» digitalocean_account

Get information on your DigitalOcean account.

» Example Usage

Get the account:

```
data "digitalocean_account" "example" {
}
```

» Attributes Reference

The following attributes are exported:

- `droplet_limit`: The total number of droplets current user or team may have active at one time.
- `floating_ip_limit`: The total number of floating IPs the current user or team may have.
- `email`: The email address used by the current user to register for DigitalOcean.
- `uuid`: The unique universal identifier for the current user.
- `email_verified`: If true, the user has verified their account via email. False otherwise.
- `status`: This value is one of ”active”, ”warning” or ”locked”.
- `status_message`: A human-readable message giving more details about the status of the account.

» digitalocean_certificate

Get information on a certificate. This data source provides the name, type, state, domains, expiry date, and the sha1 fingerprint as configured on your DigitalOcean account. This is useful if the certificate in question is not managed by Terraform or you need to utilize any of the certificates data.

An error is triggered if the provided certificate name does not exist.

» Example Usage

Get the certificate:

```
data "digitalocean_certificate" "example" {
  name = "example"
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of certificate.

» Attributes Reference

The following attributes are exported:

- `id`: The ID of the certificate.
- `type`: The type of the certificate.
- `state`: the current state of the certificate.
- `domains`: Domains for which the certificate was issued.
- `not_after`: The expiration date and time of the certificate.
- `sha1_fingerprint`: The SHA1 fingerprint of the certificate.

» digitalocean_database_cluster

Provides information on a DigitalOcean database cluster resource.

» Example Usage

```
# Create a new database cluster
data "digitalocean_database_cluster" "example" {
  name = "example-cluster"
}

output "database_output" {
  value = data.digitalocean_database_cluster.example.uri
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the database cluster.

» Attributes Reference

The following attributes are exported:

- `id` - The ID of the database cluster.
- `urn` - The uniform resource name of the database cluster.
- `engine` - Database engine used by the cluster (ex. `pg` for PostreSQL).
- `version` - Engine version used by the cluster (ex. `11` for PostgreSQL 11).
- `size` - Database droplet size associated with the cluster (ex. `db-s-1vcpu-1gb`).
- `region` - DigitalOcean region where the cluster will reside.
- `node_count` - Number of nodes that will be included in the cluster.
- `maintenance_window` - Defines when the automatic maintenance should be performed for the database cluster.
- `host` - Database cluster's hostname.
- `private_host` - Same as `host`, but only accessible from resources within the account and in the same region.
- `port` - Network port that the database cluster is listening on.
- `uri` - The full URI for connecting to the database cluster.
- `private_uri` - Same as `uri`, but only accessible from resources within the account and in the same region.
- `database` - Name of the cluster's default database.
- `user` - Username for the cluster's default user.
- `password` - Password for the cluster's default user.

`maintenance_window` supports the following:

- `day` - The day of the week on which to apply maintenance updates.
- `hour` - The hour in UTC at which maintenance updates will be applied in 24 hour format.

» digitalocean_domain

Get information on a domain. This data source provides the name, TTL, and zone file as configured on your DigitalOcean account. This is useful if the domain name in question is not managed by Terraform or you need to utilize TTL or zone file data.

An error is triggered if the provided domain name is not managed with your DigitalOcean account.

» Example Usage

Get the zone file for a domain:

```
data "digitalocean_domain" "example" {
  name = "example.com"
}

output "domain_output" {
  value = data.digitalocean_domain.example.zone_file
}
```

```
  $ terraform apply

data.digitalocean_domain.example: Refreshing state...

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

domain_output = $ORIGIN example.com.
$TTL 1800
example.com. IN SOA ns1.digitalocean.com. hostmaster.example.com. 1516944700 10800 3600 6048
example.com. 1800 IN NS ns1.digitalocean.com.
example.com. 1800 IN NS ns2.digitalocean.com.
example.com. 1800 IN NS ns3.digitalocean.com.
www.example.com. 3600 IN A 176.107.155.137
db.example.com. 3600 IN A 179.189.166.115
jira.example.com. 3600 IN A 207.189.228.15
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the domain.

» Attributes Reference

The following attributes are exported:

- `ttl`: The TTL of the domain.
- `urn` - The uniform resource name of the domain

- `zone_file`: The zone file of the domain.

» digitalocean_droplet

Get information on a Droplet for use in other resources. This data source provides all of the Droplet's properties as configured on your DigitalOcean account. This is useful if the Droplet in question is not managed by Terraform or you need to utilize any of the Droplet's data.

Note: This data source returns a single Droplet. When specifying a `tag`, an error is triggered if more than one Droplet is found.

» Example Usage

Get the Droplet by name:

```
data "digitalocean_droplet" "example" {
  name = "web"
}

output "droplet_output" {
  value = data.digitalocean_droplet.example.ipv4_address
}
```

Get the Droplet by tag:

```
data "digitalocean_droplet" "example" {
  tag = "web"
}
```

Get the Droplet by ID:

```
data "digitalocean_droplet" "example" {
  id = digitalocean_kubernetes_cluster.example.node_pool[0].nodes[0].droplet_id
}
```

» Argument Reference

One of following the arguments must be provided:

- `id` - (Optional) The ID of the Droplet
- `name` - (Optional) The name of the Droplet.
- `tag` - (Optional) A tag applied to the Droplet.

» Attributes Reference

The following attributes are exported:

- `id`: The ID of the Droplet.
- `urn` - The uniform resource name of the Droplet
- `region` - The region the Droplet is running in.
- `image` - The Droplet image ID or slug.
- `size` - The unique slug that indentifies the type of Droplet.
- `disk` - The size of the Droplets disk in GB.
- `vcpus` - The number of the Droplets virtual CPUs.
- `memory` - The amount of the Droplets memory in MB.
- `price_hourly` - Droplet hourly price.
- `price_monthly` - Droplet monthly price.
- `status` - The status of the Droplet.
- `locked` - Whether the Droplet is locked.
- `ipv6_address` - The Droplets public IPv6 address
- `ipv6_address_private` - The Droplets private IPv6 address
- `ipv4_address` - The Droplets public IPv4 address
- `ipv4_address_private` - The Droplets private IPv4 address
- `backups` - Whether backups are enabled.
- `ipv6` - Whether IPv6 is enabled.
- `private_networking` - Whether private networks are enabled.
- `monitoring` - Whether monitoring agent is installed.
- `volume_ids` - List of the IDs of each volumes attached to the Droplet.
- `tags` - A list of the tags associated to the Droplet.

» digitalocean_droplet_snapshot

Droplet snapshots are saved instances of a Droplet. Use this data source to retrieve the ID of a DigitalOcean Droplet snapshot for use in other resources.

» Example Usage

Get the Droplet snapshot:

```
data "digitalocean_droplet_snapshot" "web-snapshot" {
  name_regex  = "^web"
  region      = "nyc3"
  most_recent = true
}
```

» Argument Reference

- `name` - (Optional) The name of the Droplet snapshot.
- `name_regex` - (Optional) A regex string to apply to the Droplet snapshot list returned by DigitalOcean. This allows more advanced filtering not supported from the DigitalOcean API. This filtering is done locally on what DigitalOcean returns.
- `region` - (Optional) A "slug" representing a DigitalOcean region (e.g. `nyc1`). If set, only Droplet snapshots available in the region will be returned.
- `most_recent` - (Optional) If more than one result is returned, use the most recent Droplet snapshot.

NOTE: If more or less than a single match is returned by the search, Terraform will fail. Ensure that your search is specific enough to return a single Droplet snapshot ID only, or use `most_recent` to choose the most recent one.

» Attributes Reference

The following attributes are exported:

- `id` The ID of the Droplet snapshot.
- `created_at` - The date and time the Droplet snapshot was created.
- `min_disk_size` - The minimum size in gigabytes required for a Droplet to be created based on this Droplet snapshot.
- `regions` - A list of DigitalOcean region "slugs" indicating where the Droplet snapshot is available.
- `droplet_id` - The ID of the Droplet from which the Droplet snapshot originated.
- `size` - The billable size of the Droplet snapshot in gigabytes.

» digitalocean_floating_ip

Get information on a floating ip. This data source provides the region and Droplet id as configured on your DigitalOcean account. This is useful if the floating IP in question is not managed by Terraform or you need to find the Droplet the IP is attached to.

An error is triggered if the provided floating IP does not exist.

» Example Usage

Get the floating IP:

```
variable "public_ip" {}

data "digitalocean_floating_ip" "example" {
  ip_address = var.public_ip
}

output "fip_output" {
  value = data.digitalocean_floating_ip.example.droplet_id
}
```

» Argument Reference

The following arguments are supported:

- `ip_address` - (Required) The allocated IP address of the specific floating IP to retrieve.

» Attributes Reference

The following attributes are exported:

- `region`: The region that the floating IP is reserved to.
- `urn`: The uniform resource name of the floating IP.
- `droplet_id`: The Droplet id that the floating IP has been assigned to.

» digitalocean_image

Get information on an image for use in other resources (e.g. creating a Droplet based on snapshot). This data source provides all of the image properties as configured on your DigitalOcean account. This is useful if the image in question is not managed by Terraform or you need to utilize any of the image's data.

An error is triggered if zero or more than one result is returned by the query.

» Example Usage

Get the data about a snapshot:

```
data "digitalocean_image" "example1" {
  name = "example-1.0.0"
}
```

Reuse the data about a snapshot to create a Droplet:

```
data "digitalocean_image" "example" {
  name = "example-1.0.0"
}

resource "digitalocean_droplet" "example" {
  image  = data.digitalocean_image.example.id
  name   = "example-1"
  region = "nyc2"
  size   = "s-1vcpu-1gb"
}
```

Get the data about an official image:

```
data "digitalocean_image" "example2" {
  slug = "ubuntu-18-04-x64"
}
```

» Argument Reference

One of the following arguments must be provided:

- `id` - The id of the image
- `name` - The name of the image.
- `slug` - The slug of the official image.

If `name` is specified, you may also specify:

- `source` - (Optional) Restrict the search to one of the following categories of images:
  - `all` - All images (whether public or private)
  - `applications` - One-click applications
  - `distributions` - Distributions
  - `user` - (Default) User (private) images

» Attributes Reference

The following attributes are exported:

- `slug`: Unique text identifier of the image.
- `id`: The ID of the image.
- `name`: The name of the image.
- `type`: Type of the image.

- `distribution` - The name of the distribution of the OS of the image.
- `min_disk_size`: The minimum 'disk' required for the image.
- `size_gigabytes`: The size of the image in GB.
- `private` - Is image a public image or not. Public images represent Linux distributions or One-Click Applications, while non-public images represent snapshots and backups and are only available within your account.
- `regions`: A set of the regions that the image is available in.
- `tags`: A set of tags applied to the image
- `created`: When the image was created
- `status`: Current status of the image
- `error_message`: Any applicable error message pertaining to the image
- `image` - The id of the image (legacy parameter).

» digitalocean_images

Get information on images for use in other resources (e.g. creating a Droplet based on a snapshot), with the ability to filter and sort the results. If no filters are specified, all images will be returned.

This data source is useful if the image in question is not managed by Terraform or you need to utilize any of the image's data.

Note: You can use the `digitalocean_image` data source to obtain metadata about a single image if you already know the `slug`, unique `name`, or `id` to retrieve.

» Example Usage

Use the `filter` block with a `key` string and `values` list to filter images.

For example to find all Ubuntu images:

```
data "digitalocean_images" "ubuntu" {
  filter {
    key = "distribution"
    values = ["Ubuntu"]
  }
}
```

You can filter on multiple fields and sort the results as well:

```
data "digitalocean_images" "available" {
  filter {
    key = "distribution"
    values = ["Ubuntu"]
  }
  filter {
    key = "regions"
    values = ["nyc3"]
  }
  sort {
    key = "created"
    direction = "desc"
  }
}
```

» Argument Reference

- `filter` - (Optional) Filter the results. The `filter` block is documented below.
- `sort` - (Optional) Sort the results. The `sort` block is documented below.

`filter` supports the following arguments:

- `key` - (Required) Filter the images by this key. This may be one of `distribution`, `error_message`, `id`, `image`, `min_disk_size`, `name`, `private`, `regions`, `size_gigabytes`, `slug`, `status`, `tags`, or `type`.
- `values` - (Required) A list of values to match against the `key` field. Only retrieves images where the `key` field takes on one or more of the values provided here.

`sort` supports the following arguments:

- `key` - (Required) Sort the images by this key. This may be one of `distribution`, `error_message`, `id`, `image`, `min_disk_size`, `name`, `private`, `size_gigabytes`, `slug`, `status`, or `type`.
- `direction` - (Required) The sort direction. This may be either `asc` or `desc`.

» Attributes Reference

- `images` - A set of images satisfying any `filter` and `sort` criteria. Each image has the following attributes:

  - `slug`: Unique text identifier of the image.
  - `id`: The ID of the image.
  - `name`: The name of the image.
  - `type`: Type of the image.
  - `distribution` - The name of the distribution of the OS of the image.
  - `min_disk_size`: The minimum 'disk' required for the image.
  - `size_gigabytes`: The size of the image in GB.

  - `private` - Is image a public image or not. Public images represent Linux distributions or One-Click Applications, while non-public images represent snapshots and backups and are only available within your account.
  - `regions`: A set of the regions that the image is available in.
  - `tags`: A set of tags applied to the image
  - `created`: When the image was created
  - `status`: Current status of the image
  - `error_message`: Any applicable error message pertaining to the image
  - `image` - The id of the image (legacy parameter).

» digitalocean_loadbalancer

Get information on a load balancer for use in other resources. This data source provides all of the load balancers properties as configured on your DigitalOcean account. This is useful if the load balancer in question is not managed by Terraform or you need to utilize any of the load balancers data.

An error is triggered if the provided load balancer name does not exist.

» Example Usage

Get the load balancer:

```
data "digitalocean_loadbalancer" "example" {
  name = "app"
}

output "lb_output" {
  value = data.digitalocean_loadbalancer.example.ip
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of load balancer.
- `urn` - The uniform resource name for the Load Balancer

» Attributes Reference

See the Load Balancer Resource for details on the returned attributes - they are identical.

» digitalocean_project

Get information on a single DigitalOcean project. If neither the `id` nor `name` attributes are provided, then this data source returns the default project.

» Example Usage

```
data "digitalocean_project" "default" {
}

data "digitalocean_project" "staging" {
  name = "My Staging Project"
}
```

» Argument Reference

- `id` - (Optional) the ID of the project to retrieve
- `name` - (Optional) the name of the project to retrieve. The data source will raise an error if more than one project has the provided name or if no project has that name.

» Attributes Reference

- `description` - The description of the project
- `purpose` - The purpose of the project, (Default: ”Web Application”)
- `environment` - The environment of the project's resources. The possible values are: `Development`, `Staging`, `Production`.
- `resources` - A set of uniform resource names (URNs) for the resources associated with the project
- `owner_uuid` - The unique universal identifier of the project owner.
- `owner_id` - The ID of the project owner.
- `created_at` - The date and time when the project was created, (ISO8601)
- `updated_at` - The date and time when the project was last updated, (ISO8601)

» digitalocean_projects

Retrieve information about all DigitalOcean projects associated with an account, with the ability to filter and sort the results. If no filters are specified, all projects will be returned.

Note: You can use the `digitalocean_project` data source to obtain metadata about a single project if you already know the `id` to retrieve or the unique `name` of the project.

» Example Usage

Use the `filter` block with a `key` string and `values` list to filter projects.

For example to find all staging environment projects:

```
data "digitalocean_projects" "staging" {
  filter {
    key = "environment"
    values = ["Staging"]
  }
}
```

You can filter on multiple fields and sort the results as well:

```
data "digitalocean_projects" "non-default-production" {
  filter {
    key = "environment"
    values = ["Production"]
  }
  filter {
    key = "is_default"
    values = ["false"]
  }
  sort {
    key = "name"
    direction = "asc"
  }
}
```

» Argument Reference

- `filter` - (Optional) Filter the results. The `filter` block is documented below.
- `sort` - (Optional) Sort the results. The `sort` block is documented below.

filter supports the following arguments:

- key - (Required) Filter the projects by this key. This may be one of name, purpose, description, environment, or is_default.
- values - (Required) A list of values to match against the key field. Only retrieves projects where the key field takes on one or more of the values provided here.

sort supports the following arguments:

- key - (Required) Sort the projects by this key. This may be one of name, purpose, description, or environment.
- direction - (Required) The sort direction. This may be either asc or desc.

» Attributes Reference

- projects - A set of projects satisfying any filter and sort criteria. Each project has the following attributes:
  - id - The ID of the project
  - name - The name of the project
  - description - The description of the project
  - purpose - The purpose of the project (Default: ”Web Application”)
  - environment - The environment of the project’s resources. The possible values are: Development, Staging, Production.
  - resources - A set of uniform resource names (URNs) for the resources associated with the project
  - owner_uuid - The unique universal identifier of the project owner
  - owner_id - The ID of the project owner
  - created_at - The date and time when the project was created, (ISO8601)
  - updated_at - The date and time when the project was last updated, (ISO8601)

» digitalocean_kubernetes_cluster

Retrieves information about a DigitalOcean Kubernetes cluster for use in other resources. This data source provides all of the cluster’s properties as configured on your DigitalOcean account. This is useful if the cluster in question is not managed by Terraform.

» Example Usage

```
data "digitalocean_kubernetes_cluster" "example" {
  name = "prod-cluster-01"
}

provider "kubernetes" {
  load_config_file = false
  host  = data.digitalocean_kubernetes_cluster.example.endpoint
  token = data.digitalocean_kubernetes_cluster.example.kube_config[0].token
  cluster_ca_certificate = base64decode(
    data.digitalocean_kubernetes_cluster.example.kube_config[0].cluster_ca_certificate
  )
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of Kubernetes cluster.

» Attributes Reference

The following attributes are exported:

- `id` - The unique ID that can be used to identify and reference a Kubernetes cluster.
- `region` - The slug identifier for the region where the Kubernetes cluster is located.
- `version` - The slug identifier for the version of Kubernetes used for the cluster.
- `tags` - A list of tag names to be applied to the Kubernetes cluster.
- `cluster_subnet` - The range of IP addresses in the overlay network of the Kubernetes cluster.
- `service_subnet` - The range of assignable IP addresses for services running in the Kubernetes cluster.
- `ipv4_address` - The public IPv4 address of the Kubernetes master node.
- `endpoint` - The base URL of the API server on the Kubernetes master node.
- `status` - A string indicating the current status of the cluster. Potential values include running, provisioning, and errored.
- `created_at` - The date and time when the Kubernetes cluster was created.
- `updated_at` - The date and time when the Kubernetes cluster was last updated.

- `kube_config.0` - A representation of the Kubernetes cluster's kubeconfig with the following attributes:
  - `raw_config` - The full contents of the Kubernetes cluster's kubeconfig file.
  - `host` - The URL of the API server on the Kubernetes master node.
  - `cluster_ca_certificate` - The base64 encoded public certificate for the cluster's certificate authority.
  - `token` - The DigitalOcean API access token used by clients to access the cluster.
  - `client_key` - The base64 encoded private key used by clients to access the cluster. Only available if token authentication is not supported on your cluster.
  - `client_certificate` - The base64 encoded public certificate used by clients to access the cluster. Only available if token authentication is not supported on your cluster.
  - `expires_at` - The date and time when the credentials will expire and need to be regenerated.
- `node_pool` - A list of node pools associated with the cluster. Each node pool exports the following attributes:
  - `id` - The unique ID that can be used to identify and reference the node pool.
  - `name` - The name of the node pool.
  - `size` - The slug identifier for the type of Droplet used as workers in the node pool.
  - `node_count` - The number of Droplet instances in the node pool.
  - `actual_node_count` - The actual number of nodes in the node pool, which is especially useful when auto-scaling is enabled.
  - `auto_scale` - A boolean indicating whether auto-scaling is enabled on the node pool.
  - `min_nodes` - If auto-scaling is enabled, this represents the minimum number of nodes that the node pool can be scaled down to.
  - `max_nodes` - If auto-scaling is enabled, this represents the maximum number of nodes that the node pool can be scaled up to.
  - `tags` - A list of tag names applied to the node pool.
  - `labels` - A map of key/value pairs applied to nodes in the pool. The labels are exposed in the Kubernetes API as labels in the metadata of the corresponding Node resources.
  - `nodes` - A list of nodes in the pool. Each node exports the following attributes:
    - `id` - A unique ID that can be used to identify and reference the node.
    - `name` - The auto-generated name for the node.
    - `status` - A string indicating the current status of the individual node.
    - `created_at` - The date and time when the node was created.
    - `updated_at` - The date and time when the node was last up-

dated.

» digitalocean_kubernetes_versions

Provides access to the available DigitalOcean Kubernetes Service versions.

» Example Usage

» Output a list of all available versions

```
data "digitalocean_kubernetes_versions" "example" {}

output "k8s-versions" {
  value = data.digitalocean_kubernetes_versions.example.valid_versions
}
```

» Create a Kubernetes cluster using the most recent version available

```
data "digitalocean_kubernetes_versions" "example" {}

resource "digitalocean_kubernetes_cluster" "example-cluster" {
  name    = "example-cluster"
  region  = "lon1"
  version = data.digitalocean_kubernetes_versions.example.latest_version

  node_pool {
    name = "default"
    size  = "s-1vcpu-2gb"
    node_count = 3
  }
}
```

» Pin a Kubernetes cluster to a specific minor version

```
data "digitalocean_kubernetes_versions" "example" {
  version_prefix = "1.16."
}

resource "digitalocean_kubernetes_cluster" "example-cluster" {
  name    = "example-cluster"
  region  = "lon1"
  version = data.digitalocean_kubernetes_versions.example.latest_version
  node_pool {
    name = "default"
    size  = "s-1vcpu-2gb"
    node_count = 3
  }
}
```

» Argument Reference

The following arguments are supported:

- `version_prefix` - (Optional) If provided, Terraform will only return versions that match the string prefix. For example, `1.15.` will match all 1.15.x series releases.

» Attributes Reference

The following attributes are exported:

- `valid_versions` - A list of available versions.
- `latest_version` - The most recent version available.

» digitalocean_record

Get information on a DNS record. This data source provides the name, TTL, and zone file as configured on your DigitalOcean account. This is useful if the record in question is not managed by Terraform.

An error is triggered if the provided domain name or record are not managed with your DigitalOcean account.

» Example Usage

Get data from a DNS record:

```
data "digitalocean_record" "example" {
  domain  = "example.com"
  name    = "test"
}

output "record_type" {
  value = data.digitalocean_record.example.type
}

output "record_ttl" {
  value = data.digitalocean_record.example.ttl
}
```

```
  $ terraform apply

data.digitalocean_record.example: Refreshing state...

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

record_ttl = 3600
record_type = A
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the record.
- `domain` - (Required) The domain name of the record.

» Attributes Reference

The following attributes are exported:

- `id`: The ID of the record.
- `type`: The type of the DNS record.
- `data`: Variable data depending on record type. For example, the ”data” value for an A record would be the IPv4 address to which the domain will be mapped. For a CAA record, it would contain the domain name of the CA being granted permission to issue certificates.
- `priority`: The priority for SRV and MX records.
- `port`: The port for SRV records.
- `ttl`: This value is the 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.
- `weight`: The weight for SRV records.
- `flags`: An unsigned integer between 0-255 used for CAA records.
- `tag`: The parameter tag for CAA records.

» digitalocean_region

Get information on a single DigitalOcean region. This is useful to find out what Droplet sizes and features are supported within a region.

» Example Usage

```
data "digitalocean_region" "sfo2" {
  slug = "sfo2"
}

output "region_name" {
  value = data.digitalocean_region.sfo2.name
}
```

» Argument Reference

- `slug` - (Required) A human-readable string that is used as a unique identifier for each region.

» Attributes Reference

- `slug` - A human-readable string that is used as a unique identifier for each region.
- `name` - The display name of the region.
- `available` - A boolean value that represents whether new Droplets can be created in this region.
- `sizes` - A set of identifying slugs for the Droplet sizes available in this region.
- `features` - A set of features available in this region.

» digitalocean_regions

Retrieve information about all supported DigitalOcean regions, with the ability to filter and sort the results. If no filters are specified, all regions will be returned.

Note: You can use the `digitalocean_region` data source to obtain metadata about a single region if you already know the `slug` to retrieve.

» Example Usage

Use the `filter` block with a `key` string and `values` list to filter regions.

For example to find all available regions:

```
data "digitalocean_regions" "available" {
  filter {
    key = "available"
    values = ["true"]
  }
}
```

You can filter on multiple fields and sort the results as well:

```
data "digitalocean_regions" "available" {
  filter {
    key = "available"
    values = ["true"]
  }
  filter {
    key = "features"
    values = ["private_networking"]
  }
  sort {
    key = "name"
    direction = "desc"
  }
}
```

» Argument Reference

- `filter` - (Optional) Filter the results. The `filter` block is documented below.
- `sort` - (Optional) Sort the results. The `sort` block is documented below.

`filter` supports the following arguments:

- `key` - (Required) Filter the regions by this key. This may be one of `slug`, `name`, `available`, `features`, or `sizes`.
- `values` - (Required) A list of values to match against the `key` field. Only retrieves regions where the `key` field takes on one or more of the values provided here.

`sort` supports the following arguments:

- `key` - (Required) Sort the regions by this key. This may be one of `slug`, `name`, or `available`.

- `direction` - (Required) The sort direction. This may be either `asc` or `desc`.

» Attributes Reference

- `regions` - A set of regions satisfying any `filter` and `sort` criteria. Each region has the following attributes:
  - `slug` - A human-readable string that is used as a unique identifier for each region.
  - `name` - The display name of the region.
  - `available` - A boolean value that represents whether new Droplets can be created in this region.
  - `sizes` - A set of identifying slugs for the Droplet sizes available in this region.
  - `features` - A set of features available in this region.

» digitalocean_sizes

Retrieves information about the Droplet sizes that DigitalOcean supports, with the ability to filter and sort the results. If no filters are specified, all sizes will be returned.

» Example Usage

Most common usage will probably be to supply a size to droplet:

```
data "digitalocean_sizes" "main" {
  filter {
    key    = "slug"
    values = ["s-1vcpu-1gb"]
  }
}

resource "digitalocean_droplet" "web" {
  image  = "ubuntu-18-04-x64"
  name   = "web-1"
  region = "sgp1"
  size   = element(data.digitalocean_sizes.main.sizes, 0).slug
}
```

The data source also supports multiple filters and sorts. For example, to fetch sizes with 1 or 2 virtual CPU that are available ”sgp1” region, then pick the cheapest one:

```
data "digitalocean_sizes" "main" {
  filter {
    key    = "vcpus"
    values = [1, 2]
  }

  filter {
    key    = "regions"
    values = ["sgp1"]
  }

  sort {
    key       = "price_monthly"
    direction = "asc"
  }
}

resource "digitalocean_droplet" "web" {
  image  = "ubuntu-18-04-x64"
  name   = "web-1"
  region = "sgp1"
  size   = element(data.digitalocean_sizes.main.sizes, 0).slug
}
```

The data source can also handle multiple sorts. In which case, the sort will be applied in the order it is defined. For example, to sort by memory in ascending order, then sort by disk in descending order between sizes with same memory:

```
data "digitalocean_sizes" "main" {
  sort {
    // Sort by memory ascendingly
    key       = "memory"
    direction = "asc"
  }

  sort {
    // Then sort by disk descendingly for sizes with same memory
    key       = "disk"
    direction = "desc"
  }
}
```

» Argument Reference

The following arguments are supported:

- `filter` - (Optional) Filter the results. The `filter` block is documented below.
- `sort` - (Optional) Sort the results. The `sort` block is documented below.

`filter` supports the following arguments:

- `key` - (Required) Filter the sizes by this key. This may be one of `slug`, `regions`, `memory`, `vcpus`, `disk`, `transfer`, `price_monthly`, `price_hourly`, or `available`.
- `values` - (Required) Only retrieves images which keys has value that matches one of the values provided here.

`sort` supports the following arguments:

- `key` - (Required) Sort the sizes by this key. This may be one of `slug`, `memory`, `vcpus`, `disk`, `transfer`, `price_monthly`, or `price_hourly`.
- `direction` - (Required) The sort direction. This may be either `asc` or `desc`.

» Attributes Reference

The following attributes are exported:

- `slug` - A human-readable string that is used to uniquely identify each size.
- `available` - This represents whether new Droplets can be created with this size.
- `transfer` - The amount of transfer bandwidth that is available for Droplets created in this size. This only counts traffic on the public interface. The value is given in terabytes.
- `price_monthly` - The monthly cost of Droplets created in this size if they are kept for an entire month. The value is measured in US dollars.
- `price_hourly` - The hourly cost of Droplets created in this size as measured hourly. The value is measured in US dollars.
- `memory` - The amount of RAM allocated to Droplets created of this size. The value is measured in megabytes.
- `vcpus` - The number of CPUs allocated to Droplets of this size.
- `disk` - The amount of disk space set aside for Droplets of this size. The value is measured in gigabytes.
- `regions` - List of region slugs where Droplets can be created in this size.

» digitalocean_ssh_key

Get information on a ssh key. This data source provides the name, public key, and fingerprint as configured on your DigitalOcean account. This is useful if the ssh key in question is not managed by Terraform or you need to utilize any of the keys data.

An error is triggered if the provided ssh key name does not exist.

» Example Usage

Get the ssh key:

```
data "digitalocean_ssh_key" "example" {
  name = "example"
}

resource "digitalocean_droplet" "example" {
  image    = "ubuntu-18-04-x64"
  name     = "example-1"
  region   = "nyc2"
  size     = "s-1vcpu-1gb"
  ssh_keys = [data.digitalocean_ssh_key.example.id]
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the ssh key.

» Attributes Reference

The following attributes are exported:

- `id`: The ID of the ssh key.
- `public_key`: The public key of the ssh key.
- `fingerprint`: The fingerprint of the public key of the ssh key.

» digitalocean_tag

Get information on a tag. This data source provides the name as configured on your DigitalOcean account. This is useful if the tag name in question is not managed by Terraform or you need validate if the tag exists in the account.

An error is triggered if the provided tag name does not exist.

» Example Usage

Get the tag:

```
data "digitalocean_tag" "example" {
  name = "example"
}

resource "digitalocean_droplet" "example" {
  image  = "ubuntu-18-04-x64"
  name   = "example-1"
  region = "nyc2"
  size   = "s-1vcpu-1gb"
  tags   = [data.digitalocean_tag.example.name]
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the tag.

» Attributes Reference

The following attributes are exported:

- `id`: The ID of the tag.

» digitalocean_volume

Get information on a volume for use in other resources. This data source provides all of the volumes properties as configured on your DigitalOcean account. This is useful if the volume in question is not managed by Terraform or you need to utilize any of the volumes data.

An error is triggered if the provided volume name does not exist.

» Example Usage

Get the volume:

```
data "digitalocean_volume" "example" {
  name   = "app-data"
  region = "nyc3"
}
```

Reuse the data about a volume to attach it to a Droplet:

```
data "digitalocean_volume" "example" {
  name   = "app-data"
  region = "nyc3"
}

resource "digitalocean_droplet" "example" {
  name       = "foo"
  size       = "s-1vcpu-1gb"
  image      = "ubuntu-18-04-x64"
  region     = "nyc3"
}

resource "digitalocean_volume_attachment" "foobar" {
  droplet_id = digitalocean_droplet.example.id
  volume_id  = data.digitalocean_volume.example.id
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of block storage volume.
- `region` - (Optional) The region the block storage volume is provisioned in.

» Attributes Reference

The following attributes are exported:

- `id`: The ID of the block storage volume.
- `urn`: The uniform resource name for the storage volume.
- `size` - The size of the block storage volume in GiB.
- `description` - Text describing a block storage volume.
- `filesystem_type` - Filesystem type currently in-use on the block storage volume.

- `filesystem_label` - Filesystem label currently in-use on the block storage volume.
- `droplet_ids` - A list of associated Droplet ids.
- `tags` - A list of the tags associated to the Volume.

» digitalocean_volume_snapshot

Volume snapshots are saved instances of a block storage volume. Use this data source to retrieve the ID of a DigitalOcean volume snapshot for use in other resources.

» Example Usage

Get the volume snapshot:

```
data "digitalocean_volume_snapshot" "snapshot" {
  name_regex  = "^web"
  region      = "nyc3"
  most_recent = true
}
```

Reuse the data about a volume snapshot to create a new volume based on it:

```
data "digitalocean_volume_snapshot" "snapshot" {
  name_regex  = "^web"
  region      = "nyc3"
  most_recent = true
}

resource "digitalocean_volume" "foobar" {
  region      = "nyc3"
  name        = "baz"
  size        = 100
  snapshot_id = data.digitalocean_volume_snapshot.snapshot.id
}
```

» Argument Reference

- `name` - (Optional) The name of the volume snapshot.

- `name_regex` - (Optional) A regex string to apply to the volume snapshot list returned by DigitalOcean. This allows more advanced filtering not supported from the DigitalOcean API. This filtering is done locally on what DigitalOcean returns.

- `region` - (Optional) A ”slug” representing a DigitalOcean region (e.g. `nyc1`). If set, only volume snapshots available in the region will be returned.
- `most_recent` - (Optional) If more than one result is returned, use the most recent volume snapshot.

NOTE: If more or less than a single match is returned by the search, Terraform will fail. Ensure that your search is specific enough to return a single volume snapshot ID only, or use `most_recent` to choose the most recent one.

» Attributes Reference

The following attributes are exported:

- `id` The ID of the volume snapshot.
- `created_at` - The date and time the volume snapshot was created.
- `min_disk_size` - The minimum size in gigabytes required for a volume to be created based on this volume snapshot.
- `regions` - A list of DigitalOcean region ”slugs” indicating where the volume snapshot is available.
- `volume_id` - The ID of the volume from which the volume snapshot originated.
- `size` - The billable size of the volume snapshot in gigabytes.
- `tags` - A list of the tags associated to the volume snapshot.

» digitalocean_cdn

Provides a DigitalOcean CDN Endpoint resource for use with Spaces.

» Example Usage

» Basic Example

```
# Create a new Spaces Bucket
resource "digitalocean_spaces_bucket" "mybucket" {
  name   = "example"
  region = "sfo2"
  acl    = "public-read"
}

# Add a CDN endpoint to the Spaces Bucket
resource "digitalocean_cdn" "mycdn" {
  origin = digitalocean_spaces_bucket.mybucket.bucket_domain_name
}

# Output the endpoint for the CDN resource
output "fqdn" {
  value = digitalocean_cdn.mycdn.endpoint
}
```

» Custom Sub-Domain Example

```
# Create a new Spaces Bucket
resource "digitalocean_spaces_bucket" "mybucket" {
  name   = "example"
  region = "sfo2"
  acl    = "public-read"
}

# Create a DigitalOcean managed Let's Encrypt Certificate
resource "digitalocean_certificate" "cert" {
  name    = "cdn-cert"
  type    = "lets_encrypt"
  domains = ["static.example.com"]
}

# Add a CDN endpoint with a custom sub-domain to the Spaces Bucket
resource "digitalocean_cdn" "mycdn" {
  origin         = digitalocean_spaces_bucket.mybucket.bucket_domain_name
  custom_domain  = "static.example.com"
  certificate_id = digitalocean_certificate.cert.id
}
```

» Argument Reference

The following arguments are supported:

- `origin` - (Required) The fully qualified domain name, (FQDN) for a Space.
- `ttl` - (Optional) The time to live for the CDN Endpoint, in seconds. Default is 3600 seconds.
- `certificate_id`- (Optional) The ID of a DigitalOcean managed TLS certificate used for SSL when a custom subdomain is provided.
- `custom_domain` - (Optional) The fully qualified domain name (FQDN) of the custom subdomain used with the CDN Endpoint.

» Attributes Reference

The following attributes are exported:

- `id` - A unique ID that can be used to identify and reference a CDN Endpoint.
- `origin` - The fully qualified domain name, (FQDN) of a space referenced by the CDN Endpoint.
- `endpoint` - The fully qualified domain name (FQDN) from which the CDN-backed content is served.
- `created_at` - The date and time when the CDN Endpoint was created.
- `ttl` - The time to live for the CDN Endpoint, in seconds.
- `certificate_id`- The ID of a DigitalOcean managed TLS certificate used for SSL when a custom subdomain is provided.
- `custom_domain` - The fully qualified domain name (FQDN) of the custom subdomain used with the CDN Endpoint.

» Import

CDN Endpoints can be imported using the CDN `id`, e.g.

```
terraform import digitalocean_cdn.mycdn fb06ad00-351f-45c8-b5eb-13523c438661
```

» digitalocean_certificate

Provides a DigitalOcean Certificate resource that allows you to manage certificates for configuring TLS termination in Load Balancers. Certificates created with this resource can be referenced in your Load Balancer configuration via their ID. The certificate can either be a custom one provided by you or automatically generated one with Let's Encrypt.

» Example Usage

» Custom Certificate

```
resource "digitalocean_certificate" "cert" {
  name              = "custom-terraform-example"
  type              = "custom"
  private_key       = file("/Users/terraform/certs/privkey.pem")
  leaf_certificate  = file("/Users/terraform/certs/cert.pem")
  certificate_chain = file("/Users/terraform/certs/fullchain.pem")
}
```

» Let's Encrypt Certificate

```
resource "digitalocean_certificate" "cert" {
  name    = "le-terraform-example"
  type    = "lets_encrypt"
  domains = ["example.com"]
}
```

» Use with Other Resources

Both custom and Let's Encrypt certificates can be used with other resources including the `digitalocean_loadbalancer` and `digitalocean_cdn` resources.

```
resource "digitalocean_certificate" "cert" {
  name    = "le-terraform-example"
  type    = "lets_encrypt"
  domains = ["example.com"]
}

# Create a new Load Balancer with TLS termination
resource "digitalocean_loadbalancer" "public" {
  name        = "secure-loadbalancer-1"
  region      = "nyc3"
  droplet_tag = "backend"

  forwarding_rule {
    entry_port     = 443
    entry_protocol = "https"

    target_port     = 80
    target_protocol = "http"

    certificate_id = digitalocean_certificate.cert.id
  }
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the certificate for identification.
- `type` - (Optional) The type of certificate to provision. Can be either `custom` or `lets_encrypt`. Defaults to `custom`.
- `private_key` - (Optional) The contents of a PEM-formatted private-key corresponding to the SSL certificate. Only valid when type is `custom`.

- `leaf_certificate` - (Optional) The contents of a PEM-formatted public TLS certificate. Only valid when type is `custom`.
- `certificate_chain` - (Optional) The full PEM-formatted trust chain between the certificate authority's certificate and your domain's TLS certificate. Only valid when type is `custom`.
- `domains` - (Optional) List of fully qualified domain names (FQDNs) for which the certificate will be issued. The domains must be managed using DigitalOcean's DNS. Only valid when type is `lets_encrypt`.

» Attributes Reference

The following attributes are exported:

- `id` - The unique ID of the certificate
- `name` - The name of the certificate
- `not_after` - The expiration date of the certificate
- `sha1_fingerprint` - The SHA-1 fingerprint of the certificate

» Import

Certificates can be imported using the certificate `id`, e.g.

```
terraform import digitalocean_certificate.mycertificate 892071a0-bb95-49bc-8021-3afd67a210bf
```

» digitalocean_database_cluster

Provides a DigitalOcean database cluster resource.

» Example Usage

» Create a new PostgreSQL database cluster

```
resource "digitalocean_database_cluster" "postgres-example" {
  name       = "example-postgres-cluster"
  engine     = "pg"
  version    = "11"
  size       = "db-s-1vcpu-1gb"
  region     = "nyc1"
  node_count = 1
}
```

» Create a new MySQL database cluster

```
resource "digitalocean_database_cluster" "mysql-example" {
  name       = "example-mysql-cluster"
  engine     = "mysql"
  version    = "8"
  size       = "db-s-1vcpu-1gb"
  region     = "nyc1"
  node_count = 1
}
```

» Create a new Redis database cluster

```
resource "digitalocean_database_cluster" "redis-example" {
  name       = "example-redis-cluster"
  engine     = "redis"
  version    = "5"
  size       = "db-s-1vcpu-1gb"
  region     = "nyc1"
  node_count = 1
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the database cluster.
- `engine` - (Required) Database engine used by the cluster (ex. `pg` for PostreSQL, `mysql` for MySQL, or `redis` for Redis).
- `size` - (Required) Database Droplet size associated with the cluster (ex. `db-s-1vcpu-1gb`).
- `region` - (Required) DigitalOcean region where the cluster will reside.
- `node_count` - (Required) Number of nodes that will be included in the cluster.
- `version` - (Required) Engine version used by the cluster (ex. `11` for PostgreSQL 11).
- `tags` - (Optional) A list of tag names to be applied to the database cluster.
- `eviction_policy` - (Optional) A string specifying the eviction policy for a Redis cluster. Valid values are: `noeviction`, `allkeys_lru`, `allkeys_random`, `volatile_lru`, `volatile_random`, or `volatile_ttl`.
- `sql_mode` - (Optional) A comma separated string specifying the SQL modes for a MySQL cluster.
- `maintenance_window` - (Optional) Defines when the automatic maintenance should be performed for the database cluster.

`maintenance_window` supports the following:

- `day` - (Required) The day of the week on which to apply maintenance updates.
- `hour` - (Required) The hour in UTC at which maintenance updates will be applied in 24 hour format.

» Attributes Reference

In addition to the above arguments, the following attributes are exported:

- `id` - The ID of the database cluster.
- `urn` - The uniform resource name of the database cluster.
- `host` - Database cluster's hostname.
- `private_host` - Same as `host`, but only accessible from resources within the account and in the same region.
- `port` - Network port that the database cluster is listening on.
- `uri` - The full URI for connecting to the database cluster.
- `private_uri` - Same as `uri`, but only accessible from resources within the account and in the same region.
- `database` - Name of the cluster's default database.
- `user` - Username for the cluster's default user.
- `password` - Password for the cluster's default user.

» Import

Database clusters can be imported using the `id` returned from DigitalOcean, e.g.

```
terraform import digitalocean_database_cluster.mycluster 245bcfd0-7f31-4ce6-a2bc-475a116cca9
```

» digitalocean_database_connection_pool

Provides a DigitalOcean database connection pool resource.

» Example Usage

» Create a new PostgreSQL database connection pool

```
resource "digitalocean_database_connection_pool" "pool-01" {
  cluster_id = digitalocean_database_cluster.postgres-example.id
  name       = "pool-01"
  mode       = "transaction"
  size       = 20
  db_name    = "defaultdb"
  user       = "doadmin"
}

resource "digitalocean_database_cluster" "postgres-example" {
  name       = "example-postgres-cluster"
  engine     = "pg"
  version    = "11"
  size       = "db-s-1vcpu-1gb"
  region     = "nyc1"
  node_count = 1
}
```

» Argument Reference

The following arguments are supported:

- `cluster_id` - (Required) The ID of the source database cluster. Note: This must be a PostgreSQL cluster.
- `name` - (Required) The name for the database connection pool.
- `mode` - (Required) The PGBouncer transaction mode for the connection pool. The allowed values are session, transaction, and statement.
- `size` - (Required) The desired size of the PGBouncer connection pool.
- `db_name` - (Required) The database for use with the connection pool.
- `user` - (Required) The name of the database user for use with the connection pool.

» Attributes Reference

In addition to the above arguments, the following attributes are exported:

- `id` - The ID of the database connection pool.
- `host` - The hostname used to connect to the database connection pool.
- `private_host` - Same as `host`, but only accessible from resources within the account and in the same region.
- `port` - Network port that the database connection pool is listening on.
- `uri` - The full URI for connecting to the database connection pool.
- `private_uri` - Same as `uri`, but only accessible from resources within the account and in the same region.
- `password` - Password for the connection pool's user.

» Import

Database connection pools can be imported using the `id` of the source database cluster and the `name` of the connection pool joined with a comma. For example:

```
terraform import digitalocean_database_connection_pool.pool-01 245bcfd0-7f31-4ce6-a2bc-475a1
```

» digitalocean_database_db

Provides a DigitalOcean database resource. When creating a new database cluster, a default database with name `defaultdb` will be created. Then, this resource can be used to provide additional database inside the cluster.

» Example Usage

» Create a new PostgreSQL database

```
resource "digitalocean_database_db" "database-example" {
  cluster_id = digitalocean_database_cluster.postgres-example.id
  name       = "foobar"
}

resource "digitalocean_database_cluster" "postgres-example" {
  name       = "example-postgres-cluster"
  engine     = "pg"
  version    = "11"
  size       = "db-s-1vcpu-1gb"
  region     = "nyc1"
  node_count = 1
}
```

» Argument Reference

The following arguments are supported:

- `cluster_id` - (Required) The ID of the original source database cluster.
- `name` - (Required) The name for the database.

» Attributes Reference

Only the above arguments are exported.

» Import

Database can be imported using the `id` of the source database cluster and the `name` of the database joined with a comma. For example:

```
terraform import digitalocean_database_db.database-example 245bcfd0-7f31-4ce6-a2bc-475a116cc
```

» digitalocean_database_firewall

Provides a DigitalOcean database firewall resource allowing you to restrict connections to your database to trusted sources. You may limit connections to specific Droplets, Kubernetes clusters, or IP addresses.

» Example Usage

» Create a new database firewall allowing multiple IP addresses

```
resource "digitalocean_database_firewall" "example-fw" {
  cluster_id = digitalocean_database_cluster.postgres-example.id

  rule {
    type  = "ip_addr"
    value = "192.168.1.1"
  }

  rule {
    type  = "ip_addr"
    value = "192.0.2.0"
  }
}

resource "digitalocean_database_cluster" "postgres-example" {
  name       = "example-postgres-cluster"
  engine     = "pg"
  version    = "11"
  size       = "db-s-1vcpu-1gb"
  region     = "nyc1"
  node_count = 1
}
```

» Create a new database firewall allowing a Droplet

```
resource "digitalocean_database_firewall" "example-fw" {
  cluster_id = digitalocean_database_cluster.postgres-example.id

  rule {
    type  = "droplet"
    value = digitalocean_droplet.web.id
  }
}

resource "digitalocean_droplet" "web" {
  name   = "web-01"
  size   = "s-1vcpu-1gb"
  image  = "centos-7-x64"
  region = "nyc3"
}

resource "digitalocean_database_cluster" "postgres-example" {
  name       = "example-postgres-cluster"
  engine     = "pg"
  version    = "11"
  size       = "db-s-1vcpu-1gb"
  region     = "nyc1"
  node_count = 1
}
```

» Argument Reference

The following arguments are supported:

- `cluster_id` - (Required) The ID of the target database cluster.
- `rule` - (Required) A rule specifying a resource allowed to access the database cluster. The following arguments must be specified:
  - `type` - (Required) The type of resource that the firewall rule allows to access the database cluster. The possible values are: `droplet`, `k8s`, `ip_addr`, or `tag`.
  - `value` - (Required) The ID of the specific resource, the name of a tag applied to a group of resources, or the IP address that the firewall rule allows to access the database cluster.

» Attributes Reference

In addition to the above arguments, the following attributes are exported:

- `uuid` - A unique identifier for the firewall rule.
- `created_at` - The date and time when the firewall rule was created.

» Import

Database firewalls can be imported using the `id` of the target database cluster
For example:

```
terraform import digitalocean_database_firewall.example-fw 5f55c6cd-863b-4907-99b8-7e09b0275
```

» digitalocean_database_replica

Provides a DigitalOcean database replica resource.

» Example Usage

» Create a new PostgreSQL database replica

```
resource "digitalocean_database_replica" "read-replica" {
  cluster_id = digitalocean_database_cluster.postgres-example.id
  name       = "read-replica"
  size       = "db-s-1vcpu-1gb"
  region     = "nyc1"
}

resource "digitalocean_database_cluster" "postgres-example" {
  name       = "example-postgres-cluster"
  engine     = "pg"
  version    = "11"
  size       = "db-s-1vcpu-1gb"
  region     = "nyc1"
  node_count = 1
}
```

» Argument Reference

The following arguments are supported:

- `cluster_id` - (Required) The ID of the original source database cluster.
- `name` - (Required) The name for the database replica.
- `size` - (Required) Database Droplet size associated with the replica (ex. `db-s-1vcpu-1gb`).
- `region` - (Required) DigitalOcean region where the replica will reside.

» Attributes Reference

In addition to the above arguments, the following attributes are exported:

- `id` - The ID of the database replica.
- `host` - Database replica's hostname.
- `private_host` - Same as `host`, but only accessible from resources within the account and in the same region.
- `port` - Network port that the database replica is listening on.
- `uri` - The full URI for connecting to the database replica.
- `private_uri` - Same as `uri`, but only accessible from resources within the account and in the same region.
- `database` - Name of the replica's default database.
- `user` - Username for the replica's default user.
- `password` - Password for the replica's default user.

» Import

Database replicas can be imported using the `id` of the source database cluster and the `name` of the replica joined with a comma. For example:

```
terraform import digitalocean_database_replica.read-replica 245bcfd0-7f31-4ce6-a2bc-475a116c
```

» digitalocean_database_user

Provides a DigitalOcean database user resource. When creating a new database cluster, a default admin user with name `doadmin` will be created. Then, this resource can be used to provide additional normal users inside the cluster.

NOTE: Any new users created will always have `normal` role, only the default user that comes with database cluster creation has `primary` role. Additional permissions must be managed manually.

» Example Usage

» Create a new PostgreSQL database user

```
resource "digitalocean_database_user" "user-example" {
  cluster_id = digitalocean_database_cluster.postgres-example.id
  name       = "foobar"
}

resource "digitalocean_database_cluster" "postgres-example" {
  name       = "example-postgres-cluster"
  engine     = "pg"
  version    = "11"
  size       = "db-s-1vcpu-1gb"
  region     = "nyc1"
  node_count = 1
}
```

» Argument Reference

The following arguments are supported:

- `cluster_id` - (Required) The ID of the original source database cluster.
- `name` - (Required) The name for the database user.
- `mysql_auth_plugin` - (Optional) The authentication method to use for connections to the MySQL user account. The valid values are `mysql_native_password` or `caching_sha2_password` (this is the default).

» Attributes Reference

In addition to the above arguments, the following attributes are exported:

- `role` - Role for the database user. The value will be either ”primary” or ”normal”.
- `password` - Password for the database user.

» Import

Database user can be imported using the `id` of the source database cluster and the `name` of the user joined with a comma. For example:

```
terraform import digitalocean_database_user.user-example 245bcfd0-7f31-4ce6-a2bc-475a116cca9
```

» digitalocean_domain

Provides a DigitalOcean domain resource.

» Example Usage

```
# Create a new domain
resource "digitalocean_domain" "default" {
  name       = "example.com"
  ip_address = digitalocean_droplet.foo.ipv4_address
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the domain
- `ip_address` - (Optional) The IP address of the domain. If specified, this IP is used to created an initial A record for the domain.

» Attributes Reference

The following attributes are exported:

- `id` - The name of the domain
- `urn` - The uniform resource name of the domain

» Import

Domains can be imported using the `domain name`, e.g.

```
terraform import digitalocean_domain.mydomain mytestdomain.com
```

» digitalocean_droplet

Provides a DigitalOcean Droplet resource. This can be used to create, modify, and delete Droplets. Droplets also support provisioning.

» Example Usage

```
# Create a new Web Droplet in the nyc2 region
resource "digitalocean_droplet" "web" {
  image  = "ubuntu-18-04-x64"
  name   = "web-1"
  region = "nyc2"
  size   = "s-1vcpu-1gb"
}
```

» Argument Reference

The following arguments are supported:

- `image` - (Required) The Droplet image ID or slug.
- `name` - (Required) The Droplet name.
- `region` - (Required) The region to start in.
- `size` - (Required) The unique slug that indentifies the type of Droplet. You can find a list of available slugs on DigitalOcean API documentation.
- `backups` - (Optional) Boolean controlling if backups are made. Defaults to false.
- `monitoring` - (Optional) Boolean controlling whether monitoring agent is installed. Defaults to false.
- `ipv6` - (Optional) Boolean controlling if IPv6 is enabled. Defaults to false.
- `private_networking` - (Optional) Boolean controlling if private networks are enabled. Defaults to false.
- `ssh_keys` - (Optional) A list of SSH IDs or fingerprints to enable in the format `[12345, 123456]`. To retrieve this info, use a tool such as `curl` with the DigitalOcean API, to retrieve them.
- `resize_disk` - (Optional) Boolean controlling whether to increase the disk size when resizing a Droplet. It defaults to `true`. When set to `false`, only the Droplet's RAM and CPU will be resized. **Increasing a Droplet's disk size is a permanent change**. Increasing only RAM and CPU is reversible.
- `tags` - (Optional) A list of the tags to be applied to this Droplet.
- `user_data` (Optional) - A string of the desired User Data for the Droplet.
- `volume_ids` (Optional) - A list of the IDs of each block storage volume to be attached to the Droplet.

NOTE: If you use `volume_ids` on a Droplet, Terraform will assume management over the full set volumes for the instance, and treat additional volumes as a drift. For this reason, `volume_ids` must not be mixed with external `digitalocean_volume_attachment` resources for a given instance.

» Attributes Reference

The following attributes are exported:

- `id` - The ID of the Droplet
- `urn` - The uniform resource name of the Droplet
- `name`- The name of the Droplet
- `region` - The region of the Droplet
- `image` - The image of the Droplet
- `ipv6` - Is IPv6 enabled
- `ipv6_address` - The IPv6 address
- `ipv4_address` - The IPv4 address

- `ipv4_address_private` - The private networking IPv4 address
- `locked` - Is the Droplet locked
- `private_networking` - Is private networking enabled
- `price_hourly` - Droplet hourly price
- `price_monthly` - Droplet monthly price
- `size` - The instance size
- `disk` - The size of the instance's disk in GB
- `vcpus` - The number of the instance's virtual CPUs
- `status` - The status of the Droplet
- `tags` - The tags associated with the Droplet
- `volume_ids` - A list of the attached block storage volumes

» Import

Droplets can be imported using the Droplet `id`, e.g.

```
terraform import digitalocean_droplet.mydroplet 100823
```

» digitalocean_droplet_snapshot

Provides a resource which can be used to create a snapshot from an existing DigitalOcean Droplet.

» Example Usage

```
resource "digitalocean_droplet" "web" {
  name   = "web-01"
  size   = "s-1vcpu-1gb"
  image  = "centos-7-x64"
  region = "nyc3"
}

resource "digitalocean_droplet_snapshot" "web-snapshot" {
  droplet_id = digitalocean_droplet.web.id
  name       = "web-snapshot-01"
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) A name for the Droplet snapshot.

- `droplet_id` - (Required) The ID of the Droplet from which the snapshot will be taken.

» Attributes Reference

The following attributes are exported:

- `id` The ID of the Droplet snapshot.
- `created_at` - The date and time the Droplet snapshot was created.
- `min_disk_size` - The minimum size in gigabytes required for a Droplet to be created based on this snapshot.
- `regions` - A list of DigitalOcean region ”slugs” indicating where the droplet snapshot is available.
- `size` - The billable size of the Droplet snapshot in gigabytes.

» Import

Droplet Snapshots can be imported using the `snapshot id`, e.g.

```
terraform import digitalocean_droplet_snapshot.mysnapshot 123456
```

» digitalocean_firewall

Provides a DigitalOcean Cloud Firewall resource. This can be used to create, modify, and delete Firewalls.

» Example Usage

```
resource "digitalocean_droplet" "web" {
  name   = "web-1"
  size   = "s-1vcpu-1gb"
  image  = "ubuntu-18-04-x64"
  region = "nyc3"
}

resource "digitalocean_firewall" "web" {
  name = "only-22-80-and-443"

  droplet_ids = [digitalocean_droplet.web.id]

  inbound_rule {
    protocol         = "tcp"
    port_range       = "22"
    source_addresses = ["192.168.1.0/24", "2002:1:2::/48"]
  }

  inbound_rule {
    protocol         = "tcp"
    port_range       = "80"
    source_addresses = ["0.0.0.0/0", "::/0"]
  }

  inbound_rule {
    protocol         = "tcp"
    port_range       = "443"
    source_addresses = ["0.0.0.0/0", "::/0"]
  }

  inbound_rule {
    protocol         = "icmp"
    source_addresses = ["0.0.0.0/0", "::/0"]
  }

  outbound_rule {
    protocol              = "tcp"
    port_range            = "53"
    destination_addresses = ["0.0.0.0/0", "::/0"]
  }

  outbound_rule {
    protocol              = "udp"
    port_range            = "53"
    destination_addresses = ["0.0.0.0/0", "::/0"]
  }

  outbound_rule {
    protocol              = "icmp"
    destination_addresses = ["0.0.0.0/0", "::/0"]
  }
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The Firewall name
- `droplet_ids` (Optional) - The list of the IDs of the Droplets assigned to

the Firewall.

- `tags` (Optional) - The names of the Tags assigned to the Firewall.
- `inbound_rule` - (Optional) The inbound access rule block for the Firewall. The `inbound_rule` block is documented below.
- `outbound_rule` - (Optional) The outbound access rule block for the Firewall. The `outbound_rule` block is documented below.

`inbound_rule` supports the following:

- `protocol` - (Required) The type of traffic to be allowed. This may be one of ”tcp”, ”udp”, or ”icmp”.
- `port_range` - (Optional) The ports on which traffic will be allowed specified as a string containing a single port, a range (e.g. ”8000-9000”), or ”1-65535” to open all ports for a protocol. Required for when protocol is `tcp` or `udp`.
- `source_addresses` - (Optional) An array of strings containing the IPv4 addresses, IPv6 addresses, IPv4 CIDRs, and/or IPv6 CIDRs from which the inbound traffic will be accepted.
- `source_droplet_ids` - (Optional) An array containing the IDs of the Droplets from which the inbound traffic will be accepted.
- `source_tags` - (Optional) An array containing the names of Tags corresponding to groups of Droplets from which the inbound traffic will be accepted.
- `source_load_balancer_uids` - (Optional) An array containing the IDs of the Load Balancers from which the inbound traffic will be accepted.

`outbound_rule` supports the following:

- `protocol` - (Required) The type of traffic to be allowed. This may be one of ”tcp”, ”udp”, or ”icmp”.
- `port_range` - (Optional) The ports on which traffic will be allowed specified as a string containing a single port, a range (e.g. ”8000-9000”), or ”1-65535” to open all ports for a protocol. Required for when protocol is `tcp` or `udp`.
- `destination_addresses` - (Optional) An array of strings containing the IPv4 addresses, IPv6 addresses, IPv4 CIDRs, and/or IPv6 CIDRs to which the outbound traffic will be allowed.
- `destination_droplet_ids` - (Optional) An array containing the IDs of the Droplets to which the outbound traffic will be allowed.
- `destination_tags` - (Optional) An array containing the names of Tags corresponding to groups of Droplets to which the outbound traffic will be allowed. traffic.
- `destination_load_balancer_uids` - (Optional) An array containing the IDs of the Load Balancers to which the outbound traffic will be allowed.

» Attributes Reference

The following attributes are exported:

- `id` - A unique ID that can be used to identify and reference a Firewall.
- `status` - A status string indicating the current state of the Firewall. This can be ”waiting”, ”succeeded”, or ”failed”.
- `created_at` - A time value given in ISO8601 combined date and time format that represents when the Firewall was created.
- `pending_changes` - An list of object containing the fields, ”droplet_id”, ”removing”, and ”status”. It is provided to detail exactly which Droplets are having their security policies updated. When empty, all changes have been successfully applied.
- `name` - The name of the Firewall.
- `droplet_ids` - The list of the IDs of the Droplets assigned to the Firewall.
- `tags` - The names of the Tags assigned to the Firewall.
- `inbound_rules` - The inbound access rule block for the Firewall.
- `outbound_rules` - The outbound access rule block for the Firewall.

» Import

Firewalls can be imported using the firewall `id`, e.g.

```
terraform import digitalocean_firewall.myfirewall b8ecd2ab-2267-4a5e-8692-cbf1d32583e3
```

» digitalocean_floating_ip

Provides a DigitalOcean Floating IP to represent a publicly-accessible static IP addresses that can be mapped to one of your Droplets.

NOTE: Floating IPs can be assigned to a Droplet either directly on the `digitalocean_floating_ip` resource by setting a `droplet_id` or using the `digitalocean_floating_ip_assignment` resource, but the two cannot be used together.

» Example Usage

```
resource "digitalocean_droplet" "foobar" {
  name               = "baz"
  size               = "s-1vcpu-1gb"
  image              = "ubuntu-18-04-x64"
  region             = "sgp1"
  ipv6               = true
  private_networking = true
}

resource "digitalocean_floating_ip" "foobar" {
  droplet_id = digitalocean_droplet.foobar.id
  region     = digitalocean_droplet.foobar.region
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Required) The region that the Floating IP is reserved to.
- `droplet_id` - (Optional) The ID of Droplet that the Floating IP will be assigned to.

» Attributes Reference

The following attributes are exported:

- `ip_address` - The IP Address of the resource
- `urn` - The uniform resource name of the floating ip

» Import

Floating IPs can be imported using the `ip`, e.g.

```
terraform import digitalocean_floating_ip.myip 192.168.0.1
```

» digitalocean_floating_ip_assignment

Provides a resource for assigning an existing DigitalOcean Floating IP to a Droplet. This makes it easy to provision floating IP addresses that are not tied to the lifecycle of your Droplet.

» Example Usage

```
resource "digitalocean_floating_ip" "foobar" {
  region = "sgp1"
}

resource "digitalocean_droplet" "foobar" {
  name               = "baz"
  size               = "s-1vcpu-1gb"
  image              = "ubuntu-18-04-x64"
  region             = "sgp1"
  ipv6               = true
  private_networking = true
}

resource "digitalocean_floating_ip_assignment" "foobar" {
  ip_address = digitalocean_floating_ip.foobar.ip_address
  droplet_id = digitalocean_droplet.foobar.id
}
```

» Argument Reference

The following arguments are supported:

- `ip_address` - (Required) The Floating IP to assign to the Droplet.
- `droplet_id` - (Optional) The ID of Droplet that the Floating IP will be assigned to.

» digitalocean_loadbalancer

Provides a DigitalOcean Load Balancer resource. This can be used to create, modify, and delete Load Balancers.

» Example Usage

```
resource "digitalocean_droplet" "web" {
  name   = "web-1"
  size   = "s-1vcpu-1gb"
  image  = "ubuntu-18-04-x64"
  region = "nyc3"
}

resource "digitalocean_loadbalancer" "public" {
  name   = "loadbalancer-1"
  region = "nyc3"

  forwarding_rule {
    entry_port     = 80
    entry_protocol = "http"

    target_port     = 80
    target_protocol = "http"
  }

  healthcheck {
    port     = 22
    protocol = "tcp"
  }

  droplet_ids = [digitalocean_droplet.web.id]
}
```

When managing certificates attached to the load balancer, make sure to add the `create_before_destroy` lifecycle property in order to ensure the certificate is correctly updated when changed. The order of operations will then be: `Create new certificate` -> `Update loadbalancer with new certificate` -> `Delete old certificate`. When doing so, you must also change the name of the certificate, as there cannot be multiple certificates with the same name in an account.

```
resource "digitalocean_certificate" "cert" {
  name             = "cert"
  private_key      = "file("key.pem")}"
  leaf_certificate = "file("cert.pem")"

  lifecycle {
    create_before_destroy = true
  }
}

resource "digitalocean_droplet" "web" {
  name      = "web-1"
  size      = "s-1vcpu-1gb"
  image     = "ubuntu-18-04-x64"
  region    = "nyc3"
}

resource "digitalocean_loadbalancer" "public" {
  name = "loadbalancer-1"
  region = "nyc3"

  forwarding_rule {
    entry_port = 443
    entry_protocol = "https"

    target_port = 80
    target_protocol = "http"
    certificate_id = digitalocean_certificate.cert.id
  }

  healthcheck {
    port = 22
    protocol = "tcp"
  }

  droplet_ids = [digitalocean_droplet.web.id]
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The Load Balancer name
- `region` - (Required) The region to start in
- `algorithm` - (Optional) The load balancing algorithm used to determine which backend Droplet will be selected by a client. It must be either `round_robin` or `least_connections`. The default value is `round_robin`.
- `forwarding_rule` - (Required) A list of `forwarding_rule` to be assigned to the Load Balancer. The `forwarding_rule` block is documented below.
- `healthcheck` - (Optional) A `healthcheck` block to be assigned to the Load Balancer. The `healthcheck` block is documented below. Only 1 healthcheck is allowed.
- `sticky_sessions` - (Optional) A `sticky_sessions` block to be assigned to the Load Balancer. The `sticky_sessions` block is documented below. Only 1 sticky_sessions block is allowed.
- `redirect_http_to_https` - (Optional) A boolean value indicating whether HTTP requests to the Load Balancer on port 80 will be redirected to HTTPS on port 443. Default value is `false`.
- `enable_proxy_protocol` - (Optional) A boolean value indicating whether PROXY Protocol should be used to pass information from connecting client requests to the backend service. Default value is `false`.
- `droplet_ids` (Optional) - A list of the IDs of each droplet to be attached to the Load Balancer.
- `droplet_tag` (Optional) - The name of a Droplet tag corresponding to Droplets to be assigned to the Load Balancer.

`forwarding_rule` supports the following:

- `entry_protocol` - (Required) The protocol used for traffic to the Load Balancer. The possible values are: `http`, `https`, `http2` or `tcp`.
- `entry_port` - (Required) An integer representing the port on which the Load Balancer instance will listen.
- `target_protocol` - (Required) The protocol used for traffic from the Load

Balancer to the backend Droplets. The possible values are: `http`, `https`, `http2` or `tcp`.

- `target_port` - (Required) An integer representing the port on the backend Droplets to which the Load Balancer will send traffic.
- `certificate_id` - (Optional) The ID of the TLS certificate to be used for SSL termination.
- `tls_passthrough` - (Optional) A boolean value indicating whether SSL encrypted traffic will be passed through to the backend Droplets. The default value is `false`.

`sticky_sessions` supports the following:

- `type` - (Required) An attribute indicating how and if requests from a client will be persistently served by the same backend Droplet. The possible values are `cookies` or `none`. If not specified, the default value is `none`.
- `cookie_name` - (Optional) The name to be used for the cookie sent to the client. This attribute is required when using `cookies` for the sticky sessions type.
- `cookie_ttl_seconds` - (Optional) The number of seconds until the cookie set by the Load Balancer expires. This attribute is required when using `cookies` for the sticky sessions type.

`healthcheck` supports the following:

- `protocol` - (Required) The protocol used for health checks sent to the backend Droplets. The possible values are `http` or `tcp`.
- `port` - (Optional) An integer representing the port on the backend Droplets on which the health check will attempt a connection.
- `path` - (Optional) The path on the backend Droplets to which the Load Balancer instance will send a request.
- `check_interval_seconds` - (Optional) The number of seconds between between two consecutive health checks. If not specified, the default value is `10`.
- `response_timeout_seconds` - (Optional) The number of seconds the Load Balancer instance will wait for a response until marking a health check as failed. If not specified, the default value is `5`.
- `unhealthy_threshold` - (Optional) The number of times a health check must fail for a backend Droplet to be marked "unhealthy" and be removed from the pool. If not specified, the default value is `3`.
- `healthy_threshold` - (Optional) The number of times a health check must pass for a backend Droplet to be marked "healthy" and be re-added to the pool. If not specified, the default value is `5`.

» Attributes Reference

The following attributes are exported:

- `id` - The ID of the Load Balancer
- `ip`- The ip of the Load Balancer
- `urn` - The uniform resource name for the Load Balancer

» Import

Load Balancers can be imported using the `id`, e.g.

```
terraform import digitalocean_loadbalancer.myloadbalancer 4de7ac8b-495b-4884-9a69-1050c6793c
```

» digitalocean_kubernetes_cluster

Provides a DigitalOcean Kubernetes cluster resource. This can be used to create, delete, and modify clusters. For more information see the official documentation.

» Example Usage

» Basic Example

```
resource "digitalocean_kubernetes_cluster" "foo" {
  name    = "foo"
  region  = "nyc1"
  # Grab the latest version slug from `doctl kubernetes options versions`
  version = "1.15.5-do.1"

  node_pool {
    name       = "worker-pool"
    size       = "s-2vcpu-2gb"
    node_count = 3
  }
}
```

» Kubernetes Terraform Provider Example

The cluster's kubeconfig is exported as an attribute allowing you to use it with the Kubernetes Terraform provider. For example:

```
resource "digitalocean_kubernetes_cluster" "foo" {
  name    = "foo"
  region  = "nyc1"
  # Grab the latest version slug from `doctl kubernetes options versions`
  version = "1.15.5-do.1"
  tags    = ["staging"]
  node_pool {
    name       = "worker-pool"
    size       = "s-2vcpu-2gb"
    node_count = 3
  }
}

provider "kubernetes" {
  load_config_file = false
  host  = digitalocean_kubernetes_cluster.foo.endpoint
  token = digitalocean_kubernetes_cluster.foo.kube_config[0].token
  cluster_ca_certificate = base64decode(
    digitalocean_kubernetes_cluster.foo.kube_config[0].cluster_ca_certificate
  )
}
```

» Autoscaling Example

Node pools may also be configured to autoscale. For example:

```
resource "digitalocean_kubernetes_cluster" "foo" {
  name    = "foo"
  region  = "nyc1"
  version = "1.15.5-do.1"

  node_pool {
    name       = "autoscale-worker-pool"
    size       = "s-2vcpu-2gb"
    auto_scale = true
    min_nodes  = 1
    max_nodes  = 5
  }
}
```

Note that, while individual node pools may scale to 0, a cluster must always include at least one node.

» Argument Reference

The following arguments are supported:

- `name` - (Required) A name for the Kubernetes cluster.
- `region` - (Required) The slug identifier for the region where the Kubernetes cluster will be created.

- `version` - (Required) The slug identifier for the version of Kubernetes used for the cluster. Use doctl to find the available versions `doctl kubernetes options versions`. (**Note:** A cluster may only be upgraded to newer versions in-place. If the version is decreased, a new resource will be created.)
- `node_pool` - (Required) A block representing the cluster's default node pool. Additional node pools may be added to the cluster using the `digitalocean_kubernetes_node_pool` resource. The following arguments may be specified:
  - `name` - (Required) A name for the node pool.
  - `size` - (Required) The slug identifier for the type of Droplet to be used as workers in the node pool.
  - `node_count` - (Optional) The number of Droplet instances in the node pool. If auto-scaling is enabled, this should only be set if the desired result is to explicitly reset the number of nodes to this value. If auto-scaling is enabled, and the node count is outside of the given min/max range, it will use the min nodes value.
  - `auto_scale` - (Optional) Enable auto-scaling of the number of nodes in the node pool within the given min/max range.
  - `min_nodes` - (Optional) If auto-scaling is enabled, this represents the minimum number of nodes that the node pool can be scaled down to.
  - `max_nodes` - (Optional) If auto-scaling is enabled, this represents the maximum number of nodes that the node pool can be scaled up to.
  - `tags` - (Optional) A list of tag names to be applied to the Kubernetes cluster.
  - `labels` - (Optional) A map of key/value pairs to apply to nodes in the pool. The labels are exposed in the Kubernetes API as labels in the metadata of the corresponding Node resources.
- `tags` - (Optional) A list of tag names to be applied to the Kubernetes cluster.

» Attributes Reference

In addition to the arguments listed above, the following additional attributes are exported:

- `id` - A unique ID that can be used to identify and reference a Kubernetes cluster.
- `cluster_subnet` - The range of IP addresses in the overlay network of the Kubernetes cluster.
- `service_subnet` - The range of assignable IP addresses for services running in the Kubernetes cluster.
- `ipv4_address` - The public IPv4 address of the Kubernetes master node.
- `endpoint` - The base URL of the API server on the Kubernetes master node.

- `status` - A string indicating the current status of the cluster. Potential values include running, provisioning, and errored.
- `created_at` - The date and time when the Kubernetes cluster was created.
- `updated_at` - The date and time when the Kubernetes cluster was last updated.
- `kube_config.0` - A representation of the Kubernetes cluster's kubeconfig with the following attributes:
  - `raw_config` - The full contents of the Kubernetes cluster's kubeconfig file.
  - `host` - The URL of the API server on the Kubernetes master node.
  - `cluster_ca_certificate` - The base64 encoded public certificate for the cluster's certificate authority.
  - `token` - The DigitalOcean API access token used by clients to access the cluster.
  - `client_key` - The base64 encoded private key used by clients to access the cluster. Only available if token authentication is not supported on your cluster.
  - `client_certificate` - The base64 encoded public certificate used by clients to access the cluster. Only available if token authentication is not supported on your cluster.
  - `expires_at` - The date and time when the credentials will expire and need to be regenerated.
- `node_pool` - In addition to the arguments provided, these additional attributes about the cluster's default node pool are exported:
  - `id` - A unique ID that can be used to identify and reference the node pool.
  - `actual_node_count` - A computed field representing the actual number of nodes in the node pool, which is especially useful when autoscaling is enabled.
  - `nodes` - A list of nodes in the pool. Each node exports the following attributes:
    - `id` - A unique ID that can be used to identify and reference the node.
    - `name` - The auto-generated name for the node.
    - `status` - A string indicating the current status of the individual node.
    - `droplet_id` - The id of the node's droplet
    - `created_at` - The date and time when the node was created.
    - `updated_at` - The date and time when the node was last updated.

» Import

Before importing a Kubernetes cluster, the cluster's default node pool must be tagged with the `terraform:default-node-pool` tag. The provider will automatically add this tag if the cluster has a single node pool. Clusters with more than one node pool, however, will require that you manually add the `terraform:default-node-pool` tag to the node pool that you intend to be the default node pool.

Then the Kubernetes cluster and all of its node pools can be imported using the cluster's `id`, e.g.

```
terraform import digitalocean_kubernetes_cluster.mycluster 1b8b2100-0e9f-4e8f-ad78-9eb578c2a
```

» digitalocean_kubernetes_node_pool

Provides a DigitalOcean Kubernetes node pool resource. While the default node pool must be defined in the `digitalocean_kubernetes_cluster` resource, this resource can be used to add additional ones to a cluster.

» Example Usage

» Basic Example

```
resource "digitalocean_kubernetes_cluster" "foo" {
  name    = "foo"
  region  = "nyc1"
  version = "1.15.5-do.1"

  node_pool {
    name       = "front-end-pool"
    size       = "s-2vcpu-2gb"
    node_count = 3
  }
}

resource "digitalocean_kubernetes_node_pool" "bar" {
  cluster_id = digitalocean_kubernetes_cluster.foo.id

  name       = "backend-pool"
  size       = "c-2"
  node_count = 2
  tags       = ["backend"]

  labels = {
    service  = "backend"
    priority = "high"
  }
}
```

» Autoscaling Example

Node pools may also be configured to autoscale. For example:

```
resource "digitalocean_kubernetes_node_pool" "autoscale-pool-01" {
  cluster_id = digitalocean_kubernetes_cluster.foo.id
  name       = "autoscale-pool-01"
  size       = "s-1vcpu-2gb"
  auto_scale = true
  min_nodes = 0
  max_nodes = 5
}
```

» Argument Reference

The following arguments are supported:

- `cluster_id` - (Required) The ID of the Kubernetes cluster to which the node pool is associated.
- `name` - (Required) A name for the node pool.
- `size` - (Required) The slug identifier for the type of Droplet to be used as workers in the node pool.
- `node_count` - (Optional) The number of Droplet instances in the node pool. If auto-scaling is enabled, this should only be set if the desired result is to explicitly reset the number of nodes to this value. If auto-scaling is enabled, and the node count is outside of the given min/max range, it will use the min nodes value.
- `auto_scale` - (Optional) Enable auto-scaling of the number of nodes in the node pool within the given min/max range.
- `min_nodes` - (Optional) If auto-scaling is enabled, this represents the minimum number of nodes that the node pool can be scaled down to.
- `max_nodes` - (Optional) If auto-scaling is enabled, this represents the maximum number of nodes that the node pool can be scaled up to.
- `tags` - (Optional) A list of tag names to be applied to the Kubernetes cluster.
- `labels` - (Optional) A map of key/value pairs to apply to nodes in the pool. The labels are exposed in the Kubernetes API as labels in the metadata of the corresponding Node resources.

» Attributes Reference

In addition to the arguments listed above, the following additional attributes are exported:

- `id` - A unique ID that can be used to identify and reference the node pool.
- `actual_node_count` - A computed field representing the actual number of nodes in the node pool, which is especially useful when auto-scaling is enabled.
- `nodes` - A list of nodes in the pool. Each node exports the following attributes:
  - `id` - A unique ID that can be used to identify and reference the node.
  - `name` - The auto-generated name for the node.
  - `status` - A string indicating the current status of the individual node.
  - `droplet_id` - The id of the node's droplet
  - `created_at` - The date and time when the node was created.
  - `updated_at` - The date and time when the node was last updated.

» Import

If you are importing an existing Kubernetes cluster, just import the cluster. Importing a cluster also imports all of its associated node pools.

If you still need to import a single node pool, then import it by using its `id`, e.g.

```
terraform import digitalocean_kubernetes_node_pool.mynodepool 9d76f410-9284-4436-9633-406685
```

Note: If the node pool has the `terraform:default-node-pool` tag, then it is a default node pool for an existing cluster. The provider will refuse to import the node pool in that case because the node pool is managed by the `digitalocean_kubernetes_cluster` resource and not by this `digitalocean_kubernetes_node_pool` resource.

» digitalocean_project

Provides a DigitalOcean Project resource.

Projects allow you to organize your resources into groups that fit the way you work. You can group resources (like Droplets, Spaces, Load Balancers, domains, and Floating IPs) in ways that align with the applications you host on DigitalOcean.

The following resource types can be associated with a project:

- Database Clusters
- Domains

- Droplets
- Floating IP
- Load Balancers
- Spaces Bucket
- Volume

Note: A Terraform managed project cannot be set as a default project.

» Example Usage

The following example demonstrates the creation of an empty project:

```
resource "digitalocean_project" "playground" {
  name        = "playground"
  description = "A project to represent development resources."
  purpose     = "Web Application"
  environment = "Development"
}
```

The following example demonstrates the creation of a project with a Droplet resource:

```
resource "digitalocean_droplet" "foobar" {
  name   = "example"
  size   = "512mb"
  image  = "centos-7-x64"
  region = "nyc3"
}

resource "digitalocean_project" "playground" {
  name        = "playground"
  description = "A project to represent development resources."
  purpose     = "Web Application"
  environment = "Development"
  resources   = [digitalocean_droplet.foobar.urn]
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the Project
- `description` - (Optional) the description of the project
- `purpose` - (Optional) the purpose of the project, (Default: ”Web Application”)

- `environment` - (Optional) the environment of the project's resources. The possible values are: `Development`, `Staging`, `Production`)
- `resources` - a list of uniform resource names (URNs) for the resources associated with the project

» Attributes Reference

The following attributes are exported:

- `id` - The id of the project
- `owner_uuid` - the unique universal identifier of the project owner.
- `owner_id` - the id of the project owner.
- `created_at` - the date and time when the project was created, (ISO8601)
- `updated_at` - the date and time when the project was last updated, (ISO8601)

» Import

Projects can be imported using the `id` returned from DigitalOcean, e.g.

```
terraform import digitalocean_project.myproject 245bcfd0-7f31-4ce6-a2bc-475a116cca97
```

» digitalocean_project_resources

Assign resources to a DigitalOcean Project. This is useful if you need to assign resources managed in Terraform to a DigitalOcean Project managed outside of Terraform.

The following resource types can be associated with a project:

- Database Clusters
- Domains
- Droplets
- Floating IP
- Load Balancers
- Spaces Bucket
- Volume

» Example Usage

The following example assigns a droplet to a Project managed outside of Terraform:

```
data "digitalocean_project" "playground" {
  name = "playground"
}

resource "digitalocean_droplet" "foobar" {
  name   = "example"
  size   = "512mb"
  image  = "centos-7-x64"
  region = "nyc3"
}

resource "digitalocean_project_resources" "barfoo" {
  project = data.digitalocean_project.foo.id
  resources = [
    digitalocean_droplet.foobar.urn
  ]
}
```

» Argument Reference

The following arguments are supported:

- `project` - (Required) the ID of the project
- `resources` - (Required) a list of uniform resource names (URNs) for the resources associated with the project

» Attributes Reference

No additional attributes are exported.

» Import

Importing this resource is not supported.

» digitalocean_record

Provides a DigitalOcean DNS record resource.

» Example Usage

```
resource "digitalocean_domain" "default" {
  name = "example.com"
}

# Add an A record to the domain for www.example.com.
resource "digitalocean_record" "www" {
  domain = digitalocean_domain.default.name
  type   = "A"
  name   = "www"
  value  = "192.168.0.11"
}

# Add a MX record for the example.com domain itself.
resource "digitalocean_record" "mx" {
  domain   = digitalocean_domain.default.name
  type     = "MX"
  name     = "@"
  priority = 10
  value    = "mail.example.com."
}

# Output the FQDN for the www A record.
output "www_fqdn" {
  value = digitalocean_record.www.fqdn  # => www.example.com
}

# Output the FQDN for the MX record.
output "mx_fqdn" {
  value = digitalocean_record.mx.fqdn  # => example.com
}
```

» Argument Reference

The following arguments are supported:

- `type` - (Required) The type of record. Must be one of `A`, `AAAA`, `CAA`, `CNAME`, `MX`, `NS`, `TXT`, or `SRV`.
- `domain` - (Required) The domain to add the record to.
- `value` - (Required) The value of the record.
- `name` - (Required) The name of the record. Use `@` for records on domain's name itself.
- `port` - (Optional) The port of the record. Only valid when type is `SRV`. Must be between 1 and 65535.
- `priority` - (Optional) The priority of the record. Only valid when type is `MX` or `SRV`. Must be between 0 and 65535.
- `weight` - (Optional) The weight of the record. Only valid when type is

`SRV`. Must be between 0 and 65535.
- `ttl` - (Optional) The time to live for the record, in seconds. Must be at least 0.
- `flags` - (Optional) The flags of the record. Only valid when type is `CAA`. Must be between 0 and 255.
- `tag` - (Optional) The tag of the record. Only valid when type is `CAA`. Must be one of `issue`, `issuewild`, or `iodef`.

» Attributes Reference

The following attributes are exported:

- `id` - The record ID
- `fqdn` - The FQDN of the record

» Import

Records can be imported using the domain name and record `id` when joined with a comma. See the following example:

```
terraform import digitalocean_record.example_record example.com,12345678
```

» digitalocean_spaces_bucket

Provides a bucket resource for Spaces, DigitalOcean's object storage product.

The Spaces API was designed to be interoperable with Amazon's AWS S3 API. This allows users to interact with the service while using the tools they already know. Spaces mirrors S3's authentication framework and requests to Spaces require a key pair similar to Amazon's Access ID and Secret Key.

The authentication requirement can be met by either setting the `SPACES_ACCESS_KEY_ID` and `SPACES_SECRET_ACCESS_KEY` environment variables or the provider's `spaces_access_id` and `spaces_secret_key` arguments to the access ID and secret you generate via the DigitalOcean control panel. For example:

```
provider "digitalocean" {
  token             = var.digitalocean_token

  spaces_access_id  = var.access_id
  spaces_secret_key = var.secret_key
}

resource "digitalocean_spaces_bucket" "static-assets" {
  # ...
}
```

For more information, See An Introduction to DigitalOcean Spaces

» Example Usage

» Create a New Bucket

```
resource "digitalocean_spaces_bucket" "foobar" {
  name   = "foobar"
  region = "nyc3"
}
```

» Create a New Bucket With CORS Rules

```
resource "digitalocean_spaces_bucket" "foobar" {
  name   = "foobar"
  region = "nyc3"

  cors_rule {
    allowed_headers = ["*"]
    allowed_methods = ["GET"]
    allowed_origins = ["*"]
    max_age_seconds = 3000
  }

  cors_rule {
    allowed_headers = ["*"]
    allowed_methods = ["PUT", "POST", "DELETE"]
    allowed_origins = ["https://www.example.com"]
    max_age_seconds = 3000
  }
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the bucket
- `region` - The region where the bucket resides (Defaults to `nyc3`)
- `acl` - Canned ACL applied on bucket creation (`private` or `public-read`)
- `force_destroy` - Unless `true`, the bucket will only be destroyed if empty (Defaults to `false`)

The `cors_rule` object supports the following:

- `allowed_headers` - (Optional) A list of headers that will be included in the CORS preflight request's `Access-Control-Request-Headers`. A header may contain one wildcard (e.g. `x-amz-*`).
- `allowed_methods` - (Required) A list of HTTP methods (e.g. `GET`) which are allowed from the specified origin.
- `allowed_origins` - (Required) A list of hosts from which requests using the specified methods are allowed. A host may contain one wildcard (e.g. http://*.example.com).
- `max_age_seconds` - (Optional) The time in seconds that browser can cache the response for a preflight request.

» Attributes Reference

The following attributes are exported:

- `name` - The name of the bucket
- `urn` - The uniform resource name for the bucket
- `region` - The name of the region
- `bucket_domain_name` - The FQDN of the bucket (e.g. bucket-name.nyc3.digitaloceanspaces.com)

» Import

Buckets can be imported using the `region` and `name` attributes (delimited by a comma):

```
terraform import digitalocean_spaces_bucket.foobar `region`,`name`
```

» digitalocean_ssh_key

Provides a DigitalOcean SSH key resource to allow you to manage SSH keys for Droplet access. Keys created with this resource can be referenced in your Droplet configuration via their ID or fingerprint.

» Example Usage

```
# Create a new SSH key
resource "digitalocean_ssh_key" "default" {
  name       = "Terraform Example"
  public_key = file("/Users/terraform/.ssh/id_rsa.pub")
}
```

```
# Create a new Droplet using the SSH key
resource "digitalocean_droplet" "web" {
  image    = "ubuntu-18-04-x64"
  name     = "web-1"
  region   = "nyc3"
  size     = "s-1vcpu-1gb"
  ssh_keys = [digitalocean_ssh_key.default.fingerprint]
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the SSH key for identification
- `public_key` - (Required) The public key. If this is a file, it can be read using the file interpolation function

» Attributes Reference

The following attributes are exported:

- `id` - The unique ID of the key
- `name` - The name of the SSH key
- `public_key` - The text of the public key
- `fingerprint` - The fingerprint of the SSH key

» Import

SSH Keys can be imported using the `ssh key id`, e.g.

```
terraform import digitalocean_ssh_key.mykey 263654
```

» digitalocean_tag

Provides a DigitalOcean Tag resource. A Tag is a label that can be applied to a Droplet resource in order to better organize or facilitate the lookups and actions on it. Tags created with this resource can be referenced in your Droplet configuration via their ID or name.

» Example Usage

```
# Create a new tag
```

```
resource "digitalocean_tag" "foobar" {
  name = "foobar"
}

# Create a new Droplet in nyc3 with the foobar tag
resource "digitalocean_droplet" "web" {
  image  = "ubuntu-18-04-x64"
  name   = "web-1"
  region = "nyc3"
  size   = "s-1vcpu-1gb"
  tags   = [digitalocean_tag.foobar.id]
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the tag

» Attributes Reference

The following attributes are exported:

- `id` - The id of the tag
- `name` - The name of the tag

» Import

Tags can be imported using the `name`, e.g.

```
terraform import digitalocean_tag.mytag tagname
```

» digitalocean_volume

Provides a DigitalOcean Block Storage volume which can be attached to a Droplet in order to provide expanded storage.

» Example Usage

```
resource "digitalocean_volume" "foobar" {
  region                  = "nyc1"
  name                    = "baz"
  size                    = 100
  initial_filesystem_type = "ext4"
  description             = "an example volume"
}

resource "digitalocean_droplet" "foobar" {
  name   = "baz"
  size   = "s-1vcpu-1gb"
  image  = "ubuntu-18-04-x64"
  region = "nyc1"
}

resource "digitalocean_volume_attachment" "foobar" {
  droplet_id = digitalocean_droplet.foobar.id
  volume_id  = digitalocean_volume.foobar.id
}
```

You can also create a volume from an existing snapshot.

```
data "digitalocean_volume_snapshot" "foobar" {
  name = "baz"
}

resource "digitalocean_volume" "foobar" {
  region      = "lon1"
  name        = "foo"
  size        = data.digitalocean_volume_snapshot.foobar.min_disk_size
  snapshot_id = data.digitalocean_volume_snapshot.foobar.id
}
```

» Argument Reference

The following arguments are supported:

- `region` - (Required) The region that the block storage volume will be created in.
- `name` - (Required) A name for the block storage volume. Must be lowercase and be composed only of numbers, letters and "-", up to a limit of 64 characters.
- `size` - (Required) The size of the block storage volume in GiB. If updated, can only be expanded.
- `description` - (Optional) A free-form text field up to a limit of 1024 bytes to describe a block storage volume.
- `snapshot_id` - (Optional) The ID of an existing volume snapshot from which the new volume will be created. If supplied, the region and size will be limitied on creation to that of the referenced snapshot

- `initial_filesystem_type` - (Optional) Initial filesystem type (`xfs` or `ext4`) for the block storage volume.
- `initial_filesystem_label` - (Optional) Initial filesystem label for the block storage volume.
- `tags` - (Optional) A list of the tags to be applied to this Volume.

» Attributes Reference

The following attributes are exported:

- `id` - The unique identifier for the volume.
- `urn` - The uniform resource name for the volume.
- `name` - Name of the volume.
- `description` - Description of the volume.
- `tags` - List of applied tags to the volume.
- `region` - The region that the volume is created in.
- `droplet_ids` - A list of associated droplet ids.
- `snapshot_id` - The ID of the existing volume snapshot from which this volume was created from.
- `filesystem_type` - Filesystem type (`xfs` or `ext4`) for the block storage volume.
- `filesystem_label` - Filesystem label for the block storage volume.
- `initial_filesystem_type` - Filesystem type (`xfs` or `ext4`) for the block storage volume when it was first created.
- `initial_filesystem_label` - Filesystem label for the block storage volume when it was first created.

» Import

Volumes can be imported using the `volume id`, e.g.

```
terraform import digitalocean_volume.volume 506f78a4-e098-11e5-ad9f-000f53306ae1
```

» digitalocean_volume_attachment

Manages attaching a Volume to a Droplet.

NOTE: Volumes can be attached either directly on the `digitalocean_droplet` resource, or using the `digitalocean_volume_attachment` resource - but the two cannot be used together. If both are used against the same Droplet, the volume attachments will constantly drift.

» Example Usage

```
resource "digitalocean_volume" "foobar" {
  region                  = "nyc1"
  name                    = "baz"
  size                    = 100
  initial_filesystem_type = "ext4"
  description             = "an example volume"
}

resource "digitalocean_droplet" "foobar" {
  name   = "baz"
  size   = "s-1vcpu-1gb"
  image  = "ubuntu-18-04-x64"
  region = "nyc1"
}

resource "digitalocean_volume_attachment" "foobar" {
  droplet_id = digitalocean_droplet.foobar.id
  volume_id  = digitalocean_volume.foobar.id
}
```

» Argument Reference

The following arguments are supported:

- `droplet_id` - (Required) ID of the Droplet to attach the volume to.
- `volume_id` - (Required) ID of the Volume to be attached to the Droplet.

» Attributes Reference

The following attributes are exported:

- `id` - The unique identifier for the volume attachment.

» digitalocean_volume_snapshot

Provides a DigitalOcean Volume Snapshot which can be used to create a snapshot from an existing volume.

» Example Usage

```
resource "digitalocean_volume" "foobar" {
  region      = "nyc1"
  name        = "baz"
  size        = 100
  description = "an example volume"
}

resource "digitalocean_volume_snapshot" "foobar" {
  name      = "foo"
  volume_id = digitalocean_volume.foobar.id
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) A name for the volume snapshot.
- `volume_id` - (Required) The ID of the volume from which the volume snapshot originated.
- `tags` - (Optional) A list of the tags to be applied to this volume snapshot.

» Attributes Reference

The following attributes are exported:

- `id` The ID of the volume snapshot.
- `created_at` - The date and time the volume snapshot was created.
- `min_disk_size` - The minimum size in gigabytes required for a volume to be created based on this volume snapshot.
- `regions` - A list of DigitalOcean region ”slugs” indicating where the volume snapshot is available.
- `size` - The billable size of the volume snapshot in gigabytes.

» Import

Volume Snapshots can be imported using the `snapshot id`, e.g.

```
terraform import digitalocean_volume_snapshot.snapshot 506f78a4-e098-11e5-ad9f-000f53306ae1
```