ip`: Whether the instance should be publicly accessible.
- `restart`: Whether to restart the instance should an event occur.
- `...`: Other named arguments to pass to the az_resource initialization function.

Details

An ACI resource is a running container hosted in Azure. See the documentation for the resource for more information. Currently ACI only supports a single image in an instance.

To supply the registry authentication credentials, the `registry_creds` argument should contain either an ACR object, a docker_registry object, or the result of a call to the aci_creds function.

The ports to open should be obtained by calling the aci_ports function. This takes a vector of port numbers as well as the protocol (TCP or UDP) for each port.

Value

An object of class `az_container_instance` representing the instance.

See Also

get_aci, delete_aci, list_acis

az_container_instance

ACI documentation and API reference

Docker commandline reference

Examples

```
## Not run:

rg <- AzureRMR::az_rm$
    new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")$
    get_subscription("subscription_id")$
    get_resource_group("rgname")

# get the ACR resource that contains the image
myacr <- rg$get_acr("myregistry")

rg$create_aci("mycontainer",
    image_name="myregistry.azurecr.io/myimage:latest",
    registry_creds=myacr)


## End(Not run)
```

`create_acr` *Create Azure Container Registry (ACR)*

Description

Method for the AzureRMR::az_resource_group class.

Usage

```
create_acr(name, location = self$location,
           admin_user_enabled = TRUE, sku = "Standard", ...)
```

Arguments

- `name`: The name of the container registry.
- `location`: The location/region in which to create the container registry. Defaults to this resource group's location.
- `admin_user_enabled`: Whether to enable the Admin user. Currently this must be TRUE to allow Docker to access the registry.
- `sku`: The SKU.
- `...`: Other named arguments to pass to the az_resource initialization function.

Details

An ACR resource is a Docker registry hosted in Azure. See the documentation for the resource for more information. To work with the registry (transfer images, retag images, etc) see the documentation for the registry endpoint.

Value

An object of class `az_container_registry` representing the registry resource.

See Also

get_acr, delete_acr, list_acrs

az_container_registry

docker_registry for the registry endpoint

ACR documentation and API reference

Docker registry API

Examples

```
## Not run:

rg <- AzureRMR::az_rm$
    new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")$
    get_subscription("subscription_id")$
    get_resource_group("rgname")

rg$create_acr("myregistry")


## End(Not run)
```

`create_aks` *Create Azure Kubernetes Service (AKS)*

Description

Method for the AzureRMR::az_resource_group class.

Usage

```
create_aks(name, location = self$location,
           dns_prefix = name, kubernetes_version = NULL,
           enable_rbac = FALSE, agent_pools = list(),
           login_user = "", login_passkey = "",
           cluster_service_principal = NULL,
           properties = list(), ..., wait = TRUE)
```

Arguments

- `name`: The name of the Kubernetes service.
- `location`: The location/region in which to create the service. Defaults to this resource group's location.
- `dns_prefix`: The domain name prefix to use for the cluster endpoint. The actual domain name will start with this argument, followed by a string of pseudorandom characters.
- `kubernetes_version`: The Kubernetes version to use. If not specified, uses the most recent version of Kubernetes available.
- `enable_rbac`: Whether to enable role-based access controls.
- `agent_pools`: A list of pool specifications. See 'Details'.
- `login_user,login_passkey`: Optionally, a login username and public key (on Linux). Specify these if you want to be able to ssh into the cluster nodes.
- `cluster_service_principal`: The service principal (client) that AKS will use to manage the cluster resources. This should be a list, with the first component being the client ID and the second the client secret. If not supplied, the values are obtained from the service principal used for this ARM login.

- `properties`: A named list of further Kubernetes-specific properties to pass to the initialization function.
- `wait`: Whether to wait until the AKS resource provisioning is complete. Note that provisioning a Kubernetes cluster can take several minutes.
- `...`: Other named arguments to pass to the initialization function.

Details

An AKS resource is a Kubernetes cluster hosted in Azure. See the documentation for the resource for more information. To work with the cluster (deploy images, define and start services, etc) see the documentation for the cluster endpoint.

To specify the agent pools for the cluster, it is easiest to use the aks_pools function. This takes as arguments the name(s) of the pools, the number of nodes, the VM size(s) to use, and the operating system (Windows or Linux) to run on the VMs.

Value

An object of class `az_kubernetes_service` representing the service.

See Also

get_aks, delete_aks, list_aks, aks_pools

az_kubernetes_service

kubernetes_cluster for the cluster endpoint

AKS documentation and API reference

Kubernetes reference

Examples

```
## Not run:

rg <- AzureRMR::az_rm$
    new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")$
    get_subscription("subscription_id")$
    get_resource_group("rgname")

rg$create_aks("mycluster", agent_pools=aks_pools("pool1", 5))

# GPU-enabled cluster
rg$create_aks("mygpucluster", agent_pools=aks_pools("pool1", 5, size="Standard_NC6s_v3"))


## End(Not run)
```

`delete_aci` *Delete an Azure Container Instance (ACI)*

Description

Method for the AzureRMR::az_resource_group class.

Usage

```
delete_aci(name, confirm=TRUE, wait=FALSE)
```

Arguments

- `name`: The name of the container instance.
- `confirm`: Whether to ask for confirmation before deleting.
- `wait`: Whether to wait until the deletion is complete.

Value

NULL on successful deletion.

See Also

create_aci, get_aci

az_container_instance

ACI documentation and API reference

Docker commandline reference

Examples

```
## Not run:

rg <- AzureRMR::az_rm$
    new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")$
    get_subscription("subscription_id")$
    get_resource_group("rgname")

rg$delete_aci("mycontainer")


## End(Not run)
```

delete_acr *Delete an Azure Container Registry (ACR)*

Description

Method for the AzureRMR::az_resource_group class.

Usage

```
delete_acr(name, confirm=TRUE, wait=FALSE)
```

Arguments

- `name`: The name of the container registry.
- `confirm`: Whether to ask for confirmation before deleting.
- `wait`: Whether to wait until the deletion is complete.

Value

NULL on successful deletion.

See Also

create_acr, get_acr

az_container_registry

docker_registry for the registry endpoint

ACR documentation and API reference

Docker registry API

Examples

```
## Not run:

rg <- AzureRMR::az_rm$
    new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")$
    get_subscription("subscription_id")$
    get_resource_group("rgname")

rg$delete_acr("myregistry")


## End(Not run)
```

`delete_aks` *Delete an Azure Kubernetes Service (AKS)*

Description

Method for the AzureRMR::az_resource_group class.

Usage

```
delete_aks(name, confirm=TRUE, wait=FALSE)
```

Arguments

- `name`: The name of the Kubernetes service.
- `confirm`: Whether to ask for confirmation before deleting.
- `wait`: Whether to wait until the deletion is complete.

Value

NULL on successful deletion.

See Also

create_aks, get_aks

az_kubernetes_service

kubernetes_cluster for the cluster endpoint

AKS documentation and API reference

Kubernetes reference

Examples

```
## Not run:

rg <- AzureRMR::az_rm$
    new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")$
    get_subscription("subscription_id")$
    get_resource_group("rgname")

rg$delete_aks("mycluster")


## End(Not run)
```

`docker_registry` *Docker registry class*

Description

Class representing a Docker registry. Note that this class can be used to interface with any Docker registry that supports the HTTP V2 API, not just those created via the Azure Container Registry service.

Usage

```
docker_registry
```

Format

An object of class `R6ClassGenerator` of length 24.

Methods

The following methods are available, in addition to those provided by the AzureRMR::az_resource class:

- `new(...)`: Initialize a new registry object. See 'Details'.
- `login`: Login to the registry via `docker login`.
- `push(src_image, dest_image)`: Push an image to the registry, using `docker tag` and `docker push`.
- `pull(image)`: Pulls an image from the registry, using `docker pull`.
- `delete_layer(layer, digest, confirm=TRUE)`: Deletes a layer from the registry.
- `delete_image(image, digest, confirm=TRUE)`: Deletes an image from the registry.
- `list_repositories`: Lists the repositories (images) in the registry.

Details

The arguments to the `new()` method are:

- `server`: The name of the registry server.
- `username`: The username that Docker will use to authenticate with the registry.
- `password`: The password that Docker will use to authenticate with the registry.
- `login`: Whether to login to the registry immediately; defaults to TRUE.

Currently this class does not support authentication methods other than a username/password combination.

The `login()`, `push()` and `pull()` methods for this class call the `docker` commandline tool under the hood. This allows all the features supported by Docker to be available immediately, with a minimum of effort. Any calls to the `docker` tool will also contain the full commandline as the `cmdline` attribute of the (invisible) returned value; this allows scripts to be developed that can be run outside R.

See Also

acr, call_docker

Docker commandline reference

Docker registry API

Examples

```
## Not run:

# recommended way of retrieving a registry: via a resource group object
rg <- AzureRMR::az_rm$
    new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")$
    get_subscription("subscription_id")$
    get_resource_group("rgname")

# get the registry endpoint
dockerreg <- rg$get_acr("myregistry")$get_docker_registry()

dockerreg$login()
dockerreg$list_repositories()

# create an image from a Dockerfile in the current directory
call_docker("build -t myimage .")

# push the image
dockerreg$push("myimage")


## End(Not run)
```

---

get_aci *Get Azure Container Instance (ACI)*

Description

Method for the AzureRMR::az_resource_group class.

Usage

```
get_aci(name)
list_acis()
```

Arguments

- name: For get_aci(), the name of the container instance resource.

Details

The AzureRMR::az_resource_group class has both get_aci() and list_acis() methods, while the AzureRMR::az_subscription class only has the latter.

Value

For get_aci(), an object of class az_container_instance representing the instance resource.

For list_acis(), a list of such objects.

See Also

create_aci, delete_aci

az_container_instance

ACI documentation and API reference

Docker commandline reference

Examples

```
## Not run:

rg <- AzureRMR::az_rm$
    new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")$
    get_subscription("subscription_id")$
    get_resource_group("rgname")

rg$get_aci("mycontainer")


## End(Not run)
```

---

get_acr *Get Azure Container Registry (ACR)*

Description

Method for the AzureRMR::az_resource_group class.

Usage

```
get_acr(name)
list_acrs()
```

Arguments

- name: For get_acr(), the name of the container registry resource.

Details

The `AzureRMR::az_resource_group` class has both `get_acr()` and `list_acrs()` methods, while the `AzureRMR::az_subscription` class only has the latter.

Value

For `get_acr()`, an object of class `az_container_registry` representing the registry resource.

For `list_acrs()`, a list of such objects.

See Also

create_acr, delete_acr

az_container_registry

docker_registry for the registry endpoint

ACR documentation and API reference

Docker registry API

Examples

```
## Not run:

rg <- AzureRMR::az_rm$
    new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")$
    get_subscription("subscription_id")$
    get_resource_group("rgname")

rg$get_acr("myregistry")


## End(Not run)
```

`get_aks` *Get Azure Kubernetes Service (AKS)*

Description

Method for the AzureRMR::az_resource_group class.

Usage

```
get_aks(name)
list_aks()
```

Arguments

- `name`: For `get_aks()`, the name of the Kubernetes service.

Details

The AzureRMR::az_resource_group class has both get_aks() and list_aks() methods, while the AzureRMR::az_subscription class only has the latter.

Value

For get_aks(), an object of class az_kubernetes_service representing the service.

For list_aks(), a list of such objects.

See Also

create_aks, delete_aks

az_kubernetes_service

kubernetes_cluster for the cluster endpoint

AKS documentation and API reference

Kubernetes reference

Examples

```
## Not run:

rg <- AzureRMR::az_rm$
    new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")$
    get_subscription("subscription_id")$
    get_resource_group("rgname")

rg$get_aks("mycluster")


## End(Not run)
```

---

is_acr *Utility functions to test whether an object is of the given class.*

Description

Utility functions to test whether an object is of the given class.

Usage

```
is_acr(object)

is_aks(object)

is_aci(object)
```

```
is_docker_registry(object)

is_kubernetes_cluster(object)
```

Arguments

object	An R object

Details

These functions are simple wrappers around `R6::is.R6` and `inherits`.

Value

TRUE or FALSE depending on whether the object is an R6 object of the specified class.

---

kubernetes_cluster *Kubernetes cluster class*

Description

Class representing a Kubernetes cluster. Note that this class can be used to interface with any Docker registry that supports the HTTP V2 API, not just those created via the Azure Container Registry service.

Usage

```
kubernetes_cluster
```

Format

An object of class `R6ClassGenerator` of length 24.

Methods

The following methods are available, in addition to those provided by the AzureRMR::az_resource class:

- `new(...)`: Initialize a new registry object. See 'Initialization' below.
- `create_registry_secret(registry, secret_name, email)`: Provide authentication secret for a Docker registry. See 'Secrets' below.
- `delete_registry_secret(secret_name)`: Delete a registry authentication secret.
- `create(file)`: Creates a deployment or service from a file, using `kubectl create -f`.
- `get(type)`: Get information about resources, using `kubectl get`.
- `run(name, image)`: Run an image using `kubectl run --image`.
- `expose(name, type, file)`: Expose a service using `kubectl expose`. If the `file` argument is provided, read service information from there.

- `delete(type, name, file)`: Delete a resource (deployment or service) using `kubectl delete`. If the `file` argument is provided, read resource information from there.
- `apply(file)`: Apply a configuration file, using `kubectl apply -f`.
- `show_dashboard(port)`: Display the cluster dashboard. By default, use local port 30000.
- `kubectl(cmd)`: Run an arbitrary `kubectl` command on this cluster. Called by the other methods above.
- `helm(cmd)`: Run a `helm` command on this cluster.

Initialization

The `new()` method takes one argument: `config`, the name of the file containing the configuration details for the cluster. This should be a yaml or json file in the standard Kubernetes configuration format. Set this to NULL to use the default `~/.kube/config` file.

Secrets

To allow a cluster to authenticate with a Docker registry, call the `create_registry_secret` method with the following arguments:

- `registry`: An object of class either acr representing an Azure Container Registry service, or docker_registry representing the registry itself.
- `secret_name`: The name to give the secret. Defaults to the name of the registry server.
- `email`: The email address for the Docker registry.

Kubectl

The methods for this class call the `kubectl` commandline tool, passing it the `--config` option to specify the configuration information for the cluster. This allows all the features supported by Kubernetes to be available immediately and with a minimum of effort, although it does require that `kubectl` be installed. Any calls to `kubectl` will also contain the full commandline as the `cmdline` attribute of the (invisible) returned value; this allows scripts to be developed that can be run outside R.

See Also

aks, call_kubectl

Kubectl commandline reference

Examples

```
## Not run:

# recommended way of retrieving a cluster: via a resource group object
rg <- AzureRMR::az_rm$
    new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")$
    get_subscription("subscription_id")$
    get_resource_group("rgname")
```

```
# get the cluster endpoint
kubclus <- rg$get_aks("mycluster")$get_cluster()

# get registry authentication secret
kubclus$create_registry_secret(rg$get_acr("myregistry"))

# deploy a service
kubclus$create("deployment.yaml")

# can also supply the deployment parameters inline
kubclus$create("
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: model1
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: model1
    spec:
      containers:
      - name: model1
        image: myregistry.azurecr.io/model1
        ports:
        - containerPort: 8000
      imagePullSecrets:
      - name: myregistry.azurecr.io
---
apiVersion: v1
kind: Service
metadata:
  name: model1-svc
spec:
  selector:
    app: model1
  type: LoadBalancer
  ports:
  - protocol: TCP
    port: 8000")

# track status
kubclus$get("deployment")
kubclus$get("service")


## End(Not run)
```

---

list_kubernetes_versions

List available Kubernetes versions

Description

Method for the AzureRMR::az_subscription and AzureRMR::az_resource_group classes.

Usage

```
## R6 method for class 'az_subscription'
list_kubernetes_versions(location)

## R6 method for class 'az_resource_group'
list_kubernetes_versions()
```

Arguments

- `location`: For the az_subscription class method, the location for which to obtain available Kubernetes versions.

Value

A vector of strings, which are the Kubernetes versions that can be used when creating a cluster.

See Also

create_aks

Kubernetes reference

Examples

```
## Not run:

rg <- AzureRMR::az_rm$
    new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")$
    get_subscription("subscription_id")$
    get_resource_group("rgname")

rg$list_kubernetes_versions()


## End(Not run)
```

`list_vm_sizes` *List available VM sizes*

Description

Method for the AzureRMR::az_subscription and AzureRMR::az_resource_group classes.

Usage

```
## R6 method for class 'az_subscription'
list_vm_sizes(location, name_only = FALSE)

## R6 method for class 'az_resource_group'
list_vm_sizes(name_only = FALSE)
```

Arguments

- `location`: For the subscription class method, the location/region for which to obtain available VM sizes.
- `name_only`: Whether to return only a vector of names, or all information on each VM size.

Value

If `name_only` is TRUE, a character vector of names. If FALSE, a data frame containing the following information for each VM size: the name, number of cores, OS disk size, resource disk size, memory, and maximum data disks.

Examples

```
## Not run:

sub <- AzureRMR::az_rm$
    new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")$
    get_subscription("subscription_id")

sub$list_vm_sizes("australiaeast")

# same output as above
rg <- sub$create_resource_group("rgname", location="australiaeast")
rg$list_vm_sizes()


## End(Not run)
```

Index