Rancher Provider

The Rancher provider is used to interact with the resources supported by Rancher. The provider needs to be configured with the URL of the Rancher server at minimum and API credentials if access control is enabled on the server.

Example Usage

```
# Configure the Rancher provider
provider "rancher" {
  api_url    = "http://rancher.my-domain.com:8080"
  access_key = "${var.rancher_access_key}"
  secret_key = "${var.rancher_secret_key}"
}
```

Argument Reference

The following arguments are supported:

- `api_url` - (Required) Rancher API url. It must be provided, but it can also be sourced from the `RANCHER_URL` environment variable.
- `access_key` - (Optional) Rancher API access key. It can also be sourced from the `RANCHER_ACCESS_KEY` environment variable.
- `secret_key` - (Optional) Rancher API access key. It can also be sourced from the `RANCHER_SECRET_KEY` environment variable.

rancher_certificate

Use this data source to retrieve information about a Rancher certificate.

Example Usage

Simple datasource declaration

```
data "rancher_certificate" "foo" {
  name           = "foo"
  environment_id = "1a5"
}
```

Let's encrypt with DNS challenge

This setup will ensure that the Load Balancer stack is not created before the Let's Encrypt's certificate is actually present in Rancher's certificates manager.

```
locals {
  environment_id = "1a5"
}

resource "rancher_stack" "letsencrypt" {
  name            = "letsencrypt"
  environment_id  = "${local.environment_id}"
  catalog_id      = "community:letsencrypt:4"

  environment {
    CERT_NAME      = "letsencrypt"
    DOMAINS        = "foo.example.com"
    PROVIDER       = "Route53"
    AWS_ACCESS_KEY = "${var.aws_access_key}"
    AWS_SECRET_KEY = "${var.aws_secret_key}"
    ...
  }
}

data "rancher_certificate" "letsencrypt" {
  environment_id = "${local.environment_id}"
  name           = "${rancher_stack.letsencrypt.environment["CERT_NAME"]}"
}

resource "rancher_stack" "lb" {
  name           = "lb"
  environment_id = "${local.environment_id}"

  docker_compose = <<EOF
version: '2'
services:
  lb:
    image: rancher/lb-service-haproxy:v0.7.9
    ports:
    - 443:443/tcp
    labels:
      io.rancher.container.agent.role: environmentAdmin
      io.rancher.container.create_agent: 'true'
EOF

  rancher_compose = <<EOF
version: '2'
services:
  lb:
    scale: 1
    start_on_create: true
    lb_config:
      certs: []
      default_cert: ${data.rancher_certificate.letsencrypt.name}
      port_rules:
      - protocol: https
        service: mystack/myservice
        source_port: 443
        target_port: 80
    health_check:
      healthy_threshold: 2
      response_timeout: 2000
      port: 42
      unhealthy_threshold: 3
      interval: 2000
      strategy: recreate
EOF
}
```

Let's encrypt with HTTP challenge

This setup will ensure that the HTTPS Load Balancer stack is not created before the Let's Encrypt's certificate is actually present in Rancher's certificates manager.

```
locals {
  environment_id = "1a5"
}

resource "rancher_stack" "letsencrypt" {
  name            = "letsencrypt"
  environment_id  = "${local.environment_id}"
  catalog_id      = "community:letsencrypt:4"

  environment {
    CERT_NAME      = "letsencrypt"
    DOMAINS        = "foo.example.com"
    PROVIDER       = "HTTP"
    ...
  }
}

resource "rancher_stack" "lb-http" {
  name           = "lb-http"
  environment_id = "${local.environment_id}"

  docker_compose = <<EOF
version: '2'
services:
  lb:
    image: rancher/lb-service-haproxy:v0.7.9
    ports:
    - 80:80/tcp
    labels:
      io.rancher.container.agent.role: environmentAdmin
      io.rancher.container.create_agent: 'true'
EOF

  rancher_compose = <<EOF
version: '2'
services:
  lb:
    scale: 1
    start_on_create: true
    lb_config:
      certs: []
      - hostname: ''
        path: /.well-known/acme-challenge
        priority: 1
        protocol: http
        service: letsencrypt/letsencrypt
        source_port: 80
        target_port: 80
    health_check:
      healthy_threshold: 2
      response_timeout: 2000
      port: 42
      unhealthy_threshold: 3
      interval: 2000
      strategy: recreate
EOF
}

data "rancher_certificate" "letsencrypt" {
```

```
data "rancher_certificate" "letsencrypt" {
  environment_id = "${local.environment_id}"
  name           = "${rancher_stack.letsencrypt.environment["CERT_NAME"]}"
}

resource "rancher_stack" "lb-https" {
  name           = "lb-https"
  environment_id = "${local.environment_id}"

  docker_compose = <<EOF
version: '2'
services:
  lb:
    image: rancher/lb-service-haproxy:v0.7.9
    ports:
    - 443:443/tcp
    labels:
      io.rancher.container.agent.role: environmentAdmin
      io.rancher.container.create_agent: 'true'
EOF

  rancher_compose = <<EOF
version: '2'
services:
  lb:
    scale: 1
    start_on_create: true
    lb_config:
      certs: []
      default_cert: ${data.rancher_certificate.letsencrypt.name}
      port_rules:
      - protocol: https
        service: mystack/myservice
        source_port: 443
        target_port: 80
    health_check:
      healthy_threshold: 2
      response_timeout: 2000
      port: 42
      unhealthy_threshold: 3
      interval: 2000
      strategy: recreate
EOF
}
```

Argument Reference

- `name` - (Required) The setting name.
- `environment_id` - (Required) The ID of the environment.

Attributes Reference

- `id` - The ID of the resource.
- `cn` - The certificate CN.
- `algorithm` - The certificate algorithm.

- `cert_fingerprint` - The certificate fingerprint.
- `expires_at` - The certificate expiration date.
- `issued_at` - The certificate creation date.
- `issuer` - The certificate issuer.
- `serial_number` - The certificate serial number.
- `subject_alternative_names` - The list of certificate Subject Alternative Names.
- `version` - The certificate version.

rancher_environment

Use this data source to retrieve information about a Rancher environment.

Example Usage

```
data "rancher_environment" "foo" {
  name = "foo"
}
```

Argument Reference

- `name` - (Required) The setting name.

Attributes Reference

- `id` - The ID of the resource.
- `description` - The environment description.
- `orchestration` - The environment orchestration engine.
- `project_template_id` - The environment project template ID.
- `member` - The environment members.

rancher_setting

Use this data source to retrieve information about a Rancher setting.

Example Usage

```
data "rancher_setting" "cattle.cattle.version" {
  name = "cattle.cattle.version"
}
```

Argument Reference

- `name` - (Required) The setting name.

Attributes Reference

- `value` - the settting's value.

rancher_certificate

Provides a Rancher Certificate resource. This can be used to create certificates for rancher environments and retrieve their information.

Example Usage

```
# Create a new Rancher Certificate
resource rancher_certificate "foo" {
  name           = "foo"
  description    = "my foo certificate"
  environment_id = "${rancher_environment.test.id}"
  cert = "${file("server.crt")}"
  key = "${file("server.key")}"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the certificate.
- `description` - (Optional) A certificate description.
- `environment_id` - (Required) The ID of the environment to create the certificate for.
- `cert` - (Required) The certificate content.
- `cert_chain` - (Optional) The certificate chain.
- `key` - (Required) The certificate key.

Attributes Reference

The following attributes are exported:

- `id` - (Computed) The ID of the resource.
- `cn` - The certificate CN.
- `algorithm` - The certificate algorithm.
- `cert_fingerprint` - The certificate fingerprint.
- `expires_at` - The certificate expiration date.
- `issued_at` - The certificate creation date.
- `issuer` - The certificate issuer.
- `key_size` - The certificate key size.

- `serial_number` - The certificate serial number.
- `subject_alternative_names` - The list of certificate Subject Alternative Names.
- `version` - The certificate version.

Import

Certificates can be imported using the Certificate ID in the format `<environment_id>/<certificate_id>`

```
$ terraform import rancher_certificate.mycert 1a5/1c605
```

If the credentials for the Rancher provider have access to the global API, then `environment_id` can be omitted e.g.

```
$ terraform import rancher_certificate.mycert 1c605
```

rancher_environment

Provides a Rancher Environment resource. This can be used to create and manage environments on rancher.

Example Usage

```
# Create a new Rancher environment
resource "rancher_environment" "default" {
  name = "staging"
  description = "The staging environment"
  orchestration = "cattle"

  member {
    external_id = "650430"
    external_id_type = "github_user"
    role = "owner"
  }

  member {
    external_id = "1234"
    external_id_type = "github_team"
    role = "member"
  }
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the environment.
- `description` - (Optional) An environment description.
- `orchestration` - (Optional) Must be one of **cattle**, **swarm**, **mesos**, **windows** or **kubernetes**. This is a helper for setting the project_template_ids for the included Rancher templates. This will conflict with project_template_id setting. Changing this forces a new resource to be created.
- `project_template_id` - (Optional) This can be any valid project template ID. If this is set, then orchestration can not be. Changing this forces a new resource to be created.
- `member` - (Optional) Members to add to the environment.

Member Parameters Reference

A `member` takes three parameters:

- `external_id` - (Required) The external ID of the member.
- `external_id_type` - (Required) The external ID type of the member.
- `role` - (Required) The role of the member in the environment.

Attributes Reference

- `id` - The ID of the environment (ie `1a11`) that can be used in other Terraform resources such as Rancher Stack definitions.

Import

Environments can be imported using their Rancher API ID, e.g.

```
$ terraform import rancher_environment.dev 1a15
```

rancher_host

Provides a Rancher Host resource. This can be used to manage and delete hosts on Rancher.

Example usage

```
# Manage an existing Rancher host
resource rancher_host "foo" {
  name           = "foo"
  description    = "The foo node"
  environment_id = "1a5"
  hostname       = "foo.example.com"
  labels {
    role = "database"
  }
}
```

Argument Reference

The following arguments are supported:

- `id` - (Computed) The ID of the resource.
- `name` - (Required) The name of the host.
- `description` - (Optional) A host description.
- `environment_id` - (Required) The ID of the environment the host is associated to.
- `hostname` - (Required) The host name. Used as the primary key to detect the host ID.
- `labels` - (Optional) A dictionary of labels to apply to the host. Computed internal labels are excluded from that list.

rancher_registration_token

Provides a Rancher Registration Token resource. This can be used to create registration tokens for rancher environments and retrieve their information.

Example Usage

```
# Create a new Rancher registration token
resource "rancher_registration_token" "default" {
  name           = "staging_token"
  description    = "Registration token for the staging environment"
  environment_id = "${rancher_environment.default.id}"
  agent_ip       = "1.2.3.4"

  host_labels    {
    orchestration = "true",
    etcd          = "true",
    compute       = "true"
  }
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the registration token.
- `description` - (Optional) A registration token description.
- `environment_id` - (Required) The ID of the environment to create the token for.
- `host_labels` - (Optional) A map of host labels to add to the registration command.
- `agent_ip` - (Optional) A string containing the CATTLE_AGENT_IP to add to the registration command.

Attributes Reference

The following attributes are exported:

- `id` - (Computed) The ID of the resource.
- `image` - (Computed)
- `command` - The command used to start a rancher agent for this environment.
- `registration_url` - The URL to use to register new nodes to the environment.
- `token` - The token to use to register new nodes to the environment.

Import

Registration tokens can be imported using the Environment and Registration token IDs in the form `<environment_id>/<registration_token_id>`.

```
$ terraform import rancher_registration_token.dev_token 1a5/1c11
```

If the credentials for the Rancher provider have access to the global API, then then `environment_id` can be omitted e.g.

```
$ terraform import rancher_registration_token.dev_token 1c11
```

rancher_registry

Provides a Rancher Registy resource. This can be used to create registries for rancher environments and retrieve their information

Example Usage

```
# Create a new Rancher registry
resource "rancher_registry" "dockerhub" {
  name           = "dockerhub"
  description    = "DockerHub Registry"
  environment_id = "${rancher_environment.default.id}"
  server_address = "index.dockerhub.io"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the registry.
- `description` - (Optional) A registry description.
- `environment_id` - (Required) The ID of the environment to create the registry for.
- `server_address` - (Required) The server address for the registry.

Attributes Reference

- `id` - (Computed) The ID of the resource.

Import

Registries can be imported using the Environment and Registry IDs in the form `<environment_id>/<registry_id>`

```
$ terraform import rancher_registry.private_registry 1a5/1sp31
```

If the credentials for the Rancher provider have access to the global API, then then `environment_id` can be omitted e.g.

```
$ terraform import rancher_registry.private_registry 1sp31
```

rancher_registry_credential

Provides a Rancher Registy Credential resource. This can be used to create registry credentials for rancher environments and retrieve their information.

Example Usage

```
# Create a new Rancher registry
resource "rancher_registry_credential" "dockerhub" {
  name         = "dockerhub"
  description  = "DockerHub Registry Credential"
  registry_id  = "${rancher_registry.dockerhub.id}"
  public_value = "myself"
  secret_value = "mypass"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the registry credential.
- `description` - (Optional) A registry credential description.
- `registry_id` - (Required) The ID of the registry to create the credential for.
- `public_value` - (Required) The public value (user name) of the account.
- `secret_value` - (Required) The secret value (password) of the account.

Attributes Reference

- `id` - (Computed) The ID of the resource.

Import

Registry credentials can be imported using the Registry and credentials IDs in the format `<registry_id>/<credential_id>`

```
$ terraform import rancher_registry_credential.private_registry 1sp31/1c605
```

If the credentials for the Rancher provider have access to the global API, then then `registry_id` can be omitted e.g.

```
$ terraform import rancher_registry_credential.private_registry 1c605
```

rancher_secrets

Provides a Rancher Secret resource. This can be used to create secrets for rancher environments and retrieve their information.

Example Usage

```
# Create a new Rancher Secret
resource rancher_secret "foo" {
  name           = "foo"
  environment_id = "${rancher_environment.test.id}"
  value          = "my great password"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the secret.
- `description` - (Optional) A description of the secret.
- `environment_id` - (Required) The ID of the environment to create the secret for.
- `value` - (Required) The secret value.

Import

Secrets can be imported using the Secret ID in the format `<environment_id>/<secret_id>`

```
$ terraform import rancher_secret.mysec 1a5/1se10
```

If the credentials for the Rancher provider have access to the global API, then `environment_id` can be omitted e.g.

```
$ terraform import rancher_secret.mysec 1se10
```

rancher_stack

Provides a Rancher Stack resource. This can be used to create and manage stacks on rancher.

Example Usage

```
# Create a new empty Rancher stack
resource "rancher_stack" "external-dns" {
  name           = "route53"
  description    = "Route53 stack"
  environment_id = "${rancher_environment.default.id}"
  catalog_id     = "library:route53:7"
  scope          = "system"

  environment {
    AWS_ACCESS_KEY        = "MYKEY"
    AWS_SECRET_KEY        = "MYSECRET"
    AWS_REGION            = "eu-central-1"
    TTL                   = "60"
    ROOT_DOMAIN           = "example.com"
    ROUTE53_ZONE_ID       = ""
    HEALTH_CHECK_INTERVAL = "15"
  }
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the stack.
- `description` - (Optional) A stack description.
- `environment_id` - (Required) The ID of the environment to create the stack for.
- `docker_compose` - (Optional) The `docker-compose.yml` content to apply for the stack.
- `rancher_compose` - (Optional) The `rancher-compose.yml` content to apply for the stack.
- `environment` - (Optional) The environment to apply to interpret the docker-compose and rancher-compose files.
- `catalog_id` - (Optional) The catalog ID to link this stack to. When provided, `docker_compose` and `rancher_compose` will be retrieved from the catalog unless they are overridden.
- `scope` - (Optional) The scope to attach the stack to. Must be one of **user** or **system**. Defaults to **user**.
- `start_on_create` - (Optional) Whether to start the stack automatically.
- `finish_upgrade` - (Optional) Whether to automatically finish upgrades to this stack.

Attributes Reference

The following attributes are exported:

- `id` - (Computed) The ID of the resource.
- `rendered_docker_compose` - The interpolated `docker_compose` applied to the stack.
- `rendered_rancher_compose` - The interpolated `rancher_compose` applied to the stack.

Import

Stacks can be imported using the Environment and Stack ID in the form `<environment_id>/<stack_id>`

```
$ terraform import rancher_stack.foo 1a5/1e149
```

If the credentials for the Rancher provider have access to the global API, then then `environment_id` can be omitted e.g.

```
$ terraform import rancher_stack.foo 1e149
```

rancher_volumes

Provides a Rancher Volume resource. This can be used to create volumes for rancher environments and retrieve their information.

Example Usage

```
# Create a new Rancher Volume
resource rancher_volume "foo" {
  name           = "foo"
  environment_id = "${rancher_environment.test.id}"
  driver         = "rancher-nfs"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the volume.
- `description` - (Optional) A description of the volume.
- `environment_id` - (Required) The ID of the environment to create the volume for.
- `driver` - (Required) The volume driver.

Import

Volumes can be imported using the Volume ID in the format `<environment_id>/<volume_id>`

```
$ terraform import rancher_volume.mysec 1a5/1v123456
```

If the credentials for the Rancher provider have access to the global API, then `environment_id` can be omitted e.g.

```
$ terraform import rancher_volume.mysec 1se10
```