Package 'AzureRMR'

December 2, 2018

Title Interface to 'Azure Resource Manager'

Version 1.0.0

Description
A lightweight but powerful R interface to the 'Azure Resource Manager' REST API. The package exposes classes and methods for 'OAuth' authentication and working with subscriptions and resource groups. It also provides functionality for creating and deleting 'Azure' resources and deploying templates. While 'AzureRMR' can be used to manage any 'Azure' service, it can also be extended by other packages to provide extra functionality for specific services.

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

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

License MIT + file LICENSE

VignetteBuilder knitr

Depends R (>= 3.3)

Imports utils, httr (>= 1.3), jsonlite, R6

Suggests knitr, testthat

RoxygenNote 6.1.0.9000

NeedsCompilation no

Author Hong Ooi [aut, cre],
Microsoft [cph]

Maintainer Hong Ooi <hongooi@microsoft.com>

Repository CRAN

Date/Publication 2018-12-02 17:20:07 UTC

R topics documented:

`AzureToken` *Azure OAuth authentication*

Description

Azure OAuth 2.0 token class, inheriting from the Token2.0 class in httr. Rather than calling the initialization method directly, tokens should be created via `get_azure_token()`.

Usage

```
AzureToken
```

Format

An R6 object of class `AzureToken`.

Methods

- `refresh`: Refreshes the token. For expired Azure tokens using client credentials, refreshing really means requesting a new token.
- `validate`: Checks if the token is still valid. For Azure tokens using client credentials, this just checks if the current time is less than the token's expiry time.

Caching

This class never caches its tokens, unlike httr::Token2.0.

See Also

get_azure_token, httr::Token

az_resource *Azure resource class*

Description

Class representing a generic Azure resource.

Usage

```
az_resource
```

Format

An R6 object of class `az_resource`.

Methods

- `new(...)`: Initialize a new resource object. See 'Initialization' for more details.
- `delete(..., confirm=TRUE, wait=FALSE)`: Delete this resource, after a confirmation check. Optionally wait for the delete to finish.
- `update(...)`: Update this resource on the host.
- `sync_fields()`: Update the fields in this object with information from the host. Returns the `properties$provisioningState` field, so you can query this programmatically to check if a resource has finished provisioning. Not all resource types require explicit provisioning, in which case this method will return NULL.
- `set_api_version(api_version)`: Set the API version to use when interacting with the host. By default, use the latest API version available.
- `do_operation(...)` Carry out an operation. See 'Operations' for more details.

Initialization

There are multiple ways to initialize a new resource object. The `new()` method can retrieve an existing resource, deploy/create a new resource, or create an empty/null object (without communicating with the host), based on the arguments you supply.

All of these initialization options have the following arguments in common.

1. `token`: An OAuth 2.0 token, as generated by get_azure_token.
2. `subscription`: The subscription ID.
3. `api_version`: Optionally, the API version to use when interacting with the host. By default, this is NULL in which case the latest API version will be used.
4. A set of *identifying arguments*:
   - `resource_group`: The resource group containing the resource.
   - `id`: The full ID of the resource. This is a string of the form `/subscriptions/{uuid}/resourceGroups/{resourc`
   - `provider`: The provider of the resource, eg `Microsoft.Compute`.

- path: The path to the resource, eg virtualMachines.
- type: The combination of provider and path, eg Microsoft.Compute/virtualMachines.
- name: The name of the resource instance, eg myWindowsVM.

Providing id will fill in the values for all the other identifying arguments. Similarly, providing type will fill in the values for provider and path. Unless you provide id, you must also provide name.

The default behaviour for new() is to retrieve an existing resource, which occurs if you supply only the arguments listed above. If you also supply an argument deployed_properties=NULL, this will create a null object. If you supply any other (named) arguments, new() will create a new object on the host, with the supplied arguments as parameters.

Generally, the easiest way to initialize an object is via the get_resource, create_resource or list_resources methods of the az_resource_group class, which will handle all the gory details automatically.

Operations

The do_operation() method allows you to carry out arbitrary operations on the resource. It takes the following arguments:

- op: The operation in question, which will be appended to the URL path of the request.
- options: A named list giving the URL query parameters.
- ...: Other named arguments passed to call_azure_rm, and then to the appropriate call in httr. In particular, use body to supply the body of a PUT, POST or PATCH request.
- http_verb: The HTTP verb as a string, one of GET, PUT, POST, DELETE, HEAD or PATCH.

Consult the Azure documentation for your resource to find out what operations are supported.

See Also

az_resource_group, call_azure_rm, call_azure_url, Resources API reference

Examples

```
## Not run:

# recommended way to retrieve a resource: via a resource group object
# storage account:
stor <- resgroup$get_resource(type="Microsoft.Storage/storageAccounts", name="mystorage")
# virtual machine:
vm <- resgroup$get_resource(type="Microsoft.Compute/virtualMachines", name="myvm")

## carry out operations on a resource

# storage account: get access keys
stor$do_operation("listKeys", http_verb="POST")

# virtual machine: run a script
vm$do_operation("runCommand",
    body=list(
        commandId="RunShellScript", # RunPowerShellScript for Windows
```

```
        script=as.list("ifconfig > /tmp/ifconfig.out")
    ),
    encode="json",
    http_verb="POST")

## retrieve properties

# storage account: endpoint URIs
stor$properties$primaryEndpoints$file
stor$properties$primaryEndpoints$blob

# virtual machine: hardware profile
vm$properties$hardwareProfile

## update a resource: resizing a VM
properties <- list(hardwareProfile=list(vmSize="Standard_DS3_v2"))
vm$do_operation(http_verb="PATCH",
    body=list(properties=properties),
    encode="json")

# sync with Azure: useful to track resource creation/update status
vm$sync_fields()

# delete a resource
stor$delete()


## End(Not run)
```

`az_resource_group` *Azure resource group class*

Description

Class representing an Azure resource group.

Usage

```
az_resource_group
```

Format

An R6 object of class `az_resource_group`.

Methods

- `new(token, subscription, id, ...)`: Initialize a resource group object. See 'Initialization' for more details.

- `delete(confirm=TRUE)`: Delete this resource group, after a confirmation check. This is asynchronous: while the method returns immediately, the delete operation continues on the host in the background. For resource groups containing a large number of deployed resources, this may take some time to complete.
- `list_templates()`: List deployed templates in this resource group.
- `get_template(name)`: Return an object representing an existing template.
- `deploy_template(...)`: Deploy a new template. See 'Templates' for more details.
- `delete_template(name, confirm=TRUE, free_resources=FALSE)`: Delete a deployed template, and optionally free any resources that were created.
- `get_resource(...)`: Return an object representing an existing resource. See 'Resources' for more details.
- `create_resource(...)`: Create a new resource.
- `delete_resource(..., confirm=TRUE, wait=FALSE)`: Delete an existing resource. Optionally wait for the delete to finish.
- `resource_exists(...)`: Check if a resource exists.
- `list_resources()`: Return a list of resource group objects for this subscription.

Initialization

Initializing a new object of this class can either retrieve an existing resource group, or create a new resource group on the host. Generally, the easiest way to create a resource group object is via the `get_resource_group`, `create_resource_group` or `list_resource_groups` methods of the az_subscription class, which handle this automatically.

To create a resource group object in isolation, supply (at least) an Oauth 2.0 token of class AzureToken, the subscription ID, and the resource group name. If this object refers to a *new* resource group, supply the location as well (use the `list_locations` method of the `az_subscription class` for possible locations). You can also pass any optional parameters for the resource group as named arguments to `new()`.

Templates

To deploy a new template, pass the following arguments to `deploy_template()`:

- `name`: The name of the deployment.
- `template`: The template to deploy. This can be provided in a number of ways:
  1. A nested list of name-value pairs representing the parsed JSON
  2. The name of a template file
  3. A vector of strings containing unparsed JSON
  4. A URL from which the template can be downloaded
- `parameters`: The parameters for the template. This can be provided using any of the same methods as the `template` argument.

Retrieving or deleting a deployed template requires only the name of the deployment.

Resources

There are a number of arguments to `get_resource()`, `create_resource()` and `delete_resource()` that serve to identify the specific resource in question:

- `id`: The full ID of the resource, including subscription ID and resource group.
- `provider`: The provider of the resource, eg `Microsoft.Compute`.
- `path`: The full path to the resource, eg `virtualMachines`.
- `type`: The combination of provider and path, eg `Microsoft.Compute/virtualMachines`.
- `name`: The name of the resource instance, eg `myWindowsVM`.

Providing the `id` argument will fill in the values for all the other arguments. Similarly, providing the `type` argument will fill in the values for `provider` and `path`. Unless you provide `id`, you must also provide `name`.

To create/deploy a new resource, specify any extra parameters that the provider needs as named arguments to `create_resource()`.

See Also

az_subscription, az_template, az_resource, Azure resource group overview, Resources API reference, Template API reference

Examples

```
## Not run:

# recommended way to retrieve a resource group object
rg <- az_rm$
    new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")$
    get_subscription("subscription_id")$
    get_resource_group("rgname")

# list resources & templates in this resource group
rg$list_resources()
rg$list_templates()

# get a resource (virtual machine)
rg$get_resource(type="Microsoft.Compute/virtualMachines", name="myvm")

# create a resource (storage account)
rg$create_resource(type="Microsoft.Storage/storageAccounts", name="mystorage",
    kind="StorageV2",
    sku=list(name="Standard_LRS"))

# delete a resource
rg$delete_resource(type="Microsoft.Storage/storageAccounts", name="mystorage")

# deploy a template
rg$deploy_template("tplname",
    template="template.json",
    parameters="parameters.json")
```

```
# deploy a template with parameters inline
rg$deploy_template("mydeployment",
    template="template.json",
    parameters=list(parm1="foo", parm2="bar"))

# delete a template and free resources
rg$delete_template("tplname", free_resources=TRUE)

# delete the resource group itself
rg$delete()


## End(Not run)
```

---

az_rm *Azure Resource Manager*

Description

Base class for interacting with Azure Resource Manager.

Usage

```
az_rm
```

Format

An R6 object of class `az_rm`.

Methods

- `new(tenant, app, ...)`: Initialize a new ARM connection with the given credentials. See 'Authentication' for more details.
- `list_subscriptions()`: Returns a list of objects, one for each subscription associated with this app ID.
- `get_subscription(id)`: Returns an object representing a subscription.

Authentication

To authenticate with ARM, provide the following arguments to the `new` method:

- `tenant`: Your tenant ID.
- `app`: Your client/app ID which you registered in Azure Active Directory.
- `auth_type`: Either `"client_credentials"` (the default) or `"device_code"`.
- `password`: if `auth_type == "client_credentials"`, your password.
- `host`: your ARM host. Defaults to `https://management.azure.com/`. Change this if you are using a government or private cloud.

- aad_host: Azure Active Directory host for authentication. Defaults to https://login.microsoftonline.com/. Change this if you are using a government or private cloud.
- config_file: Optionally, a JSON file containing any of the arguments listed above. Arguments supplied in this file take priority over those supplied on the command line. You can also use the output from the Azure CLI az ad sp create-for-rbac command.
- token: Optionally, an OAuth 2.0 token, of class AzureToken. This allows you to reuse the authentication details for an existing session. If supplied, all other arguments will be ignored.

See Also

get_azure_token, AzureToken, Azure Resource Manager overview, REST API reference

Examples

```
## Not run:

# start a new Resource Manager session
az <- az_rm$new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")

# authenticate with credentials in a file
az <- az_rm$new(config_file="creds.json")

# authenticate with device code
az <- az_rm$new(tenant="myaadtenant.onmicrosoft.com", app="app_id", auth_type="device_code")

# retrieve a list of subscription objects
az$list_subscriptions()

# a specific subscription
az$get_subscription("subscription_id")


## End(Not run)
```

---

az_subscription *Azure subscription class*

Description

Class representing an Azure subscription.

Usage

```
az_subscription
```

Format

An R6 object of class az_subscription.

Methods

- `new(token, id, ...)`: Initialize a subscription object.
- `list_resource_groups()`: Return a list of resource group objects for this subscription.
- `get_resource_group(name)`: Return an object representing an existing resource group.
- `create_resource_group(name, location)`: Create a new resource group in the specified region/location, and return an object representing it.
- `delete_resource_group(name, confirm=TRUE)`: Delete a resource group, after asking for confirmation.
- `resource_group_exists(name)`: Check if a resource group exists.
- `list_resources()`: List all resources deployed under this subscription.
- `list_locations()`: List locations available.
- `get_provider_api_version(provider, type)`: Get the current API version for the given resource provider and type. If no resource type is supplied, returns a vector of API versions, one for each resource type for the given provider. If neither provider nor type is supplied, returns the API versions for all resources and providers.

Details

Generally, the easiest way to create a subscription object is via the `get_subscription` or `list_subscriptions` methods of the az_rm class. To create a subscription object in isolation, call the `new()` method and supply an Oauth 2.0 token of class AzureToken, along with the ID of the subscription.

See Also

Azure Resource Manager overview

Examples

```
## Not run:

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

# retrieve list of resource group objects under this subscription
sub$list_resource_groups()

# get a resource group
sub$get_resource_group("rgname")

# check if a resource group exists, and if not, create it
rg_exists <- sub$resource_group_exists("rgname")
if(!rg_exists)
    sub$create_resource_group("rgname", location="australiaeast")

# delete a resource group
sub$delete_resource_group("rgname")
```

```
# get provider API versions for some resource types
sub$get_provider_api_version("Microsoft.Compute", "virtualMachines")
sub$get_provider_api_version("Microsoft.Storage", "storageAccounts")


## End(Not run)
```

`az_template` *Azure template class*

Description

Class representing an Azure deployment template.

Usage

```
az_template
```

Format

An R6 object of class `az_template`.

Methods

- `new(token, subscription, resource_group, name, ...)`: Initialize a new template object. See 'Initialization' for more details.
- `check()`: Check the deployment status of the template; throw an error if the template has been deleted.
- `cancel(free_resources=FALSE)`: Cancel an in-progress deployment. Optionally free any resources that have already been created.
- `delete(confirm=TRUE, free_resources=FALSE)`: Delete a deployed template, after a confirmation check. Optionally free any resources that were created. If the template was deployed in Complete mode (its resource group is exclusive to its use), the latter process will delete the entire resource group. Otherwise resources are deleted in the order given by the template's output resources list; in this case, some may be left behind if the ordering is incompatible with dependencies.
- `list_resources()`: Returns a list of Azure resource objects that were created by the template. This returns top-level resources only, not those that represent functionality provided by another resource.

Initialization

Initializing a new object of this class can either retrieve an existing template, or deploy a new template on the host. Generally, the easiest way to create a template object is via the `get_template`, `deploy_template` or `list_templates` methods of the az_resource_group class, which handle the details automatically.

To initialize an object that refers to an existing deployment, supply the following arguments to `new()`:

- `token`: An OAuth 2.0 token, as generated by get_azure_token.
- `subscription`: The subscription ID.
- `resource_group`: The resource group.
- `name`: The deployment name‘.

If you also supply the following arguments to `new()`, a new template will be deployed:

- `template`: The template to deploy. This can be provided in a number of ways:
  1. A nested list of name-value pairs representing the parsed JSON
  2. The name of a template file
  3. A vector of strings containing unparsed JSON
  4. A URL from which the host can download the template
- `parameters`: The parameters for the template. This can be provided using any of the same methods as the `template` argument.
- `wait`: Optionally, whether to wait until the deployment is complete. Defaults to FALSE, in which case the method will return immediately.

See Also

az_resource_group, az_resource, Template overview, Template API reference

Examples

```
## Not run:

# recommended way to deploy a template: via a resource group object

tpl <- resgroup$deploy_template("mydeployment",
    template="template.json",
    parameters="parameters.json")

# retrieve list of created resource objects
tpl$list_resources()

# delete template (will not touch resources)
tpl$delete()

# delete template and free resources
tpl$delete(free_resources=TRUE)


## End(Not run)
```

call_azure_rm *Call the Azure Resource Manager REST API*

Description

Call the Azure Resource Manager REST API

Usage

```
call_azure_rm(token, subscription, operation, ..., options = list(),
  api_version = getOption("azure_api_version"))

call_azure_url(token, url, ..., http_verb = c("GET", "DELETE", "PUT",
  "POST", "HEAD", "PATCH"), http_status_handler = c("stop", "warn",
  "message", "pass"), auto_refresh = TRUE)
```

Arguments

token	An Azure OAuth token, of class AzureToken.
subscription	A subscription ID.
operation	The operation to perform, which will form part of the URL path.
...	Other arguments passed to lower-level code, ultimately to the appropriate functions in httr.
options	A named list giving the URL query parameters.
api_version	The API version to use, which will form part of the URL sent to the host.
url	A complete URL to send to the host.
http_verb	The HTTP verb as a string, one of GET, PUT, POST, DELETE, HEAD or PATCH.
http_status_handler	How to handle in R the HTTP status code of a response. "stop", "warn" or "message" will call the appropriate handlers in httr, while "pass" ignores the status code.
auto_refresh	Whether to refresh/renew the OAuth token if it is no longer valid.

Details

These functions form the low-level interface between AzureRMR and Resource Manager. call_azure_rm builds a URL from its arguments and passes it to call_azure_url. Authentication is handled automatically.

Value

If http_status_handler is one of "stop", "warn" or "message", the status code of the response is checked. If an error is not thrown, the parsed content of the response is returned with the status code attached as the "status" attribute.

If http_status_handler is "pass", the entire response is returned without modification.

See Also

httr::GET, httr::PUT, httr::POST, httr::DELETE, httr::stop_for_status, httr::content

---

`format_auth_header` *Format an Azure object*

Description

Miscellaneous functions for printing Azure R6 objects

Usage

```
format_auth_header(token)

format_public_fields(env, exclude = character(0))

format_public_methods(env)
```

Arguments

`token`	An Azure OAuth token.
`env`	An R6 object's environment for printing.
`exclude`	Objects in `env` to exclude from the printout.

Details

These functions are utilities to aid in printing Azure R6 objects. They are not meant to be called by the user.

`get_azure_token` *Generate an Azure OAuth token*

Description

This extends the OAuth functionality in httr to allow for device code authentication.

Usage

```
get_azure_token(aad_host, tenant, app,
  auth_type = c("client_credentials", "device_code"), password,
  resource_host)
```

Arguments

aad_host	URL for your Azure Active Directory host. For the public Azure cloud, this is https://login.microsoftonline.com/.
tenant	Your tenant ID.
app	Your client/app ID which you registered in AAD.
auth_type	The authentication type, either "client_credentials" or "device_code".
password	Your password. Required for auth_type == "client_credentials", ignored for auth_type == "device_code".
resource_host	URL for your resource host. For Resource Manager in the public Azure cloud, this is https://management.azure.com/.

Details

This function does much the same thing as httr::oauth2.0_token(), but with support for device authentication and with unnecessary options removed. Device authentication removes the need to save a password on your machine. Instead, the server provides you with a code, along with a URL. You then visit the URL in your browser and enter the code, which completes the authentication process.

See Also

AzureToken, httr::oauth2.0_token, httr::Token, OAuth authentication for Azure Active Directory, Device code flow on OAuth.com

Examples

```
## Not run:

token <- get_azure_token(
   aad_host="https://login.microsoftonline.com/",
   tenant="myaadtenant.onmicrosoft.com",
   app="app_id",
   password="password",
   resource_host="https://management.azure.com/")


## End(Not run)
```

---

is_subscription *Informational functions*

Description

These functions return whether the object is of the corresponding AzureRMR class.

Usage

```
is_subscription(object)

is_resource_group(object)

is_resource(object)

is_template(object)
```

Arguments

object	An R object.

Value

A boolean.

---

named_list *Miscellaneous utility functions*

Description

Miscellaneous utility functions

Usage

```
named_list(lst, name_fields = "name")

is_url(x, https_only = FALSE)

is_empty(x)
```

Arguments

lst	A named list of objects.
name_fields	The components of the objects in lst, to be used as names.
x	For is_url and is_empty, An R object.
https_only	For is_url, whether to allow only HTTPS URLs.

Details

named_list extracts from each object in lst, the components named by name_fields. It then constructs names for lst from these components, separated by a "/".

Value

For `named_list`, the list that was passed in but with names. For `is_url`, whether the object appears to be a URL (is character of length 1, and starts with the string `"http"`). Optionally, restricts the check to HTTPS URLs only. For `is_empty`, whether the length of the object is zero (this includes the special case of `NULL`).

Index