# » consul_agent_self

The `consul_agent_self` data source returns configuration and status data from the agent specified in the `provider`.

## » Example Usage

```
data "consul_agent_self" "read-dc1-agent" {
  query_options {
    # Optional parameter: implicitly uses the current datacenter of the agent
    datacenter = "dc1"
  }
}

# Set the description to a whitespace delimited list of the services
resource "example_resource" "app" {
  description = "Consul datacenter ${data.consul_agent_self.read-dc1-agent.datacenter}"

  # ...
}
```

## » Attributes Reference

The following attributes are exported:

- `acl_datacenter`
- `acl_default_policy`
- `acl_disabled_ttl`
- `acl_down_policy`
- `acl_enforce_0_8_semantics`
- `acl_ttl`
- `addresses`
- `advertise_addr`
- `advertise_addr_wan`
- `advertise_addrs`
- `atlas_join`
- `bind_addr`
- `bootstrap_expect`
- `bootstrap_mode`
- `check_deregister_interval_min`
- `check_reap_interval`
- `check_update_interval`
- `client_addr`

- `dns` - A map of DNS configuration attributes. See below for details on the contents of the `dns` attribute.
- `dns_recursors` - A list of all DNS recursors.
- `data_dir`
- `datacenter`
- `dev_mode`
- `domain`
- `enable_anonymous_signature`
- `enable_coordinates`
- `enable_debug`
- `enable_remote_exec`
- `enable_syslog`
- `enable_ui`
- `enable_update_check`
- `id`
- `leave_on_int`
- `leave_on_term`
- `log_level`
- `name`
- `performance`
- `pid_file`
- `ports`
- `protocol_version`
- `reconnect_timeout_lan`
- `reconnect_timeout_wan`
- `rejoin_after_leave`
- `retry_join`
- `retry_join_ec2` - A map of EC2 retry attributes. See below for details on the available information.
- `retry_join_gce` - A map of GCE retry attributes. See below for details on the available information.
- `retry_join_wan`
- `retry_max_attempts`
- `retry_max_attempts_wan`
- `serf_lan_bind_addr`
- `serf_wan_bind_addr`
- `server_mode`
- `server_name`
- `session_ttl_min`
- `start_join`
- `start_join_wan`
- `syslog_facility`
- `tls_ca_file`
- `tls_cert_file`
- `tls_key_file`
- `tls_min_version`

- tls_verify_incoming
- tls_verify_outgoing
- tls_verify_server_hostname
- tagged_addresses
- telemetry - A map of telemetry configuration.
- translate_wan_addrs
- ui_dir
- unix_sockets
- version - The version of the Consul agent.
- version_prerelease
- version_revision

### » DNS Attributes

- allow_stale
- enable_compression
- enable_truncate
- max_stale
- node_ttl
- only_passing
- recursor_timeout
- service_ttl
- udp_answer_limit

### » Retry Join EC2 Attributes

- region
- tag_key
- tag_value

### » Retry Join GCE Attributes

- credentials_file
- project_name
- tag_value
- zone_pattern

### » Telemetry Attributes

- circonus_api_app
- circonus_api_token
- circonus_api_url
- circonus_broker_id

- circonus_check_id
- circonus_check_tags
- circonus_display_name
- circonus_force_metric_activation
- circonus_instance_id
- circonus_search_tag
- circonus_select_tag
- circonus_submission_interval
- circonus_submission_url
- dogstatsd_addr
- dogstatsd_tags
- enable_hostname
- statsd_addr
- statsite_addr
- statsite_prefix

## » consul_catalog_nodes

The consul_catalog_nodes data source returns a list of Consul nodes that have been registered with the Consul cluster in a given datacenter. By specifying a different datacenter in the query_options it is possible to retrieve a list of nodes from a different WAN-attached Consul datacenter.

### » Example Usage

```
data "consul_catalog_nodes" "read-dc1-nodes" {
  query_options {
    # Optional parameter: implicitly uses the current datacenter of the agent
    datacenter = "dc1"
  }
}

# Set the description to a whitespace delimited list of the node names
resource "example_resource" "app" {
  description = "${join(" ", formatlist("%s", data.consul_catalog_nodes.node_names))}"

  # ...
}
```

### » Argument Reference

The following arguments are supported:

- `datacenter` - (Optional) The Consul datacenter to query. Defaults to the same value found in `query_options` parameter specified below, or if that is empty, the `datacenter` value found in the Consul agent that this provider is configured to talk to.
- `query_options` - (Optional) See below.

The `query_options` block supports the following:

- `allow_stale` - (Optional) When `true`, the default, allow responses from Consul servers that are followers.
- `require_consistent` - (Optional) When `true` force the client to perform a read on at least quorum servers and verify the result is the same. Defaults to `false`.
- `token` - (Optional) Specify the Consul ACL token to use when performing the request. This defaults to the same API token configured by the `consul` provider but may be overriden if necessary.
- `wait_index` - (Optional) Index number used to enable blocking quereis.
- `wait_time` - (Optional) Max time the client should wait for a blocking query to return.

## » Attributes Reference

The following attributes are exported:

- `datacenter` - The datacenter the keys are being read from to.
- `node_ids` - A list of the Consul node IDs.
- `node_names` - A list of the Consul node names.
- `nodes` - A list of nodes and details about each Consul agent. The list of per-node attributes is detailed below.

The following is a list of the per-node attributes contained within the `nodes` map:

- `id` - The Node ID of the Consul agent.
- `meta` - Node meta data tag information, if any.
- `name` - The name of the Consul node.
- `address` - The IP address the node is advertising to the Consul cluster.
- `tagged_addresses` - List of explicit LAN and WAN IP addresses for the agent.

# » consul_catalog_service

`consul_catalog_service` provides details about a specific Consul service in a given datacenter. The results include a list of nodes advertising the specified service, the node's IP address, port number, node ID, etc. By specifying a different datacenter in the `query_options` it is possible to retrieve a list of services from a different WAN-attached Consul datacenter.

This data source is different from the `consul_catalog_services` (plural) data source, which provides a summary of the current Consul services.

## » Example Usage

```
data "consul_catalog_service" "read-consul-dc1" {
  query_options {
    # Optional parameter: implicitly uses the current datacenter of the agent
    datacenter = "dc1"
  }

  name = "consul"
}

# Set the description to a whitespace delimited list of the node names
resource "example_resource" "app" {
  description = "${join(" ", data.consul_catalog_service.nodes)}"

  # ...
}
```

## » Argument Reference

The following arguments are supported:

- `datacenter` - (Optional) The Consul datacenter to query. Defaults to the same value found in `query_options` parameter specified below, or if that is empty, the `datacenter` value found in the Consul agent that this provider is configured to talk to.
- `name` - (Required) The service name to select.
- `query_options` - (Optional) See below.
- `tag` - (Optional) A single tag that can be used to filter the list of nodes to return based on a single matching tag..

The `query_options` block supports the following:

- `allow_stale` - (Optional) When `true`, the default, allow responses from Consul servers that are followers.
- `require_consistent` - (Optional) When `true` force the client to perform a read on at least quorum servers and verify the result is the same. Defaults to `false`.
- `token` - (Optional) Specify the Consul ACL token to use when performing the request. This defaults to the same API token configured by the `consul` provider but may be overriden if necessary.
- `wait_index` - (Optional) Index number used to enable blocking quereis.
- `wait_time` - (Optional) Max time the client should wait for a blocking query to return.

## » Attributes Reference

The following attributes are exported:

- `datacenter` - The datacenter the keys are being read from to.
- `name` - The name of the service
- `tag` - The name of the tag used to filter the list of nodes in `service`.
- `service` - A list of nodes and details about each endpoint advertising a service. Each element in the list is a map of attributes that correspond to each individual node. The list of per-node attributes is detailed below.

The following is a list of the per-node `service` attributes:

- `create_index` - The index entry at which point this entry was added to the catalog.
- `modify_index` - The index entry at which point this entry was modified in the catalog.
- `node_address` - The address of the Consul node advertising the service.
- `node_id` - The Node ID of the Consul agent advertising the service.
- `node_meta` - Node meta data tag information, if any.
- `node_name` - The name of the Consul node.
- `address` - The IP address of the service. If the `ServiceAddress` in the Consul catalog is empty, this value is automatically populated with the `node_address` (the `Address` in the Consul Catalog).
- `enable_tag_override` - Whether service tags can be overridden on this service.
- `id` - A unique service instance identifier.
- `name` - The name of the service.
- `port` - Port number of the service.
- `tagged_addresses` - List of explicit LAN and WAN IP addresses for the agent.
- `tags` - List of tags for the service.

## » consul_catalog_services

The `consul_catalog_services` data source returns a list of Consul services that have been registered with the Consul cluster in a given datacenter. By specifying a different datacenter in the `query_options` it is possible to retrieve a list of services from a different WAN-attached Consul datacenter.

This data source is different from the `consul_catalog_service` (singular) data source, which provides a detailed response about a specific Consul service.

### » Example Usage

```
data "consul_catalog_services" "read-dc1" {
  query_options {
    # Optional parameter: implicitly uses the current datacenter of the agent
    datacenter = "dc1"
  }
}

# Set the description to a whitespace delimited list of the services
resource "example_resource" "app" {
  description = "${join(" ", data.consul_catalog_services.names)}"

  # ...
}
```

### » Argument Reference

The following arguments are supported:

- `datacenter` - (Optional) The Consul datacenter to query. Defaults to the same value found in `query_options` parameter specified below, or if that is empty, the `datacenter` value found in the Consul agent that this provider is configured to talk to.
- `query_options` - (Optional) See below.

The `query_options` block supports the following:

- `allow_stale` - (Optional) When `true`, the default, allow responses from Consul servers that are followers.
- `require_consistent` - (Optional) When `true` force the client to perform a read on at least quorum servers and verify the result is the same. Defaults to `false`.

- `token` - (Optional) Specify the Consul ACL token to use when performing the request. This defaults to the same API token configured by the `consul` provider but may be overriden if necessary.
- `wait_index` - (Optional) Index number used to enable blocking quereis.
- `wait_time` - (Optional) Max time the client should wait for a blocking query to return.

### » Attributes Reference

The following attributes are exported:

- `datacenter` - The datacenter the keys are being read from to.
- `names` - A list of the Consul services found. This will always contain the list of services found.
- `services.<service>` - For each name given, the corresponding attribute is a Terraform map of services and their tags. The value is an alphanumerically sorted, whitespace delimited set of tags associated with the service.
- `tags` - A map of the tags found for each service. If more than one service shares the same tag, unique service names will be joined by whitespace (this is the inverse of `services` and can be used to lookup the services that match a single tag).

## » consul_keys

The `consul_keys` resource reads values from the Consul key/value store. This is a powerful way dynamically set values in templates.

### » Example Usage

```
data "consul_keys" "app" {
  datacenter = "nyc1"
  token      = "abcd"

  # Read the launch AMI from Consul
  key {
    name    = "ami"
    path    = "service/app/launch_ami"
    default = "ami-1234"
  }
}

# Start our instance with the dynamic ami value
```

```
resource "aws_instance" "app" {
  ami = "${data.consul_keys.app.var.ami}"

  # ...
}
```

### » Argument Reference

The following arguments are supported:

- `datacenter` - (Optional) The datacenter to use. This overrides the datacenter in the provider setup and the agent's default datacenter.
- `token` - (Optional) The ACL token to use. This overrides the token that the agent provides by default.
- `key` - (Required) Specifies a key in Consul to be read or written. Supported values documented below.

The `key` block supports the following:

- `name` - (Required) This is the name of the key. This value of the key is exposed as `var.<name>`. This is not the path of the key in Consul.
- `path` - (Required) This is the path in Consul that should be read or written to.
- `default` - (Optional) This is the default value to set for `var.<name>` if the key does not exist in Consul. Defaults to an empty string.

### » Attributes Reference

The following attributes are exported:

- `datacenter` - The datacenter the keys are being read from to.
- `var.<name>` - For each name given, the corresponding attribute has the value of the key.

## » consul_agent_service

Provides access to the agent service data in Consul. This can be used to define a service associated with a particular agent. Currently, defining health checks for an agent service is not supported.

### » Example Usage

```
resource "consul_agent_service" "app" {
  address = "www.google.com"
  name    = "google"
  port    = 80
  tags    = ["tag0", "tag1"]
}
```

### » Argument Reference

The following arguments are supported:

- `address` - (Optional) The address of the service. Defaults to the address of the agent.
- `name` - (Required) The name of the service.
- `port` - (Optional) The port of the service.
- `tags` - (Optional) A list of values that are opaque to Consul, but can be used to distinguish between services or nodes.

### » Attributes Reference

The following attributes are exported:

- `address` - The address of the service.
- `id` - The ID of the service, defaults to the value of `name`.
- `name` - The name of the service.
- `port` - The port of the service.
- `tags` - The tags of the service.

## » consul_catalog_entry

Registers a node or service with the Consul Catalog. Currently, defining health checks is not supported.

### » Example Usage

```
resource "consul_catalog_entry" "app" {
  address = "192.168.10.10"
  node    = "foobar"
```

```
  service = {
    address = "127.0.0.1"
    id      = "redis1"
    name    = "redis"
    port    = 8000
    tags    = ["master", "v1"]
  }
}
```

## » Argument Reference

The following arguments are supported:

- `address` - (Required) The address of the node being added to, or referenced in the catalog.
- `node` - (Required) The name of the node being added to, or referenced in the catalog.
- `service` - (Optional) A service to optionally associated with the node. Supported values are documented below.
- `datacenter` - (Optional) The datacenter to use. This overrides the datacenter in the provider setup and the agent's default datacenter.
- `token` - (Optional) ACL token.

The `service` block supports the following:

- `address` - (Optional) The address of the service. Defaults to the node address.
- `id` - (Optional) The ID of the service. Defaults to the `name`.
- `name` - (Required) The name of the service
- `port` - (Optional) The port of the service.
- `tags` - (Optional) A list of values that are opaque to Consul, but can be used to distinguish between services or nodes.

## » Attributes Reference

The following attributes are exported:

- `address` - The address of the service.
- `node` - The ID of the service, defaults to the value of `name`.

# » consul_keys

The `consul_keys` resource writes sets of individual values into Consul. This is a powerful way to expose infrastructure details to clients.

This resource manages individual keys, and thus it can create, update and delete the keys explicitly given. However, it is not able to detect and remove additional keys that have been added by non-Terraform means. To manage *all* keys sharing a common prefix, and thus have Terraform remove errant keys not present in the configuration, consider using the `consul_key_prefix` resource instead.

## » Example Usage

```
resource "consul_keys" "app" {
  datacenter = "nyc1"
  token      = "abcd"

  # Set the CNAME of our load balancer as a key
  key {
    path  = "service/app/elb_address"
    value = "${aws_elb.app.dns_name}"
  }
}
```

## » Argument Reference

The following arguments are supported:

- `datacenter` - (Optional) The datacenter to use. This overrides the datacenter in the provider setup and the agent's default datacenter.
- `token` - (Optional) The ACL token to use. This overrides the token that the agent provides by default.
- `key` - (Required) Specifies a key in Consul to be written. Supported values documented below.

The `key` block supports the following:

- `path` - (Required) This is the path in Consul that should be written to.
- `value` - (Required) The value to write to the given path.
- `delete` - (Optional) If true, then the key will be deleted when either its configuration block is removed from the configuration or the entire resource is destroyed. Otherwise, it will be left in Consul. Defaults to false.

### » Deprecated `key` arguments

Prior to Terraform 0.7, this resource was used both to read *and* write the Consul key/value store. The read functionality has moved to the `consul_keys` *data source*, whose documentation can be found via the navigation.

The pre-0.7 interface for reading keys is still supported for backward compatibility, but will be removed in a future version of Terraform.

## » Attributes Reference

The following attributes are exported:

- `datacenter` - The datacenter the keys are being written to.

# » consul_key_prefix

Allows Terraform to manage a "namespace" of Consul keys that share a common name prefix.

Like `consul_keys`, this resource can write values into the Consul key/value store, but *unlike* `consul_keys` this resource can detect and remove extra keys that have been added some other way, thus ensuring that rogue data added outside of Terraform will be removed on the next run.

This resource is thus useful in the case where Terraform is exclusively managing a set of related keys.

To avoid accidentally clobbering matching data that existed in Consul before a `consul_key_prefix` resource was created, creation of a key prefix instance will fail if any matching keys are already present in the key/value store. If any conflicting data is present, you must first delete it manually.

**Warning** After this resource is instantiated, Terraform takes control over *all* keys with the given path prefix, and will remove any matching keys that are not present in the configuration. It will also delete *all* keys under the given prefix when a `consul_key_prefix` resource is destroyed, even if those keys were created outside of Terraform.

## » Example Usage

```
resource "consul_key_prefix" "myapp_config" {
  datacenter = "nyc1"
  token      = "abcd"
```

```
  # Prefix to add to prepend to all of the subkey names below.
  path_prefix = "myapp/config/"

  subkeys = {
    "elb_cname"         = "${aws_elb.app.dns_name}"
    "s3_bucket_name"    = "${aws_s3_bucket.app.bucket}"
    "database/hostname" = "${aws_db_instance.app.address}"
    "database/port"     = "${aws_db_instance.app.port}"
    "database/username" = "${aws_db_instance.app.username}"
    "database/password" = "${aws_db_instance.app.password}"
    "database/name"     = "${aws_db_instance.app.name}"
  }
}
```

### » Argument Reference

The following arguments are supported:

- `datacenter` - (Optional) The datacenter to use. This overrides the datacenter in the provider setup and the agent's default datacenter.
- `token` - (Optional) The ACL token to use. This overrides the token that the agent provides by default.
- `path_prefix` - (Required) Specifies the common prefix shared by all keys that will be managed by this resource instance. In most cases this will end with a slash, to manage a "folder" of keys.
- `subkeys` - (Required) A mapping from subkey name (which will be appended to the given `path_prefix`) to the value that should be stored at that key. Use slashes, as shown in the above example, to create "subfolders" under the given path prefix.

### » Attributes Reference

The following attributes are exported:

- `datacenter` - The datacenter the keys are being read/written to.

## » consul_node

Provides access to Node data in Consul. This can be used to define a node. Currently, defining health checks is not supported.

### » Example Usage

```
resource "consul_node" "foobar" {
  address = "192.168.10.10"
  name    = "foobar"
}
```

### » Argument Reference

The following arguments are supported:

- `address` - (Required) The address of the node being added to, or referenced in the catalog.
- `name` - (Required) The name of the node being added to, or referenced in the catalog.

### » Attributes Reference

The following attributes are exported:

- `address` - The address of the service.
- `name` - The name of the service.

## » consul_prepared_query

Allows Terraform to manage a Consul prepared query.

Managing prepared queries is done using Consul's REST API. This resource is useful to provide a consistent and declarative way of managing prepared queries in your Consul cluster using Terraform.

### » Example Usage

```
# Creates a prepared query myquery.query.consul that finds the nearest
# healthy myapp.service.consul instance that has the active tag and not
# the standby tag.
resource "consul_prepared_query" "myapp-query" {
  name         = "myquery"
  datacenter   = "us-central1"
  token        = "abcd"
  stored_token = "wxyz"
  only_passing = true
```

```
  near         = "_agent"

  service = "myapp"
  tags    = ["active", "!standby"]

  failover {
    nearest_n   = 3
    datacenters = ["us-west1", "us-east-2", "asia-east1"]
  }

  dns {
    ttl = "30s"
  }
}

# Creates a Prepared Query Template that matches *-near-self.query.consul
# and finds the nearest service that matches the glob character (e.g.
# foo-near-self.query.consul will find the nearest healthy foo.service.consul).
resource "consul_prepared_query" "service-near-self" {
  datacenter   = "nyc1"
  token        = "abcd"
  stored_token = "wxyz"
  name         = ""
  only_passing = true
  near         = "_agent"

  template {
    type   = "name_prefix_match"
    regexp = "^(.*)-near-self$"
  }

  service = "$${match(1)}"

  failover {
    nearest_n   = 3
    datacenters = ["dc2", "dc3", "dc4"]
  }

  dns {
    ttl = "5m"
  }
}
```

## » Argument Reference

The following arguments are supported:

- `datacenter` - (Optional) The datacenter to use. This overrides the datacenter in the provider setup and the agent's default datacenter.
- `token` - (Optional) The ACL token to use when saving the prepared query. This overrides the token that the agent provides by default.
- `stored_token` - (Optional) The ACL token to store with the prepared query. This token will be used by default whenever the query is executed.
- `name` - (Required) The name of the prepared query. Used to identify the prepared query during requests. Can be specified as an empty string to configure the query as a catch-all.
- `service` - (Required) The name of the service to query.
- `session` - (Optional) The name of the Consul session to tie this query's lifetime to. This is an advanced parameter that should not be used without a complete understanding of Consul sessions and the implications of their use (it is recommended to leave this blank in nearly all cases). If this parameter is omitted the query will not expire.
- `tags` - (Optional) The list of required and/or disallowed tags. If a tag is in this list it must be present. If the tag is preceded with a "!" then it is disallowed.
- `only_passing` - (Optional) When `true`, the prepared query will only return nodes with passing health checks in the result.
- `near` - (Optional) Allows specifying the name of a node to sort results near using Consul's distance sorting and network coordinates. The magic `_agent` value can be used to always sort nearest the node servicing the request.
- `failover` - (Optional) Options for controlling behavior when no healthy nodes are available in the local DC.
  - `nearest_n` - (Optional) Return results from this many datacenters, sorted in ascending order of estimated RTT.
  - `datacenters` - (Optional) Remote datacenters to return results from.
- `dns` - (Optional) Settings for controlling the DNS response details.
  - `ttl` - (Optional) The TTL to send when returning DNS results.
- `template` - (Optional) Query templating options. This is used to make a single prepared query respond to many different requests.
  - `type` - (Required) The type of template matching to perform. Currently only `name_prefix_match` is supported.

  - `regexp` - (Required) The regular expression to match with. When using `name_prefix_match`, this regex is applied against the query name.

### » Attributes Reference

The following attributes are exported:

- `id` - The ID of the prepared query, generated by Consul.

## » consul_service

A high-level resource for creating a Service in Consul. Currently, defining health checks for a service is not supported.

**Most users should not use this resource**. When using Consul with compute instances, it's better to install the Consul Agent on these machines and register services via the agent. This ensures that services get assigned to the appropriate Consul "nodes" and allows service health to integrate with general node health as reported by the agent.

To register a non-compute resource, such as a hosted database, as a service, as described in Consul's *External Services* guide, use `consul_catalog_entry` instead, which can create an arbitrary service record in the Consul catalog.

### » Example Usage

```
resource "consul_service" "google" {
  address = "www.google.com"
  name    = "google"
  port    = 80
  tags    = ["tag0", "tag1"]
}
```

### » Argument Reference

The following arguments are supported:

- `service_id` - (Optional, string) The ID of the service, defaults to the value of `name` if not supplied.
- `address` - (Optional, string) The address of the service. Defaults to the address of the agent.

- `name` - (Required, string) The name of the service.
- `port` - (Optional, int) The port of the service.
- `tags` - (Optional, set of strings) A list of values that are opaque to Consul, but can be used to distinguish between services or nodes.

## » Attributes Reference

The following attributes are exported:

- `service_id` - The id of the service, defaults to the value of `name`.
- `address` - The address of the service.
- `name` - The name of the service.
- `port` - The port of the service.
- `tags` - The tags of the service.