Circonus Provider

The Circonus provider gives the ability to manage a Circonus account.

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

Usage

```
provider "circonus" {
  key = "b8fec159-f9e5-4fe6-ad2c-dc1ec6751586"
}
```

Argument Reference

The following arguments are supported:

- `key` - (Required) The Circonus API Key. It can be sourced from the `CIRCONUS_API_KEY` environment variable.
- `api_url` - (Optional) The API URL to use to talk with. The default is `https://api.circonus.com/v2`. It can be sourced from the `CIRCONUS_API_URL` environment variable.

circonus_account

`circonus_account` provides details (https://login.circonus.com/resources/api/calls/account) about a specific Circonus Account (https://login.circonus.com/user/docs/Administration/Account).

The `circonus_account` data source can be used for pulling various attributes about a specific Circonus Account.

Example Usage

The following example shows how the resource might be used to obtain the metrics usage and limit of a given Circonus Account.

```
data "circonus_account" "current" {
  current = true
}
```

Argument Reference

The arguments of this data source act as filters for querying the available regions. The given filters must match exactly one region whose data will be exported as attributes.

- `id` - (Optional) The Circonus ID of a given account.
- `current` - (Optional) Automatically use the current Circonus Account attached to the API token making the request.

At least one of the above attributes should be provided when searching for a account.

Attributes Reference

The following attributes are exported:

- `address1` - The first line of the address associated with the account.
- `address2` - The second line of the address associated with the account.
- `cc_email` - An optionally specified email address used in the CC line of invoices.
- `id` - The Circonus ID of the selected Account.
- `city` - The city part of the address associated with the account.
- `contact_groups` - A list of IDs for each contact group in the account.
- `country` - The country of the user's address.
- `description` - Description of the account.
- `invites` - An list of users invited to use the platform. Each element in the list has both an `email` and `role` attribute.
- `name` - The name of the account.

- `owner` - The Circonus ID of the user who owns this account.
- `state` - The state or province of the address associated with the account.
- `timezone` - The timezone that events will be displayed in the web interface for this account.
- `ui_base_url` - The base URL of this account.
- `usage` - A list of account usage limits. Each element in the list will have a `limit` attribute, a limit `type`, and a `used` attribute.
- `users` - A list of users who have access to this account. Each element in the list has both an `id` and a `role`. The `id` is a Circonus ID referencing the user.

circonus_collector

`circonus_collector` provides details (https://login.circonus.com/resources/api/calls/broker) about a specific Circonus Collector (https://login.circonus.com/user/docs/Administration/Brokers).

As well as validating a given Circonus ID, this resource can be used to discover the additional details about a collector configured within the provider. The results of a `circonus_collector` API call can return more than one collector per Circonus ID. Details of each individual collector in the group of collectors can be found via the `details` attribute described below.

NOTE regarding `cirocnus_collector`: The `circonus_collector` data source actually queries and operates on Circonus "brokers" at the broker group level. The `circonus_collector` is simply a renamed Circonus "broker" to make it clear what the function of the "broker" actually does: act as a fan-in agent that either pulls or has metrics pushed into it and funneled back through Circonus.

Example Usage

The following example shows how the resource might be used to obtain the name of the Circonus Collector configured on the provider.

```
data "circonus_collector" "ashburn" {
  id = "/broker/1"
}
```

Argument Reference

The arguments of this data source act as filters for querying the available regions. The given filters must match exactly one region whose data will be exported as attributes.

- `id` - (Optional) The Circonus ID of a given collector.

At least one of the above attributes should be provided when searching for a collector.

Attributes Reference

The following attributes are exported:

- `id` - The Circonus ID of the selected Collector.
- `details` - A list of details about the individual Collector instances that make up the group of collectors. See below for a list of attributes within each collector.
- `latitude` - The latitude of the selected Collector.
- `longitude` - The longitude of the selected Collector.
- `name` - The name of the selected Collector.

- `tags` - A list of tags assigned to the selected Collector.
- `type` - The of the selected Collector. This value is either `circonus` for a Circonus-managed, public Collector, or `enterprise` for a private collector that is private to an account.

Collector Details

- `cn` - The CN of an individual Collector in the Collector Group.
- `external_host` - The external host information for an individual Collector in the Collector Group. This is useful or important when talking with a Collector through a NAT'ing firewall.
- `external_port` - The external port number for an individual Collector in the Collector Group. This is useful or important when talking with a Collector through a NAT'ing firewall.
- `ip` - The IP address of an individual Collector in the Collector Group. This is the IP address of the interface listening on the network.
- `min_version` - ??
- `modules` - A list of what modules (types of checks) this collector supports.
- `port` - The port the collector responds to the Circonus HTTPS REST wire protocol on.
- `skew` - The clock drift between this collector and the Circonus server.
- `status` - The status of this particular collector. A string containing either `active`, `unprovisioned`, `pending`, `provisioned`, or `retired`.
- `version` - The version of the collector software the collector is running.

circonus_check

The `circonus_check` resource creates and manages a Circonus Check (https://login.circonus.com/resources/api/calls/check_bundle).

NOTE regarding `circonus_check` vs a Circonus Check Bundle: The `circonus_check` resource is implemented in terms of a Circonus Check Bundle (https://login.circonus.com/resources/api/calls/check_bundle). The `circonus_check` creates a higher-level abstraction over the implementation of a Check Bundle. As such, the naming and structure does not map 1:1 with the underlying Circonus API.

Usage

```
variable api_token {
  default = "my-token"
}

resource "circonus_check" "usage" {
  name = "Circonus Usage Check"

  notes = <<-EOF
A check to extract a usage metric.
EOF

  collector {
    id = "/broker/1"
  }

  metric {
    name = "${circonus_metric.used.name}"
    tags = "${circonus_metric.used.tags}"
    type = "${circonus_metric.used.type}"
    unit = "${circonus_metric.used.unit}"
  }

  json {
    url = "https://api.circonus.com/v2"

    headers = {
      Accept                = "application/json"
      X-Circonus-App-Name   = "TerraformCheck"
      X-Circonus-Auth-Token = "${var.api_token}"
    }
  }

  period       = 60
  tags         = ["source:circonus", "author:terraform"]
  timeout      = 10
}

resource "circonus_metric" "used" {
  name = "_usage`0`_used"
  type = "numeric"
  unit = "qty"

  tags = {
    source = "circonus"
  }
}
```

Argument Reference

- `active` - (Optional) Whether or not the check is enabled or not (default `true`).
- `caql` - (Optional) A Circonus Analytics Query Language (CAQL) (https://login.circonus.com/user/docs/CAQL) check. See below for details on how to configure a `caql` check.
- `cloudwatch` - (Optional) A CloudWatch check (https://login.circonus.com/user/docs/Data/CheckTypes/CloudWatch) check. See below for details on how to configure a `cloudwatch` check.
- `collector` - (Required) A collector ID. The collector(s) that are responsible for running a `circonus_check`. The `id` can be the Circonus ID for a Circonus collector (a.k.a. "broker") running in the cloud or an enterprise collector running in

your datacenter. One collection of metrics will be automatically created for each `collector` specified.

- `consul` - (Optional) A native Consul check. See below for details on how to configure a `consul` check.
- `http` - (Optional) A poll-based HTTP check. See below for details on how to configure the `http` check.
- `httptrap` - (Optional) An push-based HTTP check. This check method expects clients to send a specially crafted HTTP JSON payload. See below for details on how to configure the `httptrap` check.
- `icmp_ping` - (Optional) An ICMP ping check. See below for details on how to configure the `icmp_ping` check.
- `json` - (Optional) A JSON check. See below for details on how to configure the `json` check.
- `metric` - (Required) A list of one or more `metric` configurations. All metrics obtained from this check instance will be available as individual metric streams. See below for a list of supported `metric` attrbutes.
- `metric_limit` - (Optional) Setting a metric limit will tell the Circonus backend to periodically look at the check to see if there are additional metrics the collector has seen that we should collect. It will not reactivate metrics previously collected and then marked as inactive. Values are `0` to disable, `-1` to enable all metrics or `N+` to collect up to the value `N` (both `-1` and `N+` can not exceed other account restrictions).
- `mysql` - (Optional) A MySQL check. See below for details on how to configure the `mysql` check.
- `name` - (Optional) The name of the check that will be displayed in the web interface.
- `notes` - (Optional) Notes about this check.
- `period` - (Optional) The period between each time the check is made in seconds.
- `postgresql` - (Optional) A PostgreSQL check. See below for details on how to configure the `postgresql` check.
- `statsd` - (Optional) A statsd check. See below for details on how to configure the `statsd` check.
- `tags` - (Optional) A list of tags assigned to this check.
- `target` - (Required) A string containing the location of the thing being checked. This value changes based on the check type. For example, for an `http` check type this would be the URL you're checking. For a DNS check it would be the hostname you wanted to look up.
- `tcp` - (Optional) A TCP check. See below for details on how to configure the `tcp` check (includes TLS support).
- `timeout` - (Optional) A floating point number representing the maximum number of seconds this check should wait for a result. Defaults to `10.0`.

Supported `metric` Attributes

The following attributes are available within a `metric`.

- `active` - (Optional) Whether or not the metric is active or not. Defaults to `true`.
- `name` - (Optional) The name of the metric. A string containing freeform text.
- `tags` - (Optional) A list of tags assigned to the metric.
- `type` - (Required) A string containing either `numeric`, `text`, `histogram`, `composite`, or `caql`.
- `units` - (Optional) The unit of measurement the metric represents (e.g., bytes, seconds, milliseconds). A string

containing freeform text.

Supported Check Types

Circonus supports a variety of different checks. Each check type has its own set of options that must be configured. Each check type conflicts with every other check type (i.e. a `circonus_check` configured for a `json` check will conflict with all other check types, therefore a `postgresql` check must be a different `circonus_check` resource).

`caql` Check Type Attributes

- `query` - (Required) The CAQL Query (https://login.circonus.com/user/docs/caql_reference) to run.

Available metrics depend on the payload returned in the `caql` check. See the `caql` check type (https://login.circonus.com/resources/api/calls/check_bundle) for additional details.

`cloudwatch` Check Type Attributes

- `api_key` - (Required) The AWS access key. If this value is not explicitly set, this value is populated by the environment variable `AWS_ACCESS_KEY_ID`.
- `api_secret` - (Required) The AWS secret key. If this value is not explicitly set, this value is populated by the environment variable `AWS_SECRET_ACCESS_KEY`.
- `dimmensions` - (Required) A map of the CloudWatch dimmensions to include in the check.
- `metric` - (Required) A list of metric names to collect in this check.
- `namespace` - (Required) The namespace to pull parameters from.
- `url` - (Required) The AWS URL to pull from. This should be set to the region-specific endpoint (e.g. prefer `https://monitoring.us-east-1.amazonaws.com` over `https://monitoring.amazonaws.com`).
- `version` - (Optional) The version of the Cloudwatch API to use. Defaults to `2010-08-01`.

Available metrics depend on the payload returned in the `cloudwatch` check. See the `cloudwatch` check type (https://login.circonus.com/resources/api/calls/check_bundle) for additional details. The `circonus_check` `period` attribute must be set to either `60s` or `300s` for CloudWatch metrics.

Example CloudWatch check (partial metrics collection):

```
variable "cloudwatch_rds_tags" {
  type = "list"
  default = [
    "app:postgresql",
    "app:rds",
    "source:cloudwatch",
  ]
}

resource "circonus_check" "rds_metrics" {
  active = true
  name = "Terraform test: RDS Metrics via CloudWatch"
  notes = "Collect RDS metrics"
  period = "60s"

  collector {
    id = "/broker/1"
  }

  cloudwatch {
    dimmensions = {
      DBInstanceIdentifier = "my-db-name",
    }

    metric = [
      "CPUUtilization",
      "DatabaseConnections",
    ]

    namespace = "AWS/RDS"
    url = "https://monitoring.us-east-1.amazonaws.com"
  }

  metric {
    name = "CPUUtilization"
    tags = [ "${var.cloudwatch_rds_tags}" ]
    type = "numeric"
    unit = "%"
  }

  metric {
    name = "DatabaseConnections"
    tags = [ "${var.cloudwatch_rds_tags}" ]
    type = "numeric"
    unit = "connections"
  }
}
```

`consul` Check Type Attributes

- `acl_token` - (Optional) An ACL Token authenticate the API request. When an ACL Token is set, this value is transmitted as an HTTP Header in order to not show up in any logs. The default value is an empty string.
- `allow_stale` - (Optional) A boolean value that indicates whether or not this check should require the health information come from the Consul leader node. For scalability reasons, this value defaults to `false`. See below for details on detecting the staleness of health information.
- `ca_chain` - (Optional) A path to a file containing all the certificate authorities that should be loaded to validate the remote certificate (required when `http_addr` is a TLS-enabled endpoint).

- `certificate_file` - (Optional) A path to a file containing the client certificate that will be presented to the remote server (required when `http_addr` is a TLS-enabled endpoint).

- `check_blacklist` - (Optional) A list of check names to exclude from the result of checks (i.e. no metrics will be generated by whose check name is in the `check_blacklist`). This blacklist is applied to the `node`, `service`, and `state` check modes.

- `ciphers` - (Optional) A list of ciphers to be used in the TLS protocol (only used when `http_addr` is a TLS-enabled endpoint).

- `dc` - (Optional) Explicitly name the Consul datacenter to use. The default value is an empty string. When an empty value is specified, the Consul datacenter of the agent at the `http_addr` is implicitly used.

- `headers` - (Optional) A map of the HTTP headers to be sent when executing the check. NOTE: the `headers` attribute is processed last and will takes precidence over any other derived value that is transmitted as an HTTP header to Consul (i.e. it is possible to override the `acl_token` by setting a headers value).

- `http_addr` - (Optional) The Consul HTTP endpoint to to query for health information. The default value is `http://consul.service.consul:8500`. The scheme must change from `http` to `https` when the endpoint has been TLS-enabled.

- `key_file` - (Optional) A path to a file containing key to be used in conjunction with the cilent certificate (required when `http_addr` is a TLS-enabled endpoint).

- `node` - (Optional) Check the health of this node. The value can be either a Consul Node ID (Consul Version >= 0.7.4) or Node Name. See also the `service_blacklist`, `node_blacklist`, and `check_blacklist` attributes. This attribute conflicts with the `service` and `state` attributes.

- `node_blacklist` - (Optional) A list of node IDs or node names to exclude from the results of checks (i.e. no metrics will be generated from nodes in the `node_blacklist`). This blacklist is applied to the `node`, `service`, and `state` check modes.

- `service` - (Optional) Check the cluster-wide health of this named service. See also the `service_blacklist`, `node_blacklist`, and `check_blacklist` attributes. This attribute conflicts with the `node` and `state` attributes.

- `service_blacklist` - (Optional) A list of service names to exclude from the result of checks (i.e. no metrics will be generated by services whose service name is in the `service_blacklist`). This blacklist is applied to the `node`, `service`, and `state` check modes.

- `state` - (Optional) A Circonus check to monitor Consul checks across the entire Consul cluster. This value may be either `passing`, `warning`, or `critical`. This `consul` check mode is intended to act as the cluster check of last resort. This check type is useful when first starting and is intended to act as a check of last resort before transitioning to explicitly defined checks for individual services or nodes. The metrics returned from check will be sorted based on the `CreateIndex` of the entry in order to have a stable set of metrics in the array of returned values. See also the `service_blacklist`, `node_blacklist`, and `check_blacklist` attributes. This attribute conflicts with the `node` and `state` attributes.

Available metrics depend on the consul check being performed (`node`, `service`, or `state`). In addition to the data avilable from the endpoints, the `consul` check also returns a set of metrics that are a variant of: `{Num,Pct}{,Passing,Warning,Critical}{Checks,Nodes,Services}` (see the `GLOB_BRACE` section of your local `glob(3)` documentation).

Example Consul check (partial metrics collection):

```
resource "circonus_check" "consul_server" {
  active = true
  name = "%s"
  period = "60s"

  collector {
    # Collector ID must be an Enterprise broker able to reach the Consul agent
    # listed in `http_addr`.
    id = "/broker/2110"
  }

  consul {
    service = "consul"

    # Other consul check modes:
    # node = "consul1"
    # state = "critical"
  }

  metric {
    name = "NumNodes"
    tags = [ "source:consul", "lifecycle:unittest" ]
    type = "numeric"
  }

  metric {
    name = "LastContact"
    tags = [ "source:consul", "lifecycle:unittest" ]
    type = "numeric"
    unit = "seconds"
  }

  metric {
    name = "Index"
    tags = [ "source:consul", "lifecycle:unittest" ]
    type = "numeric"
    unit = "transactions"
  }

  metric {
    name = "KnownLeader"
    tags = [ "source:consul", "lifecycle:unittest" ]
    type = "text"
  }

  tags = [ "source:consul", "lifecycle:unittest" ]
}
```

`http` Check Type Attributes

- `auth_method` - (Optional) HTTP Authentication method to use. When set must be one of the values `Basic`, `Digest`, or `Auto`.
- `auth_password` - (Optional) The password to use during authentication.
- `auth_user` - (Optional) The user to authenticate as.
- `body_regexp` - (Optional) This regular expression is matched against the body of the response. If a match is not found, the check will be marked as "bad."

- `ca_chain` - (Optional) A path to a file containing all the certificate authorities that should be loaded to validate the remote certificate (for TLS checks).
- `certificate_file` - (Optional) A path to a file containing the client certificate that will be presented to the remote server (for TLS checks).
- `ciphers` - (Optional) A list of ciphers to be used in the TLS protocol (for HTTPS checks).
- `code` - (Optional) The HTTP code that is expected. If the code received does not match this regular expression, the check is marked as "bad."
- `extract` - (Optional) This regular expression is matched against the body of the response globally. The first capturing match is the key and the second capturing match is the value. Each key/value extracted is registered as a metric for the check.
- `headers` - (Optional) A map of the HTTP headers to be sent when executing the check.
- `key_file` - (Optional) A path to a file containing key to be used in conjunction with the cilent certificate (for TLS checks).
- `method` - (Optional) The HTTP Method to use. Defaults to `GET`.
- `payload` - (Optional) The information transferred as the payload of an HTTP request.
- `read_limit` - (Optional) Sets an approximate limit on the data read (`0` means no limit). Default `0`.
- `redirects` - (Optional) The maximum number of HTTP `Location` header redirects to follow. Default `0`.
- `url` - (Required) The target for this `json` check. The `url` must include the scheme, host, port (optional), and path to use (e.g. `https://app1.example.org/healthz`)
- `version` - (Optional) The HTTP version to use. Defaults to `1.1`.

Available metrics include: `body_match`, `bytes`, `cert_end`, `cert_end_in`, `cert_error`, `cert_issuer`, `cert_start`, `cert_subject`, `code`, `duration`, `truncated`, `tt_connect`, and `tt_firstbyte`. See the `http` check type (https://login.circonus.com/resources/api/calls/check_bundle) for additional details.

`httptrap` Check Type Attributes

- `async_metrics` - (Optional) Boolean value specifies whether or not httptrap metrics are logged immediately or held until the status message is to be emitted. Default `false`.
- `secret` - (Optional) Specify the secret with which metrics may be submitted.

Available metrics depend on the payload returned in the `httptrap` doc. See the `httptrap` check type (https://login.circonus.com/resources/api/calls/check_bundle) for additional details.

`json` Check Type Attributes

- `auth_method` - (Optional) HTTP Authentication method to use. When set must be one of the values `Basic`, `Digest`, or `Auto`.
- `auth_password` - (Optional) The password to use during authentication.

- `auth_user` - (Optional) The user to authenticate as.
- `ca_chain` - (Optional) A path to a file containing all the certificate authorities that should be loaded to validate the remote certificate (for TLS checks).
- `certificate_file` - (Optional) A path to a file containing the client certificate that will be presented to the remote server (for TLS checks).
- `ciphers` - (Optional) A list of ciphers to be used in the TLS protocol (for HTTPS checks).
- `headers` - (Optional) A map of the HTTP headers to be sent when executing the check.
- `key_file` - (Optional) A path to a file containing key to be used in conjunction with the cilent certificate (for TLS checks).
- `method` - (Optional) The HTTP Method to use. Defaults to `GET`.
- `port` - (Optional) The TCP Port number to use. Defaults to `81`.
- `read_limit` - (Optional) Sets an approximate limit on the data read (`0` means no limit). Default `0`.
- `redirects` - (Optional) The maximum number of HTTP `Location` header redirects to follow. Default `0`.
- `url` - (Required) The target for this `json` check. The `url` must include the scheme, host, port (optional), and path to use (e.g. `https://app1.example.org/healthz`)
- `version` - (Optional) The HTTP version to use. Defaults to `1.1`.

Available metrics depend on the payload returned in the `json` doc. See the `json` check type (https://login.circonus.com/resources/api/calls/check_bundle) for additional details.

`icmp_ping` Check Type Attributes

The `icmp_ping` check requires the `target` top-level attribute to be set.

- `availability` - (Optional) The percentage of ping packets that must be returned for this measurement to be considered successful. Defaults to `100.0`.
- `count` - (Optional) The number of ICMP ping packets to send. Defaults to `5`.
- `interval` - (Optional) Interval between packets. Defaults to `2s`.

Available metrics include: `available`, `average`, `count`, `maximum`, and `minimum`. See the `ping_icmp` check type (https://login.circonus.com/resources/api/calls/check_bundle) for additional details.

`mysql` Check Type Attributes

The `mysql` check requires the `target` top-level attribute to be set.

- `dsn` - (Required) The MySQL DSN/connect string (https://github.com/go-sql-driver/mysql/blob/master/README.md) to use to talk to MySQL.
- `query` - (Required) The SQL query to execute.

`postgresql` Check Type Attributes

The `postgresql` check requires the `target` top-level attribute to be set.

- `dsn` - (Required) The PostgreSQL DSN/connect string (https://www.postgresql.org/docs/current/static/libpq-connect.html) to use to talk to PostgreSQL.
- `query` - (Required) The SQL query to execute.

Available metric names are dependent on the output of the `query` being run.

`statsd` Check Type Attributes

- `source_ip` - (Required) Any statsd messages from this IP address (IPv4 or IPv6) will be associated with this check.

Available metrics depend on the metrics sent to the `statsd` check.

`tcp` Check Type Attributes

- `banner_regexp` - (Optional) This regular expression is matched against the response banner. If a match is not found, the check will be marked as bad.
- `ca_chain` - (Optional) A path to a file containing all the certificate authorities that should be loaded to validate the remote certificate (for TLS checks).
- `certificate_file` - (Optional) A path to a file containing the client certificate that will be presented to the remote server (for TLS checks).
- `ciphers` - (Optional) A list of ciphers to be used in the TLS protocol (for HTTPS checks).
- `host` - (Required) Hostname or IP address of the host to connect to.
- `key_file` - (Optional) A path to a file containing key to be used in conjunction with the cilent certificate (for TLS checks).
- `port` - (Required) Integer specifying the port on which the management interface can be reached.
- `tls` - (Optional) When enabled establish a TLS connection.

Available metrics include: `banner`, `banner_match`, `cert_end`, `cert_end_in`, `cert_error`, `cert_issuer`, `cert_start`, `cert_subject`, `duration`, `tt_connect`, `tt_firstbyte`. See the `tcp` check type (https://login.circonus.com/resources/api/calls/check_bundle) for additional details.

Sample `tcp` check:

```
resource "circonus_check" "tcp_check" {
  name = "TCP and TLS check"
  notes = "Obtains the connect time and TTL for the TLS cert"
  period = "60s"

  collector {
    id = "/broker/1"
  }

  tcp {
    host = "127.0.0.1"
    port = 443
    tls = true
  }

  metric {
    name = "cert_end_in"
    tags = [ "${var.tcp_check_tags}" ]
    type = "numeric"
    unit = "seconds"
  }

  metric {
    name = "tt_connect"
    tags = [ "${var.tcp_check_tags}" ]
    type = "numeric"
    unit = "miliseconds"
  }

  tags = [ "${var.tcp_check_tags}" ]
}
```

Out Parameters

- `check_by_collector` - Maps the ID of the collector (`collector_id`, the map key) to the `check_id` (value) that is registered to a collector.
- `check_id` - If there is only one `collector` specified for the check, this value will be populated with the `check_id`. If more than one `collector` is specified in the check, then this value will be an empty string. `check_by_collector` will always be populated.
- `checks` - List of `check_ids` created by this `circonus_check`. There is one element in this list per collector specified in the check.
- `created` - UNIX time at which this check was created.
- `last_modified` - UNIX time at which this check was last modified.
- `last_modified_by` - User ID in Circonus who modified this check last.
- `reverse_connect_urls` - Only relevant to Circonus support.
- `uuids` - List of Check `uuids` created by this `circonus_check`. There is one element in this list per collector specified in the check.

Import Example

`circonus_check` supports importing resources. Supposing the following Terraform (and that the referenced `circonus_metric` (/docs/providers/circonus/r/metric.html) has already been imported):

```
provider "circonus" {
  alias = "b8fec159-f9e5-4fe6-ad2c-dc1ec6751586"
}

resource "circonus_metric" "used" {
  name = "_usage`0`_used"
  type = "numeric"
}

resource "circonus_check" "usage" {
  collector {
    id = "/broker/1"
  }

  json {
    url = "https://api.circonus.com/account/current"

    headers = {
      "Accept"                = "application/json"
      "X-Circonus-App-Name"   = "TerraformCheck"
      "X-Circonus-Auth-Token" = "${var.api_token}"
    }
  }

  metric {
    name = "${circonus_metric.used.name}"
    type = "${circonus_metric.used.type}"
  }
}
```

It is possible to import a `circonus_check` resource with the following command:

```
$ terraform import circonus_check.usage ID
```

Where `ID` is the `_cid` or Circonus ID of the Check Bundle (e.g. `/check_bundle/12345`) and `circonus_check.usage` is the name of the resource whose state will be populated as a result of the command.

circonus_contact_group

The `circonus_contact_group` resource creates and manages a Circonus Contact Group (https://login.circonus.com/user/docs/Alerting/ContactGroups).

Usage

```
resource "circonus_contact_group" "myteam-alerts" {
  name = "MyTeam Alerts"

  email {
    user = "/user/1234"
  }

  email {
    user = "/user/5678"
  }

  email {
    address = "user@example.com"
  }

  http {
    address = "https://www.example.org/post/endpoint"
    format = "json"
    method = "POST"
  }

  irc {
    user = "/user/6331"
  }

  slack {
    channel = "#myteam"
    team = "T038UT13D"
  }

  sms {
    user = "/user/1234"
  }

  sms {
    address = "8005551212"
  }

  victorops {
    api_key = "xxxx"
    critical = 2
    info = 5
    team = "myteam"
    warning = 3
  }

  xmpp {
    user = "/user/9876"
  }

  aggregation_window = "5m"

  alert_option {
    severity = 1
    reminder = "5m"
    escalate_to = "/contact_group/4444"
  }

  alert_option {
    severity = 2
    reminder = "15m"
    escalate_after = "2h"
    escalate_to = "/contact_group/4444"
  }

  alert_option {
    severity = 3
    reminder = "24m"
    escalate_after = "3d"
    escalate_to = "/contact_group/4444"
  }
}
```

Argument Reference

- `aggregation_window` - (Optional) The aggregation window for batching up alert notifications.
- `alert_option` - (Optional) There is one `alert_option` per severity, where severity can be any number between 1 (high) and 5 (low). If configured, the alerting system will remind or escalate alerts to further contact groups if an alert sent to this contact group is not acknowledged or resolved. See below for details.
- `email` - (Optional) Zero or more `email` attributes may be present to dispatch email to Circonus users by referencing their user ID, or by specifying an email address. See below for details on supported attributes.
- `http` - (Optional) Zero or more `http` attributes may be present to dispatch Webhook/HTTP requests (https://login.circonus.com/user/docs/Alerting/ContactGroups#WebhookNotifications) by Circonus. See below for details on supported attributes.
- `irc` - (Optional) Zero or more `irc` attributes may be present to dispatch IRC notifications to users. See below for details on supported attributes.
- `long_message` - (Optional) The bulk of the message used in long form alert messages.
- `long_subject` - (Optional) The subject used in long form alert messages.
- `long_summary` - (Optional) The brief summary used in long form alert messages.
- `name` - (Required) The name of the contact group.
- `pager_duty` - (Optional) Zero or more `pager_duty` attributes may be present to dispatch to Pager Duty teams (https://login.circonus.com/user/docs/Alerting/ContactGroups#PagerDutyOptions). See below for details on supported attributes.
- `short_message` - (Optional) The subject used in short form alert messages.
- `short_summary` - (Optional) The brief summary used in short form alert messages.
- `slack` - (Optional) Zero or more `slack` attributes may be present to dispatch to Slack teams. See below for details on supported attributes.

- `sms` - (Optional) Zero or more `sms` attributes may be present to dispatch SMS messages to Circonus users by referencing their user ID, or by specifying an SMS Phone Number. See below for details on supported attributes.
- `tags` - (Optional) A list of tags attached to the Contact Group.
- `victorops` - (Optional) Zero or more `victorops` attributes may be present to dispatch to VictorOps teams (https://login.circonus.com/user/docs/Alerting/ContactGroups#VictorOps). See below for details on supported attributes.

Supported Contact Group `alert_option` Attributes

- `escalate_after` - (Optional) How long to wait before escalating an alert that is received at a given severity.
- `escalate_to` - (Optional) The Contact Group ID who will receive the escalation.
- `reminder` - (Optional) If specified, reminders will be sent after a user configurable number of minutes for open alerts.
- `severity` - (Required) An `alert_option` must be assigned to a given severity level. Valid severity levels range from 1 (highest severity) to 5 (lowest severity).

Supported Contact Group `email` Attributes

Either an `address` or `user` attribute is required.

- `address` - (Optional) A well formed email address.
- `user` - (Optional) An email will be sent to the email address of record for the corresponding user ID (e.g. `/user/1234`).

A `user`'s email address is automatically maintained and kept up to date by the recipient, whereas an `address` provides no automatic layer of indirection for keeping the information accurate (including LDAP and SAML-based authentication mechanisms).

Supported Contact Group `http` Attributes

- `address` - (Required) URL to send a webhook request to.
- `format` - (Optional) The payload of the request is a JSON-encoded payload when the `format` is set to `json` (the default). The alternate payload encoding is `params`.
- `method` - (Optional) The HTTP verb to use when making a request. Either `GET` or `POST` may be specified. The default verb is `POST`.

Supported Contact Group `irc` Attributes

- `user` - (Required) When a user has configured IRC on their user account, they will receive an IRC notification.

Supported Contact Group `pager_duty` Attributes

- `contact_group_fallback` - (Optional) If there is a problem contacting PagerDuty, relay the notification automatically to the specified Contact Group (e.g. `/contact_group/1234`).
- `service_key` - (Required) The PagerDuty Service Key.
- `webhook_url` - (Required) The PagerDuty webhook URL that PagerDuty uses to notify Circonus of acknowledged actions.

Supported Contact Group `slack` Attributes

- `contact_group_fallback` - (Optional) If there is a problem contacting Slack, relay the notification automatically to the specified Contact Group (e.g. `/contact_group/1234`).
- `buttons` - (Optional) Slack notifications can have acknowledgement buttons built into the notification message itself when enabled. Defaults to `true`.
- `channel` - (Required) Specify what Slack channel Circonus should send alerts to.
- `team` - (Required) Specify what Slack team Circonus should look in for the aforementioned `channel`.
- `username` - (Optional) Specify the username Circonus should advertise itself as in Slack. Defaults to `Circonus`.

Supported Contact Group `sms` Attributes

Either an `address` or `user` attribute is required.

- `address` - (Optional) SMS Phone Number to send a short notification to.
- `user` - (Optional) An SMS page will be sent to the phone number of record for the corresponding user ID (e.g. `/user/1234`).

A `user`'s phone number is automatically maintained and kept up to date by the recipient, whereas an `address` provides no automatic layer of indirection for keeping the information accurate (including LDAP and SAML-based authentication mechanisms).

Supported Contact Group `victorops` Attributes

- `contact_group_fallback` - (Optional) If there is a problem contacting VictorOps, relay the notification automatically to the specified Contact Group (e.g. `/contact_group/1234`).
- `api_key` - (Required) The API Key for talking with VictorOps.
- `critical` - (Required)
- `info` - (Required)
- `team` - (Required)

- `warning` - (Required)

Supported Contact Group `xmpp` Attributes

Either an `address` or `user` attribute is required.

- `address` - (Optional) XMPP address to send a short notification to.
- `user` - (Optional) An XMPP notification will be sent to the XMPP address of record for the corresponding user ID (e.g. `/user/1234`).

Import Example

`circonus_contact_group` supports importing resources. Supposing the following Terraform:

```
provider "circonus" {
  alias = "b8fec159-f9e5-4fe6-ad2c-dc1ec6751586"
}

resource "circonus_contact_group" "myteam" {
  name = "My Team's Contact Group"

  email {
    address = "myteam@example.com"
  }

  slack {
    channel = "#myteam"
    team = "T024UT03C"
  }
}
```

It is possible to import a `circonus_contact_group` resource with the following command:

```
$ terraform import circonus_contact_group.myteam ID
```

Where `ID` is the `_cid` or Circonus ID of the Contact Group (e.g. `/contact_group/12345`) and `circonus_contact_group.myteam` is the name of the resource whose state will be populated as a result of the command.

circonus_graph

The `circonus_graph` resource creates and manages a Circonus Graph (https://login.circonus.com/user/docs/Visualization/Graph/Create).

https://login.circonus.com/resources/api/calls/graph (https://login.circonus.com/resources/api/calls/graph)).

Usage

```
variable "myapp-tags" {
  type    = "list"
  default = [ "app:myapp", "owner:myteam" ]
}

resource "circonus_graph" "latency-graph" {
  name        = "Latency Graph"
  description = "A sample graph showing off two data points"
  notes       = "Misc notes about this graph"
  graph_style = "line"
  line_style  = "stepped"

  metric {
    check       = "${circonus_check.api_latency.checks[0]}"
    metric_name = "maximum"
    metric_type = "numeric"
    name        = "Maximum Latency"
    axis        = "left"
    color       = "#657aa6"
  }

  metric {
    check       = "${circonus_check.api_latency.checks[0]}"
    metric_name = "minimum"
    metric_type = "numeric"
    name        = "Minimum Latency"
    axis        = "right"
    color       = "#0000ff"
  }

  tags = [ "${var.myapp-tags}" ]
}
```

Argument Reference

- `description` - (Optional) Description of what the graph is for.
- `graph_style` - (Optional) How the graph should be rendered. Valid options are `area` or `line` (default).
- `left` - (Optional) A map of graph left axis options. Valid values in `left` include: `logarithmic` can be set to `0` (default) or `1`; `min` is the `min` Y axis value on the left; and `max` is the Y axis max value on the left.
- `line_style` - (Optional) How the line should change between points. Can be either `stepped` (default) or `interpolated`.
- `name` - (Required) The title of the graph.

- `notes` - (Optional) A place for storing notes about this graph.
- `right` - (Optional) A map of graph right axis options. Valid values in `right` include: `logarithmic` can be set to `0` (default) or `1`; `min` is the `min` Y axis value on the right; and `max` is the Y axis max value on the right.
- `metric` - (Optional) A list of metric streams to graph. See below for options.
- `metric_cluster` - (Optional) A metric cluster to graph. See below for options.
- `tags` - (Optional) A list of tags assigned to this graph.

`metric` Configuration

An individual metric stream is the underlying source of data points used for visualization in a graph. Either a `caql` attribute is required or a `check` and `metric` must be set. The `metric` attribute can have the following options set.

- `active` - (Optional) A boolean if the metric stream is enabled or not.
- `alpha` - (Optional) A floating point number between 0 and 1.
- `axis` - (Optional) The axis that the metric stream will use. Valid options are `left` (default) or `right`.
- `caql` - (Optional) A CAQL formula. Conflicts with the `check` and `metric` attributes.
- `check` - (Optional) The check that this metric stream belongs to.
- `color` - (Optional) A hex-encoded color of the line / area on the graph.
- `formula` - (Optional) Formula that should be aplied to both the values in the graph and the legend.
- `legend_formula` - (Optional) Formula that should be applied to values in the legend.
- `function` - (Optional) What derivative value, if any, should be used. Valid values are: `gauge` (default), `derive`, and `counter (_stddev)`
- `metric_type` - (Required) The type of the metric. Valid values are: `numeric`, `text`, `histogram`, `composite`, or `caql`.
- `name` - (Optional) A name which will appear in the graph legend.
- `metric_name` - (Optional) The name of the metric stream within the check to graph.
- `stack` - (Optional) If this metric is to be stacked, which stack set does it belong to (starting at `0`).

`metric_cluster` Configuration

A metric cluster selects multiple metric streams together dynamically using a query language and returns the set of matching metric streams as a single result set to the graph rendering engine.

- `active` - (Optional) A boolean if the metric cluster is enabled or not.
- `aggregate` - (Optional) The aggregate function to apply across this metric cluster to create a single value. Valid values are: `none` (default), `min`, `max`, `sum`, `mean`, or `geometric_mean`.
- `axis` - (Optional) The axis that the metric cluster will use. Valid options are `left` (default) or `right`.

- `color` - (Optional) A hex-encoded color of the line / area on the graph. This is a required attribute when `aggregate` is specified.
- `group` - (Optional) The `metric_cluster` that will provide datapoints for this graph.
- `name` - (Optional) A name which will appear in the graph legend for this metric cluster.

Import Example

`circonus_graph` supports importing resources. Supposing the following Terraform (and that the referenced `circonus_metric` (/docs/providers/circonus/r/metric.html) and `circonus_check` (/docs/providers/circonus/r/check.html) have already been imported):

```
resource "circonus_graph" "icmp-graph" {
  name        = "Test graph"
  graph_style = "line"
  line_style  = "stepped"

  metric {
    check       = "${circonus_check.api_latency.checks[0]}"
    metric_name = "maximum"
    metric_type = "numeric"
    name        = "Maximum Latency"
    axis        = "left"
  }
}
```

It is possible to import a `circonus_graph` resource with the following command:

```
$ terraform import circonus_graph.icmp-graph ID
```

Where `ID` is the `_cid` or Circonus ID of the graph (e.g. `/graph/bd72aabc-90b9-4039-cc30-c9ab838c18f5`) and `circonus_graph.icmp-graph` is the name of the resource whose state will be populated as a result of the command.

circonus_metric

The `circonus_metric` resource creates and manages a single metric resource (https://login.circonus.com/resources/api/calls/metric) that will be instantiated only once a referencing `circonus_check` has been created.

Usage

```
resource "circonus_metric" "used" {
  name  = "_usage`0`_used"
  type  = "numeric"
  units = "qty"

  tags = {
    author = "terraform"
    source = "circonus"
  }
}
```

Argument Reference

- `active` - (Optional) A boolean indicating if the metric is being filtered out at the `circonus_check`'s collector(s) or not.
- `name` - (Required) The name of the metric. A `name` must be unique within a `circonus_check` and its meaning is `circonus_check.type` specific.
- `tags` - (Optional) A list of tags assigned to the metric.
- `type` - (Required) The type of metric. This value must be present and can be one of the following values: `numeric`, `text`, `histogram`, `composite`, or `caql`.
- `unit` - (Optional) The unit of measurement for this `circonus_metric`.

Import Example

`circonus_metric` supports importing resources. Supposing the following Terraform:

```
provider "circonus" {
  alias = "b8fec159-f9e5-4fe6-ad2c-dc1ec6751586"
}

resource "circonus_metric" "usage" {
  name = "_usage`0`_used"
  type = "numeric"
  unit = "qty"
  tags = { source = "circonus" }
}
```

It is possible to import a `circonus_metric` resource with the following command:

```
$ terraform import circonus_metric.usage ID
```

Where `ID` is a random, never before used UUID and `circonus_metric.usage` is the name of the resource whose state will be populated as a result of the command.

circonus_metric_cluster

The `circonus_metric_cluster` resource creates and manages a Circonus Metric Cluster (https://login.circonus.com/user/docs/Data/View/MetricClusters).

Usage

```
resource "circonus_metric_cluster" "nomad-job-memory-rss" {
  name = "My Job's Resident Memory"
  description = <<-EOF
An aggregation of all resident memory metric streams across allocations in a Nomad job.
EOF

  query {
    definition = "*`nomad-jobname`memory`rss"
    type       = "average"
  }
  tags = ["source:nomad","resource:memory"]
}
```

Argument Reference

- `description` - (Optional) A long-form description of the metric cluster.
- `name` - (Required) The name of the metric cluster. This name must be unique across all metric clusters in a given Circonus Account.
- `query` - (Required) One or more `query` attributes must be present. Each `query` must contain both a `definition` and a `type`. See below for details on supported attributes.
- `tags` - (Optional) A list of tags attached to the metric cluster.

Supported Metric Cluster `query` Attributes

- `definition` - (Required) The definition of a metric cluster query (https://login.circonus.com/resources/api/calls/metric_cluster).
- `type` - (Required) The query type to execute per metric cluster. Valid query types are: `average`, `count`, `counter`, `counter2`, `counter2_stddev`, `counter_stddev`, `derive`, `derive2`, `derive2_stddev`, `derive_stddev`, `histogram`, `stddev`, `text`.

Out parameters

- `id` - ID of the Metric Cluster.

Import Example

`circonus_metric_cluster` supports importing resources. Supposing the following Terraform:

```
provider "circonus" {
  alias = "b8fec159-f9e5-4fe6-ad2c-dc1ec6751586"
}

resource "circonus_metric_cluster" "mymetriccluster" {
  name = "Metric Cluster for a particular metric in a job"

  query {
    definition = "*`nomad-jobname`memory`rss"
    type       = "average"
  }
}
```

It is possible to import a `circonus_metric_cluster` resource with the following command:

```
$ terraform import circonus_metric_cluster.mymetriccluster ID
```

Where `ID` is the `_cid` or Circonus ID of the Metric Cluster (e.g. `/metric_cluster/12345`) and `circonus_metric_cluster.mymetriccluster` is the name of the resource whose state will be populated as a result of the command.

circonus_rule_set

The `circonus_rule_set` resource creates and manages a Circonus Rule Set (https://login.circonus.com/resources/api/calls/rule_set).

Usage

```
variable "myapp-tags" {
  type    = "list"
  default = [ "app:myapp", "owner:myteam" ]
}

resource "circonus_rule_set" "myapp-cert-ttl-alert" {
  check       = "${circonus_check.myapp-https.checks[0]}"
  metric_name = "cert_end_in"
  link        = "https://wiki.example.org/playbook/how-to-renew-cert"

  if {
    value {
      min_value = "${2 * 24 * 3600}"
    }

    then {
      notify = [ "${circonus_contact_group.myapp-owners.id}" ]
      severity = 1
    }
  }

  if {
    value {
      min_value = "${7 * 24 * 3600}"
    }

    then {
      notify = [ "${circonus_contact_group.myapp-owners.id}" ]
      severity = 2
    }
  }

  if {
    value {
      min_value = "${21 * 24 * 3600}"
    }

    then {
      notify = [ "${circonus_contact_group.myapp-owners.id}" ]
      severity = 3
    }
  }

  if {
    value {
      absent = "24h"
    }

    then {
      notify = [ "${circonus_contact_group.myapp-owners.id}" ]
      severity = 1
    }
  }
  tags = [ "${var.myapp-tags}" ]
}

resource "circonus_rule_set" "myapp-healthy-alert" {
  check = "${circonus_check.myapp-https.checks[0]}"
  metric_name = "duration"
  link = "https://wiki.example.org/playbook/debug-down-app"

  if {
    value {
      # SEV1 if it takes more than 9.5s for us to complete an HTTP request
      max_value = "${9.5 * 1000}"
    }

    then {
      notify = [ "${circonus_contact_group.myapp-owners.id}" ]
      severity = 1
    }
  }

  if {
    value {
      # SEV2 if it takes more than 5s for us to complete an HTTP request
      max_value = "${5 * 1000}"
    }

    then {
      notify = [ "${circonus_contact_group.myapp-owners.id}" ]
      severity = 2
    }
  }

  if {
    value {
      # SEV3 if the average response time is more than 500ms using a moving
      # average over the last 10min.  Any transient problems should have
      # resolved themselves by now.  Something's wrong, need to page someone.
      over {
        last  = "10m"
        using = "average"
      }
      max_value = "500"
    }

    then {
      notify = [ "${circonus_contact_group.myapp-owners.id}" ]
      severity = 3
    }
  }

  if {
    value {
      # SEV4 if it takes more than 500ms for us to complete an HTTP request.  We
      # want to record that things were slow, but not wake anyone up if it
      # momentarily pops above 500ms.
      min_value = "500"
    }

    then {
      notify   = [ "${circonus_contact_group.myapp-owners.id}" ]
      severity = 3
    }
  }

  if {
```

```
  if {
    value {
      # If for whatever reason we're not recording any values for the last
      # 24hrs, fire off a SEV1.
      absent = "24h"
    }

    then {
      notify = [ "${circonus_contact_group.myapp-owners.id}" ]
      severity = 1
    }
  }

  tags = [ "${var.myapp-tags}" ]
}

resource "circonus_contact_group" "myapp-owners" {
  name = "My App Owners"
  tags = [ "${var.myapp-tags}" ]
}

resource "circonus_check" "myapp-https" {
  name = "My App's HTTPS Check"

  notes = <<-EOF
A check to create metric streams for Time to First Byte, HTTP transaction
duration, and the TTL of a TLS cert.
EOF

  collector {
    id = "/broker/1"
  }

  http {
    code = "^200$"
    headers = {
      X-Request-Type = "health-check",
    }
    url = "https://www.example.com/myapp/healthz"
  }

  metric {
    name = "${circonus_metric.myapp-cert-ttl.name}"
    tags = "${circonus_metric.myapp-cert-ttl.tags}"
    type = "${circonus_metric.myapp-cert-ttl.type}"
    unit = "${circonus_metric.myapp-cert-ttl.unit}"
  }

  metric {
    name = "${circonus_metric.myapp-duration.name}"
    tags = "${circonus_metric.myapp-duration.tags}"
    type = "${circonus_metric.myapp-duration.type}"
    unit = "${circonus_metric.myapp-duration.unit}"
  }

  period       = 60
  tags         = ["source:circonus", "author:terraform"]
  timeout      = 10
}

resource "circonus_metric" "myapp-cert-ttl" {
  name = "cert_end_in"
  type = "numeric"
  unit = "seconds"
  tags = [ "${var.myapp-tags}", "resource:tls" ]
}
```

```
resource "circonus_metric" "myapp-duration" {
  name = "duration"
  type = "numeric"
  unit = "miliseconds"
  tags = [ "${var.myapp-tags}" ]
}
```

Argument Reference

- `check` - (Required) The Circonus ID that this Rule Set will use to search for a metric stream to alert on.
- `if` - (Required) One or more ordered predicate clauses that describe when Circonus should generate a notification. See below for details on the structure of an `if` configuration clause.
- `link` - (Optional) A link to external documentation (or anything else you feel is important) when a notification is sent. This value will show up in email alerts and the Circonus UI.
- `metric_type` - (Optional) The type of metric this rule set will operate on. Valid values are `numeric` (the default) and `text`.
- `notes` - (Optional) Notes about this rule set.
- `parent` - (Optional) A Circonus Metric ID that, if specified and active with a severity 1 alert, will silence this rule set until all of the severity 1 alerts on the parent clear. This value must match the format `${check_id}_${metric_name}`.
- `metric_name` - (Required) The name of the metric stream within a given check that this rule set is active on.
- `tags` - (Optional) A list of tags assigned to this rule set.

`if` Configuration

The `if` configuration block is an ordered list of rules (https://login.circonus.com/user/docs/Alerting/Rules/Configure) that are evaluated in order, first to last. The first `if` condition to evaluate true shortcircuits all other `if` blocks in this rule set. An `if` block is also referred to as a "rule." It is advised that all high-severity rules are ordered before low-severity rules otherwise low-severity rules will mask notifications that should be delivered with a high-severity.

`if` blocks are made up of two configuration blocks: `value` and `then`. The `value` configuration block specifies the criteria underwhich the metric streams are evaluated. The `then` configuration block, optional, specifies what action to take.

`value` Configuration

A `value` block can have only one of several "predicate" attributes specified because they conflict with each other. The list of mutually exclusive predicates is dependent on the `metric_type`. To evaluate multiple predicates, create multiple `if` configuration blocks in the proper order.

`numeric` Predicates

Metric types of type `numeric` support the following predicates. Only one of the following predicates may be specified at a time.

- `absent` - (Optional) If a metric has not been observed in this duration the rule will fire. When present, this duration is evaluated in terms of seconds.
- `changed` - (Optional) A boolean indicating this rule should fire when the value changes (e.g. `n != n<sub>1</sub>`).
- `min_value` - (Optional) When the value is less than this value, this rule will fire (e.g. `n < ${min_value}`).
- `max_value` - (Optional) When the value is greater than this value, this rule will fire (e.g. `n > ${max_value}`).

Additionally, a `numeric` check can also evaluate data based on a windowing function versus the last measured value in the metric stream. In order to have a rule evaluate on derived value from a window, include a nested `over` attribute inside of the `value` configuration block. An `over` attribute needs two attributes:

- `last` - (Optional) A duration for the sliding window. Default `300s`.
- `using` - (Optional) The window function to use over the `last` interval. Valid window functions include: `average` (the default), `stddev`, `derive`, `derive_stddev`, `counter`, `counter_stddev`, `derive_2`, `derive_2_stddev`, `counter_2`, and `counter_2_stddev`.

`text` Predicates

Metric types of type `text` support the following predicates:

- `absent` - (Optional) If a metric has not been observed in this duration the rule will fire. When present, this duration is evaluated in terms of seconds.
- `changed` - (Optional) A boolean indicating this rule should fire when the last value in the metric stream changed from it's previous value (e.g. `n != n-1`).
- `contains` - (Optional) When the last value in the metric stream the value is less than this value, this rule will fire (e.g. `strstr(n, ${contains}) != NULL`).
- `match` - (Optional) When the last value in the metric stream value exactly matches this configured value, this rule will fire (e.g. `strcmp(n, ${match}) == 0`).
- `not_contain` - (Optional) When the last value in the metric stream does not match this configured value, this rule will fire (e.g. `strstr(n, ${contains}) == NULL`).
- `not_match` - (Optional) When the last value in the metric stream does not match this configured value, this rule will fire (e.g. `strstr(n, ${not_match}) == NULL`).

`then` Configuration

A `then` block can have the following attributes:

- `after` - (Optional) Only execute this notification after waiting for this number of minutes. Defaults to immediately, or `0m`.
- `notify` - (Optional) A list of contact group IDs to notify when this rule is sends off a notification.
- `severity` - (Optional) The severity level of the notification. This can be set to any value between `0` and `5`. Defaults to `1`.

Import Example

`circonus_rule_set` supports importing resources. Supposing the following Terraform (and that the referenced `circonus_metric` (/docs/providers/circonus/r/metric.html) and `circonus_check` (/docs/providers/circonus/r/check.html) have already been imported):

```
resource "circonus_rule_set" "icmp-latency-alert" {
  check = "${circonus_check.api_latency.checks[0]}"
  metric_name = "maximum"

  if {
    value {
      absent = "600s"
    }

    then {
      notify = [ "${circonus_contact_group.test-trigger.id}" ]
      severity = 1
    }
  }

  if {
    value {
      over {
        last = "120s"
        using = "average"
      }

      max_value = 0.5 # units are in miliseconds
    }

    then {
      notify = [ "${circonus_contact_group.test-trigger.id}" ]
      severity = 2
    }
  }
}
```

It is possible to import a `circonus_rule_set` resource with the following command:

```
$ terraform import circonus_rule_set.icmp-latency-alert ID
```

Where `ID` is the `_cid` or Circonus ID of the Rule Set (e.g. `/rule_set/201285_maximum`) and `circonus_rule_set.icmp-latency-alert` is the name of the resource whose state will be populated as a result of the command.

circonus_worksheet

The `circonus_worksheet` resource creates and manages a Circonus Worksheet (https://login.circonus.com/resources/api/calls/worksheet).

Usage

```
variable "myapp-tags" {
  type    = "list"
  default = [ "app:myapp", "owner:myteam" ]
}

resource "circonus_graph" "latency-graph" {
  name        = "Latency Graph"
  description = "A sample graph showing off two data points"
  notes       = "Misc notes about this graph"
  graph_style = "line"
  line_style  = "stepped"

  metric {
    check       = "${circonus_check.api_latency.checks[0]}"
    metric_name = "maximum"
    metric_type = "numeric"
    name        = "Maximum Latency"
    axis        = "left"
    color       = "#657aa6"
  }

  metric {
    check       = "${circonus_check.api_latency.checks[0]}"
    metric_name = "minimum"
    metric_type = "numeric"
    name        = "Minimum Latency"
    axis        = "right"
    color       = "#0000ff"
  }

  tags = [ "${var.myapp-tags}" ]
}

resource "circonus_worksheet" "myapp_latency" {
  title = "MyApp: Latencies"
  graphs = [
    "${circonus_graph.latency-graph.id}",
  ]
}

resource "circonus_worksheet" "service_myapp" {
  title = "Service: MyApp"
  smart_queries = [
    {
      name  = "MyApp"
      query = "(tags:${var.myapp-tags})"
    }
  ]
}
```

Argument Reference

- `title` - (Required) The title of the worksheet.
- `description` - (Optional) Description of what the worksheet is for.
- `favourite` - (Optional) Mark (star) this worksheet as a favorite. Default is `false`.
- `notes` - (Optional) A place to store notes about this worksheet.
- `graphs` - (Optional) A list of graphs that compose this worksheet.
- `smart_queries` - (Optional) The smart queries that will be displayed on this worksheet. See below for details on how to configure a `smart_query`.
- `tags` - (Optional) A list of tags assigned to this worksheet.

`smart_queries` Attributes

`smart_queries` is a list of smart query objects. Each smart query object has the following required attributes:

- `name` - (Required) The name (heading) for the smart graph section in the worksheet.
- `query` - (Required) A search query that determines which graphs will be shown..

Import Example

It is possible to import a `circonus_worksheet` resource with the following command:

```
$ terraform import circonus_worksheet.icmp-latency ID
```

Where `ID` is the `_cid` or Circonus ID of the worksheet (e.g. `worksheets/45640239-bb81-4ecb-81e6-b5c6015e5dd5`) and `circonus_worksheet.icmp-latency` is the name of the resource whose state will be populated as a result of the command.