Datadog Provider

The Datadog (https://www.datadoghq.com) provider is used to interact with the resources supported by Datadog. The provider needs to be configured with the proper credentials before it can be used.

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

Example Usage

```
# Configure the Datadog provider
provider "datadog" {
  api_key = "${var.datadog_api_key}"
  app_key = "${var.datadog_app_key}"
}

# Create a new monitor
resource "datadog_monitor" "default" {
  # ...
}

# Create a new timeboard
resource "datadog_timeboard" "default" {
  # ...
}
```

Argument Reference

The following arguments are supported:

- `api_key` - (Required) Datadog API key. This can also be set via the `DATADOG_API_KEY` environment variable.
- `app_key` - (Required) Datadog APP key. This can also be set via the `DATADOG_APP_KEY` environment variable.
- `api_url` - (Optional) The API Url. This can be also be set via the `DATADOG_HOST` environment variable.

datadog_downtime

Provides a Datadog downtime resource. This can be used to create and manage Datadog downtimes.

Example Usage

```
# Create a new daily 1700-0900 Datadog downtime
resource "datadog_downtime" "foo" {
  scope = ["*"]
  start = 1483308000
  end   = 1483365600

  recurrence {
    type   = "days"
    period = 1
  }

  # Datadog API will reject dates in the past so let's ignore `start` and `end`
  # arguments during lifecycle. To update or extend an existing downtime, temporarily
  # remove the `ignore` section, apply timestamp changes, and re-apply the `ignore`
  # section.
  lifecycle {
    ignore_changes = ["start", "end"]
  }
}
```

Argument Reference

The following arguments are supported:

- `scope` - (Required) A list of items to apply the downtime to, e.g. host:X
- `active` - (Optional) A flag indicating if the downtime is active now.
- `disabled` - (Optional) A flag indicating if the downtime was disabled.
- `start` - (Optional) POSIX timestamp to start the downtime.
- `start_date` - (Optional) String representing date and time to start the downtime in RFC3339 format.
- `end` - (Optional) POSIX timestamp to end the downtime.
- `end_date` - (Optional) String representing date and time to end the downtime in RFC3339 format.
- `recurrence` - (Optional) A dictionary to configure the downtime to be recurring.
    - `type` - days, weeks, months, or years
    - `period` - How often to repeat as an integer. For example to repeat every 3 days, select a type of days and a period of 3.
    - `week_days` - (Optional) A list of week days to repeat on. Choose from: Mon, Tue, Wed, Thu, Fri, Sat or Sun. Only applicable when type is weeks. First letter must be capitalized.
    - `until_occurrences` - (Optional) How many times the downtime will be rescheduled. `until_occurrences` and

`until_date` are mutually exclusive.

    - `until_date` - (Optional) The date at which the recurrence should end as a POSIX timestamp. `until_occurrences` and `until_date` are mutually exclusive.

- `message` - (Optional) A message to include with notifications for this downtime.
- `monitor_id` - (Optional) Reference to which monitor this downtime is applied. When scheduling downtime for a given monitor, datadog changes `silenced` property of the monitor to match the `end` POSIX timestamp.

Attributes Reference

The following attributes are exported:

- `id` - ID of the Datadog downtime

Import

Downtimes can be imported using their numeric ID, e.g.

```
$ terraform import datadog_downtime.bytes_received_localhost 2081
```

datadog_integration_aws

Provides a Datadog - Amazon Web Services integration resource. This can be used to create and manage Datadog - Amazon Web Services integration.

Update operations are currently not supported with datadog API so any change forces a new resource.

Example Usage

```
# Create a new Datadog - Amazon Web Services integration
resource "datadog_integration_aws" "sandbox" {
    account_id = "1234567890"
    role_name = "DatadogAWSIntegrationRole"
    filter_tags = ["key:value"]
    host_tags = ["key:value", "key2:value2"]
    account_specific_namespace_rules = {
        "auto_scaling" = false
        "opsworks" = false
    }
}
```

Argument Reference

The following arguments are supported:

- `account_id` - (Required) Your AWS Account ID without dashes.
- `role_name` - (Required) Your Datadog role delegation name.
- `filter_tags` - (Optional) Array of EC2 tags (in the form `key:value`) defines a filter that Datadog use when collecting metrics from EC2. Wildcards, such as ? (for single characters) and `*` (for multiple characters) can also be used.

Only hosts that match one of the defined tags will be imported into Datadog. The rest will be ignored. Host matching a given tag can also be excluded by adding `!` before the tag.

e.x. `env:production,instance-type:c1.*,!region:us-east-1`

- `host_tags` - (Optinal) Array of tags (in the form key:value) to add to all hosts and metrics reporting through this integration.
- `account_specific_namespace_rules` - (Optinal) Enables or disables metric collection for specific AWS namespaces for this AWS account only. A list of namespaces can be found at the available namespace rules API endpoint (https://api.datadoghq.com/api/v1/integration/aws/available_namespace_rules).

See also

- Datadog API Reference > Integrations > AWS (https://docs.datadoghq.com/api/?lang=bash#aws)

Attributes Reference

The following attributes are exported:

- `external_id` - AWS External ID

Import

Amazon Web Services integrations can be imported using their `account ID` and `role name` separated with a colon (`:`), while the `external_id` should be passed by setting an environment variable called `EXTERNAL_ID`

```
$ EXTERNAL_ID=${external_id} terraform import datadog_integration_aws.test ${account_id}:${role_name}
```

datadog_integration_gcp

Provides a Datadog - Google Cloud Platform integration resource. This can be used to create and manage Datadog - Google Cloud Platform integration.

Example Usage

```
# Create a new Datadog - Google Cloud Platform integration
resource "datadog_integration_gcp" "awesome_gcp_project_integration" {
  project_id     = "awesome-project-id"
  private_key_id = "12345678901234567890123456789012345678901"
  private_key    = "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----\n"
  client_email   = "awesome-service-account@awesome-project-id.iam.gserviceaccount.com"
  client_id      = "123456789012345678901"
  host_filters   = "foo:bar,buzz:lightyear"
}
```

Argument Reference

The following arguments are supported:

- `project_id` - (Required) Your Google Cloud project ID found in your JSON service account key.
- `private_key_id` - (Required) Your private key ID found in your JSON service account key.
- `private_key` - (Required) Your private key name found in your JSON service account key.
- `client_email` - (Required) Your email found in your JSON service account key.
- `client_id` - (Required) Your ID found in your JSON service account key.
- `host_filters` - (Optional) Limit the GCE instances that are pulled into Datadog by using tags. Only hosts that match one of the defined tags are imported into Datadog.

See also

- Google Cloud > Creating and Managing Service Account Keys (https://cloud.google.com/iam/docs/creating-managing-service-account-keys)
- Datadog API Reference > Integrations > Google Cloud Platform (https://docs.datadoghq.com/api/?lang=bash#google-cloud-platform)

Attributes Reference

The following attributes are exported:

- `project_id` - Google Cloud project ID

- `client_email` - Google Cloud project service account email
- `host_filters` - Host filters

Import

Google Cloud Platform integrations can be imported using their project ID, e.g.

```
$ terraform import datadog_integration_gcp.awesome_gcp_project_integration project_id
```

datadog_metric_metadata

Provides a Datadog metric_metadata resource. This can be used to manage a metric's metadata.

Example Usage

```
# Manage a Datadog metric's metadata
resource "datadog_metric_metadata" "request_time" {
  metric = "request.time"
  short_name = "Request time"
  description = "99th percentile request time in millseconds"
  type = "gauge"
  unit = "millisecond"
}
```

Argument Reference

The following arguments are supported:

- `metric` - (Required) The name of the metric.
- `description` - (Optional) A description of the metric.
- `short_name` - (Optional) A short name of the metric.
- `unit` - (Optional) Primary unit of the metric such as 'byte' or 'operation'.
- `per_unit` - (Optional) 'Per' unit of the metric such as 'second' in 'bytes per second'.
- `statsd_interval` - (Optional) If applicable, stasd flush interval in seconds for the metric.

datadog_monitor

Provides a Datadog monitor resource. This can be used to create and manage Datadog monitors.

Example Usage

```
# Create a new Datadog monitor
resource "datadog_monitor" "foo" {
  name               = "Name for monitor foo"
  type               = "metric alert"
  message            = "Monitor triggered. Notify: @hipchat-channel"
  escalation_message = "Escalation message @pagerduty"

  query = "avg(last_1h):avg:aws.ec2.cpu{environment:foo,host:foo} by {host} > 4"

  thresholds {
    ok                = 0
    warning           = 2
    warning_recovery  = 1
    critical          = 4
    critical_recovery = 3
  }

  notify_no_data    = false
  renotify_interval = 60

  notify_audit = false
  timeout_h    = 60
  include_tags = true

  silenced {
    "*" = 0
  }

  tags = ["foo:bar", "baz"]
}
```

Argument Reference

The following arguments are supported:

- `type` - (Required) The type of the monitor, chosen from:
    - `metric alert`
    - `service check`
    - `event alert`
    - `query alert`
    - `composite`
    - `log alert`
- `name` - (Required) Name of Datadog monitor

- `query` - (Required) The monitor query to notify on. Note this is not the same query you see in the UI and the syntax is different depending on the monitor `type`, please see the API Reference (https://docs.datadoghq.com/api/?lang=python#create-a-monitor) for details. **Warning:** `terraform plan` won't perform any validation of the query contents.
- `message` - (Required) A message to include with notifications for this monitor. Email notifications can be sent to specific users by using the same '@username' notation as events.
- `escalation_message` - (Optional) A message to include with a re-notification. Supports the '@username' notification allowed elsewhere.
- `thresholds` - (Optional)
    - Metric alerts: A dictionary of thresholds by threshold type. Currently we have four threshold types for metric alerts: critical, critical recovery, warning, and warning recovery. Critical is defined in the query, but can also be specified in this option. Warning and recovery thresholds can only be specified using the thresholds option. Example usage: `thresholds { critical = 90 critical_recovery = 85 warning = 80 warning_recovery = 75 }` **Warning:** the `critical` threshold value must match the one contained in the `query` argument. The `threshold` from the previous example is valid along with a query like `avg(last_1h):avg:system.disk.in_use{role:sqlserver} by {host} > 90` but along with something like `avg(last_1h):avg:system.disk.in_use{role:sqlserver} by {host} > 95` would make the Datadog API return a HTTP error 400, complaining "The value provided for parameter 'query' is invalid".
    - Service checks: A dictionary of thresholds by status. Because service checks can have multiple thresholds, we don't define them directly in the query. Default values: `thresholds { ok = 1 critical = 1 warning = 1 unknown = 1 }`
- `notify_no_data` (Optional) A boolean indicating whether this monitor will notify when data stops reporting. Defaults to false.
- `new_host_delay` (Optional) Time (in seconds) to allow a host to boot and applications to fully start before starting the evaluation of monitor results. Should be a non negative integer. Defaults to 300.
- `evaluation_delay` (Optional, only applies to metric alert) Time (in seconds) to delay evaluation, as a non-negative integer. For example, if the value is set to 300 (5min), the timeframe is set to last_5m and the time is 7:00, the monitor will evaluate data from 6:50 to 6:55. This is useful for AWS CloudWatch and other backfilled metrics to ensure the monitor will always have data during evaluation.
- `no_data_timeframe` (Optional) The number of minutes before a monitor will notify when data stops reporting. Must be at least 2x the monitor timeframe for metric alerts or 2 minutes for service checks. Default: 2x timeframe for metric alerts, 2 minutes for service checks.
- `renotify_interval` (Optional) The number of minutes after the last notification before a monitor will re-notify on the current status. It will only re-notify if it's not resolved.
- `notify_audit` (Optional) A boolean indicating whether tagged users will be notified on changes to this monitor. Defaults to false.
- `timeout_h` (Optional) The number of hours of the monitor not reporting data before it will automatically resolve from a triggered state. Defaults to false.
- `include_tags` (Optional) A boolean indicating whether notifications from this monitor will automatically insert its triggering tags into the title. Defaults to true.

- `require_full_window` (Optional) A boolean indicating whether this monitor needs a full window of data before it's evaluated. We highly recommend you set this to False for sparse metrics, otherwise some evaluations will be skipped. Default: True for "on average", "at all times" and "in total" aggregation. False otherwise.
- `locked` (Optional) A boolean indicating whether changes to to this monitor should be restricted to the creator or admins. Defaults to False.
- `tags` (Optional) A list of tags to associate with your monitor. This can help you categorize and filter monitors in the manage monitors page of the UI. Note: it's not currently possible to filter by these tags when querying via the API
- `silenced` (Optional) Each scope will be muted until the given POSIX timestamp or forever if the value is 0. To mute the alert completely:

  ```
  silenced {
    "*" =  0
  }
  ```

  To mute role:db for a short time:

  ```
  silenced {
    "role:db" = 1412798116
  }
  ```

Attributes Reference

The following attributes are exported:

- `id` - ID of the Datadog monitor

Import

Monitors can be imported using their numeric ID, e.g.

```
$ terraform import datadog_monitor.bytes_received_localhost 2081
```

datadog_screenboard

Provides a Datadog screenboard resource. This can be used to create and manage Datadog screenboards.

Example Usage

```
# Create a new Datadog screenboard
resource "datadog_screenboard" "acceptance_test" {
  title     = "Test Screenboard"
  read_only = true

  template_variable {
    name    = "varname 1"
    prefix  = "pod_name"
    default = "*"
  }

  template_variable {
    name    = "varname 2"
    prefix  = "service_name"
    default = "autoscaling"
  }

  widget {
    type       = "free_text"
    x          = 5
    y          = 5
    text       = "test text"
    text_align = "right"
    font_size  = "36"
    color      = "#ffc0cb"
  }

  widget {
    type        = "timeseries"
    x           = 25
    y           = 5
    title       = "graph title terraform"
    title_size  = 16
    title_align = "right"
    legend      = true
    legend_size = 16

    time {
      live_span = "1d"
    }

    tile_def {
      viz = "timeseries"

      request {
        q    = "avg:system.cpu.user{*}"
        type = "line"

        style {
          palette = "purple"
          type    = "dashed"
          width   = "thin"
        }
      }
```

```
    marker {
      label = "test marker"
      type  = "error dashed"
      value = "y < 6"
    }

    event {
      q = "test event"
    }
  }
}

widget {
  type        = "query_value"
  x           = 45
  y           = 25
  title       = "query value title terraform"
  title_size  = 20
  title_align = "center"
  legend      = true
  legend_size = 16

  tile_def {
    viz = "query_value"

    request {
      q    = "avg:system.cpu.user{*}"
      type = "line"

      style {
        palette = "purple"
        type    = "dashed"
        width   = "thin"
      }

      conditional_format {
        comparator = ">"
        value      = "1"
        palette    = "white_on_red"
      }

      conditional_format {
        comparator = ">="
        value      = "2"
        palette    = "white_on_yellow"
      }

      aggregator = "max"
    }

    custom_unit = "%"
    autoscale   = false
    precision   = "6"
    text_align  = "right"
  }
}

widget {
  type        = "toplist"
  x           = 65
  y           = 5
  title       = "toplist title terraform"
  legend      = true
  legend_size = "auto"
```

```
  time {
    live_span = "1d"
  }

  tile_def {
    viz = "toplist"

    request {
      q = "top(avg:system.load.1{*} by {host}, 10, 'mean', 'desc')"

      style {
        palette = "purple"
        type    = "dashed"
        width   = "thin"
      }

      conditional_format {
        comparator = ">"
        value      = "4"
        palette    = "white_on_green"
      }
    }
  }
}

widget {
  type  = "change"
  x     = 85
  y     = 5
  title = "change title terraform"

  tile_def {
    viz = "change"

    request {
      q             = "min:system.load.1{*} by {host}"
      compare_to    = "week_before"
      change_type   = "relative"
      order_by      = "present"
      order_dir     = "asc"
      extra_col     = ""
      increase_good = false
    }
  }
}

widget {
  type  = "event_timeline"
  x     = 105
  y     = 5
  title = "event_timeline title terraform"
  query = "status:error"

  time {
    live_span = "1d"
  }
}

widget {
  type       = "event_stream"
  x          = 115
  y          = 5
  title      = "event_stream title terraform"
  query      = "*"
  event_size = "l"
```

```
  time {
    live_span = "4h"
  }
}

widget {
  type   = "image"
  x      = 145
  y      = 5
  title  = "image title terraform"
  sizing = "fit"
  margin = "large"
  url    = "https://datadog-prod.imgix.net/img/dd_logo_70x75.png"
}

widget {
  type       = "note"
  x          = 165
  y          = 5
  bgcolor    = "pink"
  text_align = "right"
  font_size  = "36"
  tick       = true
  tick_edge  = "bottom"
  tick_pos   = "50%"
  html       = "<b>test note</b>"
}

widget {
  type     = "alert_graph"
  x        = 185
  y        = 5
  title    = "alert graph title terraform"
  alert_id = "123456"
  viz_type = "toplist"

  time {
    live_span = "15m"
  }
}

widget {
  type       = "alert_value"
  x          = 205
  y          = 5
  title      = "alert value title terraform"
  alert_id   = "123456"
  text_size  = "fill_height"
  text_align = "right"
  precision  = "*"
  unit       = "b"
}

widget {
  type = "iframe"
  x    = 225
  y    = 5
  url  = "https://www.datadoghq.org"
}

widget {
  type        = "check_status"
  x           = 245
  y           = 5
  title       = "test title"
```

```
    title       = "test title"
    title_align = "left"
    grouping    = "check"
    check       = "aws.ecs.agent_connected"
    tags        = ["*"]
    group       = "cluster:test"

    time {
      live_span = "30m"
    }
  }

  widget {
    type                    = "trace_service"
    x                       = 265
    y                       = 5
    env                     = "testEnv"
    service_service         = ""
    service_name            = ""
    size_version            = "large"
    layout_version          = "three_column"
    must_show_hits          = true
    must_show_errors        = true
    must_show_latency       = true
    must_show_breakdown     = true
    must_show_distribution  = true
    must_show_resource_list = true

    time {
      live_span = "30m"
    }
  }

  widget {
    type  = "hostmap"
    x     = 285
    y     = 5
    query = "avg:system.load.1{*} by {host}"

    tile_def {
      viz             = "hostmap"
      node_type       = "container"
      scope           = ["datacenter:test"]
      group           = ["pod_name"]
      no_group_hosts  = false
      no_metric_hosts = false

      request {
        q    = "max:process.stat.container.io.wbps{datacenter:test} by {host}"
        type = "fill"
      }

      style {
        palette      = "hostmap_blues"
        palette_flip = true
        fill_min     = 20
        fill_max     = 300
      }
    }
  }

  widget {
    type                      = "manage_status"
    x                         = 305
    y                         = 5
    display_format            = "countsAndList"
```

```
    color_preference          = "background"
    hide_zero_counts          = true
    manage_status_show_title  = false
    manage_status_title_text  = "test title"
    manage_status_title_size  = "20"
    manage_status_title_align = "right"

    params {
      sort  = "status,asc"
      text  = "status:alert"
      count = 50
      start = 0
    }
  }

  widget {
    type    = "log_stream"
    x       = 325
    y       = 5
    query   = "source:kubernetes"
    columns = "[\"column1\",\"column2\",\"column3\"]"
    logset  = "1234"

    time {
      live_span = "1h"
    }
  }

  widget {
    type = "process"
    x    = 365
    y    = 5

    tile_def {
      viz = "process"

      request {
        query_type  = "process"
        metric      = "process.stat.cpu.total_pct"
        text_filter = ""
        tag_filters = []
        limit       = 200

        style = {
          palette = "dog_classic_area"
        }
      }
    }
  }
}
```

Argument Reference

The following arguments are supported:

- `title` - (Required) The name of the screenboard.
- `height` - (Optional) The screenboard's height.
- `width` - (Optional) The screenboard's width.

- `read_only` - (Optional) The read-only status of the screenboard. Default is false.
- `shared` - (Optional) Whether the screenboard is shared or not. Default is false.
- `widget` - (Required) Nested block describing a widget. The structure of this block is described below. Multiple widget blocks are allowed within a datadog_screenboard resource.
- `template_variable` - (Optional) Nested block describing a template variable. The structure of this block is described below. Multiple template_variable blocks are allowed within a datadog_screenboard resource.

Nested `widget` blocks

Nested `widget` blocks have the following structure:

- `type` - (Required) The type of the widget. One of "free_text", "timeseries", "query_value", "toplist", "change", "event_timeline", "event_stream", "image", "note", "alert_graph", "alert_value", "iframe", "check_status", "trace_service", "hostmap", "manage_status", "log_stream", or "process".
- `x` - (Required) The position of the widget on the x (vertical) axis. Should be greater or equal to 0.
- `y` - (Required) The position of the widget on the y (horizontal) axis. Should be greater or equal to 0.
- `title` - (Optional) The title of the widget.
- `title_align` - (Optional) The alignment of the widget's title. One of "left", "center", or "right".
- `title_size` - (Optional) The size of the widget's title. Default is 16.
- `height` - (Optional) The height of the widget. Default is 15.
- `width` - (Optional) The width of the widget. Default is 50.
- `text` - (Optional, only for widgets of type "free_text") The text to display in the widget.
- `color` - (Optional, only for widgets of type "free_text") The color of the text in the widget.
- `font_size` - (Optional, only for widgets of type "free_text", "note") The size of the text in the widget.
- `text_size` - (Optional, only for widgets of type "alert_value") The size of the text in the widget.
- `unit` - (Optional, only for widgets of type "alert_value") The unit for the value displayed in the widget.
- `precision` - (Optional, only for widgets of type "alert_value") The precision to use when displaying the value. Use "*" for maximum precision.
- `text_align` - (Optional, only for widgets of type "free_text", "alert_value", "note") The alignment of the text in the widget.
- `alert_id` - (Optional, only for widgets of type "alert_value", "alert_graph") The ID of the monitor used by the widget.
- `auto_refresh` - (Optional, only for widgets of type "alert_value", "alert_graph") Boolean indicating whether the widget is refreshed automatically.
- `legend` - (Optional, only for widgets of type "timeseries", "query_value", "toplist") Boolean indicating whether to display a legend in the widget.
- `legend_size` - (Optional, only for widgets of type "timeseries", "query_value", "toplist") The size of the legend displayed

in the widget.

- `query` - (Optional, only for widgets of type "event_timeline", "event_stream", "hostmap", "log_stream") The query to use in the widget.
- `url` - (Optional, only for widgets of type "image", "iframe") The URL to use as a data source for the widget.
- `viz_type` - (Optional, only for widgets of type "alert_graph") Type of visualization to use when displaying the widget. Either "timeseries" or "toplist".
- `tags` - (Optional, only for widgets of type "check_status") List of tags to use in the widget.
- `check` - (Optional, only for widgets of type "check_status") The check to use in the widget.
- `group` - (Optional, only for widgets of type "check_status") The check group to use in the widget.
- `grouping` - (Optional, only for widgets of type "check_status") Either "check" or "cluster", depending on whether the widget should use a single check or a cluster of checks.
- `group_by` - (Optional, only for widgets of type "check_status") When grouping = "cluster", indicates a list of tags to use for grouping.
- `tick` - (Optional, only for widgets of type "note") Boolean indicating whether a tick should be displayed on the border of the widget.
- `tick_pos` - (Optional, only for widgets of type "note") When tick = true, string with a percent sign indicating the position of the tick. Example: use tick_pos = "50%" for centered alignment.
- `tick_edge` - (Optional, only for widgets of type "note") When tick = true, string indicating on which side of the widget the tick should be displayed. One of "bottom", "top", "left", "right".
- `html` - (Optional, only for widgets of type "note") The content of the widget. HTML tags supported.
- `bgcolor` - (Optional, only for widgets of type "note") The color of the background of the widget.
- `event_size` - (Optional, only for widgets of type "event_stream") The size of the events in the widget. Either "s" (small, title only) or "l" (large, full event).
- `sizing` - (Optional, only for widgets of type "image") The preferred method to adapt the dimensions of the image to those of the widget. One of "center" (center the image in the tile), "zoom" (zoom the image to cover the whole tile) or "fit" (fit the image dimensions to those of the tile).
- `margin` - (Optional, only for widgets of type "image") The margins to use around the image. Either "small" or "large".
- `env` - (Optional, only for widgets of type "trace_service") The environment to use.
- `service_service` - (Optional, only for widgets of type "trace_service") The trace service to use.
- `service_name` - (Optional, only for widgets of type "trace_service") The name of the service to use.
- `size_version` - (Optional, only for widgets of type "trace_service") The size of the widget. One of "small", "medium", "large".
- `layout_version` - (Optional, only for widgets of type "trace_service") The number of columns to use when displaying values. One of "one_column", "two_column", "three_column".
- `must_show_hits` - (Optional, only for widgets of type "trace_service") Boolean indicating whether to display hits.

- `must_show_errors` - (Optional, only for widgets of type "trace_service") Boolean indicating whether to display errors.
- `must_show_latency` - (Optional, only for widgets of type "trace_service") Boolean indicating whether to display latency.
- `must_show_breakdown` - (Optional, only for widgets of type "trace_service") Boolean indicating whether to display breakdown.
- `must_show_distribution` - (Optional, only for widgets of type "trace_service") Boolean indicating whether to display distribution.
- `must_show_resource_list` - (Optional, only for widgets of type "trace_service") Boolean indicating whether to display resources.
- `display_format` - (Optional, only for widgets of type "manage_status") The display setting to use. One of "counts", "list", or "countsAndList".
- `color_preference` - (Optional, only for widgets of type "manage_status") Whether to colorize text or background. One of "text", "background".
- `hide_zero_counts` - (Optional, only for widgets of type "manage_status") Boolean indicating whether to hide empty categories.
- `manage_status_show_title` - (Optional, only for widgets of type "manage_status") Boolean indicating whether to show a title.
- `manage_status_title_text` - (Optional, only for widgets of type "manage_status") The title of the widget.
- `manage_status_title_size` - (Optional, only for widgets of type "manage_status") The size of the widget's title.
- `manage_status_title_align` - (Optional, only for widgets of type "manage_status") The alignment of the widget's title. One of "left", "center", or "right".
- `columns` - (Optional, only for widgets of type "log_stream") Stringified list of columns to use. Example: `"[\"column1\",\"column2\",\"column3\"]"`
- `logset` - (Optional, only for widgets of type "log_stream") ID of the logset to use.
- `time` - (Optional, only for widgets of type "timeseries", "toplist", "event_timeline", "event_stream", "alert_graph", "check_status", "trace_service", "log_stream") Nested block describing the timeframe to use when displaying the widget. The structure of this block is described below. At most one such block should be present in a given widget.
- `tile_def` - (Optional, only for widgets of type "timeseries", "query_value", "hostmap", "change", "toplist", "process") Nested block describing the content to display in the widget. The structure of this block is described below. At most one such block should be present in a given widget.
- `params` - (Optional, only for widgets of type "manage_status") Nested block describing the monitors to display. The structure of this block is described below. At most one such block should be present in a given widget.

Nested `widget` `time` blocks

Only for widgets of type "timeseries", "toplist", "event_timeline", "event_stream", "alert_graph", "check_status", "trace_service", "log_stream".

Nested `widget` `time` blocks have the following structure:

- `live_span` - (Required) The timeframe to use when displaying the widget. One of "10m", "30m", "1h", "4h", "1d", "2d", "1w".

Nested `widget params` blocks

Only for widgets of type "manage_status".

Nested `widget params` blocks have the following structure:

- `sort` - (Optional) The method to use to sort monitors. Example: "status,asc".
- `text` - (Optional) The query to use to get monitors. Example: "status:alert".
- `count` - (Optional) The number of monitors to display.
- `start` - (Optional) The start of the list. Typically 0.

Nested `widget tile_def` blocks

Only for widgets of type "timeseries", "query_value", "hostmap", "change", "toplist", "process".

Nested `widget tile_def` blocks have the following structure:

- `viz` - (Required) Should be the same as the widget's type. One of "timeseries", "query_value", "hostmap", "change", "toplist", "process".
- `request` - (Required) Nested block describing the request to use when displaying the widget. The structure of this block is described below. Multiple request blocks are allowed within a given tile_def block.
- `marker` - (Optional, only for widgets of type "timeseries") Nested block describing the marker to use when displaying the widget. The structure of this block is described below. Multiple marker blocks are allowed within a given tile_def block.
- `event` - (Optional, only for widgets of type "timeseries") Nested block describing the event overlays to use when displaying the widget. The structure of this block is described below. At most one such block should be present in a given tile_def block.
- `custom_unit` - (Optional, only for widgets of type "query_value") The unit for the value displayed in the widget
- `autoscale` - (Optional, only for widgets of type "query_value") Boolean indicating whether to automatically scale the tile.
- `precision` - (Optional, only for widgets of type "query_value") The precision to use when displaying the tile.
- `text_align` - (Optional, only for widgets of type "query_value") The alignment of the text.
- `node_type` - (Optional, only for widgets of type "hostmap") The type of node used. Either "host" or "container".
- `scope` - (Optional, only for widgets of type "hostmap") The list of tags to filter nodes by.
- `group` - (Optional, only for widgets of type "hostmap") The list of tags to group nodes by.
- `no_group_host` - (Optional, only for widgets of type "hostmap") Boolean indicating whether to show ungrouped nodes.
- `no_metric_host` - (Optional, only for widgets of type "hostmap") Boolean indicating whether to show nodes with no

metrics.

- `style` - (Optional, only for widgets of type "hostmap") Nested block describing how to display the widget. The structure of this block is described below. At most one such block should be present in a given tile_def block.

Nested `widget` `tile_def` `style` blocks

Only for widgets of type "hostmap".

Nested `widget` `tile_def` `style` blocks have the following structure:

- `palette` - (Optional) Color set to use to display nodes. One of "green_to_orange", "yellow_to_green", "YlOrRd" (warm), "hostmap_blues" (cool).
- `palette_flip` - (Optional) Boolean indicating whether to flip how the hostmap is rendered. For example, with the default palette, low values are represented as green, with high values as orange. If palette_flip is "true", then low values will be orange, and high values will be green.
- `fill_min` - (Optional) Metric value corresponding to minimum color fill.
- `fill_max` - (Optional) Metric value corresponding to maximum color fill.

Nested `widget` `tile_def` `marker` blocks

Only for widgets of type "timeseries".

Nested `widget` `tile_def` `marker` blocks have the following structure:

- `type` - (Required) How the marker lines will look. Possible values are {"error", "warning", "info", "ok"} {"dashed", "solid", "bold"}. Example: "error dashed".
- `value` - (Required) Mathematical expression describing the marker. Examples: "y > 1", "-5 < y < 0", "y = 19".
- `label` - (Optional) A label for the line or range.

Nested `widget` `tile_def` `event` block

Only for widgets of type "timeseries".

Nested `widget` `tile_def` `event` blocks have the following structure:

- `q` - (Required) The search query for event overlays.

Nested `widget` `tile_def` `request` blocks

Only for widgets of type "timeseries", "query_value", "toplist", "change", "hostmap", "process".

Nested `widget` `tile_def` `request` blocks have the following structure:

- `q` - (Optional, only for widgets of type "timeseries", "query_value", "toplist", "change", "hostmap") The query of the request. Pro tip: Use the JSON tab inside the Datadog UI to help build you query strings.

- `type` - (Optional, only for widgets of type "timeseries", "query_value", "hostmap") Choose the type of representation to use for this query. For widgets of type "timeseries" and "query_value", use one of "line", "bars" or "area". For widgets of type "hostmap", use "fill" or "size".
- `query_type` - (Optional, only for widgets of type "process") Use "process".
- `metric` - (Optional, only for widgets of type "process") The metric you want to use for the widget.
- `text_filter` - (Optional, only for widgets of type "process") The search query for the widget.
- `tag_filters` - (Optional, only for widgets of type "process") Tags to use for filtering.
- `limit` - (Optional, only for widgets of type "process") Integer indicating the number of hosts to limit to.
- `aggregator` - (Optional, only for widgets of type "query_value") The aggregator to use for time aggregation. One of "avg", "min", "max", "sum", "last".
- `compare_to` - (Optional, only for widgets of type "change") Choose from when to compare current data to. One of "hour_before", "day_before", "week_before" or "month_before".
- `change_type` - (Optional, only for widgets of type "change") Whether to show absolute or relative change. One of "absolute", "relative".
- `order_by` - (Optional, only for widgets of type "change") One of "change", "name", "present" (present value) or "past" (past value).
- `order_dir` - (Optional, only for widgets of type "change") Either "asc" (ascending) or "desc" (descending).
- `extra_col` - (Optional, only for widgets of type "change") If set to "present", displays current value. Can be left empty otherwise.
- `increase_good` - (Optional, only for widgets of type "change") Boolean indicating whether an increase in the value is good (thus displayed in green) or not (thus displayed in red).
- `style` - (Optional, only for widgets of type "timeseries", "query_value", "toplist", "process") describing how to display the widget. The structure of this block is described below. At most one such block should be present in a given request block.
- `conditional_format` - (Optional) Nested block to customize the style if certain conditions are met. Currently only applies to `Query Value` and `Top List` type graphs.

Nested `widget` `tile_def` `request` `style` block

Only for widgets of type "timeseries", "query_value", "toplist", "process".

The nested `style` blocks has the following structure:

- `palette` - (Optional) Color of the line drawn. For widgets of type "timeseries", "query_value", "toplist", one of: "classic", "cool", "warm", "purple", "orange" or "gray". For widgets of type "process", one of: "dog_classic_area", "YlOrRd", "GnBu", "Reds", "Oranges", "Greens", "Blues", "Purples".
- `width` - (Optional) Line width. Possible values: "thin", "normal", "thick". Default: "normal".
- `type` - (Optional) Type of line drawn. Possible values: "dashed", "solid", "dotted". Default: "solid".

Nested `widget` `tile_def` `request` `conditional_format` block

The nested `conditional_format` blocks has the following structure:

- `palette` - (Optional) Color scheme to be used if the condition is met. One of: "red_on_white", "white_on_red", "yellow_on_white", "white_on_yellow", "green_on_white", "white_on_green", "gray_on_white", "white_on_gray", "custom_text", "custom_bg", "custom_image".
- `comparator` - (Required) Comparison operator. Example: ">", "<".
- `value` - (Optional) Value that is the threshold for the conditional format.
- `color` - (Optional) Custom color (e.g., #205081).
- `invert` - (Optional) Boolean indicating whether to invert color scheme.

Nested `template_variable` blocks

Nested `template_variable` blocks have the following structure:

- `name` - (Required) The variable name. Can be referenced as $name in `graph` `request` `q` query strings.
- `prefix` - (Optional) The tag group. Default: no tag group.
- `default` - (Optional) The default tag. Default: "*" (match all).

Attributes Reference

The following attributes are exported:

- `id` - The unique ID of this screenboard in your Datadog account. The web interface URL to this screenboard can be generated by appending this ID to `https://app.datadoghq.com/screen/`

Import

screenboards can be imported using their numeric ID, e.g.

```
$ terraform import datadog_screenboard.my_service_screenboard 2081
```

datadog_timeboard

Provides a Datadog timeboard resource. This can be used to create and manage Datadog timeboards.

Example Usage

```
# Create a new Datadog timeboard
resource "datadog_timeboard" "redis" {
  title       = "Redis Timeboard (created via Terraform)"
  description = "created using the Datadog provider in Terraform"
  read_only   = true

  graph {
    title = "Redis latency (ms)"
    viz   = "timeseries"

    request {
      q    = "avg:redis.info.latency_ms{$host}"
      type = "bars"
    }
  }

  graph {
    title = "Redis memory usage"
    viz   = "timeseries"

    request {
      q       = "avg:redis.mem.used{$host} - avg:redis.mem.lua{$host}, avg:redis.mem.lua{$host}"
      stacked = true
    }

    request {
      q = "avg:redis.mem.rss{$host}"

      style {
        palette = "warm"
      }
    }
  }

  graph {
    title = "Top System CPU by Docker container"
    viz   = "toplist"

    request {
      q = "top(avg:docker.cpu.system{*} by {container_name}, 10, 'mean', 'desc')"
    }
  }

  template_variable {
    name   = "host"
    prefix = "host"
  }
}
```

Argument Reference

The following arguments are supported:

- `title` - (Required) The name of the dashboard.
- `description` - (Required) A description of the dashboard's content.
- `read_only` - (Optional) The read-only status of the timeboard. Default is false.
- `graph` - (Required) Nested block describing a graph definition. The structure of this block is described below. Multiple graph blocks are allowed within a datadog_timeboard resource.
- `template_variable` - (Optional) Nested block describing a template variable. The structure of this block is described below. Multiple template_variable blocks are allowed within a datadog_timeboard resource.

Nested `graph` blocks

Nested `graph` blocks have the following structure:

- `title` - (Required) The name of the graph.
- `viz` - (Required) The type of visualization to use for the graph. Valid choices are "change", "distribution", "heatmap", "hostmap", "query_value", timeseries", and "toplist".
- `request` - Nested block describing a graph definition request (a metric query to plot on the graph). The structure of this block is described below. Multiple request blocks are allowed within a graph block.
- `events` - (Optional) A list of event filter strings. Note that, while supported by the Datadog API, the Datadog UI does not (currently) support multiple event filters very well, so use at your own risk.
- `autoscale` - (Optional) Boolean that determines whether to autoscale graphs.
- `precision` - (Optional) Number of digits displayed, use `*` for full precision.
- `custom_unit` - (Optional) Display a custom unit on the graph (such as 'hertz')
- `text_align` - (Optional) How to align text in the graph, can be one of 'left', 'center', or 'right'.
- `style` - (Optional) Nested block describing hostmaps. The structure of this block is described below.
- `group` - (Optional) List of groups for hostmaps (shown as 'group by' in the UI).
- `include_no_metric_hosts` - (Optional) If set to true, will display hosts on hostmap that have no reported metrics.
- `include_ungrouped_hosts` - (Optional) If set to true, will display hosts without groups on hostmaps.
- `node_type` - (Optional) What nodes to display in a hostmap. Can be one of 'host' (default) or 'container'.
- `scope` - (Optional) List of scopes for hostmaps (shown as 'filter by' in the UI).
- `yaxis` - (Optional) Nested block describing modifications to the yaxis rendering. The structure of this block is described below.
- `marker` - (Optional) Nested block describing lines / ranges added to graph for formatting. The structure of this block is described below. Multiple marker blocks are allowed within a graph block.

Nested `graph` `marker` blocks

Nested `graph` `marker` blocks have the following structure:

- `type` - (Required) How the marker lines will look. Possible values are {"error", "warning", "info", "ok"} {"dashed", "solid", "bold"}. Example: "error dashed".
- `value` - (Required) Mathematical expression describing the marker. Examples: "y > 1", "-5 < y < 0", "y = 19".
- `label` - (Optional) A label for the line or range. **Warning:** when a label is enabled but left empty through the UI, the Datadog API returns a boolean value, not a string. This makes `terraform plan` fail with a JSON decoding error.

Nested `graph` `yaxis` block

- `min` - (Optional) Minimum bound for the graph's yaxis, a string.
- `max` - (Optional) Maximum bound for the graph's yaxis, a string.
- `scale` - (Optional) How to scale the yaxis. Possible values are: "linear", "log", "sqrt", "pow##" (eg. pow2, pow0.5, 2 is used if only "pow" was provided). Default: "linear".

Nested `graph` `request` blocks

Nested `graph` `request` blocks have the following structure:

- `q` - (Required) The query of the request. Pro tip: Use the JSON tab inside the Datadog UI to help build you query strings.
- `aggregator` - (Optional) The aggregation method used when the number of data points outnumbers the max that can be shown.
- `stacked` - (Optional) Boolean value to determine if this is this a stacked area graph. Default: false (line chart).
- `type` - (Optional) Choose how to draw the graph. For example: "line", "bars" or "area". Default: "line".
- `style` - (Optional) Nested block to customize the graph style.
- `conditional_format` - (Optional) Nested block to customize the graph style if certain conditions are met. Currently only applies to `Query Value` and `Top List` type graphs.

Nested `graph` `style` block

The nested `style` block is used specifically for styling `hostmap` graphs, and has the following structure:

- `fill_max` - (Optional) Maximum value for the hostmap fill query.
- `fill_min` - (Optional) Minimum value for the hostmap fill query.
- `palette` - (Optional) Spectrum of colors to use when styling a hostmap. For example: "green_to_orange", "yellow_to_green", "YlOrRd", or "hostmap_blues". Default: "green_to_orange".
- `palette_flip` - (Optional) Flip how the hostmap is rendered. For example, with the default palette, low values are represented as green, with high values as orange. If palette_flip is "true", then low values will be orange, and high

values will be green.

Nested `graph` `request` `style` block

The nested `style` blocks has the following structure:

- `palette` - (Optional) Color of the line drawn. For example: "classic", "cool", "warm", "purple", "orange" or "gray". Default: "classic".
- `width` - (Optional) Line width. Possible values: "thin", "normal", "thick". Default: "normal".
- `type` - (Optional) Type of line drawn. Possible values: "dashed", "solid", "dotted". Default: "solid".

Nested `graph` `request` `conditional_format` block

The nested `conditional_format` blocks has the following structure:

- `palette` - (Optional) Color scheme to be used if the condition is met. For example: "red_on_white", "white_on_red", "yellow_on_white", "white_on_yellow", "green_on_white", "white_on_green", "gray_on_white", "white_on_gray", "custom_text", "custom_bg", "custom_image".
- `comparator` - (Required) Comparison operator. Example: ">", "<".
- `value` - (Optional) Value that is the threshold for the conditional format.
- `custom_fg_color` - (Optional) Used when `palette` is set to `custom_text`. Set the color of the text to a custom web color, such as "#205081".
- `custom_bg_color` - (Optional) Used when `palette` is set to `custom_bg`. Set the color of the background to a custom web color, such as "#205081".

Nested `template_variable` blocks

Nested `template_variable` blocks have the following structure:

- `name` - (Required) The variable name. Can be referenced as $name in `graph` `request` `q` query strings.
- `prefix` - (Optional) The tag group. Default: no tag group.
- `default` - (Optional) The default tag. Default: "*" (match all).

Attributes Reference

The following attributes are exported:

- `id` - The unique ID of this timeboard in your Datadog account. The web interface URL to this timeboard can be generated by appending this ID to `https://app.datadoghq.com/dash/`

Import

Timeboards can be imported using their numeric ID, e.g.

```
$ terraform import datadog_timeboard.my_service_timeboard 2081
```

datadog_user

Provides a Datadog user resource. This can be used to create and manage Datadog users.

Example Usage

```
# Create a new Datadog user
resource "datadog_user" "foo" {
  email  = "new@example.com"
  handle = "new@example.com"
  name   = "New User"
}
```

Argument Reference

The following arguments are supported:

- `disabled` - (Optional) Whether the user is disabled
- `email` - (Required) Email address for user
- `handle` - (Required) The user handle, must be a valid email.
- `is_admin` - (Deprecated) (Optional) Whether the user is an administrator
- `name` - (Required) Name for user
- `role` - (Deprecated) Role description for user. **Warning**: the corresponding query parameter is ignored by the Datadog API, thus the argument would always trigger an execution plan.

Attributes Reference

The following attributes are exported:

- `disabled` - Returns true if Datadog user is disabled (NOTE: Datadog does not actually delete users so this will be true for those as well)
- `id` - ID of the Datadog user
- `verified` - Returns true if Datadog user is verified

Import

users can be imported using their handle, e.g.

```
$ terraform import datadog_user.example_user existing@example.com
```