RabbitMQ Provider

RabbitMQ (http://www.rabbitmq.com) is an AMQP message broker server. The RabbitMQ provider exposes resources used to manage the configuration of resources in a RabbitMQ server.

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

Example Usage

The following is a minimal example:

```
# Configure the RabbitMQ provider
provider "rabbitmq" {
  endpoint = "http://127.0.0.1"
  username = "guest"
  password = "guest"
}

# Create a virtual host
resource "rabbitmq_vhost" "vhost_1" {
  name = "vhost_1"
}
```

Requirements

The RabbitMQ management plugin must be enabled to use this provider. You can enable the plugin by doing something similar to:

```
$ sudo rabbitmq-plugins enable rabbitmq_management
```

Argument Reference

The following arguments are supported:

- `endpoint` - (Required) The HTTP URL of the management plugin on the RabbitMQ server. The RabbitMQ management plugin *must* be enabled in order to use this provder. *Note*: This is not the IP address or hostname of the RabbitMQ server that you would use to access RabbitMQ directly.
- `username` - (Required) Username to use to authenticate with the server.
- `password` - (Optional) Password for the given user.
- `insecure` - (Optional) Trust self-signed certificates.
- `cacert_file` - (Optional) The path to a custom CA / intermediate certificate.

rabbitmq_binding

The `rabbitmq_binding` resource creates and manages a binding relationship between a queue an exchange.

Example Usage

```
resource "rabbitmq_vhost" "test" {
  name = "test"
}

resource "rabbitmq_permissions" "guest" {
  user  = "guest"
  vhost = "${rabbitmq_vhost.test.name}"

  permissions {
    configure = ".*"
    write     = ".*"
    read      = ".*"
  }
}

resource "rabbitmq_exchange" "test" {
  name  = "test"
  vhost = "${rabbitmq_permissions.guest.vhost}"

  settings {
    type        = "fanout"
    durable     = false
    auto_delete = true
  }
}

resource "rabbitmq_queue" "test" {
  name  = "test"
  vhost = "${rabbitmq_permissions.guest.vhost}"

  settings {
    durable     = true
    auto_delete = false
  }
}

resource "rabbitmq_binding" "test" {
  source           = "${rabbitmq_exchange.test.name}"
  vhost            = "${rabbitmq_vhost.test.name}"
  destination      = "${rabbitmq_queue.test.name}"
  destination_type = "queue"
  routing_key      = "#"
}
```

Argument Reference

The following arguments are supported:

- `source` - (Required) The source exchange.

- `vhost` - (Required) The vhost to create the resource in.
- `destination` - (Required) The destination queue or exchange.
- `destination_type` - (Required) The type of destination (queue or exchange).
- `routing_key` - (Optional) A routing key for the binding.
- `arguments` - (Optional) Additional key/value arguments for the binding.

Attributes Reference

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

- `properties_key` - A unique key to refer to the binding.

Import

Bindings can be imported using the `id` which is composed of `vhost/source/destination/destination_type/properties_key`. E.g.

```
$ terraform import rabbitmq_binding.test test/test/test/queue/%23
```

rabbitmq_exchange

The `rabbitmq_exchange` resource creates and manages an exchange.

Example Usage

```
resource "rabbitmq_vhost" "test" {
  name = "test"
}

resource "rabbitmq_permissions" "guest" {
  user  = "guest"
  vhost = "${rabbitmq_vhost.test.name}"

  permissions {
    configure = ".*"
    write     = ".*"
    read      = ".*"
  }
}

resource "rabbitmq_exchange" "test" {
  name  = "test"
  vhost = "${rabbitmq_permissions.guest.vhost}"

  settings {
    type        = "fanout"
    durable     = false
    auto_delete = true
  }
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the exchange.
- `vhost` - (Required) The vhost to create the resource in.
- `settings` - (Required) The settings of the exchange. The structure is described below.

The `settings` block supports:

- `type` - (Required) The type of exchange.
- `durable` - (Optional) Whether the exchange survives server restarts. Defaults to `false`.
- `auto_delete` - (Optional) Whether the exchange will self-delete when all queues have finished using it.
- `arguments` - (Optional) Additional key/value settings for the exchange.

Attributes Reference

No further attributes are exported.

Import

Exchanges can be imported using the `id` which is composed of `name@vhost`. E.g.

```
terraform import rabbitmq_exchange.test test@vhost
```

rabbitmq_permissions

The `rabbitmq_permissions` resource creates and manages a user's set of permissions.

Example Usage

```
resource "rabbitmq_vhost" "test" {
  name = "test"
}

resource "rabbitmq_user" "test" {
  name     = "mctest"
  password = "foobar"
  tags     = ["administrator"]
}

resource "rabbitmq_permissions" "test" {
  user  = "${rabbitmq_user.test.name}"
  vhost = "${rabbitmq_vhost.test.name}"

  permissions {
    configure = ".*"
    write     = ".*"
    read      = ".*"
  }
}
```

Argument Reference

The following arguments are supported:

- `user` - (Required) The user to apply the permissions to.
- `vhost` - (Required) The vhost to create the resource in.
- `permissions` - (Required) The settings of the permissions. The structure is described below.

The `permissions` block supports:

- `configure` - (Required) The "configure" ACL.
- `write` - (Required) The "write" ACL.
- `read` - (Required) The "read" ACL.

Attributes Reference

No further attributes are exported.

Import

Permissions can be imported using the `id` which is composed of `user@vhost`. E.g.

```
terraform import rabbitmq_permissions.test user@vhost
```

rabbitmq_policy

The `rabbitmq_policy` resource creates and manages policies for exchanges and queues.

Example Usage

```
resource "rabbitmq_vhost" "test" {
  name = "test"
}

resource "rabbitmq_permissions" "guest" {
  user  = "guest"
  vhost = "${rabbitmq_vhost.test.name}"

  permissions {
    configure = ".*"
    write     = ".*"
    read      = ".*"
  }
}

resource "rabbitmq_policy" "test" {
  name  = "test"
  vhost = "${rabbitmq_permissions.guest.vhost}"

  policy {
    pattern  = ".*"
    priority = 0
    apply_to = "all"

    definition {
      ha-mode = "all"
    }
  }
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the policy.
- `vhost` - (Required) The vhost to create the resource in.
- `policy` - (Required) The settings of the policy. The structure is described below.

The `policy` block supports:

- `pattern` - (Required) A pattern to match an exchange or queue name.
- `priority` - (Required) The policy with the greater priority is applied first.
- `apply_to` - (Required) Can either be "exchange", "queues", or "all".
- `definition` - (Required) Key/value pairs of the policy definition. See the RabbitMQ documentation for definition

references and examples.

Attributes Reference

No further attributes are exported.

Import

Policies can be imported using the `id` which is composed of `name@vhost`. E.g.

```
terraform import rabbitmq_policy.test name@vhost
```

rabbitmq_queue

The `rabbitmq_queue` resource creates and manages a queue.

Example Usage

Basic Example

```
resource "rabbitmq_vhost" "test" {
  name = "test"
}

resource "rabbitmq_permissions" "guest" {
  user  = "guest"
  vhost = "${rabbitmq_vhost.test.name}"

  permissions {
    configure = ".*"
    write     = ".*"
    read      = ".*"
  }
}

resource "rabbitmq_queue" "test" {
  name  = "test"
  vhost = "${rabbitmq_permissions.guest.vhost}"

  settings {
    durable     = false
    auto_delete = true
  }
}
```

Example With JSON Arguments

```
variable "arguments" {
  default = <<EOF
{
  "x-message-ttl": 5000
}
EOF
}

resource "rabbitmq_vhost" "test" {
  name = "test"
}

resource "rabbitmq_permissions" "guest" {
  user  = "guest"
  vhost = "${rabbitmq_vhost.test.name}"

  permissions {
    configure = ".*"
    write     = ".*"
    read      = ".*"
  }
}

resource "rabbitmq_queue" "test" {
  name  = "test"
  vhost = "${rabbitmq_permissions.guest.vhost}"

  settings {
    durable     = false
    auto_delete = true
    arguments_json = "${var.arguments}"
  }
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the queue.
- `vhost` - (Required) The vhost to create the resource in.
- `settings` - (Required) The settings of the queue. The structure is described below.

The `settings` block supports:

- `durable` - (Optional) Whether the queue survives server restarts. Defaults to `false`.
- `auto_delete` - (Optional) Whether the queue will self-delete when all consumers have unsubscribed.
- `arguments` - (Optional) Additional key/value settings for the queue. All values will be sent to RabbitMQ as a string. If you require non-string values, use `arguments_json`.
- `arguments_json` - (Optional) A nested JSON string which contains additional settings for the queue. This is useful for when the arguments contain non-string values.

Attributes Reference

No further attributes are exported.

Import

Queues can be imported using the `id` which is composed of `name@vhost`. E.g.

```
terraform import rabbitmq_queue.test name@vhost
```

rabbitmq_user

The `rabbitmq_user` resource creates and manages a user.

Note: All arguments including username and password will be stored in the raw state as plain-text. Read more about sensitive data in state (/docs/state/sensitive-data.html).

Example Usage

```
resource "rabbitmq_user" "test" {
  name     = "mctest"
  password = "foobar"
  tags     = ["administrator", "management"]
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the user.
- `password` - (Required) The password of the user. The value of this argument is plain-text so make sure to secure where this is defined.
- `tags` - (Optional) Which permission model to apply to the user. Valid options are: management, policymaker, monitoring, and administrator.

Attributes Reference

No further attributes are exported.

Import

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

```
terraform import rabbitmq_user.test mctest
```

rabbitmq_vhost

The `rabbitmq_vhost` resource creates and manages a vhost.

Example Usage

```
resource "rabbitmq_vhost" "my_vhost" {
  name = "my_vhost"
}
```

Argument Reference

The following arguments are supported:

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

Attributes Reference

No further attributes are exported.

Import

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

```
terraform import rabbitmq_vhost.my_vhost my_vhost
```