Package 'AzureVM'

December 25, 2018

Title Virtual Machines in 'Azure'

Version 1.0.0

Description Functionality for working with virtual machines (VMs) in Microsoft's 'Azure' cloud: <https://azure.microsoft.com/en-us/services/virtual-machines/>. Includes facilities to create, startup, shutdown, and cleanly delete VMs and VM clusters. With a running VM, execute scripts and install optional extensions. A selection of VM templates based on the 'Data Science Virtual Machine' (DSVM) is supplied; this allows fast and easy provisioning of a VM preinstalled with several software packages useful for data science. Alternatively, users can provide VM templates of their own.

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

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

License MIT + file LICENSE

VignetteBuilder knitr

Depends R (>= 3.3),

Imports R6, AzureRMR

Suggests knitr, testthat

RoxygenNote 6.1.0.9000

NeedsCompilation no

Author Hong Ooi [aut, cre],
Data Science Virtual Machine team [ctb] (DSVM template provider),
Microsoft [cph]

Maintainer Hong Ooi <hongooi@microsoft.com>

Repository CRAN

Date/Publication 2018-12-25 22:30:03 UTC

R topics documented:

`az_vm_resource` *Virtual machine resource class*

Description

Class representing a virtual machine resource. In general, the methods in this class should not be called directly, nor should objects be directly instantiated from it. Use the `az_vm_template` class for interacting with VMs instead.

Usage

```
az_vm_resource
```

Format

An R6 object of class `az_vm_resource`, inheriting from `AzureRMR::az_resource`.

Methods

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

- `new(...)`: Initialize a new VM object.
- `start(wait=TRUE)`: Start the VM. By default, wait until the startup process is complete.
- `stop(deallocate=TRUE, wait=FALSE)`: Stop the VM. By default, deallocate it as well.
- `restart(wait=TRUE)`: Restart the VM.
- `run_deployed_command(command, parameters, script)`: Run a PowerShell command on the VM.
- `run_script(script, parameters)`: Run a script on the VM. For a Linux VM, this will be a shell script; for a Windows VM, a PowerShell script. Pass the script as a character vector.
- `sync_vm_status()`: Update the VM status fields in this object with information from the host.
- `resize(size, deallocate=FALSE, wait=FALSE)`: Resize the VM. Optionally deallocate it first (may sometimes be necessary).

See Also

AzureRMR::az_resource, VM API reference

az_vm_template *Virtual machine cluster template class*

Description

Class representing a virtual machine template. This class keeps track of all resources that are created as part of deploying a VM or cluster of VMs, and exposes methods for managing them. In this page, "VM" refers to both a cluster of virtual machines, as well as a single virtual machine (which is treated as the special case of a cluster containing a single node).

Usage

```
az_vm_template
```

Format

An R6 object of class `az_vm_template`, inheriting from `AzureRMR::az_template`.

Details

A single virtual machine in Azure is actually a collection of resources, including any and all of the following. A cluster can share a storage account and virtual network, but each individual node will still have its own IP address and network interface.

- Storage account
- Network interface
- Network security group
- Virtual network
- IP address
- The VM itself

By wrapping the deployment template used to create these resources, the `az_vm_template` class allows managing them all as a single entity.

Methods

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

- `new(...)`: Initialize a new VM object. See 'Initialization' for more details.
- `start(wait=TRUE)`: Start the VM. By default, wait until the startup process is complete.
- `stop(deallocate=TRUE, wait=FALSE)`: Stop the VM. By default, deallocate it as well.
- `restart(wait=TRUE)`: Restart the VM.
- `run_deployed_command(command, parameters, script)`: Run a PowerShell command on the VM.

- `run_script(script, parameters)`: Run a script on the VM. For a Linux VM, this will be a shell script; for a Windows VM, a PowerShell script. Pass the script as a character vector.
- `sync_vm_status()`: Update the VM status fields in this object with information from the host.
- `resize(size, deallocate=FALSE, wait=FALSE)`: Resize the VM. Optionally deallocate it first (may sometimes be necessary).

Fields

The following fields are available, in addition to those provided by the `AzureRMR::az_template` class. Each is a list with one element per node in the cluster.

- `disks`: The status of any attached disks.
- `ip_address`: The IP address. NULL if the node is currently deallocated.
- `dns_name`: The fully qualified domain name.
- `status`: The status of the node, giving the provisioning state and power state.

Initialization

Initializing a new object of this class can either retrieve an existing VM template, or deploy a new VM template on the host. Generally, the best way to initialize an object is via the VM-related methods of the az_subscription and az_resource_group class, which handle the details automatically.

A new VM can be created in *exclusive* mode, meaning a new resource group is created solely to hold the VM. This simplifies deleting a VM considerably, as deleting the resource group will also automatically delete all the VM's resources. This can be done asynchronously, meaning that the `delete()` method returns immediately while the process continues on the host. Otherwise, deleting a VM will explicitly delete each of its resources, a task that must be done synchronously to allow for dependencies.

See Also

AzureRMR::az_resource, create_vm, create_vm_cluster, get_vm, get_vm_cluster, list_vms, delete_vm, delete_vm_cluster, VM API reference

Examples

```
## Not run:

# recommended way to retrieve a VM: via a resource group or subscription object
sub <- AzureRMR::az_rm$
    new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")$
    get_subscription("subscription_id")

vm <- sub$get_vm("myLinuxDSVM")

# start the VM
vm$start()

# run a shell command
```

```
vm$run_script("ifconfig > /tmp/ifc.out")

# stop (and deallocate) the VM
vm$stop()

# resize the VM
vm$resize("Standard_DS13_v2")

# get the VM status
vm$sync_vm_status()


## End(Not run)
```

`create_vm` *Create a new virtual machine or cluster of virtual machines*

Description

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

Usage

```
## R6 method for class 'az_resource_group'
create_vm(name, os = c("Windows", "Ubuntu"), size = "Standard_DS3_v2",
          username, passkey, userauth_type = c("password", "key"),
          ext_file_uris = NULL, inst_command = NULL,
          template, parameters, ..., wait = TRUE)

## R6 method for class 'az_subscription'
create_vm(name, location, os = c("Windows", "Ubuntu"), size = "Standard_DS3_v2",
          username, passkey, userauth_type = c("password", "key"),
          ext_file_uris = NULL, inst_command = NULL,
          template, parameters, ..., wait = TRUE)

## R6 method for class 'az_resource_group'
create_vm_cluster(name, os = c("Windows", "Ubuntu"), size = "Standard_DS3_v2",
                  username, passkey, userauth_type = c("password", "key"),
                  ext_file_uris = NULL, inst_command = NULL, clust_size,
                  template, parameters, ..., wait = TRUE)

## R6 method for class 'az_subscription'
create_vm_cluster(name, location, os = c("Windows", "Ubuntu"), size = "Standard_DS3_v2",
                  username, passkey, userauth_type = c("password", "key"),
                  ext_file_uris = NULL, inst_command = NULL, clust_size,
                  template, parameters, ..., wait = TRUE)
```

Arguments

- `name`: The name of the VM or cluster.
- `location`: For the subscription class methods, the location for the VM. Use the `list_locations()` method of the `AzureRMR::az_subscription` class to see what locations are available.
- `os`: The operating system for the VM.
- `size`: The VM size. Use the `list_vm_sizes()` method of the `AzureRMR::az_subscription` class to see what sizes are available.
- `username`: The login username for the VM.
- `passkey`: The login password or public key.
- `userauth_type`: The type of login authentication to use. Only has an effect for Linux-based VMs; Windows VMs will always use `"password"`.
- `ext_file_uris`: Optional link to download extension packages.
- `inst_command`: If `ext_file_uris` is supplied, the install script to run. Defaults to `install.sh` for an Ubuntu VM, or `install.ps1` for a Windows VM.
- `clust_size`: For a cluster, the number of nodes to create.
- `template`: Optional: the VM template to deploy. By default, this is determined by the values of the other arguments; see 'Details' below.
- `parameters`: Optional: other parameters to pass to the deployment.
- `wait`: Whether to wait until the deployment is complete.
- `...`: Other arguments to lower-level methods.

Details

This method deploys a template to create a new virtual machine or cluster of VMs. Currently, seven templates are supplied with this package, based on the Azure Data Science Virtual Machine:

- Ubuntu DSVM
- Ubuntu DSVM using public key authentication
- Ubuntu DSVM with extensions
- Ubuntu DSVM cluster
- Ubuntu DSVM cluster with extensions
- Windows Server 2016 DSVM
- Windows Server 2016 DSVM cluster with extensions

An individual virtual machine is treated as a cluster containing only a single node.

You can also supply your own VM template for deployment, via the `template` argument. See AzureRMR::az_template for information how to supply templates. Note that if you do this, you may also have to supply a `parameters` argument, as the standard parameters for this method are customised for the DSVM.

For the `AzureRMR::az_subscription` method, this will by default create the VM in *exclusive* mode, meaning a new resource group is created solely to hold the VM. This simplifies managing the VM considerably, in particular deleting the resource group will also automatically delete all the VM's resources.

Value

An object of class `az_vm_template` representing the created VM.

See Also

az_vm_template, AzureRMR::az_subscription, AzureRMR::az_resource_group, Data Science Virtual Machine

Examples

```
## Not run:

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

# default Windows Server DSVM: make sure to use a strong password!
sub$create_vm("myWindowsDSVM",
   location="australiaeast",
   username="ds",
   passkey="Password123!")

# upsized Linux (Ubuntu) DSVM
sub$create_vm("myLinuxDSVM",
   location="australiaeast",
   os="Linux",
   username="ds",
   passkey=readLines("~/id_rsa.pub"),
   size="Standard_DS13_v2")

sub$create_vm_cluster("myLinuxCluster",
   location="australiaeast",
   os="Linux",
   username="ds",
   passkey=readLines("~/id_rsa.pub"),
   clust_size=5)


## End(Not run)
```

`delete_vm` *Delete virtual machine*

Description

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

Usage

```
## R6 method for class 'az_resource_group'
delete_vm(name, confirm = TRUE, free_resources = TRUE)

## R6 method for class 'az_subscription'
delete_vm(name, confirm = TRUE, free_resources = TRUE,
          resource_group = name)

## R6 method for class 'az_resource_group'
delete_vm_cluster(name, confirm = TRUE, free_resources = TRUE)

## R6 method for class 'az_subscription'
delete_vm_cluster(name, confirm = TRUE, free_resources = TRUE,
                  resource_group = name)
```

Arguments

- `name`: The name of the VM or cluster.
- `confirm`: Whether to confirm the delete.
- `free_resources`: If this was a deployed template, whether to free all resources created during the deployment process.
- `resource_group`: For the `AzureRMR::az_subscription` method, the resource group containing the VM or cluster.

Details

If the VM or cluster is of class az_vm_template and was created in exclusive mode, this method deletes the entire resource group that it occupies. This automatically frees all resources that were created during the deployment process. Otherwise, if `free_resources=TRUE`, it manually deletes each individual resource in turn. This is done synchronously (the method does not return until the deletion is complete) to allow for dependencies.

If the VM is of class az_vm_resource, this method only deletes the VM resource itself, not any other resources it may depend on.

See Also

create_vm, az_vm_template, az_vm_resource, AzureRMR::az_subscription, AzureRMR::az_resource_group

Examples

```
## Not run:

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

sub$delete_vm("myWindowsDSVM")
sub$delete_vm("myLinuxDSVM")
```

```
## End(Not run)
```

`get_vm` *Get existing virtual machine(s)*

Description

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

Usage

```
## R6 method for class 'az_subscription'
get_vm(name, resource_group = name)

## R6 method for class 'az_resource_group'
get_vm(name)

## R6 method for class 'az_subscription'
get_vm_cluster(name, resource_group = name)

## R6 method for class 'az_resource_group'
get_vm_cluster(name)

## R6 method for class 'az_resource_group'
## R6 method for class 'az_subscription'
list_vms()
```

Arguments

- `name`: The name of the VM or cluster.
- `resource_group`: For the `az_subscription` method, the resource group in which `get_vm()` will look for the VM. Defaults to the VM name.

Details

Despite the names, `get_vm` and `get_vm_cluster` can both be used to retrieve individual VMs and clusters. The main difference is in their behaviour if a deployment template is not found. In the case of `get_vm`, it also searches for a raw VM resource of the given name, whereas `get_vm_cluster` will throw an error immediately.

Value

For `get_vm()`, an object representing the VM, either of class `az_vm_template` or `az_vm_resource`.

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

For `get_vm_cluster()`, an object representing the cluster.

See Also

az_vm_template, az_vm_resource, AzureRMR::az_subscription, AzureRMR::az_resource_group

Examples

```
## Not run:

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

sub$list_vms()
sub$get_vm("myVirtualMachine")

rg <- sub$get_resource_group("rgname")
rg$get_vm("myOtherVirtualMachine")


## End(Not run)
```

---

is_vm_template *Is an object an Azure VM template*

Description

Is an object an Azure VM template

Usage

```
is_vm_template(object)
```

Arguments

object	an R object.

Details

This function returns TRUE only for an object representing a VM template deployment. In particular, it returns FALSE for a raw VM resource.

Value

A boolean.

`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, suitable for passing to `create_vm`. 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.

See Also

create_vm

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