Rundeck Provider

The Rundeck provider allows Terraform to create and configure Projects, Jobs and Keys in Rundeck (http://rundeck.org/). Rundeck is a tool for runbook automation and execution of arbitrary management tasks, allowing operators to avoid logging in to individual machines directly via SSH.

The provider configuration block accepts the following arguments:

- `url` - (Required) The root URL of a Rundeck server. May alternatively be set via the `RUNDECK_URL` environment variable.
- `api_version` - (Optional) The API version of the server. Defaults to `14`, the minium supported version. May alternatively be set via the `RUNDECK_API_VERSION` environment variable.
- `auth_token` - (Required) The API auth token to use when making requests. May alternatively be set via the `RUNDECK_AUTH_TOKEN` environment variable.

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

Example Usage

```
provider "rundeck" {
  url         = "http://rundeck.example.com/"
  api_version = "26"
  auth_token  = "abcd1234"
}

resource "rundeck_project" "anvils" {
  name        = "anvils"
  description = "Application for managing Anvils"

  ssh_key_storage_path = "${rundeck_private_key.anvils.path}"

  resource_model_source {
    type = "file"

    config = {
      format = "resourcexml"

      # This path is interpreted on the Rundeck server.
      file = "/var/rundeck/projects/anvils/resources.xml"
    }
  }
}

resource "rundeck_job" "bounceweb" {
  name              = "Bounce Web Servers"
  project_name      = "${rundeck_project.anvils.name}"
  node_filter_query = "tags: web"
  description       = "Restart the service daemons on all the web servers"

  command {
    shell_command = "sudo service anvils restart"
  }
}

resource "rundeck_public_key" "anvils" {
  path         = "anvils/id_rsa.pub"
  key_material = "ssh-rsa yada-yada-yada"
}

resource "rundeck_private_key" "anvils" {
  path         = "anvils/id_rsa"
  key_material = "${file(\"id_rsa.pub\")}"
}

data "local_file" "acl" {
  filename = "${path.module}/acl.yaml"
}

resource "rundeck_acl_policy" "example" {
  name = "ExampleAcl.aclpolicy"

  policy = "${data.local_file.acl.content}"
}
```

rundeck_acl_policy

The acl policy resource allows Rundeck projects to be managed by Terraform.

Example Usage

```
data "local_file" "acl" {
  filename = "${path.module}/acl.yaml"
}

resource "rundeck_acl_policy" "example" {
  name = "ExampleAcl.aclpolicy"

  policy = "${data.local_file.acl.content}"
}
```

Note that the above configuration assumes the existence of an `acl.yaml` file in the project directory. This resource passes the raw YAML policy string to Rundeck which stores and returns it as-is. A future `acl_policy_document` data source is planned to allow defining the policy in terraform configuration.

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the policy. Must end with `.aclpolicy`.
- `policy` - (Required) The name of the job, used to describe the job in the Rundeck UI.

rundeck_job

The job resource allows Rundeck jobs to be managed by Terraform. In Rundeck a job is a particular named set of steps that can be executed against one or more of the nodes configured for its associated project.

Each job belongs to a project. A project can be created with the `rundeck_project` resource.

Example Usage

```
resource "rundeck_job" "bounceweb" {
    name = "Bounce Web Servers"
    project_name = "anvils"
    node_filter_query = "tags: web"
    description = "Restart the service daemons on all the web servers"

    command {
        shell_command = "sudo service anvils restart"
    }
    notification {
        type = "on_success"
        email {
            recipients = ["example@foo.bar"]
        }
    }
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the job, used to describe the job in the Rundeck UI.
- `description` - (Required) A longer description of the job, describing the job in the Rundeck UI.
- `project_name` - (Required) The name of the project that this job should belong to.
- `execution_enabled` - (Optional) If you want job execution to be enabled or disabled. Defaults to `true`.
- `group_name` - (Optional) The name of a group within the project in which to place the job. Setting this creates collapsable subcategories within the Rundeck UI's project job index.
- `log_level` - (Optional) The log level that Rundeck should use for this job. Defaults to "INFO".
- `schedule` - (Optional) The jobs schedule in Unix crontab format
- `schedule_enabled` - (Optional) Sets the job schedule to be enabled or disabled. Defaults to `true`.
- `allow_concurrent_executions` - (Optional) Boolean defining whether two or more executions of this job can run concurrently. The default is `false`, meaning that jobs will only run sequentially.

- `max_thread_count` - (Optional) The maximum number of threads to use to execute this job, which controls on how many nodes the commands can be run simulateneously. Defaults to 1, meaning that the nodes will be visited sequentially.
- `continue_on_error` - (Optional) Boolean defining whether Rundeck will continue to run subsequent steps if any intermediate step fails. Defaults to `false`, meaning that execution will stop and the execution will be considered to have failed.
- `rank_attribute` - (Optional) The name of the attribute that will be used to decide in which order the nodes will be visited while executing the job across multiple nodes.
- `rank_order` - (Optional) Keyword deciding which direction the nodes are sorted in terms of the chosen `rank_attribute`. May be either "ascending" (the default) or "descending".
- `success_on_empty_node_filter` - (Optional) Boolean determining if an empty node filter yields a successful result.
- `preserve_options_order`: (Optional) Boolean controlling whether the configured options will be presented in their configuration order when shown in the Rundeck UI. The default is `false`, which means that the options will be displayed in alphabetical order by name.
- `command_ordering_strategy`: (Optional) The name of the strategy used to describe how to traverse the matrix of nodes and commands. The default is "node-first", meaning that all commands will be executed on a single node before moving on to the next. May also be set to "step-first", meaning that a single step will be executed across all nodes before moving on to the next step.
- `node_filter_query` - (Optional) A query string using Rundeck's node filter language (http://rundeck.org/docs/manual/node-filters.html#node-filter-syntax) that defines which subset of the project's nodes will be used to execute this job.
- `node_filter_exclude_precedence`: (Optional) Boolean controlling a deprecated Rundeck feature that controls whether node exclusions take priority over inclusions.
- `option`: (Optional) Nested block defining an option a user may set when executing this job. A job may have any number of options. The structure of this nested block is described below.
- `command`: (Required) Nested block defining one step in the job workflow. A job must have one or more commands. The structure of this nested block is described below.
- `notification`: (Optional) Nested block defining notifications on the job workflow. The structure of this nested block is described below.

`option` blocks have the following structure:

- `name`: (Required) Unique name that will be shown in the UI when entering values and used as a variable name for template substitutions.
- `default_value`: (Optional) A default value for the option.
- `value_choices`: (Optional) A list of strings giving a set of predefined values that the user may choose from when entering a value for the option.
- `value_choices_url`: (Optional) Can be used instead of `value_choices` to cause Rundeck to obtain a list of choices dynamically by fetching this URL.
- `require_predefined_choice`: (Optional) Boolean controlling whether the user is allowed to enter values not included in the predefined set of choices (`false`, the default) or whether a predefined choice is required (`true`).

- `validation_regex` : (Optional) A regular expression that a provided value must match in order to be accepted.
- `description` : (Optional) A longer description of the option to be shown in the UI.
- `required` : (Optional) Boolean defining whether the user must provide a value for the option. Defaults to `false` .
- `allow_multiple_values` : (Optional) Boolean defining whether the user may select multiple values from the set of predefined values. Defaults to `false` , meaning that the user may choose only one value.
- `multi_value_delimiter` : (Optional) Delimiter used to join together multiple values into a single string when `allow_multiple_values` is set and the user chooses multiple values.
- `obscure_input` : (Optional) Boolean controlling whether the value of this option should be obscured during entry and in execution logs. Defaults to `false` , but should be set to `true` when the requested value is a password, private key or any other secret value.
- `exposed_to_scripts` : (Optional) Boolean controlling whether the value of this option is available to scripts executed by job commands. Defaults to `false` .

`command` blocks must have any one of the following combinations of arguments as contents:

- `description` : (Optional) gives a description to the command block.
- `shell_command` gives a single shell command to execute on the nodes.
- `inline_script` gives a whole shell script, inline in the configuration, to execute on the nodes.
- `script_file` and `script_file_args` together describe a script that is already pre-installed on the nodes which is to be executed.
- A `job` block, described below, causes another job within the same project to be executed as a command.
- A `step_plugin` block, described below, causes a step plugin to be executed as a command.
- A `node_step_plugin` block, described below, causes a node step plugin to be executed once for each node.

A command's `job` block has the following structure:

- `name` : (Required) The name of the job to execute. The target job must be in the same project as the current job.
- `group_name` : (Optional) The name of the group that the target job belongs to, if any.
- `run_for_each_node` : (Optional) Boolean controlling whether the job is run only once (`false` , the default) or whether it is run once for each node (`true`).
- `args` : (Optional) A string giving the arguments to pass to the target job, using Rundeck's job arguments syntax (http://rundeck.org/docs/manual/jobs.html#job-reference-step).
- `nodefilters` : (Optional) A map for overriding the referenced job's node filters.

A command's `nodefilters` block has the following structure:

- `excludeprecedence` : (Optional) Whether to exclude precedence or not.
- `filter` : (Optional) The node filter query string to use.

A command's `step_plugin` or `node_step_plugin` block both have the following structure:

- `type` : (Required) The name of the plugin to execute.

- `config` : (Optional) Map of arbitrary configuration parameters for the selected plugin.

`notification` blocks have the following structure:

- `type` : (Required) The name of the type of notification. Can be of type `on_success` , `on_failure` , `on_start` .
- `email` : (Optional) block listed below to send emails as notificiation.
- `webhook_urls` : (Optional) A list of urls to send a webhook notification.
- `plugin` : (Optional) A block listed below to send notifications using a plugin.

A notification's `email` block has the following structure:

- `attach_log` : (Optional) A boolean to attach log to email or not. Defaults to false.
- `recipients` : (Required) A list of recipients to receive email.
- `subject` : (Optional) Name of email subject.

A notification's `plugin` block has the following structure:

- `type` - (Required) The name of the plugin to use.
- `config` - (Required) Map of arbitrary configuration properties for the selected plugin.

Attributes Reference

The following attribute is exported:

- `id` - A unique identifier for the job.

rundeck_private_key

The private key resource allows SSH private keys to be stored into Rundeck's key store. The key store is where Rundeck keeps credentials that are needed to access the nodes on which it runs commands.

Example Usage

```
resource "rundeck_private_key" "anvils" {
    path = "anvils/id_rsa"
    key_material = "${file("/id_rsa")}"
}
```

Argument Reference

The following arguments are supported:

- `path` - (Required) The path within the key store where the key will be stored.
- `key_material` - (Required) The private key material to store, serialized in any way that is accepted by OpenSSH.

The key material is hashed before it is stored in the state file, so sharing the resulting state will not disclose the private key contents.

Attributes Reference

Rundeck does not allow stored private keys to be retrieved via the API, so this resource does not export any attributes.

rundeck_project

The project resource allows Rundeck projects to be managed by Terraform. In Rundeck a project is the container object for a set of jobs and the configuration for which servers those jobs can be run on.

Example Usage

```
resource "rundeck_project" "anvils" {
    name = "anvils"
    description = "Application for managing Anvils"

    ssh_key_storage_path = "anvils/id_rsa"

    resource_model_source {
        type = "file"
        config = {
            format = "resourcexml"
            # This path is interpreted on the Rundeck server.
            file = "/var/rundeck/projects/anvils/resources.xml"
        }
    }
}
```

Note that the above configuration assumes the existence of a `resources.xml` file in the filesystem on the Rundeck server. The Rundeck provider does not itself support creating such a file, but one way to place it would be to use the `file` provisioner to copy a configuration file from the module directory.

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the project, used both in the UI and to uniquely identify the project. Must therefore be unique across a single Rundeck installation.
- `resource_model_source` - (Required) Nested block instructing Rundeck on how to determine the set of resources (nodes) for this project. The nested block structure is described below.
- `description` - (Optional) A description of the project, to be displayed in the Rundeck UI. Defaults to "Managed by Terraform".
- `default_node_file_copier_plugin` - (Optional) The name of a plugin to use to copy files onto nodes within this project. Defaults to `jsch-scp`, which uses the "Secure Copy" protocol to send files over SSH.
- `default_node_executor_plugin` - (Optional) The name of a plugin to use to run commands on nodes within this project. Defaults to `jsch-ssh`, which uses the SSH protocol to access the nodes.
- `ssh_authentication_type` - (Optional) When the SSH-based file copier and executor plugins are used, the type of SSH authentication to use. Defaults to `privateKey`.

- `ssh_key_storage_path` - (Optional) When the SSH-based file copier and executor plugins are used, the location within Rundeck's key store where the SSH private key can be found. Private keys can be uploaded to rundeck using the `rundeck_private_key` resource.
- `ssh_key_file_path` - (Optional) Like `ssh_key_storage_path` except that the key is read from the Rundeck server's local filesystem, rather than from the key store.
- `extra_config` - (Optional) Behind the scenes a Rundeck project is really an arbitrary set of key/value pairs. This map argument allows setting any configuration properties that aren't explicitly supported by the other arguments described above, but due to limitations of Terraform the key names must be written with slashes in place of dots. Do not use this argument to set properties that the above arguments set, or undefined behavior will result.

`resource_model_source` blocks have the following nested arguments:

- `type` - (Required) The name of the resource model plugin to use.
- `config` - (Required) Map of arbitrary configuration properties for the selected resource model plugin.

Attributes Reference

The following attributes are exported:

- `name` - The unique name that identifies the project, as set in the arguments.
- `ui_url` - The URL of the index page for this project in the Rundeck UI.

rundeck_public_key

The public key resource allows SSH public keys to be stored into Rundeck's key store. The key store is where Rundeck keeps credentials that are needed to access the nodes on which it runs commands.

This resource also allows the retrieval of an existing public key from the store, so that it may be used in the configuration of other resources such as `aws_key_pair`.

Example Usage

```
resource "rundeck_public_key" "anvils" {
    path = "anvils/id_rsa.pub"
    key_material = "ssh-rsa yada-yada-yada"
}
```

Argument Reference

The following arguments are supported:

- `delete` - (Computed) True if the key should be deleted when the resource is deleted. Defaults to true if key_material is provided in the configuration.
- `path` - (Required) The path within the key store where the key will be stored. By convention this path name normally ends with ".pub" and otherwise has the same name as the associated private key.
- `key_material` - (Optional) The public key string to store, serialized in any way that is accepted by OpenSSH. If this is not included, `key_material` becomes an attribute that can be used to read the already-existing key material in the Rundeck store.

The key material is included inline as a string, which is consistent with the way a public key is provided to the `aws_key_pair`, `cloudstack_ssh_keypair`, `digitalocean_ssh_key` and `openstack_compute_keypair_v2` resources. This means the `key_material` argument can be populated from the interpolation of the `public_key` attribute of such a keypair resource, or vice-versa.

Attributes Reference

The following attributes are exported:

- `url` - The URL at which the key material can be retrieved from the key store by other clients.
- `key_material` - If `key_material` is omitted in the configuration, it becomes an attribute that exposes the key material already stored at the given `path`.