# Package 'analogsea'

October 12, 2022

**Title** Interface to 'Digital Ocean'

**Description** Provides a set of functions for interacting with the 'Digital Ocean' API <https://www.digitalocean.com/>, including creating images, destroying them, rebooting, getting details on regions, and available images.

**Version** 1.0.6

**License** Apache License (>= 2)

**URL** https://github.com/pachadotdev/analogsea (devel)
https://pacha.dev/analogsea/ (docs)

**BugReports** https://github.com/pachadotdev/analogsea/issues

**Encoding** UTF-8

**Language** en-US

**Imports** stats, utils, httr (>= 1.2.0), jsonlite (>= 1.1), magrittr, yaml

**Suggests** testthat, knitr, ssh (>= 0.6), aws.s3, arrow

**RoxygenNote** 7.1.2

**NeedsCompilation** no

**Author** Scott Chamberlain [aut] (<https://orcid.org/0000-0003-1444-9135>), Hadley Wickham [aut], Winston Chang [aut], Bob Rudis [ctb], Bryce Mecum [ctb] (<https://orcid.org/0000-0002-0381-3766>), Mauricio Vargas [aut, cre] (<https://orcid.org/0000-0003-1017-7574>), RStudio [cph], DigitalOcean [cph]

**Maintainer** Mauricio Vargas <mavargas11@uc.cl>

**Repository** CRAN

**Date/Publication** 2022-05-08 17:30:02 UTC

# R topics documented:

---

`analogsea-package` *R client for Digital Ocean*

---

## Description

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 Digitial 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

Hadley Wickham

Winston Chang

Bob Rudis

Bryce Mecum

---

## 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

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

action_wait(x)
```

### 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. |
| x | Input object |

## 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

```
## 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/docun v2/deprecating-update-tag/ for details.

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

### Description

Debian functions, prefer the Ubuntu equivalents.

- debian_add_swap()
- debian_install_r()
- debian_install_rstudio()
- debian_install_shiny()
- debian_install_opencpu()
- debian_apt_get_update()
- debian_apt_get_install()

## as.certificate *Get list of certificate and their metadata, or a single certificate*

### Description

Get list of certificate and their metadata, or a single certificate

### Usage

```
as.certificate(x)

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

certificate(id, ...)

certificate_create(
  name,
  type,
  private_key = NULL,
  leaf_certificate = NULL,
  certificate_chain = NULL,
  dns_names = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| x | Object to coerce to an certificate |
| page | Page to return. Default: 1. |
| per_page | Number of results per page. Default: 25. |
| ... | Additional arguments passed down to low-level API function (do_*) |
| id | (numeric) certificate id |
| name | (character) a certificate name |
| type | (character) a string representing the type of certificate. The value should be "custom" for a user-uploaded certificate or "lets_encrypt" for one automatically generated with Let's Encrypt. If not provided, "custom" will be assumed by default. |
| private_key | (character) the contents of a PEM-formatted private-key corresponding to the SSL certificate |
| leaf_certificate | (character) the contents of a PEM-formatted public SSL certificate |
| certificate_chain | (character) the full PEM-formatted trust chain between the certificate authority's certificate and your domain's SSL certificate |
| dns_names | (character) a vector of fully qualified domain names (FQDNs) for which the certificate will be issued. The domains must be managed using DigitalOcean's DNS |

## Examples

```
## Not run:
# list certificates
certificates()

# create a certificate (create a fake domain first)
d <- domain_create('tablesandchairsbunnies.stuff', '107.170.220.59')
certificate_create("mycert", "lets_encrypt",
  dns_names = list('tablesandchairsbunnies.stuff'))

## End(Not run)
```

---

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

---

## Description

List, create, update, and delete domain records.

## Usage

```
as.domain_record(x, domain)

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

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

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

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", "wildissue", 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.firewall` *Get list of firewalls and their metadata, or a single firewall*

## Description

Get list of firewalls and their metadata, or a single firewall

## Usage

```
as.firewall(x)

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

firewall(id, ...)

firewall_create(
  name,
  inbound_rules,
  outbound_rules,
  droplet_ids = NULL,
  tags = NULL,
  ...
)

firewall_update(
  name,
  inbound_rules,
  outbound_rules,
  droplet_ids = NULL,
  tags = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `x` | Object to coerce to an firewall. |
| `page` | Page to return. Default: 1. |
| `per_page` | Number of results per page. Default: 25. |
| `...` | Additional arguments passed down to low-level API function (`do_*`) |
| `id` | (numeric) firewall id. |
| `name` | (character) a firewall name |
| `inbound_rules` | (list) inbound rules |
| `outbound_rules` | (list) outbound rules |
| `droplet_ids` | (numeric/integer) droplet ids |
| `tags` | (character) tag strings |

## Examples

```
## Not run:
# list firewalls
firewalls()

# create a firewall
inbound <- list(list(protocol = "tcp", ports = "80",
  sources = list(addresses = "18.0.0.0/8")))
outbound <- list(list(protocol = "tcp", ports = "80",
  destinations = list(addresses = "0.0.0.0/0")))
res <- firewall_create("myfirewall", inbound, outbound)
res

# get a firewall
firewall("d19b900b-b03e-4e5d-aa85-2ff8d2786f28")
as.firewall("d19b900b-b03e-4e5d-aa85-2ff8d2786f28")

## 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

```
as.image(x)

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

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. |

## Examples

```
## 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.project *Get list of projects and their metadata, or a single project*

---

## Description

Get list of projects and their metadata, or a single project

## Usage

```
as.project(x)

projects(page = NULL, per_page = NULL, ...)

project(id = "default", ...)
```

## Arguments

| | |
|---|---|
| x | Object to coerce to a project. |
| page | Page to return. Default: 1. |
| per_page | Number of results per page. Default: 25. |
| ... | Additional arguments passed down to low-level API function (do_*) |
| id | (character) project id, default: "default" |

## Examples

```
## Not run:
projects()
project("f9597f51-6fb0-492c-866d-bc67bff6d409")

## 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

```
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

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

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

# list volume snapshots
snapshots(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(snapshots(), snapshot_delete)

## End(Not run)
```

---

`as.space` *Coerce an object to a* `space`

---

## Description

Coerce an object to a `space`

## Usage

```
as.space(x)
```

## Arguments

| | |
|---|---|
| `x` | Object to coerce to a space |

---

`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

```
as.volume(x)

volumes(...)

volume(volume, ...)

volume_create(
  name,
  size,
  description = NULL,
  region = "nyc1",
  snapshot_id = NULL,
  filesystem_type = NULL,
  filesystem_label = NULL,
  tags = 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. |
| filesystem_type | (character) The name of the filesystem type to be used on the volume. When provided, the volume will automatically be formatted to the specified filesystem type. Currently, the available options are "ext4" and "xfs". Pre-formatted volumes are automatically mounted when attached to Ubuntu, Debian, Fedora, Fedora Atomic, and CentOS Droplets created on or after April 26, 2018. Attaching pre-formatted volumes to other Droplets is not recommended. |

filesystem_label
: (character) The label to be applied to the filesystem. Labels for ext4 type filesystems may contain 16 characters while lables for xfs type filesystems are limited to 12 characters. May only be used in conjunction with filesystem_type.

tags
: (character) tag names to apply to the Volume after it is created. Tag names can either be existing or new tags.

## 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, tags = c('stuff', 'things'))

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

# list snaphots for a volume
volume_snapshots(xx)

# list volumes again
res <- volumes()

# get a single volume
## a whole volume class object
volume(res$testing)
## by id
volume(res[[1]]$id)
## by name
volume(res[[1]]$name)

# delete a volume
## a whole volume class object
volume_delete(res$testing)
## by id
volume_delete(res[[1]]$id)
## by name
volume_delete(res[[1]]$name)

# delete many volumes
lapply(volumes(), volume_delete)

## End(Not run)
```

## certificate_delete *Delete a certificate*

### Description

Delete a certificate

### Usage

```
certificate_delete(id, ...)
```

### Arguments

| | |
|---|---|
| id | A certificate id (not the name) to delete |
| ... | Options passed on to httr::DELETE |

## create_password *Create a password with digits, letters and special characters*

### Description

Create a password with digits, letters and special characters

### Usage

```
create_password(n = 8)
```

### Arguments

| | |
|---|---|
| n | Password length (8-15 characters) |

### Examples

```
create_password(10)
```

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

## Description

**database** get a single database

**databases** list databases

**database_create** create a database

**database_snapshot_create** create a snapshot of a database

**database_snapshots** list snapshots for a database

**database_delete** delete a database

## Usage

```
databases(...)

as.database(x)

databases(...)

database(database, ...)

database_create(
  name,
  size,
  description = NULL,
  region = "nyc1",
  snapshot_id = NULL,
  engine = NULL,
  tags = NULL,
  ...
)

database_snapshot_create(database, name, ...)

database_snapshots(database, ...)

database_delete(database, ...)
```

## Arguments

| | |
|---|---|
| `...` | Additional options passed down to `GET`, `POST`, etc. |
| `x` | Object to coerce to an database |
| `database` | A database, or something that can be coerced to a database by `as.database`. |
| `name` | (character) Name of the new database. required. |

| | |
|---|---|
| size | (integer) The size of the Block Storage database in GiB |
| description | (character) An optional free-form text field to describe a Block Storage database. |
| region | (character) The region where the Block Storage database will be created. When setting a region, the value should be the slug identifier for the region. When you query a Block Storage database, 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 database snapshot from which to create the database. Should not be specified with a region_id. |
| engine | (character) The name of the engine type to be used on the database. When provided, the database will be created with the specified backend type. Currently, the available options are "pg", "mysql", "redis" and "mongodb". |
| tags | (character) tag names to apply to the database after it is created. Tag names can either be existing or new tags. |

## Details

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

## Value

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

## Examples

```
## Not run:
databases()

## End(Not run)
## Not run:
# list databases
databases()

# create a database
vol1 <- database_create('testing', 5)
vol2 <- database_create('foobar', 6, tags = c('stuff', 'things'))

# create snapshot of a database
xx <- database_snapshot_create(vol2, "howdy")

# list snaphots for a database
database_snapshots(xx)

# list databases again
res <- databases()

# get a single database
## a whole database class object
database(res$testing)
## by id
database(res[[1]]$id)
```

```
## by name
database(res[[1]]$name)

# delete a database
## a whole database class object
database_delete(res$testing)
## by id
database_delete(res[[1]]$id)
## by name
database_delete(res[[1]]$name)

# delete many databases
lapply(databases(), database_delete)

## End(Not run)
```

---

debian *Helpers for managing a debian droplets.*

---

## Description

Helpers for managing a debian droplets.

## Usage

```
debian_add_swap(
  droplet,
  user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE
)

debian_install_r(
  droplet,
  user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE,
  rprofile = "options(repos=c('CRAN'='https://cloud.r-project.org/'))"
)

debian_install_rstudio(
  droplet,
  user = "rstudio",
  password = "server",
  version = "0.99.484",
  keyfile = NULL,
```

```
  ssh_passwd = NULL,
  verbose = FALSE
)

debian_install_shiny(
  droplet,
  version = "1.4.0.756",
  user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE,
  rprofile = "options(repos=c('CRAN'='https://cloud.r-project.org/'))"
)

debian_apt_get_update(
  droplet,
  user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE
)

debian_apt_get_install(
  droplet,
  ...,
  user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| droplet | A droplet, or object that can be coerced to a droplet by as.droplet. |
| user | Default username for Rstudio. |
| keyfile | Optional private key file. |
| ssh_passwd | Optional passphrase or callback function for authentication. Refer to the ssh::ssh_connect documentation for more details. |
| verbose | If TRUE, will print command before executing it. |
| rprofile | A character string that will be added to the .Rprofile |
| password | Default password for Rstudio. |
| version | Version of rstudio to install. |
| ... | Arguments to apt-get install. |

## Examples

```
## Not run:
```

```
d <- droplet_create()
d %>% debian_add_swap()
d %>% debian_apt_get_update()

d %>% debian_install_r()
d %>% debian_install_rstudio()

# Install libcurl, then build RCurl from source
d %>% debian_apt_get_install("libcurl4-openssl-dev")
d %>% install_r_package("RCurl")
droplet_delete(d)

## End(Not run)
```

---

`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", "s-1vcpu-2gb"),
  region = getOption("do_region", "sfo3"),
  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-18-04",
  ...
)
```

## 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: s-1vcpu-1gb, the smallest |

region (character) The unique slug identifier for the region that you wish to deploy in. See regions() for a complete list. Default: sfo3

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. Use rstudio-20-04 for a DigitalOcean Marketplace image with R and Tidyverse readily available. Default: ubuntu-18-04-x64

... Additional options passed down to POST

## Value

Two or more droplet objects

## Missing droplet ID

If you get a droplet object back without an IP address, the IP address was not assigned when the payload was returned by DigitalOcean. Simply run d <- droplet(d$id) to update your droplet object and the IP address will populate.

## 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", "s-1vcpu-2gb"),
  region = getOption("do_region", "sfo3"),
  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-20-04",
  keyfile = NULL,
  ...
)

docklet_ps(droplet, all = TRUE, ssh_user = "root")

docklet_images(droplet, all = TRUE, ssh_user = "root")

docklet_pull(
  droplet,
  repo,
  ssh_user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE
)

docklet_run(
  droplet,
  ...,
  rm = FALSE,
  name = NULL,
```

```
  ssh_user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE
)

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",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE
)

docklet_rstudio(
  droplet,
  user,
  password,
  email = "rstudio@example.com",
  img = "rocker/rstudio",
  port = "8787",
  volume = "",
  dir = "",
  browse = TRUE,
  add_users = FALSE,
  ssh_user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE
)

docklet_rstudio_addusers(
  droplet,
  user,
  password,
  img = "rocker/rstudio",
  port = "8787",
  ssh_user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE
```

```
)

docklet_shinyserver(
  droplet,
  img = "rocker/shiny",
  port = "3838",
  volume = "",
  dir = "",
  browse = TRUE,
  ssh_user = "root",
  keyfile = NULL
)

docklet_shinyapp(
  droplet,
  path,
  img = "rocker/shiny",
  port = "80",
  dir = "",
  browse = TRUE,
  ssh_user = "root",
  keyfile = NULL
)
```

## 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: s-1vcpu-2gb |
| region | (character) The unique slug identifier for the region that you wish to deploy in. See regions() for a complete list. Default: sfo3 |
| 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. Use rstudio-20-04 for a DigitalOcean Marketplace image with R and Tidyverse readily available. Default: ubuntu-18-04-x64 |
| keyfile | Optional private key file. |
| ... | 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 |
| ssh_passwd | Optional passphrase or callback function for authentication. Refer to the ssh::ssh_connect documentation for more details. |
| verbose | If TRUE, will print command before executing it. |
| rm | (logical) Automatically remove the container when it exits. Default: FALSE |
| container | (character) Container name, can be partial (though has to be unique) |
| cmd | (character) A docker command (e.g., "run") |
| args | (character) Docker args |
| docker_args | (character) Docker args |
| user | (character) User name. required. |
| password | (character) Password. required. can not be 'rstudio' |
| email | (character) E-mail address. 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.

## Managing Docker containers from R

There's a few things to be note about managing Docker containers from analogsea:

- To see running containers run `docklet_ps(d)`
- To get get logs run `droplet_ssh(d, "docker logs <container ID>")`
- To get a continuous feed of the logs run `droplet_ssh(d, "docker logs -f <container ID>")`
- Do not use `docker exec -ti` as you do not want an interactive session - it will not work from within R. If you log into your DigitalOcean droplet you can do `docker exec -ti`
- To install R package dependencies for a Shiny app, or similar, run `droplet_ssh(d, "docker exec <ID> R -e 'instal` where `d` is your droplet object and `<ID>` is the docker container ID

## Missing droplet ID

If you get a droplet object back without an IP address, the IP address was not assigned when the payload was returned by DigitalOcean. Simply run `d <- droplet(d$id)` to update your droplet object and the IP address will populate.

## See Also

docklets_create

## Examples

```
## Not run:
d <- docklet_create()
d <- droplet(d$id)
d %>% docklet_pull("dockerpinata/sqlite")
d %>% docklet_images()

# sqlite
d %>% docklet_run("dockerpinata/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
d %>% docklet_pull("rocker/r-base")
d %>% docklet_pull("rocker/r-devel")
d %>% docklet_pull("rocker/r-ver:3.2")
d %>% docklet_run("rocker/r-ver:3.2", "R --version", rm = TRUE)
d %>% docklet_run("rocker/r-ver:3.2", "Rscript -e '2 + 3'", rm = TRUE)

# Run a docklet containing rstudio
d %>% docklet_rstudio(user = "foo", password = "bar")

# Delete a droplet
d %>% droplet_delete()

# Add users to an Rstudio instance
## This adds 100 users to the instance, with username/passwords
## following pattern user1/user1 ... through 100
d <- docklet_create()
d <- droplet(d$id)
d %>% docklet_rstudio(user = "foo", password = "bar") %>%
 docklet_rstudio_addusers(user = "foo", password = "bar")

# Spin up a Shiny server (opens in default browser)
(d <- docklet_create())
d %>% docklet_shinyserver()
docklet_create() %>% docklet_shinyserver()

# Spin up a Shiny server with an app (opens in default browser)
d <- docklet_create(); d <- droplet(d$id)
path <- system.file("examples", "widgets", package = "analogsea")
d %>% docklet_shinyapp(path)
## uploading more apps - use droplet_upload, then navigate in browser
### if you try to use docklet_shinyapp again on the same droplet, it will error
path2 <- system.file("examples", "mpg", package = "analogsea")
d %>% droplet_upload(path2, "/srv/shinyapps") # then go to browser

## End(Not run)
```

---

`domains` *Get information on a single domain or all your domains.*

---

## Description

Get information on a single domain or all your domains.

## Usage

```
domains(...)

as.domain(x)

domain(x, ...)
```

### Arguments

| | |
|---|---|
| ... | Further args passed on the curl call to the web. |
| x | (character) Required. Domain name |

### Examples

```
## Not run:
domains()

## End(Not run)
```

---

## domain_create *Create/delete domains.*

---

### Description

Create/delete domains.

### Usage

```
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:
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

```
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 envar.
- Interatively 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

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

## 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

```
## 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.

### 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

```
## 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 *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

```
## 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

```
droplets_create(
  names = NULL,
  size = getOption("do_size", "s-1vcpu-1gb"),
  image = getOption("do_image", "ubuntu-18-04-x64"),
  region = getOption("do_region", "sfo3"),
  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

| | |
|---|---|
| `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: s-1vcpu-1gb, 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. Use rstudio-20-04 for a DigitalOcean Marketplace image with R and Tidyverse readily available. Default: ubuntu-18-04-x64

`region` (character) The unique slug identifier for the region that you wish to deploy in. See `regions()` for a complete list. Default: sfo3

`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()

`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 `cloud_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.

`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

### Missing droplet ID

If you get a droplet object back without an IP address, the IP address was not assigned when the payload was returned by DigitalOcean. Simply run `d <- droplet(d$id)` to update your droplet object and the IP address will populate.

### Examples

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

# give names
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)
```

---

## droplet_action *Perform various actions on a droplet.*

---

### 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.

**enable_backups** Enables backups for a droplet.

**power_on** Turn on a droplet that's turned off.

## Examples

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

d <- droplet_create()
d %>% summary
d %>% droplet_enable_backups()
d %>% summary

## End(Not run)
```

## droplet_actions *Retrieve a droplet action or list all actions associatd with a droplet.*

### Description

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

### Usage

```
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

```
## 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 your'e up and running.

### Usage

```
droplet_create(
  name = random_name(),
  size = getOption("do_size", "s-1vcpu-1gb"),
  image = getOption("do_image", "ubuntu-18-04-x64"),
  region = getOption("do_region", "sfo3"),
  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: s-1vcpu-1gb, 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. Use rstudio-20-04 for a DigitalOcean Marketplace image with R and Tidyverse readily available. Default: ubuntu-18-04-x64 |
| region | (character) The unique slug identifier for the region that you wish to deploy in. See regions() for a complete list. Default: sfo3 |
| 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() |
| 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 cloud_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. |
| 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

A droplet object

## Missing droplet ID

If you get a droplet object back without an IP address, the IP address was not assigned when the payload was returned by DigitalOcean. Simply run `d <- droplet(d$id)` to update your droplet object and the IP address will populate.

## 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 = 'sfo3')

# 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

```
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

```
## 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

```
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

```
## 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

```
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

```
## Not run:
d <- droplet_create() %>%
  ubuntu_add_swap() %>%
  droplet_ssh("apt-get update") %>%
  ubuntu_install_r()

results <- d %>% 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

```
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

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

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

## End(Not run)
```

---

`droplet_functions` *Functions for DigitalOcean (DO) droplets*

---

## Description

There's a lot of functions for working with droplets. Here's a breakdown of what they all do.

## Documentation

- DigitalOcean docs overview: https://developers.digitalocean.com/documentation/
- DigitalOcean API docs: https://developers.digitalocean.com/documentation/v2/

## Functions

The main functions for creating/deleting droplets:

- `droplet()`: get a droplet object from a droplet ID
- `droplet_create()`: create a droplet
- `droplets_create()`: create two or more droplets
- `droplet_delete()`: delete a droplet
- `droplets()`: get your droplets
- `as.droplet()`: coerce various things to droplet objects

Modify a droplet:

- `droplet_resize()`: resize a droplet to a different size
- `droplet_rebuild()`: reinstall a droplet with a different image
- `droplet_rename()`: rename a droplet
- `droplet_change_kernel()`: change droplet to a new kernel

Take and restore snapshots:

- `droplet_snapshot()`: make a snapshot of a droplet
- `droplet_snapshots_list()`: list snapshots on a droplet
- `droplet_backups_list()`: list droplet backups

- droplet_restore(): Restore a droplet with a previous image or snapshot

ssh interactions with droplets:

- droplet_ssh(): Remotely execute code on your droplet via ssh
- droplet_upload(): Upload files to your droplet via ssh
- droplet_download(): Download files from your droplet via ssh

Perform various actions on droplets:

- droplet_actions(): retrieve a droplet action or list all actions associated with a droplet
- droplet_disable_backups(): Disables backups for a droplet
- droplet_do_actions(): Perform actions on one or more droplets associated with a tag
- droplet_enable_backups(): Enables backups for a droplet
- droplet_enable_ipv6(): Enable IPv6 networking on an existing droplet (within a region that has IPv6 available)
- droplet_enable_private_networking(): Enable private networking on an existing droplet (within a region that has private networking available)
- droplet_execute(): Execute R code on a droplet
- droplet_kernels_list(): List all available kernels for a droplet
- droplet_neighbors(): List a droplet's neighbors on the same physical server
- droplet_power_cycle(): power cycle a droplet. will turn off the droplet and then turn it back on
- droplet_power_off(): Shutdown a running droplet. The droplet will remain in your account and you will continue to be charged for it
- droplet_power_on(): Turn on a droplet that's turned off
- droplet_reboot(): reboot a droplet. This is the preferred method to use if a server is not responding
- droplet_reset_password(): reset the root password for a droplet
- droplet_reuse(): Reuse a droplet or image by name, creating a new droplet
- droplet_shutdown(): Shutdown a running droplet. The droplet will remain in your account and you will continue to be charged for it.
- droplet_upgrade(): Migrate a droplet - NOT SURE IF THIS STILL WORKS OR NOT
- droplet_upgrades_list(): List all droplets that are scheduled to be upgraded
- droplet_wait(): Wait for a droplet to be ready. mostly used internally
- droplets_cost(): Calculate cost across droplets

Freeze/thaw droplets:

- droplet_freeze(): power off a droplet, snapshots to create an image, and deletes the droplet
- droplet_thaw(): takes an image and turns it into a running droplet

### Working with Docker

We named a DO droplet with the Docker application installed a "docklet" for convienence

The main two functions for creating docklets:

- `docklet_create()`: create a docklet (a droplet using the docker image)
- `docklets_create()`: create many docklets

Running docker commands on your docklet:

- `docklet_images()`: list docker images on your docklet
- `docklet_ps()`: list running docker containers
- `docklet_pull()`: pull a docker image to your docklet
- `docklet_rm()`: remove a docker image from your docklet
- `docklet_run()`: run a docker command on your docklet
- `docklet_stop()`: stop a running docker container
- `docklet_docker()`: low level fxn for running docker commands on your, not realy intended for public use

Install RStudio things:

- `docklet_rstudio()`: install RStudio on your docklet using Rocker images (https://hub.docker.com/u/rocker)
- `docklet_rstudio_addusers()`: add users to an RStudio docker image
- `docklet_shinyserver()`: install Shiny server on your docklet using Rocker images (https://hub.docker.com/u/rocker)
- `docklet_shinyapp()`: install a Shiny app on your Shiny server docker container

---

## `droplet_ip` *Get droplet's IP address*

---

### Description

Get droplet's IP address

### Usage

```
droplet_ip(droplet)
```

### Arguments

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

### Examples

```
## Not run:
# Obtain the droplet's IP as a string
my_droplet <- droplet_create("demo", region = "sfo3")
droplet_ip(my_droplet)

## 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

```
## 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

```
## 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

```
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

```
## 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.

**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, wait = TRUE, ...)

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 |

| | |
|---|---|
| wait | If TRUE (default), wait until the snapshot has been completed and is ready for use. If set to FALSE we return a droplet object right away after droplet snapshot request has been sent. |
| ... | Additional options passed down to POST |
| image | (optional) The image ID of the backup image that you would like to restore. |

## Examples

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

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

# To delete safely
d %>%
  droplet_snapshot() %>%
  droplet_delete() %>%
  action_wait()

## End(Not run)
```

---

# 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",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE
)

droplet_upload(
  droplet,
  local,
  remote,
```

```
  user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE
)

droplet_download(
  droplet,
  remote,
  local,
  user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  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". |
| keyfile | Optional private key file. |
| ssh_passwd | Optional passphrase or callback function for authentication. Refer to the ssh::ssh_connect documentation for more details. |
| 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.

With the chang to package ssh, we create ssh session objects (C pointers) internally, and cache them, then look them up in the cache based on combination of user and IP address. That is, there's separate sessions for each user for the same IP address.

ssh sessions are cleaned up at the end of your R session.

## Value

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

## Examples

```
## 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, ".")
d %>% droplet_ssh("ls")

tmp2 <- tempdir()
d %>% droplet_download(basename(tmp), tmp2)
mtcars2 <- readRDS(file.path(tmp2, basename(tmp)))

stopifnot(all.equal(mtcars, mtcars2))


## another upload/download example
tmp <- tempfile(fileext = ".txt")
writeLines("foo bar", tmp)
readLines(tmp)
d %>% droplet_upload(tmp, ".")
d %>% droplet_ssh("ls")

tmp2 <- tempdir()
unlink(tmp)
d %>% droplet_download(basename(tmp), tmp2)
readLines(file.path(tmp2, basename(tmp)))

## 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

```
## 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

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

## End(Not run)
```

## firewall_add_droplets *Add/remove droplets to a firewall*

### Description

Add/remove droplets to a firewall

### Usage

```
firewall_add_droplets(id, droplet_ids, ...)

firewall_remove_droplets(id, droplet_ids, ...)
```

### Arguments

| | |
|---|---|
| id | (character) A firewall id (not the name) to delete |
| droplet_ids | (integer/numeric) a vector of droplet ids |
| ... | Options passed on to httr::POST or httr::DELETE |

### Examples

```
## Not run:
drops <- droplets_create()
drop_ids <- vapply(drops, "[[", numeric(1), "id")
inbound <- list(list(protocol = "tcp", ports = "80",
  sources = list(addresses = "18.0.0.0/8")))
outbound <- list(list(protocol = "tcp", ports = "80",
  destinations = list(addresses = "0.0.0.0/0")))
res <- firewall_create("myfirewall", inbound, outbound)
firewall_add_droplets(id = res$id, droplet_ids = drop_ids)
firewalls()[[1]]$droplet_ids
firewall_remove_droplets(id = res$id, droplet_ids = drop_ids)

## End(Not run)
```

## firewall_add_tags *Add/remove tags to a firewall*

### Description

Add/remove tags to a firewall

### Usage

```
firewall_add_tags(id, tags, ...)

firewall_remove_tags(id, tags, ...)
```

### Arguments

| | |
|---|---|
| id | (character) A firewall id (not the name) to delete |
| tags | (character) tag strings |
| ... | Options passed on to httr::POST or httr::DELETE |

### Examples

```
## Not run:
drops <- droplets_create()
drop_ids <- vapply(drops, "[[", numeric(1), "id")
inbound <- list(list(protocol = "tcp", ports = "80",
  sources = list(addresses = "18.0.0.0/8")))
outbound <- list(list(protocol = "tcp", ports = "80",
  destinations = list(addresses = "0.0.0.0/0")))
res <- firewall_create("myfirewall", inbound, outbound)

tag_create(name = "foobar")
tags()
firewall_add_tags(id = res$id, tags = "foobar")
firewalls()[[1]]$tags
firewall_remove_tags(id = res$id, tags = "foobar")

## End(Not run)
```

---

## firewall_delete *Delete a firewall*

---

### Description

Delete a firewall

### Usage

```
firewall_delete(id, ...)
```

### Arguments

| | |
|---|---|
| id | A firewall id (not the name) to delete |
| ... | Options passed on to httr::DELETE |

### Examples

```
## Not run:
firewall_delete(id="d19b900b-b03e-4e5d-aa85-2ff8d2786f28")

## 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

```
## 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. |

## Examples

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

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

## End(Not run)
```

## 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

| | |
|---|---|
| name | (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_*) |
| page | Page to return. Default: 1. |
| 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

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

## End(Not run)
```

## neighbors *List neighbors*

### Description

List neighbors

### Usage

```
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

```
## 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 - the data is licensed CC0.

`project_create` *Create a project*

## Description

Create a project

## Usage

```
project_create(name, purpose, description = NULL, environment = NULL, ...)
```

## Arguments

| | |
|---|---|
| `name` | (character) Name of the project. required |
| `purpose` | (character) The purpose of the project. The maximum length is 255 characters. For examples of valid purposes, see the "Purposes" section. required |
| `description` | (character) The description of the project. The maximum length is 255 characters. optional |
| `environment` | (character) The environment of the project's resources. optional |
| `...` | Additional options passed down to `POST` |

## Value

A project object

## Purposes

The purpose attribute can have one of the following values:

- Just trying out DigitalOcean
- Class project / Educational purposes
- Website or blog
- Web Application
- Service or API
- Mobile Application
- Machine learning / AI / Data processing
- IoT
- Operational / Developer tooling

If specify another value for purpose, for example "your custom purpose", your purpose will be stored as Other: your custom purpose

### Environments

The environment attribute must have one of the following values:

- Development
- Staging
- Production

If another value is specified, a 400 Bad Request is returned.

### Examples

```
## Not run:
project_create(name = "venus", purpose = "Web Application")

## End(Not run)
```

---

## project_delete *Delete a project*

---

### Description

Delete a project

### Usage

```
project_delete(project, ...)
```

### Arguments

| | |
|---|---|
| project | A project to modify. |
| ... | Options passed on to httr::GET. Must be named, see examples. |

### Examples

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

## End(Not run)
```

`project_patch` *Update certain aspects of a project*

## Description

Update certain aspects of a project

## Usage

```
project_patch(
  id,
  name = NULL,
  purpose = NULL,
  description = NULL,
  is_default = FALSE,
  environment = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `id` | project id. to update the default project use "default". required |
| `name` | (character) Name of the project. required |
| `purpose` | (character) The purpose of the project. The maximum length is 255 characters. For examples of valid purposes, see the "Purposes" section. required |
| `description` | (character) The description of the project. The maximum length is 255 characters. optional |
| `is_default` | (logical) If `TRUE`, all resources will be added to this project if no project is specified. default: `FALSE` |
| `environment` | (character) The environment of the project's resources. optional |
| `...` | Additional options passed down to `POST` |

`project_update` *Update all aspects of a project*

## Description

Update all aspects of a project

## Usage

```
project_update(
  id,
  name,
  purpose,
  description,
  is_default = FALSE,
  environment = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| id | project id. to update the default project use "default". required |
| name | (character) Name of the project. required |
| purpose | (character) The purpose of the project. The maximum length is 255 characters. For examples of valid purposes, see the "Purposes" section. required |
| description | (character) The description of the project. The maximum length is 255 characters. optional |
| is_default | (logical) If TRUE, all resources will be added to this project if no project is specified. default: FALSE |
| environment | (character) The environment of the project's resources. optional |
| ... | Additional options passed down to POST |

---

# 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

```
## 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

```
## 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

```
sizes(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. |

### Value

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

### Examples

```
## Not run:
sizes()

## End(Not run)
```

## spaces *List all Spaces.*

### Description

List all Spaces.

### Usage

```
spaces(spaces_region = NULL, spaces_key = NULL, spaces_secret = NULL, ...)
```

### Arguments

| | |
|---|---|
| spaces_region | (character) String containing a spaces region. If missing, defaults to value stored in an environment variable DO_SPACES_REGION. |
| spaces_key | (character) String containing a spaces access key. If missing, defaults to value stored in an environment variable DO_SPACES_ACCESS_KEY. |
| spaces_secret | (character) String containing the secret associated with the spaces key. If missing, defaults to value stored in an environment variable DO_SPACES_SECRET_KEY. |
| ... | Additional arguments to spaces_GET |

### Value

(list) A list of Spaces. Can be empty.

### References

https://developers.digitalocean.com/documentation/spaces/#get-object

### Examples

```
## Not run:
# List all of your Spaces
spaces()

## End(Not run)
```

---

## spaces_GET *Internal helper method to get information about a Space*

---

### Description

Internal helper method to get information about a Space

### Usage

```
spaces_GET(spaces_region = NULL, spaces_key = NULL, spaces_secret = NULL, ...)
```

### Arguments

| | |
|---|---|
| spaces_region | (character) String containing a spaces region. If missing, defaults to value stored in an environment variable DO_SPACES_REGION. |
| spaces_key | (character) String containing a spaces access key. If missing, defaults to value stored in an environment variable DO_SPACES_ACCESS_KEY. |
| spaces_secret | (character) String containing the secret associated with the spaces key. If missing, defaults to value stored in an environment variable DO_SPACES_SECRET_KEY. |
| ... | Additional arguments to aws.s3::s3HTTP |

### Value

The raw S3 response, or throws an error

spaces_info *DigitalOcean Spaces*

## Description

DigitalOcean provides support for storing files (Objects) in Spaces. This is useful for storing related files for fast access, sharing, etc. See https://developers.digitalocean.com/documentation/spaces/ for more information. The `aws.s3` package is required to use `analogsea`'s Spaces functionality so be sure to install it with `install.packages("aws.s3")` prior to continuing.

## Arguments

| | |
|---|---|
| space | A Space, or the name of the Space as a string. |
| object | (character) The name of the Object |

## Details

In order to get started using the Spaces API, you'll need to generate a new "Spaces access key" in the API section of your DigitalOcean control panel and set the key and its secret as environmental variables via `Sys.setenv`. Set the access key to `DO_SPACES_ACCESS_KEY` and its secret to `DO_SPACES_SECRET_KEY`. After that, set your region to `DO_SPACES_REGION` (e.g., nyc3). Alternatively, you can pass this information as arguments to whichever Spaces API functions you're using.

## Examples

```
## Not run:
# List Spaces
spaces()

# Obtain Spaces as a list of Space objects
res <- spaces()

# Print Space summary using a Space object
summary(res[["my_space_name"]])

# Create a new space
space_create("new_space_name")

## End(Not run)
```

space_create *Create a new Space*

## Description

Create a new Space

## Usage

```
space_create(
  name,
  spaces_region = NULL,
  spaces_key = NULL,
  spaces_secret = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| name | (character) The name of the new Space |
| spaces_region | (character) String containing a spaces region. If missing, defaults to value stored in an environment variable DO_SPACES_REGION. |
| spaces_key | (character) String containing a spaces access key. If missing, defaults to value stored in an environment variable DO_SPACES_ACCESS_KEY. |
| spaces_secret | (character) String containing the secret associated with the spaces key. If missing, defaults to value stored in an environment variable DO_SPACES_SECRET_KEY. |
| ... | Additional arguments to aws.s3::put_bucket |

## Value

(character) The name of the created Space.

## Examples

```
## Not run:
# Create a new Space
# (Names must be unique within region)
space_create("new_space_name")

## End(Not run)
```

space_delete *Delete an existing Space*

## Description

Delete an existing Space

## Usage

```
space_delete(
  name,
  spaces_region = NULL,
  spaces_key = NULL,
  spaces_secret = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| name | (character) The name of the existing Space |
| spaces_region | (character) String containing a spaces region. If missing, defaults to value stored in an environment variable DO_SPACES_REGION. |
| spaces_key | (character) String containing a spaces access key. If missing, defaults to value stored in an environment variable DO_SPACES_ACCESS_KEY. |
| spaces_secret | (character) String containing the secret associated with the spaces key. If missing, defaults to value stored in an environment variable DO_SPACES_SECRET_KEY. |
| ... | Additional arguments to aws.s3::delete_bucket |

## Value

(character) The name of the deleted Space.

## Examples

```
## Not run:
# Delete an existing Space
# (Check names within region)
space_delete("new_space_name")

## End(Not run)
```

space_download *Upload a directory to an existing Space*

## Description

Upload a directory to an existing Space

## Usage

```
space_download(
  name,
  local = NULL,
  remote = NULL,
  spaces_region = NULL,
  spaces_key = NULL,
  spaces_secret = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| name | (character) The name of the existing Space |
| local | (character) The name of the local directory |
| remote | (character) The name of the remote directory |
| spaces_region | (character) String containing a spaces region. If missing, defaults to value stored in an environment variable DO_SPACES_REGION. |
| spaces_key | (character) String containing a spaces access key. If missing, defaults to value stored in an environment variable DO_SPACES_ACCESS_KEY. |
| spaces_secret | (character) String containing the secret associated with the spaces key. If missing, defaults to value stored in an environment variable DO_SPACES_SECRET_KEY. |
| ... | Additional arguments to arrow::copy_files |

## Value

(character) Success/error message.

## Examples

```
## Not run:
# Upload to an existing Space
# (Check names within region)
space_download("my_space", "my_subdir", "my_subdir", "nyc3",
 spaces_key = Sys.getenv("SPACES_KEY"),
 spaces_secret = Sys.getenv("SPACES_SECRET"))

## End(Not run)
```

space_upload *Upload a directory to an existing Space*

## Description

Upload a directory to an existing Space

## Usage

```
space_upload(
  name,
  local = NULL,
  remote = NULL,
  spaces_region = NULL,
  spaces_key = NULL,
  spaces_secret = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| name | (character) The name of the existing Space |
| local | (character) The name of the local directory |
| remote | (character) The name of the remote directory |
| spaces_region | (character) String containing a spaces region. If missing, defaults to value stored in an environment variable DO_SPACES_REGION. |
| spaces_key | (character) String containing a spaces access key. If missing, defaults to value stored in an environment variable DO_SPACES_ACCESS_KEY. |
| spaces_secret | (character) String containing the secret associated with the spaces key. If missing, defaults to value stored in an environment variable DO_SPACES_SECRET_KEY. |
| ... | Additional arguments to arrow::copy_files |

## Value

(character) Success/error message.

## Examples

```
## Not run:
# Upload to an existing Space
# (Check names within region)
space_upload("my_space", "my_subdir", "my_subdir", "nyc3",
 spaces_key = Sys.getenv("SPACES_KEY"),
 spaces_secret = Sys.getenv("SPACES_SECRET"))

## End(Not run)
```

## standardise_keys *Standardise specification of ssh keys.*

### Description

Standardise specification of ssh keys.

### Usage

```
standardise_keys(ssh_keys = 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

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

## End(Not run)
```

## tags *List tags*

### Description

List tags

### Usage

```
tags(...)

tag(name, ...)

as.tag(x)
```

### Arguments

| | |
|---|---|
| ... | Additional options passed down to GET |
| name | (character) Name of the tag |
| x | Object to coerce to a tag. |

### Details

tags gets all your tag, tag gets a tag by name

### Value

Many tag objects in a list

### Examples

```
## Not run:
# get all your tags
tags()

# get a tag by name
tag("stuffthings")
tag("helloworld")

## End(Not run)
## Not run:
tag_create("pluto")
as.tag('pluto')
as.tag(tag_create("howdyhoneighbor"))

## End(Not run)
```

---

## tag_create *Create a tag*

---

### 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

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

## End(Not run)
```

---

`tag_delete` *Delete a tag*

---

## Description

Delete a tag

## Usage

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

## Arguments

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

## Value

nothing, if successful

## Examples

```
## 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

```
## 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)
```

ubuntu *Helpers for managing a ubuntu droplets.*

## Description

Helpers for managing a ubuntu droplets.

## Usage

```
ubuntu_add_swap(
  droplet,
  user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE
)

ubuntu_install_r(
  droplet,
  user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE,
  rprofile = "options(repos=c('CRAN'='https://cloud.r-project.org/'))"
)

ubuntu_install_rstudio(
  droplet,
  user = "rstudio",
  password = "server",
  version = "0.99.484",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE
)

ubuntu_install_shiny(
  droplet,
  version = "1.4.0.756",
  user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE,
  rprofile = "options(repos=c('CRAN'='https://cloud.r-project.org/'))"
)

ubuntu_apt_get_cran(
```

```
  droplet,
  user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE
)

ubuntu_apt_get_update(
  droplet,
  user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE
)

ubuntu_apt_get_install(
  droplet,
  ...,
  user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE
)

install_r_package(
  droplet,
  package,
  repo = "https://cloud.r-project.org/",
  user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE
)

install_github_r_package(
  droplet,
  package,
  repo = "https://cloud.r-project.org/",
  user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE
)

ubuntu_create_user(
  droplet,
  user,
  password,
```

```
  ssh_user = "root",
  keyfile = NULL,
  ssh_passwd = NULL,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| droplet | A droplet, or object that can be coerced to a droplet by as.droplet. |
| user | Username for non-root account. |
| keyfile | Optional private key file. |
| ssh_passwd | Optional passphrase or callback function for authentication. Refer to the ssh::ssh_connect documentation for more details. |
| verbose | If TRUE, will print command before executing it. |
| rprofile | A character string that will be added to the .Rprofile |
| password | Password for non-root account. |
| version | Version of rstudio to install. |
| ... | Arguments to apt-get install. |
| package | Name of R package to install. |
| repo | CRAN mirror to use. |
| ssh_user | (character) User account for ssh commands against droplet. |

## Examples

```
## Not run:
d <- droplet_create()
d %>% ubuntu_add_swap()
d %>% ubuntu_apt_get_update()

d %>% ubuntu_install_r()
d %>% ubuntu_install_rstudio()

# Install libcurl, then build RCurl from source
d %>% ubuntu_apt_get_install("libcurl4-openssl-dev")
d %>% install_r_package("RCurl")
droplet_delete(d)

## 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, ...)
```

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

```
## Not run:
# resize a volume
## create a volume
(vol1 <- volume_create('foobar', 5))
## resize it
```

```
volume_resize(vol1, 6)
volume(vol1)

# attach a volume to a droplet
## create a droplet
(d <- droplet_create(region = "nyc1"))
## attach volume to droplet
volume_attach(vol1, d)
## refresh droplet info, see volumes slot
droplet(d$id)

# detach a volume from a droplet
(act <- volume_detach(vol1, d))
## refresh droplet info, see volumes slot
droplet(d$id)

# list an action
volume_action(vol1, 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

# Index