Nomad Provider

HashiCorp Nomad (https://www.nomadproject.io) is an application scheduler. The Nomad provider exposes resources to interact with a Nomad cluster.

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

Example Usage

```
# Configure the Nomad provider
provider "nomad" {
  address = "http://nomad.mycompany.com:4646"
  region  = "us-east-2"
}

# Register a job
resource "nomad_job" "monitoring" {
  jobspec = "${file("${path.module}/jobspec.hcl")}"
}
```

Argument Reference

The following arguments are supported:

- `address (string: "http://127.0.0.1:4646")` - The HTTP(S) API address of the Nomad agent. This must include the leading protocol (e.g. `https://`). This can also be specified as the `NOMAD_ADDR` environment variable.
- `region (string: "")` - The Nomad region to target. This can also be specified as the `NOMAD_REGION` environment variable.
- `ca_file (string: "")` - A local file path to a PEM-encoded certificate authority used to verify the remote agent's certificate. This can also be specified as the `NOMAD_CACERT` environment variable.
- `cert_file (string: "")` - A local file path to a PEM-encoded certificate provided to the remote agent. If this is specified, `key_file` is also required. This can also be specified as the `NOMAD_CLIENT_CERT` environment variable.
- `key_file (string: "")` - A local file path to a PEM-encoded private key. This is required if `cert_file` is specified. This can also be specified via the `NOMAD_CLIENT_KEY` environment variable.
- `vault_token (string: "")` - A vault token to be inserted in the job file. This can also be specified as the `VAULT_TOKEN` environment variable or using a vault token helper (see Vault's documentation (https://www.vaultproject.io/docs/commands/token-helper.html) for more details).
- `secret_id (string: "")` - The Secret ID of an ACL token to make requests with, for ACL-enabled clusters. This can also be specified via the `NOMAD_TOKEN` environment variable.

nomad_deployments

Retrieve a list of deployments in Nomad.

Example Usage

```
data "nomad_deployments" "example" {}
```

Attribute Reference

The following attributes are exported:

- `deployments` : `list of maps` a list of deployments in the cluster.
  - `ID` : `string` Deployment ID.
  - `JobID` : `string` Job ID associated with the deployment.
  - `JobVersion` : `string` Job version.
  - `Status` : `string` Deployment status.
  - `StatusDescription` : `string` Detailed description of the deployment's status.

nomad_job

Get information on an job ID. The aim of this datasource is to enable you to act on various settings and states of a particular job.

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

Example Usage

Get the data about a snapshot:

```
data "nomad_job" "example1" {
  job_id = "example_job"
}
```

Argument Reference

The following arguments are supported:

- `job_id` : `(string)` ID of the job.

Attributes Reference

The following attributes are exported:

- `name` : `(string)` Name of the job.
- `type` : `(string)` Scheduler type used during job creation.
- `version` : `(integer)` Version of the specified job.
- `namespace` : `(string)` Namespace of the specified job.
- `region` : `(string)` Region where the Nomad cluster resides.
- `datacenters` : `(list of strings)` Datacenters allowed to run the specified job.
- `status` : `(string)` Execution status of the specified job.
- `status_description` : `(string)` Status description of the specified job.
- `submit_time` : `(integer)` Job submission date.
- `create_index` : `(integer)` Creation Index.
- `modify_index` : `(integer)` Modification Index.
- `job_modify_index` : `(integer)` Job modify index (used for version verification).
- `stop` : `(boolean)` Job enabled status.

- `priority` : `(integer)` Used for the prioritization of scheduling and resource access.
- `parent_id` : `(string)` Job's parent ID.
- `task_groups` : `(list of maps)` A list of of the job's task groups.
  - `placed_canaries` : `(string)`
  - `auto_revert` : `(boolean)`
  - `promoted` : `(boolean)`
  - `desired_canaries` : `(integer)`
  - `desired_total` : `(integer)`
  - `placed_alloc` : `(integer)`
  - `healthy_alloc` : `(integer)`
  - `unhealthy_alloc` : `(integer)`
- `stable` : `(boolean)` Job stability status.
- `all_at_once` : `(boolean)` If the scheduler can make partial placements on oversubscribed nodes.
- `contraints` : `(list of maps)` Job constraints.
  - `ltarget` : `(string)` Attribute being constrained.
  - `rtarget` : `(string)` Constraint value.
  - `operand` : `(string)` Operator used to compare the attribute to the constraint.
- `update_strategy` : `(list of maps)` Job's update strategy which controls rolling updates and canary deployments.
  - `stagger` : `(string)` Delay between migrating job allocations off cluster nodes marked for draining.
  - `max_parallel` : `(integer)` Number of task groups that can be updated at the same time.
  - `health_check` : `(string)` Type of mechanism in which allocations health is determined.
  - `min_healthy_time` : `(string)` Minimum time the job allocation must be in the healthy state.
  - `healthy_deadline` : `(string)` Deadline in which the allocation must be marked as healthy after which the allocation is automatically transitioned to unhealthy.
  - `auto_revert` : `(boolean)` Specifies if the job should auto-revert to the last stable job on deployment failure.
  - `canary` : `(integer)` Number of canary jobs that need to reach healthy status before unblocking rolling updates.
- `periodic_config` : `(list of maps)` Job's periodic configuration (time based scheduling).
  - `enabled` : `(boolean)` If periodic scheduling is enabled for the specified job.
  - `spec` : `(string)`
  - `spec_type` : `(string)`
  - `prohibit_overlap` : `(boolean)` If the specified job should wait until previous instances of the job have completed.
  - `timezone` : `(string)` Time zone to evaluate the next launch interval against.

nomad_namespaces

Retrieve a list of namespaces available in Nomad.

Example Usage

```
data "nomad_namespaces" "namespaces" {
}

resource "nomad_acl_policy" "namespace" {
  count = "${length(data.nomad_namespaces.namespaces.namespaces)}"
  name = "namespace-${data.nomad_namespaces.namespaces[count.index]}"
  description = "Write to the namespace ${data.nomad_namespaces.namespaces[count.index]}"
  rules_hcl = <<EOT
namespace "${data.nomad_namespaces.namespaces[count.index]}" {
  policy = "write"
}
EOT
}
```

Attribute Reference

The following attributes are exported:

- `namespaces (list of strings)` - a list of namespaces available in the cluster.

nomad_regions

Retrieve a list of regions available in Nomad.

Example Usage

```
data "nomad_regions" "regions" {
}

data "template_file" "jobs" {
  count = "${length(data.nomad_regions.regions.regions)}"
  template = <<EOT
job "foo" {
  datacenters = ["dc1"]
  type = "service"
  region = "$$region"
  # ... rest of your job here
}
EOT
  vars {
    region = "${data.nomad_regions.regions[count.index]}"
  }
}

resource "nomad_job" "app" {
  count = "${length(data.nomad_regions.regions.regions)}"
  jobspec = "${data.template_file.jobs[count.index].rendered}"
}
```

Attribute Reference

The following attributes are exported:

- `regions (list of strings)` - a list of regions available in the cluster.

nomad_acl_policy

Manages an ACL policy registered in Nomad.

Example Usage

Registering a policy from an HCL file:

```
resource "nomad_acl_policy" "dev" {
  name        = "dev"
  description = "Submit jobs to the dev environment."
  rules_hcl   = "${file("${path.module}/dev.hcl")}"
}
```

Registering a policy from inline HCL:

```
resource "nomad_acl_policy" "dev" {
  name        = "dev"
  description = "Submit jobs to the dev environment."

  rules_hcl = <<EOT
namespace "dev" {
  policy = "write"
}
EOT
}
```

Argument Reference

The following arguments are supported:

- `name (string: <required>)` - A unique name for the policy.
- `rules_hcl (string: <required>)` - The contents of the policy to register, as HCL or JSON.
- `description (string: "")` - A description of the policy.

nomad_acl_token

Manages an ACL token in Nomad.

Warning: this resource will store any tokens it creates in Terraform's state file. Take care to protect your state file (/docs/state/sensitive-data.html).

Example Usage

Creating a token with limited policies:

```
resource "nomad_acl_token" "ron" {
  name     = "Ron Weasley"
  type     = "client"
  policies = ["dev", "qa"]
}
```

Creating a global token that will be replicated to all regions:

```
resource "nomad_acl_token" "hermione" {
  name     = "Hermione Granger"
  type     = "client"
  policies = ["dev", "qa"]
  global   = true
}
```

Creating a token with full access to the cluster:

```
resource "nomad_acl_token" "hagrid" {
  name = "Rubeus Hagrid"

  # Hagrid is the keeper of the keys
  type = "management"
}
```

Accessing the token:

```
resource "nomad_acl_token" "token" {
  type     = "client"
  policies = ["dev"]
}

output "nomad_token" {
  value = "${nomad_acl_token.token.secret_id}"
}
```

Argument Reference

The following arguments are supported:

- `type (string: <required>)` - The type of token this is. Use `client` for tokens that will have policies associated with them. Use `management` for tokens that can perform any action.
- `name (string: "")` - A human-friendly name for this token.
- `policies (set: [])` - A set of policy names to associate with this token. Must be set on `client` -type tokens, must not be set on `management` -type tokens. Policies do not need to exist before being used here.
- `global (bool: false)` - Whether the token should be replicated to all regions, or if it will only be used in the region it was created in.

In addition to the above arguments, the following attributes are exported and can be referenced:

- `accessor_id (string)` - A non-sensitive identifier for this token that can be logged and shared safely without granting any access to the cluster.
- `secret_id (string)` - The token value itself, which is presented for access to the cluster.

nomad_job

Manages a job registered in Nomad.

This can be used to initialize your cluster with system jobs, common services, and more. In day to day Nomad use it is common for developers to submit jobs to Nomad directly, such as for general app deployment. In addition to these apps, a Nomad cluster often runs core system services that are ideally setup during infrastructure creation. This resource is ideal for the latter type of job, but can be used to manage any job within Nomad.

Example Usage

Registering a job from a jobspec file:

```
resource "nomad_job" "app" {
  jobspec = "${file("${path.module}/job.hcl")}"
}
```

Registering a job from an inline jobspec. This is less realistic but is an example of how it is possible. More likely, the contents will be paired with something such as the template_file (https://www.terraform.io/docs/providers/template/d/file.html) resource to render parameterized jobspecs.

```
resource "nomad_job" "app" {
  jobspec = <<EOT
job "foo" {
  datacenters = ["dc1"]
  type = "service"
  group "foo" {
    task "foo" {
      driver = "raw_exec"
      config {
        command = "/bin/sleep"
        args = ["1"]
      }

      resources {
        cpu = 20
        memory = 10
      }

      logs {
        max_files = 3
        max_file_size = 10
      }
    }
  }
}
EOT
}
```

Argument Reference

The following arguments are supported:

- `jobspec (string: <required>)` - The contents of the jobspec to register.
- `deregister_on_destroy (bool: true)` - Determines if the job will be deregistered when this resource is destroyed in Terraform.
- `deregister_on_id_change (bool: true)` - Determines if the job will be deregistered if the ID of the job in the jobspec changes.
- `detach (bool: true)` - If true, the provider will return immediately after creating or updating, instead of monitoring.
- `policy_override (bool: false)` - Determines if the job will override any soft-mandatory Sentinel policies and register even if they fail.
- `json (bool: false)` - Set this to true if your jobspec is structured with JSON instead of the default HCL.

nomad_namespace

Provisions a namespace within a Nomad cluster.

> **Enterprise Only!** This API endpoint and functionality only exists in Nomad Enterprise. This is not present in the open source version of Nomad.

Nomad auto-generates a default namespace called `default`. This namespace cannot be removed, so destroying a `nomad_namespace` resource where `name = "default"` will cause the namespace to be reset to its default configuration.

Example Usage

Registering a namespace:

```
resource "nomad_namespace" "dev" {
  name        = "dev"
  description = "Shared development environment."
  quota = "dev"
}
```

Registering a namespace with a quota

```
resource "nomad_quota_specification" "web_team" {
  name        = "web-team"
  description = "web team quota"

  limits {
    region = "global"

    region_limit {
      cpu       = 1000
      memory_mb = 256
    }
  }
}

resource "nomad_namespace" "web" {
  name = "web"
  description = "Web team production environment."
  quota = "${nomad_quota_specification.web_team.name}"
}
```

Argument Reference

The following arguments are supported:

- `name (string: <required>)` - A unique name for the namespace.

- `description (string: "")` - A description of the namespace.
- `quota (string: "")` - A resource quota to attach to the namespace.

nomad_quota_specification

Manages a quota specification in a Nomad cluster.

Example Usage

Registering a quota specification:

```
resource "nomad_quota_specification" "prod_api" {
  name        = "prod-api"
  description = "Production instances of backend API servers"

  limits {
    region = "global"

    region_limit {
      cpu       = 2400
      memory_mb = 1200
    }
  }
}
```

Argument Reference

The following arguments are supported:

- `name (string: <required>)` - A unique name for the quota specification.
- `description (string: "")` - A description of the quota specification.
- `limits (block: <required>)` - A block of quota limits to enforce. Can be repeated. See below for the structure of this block.

`limits` blocks

The `limits` block describes the quota limits to be enforced. It supports the following arguments:

- `region (string: <required>)` - The region these limits should apply to.
- `region_limit (block: <required>)` - The limits to enforce. This block may only be specified once in the `limits` block. Its structure is documented below.

`region_limit` blocks

The `region_limit` block describes the quota limits to be enforced on a region. It supports the following arguments:

- `cpu (int: 0)` - The amount of CPU to limit allocations to. A value of zero is treated as unlimited, and a negative

value is treated as fully disallowed.

- `memory_mb (int: 0)` - The amount of memory (in megabytes) to limit allocations to. A value of zero is treated as unlimited, and a negative value is treated as fully disallowed.

nomad_sentinel_policy

Manages a Sentinel policy registered in Nomad.

Enterprise Only! This API endpoint and functionality only exists in Nomad Enterprise. This is not present in the open source version of Nomad.

Example Usage

```
resource "nomad_sentinel_policy" "exec-only" {
  name        = "exec-only"
  description = "Only allow jobs that are based on an exec driver."

  policy = <<EOT
main = rule { all_drivers_exec }

# all_drivers_exec checks that all the drivers in use are exec
all_drivers_exec = rule {
    all job.task_groups as tg {
        all tg.tasks as task {
            task.driver is "exec"
        }
    }
}
EOT

  scope = "submit-job"

  # allow administrators to override
  enforcement_level = "soft-mandatory"
}
```

Argument Reference

The following arguments are supported:

- `name (string: <required>)` - A unique name for the policy.
- `policy (string: <required>)` - The contents of the policy to register.
- `enforcement_level (strings: <required>)` - The enforcement level (https://www.nomadproject.io/guides/sentinel-policy.html#enforcement-level) for this policy.
- `scope (strings: <required>)` - The scope (https://www.nomadproject.io/guides/sentinel-policy.html#policy-scope) for this policy.
- `description (string: "")` - A description of the policy.