Package ‘analogsea’

January 5, 2018

Title Interface to ‘Digital Ocean’

Description Provides a set of functions for interacting with the ‘Digital Ocean’ API at <https://developers.digitalocean.com/documentation/v2>, including creating images, destroying them, rebooting, getting details on regions, and available images.

Version 0.6.0

License MIT + file LICENSE

URL https://github.com/sckott/analogsea

BugReports https://github.com/sckott/analogsea/issues

LazyData yes

VignetteBuilder knitr

Imports stats, utils, httr (>= 1.2.0), jsonlite (>= 1.1), magrittr, yaml, aws.s3 (>= 0.3.3)

Suggests roxygen2 (>= 6.0.1), testthat, knitr

RoxygenNote 6.0.1

NeedsCompilation no

Author Scott Chamberlain [aut, cre], Hadley Wickham [aut], Winston Chang [aut], Bob Rudis [ctb], RStudio [cph]

Maintainer Scott Chamberlain <myrmecocystus@gmail.com>

Repository CRAN

Date/Publication 2018-01-04 23:02:45 UTC

R topics documented:

analogsea-package ................................................ 3
account ................................................................ 4
action .................................................................... 4
topics documented:

actions ................................................................. 5
adjectives ................................................................. 5
analogsea-defunct ..................................................... 6
analogsea-deprecated .................................................. 6
as.domain_record ....................................................... 6
as.image ................................................................. 8
as.snapshot ............................................................... 9
as.volume ................................................................. 10
debian ................................................................. 12
docklets_create ......................................................... 13
docklet_create ......................................................... 15
domains ................................................................. 18
domain_create .......................................................... 19
do_oauth ................................................................. 20
do_options ............................................................... 20
droplet ................................................................. 22
droplets ................................................................. 22
droplets_cost ............................................................ 23
droplets_create .......................................................... 24
droplet_action ............................................................ 26
droplet_actions ........................................................... 27
droplet_create ............................................................ 28
droplet_delete ............................................................ 30
droplet_do_actions ...................................................... 31
droplet_execute ........................................................... 31
droplet_freeze ............................................................. 32
droplet_kernels_list ...................................................... 33
droplet_modify ............................................................. 34
droplet_reuse ............................................................. 35
droplet_snapshot .......................................................... 36
droplet_ssh ............................................................... 37
droplet_upgrades_list ..................................................... 38
droplet_wait .............................................................. 39
image_actions ............................................................. 39
image_convert ............................................................. 40
image_delete ............................................................ 40
image_transfer ........................................................... 41
key-crud ................................................................. 41
keys ................................................................. 42
neighbors ............................................................... 43
nouns ................................................................. 43
regions ................................................................. 44
resize ................................................................. 44
sizes ................................................................. 45
standardise_keys ......................................................... 46
tags ................................................................. 46
tag_create .............................................................. 47
tag_delete .............................................................. 48
This package is an R client for Digital Ocean’s RESTful API, and a set of scripts that allow you to install R, RStudio server, RStudio Shiny server, or OpenCPU server, in addition to common packages used. The goal here is to spin up a cloud R environment without leaving R, and requiring no knowledge other than R. Of course if you are more experienced you can log in on the command line and modify anything you want, but for those that just want a quick cloud R environment, this should be one of the easiest options.

You need to authenticate to use this package. Get your auth token at https://cloud.digitalocean.com/settings/api/tokens - See do_oauth for more on authentication.

**ssh keys**

**analogsea** allows you to interact with your droplet(s) from R via SSH. To do this you need to setup SSH keys with Digital Ocean. Make sure you provide Digital Ocean your public key at https://cloud.digitalocean.com/ssh_keys. GitHub has some good advice on creating a new public key if you don’t already have one: https://help.github.com/articles/generating-ssh-keys.

Note that when using ssh, you’ll likely get warnings like ”The authenticity of host can’t be established...”. This is normal, don’t be worried about this.

Note that if you want to connect over SSH to a droplet you have to create the droplet with an SSH key with the `ssh_keys` parameter. If you don’t you can still interact with the droplet via the Digital Ocean API, but you can’t access the droplet over SSH.

**Author(s)**

Scott Chamberlain <myrmecocystus@gmail.com>

Hadley Wickham <hadley@rstudio.com>

Winston Chang <winston@stdout.org>

Bob Rudis <bob@rudis.net>
account

Get account information

Description
Get account information

Usage
account(...)

Arguments

... Options passed down to GET

Examples

## Not run:
account()

## End(Not run)

action

Retrieve an existing action by action id

Description
Retrieve an existing action by action id

Usage
action(actionid, ...)

Arguments

actionid (integer) Optional. An action id.

... Additional arguments passed down to low-level API function (do_*)

Examples

## Not run:
d <- droplet_create()
droplet_actions(d)[[1]]$id %>% action()

## End(Not run)
actions

List actions across all droplets.

Description

"Actions are records of events that have occurred on the resources in your account. These can be things like rebooting a Droplet, or transferring an image to a new region."

Usage

```plaintext
actions(..., page = 1, per_page = 25)

action_wait(x)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Additional arguments passed down to low-level API function (do_*)</td>
</tr>
<tr>
<td>per_page</td>
<td>Number of results per page. Default: 25.</td>
</tr>
<tr>
<td>x</td>
<td>Input object</td>
</tr>
</tbody>
</table>

Details

"An action object is created every time one of these actions is initiated. The action object contains information about the current status of the action, start and complete timestamps, and the associated resource type and ID."

"Every action that creates an action object is available through this endpoint. Completed actions are not removed from this list and are always available for querying."

Examples

```plaintext
## Not run:
actions()

## End(Not run)
```

adjectives

Adjectives to use for seeding random word selection when name not given for a droplet

Description

Adjectives to use for seeding random word selection when name not given for a droplet
Details
A vector of 999 adjectives. From the GitHub repo https://github.com/dariusk/corpora - the data is licensed CC0.

---

**analogsea-defunct**  
*Defunct functions in analogsea*

Description
These functions are gone, no longer available.

Details
- **tag_rename**: DigitalOcean removed this functionality from their API. See https://developers.digitalocean.com/documentation/changelog/api-v2/deprecating-update-tag/ for details.

---

**analogsea-deprecated**  
*Deprecated functions in analogsea*

Description
None at the moment

---

**as.domain_record**  
*List, create, update, and delete domain records.*

Description
List, create, update, and delete domain records.

Usage
```r
as.domain_record(x, domain)
```

## S3 method for class 'list'
```r
as.domain_record(x, domain)
```

## S3 method for class 'domain_record'
```r
as.domain_record(x, domain)
```

## S3 method for class 'domain_record'
```r
as.url(x, ...)
```
as.domain_record

domain_records(domain, ...)

domain_record(domain, domain_record_id, ...)

domain_record_create(domain, type, name = NULL, data = NULL, priority = NULL, port = NULL, ttl = NULL, weight = NULL, flags = NULL, tag = NULL, ...)

domain_record_update(domain_record, type = NULL, name = NULL, data = NULL, priority = NULL, port = NULL, ttl = NULL, weight = NULL, flags = NULL, tag = NULL, ...)

domain_record_delete(domain_record, ...)

Arguments

x Domain record.

domain (domain) Required. Domain Name (e.g. domain.com), specifies the domain for which to create a record.

... Further args passed on the curl call to the web.

domain_record_id (numeric/integer) A domain record ID

type (character) Required. The type of record you would like to create. 'A', 'CNAME', 'NS', 'TXT', 'MX' or 'SRV'

name (character) The host name, alias, or service being defined by the record. Required for 'A', 'CNAME', 'TXT' and 'SRV' records

data (character) Variable data depending on record type. Required for 'A', 'AAAA', 'CNAME', 'MX', 'TXT', 'SRV', and 'NS' records

priority (integer) Required for 'SRV' and 'MX' records

port (integer) Required for 'SRV' records

ttl (numeric/integer) Time to live for the record, in seconds. This defines the time frame that clients can cache queried information before a refresh should be requested. If not set, default is 1800

weight (integer) Required for 'SRV' records

flags (integer) An unsigned integer between 0-255 used for CAA records

tag (character) The parameter tag for CAA records. Valid values are "issue", "wild-issuse", or "iodef"

domain_record A domain record, or anything coercible to one

Examples

## Not run:
# list domains, then get domain records
(d <- domains()[[1]])
(rec <- domain_records(d))

# create a domain
dom <- domain_create('tablesandchairsbunnies.info', '107.170.220.59')
## list domain records
domain_records(dom)

# create a domain record
dr <- domain_record_create(dom, "CNAME", name = "helloworld", data = "@")
domain_record(dom, dr$id)

# update a domain record
dru <- domain_record_update(domain_record = dr, name = "blog")

# delete a domain record
domain_record_delete(dr)

## End(Not run)

---

**as.image**

*Get list of images and their metadata, or a single image*

**Description**

Get list of images and their metadata, or a single image

**Usage**

```r
as.image(x)
```

```r
images(private = FALSE, type = NULL, page = 1, per_page = 25,
       public = TRUE, ...)
```

```r
image(id, ...)
```

**Arguments**

- **x**: Object to coerce to an image.
- **private**: Include public images? If FALSE, returns only the images that you’ve created (with snapshots).
- **type**: (character) One of distribution or application. Default: NULL (no type parameter passed)
- **page**: Page to return. Default: 1.
- **per_page**: Number of results per page. Default: 25.
- **public**: Include public images? If FALSE, returns only the images that you’ve created (with snapshots).
- **...**: Additional arguments passed down to low-level API function (do_*)
- **id**: (numeric) Image id.
as.snapshot

Examples

```sh
## Not run:
images()

# list private images
images(private = TRUE)

# list by type
images(type = "distribution")
images(type = "application")

# paging
images(per_page = 3)
images(per_page = 3, page = 2)

## End(Not run)
```

---

as.snapshot          Snapshot operations

Description

- **snapshot** retrieve a snapshot
- **snapshots** list snapshots, all, droplets, or volumes
- **snapshot_delete** delete a snapshot

Usage

```sh
as.snapshot(x)
snapshots(type = NULL, ...)
snapshot(id, ...)
snapshot_delete(snapshot, ...)
```

Arguments

- **x** Object to coerce to an snapshot
- **type** (character) NULL (all snapshots), or one of droplet (droplet snapshots) or volume (volume snapshots)
- **...** Additional options passed down to `GET, POST`, etc.
- **id** A snapshot id (varies depending on droplet or volume ID)
- **snapshot** A snapshot, or something that can be coerced to a snapshot by `as.snapshot`.  

Examples

```r
## Not run:
# list all snapshots
(res <- snapshots())

# list droplet snapshots
snapsshots(type = "droplet")

# list volume snapshots
snapsshots(type = "volume")

# get a single snapshot
snapshot(res[[1]]$id)

# delete a snapshot
## a whole snapshot class object
snapshot_delete(res[[2]])
## by id
snapshot_delete(res[[2]]$id)
## by name
snapshot_delete(res[[2]]$name)

# delete many snapshots
lapply(snapsshots(), snapshot_delete)

## End(Not run)
```

---

as.volume

### Block storage operations

#### Description

- `volume` get a single volume
- `volumes` list volumes
- `volume_create` create a volume
- `volume_snapshot_create` create a snapshot of a volume
- `volume_snapshots` list snapshots for a volume
- `volume_delete` delete a volume

#### Usage

```r
as.volume(x)

volumes(...) 

volume(volume, ...)
```
as.volume

volume_create(name, size, description = NULL, region = "nyc1", snapshot_id = NULL, ...)

volume_snapshot_create(volume, name, ...)

volume_snapshots(volume, ...)

volume_delete(volume, ...)

Arguments

x Object to coerce to an volume

... Additional options passed down to GET, POST, etc.

volume A volume, or something that can be coerced to a volume by as.volume.

name (character) Name of the new volume. required.

size (integer) The size of the Block Storage volume in GiB

description (character) An optional free-form text field to describe a Block Storage volume.

region (character) The region where the Block Storage volume will be created. When setting a region, the value should be the slug identifier for the region. When you query a Block Storage volume, the entire region object will be returned. Should not be specified with a snapshot_id. Default: nyc1

snapshot_id (integer) The unique identifier for the volume snapshot from which to create the volume. Should not be specified with a region_id.

Details

note that if you delete a volume, and it has a snapshot, the snapshot still exists, so beware

Examples

## Not run:
# list volumes
volumes()

# create a volume
vol1 <- volume_create('testing', 5)
vol2 <- volume_create('foobar', 6)

# create snapshot of a volume
xx <- volume_snapshot_create(vol2, "howdy")

# list snapshots for a volume
volume_snapshots(xx)

# list volumes again
res <- volumes()

# get a single volume
## a whole volume class object
debian

Helpers for managing a debian droplets.

Description

Helpers for managing a debian droplets.

Usage

debian_add_swap(droplet)

debian_install_r(droplet)

debian_install_rstudio(droplet, user = "rstudio", password = "rstudio", version = "0.99.484")

debian_install_shiny(droplet, version = "1.4.0.756")

debian_apt_get_update(droplet)

debian_apt_get_install(droplet, ...)

install_r_package(droplet, package, repo = "http://cran.rstudio.com")

Arguments

droplet A droplet, or object that can be coerced to a droplet by \texttt{as.droplet}.
user Default username for Rstudio.
password Default password for Rstudio.
docklets_create

docklets_create

Docklets: docker on droplets - create many docklets

Description

Docklets: docker on droplets - create many docklets

Usage

docklets_create(names = NULL, size = getOption("do_size", "1gb"),
    region = getOption("do_region", "sfo1"),
    ssh_keys = getOption("do_ssh_keys", NULL),
    backups = getOption("do_backups", NULL), ipv6 = getOption("do_ipv6",
    NULL), private_networking = getOption("do_private_networking", NULL),
    tags = NULL, wait = TRUE, image = "docker", ...)

Arguments

names (character) Names of the droplets. The human-readable string you wish to use when displaying the Droplet name. The name, if set to a domain name managed in the DigitalOcean DNS management system, will configure a PTR record for the Droplet. The name set during creation will also determine the hostname for the Droplet in its internal configuration. Default: picks a random name from words if none supplied.

size (character) Size slug identifier. See sizes() for a complete list. Default: 512mb, the smallest
region  (character) The unique slug identifier for the region that you wish to deploy in. See regions() for a complete list. Default: sfo1

ssh_keys  (character) A character vector of key names, an integer vector of key ids, or NULL, to use all keys in your account. Accounts with the corresponding private key will be able to log in to the droplet. See keys() for a list of the keys that you've added. Default: NULL

backups  (logical) Enable backups. A boolean indicating whether automated backups should be enabled for the droplet. Automated backups can only be enabled when the droplet is created, and cost extra. Default: FALSE

ipv6  (logical) A boolean indicating whether IPv6 is enabled on the droplet.

private_networking  (logical) Use private networking. Private networking is currently only available in certain regions. Default: FALSE

tags  (character) A vector of tag names to apply to the Droplet after it is created. Tag names can either be existing or new tags. Default: list()

wait  If TRUE (default), wait until droplet has been initialised and is ready for use. If set to FALSE we return a droplet object right away after droplet creation request has been sent. Note that there won’t be an IP address in the object yet. Note that waiting means we ping the DigitalOcean API to check on the status of your droplet, which uses up your API requests. The option do.wait_time can be set to any positive integer to determine how many seconds between pings. The default is 1 sec. Note that if you are creating droplets in a loop, parallel or otherwise, set do.wait_time within the loop instead of outside of it.

image  (character/numeric) The image ID of a public or private image, or the unique slug identifier for a public image. This image will be the base image for your droplet. See images() for a complete list. Default: ubuntu-14-04-x64

...  Additional options passed down to POST

Value

Two or more droplet objects

Examples

## Not run:
# if no names given, creates two droplets with random names
docklets_create()

# give names
docklets_create(names = c('drop1', 'drop2'))
docklets_create(names = c('drop3', 'drop4'))

## End(Not run)
docklet_create

Docklets: docker on droplets.

Description

Docklets: docker on droplets.

Usage

docklet_create(name = random_name(), size = getOption("do_size", "1gb"),
region = getOption("do_region", "sfo1"),
ssh_keys = getOption("do_ssh_keys", NULL),
backups = getOption("do_backups", NULL), ipv6 = getOption("do_ipv6", NULL),
private_networking = getOption("do_private_networking", NULL),
tags = list(), wait = TRUE, image = "docker", ...)
docklet_ps(droplet, all = TRUE, ssh_user = "root")
docklet_images(droplet, all = TRUE, ssh_user = "root")
docklet_pull(droplet, repo, ssh_user = "root")
docklet_run(droplet, ..., rm = FALSE, name = NULL, ssh_user = "root")
docklet_stop(droplet, container, ssh_user = "root")
docklet_rm(droplet, container, ssh_user = "root")
docklet_docker(droplet, cmd, args = NULL, docker_args = NULL,
ssh_user = "root")
docklet_rstudio(droplet, user = "rstudio", password = "rstudio",
email = "rstudio@example.com", img = "rocker/rstudio", port = "8787",
volume = "", dir = "", browse = TRUE, add_users = FALSE,
ssh_user = "root")
docklet_rstudio_addusers(droplet, user = "rstudio", password = "rstudio",
img = "rocker/rstudio", port = "8787")
docklet_shinyserver(droplet, img = "rocker/shiny", port = "3838",
volume = "", dir = "", browse = TRUE, ssh_user = "root")
docklet_shinyapp(droplet, path, img = "rocker/shiny", port = "80",
dir = "", browse = TRUE, ssh_user = "root")
**Arguments**

- **name** (character) Name of the droplet. The human-readable string you wish to use when displaying the Droplet name. The name, if set to a domain name managed in the DigitalOcean DNS management system, will configure a PTR record for the Droplet. The name set during creation will also determine the hostname for the Droplet in its internal configuration. Default: picks a random name from words if none supplied.

- **size** (character) Size slug identifier. See `sizes()` for a complete list. Default: 512mb, the smallest

- **region** (character) The unique slug identifier for the region that you wish to deploy in. See `regions()` for a complete list. Default: sfo1

- **ssh_keys** (character) A character vector of key names, an integer vector of key ids, or NULL, to use all keys in your account. Accounts with the corresponding private key will be able to log in to the droplet. See `keys()` for a list of the keys that you’ve added. Default: NULL

- **backups** (logical) Enable backups. A boolean indicating whether automated backups should be enabled for the droplet. Automated backups can only be enabled when the droplet is created, and cost extra. Default: FALSE

- **ipv6** (logical) A boolean indicating whether IPv6 is enabled on the droplet.

- **private_networking** (logical) Use private networking. Private networking is currently only available in certain regions. Default: FALSE

- **tags** (character) A vector of tag names to apply to the Droplet after it is created. Tag names can either be existing or new tags. Default: list()

- **wait** If TRUE (default), wait until droplet has been initialised and is ready for use. If set to FALSE we return a droplet object right away after droplet creation request has been sent. Note that there won’t be an IP address in the object yet. Note that waiting means we ping the DigitalOcean API to check on the status of your droplet, which uses up your API requests. The option `do.wait_time` can be set to any positive integer to determine how many seconds between pings. The default is 1 sec. Note that if you are creating droplets in a loop, parallel or otherwise, set `do.wait_time` within the loop instead of outside of it.

- **image** (character/numeric) The image ID of a public or private image, or the unique slug identifier for a public image. This image will be the base image for your droplet. See `images()` for a complete list. Default: ubuntu-14-04-x64

... For `docklet_create`, additional options passed down to POST. For `docklet_run`, additional arguments combined and applied to docker statement.

- **droplet** A droplet, or something that can be coerced to a droplet by `as.droplet`.

- **all** (logical) List all containers or images. Default: TRUE

- **ssh_user** (character) User account for ssh commands against droplet. Default: root

- **repo** (character) Docker name, can be local to the Droplet or remote, e.g., rocker/rstudio

- **rm** (logical) Automatically remove the container when it exits. Default: FALSE

- **container** (character) Container name, can be partial (though has to be unique)
### fish

*fish*

- **docklet create**

  - `cmd` (character) A docker command (e.g., "run")
  - `args` (character) Docker args
  - `docker_args` (character) Docker args
  - `user` (character) User name. Default: "rstudio"
  - `password` (character) Password. Default: "rstudio"
  - `email` (character) Password. Default: "rstudio@example.com"
  - `img` (character) Docker image (not a DigitalOcean image). Default: ' rocker/rstudio' 
  - `port` (character) Port. Default: 8787
  - `volume` (character) Volume. Can use to bind a volume.
  - `dir` (character) Working directory inside the container.
  - `browse` (logical) If TRUE, open RStudio instance in your default browser.
  - `add_users` (logical) Add users or not when installing RStudio server. Default: FALSE
  - `path` (character) Path to a directory with Shiny app files

### Value

all functions return a droplet

### URLs

If you need to figure out the URL for your RStudio or Shiny server instance, you can construct like http://<ip address>:<port> where IP address can most likely be found like `d$networks$v4[[1]]$ip_address` and the port is the port you set in the function call.

### See Also

- `docklets_create`

### Examples

```r
# Not run:
d <- docklet_create()
d %>% docklet_pull("dockerpina/sqlite")
d %>% docklet_images()

# sqlite
d %>% docklet_run("dockerpina/sqlite", "sqlite3 --version", rm = TRUE)
d %>% docklet_ps()

# cowsay
d %>% docklet_pull("chuanwen/cowsay")
d %>% docklet_run("chuanwen/cowsay", rm = TRUE)

# docker images
d %>% docklet_images()

# install various R versions via rocker
```
Get information on a single domain or all your domains.

**domains**

Get information on a single domain or all your domains.

**Usage**

- `domains(...)`
- `as.domain(x)`
- `domain(x, ...)`
domain_create

Arguments

... Further args passed on the curl call to the web.

x (character) Required. Domain name

Examples

## Not run:

```r
domains()
```

## End(Not run)

domain_create Create/delete domains.

Description

Create/delete domains.

Usage

```r
domain_create(name, ip_address, ...)
domain_delete(domain, ...)
```

Arguments

name (character) Required. The domain name to add to the DigitalOcean DNS management interface. The name must be unique in DigitalOcean’s DNS system. The request will fail if the name has already been taken.

ip_address (character) Required. An IP address for the domain's initial A record.

... Further args passed on the curl call to the web.

domain A domain to modify

Examples

## Not run:

```r
d <- domain_create('tablesandchairsbunnies.info', '107.170.220.59')
domain_delete(d)
```

## End(Not run)
do_oauth  

Authorize with Digital Ocean.

Description

This function is run automatically to allow analogsea to access your digital ocean account.

Usage

```r
do_oauth(app = do_app, reauth = FALSE)
```

Arguments

- **app**: An `oauth_app` for DO. The default uses the standard ROpenSci application.
- **reauth**: (logical) Force re-authorization?

Details

There are two ways to authorise analogsea to work with your digital ocean account:

- Generate a personal access token at https://cloud.digitalocean.com/settings/api/tokens and record in the `DO_PAT` envvar.
- Interactively login into your DO account and authorise with OAuth.

Using `DO_PAT` is recommended.

---

do_options  

Set Digital Ocean options including ssh keys, etc.

Description

This function sets options and prints them so you know what options are set.

Usage

```r
do_options(size = NULL, image = NULL, region = NULL, ssh_keys = NULL, 
            private_networking = NULL, backups = NULL, ipv6 = NULL, unset = FALSE)
```
do_options 21

Arguments

size (optional) A Digital Ocean size slug name, e.g., '1gb'. Saved in options as 'do_size'

image (optional) A Digital Ocean image name, e.g., 'ubuntu-14-04-x64'. Saved in options as 'do_image'

region (optional) A Digital Ocean region name, e.g., 'nyc1'. Saved in options as 'do_region'

ssh_keys (optional) One or more ssh key id numbers or fingerprints. Put many in a list or vector. Saved in options as 'do_ssh_keys'

private_networking (optional) A logical, whether to use private networking or not. Saved in options as 'do_private_networking'

backups (optional) A logical, whether to enable backups. Automated backups can only be enabled when the Droplet is created. Saved in options as 'do_backups'

ipv6 (optional) A boolean indicating whether IPv6 is enabled on the Droplet. Saved in options as 'do_ipv6'

unset (optional) A boolean. If TRUE, unsets options so as to use defaults in droplet_create. If FALSE (default) your options are used in droplet_create.

Details

These options are read and used by droplet_create.

You can only set one value for each of size, image, and region, but multiple values for ssh_keys as you can use multiple ssh keys on one DO droplet.

Keep in mind that there are defaults set for size, image, and region in droplet_create.

Examples

```r
## Not run:
do_options()
do_options(ssh_keys=89103)
getOption('do_ssh_keys')
do_options(size="8gb")
do_options(size="1gb", image='ubuntu-14-04-x64', region='nyc1')
getOption('do_size')
getOption('do_image')
getOption('do_region')

## End(Not run)
```
droplet

Retrieve a single droplet.

Description

Retrieve a single droplet.

Usage

droplet(id, ...)

as.droplet(x)

## S3 method for class 'droplet'
summary(object, ...)

Arguments

id (integer) Droplet id.

... Additional arguments passed down to low-level API function (do_*)

x Object to coerce. Can be an integer (droplet id), string (droplet name), a droplet (duh), or an action (which waits until complete then returns the droplet)

object Droplet object to pass to summary

Examples

## Not run:
droplet(1234)

as.droplet("my-favourite-droplet")
as.droplet(10)
as.droplet(droplets()[[1]])

droplet(1234) %>% summary

## End(Not run)

droplets

List all available droplets.

Description

List all available droplets.
droplets_cost

**Usage**

droplets(..., page = 1, per_page = 25, tag = NULL)

**Arguments**

- **...** Additional arguments passed down to low-level API function (do_*)
- **page** Page to return. Default: 1.
- **per_page** Number of results per page. Default: 25.
- **tag** (character) Name of a tag. optional

**Examples**

```r
## Not run:
droplets()
droplets(per_page = 2)
droplets(per_page = 2, page = 2)

# list droplets by tag
tag_create(name = "stuffthings")
d <- droplet_create()
tag_resource(name = "stuffthings", resource_id = d$id,
  resource_type = "droplet")
droplets(tag = "stuffthings")

## End(Not run)
```

droplets_cost  \hspace{1cm} \textit{Calculate cost across droplets}

**Description**

Calculate cost across droplets

**Usage**

droplets_cost(x)

**Arguments**

- **x** Object to coerce. Can be an integer (droplet id), string (droplet name), a droplet (duh)
Examples

```r
## Not run:
droplets() %>% droplets_cost()
droplets()[[2]] %>% droplets_cost()
droplets()[2:4] %>% droplets_cost()
droplets_cost("FatedSpaghetti")
droplets_cost(11877599)
## End(Not run)
```

droplets_create

Create many new droplets.

Description

There are defaults for each of size, image, and region so that a quick one-liner with one parameter is possible: simply specify the name of the droplet and you’re up and running.

Usage

```r
droplets_create(names = NULL, size = getOption("do_size", "512mb"),
image = getOption("do_image", "ubuntu-14-04-x64"),
region = getOption("do_region", "sfo1"),
ssh_keys = getOption("do_ssh_keys", NULL),
backups = getOption("do_backups", NULL), ipv6 = getOption("do_ipv6",
NULL), private_networking = getOption("do_private_networking", NULL),
tags = NULL, user_data = NULL, cloud_config = NULL, wait = TRUE, ...)
```

Arguments

- **names** (character) Names of the droplets. The human-readable string you wish to use when displaying the Droplet name. The name, if set to a domain name managed in the DigitalOcean DNS management system, will configure a PTR record for the Droplet. The name set during creation will also determine the hostname for the Droplet in its internal configuration. Default: picks a random name from the DigitalOcean DNS management system if none supplied.

- **size** (character) Size slug identifier. See `sizes()` for a complete list. Default: 512mb, the smallest image (character/numeric) The image ID of a public or private image, or the unique slug identifier for a public image. This image will be the base image for your droplet. See `images()` for a complete list. Default: ubuntu-14-04-x64

- **region** (character) The unique slug identifier for the region that you wish to deploy in. See `regions()` for a complete list. Default: sfo1

- **ssh_keys** (character) A character vector of key names, an integer vector of key ids, or NULL, to use all keys in your account. Accounts with the corresponding private key will be able to log in to the droplet. See `keys()` for a list of the keys that you’ve added. Default: NULL
`droplets_create`  

- **backups** (logical) Enable backups. A boolean indicating whether automated backups should be enabled for the droplet. Automated backups can only be enabled when the droplet is created, and cost extra. Default: FALSE

- **ipv6** (logical) A boolean indicating whether IPv6 is enabled on the droplet.

- **private_networking** (logical) Use private networking. Private networking is currently only available in certain regions. Default: FALSE

- **tags** (character) A vector of tag names to apply to the Droplet after it is created. Tag names can either be existing or new tags. Default: list()

- **user_data** (character) Gets passed to the droplet at boot time. Not all regions have this enabled, and is not used by all images.

- **cloud_config** (character) Specify the name of a cloud config template to automatically generate and submit in user metadata. Setting this is best practice: the built-in templates use security best practices (disabling root log-in, security autoupdates) to make it harder to hack your droplet.

- **wait** If TRUE (default), wait until droplet has been initialised and is ready for use. If set to FALSE we return a droplet object right away after droplet creation request has been sent. Note that there won’t be an IP address in the object yet. Note that waiting means we ping the DigitalOcean API to check on the status of your droplet, which uses up your API requests. The option `do.wait.time` can be set to any positive integer to determine how many seconds between pings. The default is 1 sec. Note that if you are creating droplets in a loop, parallel or otherwise, set `do.wait.time` within the loop instead of outside of it.

... Additional options passed down to `POST`

**Details**

Note that if you exit the R session or kill the function call after it’s in waiting process (the string of ...), the droplet creation will continue.

**Value**

Two or more droplet objects

**Examples**

```r
# Not run:
# if no names given, creates two droplets with random names
droplets_create()

droplets_create(names = c('drop1', 'drop2'))

droplets_create(names = c('drop3', 'drop4'))

# add tags
(d <- droplets_create(tags = 'mystuff'))
invisible(lapply(d, summary))

# End(Not run)
```


**Description**

These droplet actions have no further arguments.

**Usage**

- `droplet_reboot(droplet, ...)`
- `droplet_power_cycle(droplet, ...)`
- `droplet_shutdown(droplet, ...)`
- `droplet_power_off(droplet, ...)`
- `droplet_power_on(droplet, ...)`
- `droplet_reset_password(droplet, ...)`
- `droplet_enable_ipv6(droplet, ...)`
- `droplet_enable_private_networking(droplet, ...)`
- `droplet_enable_backups(droplet, ...)`
- `droplet_disable_backups(droplet, ...)`
- `droplet_upgrade(droplet, ...)`

**Arguments**

- **droplet**: A droplet, or something that can be coerced to a droplet by `as.droplet`.
- **...**: Additional options passed down to low-level API method.

**Details**

- **reboot**: This method allows you to reboot a droplet. This is the preferred method to use if a server is not responding.
- **powercycle**: This method allows you to power cycle a droplet. This will turn off the droplet and then turn it back on.
- **shutdown**: Shutdown a running droplet. The droplet will remain in your account and you will continue to be charged for it.
- **power_off**: Shutdown a running droplet. The droplet will remain in your account and you will continue to be charged for it.
reset_password  This method will reset the root password for a droplet. Please be aware that this will reboot the droplet to allow resetting the password.

enable_ipv6  Enable IPv6 networking on an existing droplet (within a region that has IPv6 available).

enable_private_networking  Enable private networking on an existing droplet (within a region that has private networking available)

disable_backups  Disables backups for a droplet.

d power_on  Turn on a droplet that’s turned off.

Examples

```r
## Not run:
d <- droplets()
d[[1]]  droplet_reboot()
d[[2]]  droplet_power_cycle()

d <- droplet_create()
d  droplet_enable_backups()
d  droplet_summary()

## End(Not run)
```

---

droplet_actions  Retrieve a droplet action or list all actions associated with a droplet.

Description

Retrieve a droplet action or list all actions associated with a droplet.

Usage

```r
droplet_actions(droplet, actionid = NULL, ...)
```

Arguments

- **droplet**: A droplet, or something that can be coerced to a droplet by `as.droplet`.
- **actionid**: (integer) Optional. An action id.
- **...**: Additional options passed down to low-level API method.

Examples

```r
## Not run:
droplet_actions(2428384)
droplet_actions(2428384, actionid=31223385)

## End(Not run)
```
**droplet_create**  
Create a new droplet.

**Description**

There are defaults for each of size, image, and region so that a quick one-liner with one parameter is possible: simply specify the name of the droplet and you’re up and running.

**Usage**

droplet_create(name = random_name(), size = getOption("do_size", "512mb"),  
image = getOption("do_image", "ubuntu-14-04-x64"),  
region = getOption("do_region", "sfo1"),  
ssh_keys = getOption("do_ssh_keys", NULL),  
backups = getOption("do_backups", NULL), ipv6 = getOption("do_ipv6",  
NULL), private_networking = getOption("do_private_networking", NULL),  
tags = list(), user_data = NULL, cloud_config = NULL, wait = TRUE,  
...)

**Arguments**

- **name** (character) Name of the droplet. The human-readable string you wish to use when displaying the Droplet name. The name, if set to a domain name managed in the DigitalOcean DNS management system, will configure a PTR record for the Droplet. The name set during creation will also determine the hostname for the Droplet in its internal configuration. Default: picks a random name from `words` if none supplied.

- **size** (character) Size slug identifier. See `sizes()` for a complete list. Default: 512mb, the smallest

- **image** (character/numeric) The image ID of a public or private image, or the unique slug identifier for a public image. This image will be the base image for your droplet. See `images()` for a complete list. Default: ubuntu-14-04-x64

- **region** (character) The unique slug identifier for the region that you wish to deploy in. See `regions()` for a complete list. Default: sfo1

- **ssh_keys** (character) A character vector of key names, an integer vector of key ids, or NULL, to use all keys in your account. Accounts with the corresponding private key will be able to log in to the droplet. See `keys()` for a list of the keys that you’ve added. Default: NULL

- **backups** (logical) Enable backups. A boolean indicating whether automated backups should be enabled for the droplet. Automated backups can only be enabled when the droplet is created, and cost extra. Default: FALSE

- **ipv6** (logical) A boolean indicating whether IPv6 is enabled on the droplet.

- **private_networking** (logical) Use private networking. Private networking is currently only available in certain regions. Default: FALSE
tags (character) A vector of tag names to apply to the Droplet after it is created. Tag names can either be existing or new tags. Default: list()

tags

user_data (character) Gets passed to the droplet at boot time. Not all regions have this enabled, and is not used by all images.

user_data

calendar_config (character) Specify the name of a cloud config template to automatically generate calendar_config and submit in user metadata. Setting this is best practice: the built-in templates use security best practices (disabling root log-in, security autoupdates) to make it harder to hack your droplet.

calendar_config

wait If TRUE (default), wait until droplet has been initialised and is ready for use. If set to FALSE we return a droplet object right away after droplet creation request has been sent. Note that there won’t be an IP address in the object yet. Note that waiting means we ping the DigitalOcean API to check on the status of your droplet, which uses up your API requests. The option do.wait_time can be set to any positive integer to determine how many seconds between pings. The default is 1 sec. Note that if you are creating droplets in a loop, parallel or otherwise, set do.wait_time within the loop instead of outside of it.

wait

Additional options passed down to POST

Details

Note that if you exit the R session or kill the function call after it’s in waiting process (the string of ...), the droplet creation will continue.

Value

A droplet object

Examples

## Not run:
# by default we give your droplet a name
droplet_create()

# you can set your own droplet name
droplet_create('droppinit')

# set name, size, image, and region
droplet_create(name="newdrop", size = '512mb', image = 'ubuntu-14-04-x64',
region = 'sfo1')

# use an ssh key
droplet_create(ssh_keys=89103)

# add tags
(d <- droplet_create(tags = c('venus', 'mars'))) summary(d)

## End(Not run)
**droplet_delete**  
*Delete a droplet.*

### Description

This method deletes one of your droplets - this is irreversible.

### Usage

```r
droplet_delete(droplet = NULL, tag = NULL, ...)
```

### Arguments

- **droplet**: A droplet, or something that can be coerced to a droplet by `as.droplet`.
- **tag**: (character) Name of a tag. Optional
- **...**: Additional options passed down to low-level API method.

### Examples

```r
## Not run:
drops <- droplets()
drops[[1]] %>% droplet_delete()
drops[[2]] %>% droplet_delete()
droplet_create() %>% droplet_delete()

droplet_delete("lombard")
droplet_delete(12345)

# Delete all droplets
lapply(droplets(), droplet_delete)

# delete droplets by tag
## first, create a tag, then a droplet, then tag it
tag_create(name = "foobar")
e <- droplet_create()
tag_resource(name = "foobar", resource_id = e$id)
droplets(tag = "foobar")
## then delete the droplet by tag name
droplet_delete(tag = "foobar")

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

Perform actions on one or more droplets associated with a tag

**Description**

Perform actions on one or more droplets associated with a tag

**Usage**

```r
droplet_do_actions(name, type, ...)
```

**Arguments**

- **name** (character) Name of the tag. Required.
- **type** (character) action type, one of 'power_cycle', 'power_on', 'power_off', 'shutdown', 'enable_private_networking', 'enable_ipv6', 'enable_backups', 'disable_backups', or 'snapshot'. Required.
- **...** Additional options passed down to POST

**Examples**

```r
## Not run:
tag_create(name = "pluto")d <- droplet_create()
tag_resource(name = "pluto", resource_id = d$id)(x <- droplet_do_actions(name = "pluto", type = "power_off"))# wait until completed, check with action(xx$actions[[1]]$id)droplet_do_actions(name = "pluto", type = "power_on")

## End(Not run)
```

### droplet_execute

Execute R code on a droplet.

**Description**

Execute R code on a droplet.

**Usage**

```r
droplet_execute(droplet, code, verbose = TRUE)
```

**Arguments**

- **droplet** A droplet, or object that can be coerced to a droplet by `as.droplet`.
- **code** Code to execute on a droplet.
- **verbose** (logical) Print messages. Default: TRUE
Details

Assumes that the droplet has R installed.

Examples

```r
## Not run:
d <- droplet_create()
droplet_ssh("apt-get update")
debian_add_swap()
debian_install_r()
results <- droplet_execute({
  x <- letters
  numbers <- runif(1000)
})
results$x
results$numbers

droplet_delete(d)
```

## End(Not run)

---

droplet_freeze

*Freeze/thaw droplets.*

Description

Freeze powers off the droplet, snapshots to create an image, and deletes the droplet. Thaw performs the inverse: it takes an image and turns it into a running droplet.

Usage

```r
droplet_freeze(droplet, name = droplet$name, ...)
droplet_thaw(image, ...)
```

Arguments

- `droplet` A droplet, or something that can be coerced to a droplet by `as.droplet`.
- `name` Name for the image to be created, or to be used to create a new droplet. Defaults to name of the droplet.
- `...` For freeze, further args passed on to `droplet_snapshot`; thaw, args passed on to `droplet_create`.
- `image` Image to thaw into a droplet.
Value

droplet_freeze accepts a droplet as first argument, and returns an image; droplet_thaw does the opposite: it accepts an image as first argument, and returns a droplet.

Examples

```r
## Not run:
# freeze
droplet_create(region = 'nyc3') %>% droplet_freeze()

# thaw
droplet_thaw(image='chiromantical-1412718795', region='nyc3')

## End(Not run)
```

droplet_kernels_list  List all available kernels for a droplet.

Description

List all available kernels for a droplet.

Usage

droplet_kernels_list(droplet, ...)

Arguments

droplet  A droplet, or something that can be coerced to a droplet by as.droplet.

...  Additional options passed down to low-level API method.

Examples

```r
## Not run:
droplets()[[1]] %>% droplet_kernels_list

## End(Not run)
```
**droplet_modify**

*Modify a droplet.*

**Description**

These methods allow you to modify existing droplets.

**Usage**

- `droplet_resize(droplet, size, ...)`
- `droplet_rebuild(droplet, image, ...)`
- `droplet_rename(droplet, name, ...)`
- `droplet_change_kernel(droplet, kernel, ...)`

**Arguments**

- `droplet` A droplet, or something that can be coerced to a droplet by `as.droplet`.
- `size` (character) Size slug (name) of the image size. See `sizes`.
- `...` Additional options passed down to low-level API method.
- `image` (optional) The image ID of the backup image that you would like to restore.
- `name` (character) The new name for the droplet.
- `kernel` (numeric) The ID of the new kernel.

**Details**

- **resize** Resize a specific droplet to a different size. This will affect the number of processors and memory allocated to the droplet.
- **rebuild** Reinstall a droplet with a default image. This is useful if you want to start again but retain the same IP address for your droplet.
- **rename** Change the droplet name.
- **change_kernel** Change kernel ID.

Beware: `droplet_resize()` does not seem to work, see `resize()`

**Examples**

```r
## Not run:
droplets()[[1]] %>% droplet_rename(name='newname')

## End(Not run)
```
**droplet_reuse**

**Reuse a droplet or image by name**

---

**Description**

Reuse a droplet or image by name

**Usage**

```text
droplet_reuse(name, ...)
```

**Arguments**

- **name**: A name that could be a droplet or image name
- **...**: Named options passed on to `droplet_create`.

**Details**

Internally, we call the `droplets` and `images` (with `private = TRUE`) to get list of your droplets and images - and we check against those.

**Value**

A droplet

**Examples**

```text
# Not run:
# matches droplet that exists
droplet_reuse(name = 'BeguiledAmmonia')

# matching image that exists
droplet_reuse(name = 'hadleyverse1', size = "1gb")

# no matching droplet or image
droplet_reuse(name = 'tablesandchairs')
```

# End(Not run)
droplet_snapshot  

Take and restore snapshots.

Description

**snapshot**  Take a snapshot of the droplet once it has been powered off, which can later be restored or used to create a new droplet from the same image. Please be aware this may cause a reboot.

**snapshots_list**  List available snapshots

**backups_list**  List available snapshots

**restore**  Restore a droplet with a previous image or snapshot. This will be a mirror copy of the image or snapshot to your droplet. Be sure you have backed up any necessary information prior to restore.

Usage

droplet_snapshot(droplet, name = NULL, ...)
droplet_snapshots_list(droplet, ...)
droplet_restore(droplet, image, ...)
droplet_backups_list(droplet, ...)

Arguments

- **droplet**  A droplet number or the result from a call to droplets()
- **name**  (character) Optional. Name of the new snapshot you want to create. If not set, the snapshot name will default to the current date/time
- **...**  Additional options passed down to POST
- **image**  (optional) The image ID of the backup image that you would like to restore.

Examples

```r
## Not run:
d <- droplet_create()
d %>% droplet_snapshots_list()
d %>% droplet_backups_list()

d %>%
  droplet_power_off() %>%
  droplet_snapshot() %>%
  droplet_power_on() %>%
  droplet_snapshots_list()

# To delete safely
d %>%
```
droplet_ssh

Remotely execute ssh code, upload & download files.

Description
Assumes that you have ssh & scp installed, and password-less login set up on the droplet.

Usage

droplet_ssh(droplet, ..., user = "root", verbose = FALSE)

droplet_upload(droplet, local, remote, user = "root", verbose = FALSE)

droplet_download(droplet, remote, local, user = "root", verbose = FALSE, overwrite = FALSE)

Arguments

droplet A droplet, or something that can be coerced to a droplet by as.droplet.

... Shell commands to run. Multiple commands are combined with && so that execution will halt after the first failure.

user User name. Defaults to "root".

verbose If TRUE, will print command before executing it.

local, remote Local and remote paths.

overwrite If TRUE, then overwrite destination files if they already exist.

Details
Uploads and downloads are recursive, so if you specify a directory, everything inside the directory will also be downloaded.

Value
On success, the droplet (invisibly). On failure, throws an error.
Examples

```r
## Not run:

d <- droplet_create() %>% droplet_wait()

# Upgrade system packages

d %>%
  droplet_ssh("apt-get update") %>%
  droplet_ssh("sudo apt-get upgrade -y --force-yes") %>%
  droplet_ssh("apt-get autoremove -y")

# Install R

d %>%
  droplet_ssh("apt-get install r-base-core r-base-dev --yes --force-yes")

# Upload and download files

tmp <- tempfile()

saveRDS(mtcars, tmp)

d %>% droplet_upload(tmp, "mtcars.rds")

tmp2 <- tempfile()

d %>% droplet_download("mtcars.rds", tmp2)

mtcars2 <- readRDS(tmp2)

stopifnot(all.equal(mtcars, mtcars2))

## End(Not run)
```

---

droplet_upgrades_list  List all droplets that are scheduled to be upgraded.

Description

List all droplets that are scheduled to be upgraded.

Usage

droplet_upgrades_list(...)

Arguments

... Additional options passed down to low-level API method.

Examples

```r
## Not run:

droplet_upgrades_list()

## End(Not run)
```
**droplet_wait**

*Wait for a droplet to be ready.*

**Description**

Wait for a droplet to be ready.

**Usage**

`droplet_wait(droplet)`

**Arguments**

- `droplet`  
  A droplet, or something that can be coerced to a droplet by `as.droplet`.

**Examples**

```r
## Not run:
droplet_create() %>% droplet_wait()

## End(Not run)
```

**image_actions**

*Retrieve an action associated with a particular image id.*

**Description**

Retrieve an action associated with a particular image id.

**Usage**

`image_actions(image, action_id, ...)`

**Arguments**

- `image`  
  An image to modify.
- `action_id`  
  An action id associated with an image.
- `...`  
  Options passed on to `httr::GET`. Must be named, see examples.

**Examples**

```r
## Not run:
image_actions(5710271, 31221438)

## End(Not run)
```
image_convert

Convert an backup image to a snapshot.

Description

Convert an backup image to a snapshot.

Usage

image_convert(image, ...)

Arguments

image
An image to modify.

...Options passed on to httr::GET. Must be named, see examples.

Examples

## Not run:
# get a backup image
img <- images(TRUE)[[1]]
# then convert to a snapshot
# image_convert(img)

## End(Not run)

image_delete

Rename/delete an image

Description

There is no way to restore a deleted image so be careful and ensure your data is properly backed up before deleting it.

Usage

image_delete(image, ...)

image_rename(image, name, ...)

Arguments

image
An image to modify.

...Options passed on to httr::GET. Must be named, see examples.

name
(character) New name for image.
image_transfer

Examples

```
## Not run:
image_delete(5620385)

# Delete all of your snapshots
## BE CAREFUL WITH THIS ONE
# lapply(images(TRUE), image_delete)

## End(Not run)
```

documentation

image_transfer  Transfer an image to a specified region.

Description

Transfer an image to a specified region.

Usage

```
image_transfer(image, region, ...)
```

Arguments

- `image`: An image to modify.
- `region`: (numeric) Required. The region slug that represents the region target.
- `...`: Options passed on to `httr::GET`. Must be named, see examples.

Examples

```
## Not run:
image_transfer(image=images(TRUE)[[1]], region='nyc2')
image_transfer(image=images(TRUE)[[1]], region='ams2')

## End(Not run)
```

key-crud  Create, update, and delete ssh keys.

Description

Create, update, and delete ssh keys.
Usage

key_create(name, public_key, ...)

key_rename(key, name, ...)

key_delete(key, ...)

Arguments

ame (character) The name to give the new SSH key in your account.

public_key (character) A string containing the entire public key.

... Other options passed on to low-level API methods.

key (key) Key to modify.

Examples

## Not run:
k <- key_create("key", readLines("~/.ssh/id_rsa.pub"))
k <- key_rename(k, "new_name")
key_delete(k)

## End(Not run)

---

keys List your ssh keys, or get a single key

Description

List your ssh keys, or get a single key

Usage

keys(..., page = 1, per_page = 25)

key(x, ...)

as.sshkey(x)

Arguments

... Additional arguments passed down to low-level API function (do_*)


per_page Number of results per page. Default: 25.

x For key the numeric id. For as.sshkey, a number (the id), a string (the name), or a key.
Examples

```r
## Not run:
keys()
as.sshkey(328037)
as.sshkey("hadley")

## End(Not run)
```

**neighbors** | *List neighbors*
---|---

**Description**

List neighbors

**Usage**

```r
neighbors(...)
droplet_neighbors(droplet, ...)
```

**Arguments**

- `...`: Additional options passed down to low-level API method.
- `droplet`: A droplet, or something that can be coerced to a droplet by `as.droplet`.

**Examples**

```r
## Not run:
# List a droplet's neighbors on the same physical server
droplets()[[3]] %>% droplet_neighbors()
# List all neighbors on the same physical server
neighbors()

## End(Not run)
```

**nouns** | *Nouns to use for seeding random word selection when name not given for a droplet*
---|---

**Description**

Nouns to use for seeding random word selection when name not given for a droplet

**Details**

A vector of 1000 nouns From the GitHub repo [https://github.com/dariusk/corpora](https://github.com/dariusk/corpora) - the data is licensed CC0.
regions

Get list of regions and their metadata

**Description**

Get list of regions and their metadata

**Usage**

regions(page = 1, per_page = 25, ...)

**Arguments**

- **page**: Page to return. Default: 1.
- **per_page**: Number of results per page. Default: 25.
- **...**: Named options passed on to GET.

**Examples**

```r
## Not run:
regions()

## End(Not run)
```

resize

Resize a droplet by power off, snapshot, and create new droplet

**Description**

Resize a droplet by power off, snapshot, and create new droplet

**Usage**

resize(droplet, delete_original = TRUE, ...)

**Arguments**

- **droplet**: A droplet, or something that can be coerced to a droplet by as.droplet.
- **delete_original**: (logical) Delete original droplet. Default: TRUE
- **...**: Named options passed on to droplet_create.
Details

Note that you can not resize a droplet while it is powered on. Thus, this function powers off your droplet, makes a snapshot, then creates a new droplet from that snapshot. We use `droplet_wait` in between these steps to wait for each to finish. You can optionally delete the original droplet.

Value

A droplet

Examples

```r
## Not run:
D <- droplet_create()
D # current size is 512mb
D %>% resize(size = "2gb")

## End(Not run)
```

sizes

*Get all the available sizes that can be used to create a droplet.*

Description

Get all the available sizes that can be used to create a droplet.

Usage

```r
sizes(page = 1, per_page = 25, ...)
```

Arguments

- `per_page`: Number of results per page. Default: 25.
- `...`: Named options passed on to `GET`.

Value

A data.frame with available sizes (RAM, disk, no. CPU’s) and their costs

Examples

```r
## Not run:
sizes()

## End(Not run)
```
standardise_keys  

*Standardise specification of ssh keys.*

**Description**

Standardise specification of ssh keys.

**Usage**

\[\text{standardise_keys}(\text{ssh_keys} = \text{NULL})\]

**Arguments**

- `ssh_keys`  
  An integer vector of given key ids, a character vector of key ids, or NULL, to use all ssh keys in account.

**Value**

A integer vector of key ids.

**Examples**

```r
## Not run:
standardise_keys(123L)
standardise_keys("hadley")
## End(Not run)
```

---

**tags**  

*List tags*

**Description**

List tags

**Usage**

- `tags(...)`
- `tag(name, ...)`
- `as.tag(x)`
tag_create

Description
Create a tag

Usage
tag_create(name, ...)

Arguments
name (character) Name of the tag
... Additional options passed down to POST
**Value**

A tag object

**Examples**

```bash
## Not run:
tag_create(name = "venus")

## End(Not run)
```

---

**tag_delete**

*Delete a tag*

**Description**

Delete a tag

**Usage**

```bash
tag_delete(name, ...)
```

**Arguments**

- `name` (character) Name of the tag
- `...` Additional options passed down to `DELETE`

**Value**

nothing, if successful

**Examples**

```bash
## Not run:
tag_delete(name = "helloworld")

## End(Not run)
```
tag_resource  

Tag a resource

Description
Tag a resource

Usage

`tag_resource(name, resource_id = NULL, resource_type = "droplet", resources = NULL, ...)`

Arguments

- `name` (character) Name of the tag
- `resource_id` (integer) a droplet id
- `resource_type` (character) only "droplet" for now. Default: "droplet"
- `resources` (list) instead of `resource_id` and `resource_type` you can pass in a list to this parameter. see examples
  
  Additional options passed down to POST

Value
logical, TRUE if successful

Examples

```r
## Not run:
d <- droplet_create()
tag_resource(name = "stuffthings", resource_id = d$id,
  resource_type = "droplet")
tag_resource("stuffthings", resources = list(list(resource_id = d$id,
    resource_type = "droplet")))

## End(Not run)
```

tag_resource_delete  

Untag a resource

Description
Untag a resource
Usage

    tag_resource_delete(name, resource_id = NULL, resource_type = "droplet",
                        resources = NULL, ...)

Arguments

    name  (character) Name of the tag
    resource_id  (integer) a droplet id
    resource_type  (character) only "droplet" for now. Default: "droplet"
    resources  (list) instead of resource_id and resource_type you can pass in a list to this
                parameter. see examples
... Additional options passed down to DELETE

Value

    logical, TRUE if successful

Examples

    ## Not run:
    d <- droplet_create()
    tag_resource(name = "stuffthings", resource_id = d$id,
                 resource_type = "droplet")
    ## same as this because only allowed resource type right now is "droplet"
    # tag_resource(name = "stuffthings", resource_id = d$id)
    tag_resource_delete(name = "stuffthings", resource_id = d$id,
                         resource_type = "droplet")
    
    ## End(Not run)

---

volume_attach

**Attach a volume to a droplet**

Description

Attach a volume to a droplet

Usage

    volume_attach(volume, droplet, region = "nyc1", ...)
    volume_detach(volume, droplet, region = "nyc1", ...)
    volume_resize(volume, size, region = "nyc1", ...)
    volume_action(volume, actionid, ...)
    volume_actions(volume, page = 1, per_page = 25, ...)
volume_attach

Arguments

- **volume**: A volume, or something that can be coerced to a volume by `as.volume`.
- **droplet**: A droplet, or something that can be coerced to a droplet by `as.droplet`.
- **region**: (character) The region where the Block Storage volume will be created. When setting a region, the value should be the slug identifier for the region. When you query a Block Storage volume, the entire region object will be returned. Should not be specified with a snapshot_id. Default: nyc1
- **size**: (integer) The size of the Block Storage volume in GiB
- **actionid**: (integer) Optional. An action id.
- **page**: Page to return. Default: 1.
- **per_page**: Number of results per page. Default: 25.

Details

Note that there is a way to attach a volume to or remove from a droplet by name, but we only support doing this by ID. However, as the user, all you need to do is make a volume class object via `as.volume` and pass it to `volume_attach` or `volume_detach`, which is pretty darn easy.

Examples

```r
## Not run:
# resize a volume
## create a volume
(voll <- volume_create('foobar', 5))
## resize it
volume_resize(voll, 6)
volume(voll)

# attach a volume to a droplet
## create a droplet
(d <- droplet_create(region = "nyc1"))
## attach volume to droplet
volume_attach(voll, d)
## refresh droplet info, see volumes slot
droplet(d$id)

# detach a volume from a droplet
## create a droplet
(act <- volume_detach(voll, d))
## refresh droplet info, see volumes slot
droplet(d$id)

# list an action
volume_action(voll, 154689758)

# list all volume actions
volume_actions(volumes()[[1]])

## End(Not run)
```
words

1000 words to use for seeding random word selection when name not given for a droplet

Description

1000 words to use for seeding random word selection when name not given for a droplet