Docker Provider

The Docker provider is used to interact with Docker containers and images. It uses the Docker API to manage the lifecycle of Docker containers. Because the Docker provider uses the Docker API, it is immediately compatible not only with single server Docker but Swarm and any additional Docker-compatible API hosts.

Use the navigation to the left to read about the available resources.

Example Usage

```
# Configure the Docker provider
provider "docker" {
  host = "tcp://127.0.0.1:2376/"
}

# Create a container
resource "docker_container" "foo" {
  image = "${docker_image.ubuntu.latest}"
  name  = "foo"
}

resource "docker_image" "ubuntu" {
  name = "ubuntu:latest"
}
```

Note You can also use the `ssh` protocol to connect to the docker host on a remote machine. The configuration would look as follows:

```
provider "docker" {
  host = "ssh://user@remote-host:22"
}
```

Registry Credentials

Registry credentials can be provided on a per-registry basis with the `registry_auth` field, passing either a config file or the username/password directly.

Note The location of the config file is on the machine terraform runs on, nevertheless if the specified docker host is on another machine.

```
provider "docker" {
  host = "tcp://localhost:2376"

  registry_auth {
    address = "registry.hub.docker.com"
    config_file = "${pathexpand("~/.docker/config.json")}"
  }

  registry_auth {
    address = "quay.io:8181"
    username = "someuser"
    password = "somepass"
  }
}

data "docker_registry_image" "quay" {
  name = "myorg/privateimage"
}

data "docker_registry_image" "quay" {
  name = "quay.io:8181/myorg/privateimage"
}
```

Note When passing in a config file either the corresponding `auth` string of the repository is read or the os specific credential helpers (see here (https://github.com/docker/docker-credential-helpers#available-programs)) are used to retrieve the authentication credentials.

You can still use the enviroment variables `DOCKER_REGISTRY_USER` and `DOCKER_REGISTRY_PASS` .

An example content of the file `~/.docker/config.json` on OSX may look like follows:

```
{
    "auths": {
        "repo.mycompany:8181": {
            "auth": "dXNlcjpwYXNz="
        },
        "otherrepo.other-company:8181": {

        }
    },
  "credsStore" : "osxkeychain"
}
```

Certificate information

Specify certificate information either with a directory or directly with the content of the files for connecting to the Docker host via TLS.

```
provider "docker" {
  host    = "tcp://your-host-ip:2376/"

  # -> specify either
  cert_path = "${pathexpand("~/.docker")}"

  # -> or the following
  ca_material = "${file(pathexpand("~/.docker/ca.pem"))}" # this can be omitted
  cert_material = "${file(pathexpand("~/.docker/cert.pem"))}"
  key_material = "${file(pathexpand("~/.docker/key.pem"))}"
}
```

Argument Reference

The following arguments are supported:

- `host` - (Required) This is the address to the Docker host. If this is blank, the `DOCKER_HOST` environment variable will also be read.
- `cert_path` - (Optional) Path to a directory with certificate information for connecting to the Docker host via TLS. It is expected that the 3 files `{ca, cert, key}.pem` are present in the path. If the path is blank, the `DOCKER_CERT_PATH` will also be checked.
- `ca_material`, `cert_material`, `key_material`, - (Optional) Content of `ca.pem`, `cert.pem`, and `key.pem` files for TLS authentication. Cannot be used together with `cert_path`. If `ca_material` is omitted the client does not check the servers certificate chain and host name.
- `registry_auth` - (Optional) A block specifying the credentials for a target v2 Docker registry.
  - `address` - (Required) The address of the registry.
  - `username` - (Optional) The username to use for authenticating to the registry. Cannot be used with the `config_file` option. If this is blank, the `DOCKER_REGISTRY_USER` will also be checked.
  - `password` - (Optional) The password to use for authenticating to the registry. Cannot be used with the `config_file` option. If this is blank, the `DOCKER_REGISTRY_PASS` will also be checked.
  - `config_file` - (Optional) The path to a config file containing credentials for authenticating to the registry. Cannot be used with the `username` / `password` options. If this is blank, the `DOCKER_CONFIG` will also be checked.

NOTE on Certificates and `docker-machine`: As per Docker Remote API documentation (https://docs.docker.com/engine/reference/api/docker_remote_api/), in any docker-machine environment, the Docker daemon uses an encrypted TCP socket (TLS) and requires `cert_path` for a successful connection. As an alternative, if using `docker-machine`, run `eval $(docker-machine env <machine-name>)` prior to running Terraform, and the host and certificate path will be extracted from the environment.

docker_network

Finds a specific docker network and returns information about it.

Example Usage

```
data "docker_network" "main" {
  name = "main"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Optional, string) The name of the Docker network.
- `id` - (Optional, string) The id of the Docker network.

Attributes Reference

The following attributes are exported in addition to the above configuration:

- `driver` - (Optional, string) The driver of the Docker network. Possible values are `bridge`, `host`, `overlay`, `macvlan`. See [docker docs][networkdocs] for more details.
- `options` - (Optional, map) Only available with bridge networks. See [docker docs][bridgeoptionsdocs] for more details.
- `internal` (Optional, bool) Boolean flag for whether the network is internal.
- `ipam_config` (Optional, map) See IPAM below for details.
- `scope` (Optional, string) Scope of the network. One of `swarm`, `global`, or `local`.

[networkdocs] https://docs.docker.com/network/#network-drivers (https://docs.docker.com/network/#network-drivers)
[bridgeoptionsdocs] https://docs.docker.com/engine/reference/commandline/network_create/#bridge-driver-options (https://docs.docker.com/engine/reference/commandline/network_create/#bridge-driver-options)

docker_registry_image

Reads the image metadata from a Docker Registry. Used in conjunction with the docker_image (/docs/providers/docker/r/image.html) resource to keep an image up to date on the latest available version of the tag.

Example Usage

```
data "docker_registry_image" "ubuntu" {
  name = "ubuntu:precise"
}

resource "docker_image" "ubuntu" {
  name          = "${data.docker_registry_image.ubuntu.name}"
  pull_triggers = ["${data.docker_registry_image.ubuntu.sha256_digest}"]
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required, string) The name of the Docker image, including any tags. e.g. `alpine:latest`

Attributes Reference

The following attributes are exported in addition to the above configuration:

- `sha256_digest` (string) - The content digest of the image, as stored on the registry.

docker_config

Manages the configuration of a Docker service in a swarm.

Example Usage

Basic

```
# Creates a config
resource "docker_config" "foo_config" {
  name = "foo_config"
  data = "ewogICJzZXJIfQo="
}
```

Advanced

Dynamically set config with a template

In this example you can use the `${var.foo_port}` variable to dynamically set the `${port}` variable in the `foo.configs.json.tpl` template and create the data of the `foo_config` with the help of the `base64encode` interpolation function.

File `foo.config.json.tpl`

```
{
  "server": {
    "public_port": ${port}
  }
}
```

File `main.tf`

```
# Creates the template in renders the variable
data "template_file" "foo_config_tpl" {
  template = "${file("foo.config.json.tpl")}"

  vars {
    port = "${var.foo_port}"
  }
}

# Creates the config
resource "docker_config" "foo_config" {
  name = "foo_config"
  data = "${base64encode(data.template_file.foo_config_tpl.rendered)}"
}
```

Update config with no downtime

To update a `config`, Terraform will destroy the existing resource and create a replacement. To effectively use a `docker_config` resource with a `docker_service` resource, it's recommended to specify `create_before_destroy` in a `lifecycle` block. Provide a uniqie `name` attribute, for example with one of the interpolation functions `uuid` or `timestamp` as shown in the example below. The reason is moby-35803 (https://github.com/moby/moby/issues/35803).

```
resource "docker_config" "service_config" {
  name = "${var.service_name}-config-${replace(timestamp(),":", ".")}"
  data = "${base64encode(data.template_file.service_config_tpl.rendered)}"

  lifecycle {
    ignore_changes        = ["name"]
    create_before_destroy = true
  }
}

resource "docker_service" "service" {
  # ...
  configs = [
    {
      config_id   = "${docker_config.service_config.id}"
      config_name = "${docker_config.service_config.name}"
      file_name   = "/root/configs/configs.json"
    },
  ]
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required, string) The name of the Docker config.
- `data` - (Required, string) The base64 encoded data of the config.

Attributes Reference

The following attributes are exported in addition to the above configuration:

- `id` (string)

docker_container

Manages the lifecycle of a Docker container.

Example Usage

```
# Start a container
resource "docker_container" "ubuntu" {
  name  = "foo"
  image = "${docker_image.ubuntu.latest}"
}

# Find the latest Ubuntu precise image.
resource "docker_image" "ubuntu" {
  name = "ubuntu:precise"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required, string) The name of the Docker container.
- `image` - (Required, string) The ID of the image to back this container. The easiest way to get this value is to use the `docker_image` resource as is shown in the example above.
- `command` - (Optional, list of strings) The command to use to start the container. For example, to run `/usr/bin/myprogram -f baz.conf` set the command to be `["/usr/bin/myprogram", "-f", "baz.conf"]`.
- `entrypoint` - (Optional, list of strings) The command to use as the Entrypoint for the container. The Entrypoint allows you to configure a container to run as an executable. For example, to run `/usr/bin/myprogram` when starting a container, set the entrypoint to be `["/usr/bin/myprogram"]`.
- `user` - (Optional, string) User used for run the first process. Format is `user` or `user:group` which user and group can be passed literraly or by name.
- `dns` - (Optional, set of strings) Set of DNS servers.
- `dns_opts` - (Optional, set of strings) Set of DNS options used by the DNS provider(s), see `resolv.conf` documentation for valid list of options.
- `dns_search` - (Optional, set of strings) Set of DNS search domains that are used when bare unqualified hostnames are used inside of the container.
- `env` - (Optional, set of strings) Environment variables to set.
- `labels` - (Optional, map of strings) Key/value pairs to set as labels on the container.
- `links` - (Optional, set of strings) Set of links for link based connectivity between containers that are running on the same host.

Warning The --link flag is a legacy feature of Docker. It may eventually be removed. It exposes *all* environment variables originating from Docker to any linked containers. This could have serious security implications if sensitive data is stored in them. See [the docker documentation][linkdoc] for more details.

- `hostname` - (Optional, string) Hostname of the container.
- `domainname` - (Optional, string) Domain name of the container.
- `restart` - (Optional, string) The restart policy for the container. Must be one of "no", "on-failure", "always", "unless-stopped".
- `max_retry_count` - (Optional, int) The maximum amount of times to an attempt a restart when `restart` is set to "on-failure"
- `working_dir` - (Optional, string) The working directory for commands to run in
- `rm` - (Optional, bool) If true, then the container will be automatically removed after his execution. Terraform won't check this container after creation.
- `start` - (Optional, bool) If true, then the Docker container will be started after creation. If false, then the container is only created.
- `attach` - (Optional, bool) If true attach to the container after its creation and waits the end of his execution.
- `logs` - (Optional, bool) Save the container logs (`attach` must be enabled).
- `must_run` - (Optional, bool) If true, then the Docker container will be kept running. If false, then as long as the container exists, Terraform assumes it is successful.
- `capabilities` - (Optional, block) See Capabilities below for details.
- `mounts` - (Optional, set of blocks) See Mounts below for details.
- `tmpfs` - (Optional, map) A map of container directories which should be replaced by `tmpfs mounts` , and their corresponding mount options.
- `ports` - (Optional, block) See Ports below for details.
- `host` - (Optional, block) See Extra Hosts below for details.
- `privileged` - (Optional, bool) Run container in privileged mode.
- `devices` - (Optional, bool) See Devices below for details.
- `publish_all_ports` - (Optional, bool) Publish all ports of the container.
- `volumes` - (Optional, block) See Volumes below for details.
- `memory` - (Optional, int) The memory limit for the container in MBs.
- `memory_swap` - (Optional, int) The total memory limit (memory + swap) for the container in MBs. This setting may compute to `-1` after `terraform apply` if the target host doesn't support memory swap, when that is the case docker will use a soft limitation.
- `shm_size` - (Optional, int) Size of `/dev/shm` in MBs.
- `cpu_shares` - (Optional, int) CPU shares (relative weight) for the container.

- `cpu_set` - (Optional, string) A comma-separated list or hyphen-separated range of CPUs a container can use, e.g. `0-1`.
- `log_driver` - (Optional, string) The logging driver to use for the container. Defaults to "json-file".
- `log_opts` - (Optional, map of strings) Key/value pairs to use as options for the logging driver.
- `network_alias` - (Optional, set of strings) Network aliases of the container for user-defined networks only. *Deprecated:* use `networks_advanced` instead.
- `network_mode` - (Optional, string) Network mode of the container.
- `networks` - (Optional, set of strings) Id of the networks in which the container is. *Deprecated:* use `networks_advanced` instead.
- `networks_advanced` - (Optional, block) See Networks Advanced below for details. If this block has priority to the deprecated `network_alias` and `network` properties.
- `destroy_grace_seconds` - (Optional, int) If defined will attempt to stop the container before destroying. Container will be destroyed after `n` seconds or on successful stop.
- `upload` - (Optional, block) See File Upload below for details.
- `ulimit` - (Optional, block) See Ulimits below for details.
- `pid_mode` - (Optional, string) The PID (Process) Namespace mode for the container. Either `container:<name|id>` or `host`.
- `userns_mode` - (Optional, string) Sets the usernamespace mode for the container when usernamespace remapping option is enabled.
- `healthcheck` - (Optional, block) See Healthcheck below for details.
- `sysctls` - (Optional, map) A map of kernel parameters (sysctls) to set in the container.
- `ipc_mode` - (Optional, string) IPC sharing mode for the container. Possible values are: `none`, `private`, `shareable`, `container:<name|id>` or `host`.
- `group_add` - (Optional, set of strings) Add additional groups to run as.

Capabilities

`capabilities` is a block within the configuration that allows you to add or drop linux capabilities. For more information about what capabilities you can add and drop please visit the docker run documentation.

- `add` - (Optional, set of strings) list of linux capabilities to add.
- `drop` - (Optional, set of strings) list of linux capabilities to drop.

Example:

```
resource "docker_container" "ubuntu" {
  name  = "foo"
  image = "${docker_image.ubuntu.latest}"

  capabilities {
    add  = ["ALL"]
    drop = ["SYS_ADMIN"]
  }
}
```

Mounts

`mounts` is a block within the configuration that can be repeated to specify the extra mount mappings for the container. Each `mounts` block is the Specification for mounts to be added to container and supports the following:

- `target` - (Required, string) The container path.
- `source` - (Optional, string) The mount source (e.g., a volume name, a host path)
- `type` - (Required, string) The mount type: valid values are `bind|volume|tmpfs`.
- `read_only` - (Optional, string) Whether the mount should be read-only
- `bind_options` - (Optional, map) Optional configuration for the `bind` type.
  - `propagation` - (Optional, string) A propagation mode with the value.
- `volume_options` - (Optional, map) Optional configuration for the `volume` type.
  - `no_copy` - (Optional, string) Whether to populate volume with data from the target.
  - `labels` - (Optional, map of key/value pairs) Adding labels.
  - `driver_options` - (Optional, map of key/value pairs) Options for the driver.
- `tmpfs_options` - (Optional, map) Optional configuration for the `tmpf` type.
  - `size_bytes` - (Optional, int) The size for the tmpfs mount in bytes.
  - `mode` - (Optional, int) The permission mode for the tmpfs mount in an integer.

Ports

`ports` is a block within the configuration that can be repeated to specify the port mappings of the container. Each `ports` block supports the following:

- `internal` - (Required, int) Port within the container.
- `external` - (Optional, int) Port exposed out of the container. If not given a free random port `>= 32768` will be used.
- `ip` - (Optional, string) IP address/mask that can access this port, default to `0.0.0.0`
- `protocol` - (Optional, string) Protocol that can be used over this port, defaults to `tcp`.

Extra Hosts

`host` is a block within the configuration that can be repeated to specify the extra host mappings for the container. Each `host` block supports the following:

- `host` - (Required, string) Hostname to add.
- `ip` - (Required, string) IP address this hostname should resolve to.

This is equivalent to using the `--add-host` option when using the `run` command of the Docker CLI.

Volumes

`volumes` is a block within the configuration that can be repeated to specify the volumes attached to a container. Each `volumes` block supports the following:

- `from_container` - (Optional, string) The container where the volume is coming from.
- `host_path` - (Optional, string) The path on the host where the volume is coming from.
- `volume_name` - (Optional, string) The name of the docker volume which should be mounted.
- `container_path` - (Optional, string) The path in the container where the volume will be mounted.
- `read_only` - (Optional, bool) If true, this volume will be readonly. Defaults to false.

One of `from_container`, `host_path` or `volume_name` must be set.

File Upload

`upload` is a block within the configuration that can be repeated to specify files to upload to the container before starting it. Only one of `content` or `content_base64` can be set and at least one of them hast to be set. Each `upload` supports the following

- `content` - (Optional, string, conflicts with `content_base64`) Literal string value to use as the object content, which will be uploaded as UTF-8-encoded text.
- `content_base64` - (Optional, string, conflicts with `content`) Base64-encoded data that will be decoded and uploaded as raw bytes for the object content. This allows safely uploading non-UTF8 binary data, but is recommended only for larger binary content such as the result of the `base64encode` interpolation function. See here (https://github.com/terraform-providers/terraform-provider-docker/issues/48#issuecomment-374174588) for the reason.
- `file` - (Required, string) path to a file in the container.
- `executable` - (Optional, bool) If true, the file will be uploaded with user executable permission. Defaults to false.

Network advanced

`networks_advanced` is a block within the configuration that can be repeated to specify advanced options for the container in a specific network. Each `networks_advanced` supports the following:

- `name` - (Required, string) The name of the network.
- `aliases` - (Optional, set of strings) The network aliases of the container in the specific network.

- `ipv4_address` - (Optional, string) The IPV4 address of the container in the specific network.
- `ipv6_address` - (Optional, string) The IPV6 address of the container in the specific network.

Devices

`devices` is a block within the configuration that can be repeated to specify the devices exposed to a container. Each `devices` block supports the following:

- `host_path` - (Required, string) The path on the host where the device is located.
- `container_path` - (Optional, string) The path in the container where the device will be binded.
- `permissions` - (Optional, string) The cgroup permissions given to the container to access the device. Defaults to `rwm`.

Ulimits

`ulimit` is a block within the configuration that can be repeated to specify the extra ulimits for the container. Each `ulimit` block supports the following:

- `name` - (Required, string)
- `soft` - (Required, int)
- `hard` - (Required, int)

Healthcheck

`healthcheck` is a block within the configuration that can be repeated only **once** to specify the extra healthcheck configuration for the container. The `healthcheck` block is a test to perform to check that the container is healthy and supports the following:

- `test` - (Required, list of strings) Command to run to check health. For example, to run `curl -f http://localhost/health` set the command to be `["CMD", "curl", "-f", "http://localhost/health"]`.
- `interval` - (Optional, string) Time between running the check `(ms|s|m|h)`. Default: `0s`.
- `timeout` - (Optional, string) Maximum time to allow one check to run `(ms|s|m|h)`. Default: `0s`.
- `start_period` - (Optional, string) Start period for the container to initialize before counting retries towards unstable `(ms|s|m|h)`. Default: `0s`.
- `retries` - (Optional, int) Consecutive failures needed to report unhealthy. Default: `0`.

Attributes Reference

The following attributes are exported:

- `exit_code` - The exit code of the container if its execution is done (`must_run` must be disabled).
- `container_logs` - The logs of the container if its execution is done (`attach` must be disabled).

- `network_data` - (Map of a block) The IP addresses of the container on each network. Key are the network names, values are the IP addresses.
    - `ip_address` - The IP address of the container.
    - `ip_prefix_length` - The IP prefix length of the container.
    - `gateway` - The network gateway of the container.
- `bridge` - The network bridge of the container as read from its NetworkSettings.
- `ip_address` - *Deprecated:* Use `network_data` instead. The IP address of the container's first network it.
- `ip_prefix_length` - *Deprecated:* Use `network_data` instead. The IP prefix length of the container as read from its NetworkSettings.
- `gateway` - *Deprecated:* Use `network_data` instead. The network gateway of the container as read from its NetworkSettings.

[linkdoc] https://docs.docker.com/network/links/ (https://docs.docker.com/network/links/)

docker_image

Pulls a Docker image to a given Docker host from a Docker Registry.

This resource will *not* pull new layers of the image automatically unless used in conjunction with `docker_registry_image` (/docs/providers/docker/d/registry_image.html) data source to update the `pull_triggers` field.

Example Usage

```
# Find the latest Ubuntu precise image.
resource "docker_image" "ubuntu" {
  name = "ubuntu:precise"
}

# Access it somewhere else with ${docker_image.ubuntu.latest}
```

Dynamic image

```
data "docker_registry_image" "ubuntu" {
  name = "ubuntu:precise"
}

resource "docker_image" "ubuntu" {
  name          = "${data.docker_registry_image.ubuntu.name}"
  pull_triggers = ["${data.docker_registry_image.ubuntu.sha256_digest}"]
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required, string) The name of the Docker image, including any tags or SHA256 repo digests.
- `keep_locally` - (Optional, boolean) If true, then the Docker image won't be deleted on destroy operation. If this is false, it will delete the image from the docker local storage on destroy operation.
- `pull_triggers` - (Optional, list of strings) List of values which cause an image pull when changed. This is used to store the image digest from the registry when using the `docker_registry_image` data source (/docs/providers/docker/d/registry_image.html) to trigger an image update.
- `pull_trigger` - **Deprecated**, use `pull_triggers` instead.

Attributes Reference

The following attributes are exported in addition to the above configuration:

- `latest` (string) - The ID of the image.

docker_network

Manages a Docker Network. This can be used alongside docker_container (/docs/providers/docker/r/container.html) to create virtual networks within the docker environment.

Example Usage

```
# Create a new docker network
resource "docker_network" "private_network" {
  name = "my_network"
}

# Access it somewhere else with ${docker_network.private_network.name}
```

Argument Reference

The following arguments are supported:

- `name` - (Required, string) The name of the Docker network.
- `labels` - (Optional, map of string/string key/value pairs) User-defined key/value metadata.
- `check_duplicate` - (Optional, boolean) Requests daemon to check for networks with same name.
- `driver` - (Optional, string) Name of the network driver to use. Defaults to `bridge` driver.
- `options` - (Optional, map of strings) Network specific options to be used by the drivers.
- `internal` - (Optional, boolean) Restrict external access to the network. Defaults to `false` .
- `attachable` - (Optional, boolean) Enable manual container attachment to the network. Defaults to `false` .
- `ingress` - (Optional, boolean) Create swarm routing-mesh network. Defaults to `false` .
- `ipv6` - (Optional, boolean) Enable IPv6 networking. Defaults to `false` .
- `ipam_driver` - (Optional, string) Driver used by the custom IP scheme of the network.
- `ipam_config` - (Optional, block) See IPAM config below for details.

IPAM config

Configuration of the custom IP scheme of the network.

The `ipam_config` block supports:

- `subnet` - (Optional, string)
- `ip_range` - (Optional, string)

- `gateway` - (Optional, string)
- `aux_address` - (Optional, map of string)

Attributes Reference

The following attributes are exported in addition to the above configuration:

- `id` (string)
- `scope` (string)

docker_secret

Manages the secrets of a Docker service in a swarm.

Example Usage

Basic

```
# Creates a secret
resource "docker_secret" "foo_secret" {
  name = "foo_secret"
  data = "ewogICJzZXJsaasIfQo="
}
```

Update secret with no downtime

To update a `secret`, Terraform will destroy the existing resource and create a replacement. To effectively use a `docker_secret` resource with a `docker_service` resource, it's recommended to specify `create_before_destroy` in a `lifecycle` block. Provide a unique `name` attribute, for example with one of the interpolation functions `uuid` or `timestamp` as shown in the example below. The reason is moby-35803 (https://github.com/moby/moby/issues/35803).

```
resource "docker_secret" "service_secret" {
  name = "${var.service_name}-secret-${replace(timestamp(),":", ".")}"
  data = "${base64encode(data.template_file.service_secret_tpl.rendered)}"

  lifecycle {
    ignore_changes        = ["name"]
    create_before_destroy = true
  }
}

resource "docker_service" "service" {
  # ...
  secrets = [
    {
      secret_id   = "${docker_secret.service_secret.id}"
      secret_name = "${docker_secret.service_secret.name}"
      file_name   = "/root/configs/configs.json"
    },
  ]
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required, string) The name of the Docker secret.
- `data` - (Required, string) The base64 encoded data of the secret.
- `labels` - (Optional, map of string/string key/value pairs) User-defined key/value metadata.

Attributes Reference

The following attributes are exported in addition to the above configuration:

- `id` (string)

docker_service

This resource manages the lifecycle of a Docker service. By default, the creation, update and delete of services are detached.

With the Converge Config the behavior of the `docker cli` is imitated to guarantee that for example, all tasks of a service are running or successfully updated or to inform `terraform` that a service could not be updated and was successfully rolled back.

Example Usage

The following examples show the basic and advanced usage of the Docker Service resource assuming the host machine is already part of a Swarm.

Basic

The following configuration starts a Docker Service with - the given image, - 1 replica - exposes the port `8080` in `vip` mode to the host machine - moreover, uses the `container` runtime

```
resource "docker_service" "foo" {
  name = "foo-service"

  task_spec {
    container_spec {
      image = "repo.mycompany.com:8080/foo-service:v1"
    }
  }

  endpoint_spec {
    ports {
      target_port = "8080"
    }
  }
}
```

The following command is the equivalent:

```
$ docker service create -d -p 8080 --name foo-service repo.mycompany.com:8080/foo-service:v1
```

Advanced

The following configuration shows the full capabilities of a Docker Service. Currently, the Docker API 1.32 (https://docs.docker.com/engine/api/v1.32) is implemented.

```
resource "docker_volume" "test_volume" {
  name = "tftest-volume"
}
```

```
resource "docker_config" "service_config" {
  name = "tftest-full-myconfig"
  data = "ewogICJwcmVmaXgiOiAiMTIzIgp9"
}

resource "docker_secret" "service_secret" {
  name = "tftest-mysecret"
  data = "ewogICJrZXkiOiAiUVdFUlRZIgp9"
}

resource "docker_network" "test_network" {
  name   = "tftest-network"
  driver = "overlay"
}

resource "docker_service" "foo" {
  name = "tftest-service-basic"

  task_spec {
    container_spec {
      image = "repo.mycompany.com:8080/foo-service:v1"

      labels {
        foo = "bar"
      }

      command  = ["ls"]
      args     = ["-las"]
      hostname = "my-fancy-service"

      env {
        MYFOO = "BAR"
      }

      dir    = "/root"
      user   = "root"
      groups = ["docker", "foogroup"]

      privileges {
        se_linux_context {
          disable = true
          user    = "user-label"
          role    = "role-label"
          type    = "type-label"
          level   = "level-label"
        }
      }

      read_only = true

      mounts = [
        {
          target    = "/mount/test"
          source    = "${docker_volume.test_volume.name}"
          type      = "volume"
          read_only = true

          bind_options {
            propagation = "private"
          }
```

```
      }
    ]

    stop_signal       = "SIGTERM"
    stop_grace_period = "10s"

    healthcheck {
      test     = ["CMD", "curl", "-f", "http://localhost:8080/health"]
      interval = "5s"
      timeout  = "2s"
      retries  = 4
    }

    hosts {
      host = "testhost"
      ip   = "10.0.1.0"
    }

    dns_config {
      nameservers = ["8.8.8.8"]
      search      = ["example.org"]
      options     = ["timeout:3"]
    }

    secrets = [
      {
        secret_id   = "${docker_secret.service_secret.id}"
        secret_name = "${docker_secret.service_secret.name}"
        file_name   = "/secrets.json"
      },
    ]

    configs = [
      {
        config_id   = "${docker_config.service_config.id}"
        config_name = "${docker_config.service_config.name}"
        file_name   = "/configs.json"
      },
    ]
  }

  resources {
    limits {
      nano_cpus    = 1000000
      memory_bytes = 536870912

      generic_resources {
        named_resources_spec = [
          "GPU=UUID1",
        ]

        discrete_resources_spec = [
          "SSD=3",
        ]
      }
    }

    reservation {
      nano_cpus    = 1000000
```

```
      memory_bytes = 536870912

      generic_resources {
        named_resources_spec = [
          "GPU=UUID1",
        ]

        discrete_resources_spec = [
          "SSD=3",
        ]
      }
    }
  }

  restart_policy {
    condition    = "on-failure"
    delay        = "3s"
    max_attempts = 4
    window       = "10s"
  }

  placement {
    constraints = [
      "node.role==manager",
    ]

    prefs = [
      "spread=node.role.manager",
    ]
  }

  force_update = 0
  runtime      = "container"
  networks     = ["${docker_network.test_network.id}"]

  log_driver {
    name = "json-file"

    options {
      max-size = "10m"
      max-file = "3"
    }
  }
}

mode {
  replicated {
    replicas = 2
  }
}

update_config {
  parallelism       = 2
  delay             = "10s"
  failure_action    = "pause"
  monitor           = "5s"
  max_failure_ratio = "0.1"
  order             = "start-first"
}
```

```
  rollback_config {
    parallelism       = 2
    delay             = "5ms"
    failure_action    = "pause"
    monitor           = "10h"
    max_failure_ratio = "0.9"
    order             = "stop-first"
  }

  endpoint_spec {
    mode = "vip"

    ports {
      name           = "random"
      protocol       = "tcp"
      target_port    = "8080"
      published_port = "8080"
      publish_mode   = "ingress"
    }
  }
}
```

See also the `TestAccDockerService_full` test or all the other tests for a complete overview.

Argument Reference

The following arguments are supported:

- `auth` - (Optional, block) See Auth below for details.
- `name` - (Required, string) The name of the Docker service.
- `task_spec` - (Required, block) See TaskSpec below for details.
- `mode` - (Optional, block) See Mode below for details.
- `update_config` - (Optional, block) See UpdateConfig below for details.
- `rollback_config` - (Optional, block) See RollbackConfig below for details.
- `endpoint_spec` - (Optional, block) See EndpointSpec below for details.
- `converge_config` - (Optional, block) See Converge Config below for details.

Auth

`auth` can be used additionally to the `registry_auth`. If both properties are given the `auth` wins and overwrites the auth of the provider.

- `server_address` - (Required, string) The address of the registry server
- `username` - (Optional, string) The username to use for authenticating to the registry. If this is blank, the `DOCKER_REGISTRY_USER` is also be checked.

- `password` - (Optional, string) The password to use for authenticating to the registry. If this is blank, the `DOCKER_REGISTRY_PASS` is also be checked.

TaskSpec

`task_spec` is a block within the configuration that can be repeated only **once** to specify the mode configuration for the service. The `task_spec` block is the user modifiable task configuration and supports the following:

- `container_spec` (Required, block) See ContainerSpec below for details.
- `resources` (Optional, block) See Resources below for details.
- `restart_policy` (Optional, block) See Restart Policy below for details.
- `placement` (Optional, block) See Placement below for details.
- `force_update` (Optional, int) A counter that triggers an update even if no relevant parameters have been changed. See Docker Spec (https://github.com/docker/swarmkit/blob/master/api/specs.proto#L126).
- `runtime` (Optional, string) Runtime is the type of runtime specified for the task executor. See Docker Runtime (https://github.com/moby/moby/blob/master/api/types/swarm/runtime.go).
- `networks` - (Optional, set of strings) Ids of the networks in which the container will be put in.
- `log_driver` - (Optional, block) See Log Driver below for details.

ContainerSpec

`container_spec` is a block within the configuration that can be repeated only **once** to specify the mode configuration for the service. The `container_spec` block is the spec for each container and supports the following:

- `image` - (Required, string) The image used to create the Docker service.
- `labels` - (Optional, map of string/string key/value pairs) User-defined key/value metadata.
- `command` - (Optional, list of strings) The command to be run in the image.
- `args` - (Optional, list of strings) Arguments to the command.
- `hostname` - (Optional, string) The hostname to use for the container, as a valid RFC 1123 hostname.
- `env` - (Optional, map of string/string) A list of environment variables in the form VAR=value.
- `dir` - (Optional, string) The working directory for commands to run in.
- `user` - (Optional, string) The user inside the container.
- `groups` - (Optional, list of strings) A list of additional groups that the container process will run as.
- `privileges` (Optional, block) See Privileges below for details.
- `read_only` - (Optional, bool) Mount the container's root filesystem as read only.
- `mounts` - (Optional, set of blocks) See Mounts below for details.
- `stop_signal` - (Optional, string) Signal to stop the container.

- `stop_grace_period` - (Optional, string) Amount of time to wait for the container to terminate before forcefully removing it `(ms|s|m|h)`.
- `healthcheck` - (Optional, block) See Healthcheck below for details.
- `host` - (Optional, map of string/string) A list of hostname/IP mappings to add to the container's hosts file.
  - `ip` - (Required string) The ip
  - `host` - (Required string) The hostname
- `dns_config` - (Optional, block) See DNS Config below for details.
- `secrets` - (Optional, set of blocks) See Secrets below for details.
- `configs` - (Optional, set of blocks) See Configs below for details.
- `isolation` - (Optional, string) Isolation technology of the containers running the service. (Windows only). Valid values are: `default|process|hyperv`

Privileges

`privileges` is a block within the configuration that can be repeated only **once** to specify the mode configuration for the service. The `privileges` block holds the security options for the container and supports the following:

- `credential_spec` - (Optional, block) For managed service account (Windows only)
  - `file` - (Optional, string) Load credential spec from this file.
  - `registry` - (Optional, string) Load credential spec from this value in the Windows registry.
- `se_linux_context` - (Optional, block) SELinux labels of the container
  - `disable` - (Optional, bool) Disable SELinux
  - `user` - (Optional, string) SELinux user label
  - `role` - (Optional, string) SELinux role label
  - `type` - (Optional, string) SELinux type label
  - `level` - (Optional, string) SELinux level label

Mounts

`mount` is a block within the configuration that can be repeated to specify the extra mount mappings for the container. Each `mount` block is the Specification for mounts to be added to containers created as part of the service and supports the following:

- `target` - (Required, string) The container path.
- `source` - (Optional, string) The mount source (e.g., a volume name, a host path)
- `type` - (Required, string) The mount type: valid values are `bind|volume|tmpfs`.
- `read_only` - (Optional, string) Whether the mount should be read-only
- `bind_options` - (Optional, map) Optional configuration for the `bind` type.
  - `propagation` - (Optional, string) A propagation mode with the value.

- `volume_options` - (Optional, map) Optional configuration for the `volume` type.
  - `no_copy` - (Optional, string) Whether to populate volume with data from the target.
  - `labels` - (Optional, map of key/value pairs) Adding labels.
  - `driver_config` - (Optional, map) The name of the driver to create the volume.
  - `name` - (Optional, string) The name of the driver to create the volume.
  - `options` - (Optional, map of key/value pairs) Options for the driver.
- `tmpfs_options` - (Optional, map) Optional configuration for the `tmpf` type.
  - `size_bytes` - (Optional, int) The size for the tmpfs mount in bytes.
  - `mode` - (Optional, int) The permission mode for the tmpfs mount in an integer.

Healthcheck

`healthcheck` is a block within the configuration that can be repeated only **once** to specify the extra healthcheck configuration for the containers of the service. The `healthcheck` block is a test to perform to check that the container is healthy and supports the following:

- `test` - (Required, list of strings) Command to run to check health. For example, to run `curl -f http://localhost/health` set the command to be `["CMD", "curl", "-f", "http://localhost/health"]`.
- `interval` - (Optional, string) Time between running the check `(ms|s|m|h)`. Default: `0s`.
- `timeout` - (Optional, string) Maximum time to allow one check to run `(ms|s|m|h)`. Default: `0s`.
- `start_period` - (Optional, string) Start period for the container to initialize before counting retries towards unstable `(ms|s|m|h)`. Default: `0s`.
- `retries` - (Optional, int) Consecutive failures needed to report unhealthy. Default: `0`.

DNS Config

`dns_config` is a block within the configuration that can be repeated only **once** to specify the extra DNS configuration for the containers of the service. The `dns_config` block supports the following:

- `nameservers` - (Required, list of strings) The IP addresses of the name servers, for example, `8.8.8.8`
- `search` - (Optional, list of strings)A search list for host-name lookup.
- `options` - (Optional, list of strings) A list of internal resolver variables to be modified, for example, `debug`, `ndots:3`

Secrets

`secrets` is a block within the configuration that can be repeated to specify the extra mount mappings for the container. Each `secrets` block is a reference to a secret that will be exposed to the service and supports the following:

- `secret_id` - (Required, string) ConfigID represents the ID of the specific secret.
- `secret_name` - (Optional, string) The name of the secret that this references, but internally it is just provided for

lookup/display purposes

- `file_name` - (Required, string) Represents the final filename in the filesystem. The specific target file that the secret data is written within the docker container, e.g. `/root/secret/secret.json`

Configs

`configs` is a block within the configuration that can be repeated to specify the extra mount mappings for the container. Each `configs` is a reference to a secret that is exposed to the service and supports the following:

- `config_id` - (Required, string) ConfigID represents the ID of the specific config.
- `config_name` - (Optional, string) The name of the config that this references, but internally it is just provided for lookup/display purposes
- `file_name` - (Required, string) Represents the final filename in the filesystem. The specific target file that the config data is written within the docker container, e.g. `/root/config/config.json`

Resources

`resources` is a block within the configuration that can be repeated only **once** to specify the mode configuration for the service. The `resources` block represents the requirements which apply to each container created as part of the service and supports the following:

- `limits` - (Optional, list of strings) Describes the resources which can be advertised by a node and requested by a task.
  - `nano_cpus` (Optional, int) CPU shares in units of 1/1e9 (or 10^{-9}) of the CPU. Should be at least 1000000
  - `memory_bytes` (Optional, int) The amount of memory in bytes the container allocates
  - `generic_resources` (Optional, map) User-defined resources can be either Integer resources (e.g, SSD=3) or String resources (e.g, GPU=UUID1)
  - `named_resources_spec` (Optional, set of string) The String resources, delimited by `=`
  - `discrete_resources_spec` (Optional, set of string) The Integer resources, delimited by `=`
- `reservation` - (Optional, list of strings) An object describing the resources which can be advertised by a node and requested by a task.
  - `nano_cpus` (Optional, int) CPU shares in units of 1/1e9 (or 10^{-9}) of the CPU. Should be at least 1000000
  - `memory_bytes` (Optional, int) The amount of memory in bytes the container allocates
  - `generic_resources` (Optional, map) User-defined resources can be either Integer resources (e.g, SSD=3) or String resources (e.g, GPU=UUID1)
  - `named_resources_spec` (Optional, set of string) The String resources
  - `discrete_resources_spec` (Optional, set of string) The Integer resources

Restart Policy

`restart_policy` is a block within the configuration that can be repeated only **once** to specify the mode configuration for the service. The `restart_policy` block specifies the restart policy which applies to containers created as part of this service and supports the following:

- `condition` (Optional, string) Condition for restart: `(none|on-failure|any)`
- `delay` (Optional, string) Delay between restart attempts `(ms|s|m|h)`
- `max_attempts` (Optional, string) Maximum attempts to restart a given container before giving up (default value is `0`, which is ignored)
- `window` (Optional, string) The time window used to evaluate the restart policy (default value is `0`, which is unbounded) `(ms|s|m|h)`

Placement

`placement` is a block within the configuration that can be repeated only **once** to specify the mode configuration for the service. The `placement` block specifies the placement preferences and supports the following:

- `constraints` (Optional, set of strings) An array of constraints. e.g.: `node.role==manager`
- `prefs` (Optional, set of string) Preferences provide a way to make the scheduler aware of factors such as topology. They are provided in order from highest to lowest precedence, e.g.: `spread=node.role.manager`
- `platforms` (Optional, set of) Platforms stores all the platforms that the service's image can run on
  - `architecture` (Required, string) The architecture, e.g., `amd64`
  - `os` (Required, string) The operation system, e.g., `linux`

Log Driver

`log_driver` is a block within the configuration that can be repeated only **once** to specify the extra log_driver configuration for the containers of the service. The `log_driver` specifies the log driver to use for tasks created from this spec. If not present, the default one for the swarm will be used, finally falling back to the engine default if not specified. The block supports the following:

- `name` - (Required, string) The logging driver to use. Either `(none|json-file|syslog|journald|gelf|fluentd|awslogs|splunk|etwlogs|gcplogs)`.
- `options` - (Optional, a map of strings and strings) The options for the logging driver, e.g.

```
options {
  awslogs-region = "us-west-2"
  awslogs-group  = "dev/foo-service"
}
```

Mode

`mode` is a block within the configuration that can be repeated only **once** to specify the mode configuration for the service. The `mode` block supports the following:

- `global` - (Optional, bool) set it to `true` to run the service in the global mode

```
resource "docker_service" "foo" {
  ...
  mode {
    global = true
  }
  ...
}
```

- `replicated` - (Optional, map), which contains atm only the amount of `replicas`

```
resource "docker_service" "foo" {
  ...
  mode {
    replicated {
      replicas = 2
    }
  }
  ...
}
```

NOTE on `mode` : if neither `global` nor `replicated` is specified, the service is started in `replicated` mode with 1 replica. A change of service mode is not possible. The service has to be destroyed an recreated in the new mode.

UpdateConfig and RollbackConfig

`update_config` or `rollback_config` is a block within the configuration that can be repeated only **once** to specify the extra update configuration for the containers of the service. The `update_config` `rollback_config` block supports the following:

- `parallelism` - (Optional, int) The maximum number of tasks to be updated in one iteration simultaneously (0 to update all at once).
- `delay` - (Optional, int) Delay between updates `(ns|us|ms|s|m|h)`, e.g. `5s` .
- `failure_action` - (Optional, int) Action on update failure: `pause|continue|rollback` .
- `monitor` - (Optional, int) Duration after each task update to monitor for failure `(ns|us|ms|s|m|h)`
- `max_failure_ratio` - (Optional, string) The failure rate to tolerate during an update as `float` . **Important:** the `float` need to be wrapped in a `string` to avoid internal casting and precision errors.
- `order` - (Optional, int) Update order either 'stop-first' or 'start-first'.

EndpointSpec

`endpoint_spec` is a block within the configuration that can be repeated only **once** to specify properties that can be configured to access and load balance a service. The block supports the following:

- `mode` - (Optional, string) The mode of resolution to use for internal load balancing between tasks. `(vip|dnsrr)`. Default: `vip`.
- `ports` - (Optional, block) See Ports below for details.

Ports

`ports` is a block within the configuration that can be repeated to specify the port mappings of the container. Each `ports` block supports the following:

- `name` - (Optional, string) A random name for the port.
- `protocol` - (Optional, string) Protocol that can be used over this port: `tcp|udp|sctp`. Default: `tcp`.
- `target_port` - (Required, int) Port inside the container.
- `published_port` - (Required, int) The port on the swarm hosts. If not set the value of `target_port` will be used.
- `publish_mode` - (Optional, string) Represents the mode in which the port is to be published: `ingress|host`

Converge Config

`converge_config` is a block within the configuration that can be repeated only **once** to specify the extra Converging configuration for the containers of the service. This is the same behavior as the `docker cli`. By adding this configuration, it is monitored with the given interval that, e.g., all tasks/replicas of a service are up and healthy

The `converge_config` block supports the following:

- `delay` - (Optional, string) Time between each the check to check docker endpoint `(ms|s|m|h)`. For example, to check if all tasks are up when a service is created, or to check if all tasks are successfully updated on an update. Default: `7s`.
- `timeout` - (Optional, string) The timeout of the service to reach the desired state `(s|m)`. Default: `3m`.

Attributes Reference

The following attributes are exported in addition to the above configuration:

- `id` (string)

docker_volume

Creates and destroys a volume in Docker. This can be used alongside docker_container (/docs/providers/docker/r/container.html) to prepare volumes that can be shared across containers.

Example Usage

```
# Creates a docker volume "shared_volume".
resource "docker_volume" "shared_volume" {
  name = "shared_volume"
}

# Reference the volume with ${docker_volume.shared_volume.name}
```

Argument Reference

The following arguments are supported:

- `name` - (Optional, string) The name of the Docker volume (generated if not provided).
- `labels` - (Optional, map of string/string key/value pairs) User-defined key/value metadata.
- `driver` - (Optional, string) Driver type for the volume (defaults to local).
- `driver_opts` - (Optional, map of strings) Options specific to the driver.

Attributes Reference

The following attributes are exported in addition to the above configuration:

- `mountpoint` (string) - The mountpoint of the volume.