» 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.

» 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 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.
- `policy_override (bool: false)` - Determins if the job will override any soft-mandatory Sentinel policies and register even if they fail.

» 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.

» Example Usage

Registering a namespace:

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

» Argument Reference

The following arguments are supported:

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

- `description (string: "")` - A description of 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 for this policy.
- `scope (strings: <required>)` - The scope for this policy.
- `description (string: "")` - A description of the policy.