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. Note that this URL must not end with the `/api/` path. For example, `https://api.datadoghq.com/` is a correct value, while `https://api.datadoghq.com/api/` is not.

datadog_ip_ranges

Use this data source to retrieve information about Datadog's IP addresses.

Example Usage

```
data "datadog_ip_ranges" "test" {}
```

Attributes Reference

- `agents_ipv4` - An Array of IPv4 addresses in CIDR format specifying the A records for the agent endpoint.
- `api_ipv4` - An Array of IPv4 addresses in CIDR format specifying the A records for the api endpoint.
- `apm_ipv4` - An Array of IPv4 addresses in CIDR format specifying the A records for the apm endpoint.
- `logs_ipv4` - An Array of IPv4 addresses in CIDR format specifying the A records for the logs endpoint.
- `process_ipv4` - An Array of IPv4 addresses in CIDR format specifying the A records for the process endpoint.
- `synthetics_ipv4` - An Array of IPv4 addresses in CIDR format specifying the A records for the synthetics endpoint.
- `webhooks_ipv4` - An Array of IPv4 addresses in CIDR format specifying the A records for the webhooks endpoint.
- `agents_ipv6` - An Array of IPv6 addresses in CIDR format specifying the A records for the agent endpoint.
- `api_ipv6` - An Array of IPv6 addresses in CIDR format specifying the A records for the api endpoint.
- `apm_ipv6` - An Array of IPv6 addresses in CIDR format specifying the A records for the apm endpoint.
- `logs_ipv6` - An Array of IPv6 addresses in CIDR format specifying the A records for the logs endpoint.
- `process_ipv6` - An Array of IPv6 addresses in CIDR format specifying the A records for the process endpoint.
- `synthetics_ipv6` - An Array of IPv6 addresses in CIDR format specifying the A records for the synthetics endpoint.
- `webhooks_ipv6` - An Array of IPv6 addresses in CIDR format specifying the A records for the webhooks endpoint.

datadog_dashboard

Provides a Datadog dashboard resource. This can be used to create and manage Datadog dashboards.

Note: This resource uses the new Dashboard API (https://docs.datadoghq.com/api/#dashboards) which adds new features like better validation and support for the Group widget (https://docs.datadoghq.com/graphing/widgets/group/). Additionally, this resource unifies `datadog_timeboard` (/docs/providers/datadog/r/timeboard.html) and `datadog_screenboard` (/docs/providers/datadog/r/screenboard.html) resources to allow you to manage all of your dashboards using a single format.

Example Usage: Create a new Datadog dashboard - Ordered layout

```
resource "datadog_dashboard" "ordered_dashboard" {
  title         = "Ordered Layout Dashboard"
  description   = "Created using the Datadog provider in Terraform"
  layout_type   = "ordered"
  is_read_only  = true

  widget {
    alert_graph_definition {
      alert_id = "895605"
      viz_type = "timeseries"
      title = "Widget Title"
      time = {
        live_span = "1h"
      }
    }
  }

  widget {
    alert_value_definition {
      alert_id = "895605"
      precision = 3
      unit = "b"
      text_align = "center"
      title = "Widget Title"
    }
  }

  widget {
    alert_value_definition {
      alert_id = "895605"
      precision = 3
      unit = "b"
      text_align = "center"
      title = "Widget Title"
    }
  }

  widget {
    change_definition {
```

```
    request {
      q = "avg:system.load.1{env:staging} by {account}"
      change_type = "absolute"
      compare_to = "week_before"
      increase_good = true
      order_by = "name"
      order_dir = "desc"
      show_present = true
    }
    title = "Widget Title"
    time = {
      live_span = "1h"
    }
  }
}

widget {
  distribution_definition {
    request {
      q = "avg:system.load.1{env:staging} by {account}"
      style {
        palette = "warm"
      }
    }
    title = "Widget Title"
    time = {
      live_span = "1h"
    }
  }
}

widget {
  check_status_definition {
    check = "aws.ecs.agent_connected"
    grouping = "cluster"
    group_by = ["account", "cluster"]
    tags = ["account:demo", "cluster:awseb-ruthebdog-env-8-dn3m6u3gvk"]
    title = "Widget Title"
    time = {
      live_span = "1h"
    }
  }
}

widget {
  heatmap_definition {
    request {
      q = "avg:system.load.1{env:staging} by {account}"
      style {
        palette = "warm"
      }
    }
    yaxis {
      min = 1
      max = 2
      include_zero = true
      scale = "sqrt"
    }
    title = "Widget Title"
    time = {
```

```
      live_span = "1h"
    }
  }
}

widget {
  hostmap_definition {
    request {
      fill {
        q = "avg:system.load.1{*} by {host}"
      }
      size {
        q = "avg:memcache.uptime{*} by {host}"
      }
    }
    node_type= "container"
    group = ["host", "region"]
    no_group_hosts = true
    no_metric_hosts = true
    scope = ["region:us-east-1", "aws_account:727006795293"]
    style {
      palette = "yellow_to_green"
      palette_flip = true
      fill_min = "10"
      fill_max = "20"
    }
    title = "Widget Title"
  }
}

widget {
  note_definition {
    content = "note text"
    background_color = "pink"
    font_size = "14"
    text_align = "center"
    show_tick = true
    tick_edge = "left"
    tick_pos = "50%"
  }
}

widget {
  query_value_definition {
    request {
      q = "avg:system.load.1{env:staging} by {account}"
      aggregator = "sum"
      conditional_formats {
        comparator = "<"
        value = "2"
        palette = "white_on_green"
      }
      conditional_formats {
        comparator = ">"
        value = "2.2"
        palette = "white_on_red"
      }
    }
    autoscale = true
```

```
    custom_unit = "xx"
    precision = "4"
    text_align = "right"
    title = "Widget Title"
    time = {
      live_span = "1h"
    }
  }
}

widget {
  scatterplot_definition {
    request {
      x {
        q = "avg:system.cpu.user{*} by {service, account}"
        aggregator = "max"
      }
      y {
        q = "avg:system.mem.used{*} by {service, account}"
        aggregator = "min"
      }
    }
    color_by_groups = ["account", "apm-role-group"]
    xaxis {
      include_zero = true
      label = "x"
      min = "1"
      max = "2000"
      scale = "pow"
    }
    yaxis {
      include_zero = false
      label = "y"
      min = "5"
      max = "2222"
      scale = "log"
    }
    title = "Widget Title"
    time = {
      live_span = "1h"
    }
  }
}

widget {
  timeseries_definition {
    request {
      q= "avg:system.cpu.user{app:general} by {env}"
      display_type = "line"
      style {
        palette = "warm"
        line_type = "dashed"
        line_width = "thin"
      }
      metadata {
        expression = "avg:system.cpu.user{app:general} by {env}"
        alias_name = "Alpha"
      }
    }
    request {
```

```
request {
  log_query {
    index = "mcnulty"
    compute = {
      aggregation = "avg"
      facet = "@duration"
      interval = 5000
    }
    search = {
      query = "status:info"
    }
    group_by {
      facet = "host"
      limit = 10
      sort = {
        aggregation = "avg"
        order = "desc"
        facet = "@duration"
      }
    }
  }
  display_type = "area"
}
request {
  apm_query {
    index = "apm-search"
    compute = {
      aggregation = "avg"
      facet = "@duration"
      interval = 5000
    }
    search = {
      query = "type:web"
    }
    group_by {
      facet = "resource_name"
      limit = 50
      sort = {
        aggregation = "avg"
        order = "desc"
        facet = "@string_query.interval"
      }
    }
  }
  display_type = "bars"
}
request {
  process_query {
    metric = "process.stat.cpu.total_pct"
    search_by = "error"
    filter_by = ["active"]
    limit = 50
  }
  display_type = "area"
}
marker {
  display_type = "error dashed"
  label = " z=6 "
  value = "y = 4"
}
```

```
    marker {
      display_type = "ok solid"
      value = "10 < y < 999"
      label = " x=8 "
    }
    title = "Widget Title"
    show_legend = true
    time = {
      live_span = "1h"
    }
    event {
      q = "sources:test tags:1"
    }
    event {
      q = "sources:test tags:2"
    }
    yaxis {
      scale = "log"
      include_zero = false
      max = 100
    }
  }
}

widget {
  toplist_definition {
    request {
      q= "avg:system.cpu.user{app:general} by {env}"
      conditional_formats {
        comparator = "<"
        value = "2"
        palette = "white_on_green"
      }
      conditional_formats {
        comparator = ">"
        value = "2.2"
        palette = "white_on_red"
      }
    }
    title = "Widget Title"
  }
}

widget {
  group_definition {
    layout_type = "ordered"
    title = "Group Widget"

    widget {
      note_definition {
        content = "cluster note widget"
        background_color = "pink"
        font_size = "14"
        text_align = "center"
        show_tick = true
        tick_edge = "left"
        tick_pos = "50%"
      }
    }
```

```
    widget {
      alert_graph_definition {
        alert_id = "123"
        viz_type = "toplist"
        title = "Alert Graph"
        time = {
          live_span = "1h"
        }
      }
    }
  }

  template_variable {
    name   = "var_1"
    prefix = "host"
    default = "aws"
  }
  template_variable {
    name   = "var_2"
    prefix = "service_name"
    default = "autoscaling"
  }
}
```

Example Usage: Create a new Datadog dashboard - Free layout

```
resource "datadog_dashboard" "free_dashboard" {
  title         = "Free Layout Dashboard"
  description   = "Created using the Datadog provider in Terraform"
  layout_type   = "free"
  is_read_only  = false

  widget {
    event_stream_definition {
      query = "*"
      event_size = "l"
      title = "Widget Title"
      title_size = 16
      title_align = "left"
      time = {
        live_span = "1h"
      }
    }
    layout = {
      height = 43
      width = 32
      x = 5
      y = 5
    }
  }

  widget {
    event_timeline_definition {
      query = "*"
```

```
      title = "Widget Title"
      title_size = 16
      title_align = "left"
      time = {
        live_span = "1h"
      }
    }
    layout = {
      height = 9
      width = 65
      x = 42
      y = 73
    }
  }

  widget {
    free_text_definition {
      text = "free text content"
      color = "#d00"
      font_size = "88"
      text_align = "left"
    }
    layout = {
      height = 20
      width = 30
      x = 42
      y = 5
    }
  }

  widget {
    iframe_definition {
      url = "http://google.com"
    }
    layout = {
      height = 46
      width = 39
      x = 111
      y = 8
    }
  }

  widget {
    image_definition {
      url = "https://images.pexels.com/photos/67636/rose-blue-flower-rose-blooms-67636.jpeg?auto=compress
&cs=tinysrgb&h=350"
      sizing = "fit"
      margin = "small"
    }
    layout = {
      height = 20
      width = 30
      x = 77
      y = 7
    }
  }

  widget {
    log_stream_definition {
      logset = "10"
```

```
    tagset = "19"
    query = "error"
    columns = ["core_host", "core_service", "tag_source"]
  }
  layout = {
    height = 36
    width = 32
    x = 5
    y = 51
  }
}

widget {
  manage_status_definition {
    color_preference = "text"
    count = 50
    display_format = "countsAndList"
    hide_zero_counts = true
    query = "type:metric"
    sort = "status,asc"
    start = 0
    title = "Widget Title"
    title_size = 16
    title_align = "left"
  }
  layout = {
    height = 40
    width = 30
    x = 112
    y = 55
  }
}

widget {
  trace_service_definition {
    display_format = "three_column"
    env = "datad0g.com"
    service = "alerting-cassandra"
    show_breakdown = true
    show_distribution = true
    show_errors = true
    show_hits = true
    show_latency = false
    show_resource_list = false
    size_format = "large"
    span_name = "cassandra.query"
    title = "alerting-cassandra #env:datad0g.com"
    title_align = "center"
    title_size = "13"
    time = {
      live_span = "1h"
    }
  }
  layout = {
    height = 38
    width = 67
    x = 40
    y = 28
  }
}
```

```
  template_variable {
    name   = "var_1"
    prefix = "host"
    default = "aws"
  }
  template_variable {
    name   = "var_2"
    prefix = "service_name"
    default = "autoscaling"
  }
}
```

Argument Reference

The following arguments are supported:

- `title` - (Required) Title of the dashboard.
- `widget` - (Required) Nested block describing a widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-blocks). Multiple `widget` blocks are allowed within a `datadog_dashboard` resource
- `layout_type` - (Required) Layout type of the dashboard. Available values are: `ordered` (previous timeboard) or `free` (previous screenboard layout).
  Note: This value cannot be changed. Converting a dashboard from `free` <-> `ordered` requires destroying and re-creating the dashboard. Instead of using `ForceNew`, this is a manual action as many underlying widget configs need to be updated to work for the updated layout, otherwise the new dashboard won't be created properly.
- `description` - (Optional) Description of the dashboard.
- `is_read_only` - (Optional) Whether this dashboard is read-only. If `true`, only the author and admins can make changes to it.
- `notify_list` - (Optional) List of handles of users to notify when changes are made to this dashboard.
- `template_variables` - (Optional) Nested block describing a template variable. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-template_variable-blocks). Multiple template_variable blocks are allowed within a `datadog_dashboard` resource.

Nested `widget` blocks

Nested `widget` blocks have the following structure:

- `layout` - (Required for widgets in dashboards with `free` layout_type only). The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-layout-blocks)
- A widget should have exactly one of the following nested blocks describing the widget definition:
  - `alert_graph_definition`: The definition for a Alert Graph widget. Exactly one nested block is allowed with the following structure:
    - `alert_id`: (Required) The ID of the monitor used by the widget.

        - `viz_type` : (Required) Type of visualization to use when displaying the widget. Either "timeseries" or "toplist".
        - `title` : (Optional) The title of the widget.
        - `title_size` : (Optional) The size of the widget's title. Default is 16.
        - `title_align` : (Optional) The alignment of the widget's title. One of "left", "center", or "right"
        - `time` : (Optional) Nested block describing the timeframe to use when displaying the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-time-blocks).
    - `alert_value_definition` : The definition for an Alert Value widget. Exactly one nested block is allowed with the following structure:
        - `alert_id` : (Required) The ID of the monitor used by the widget.
        - `precision` : (Optional) The precision to use when displaying the value. Use "*" for maximum precision.
        - `unit` : (Optional) The unit for the value displayed in the widget.
        - `text_align` : (Optional) The alignment of the text in the widget.
        - `title` : (Optional) The title of the widget.
        - `title_size` : (Optional) The size of the widget's title. Default is 16.
        - `title_align` : (Optional) The alignment of the widget's title. One of "left", "center", or "right"
    - `change_definition` : The definition for a Change widget. Exactly one nested block is allowed with the following structure:
        - `request` : (Required) Nested block describing the request to use when displaying the widget. Multiple request blocks are allowed with the following structure:
            - `q` : (Required) The metric query to use in the widget.
            - `change_type` : (Optional) Whether to show absolute or relative change. One of "absolute", "relative".
            - `compare_to` - (Optional) Choose from when to compare current data to. One of "hour_before", "day_before", "week_before" or "month_before".
            - `increase_good` - (Optional) Boolean indicating whether an increase in the value is good (thus displayed in green) or not (thus displayed in red).
            - `order_by` - (Optional) One of "change", "name", "present" (present value) or "past" (past value).
            - `order_dir` - (Optional) Either "asc" (ascending) or "desc" (descending).
            - `show_present` - (Optional) If set to "true", displays current value.
        - `title` : (Optional) The title of the widget.
        - `title_size` : (Optional) The size of the widget's title. Default is 16.
        - `title_align` : (Optional) The alignment of the widget's title. One of "left", "center", or "right".
        - `time` : (Optional) Nested block describing the timeframe to use when displaying the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-time-blocks).
    - `check_status_definition` : The definition for a Check Status widget. Exactly one nested block is allowed with

the following structure:

- `check` - (Optional) The check to use in the widget.
- `grouping` - (Optional) Either "check" or "cluster", depending on whether the widget should use a single check or a cluster of checks.
- `group` - (Optional) The check group to use in the widget.
- `group_by` - (Optional) When grouping = "cluster", indicates a list of tags to use for grouping.
- `tags` - (Optional) List of tags to use in the widget.
- `title` : (Optional) The title of the widget.
- `title` : (Optional) The title of the widget.
- `title_size` : (Optional) The size of the widget's title. Default is 16.
- `title_align` : (Optional) The alignment of the widget's title. One of "left", "center", or "right".
- `time` : (Optional) Nested block describing the timeframe to use when displaying the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-time-blocks).

- `distribution_definition` : The definition for a Distribution widget. Exactly one nested block is allowed with the following structure:
  - `request` : (Required) Nested block describing the request to use when displaying the widget. Multiple request blocks are allowed with the following structure:
    - `q` : (Required) The metric query to use in the widget.
    - `style` - (Optional) Style of the widget graph. One nested block is allowed with the following structure:
      - `palette` - (Optional) Color palette to apply to the widget. The available options are available here: https://docs.datadoghq.com/graphing/widgets/timeseries/#appearance (https://docs.datadoghq.com/graphing/widgets/timeseries/#appearance).
  - `title` : (Optional) The title of the widget.
  - `title_size` : (Optional) The size of the widget's title. Default is 16.
  - `title_align` : (Optional) The alignment of the widget's title. One of "left", "center", or "right".
  - `time` : (Optional) Nested block describing the timeframe to use when displaying the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-time-blocks).

- `event_stream_definition` : The definition for a Event Stream widget. Exactly one nested block is allowed with the following structure:
  - `query` : (Required) The query to use in the widget.
  - `event_size` - (Optional) The size of the events in the widget. Either "s" (small, title only) or "l" (large, full event).
  - `title` : (Optional) The title of the widget.
  - `title_size` : (Optional) The size of the widget's title. Default is 16.
  - `title_align` : (Optional) The alignment of the widget's title. One of "left", "center", or "right".

        - `time` : (Optional) Nested block describing the timeframe to use when displaying the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-time-blocks).

    - `event_timeline_definition` : The definition for a Event Timeline widget. Exactly one nested block is allowed with the following structure:
        - `text` : (Required) The query to use in the widget.
        - `title` : (Optional) The title of the widget.
        - `title_size` : (Optional) The size of the widget's title. Default is 16.
        - `title_align` : (Optional) The alignment of the widget's title. One of "left", "center", or "right".
        - `time` : (Optional) Nested block describing the timeframe to use when displaying the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-time-blocks).

    - `free_text_definition` : The definition for a Free Text. Exactly one nested block is allowed with the following structure:
        - `text` - (Required) The text to display in the widget.
        - `color` - (Optional) The color of the text in the widget.
        - `font_size` - (Optional, "note") The size of the text in the widget.
        - `text_align` - (Optional, "alert_value", "note") The alignment of the text in the widget.

    - `heatmap_definition` : The definition for a Heatmap widget. Exactly one nested block is allowed with the following structure:
        - `request` : (Required) Nested block describing the request to use when displaying the widget. Multiple request blocks are allowed with the following structure:
            - `q` : (Required) The metric query to use in the widget.
            - `style` - (Optional) Style of the widget graph. One nested block is allowed with the following structure:
                - `palette` - (Optional) Color palette to apply to the widget. The available options are available here: https://docs.datadoghq.com/graphing/widgets/timeseries/#appearance (https://docs.datadoghq.com/graphing/widgets/timeseries/#appearance).
        - `yaxis` : (Optional) Nested block describing the Y-Axis Controls. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-axis-blocks)
        - `title` : (Optional) The title of the widget.
        - `title_size` : (Optional) The size of the widget's title. Default is 16.
        - `title_align` : (Optional) The alignment of the widget's title. One of "left", "center", or "right".
        - `time` : (Optional) Nested block describing the timeframe to use when displaying the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-time-blocks).

    - `hostmap_definition` : The definition for a Hostmap widget. Exactly one nested block is allowed with the following structure:
        - `request` : (Required) Nested block describing the request to use when displaying the widget. Multiple request blocks are allowed with the following structure:
            - `fill` : (Optional) The query used to fill the map. Exactly one nested block is allowed with the following structure:

            - `q` : (Required) The metric query to use in the widget.

        - `size` : (Optional) The query used to size the map. Exactly one nested block is allowed with the following structure:
            - `q` : (Required) The metric query to use in the widget.

    - `node_type` - (Optional) The type of node used. Either "host" or "container".
    - `no_metric_hosts` - (Optional) Boolean indicating whether to show nodes with no metrics.
    - `no_group_hosts` - (Optional) Boolean indicating whether to show ungrouped nodes.
    - `group` - (Optional) The list of tags to group nodes by.
    - `scope` - (Optional) The list of tags to filter nodes by.
    - `style` - (Optional) Style of the widget graph. One nested block is allowed with the following structure:
        - `palette` - (Optional) Color palette to apply to the widget. The available options are available here: https://docs.datadoghq.com/graphing/widgets/timeseries/#appearance (https://docs.datadoghq.com/graphing/widgets/timeseries/#appearance).
        - `palette_flip` - (Optional) Boolean indicating whether to flip the palette tones.
        - `fill_min` - (Optional) Min value to use to color the map.
        - `fill_max` - (Optional) Max value to use to color the map.
    - `title` : (Optional) The title of the widget.
    - `title_size` : (Optional) The size of the widget's title. Default is 16.
    - `title_align` : (Optional) The alignment of the widget's title. One of "left", "center", or "right".

- `iframe_definition` : The definition for a Iframe widget. Exactly one nested block is allowed with the following structure:
    - `url` - (Rquired) The URL to use as a data source for the widget.

- `image_definition` : The definition for a Image widget. Exactly one nested block is allowed with the following structure:
    - `url` - (Rquired) The URL to use as a data source for the widget.
    - `sizing` - (Optional) 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) The margins to use around the image. Either "small" or "large".

- `log_stream_definition` : The definition for a Log Stream widget. Exactly one nested block is allowed with the following structure:
    - `logset` - (Required) ID of the logset to use.
    - `query` : (Optional) The query to use in the widget.
    - `columns` - (Optional) Stringified list of columns to use. Example: `"["column1","column2","column3"]"` .
    - `title` : (Optional) The title of the widget.
    - `title_size` : (Optional) The size of the widget's title. Default is 16.

        - `title_align` : (Optional) The alignment of the widget's title. One of "left", "center", or "right".
        - `time` : (Optional) Nested block describing the timeframe to use when displaying the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-time-blocks).
    - `manage_status_definition` : The definition for a Manage Status widget. Exactly one nested block is allowed with the following structure:
        - `query` : (Required) The query to use in the widget.
        - `sort` - (Optional) The method to use to sort monitors. One of : "desc" or "asc". `count` - (Optional) The number of monitors to display. `start` - (Optional) The start of the list. Typically 0.
        - `display_format` - (Optional") The display setting to use. One of "counts", "list", or "countsAndList".
        - `color_preference` - (Optional") Whether to colorize text or background. One of "text", "background".
        - `hide_zero_counts` - (Optional") Boolean indicating whether to hide empty categories.
        - `title` : (Optional) The title of the widget.
        - `title_size` : (Optional) The size of the widget's title. Default is 16.
        - `title_align` : (Optional) The alignment of the widget's title. One of "left", "center", or "right".
    - `note_definition` : The definition for a Note widget. Exactly one nested block is allowed with the following structure:
        - `content` - (Required) Content of the note
        - `background_color` - (Optional) Background color of the note.
        - `font_size` - (Optional) Size of the text.
        - `text_align` - (Optional) How to align the text on the widget. Available values are: `center` , `left` , or `right` .
        - `show_tick` - (Optional) Whether to show a tick or not.
        - `tick_pos` - (Optional") 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") When tick = true, string indicating on which side of the widget the tick should be displayed. One of "bottom", "top", "left", "right".
    - `query_value_definition` : The definition for a Query Value widget. Exactly one nested block is allowed with the following structure:
        - `request` : (Required) Nested block describing the request to use when displaying the widget. Multiple request blocks are allowed with the following structure (exactly only one of `q` , `apm_query` , `log_query` or `process_query` is required within the request block):
            - `q` : (Optional) The metric query to use in the widget
            - `apm_query` : (Optional) The APM query to use in the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-apm_query-and-log_query-blocks).
            - `log_query` : (Optional) The log query to use in the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-apm_query-and-log_query-blocks).
            - `process_query` : (Optional) The process query to use in the widget. The structure of this block is

described below (/docs/providers/datadog/r/dashboard.html#nested-process_query-blocks).

- `conditional_formats` - (Optional) Conditional formats allow you to set the color of your widget content or background, depending on a rule applied to your data. Multiple request blocks are allowed. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-conditional_formats-blocks).

- `aggregator` - (Optional) The aggregator to use for time aggregation. One of `avg`, `min`, `max`, `sum`, `last`.

- `autoscale` - (Optional) Boolean indicating whether to automatically scale the tile.
- `custom_unit` - (Optional) The unit for the value displayed in the widget
- `precision` - (Optional) The precision to use when displaying the tile.
- `text_align` - (Optional, "alert_value", "note") The alignment of the text in the widget.
- `title` : (Optional) The title of the widget.
- `title_size` : (Optional) The size of the widget's title. Default is 16.
- `title_align` : (Optional) The alignment of the widget's title. One of "left", "center", or "right".
- `time` : (Optional) Nested block describing the timeframe to use when displaying the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-time-blocks).

- `scatterplot_definition` : The definition for a Scatterplot widget. Exactly one nested block is allowed with the following structure:
  - `request` : (Required) Nested block describing the request to use when displaying the widget. Exactly one request block is allowed with the following structure:
    - `x` : (Optional) The query used for the X-Axis. Exactly one nested block is allowed with the following structure:
      - `q` : (Required) The metric query to use in the widget.
      - `aggregator` - (Optional) Aggregator used for the request. One of "avg", "min", "max", "sum", "last".
    - `y` : (Optional) The query used for the Y-Axis. Exactly one nested block is allowed with the following structure:
      - `q` : (Required) The metric query to use in the widget.
      - `aggregator` - (Optional) Aggregator used for the request. One of "avg", "min", "max", "sum", "last".
  - `xaxis` : (Optional) Nested block describing the X-Axis Controls. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-axis-blocks)
  - `yaxis` : (Optional) Nested block describing the Y-Axis Controls. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-axis-blocks)
  - `color_by_groups` - (Optional) List of groups used for colors.
  - `title` : (Optional) The title of the widget.
  - `title_size` : (Optional) The size of the widget's title. Default is 16.

        - `title_align` : (Optional) The alignment of the widget's title. One of "left", "center", or "right".
        - `time` : (Optional) Nested block describing the timeframe to use when displaying the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-time-blocks).
    - `timeseries_definition` : The definition for a Timeseries widget. Exactly one nested block is allowed with the following structure:
        - `request` : (Required) Nested block describing the request to use when displaying the widget. Multiple request blocks are allowed with the following structure (exactly only one of `q`, `apm_query`, `log_query` or `process_query` is required within the request block):
            - `q` : (Optional) The metric query to use in the widget
            - `apm_query` : (Optional) The APM query to use in the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-apm_query-and-log_query-blocks).
            - `log_query` : (Optional) The log query to use in the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-apm_query-and-log_query-blocks).
            - `process_query` : (Optional) The process query to use in the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-process_query-blocks).
            - `display_type` - (Optional) Type of display to use for the request. Available values are: `area`, `bars`, or `line`.
            - `style` - (Optional) Style of the widget graph. One nested block is allowed with the following structure:
                - `palette` - (Optional) Color palette to apply to the widget. The available options are available here: https://docs.datadoghq.com/graphing/widgets/timeseries/#appearance (https://docs.datadoghq.com/graphing/widgets/timeseries/#appearance).
                - `line_type` - (Optional) Type of lines displayed. Available values are: `dashed`, `dotted`, or `solid`.
                - `line_width` - (Optional) Width of line displayed. Available values are: `normal`, `thick`, or `thin`.
            - `metadata` - (Optional). Used to define expression aliases. Multiple nested blocks are allowed with the following structure:
                - `expression` - (Required)
                - `alias_name` - (Optional)
        - `marker` - (Optional) Nested block describing the marker to use when displaying the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widgetmarker-blocks). Multiple marker blocks are allowed within a given tile_def block.
        - `title` : (Optional) The title of the widget.
        - `title_size` : (Optional) The size of the widget's title. Default is 16.
        - `title_align` : (Optional) The alignment of the widget's title. One of "left", "center", or "right".
        - `time` : (Optional) Nested block describing the timeframe to use when displaying the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-time-blocks).
        - `show_legend` : (Optional) Whether or not to show the legend on this widget.

        - `event` : (Optional) The definition of the event to overlay on the graph. Includes the following structure:
            - `q` : (Required) The event query to use in the widget
        - `yaxis` : (Optional) Nested block describing the Y-Axis Controls. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-axis-blocks)
    - `toplist_definition` : The definition for a Toplist widget. Exactly one nested block is allowed with the following structure:
        - `request` : (Required) Nested block describing the request to use when displaying the widget. Multiple request blocks are allowed with the following structure (exactly only one of `q`, `apm_query`, `log_query` or `process_query` is required within the request block):
            - `q` : (Optional) The metric query to use in the widget
            - `apm_query` : (Optional) The APM query to use in the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-apm_query-and-log_query-blocks).
            - `log_query` : (Optional) The log query to use in the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-apm_query-and-log_query-blocks).
            - `process_query` : (Optional) The process query to use in the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-process_query-blocks).
            - `conditional_formats` - (Optional) Conditional formats allow you to set the color of your widget content or background, depending on a rule applied to your data. Multiple request blocks are allowed. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-conditional_formats-blocks).
        - `title` : (Optional) The title of the widget.
        - `title_size` : (Optional) The size of the widget's title. Default is 16.
        - `title_align` : (Optional) The alignment of the widget's title. One of "left", "center", or "right".
        - `time` : (Optional) Nested block describing the timeframe to use when displaying the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-time-blocks).
    - `trace_service_definition` : The definition for a Trace Service widget. Exactly one nested block is allowed with the following structure:
        - `env` : (Required) APM environment.
        - `service` : (Required) APM service.
        - `span_name` : (Required) APM span name.
        - `show_hits` : (Optional) APM span name.
        - `show_hits` : (Optional) Whether to show the hits metrics or not.
        - `show_errors` : (Optional) Whether to show the error metrics or not.
        - `show_latency` : (Optional) Whether to show the latency metrics or not.
        - `show_breakdown` : (Optional) Whether to show the latency breakdown or not.
        - `show_distribution` : (Optional) Whether to show the latency distribution or not.
        - `show_resource_list` : (Optional) Whether to show the resource list or not.

- `size_format` : (Optional) Size of the widget. Available values are: `small` , `medium` , or `large` .
- `display_format` : (Optional) Number of columns to display. Available values are: `one_column` , `two_column` , or `three_column` .
- `title` : (Optional) The title of the widget.
- `title_size` : (Optional) The size of the widget's title. Default is 16.
- `title_align` : (Optional) The alignment of the widget's title. One of "left", "center", or "right".
- `time` : (Optional) Nested block describing the timeframe to use when displaying the widget. The structure of this block is described below (/docs/providers/datadog/r/dashboard.html#nested-widget-time-blocks).

Nested `widget layout` blocks

Nested `layout` blocks have the following structure:

- `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.
- `width` - (Required) The width of the widget.
- `height` - (Required) The height of the widget.

Nested `widget axis` blocks

Nested `axis` blocks have the following structure:

- `label` - (Optional) The label of the axis to display on the graph.
- `scale` - (Optional) Specifies the scale type. One of "linear", "log", "pow", "sqrt".
- `min` - (Optional) Specify the minimum value to show on y-axis.
- `max` - (Optional) Specify the minimum value to show on y-axis.
- `include_zero` - (Optional) Always include zero or fit the axis to the data range.

Nested `widget conditional_formats` blocks

Nested `conditional_formats` blocks have the following structure:

- `comparator` - (Required) Comparator to apply from: One of `>` , `>=` , `<` , or `<=` .
- `value` - (Required) Value for the comparator.
- `palette` - (Required) Color palette to apply; One of `blue` , `custom_bg` , `custom_image` , `custom_text` , `gray_on_white` , `green` , `green_on_white` , `grey` , `orange` , `red` , `red_on_white` , `white_on_gray` , `white_on_green` , `white_on_red` , `white_on_yellow` , or `yellow_on_white` .
- `custom_bg_color` - (Optional) Color palette to apply to the background, same values available as palette.

- `custom_fg_color` - (Optional) Color palette to apply to the foreground, same values available as palette.
- `image_url` - (Optional) Displays an image as the background. .

Nested `widget time` blocks

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 `apm_query` and `log_query` blocks

Nested `apm_query` and `log_query` blocks have the following structure (Visit the Graph Primer (https://docs.datadoghq.com/graphing/) for more information about these values):

- `index` - (Required)
- `compute` - (Required). Exactly one nested block is required with the following structure:
  - `aggregation` - (Required)
  - `facet` - (Optional)
  - `interval` - (Optional)
- `search` - (Optional). One nested block is allowed with the following structure:
  - `query` - (Optional)
- `group_by` - (Optional). Multiple nested blocks are allowed with the following structure:
  - `facet` - (Optional)
  - `limit` - (Optional)
  - `sort` - (Optional). One nested block is allowed with the following structure:
    - `aggregation` - (Required)
    - `order` - (Required)
    - `facet` - (Optional)

Nested `process_query` blocks

Nested `process_query` blocks have the following structure (Visit the Graph Primer (https://docs.datadoghq.com/graphing/) for more information about these values):

- `metric` - (Required)
- `search_by` - (Required)
- `filter_by` - (Required)
- `limit` - (Required)

Nested `widget marker` blocks

Only for widgets of type "timeseries".

Nested `widget marker` blocks have the following structure:

- `display_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 `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).

Import

dashboards can be imported using their ID, e.g.

```
$ terraform import datadog_dashboard.my_service_dashboard sv7-gyh-kas
```

datadog_dashboard_list

Provides a Datadog dashboard_list resource. This can be used to create and manage Datadog Dashboard Lists and the individual dashboards within them.

Example Usage

Create a new Dashboard list with two dashbaords

```
resource "datadog_dashboard_list" "new_list" {
    depends_on = [
        "datadog_dashboard.screen",
        "datadog_dashboard.time"
    ]

    name = "Terraform Created List"
    dash_item {
        type = "custom_timeboard"
        dash_id = "${datadog_dashboard.time.id}"
    }
    dash_item {
        type = "custom_screenboard"
        dash_id = "${datadog_dashboard.screen.id}"
    }
}

resource "datadog_dashboard" "time" {
    title         = "TF Test Layout Dashboard"
    description   = "Created using the Datadog provider in Terraform"
    layout_type   = "ordered"
    is_read_only  = true
    widget {
        alert_graph_definition {
          alert_id = "1234"
          viz_type = "timeseries"
          title = "Widget Title"
          time = {
            live_span = "1h"
          }
        }
      }

}

resource "datadog_dashboard" "screen" {
    title         = "TF Test Free Layout Dashboard"
    description   = "Created using the Datadog provider in Terraform"
    layout_type   = "free"
    is_read_only  = false
    widget {
        event_stream_definition {
          query = "*"
          event_size = "l"
          title = "Widget Title"
          title_size = 16
```

```
            title_size = 16
            title_align = "left"
            time = {
              live_span = "1h"
            }
          }
          layout = {
            height = 43
            width = 32
            x = 5
            y = 5
          }
        }

}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of this Dashbaord List.
- `dash_item` - (Optional) An individual dashboard object to add to this Dashboard List. If present, must contain the following:
  - `type` - (Required) The type of this dashboard. Available options are: `custom_timeboard`, `custom_screenboard`, `integration_screenboard`, `integration_timeboard`, and `host_timeboard`
  - `dash_id` - (Required) The ID of this dashboard.

Import

dashboard lists can be imported using their id, e.g.

```
$ terraform import datadog_dashboard_list.new_list 123456
```

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
  }
}
```

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.
- `timezone` (Optional) The timezone for the downtime, default UTC. It must be a valid IANA Time Zone.
- `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_tags` - (Optional) A list of monitor tags to match. The resulting downtime applies to monitors that match **all** provided monitor tags. This option conflicts with `monitor_id` as it will match all monitors that match these tags.
- `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. **Note:** this will effectively change the `silenced` attribute of the referenced monitor. If that monitor is also tracked by Terraform and you don't want it to be unmuted on the next `terraform apply`, see details (/docs/providers/datadog/r/monitor.html#silencing-by-hand-and-by-downtimes) in the monitor resource documentation. This option also conflicts with `monitor_tags` use none or one or the other.

Attributes Reference

The following attributes are exported:

- `id` - ID of the Datadog downtime. On updates this can sometime change based on API logic. For recurring downtimes it would be recommended to `ignore_changes` on this field.
- `active` - If true this indicates the downtime is currently active.
- `disabled` - If true this indicates the downtime is currently disabled.

Import

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

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

Monitor Examples

This page lists examples of how to create different Datadog monitor types within Terraform. This list is non exhaustive and will be updated over time to provide more examples.

Watchdog Monitors

```
resource "datadog_monitor" "watchdog_monitor" {
  name               = "Watchdog Monitor TF"
  type               = "event alert"
  message            = "Check out this Watchdog Service Monitor"
  escalation_message = "Escalation message"

  query = "events('priority:all sources:watchdog tags:story_type:service,env:test_env,service:test_servic
e:_aggregate').by('service,resource_name').rollup('count').last('30m') > 0"

  notify_no_data    = false
  renotify_interval = 60

  notify_audit = false
  timeout_h    = 60
  include_tags = true

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

Anomaly Monitors

```
resource "datadog_monitor" "cpu_anomalous" {
  name = "Anomalous CPU usage"
  type = "query alert"
  message = "CPU utilization is outside normal bounds"
  query = "avg(last_4h):anomalies(ewma_20(avg:system.cpu.system{env:prod,service:website}.as_rate()), 'ro
bust', 3, direction='below', alert_window='last_30m', interval=60, count_default_zero='true', seasonality
='weekly') >= 1"
  thresholds {
    critical          = 1.0
    critical_recovery = 0.0
  }
  threshold_windows {
    trigger_window    = "last_30m"
    recovery_window   = "last_30m"
  }

  notify_no_data    = false
  renotify_interval = 60
}
```

Process Monitors

```
resource "datadog_monitor" "process_alert_example" {
  name = "Process Alert Monitor"
  type = "process alert"
  message = "Multiple Java processes running on example-tag"
  query = "processes('java').over('example-tag').rollup('count').last('10m') > 1",
  thresholds {
    critical          = 1.0
    critical_recovery = 0.0
  }

  notify_no_data    = false
  renotify_interval = 60
}
```

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` - (Optional) Array of tags (in the form key:value) to add to all hosts and metrics reporting through this integration.
- `account_specific_namespace_rules` - (Optional) 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 = "1234567890123456789012345678901234567890"
  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_integration_pagerduty

Provides a Datadog - PagerDuty resource. This can be used to create and manage Datadog - PagerDuty integration.

Example Usage

Note: Until terraform-provider-datadog version 2.1.0, service objects under the `services` key were specified inside the `datadog_integration_pagerduty` resource. This was incompatible with multi-configuration-file setups, where users wanted to have individual service objects controlled from different Terraform configuration files. The recommended approach now is specifying service objects as individual resources using datadog_integration_pagerduty_service_object (/docs/providers/datadog/r/integration_pagerduty_service_object.html) and adding `individual_services = true` to the `datadog_integration_pagerduty` object.

Services as Individual Resources

```
resource "datadog_integration_pagerduty" "pd" {
  individual_services = true
  schedules = [
    "https://ddog.pagerduty.com/schedules/X123VF",
    "https://ddog.pagerduty.com/schedules/X321XX"
    ]
  subdomain = "ddog"
  api_token = "38457822378273432587234242874"
}

resource "datadog_integration_pagerduty_service_object" "testing_foo" {
  # when creating the integration object for the first time, the service
  # objects have to be created *after* the integration
  depends_on = ["datadog_integration_pagerduty.pd"]
  service_name = "testing_foo"
  service_key  = "9876543210123456789"
}

resource "datadog_integration_pagerduty_service_object" "testing_bar" {
  depends_on = ["datadog_integration_pagerduty.pd"]
  service_name = "testing_bar"
  service_key  = "54321098765432109876"
}
```

Inline Services

With Terraform < 0.12.0 (terraform-provider-datadog < 1.9.0):

```
# Create a new Datadog - PagerDuty integration
resource "datadog_integration_pagerduty" "pd" {
  services = [
    {
      service_name = "testing_foo"
      service_key  = "9876543210123456789"
    },
    {
      service_name = "testing_bar"
      service_key = "54321098765432109876"
    }
  ]
  schedules = [
    "https://ddog.pagerduty.com/schedules/X123VF",
    "https://ddog.pagerduty.com/schedules/X321XX"
    ]
  subdomain = "ddog"
  api_token = "38457822378273432587234242874"
}
```

With Terraform >= 0.12.0 (terraform-provider-datadog >= 1.9.0):

```
locals {
  pd_services = {
    testing_foo = "9876543210123456789"
    testing_bar = "54321098765432109876"
  }
}
# Create a new Datadog - PagerDuty integration
resource "datadog_integration_pagerduty" "pd" {
  dynamic "services" {
    for_each = local.pd_services
    content {
      service_name = services.key
      service_key = services.value
    }
  }
  schedules = [
    "https://ddog.pagerduty.com/schedules/X123VF",
    "https://ddog.pagerduty.com/schedules/X321XX"
    ]
  subdomain = "ddog"
  api_token = "38457822378273432587234242874"
}
```

Migrating from Inline Services to Individual Resources

Migrating from usage of inline services to individual resources is very simple. The following example shows how to convert an existing inline services configuration to configuration using individual resources. Doing analogous change and running `terraform apply` after every step is all that's necessary to migrate.

```
# First step - this is what the configuration looked like initially

locals {
  pd_services = {
    testing_foo = "98765432101234567 89"
    testing_bar = "54321098765432109876"
  }
}
# Create a new Datadog - PagerDuty integration
resource "datadog_integration_pagerduty" "pd" {
  dynamic "services" {
    for_each = local.pd_services
    content {
      service_name = services.key
      service_key = services.value
    }
  }
  schedules = [
    "https://ddog.pagerduty.com/schedules/X123VF",
    "https://ddog.pagerduty.com/schedules/X321XX"
    ]
  subdomain = "ddog"
  api_token = "38457822378273432587234242874"
}
```

```
# Second step - this will remove the inline-defined service objects
# Note that during this step, `individual_services` must not be defined
resource "datadog_integration_pagerduty" "pd" {
  # `services` was removed
  schedules = [
    "https://ddog.pagerduty.com/schedules/X123VF",
    "https://ddog.pagerduty.com/schedules/X321XX"
    ]
  subdomain = "ddog"
  api_token = "38457822378273432587234242874"
}
```

```
# Third step - this will reintroduce the service objects as individual resources

resource "datadog_integration_pagerduty" "pd" {
  # `individual_services = true` was added
  individual_services = true
  schedules = [
    "https://ddog.pagerduty.com/schedules/X123VF",
    "https://ddog.pagerduty.com/schedules/X321XX"
    ]
  subdomain = "ddog"
  api_token = "384578223782734325872342428 74"
}

resource "datadog_integration_pagerduty_service_object" "testing_foo" {
  depends_on = ["datadog_integration_pagerduty.pd"]
  service_name = "testing_foo"
  service_key  = "9876543210123456789"
}

resource "datadog_integration_pagerduty_service_object" "testing_bar" {
  depends_on = ["datadog_integration_pagerduty.pd"]
  service_name = "testing_bar"
  service_key  = "54321098765432109876"
}
```

Argument Reference

The following arguments are supported:

- `individual_services` - (Optional) Boolean to specify whether or not individual service objects specified by datadog_integration_pagerduty_service_object (/docs/providers/datadog/r/integration_pagerduty_service_object.html) resource are to be used. Mutually exclusive with `services` key.
- `services` - (Optional) Array of PagerDuty service objects. **Deprecated** The `services` list is now deprecated in favour of datadog_integration_pagerduty_service_object (/docs/providers/datadog/r/integration_pagerduty_service_object.html) resource. Note that `individual_services` must be set to `true` to ignore the `service` attribute and use individual services properly.
  - `service_name` - (Required) Your Service name in PagerDuty.
  - `service_key` - (Required) Your Service name associated service key in Pagerduty.
- `schedules` - (Optional) Array of your schedule URLs.
- `subdomain` - (Required) Your PagerDuty account's personalized subdomain name.
- `api_token` - (Optional) Your PagerDuty API token.

See also

- PagerDuty Integration Guide (https://www.pagerduty.com/docs/guides/datadog-integration-guide/)
- Datadog API Reference > Integrations > PagerDuty (https://docs.datadoghq.com/api/?lang=bash#pagerduty)

datadog_integration_pagerduty_service_object

Provides access to individual Service Objects of Datadog - PagerDuty integrations. Note that the Datadog - PagerDuty integration must be activated (either manually in the Datadog UI or by using datadog_integration_pagerduty (/docs/providers/datadog/r/integration_pagerduty.html)) in order for this resource to be usable.

Example Usage

```
resource "datadog_integration_pagerduty" "pd" {
  individual_services = true
  schedules = [
    "https://ddog.pagerduty.com/schedules/X123VF",
    "https://ddog.pagerduty.com/schedules/X321XX"
    ]
  subdomain = "ddog"
  api_token = "38457822378273432587234242874"
}

resource "datadog_integration_pagerduty_service_object" "testing_foo" {
  # when creating the integration object for the first time, the service
  # objects have to be created *after* the integration
  depends_on = ["datadog_integration_pagerduty.pd"]
  service_name = "testing_foo"
  service_key  = "9876543210123456789"
}

resource "datadog_integration_pagerduty_service_object" "testing_bar" {
  depends_on = ["datadog_integration_pagerduty.pd"]
  service_name = "testing_bar"
  service_key  = "54321098765432109876"
}
```

Argument Reference

The following arguments are supported:

- `service_name` - (Required) Your Service name in PagerDuty.
- `service_key` - (Required) Your Service name associated service key in PagerDuty. Note: Since the Datadog API never returns service keys, it is impossible to detect drifts (https://www.hashicorp.com/blog/detecting-and-managing-drift-with-terraform). The best way to solve a drift is to manually mark the Service Object resource with `terraform taint` (https://www.terraform.io/docs/commands/taint.html) to have it destroyed and recreated.

datadog_logs_custom_pipeline

Provides a Datadog Logs Pipeline API (https://docs.datadoghq.com/api/?lang=python#logs-pipelines) resource, which is used to create and manage Datadog logs custom pipelines.

Example Usage

Create a Datadog logs pipeline:

```
resource "datadog_logs_custom_pipeline" "sample_pipeline" {
    filter {
        query = "source:foo"
    }
    name = "sample pipeline"
    is_enabled = true
    processor {
        arithmetic_processor {
            expression = "(time1 - time2)*1000"
            target = "my_arithmetic"
            is_replace_missing = true
            name = "sample arithmetic processor"
            is_enabled = true
        }
    }
    processor {
        attribute_remapper {
            sources = ["db.instance"]
            source_type = "tag"
            target = "db"
            target_type = "tag"
            preserve_source = true
            override_on_conflict = false
            name = "sample attribute processor"
            is_enabled = true
        }
    }
    processor {
        category_processor {
            target = "foo.severity"
            category {
                name = "debug"
                filter {
                    query = "@severity: \".\""
                }
            }
            category {
                name = "verbose"
                filter {
                    query = "@severity: \"-\""
                }
            }
            name = "sample category processor"
            is_enabled = true
        }
    }
```

```
}
processor {
    date_remapper {
        sources = ["_timestamp", "published_date"]
        name = "sample date remapper"
        is_enabled = true
    }
}
processor {
    message_remapper {
        sources = ["msg"]
        name = "sample message remapper"
        is_enabled = true
    }
}
processor {
    status_remapper {
        sources = ["info", "trace"]
        name = "sample status remapper"
        is_enabled = true
    }
}
processor {
    trace_id_remapper {
        sources = ["dd.trace_id"]
        name = "sample trace id remapper"
        is_enabled = true
    }
}
processor {
    service_remapper {
        sources = ["service"]
        name = "sample service remapper"
        is_enabled = true
    }
}
processor {
    grok_parser {
        source = "message"
        grok {
            support_rules = ""
            match_rules = "Rule %%{word:my_word2} %%{number:my_float2}"
        }
        name = "sample grok parser"
        is_enabled = true
    }
}
processor {
    pipeline {
        filter {
            query = "source:foo"
        }
        processor {
            url_parser {
                name = "sample url parser"
                sources = ["url", "extra"]
                target = "http_url"
                normalize_ending_slashes = true
            }
        }
```

```
            name = "nested pipeline"
            is_enabled = true
        }
    }
    processor {
        user_agent_parser {
            sources = ["user", "agent"]
            target = "http_agent"
            is_encoded = false
            name = "sample user agent parser"
            is_enabled = true
        }
    }
}
```

Argument Reference

The following arguments are supported:

- `filter` - (Required) Defines your pipeline filter. Only logs that match the filter criteria are processed by this pipeline.
  - `query` - (Required) Defines the filter criteria.
- `name` - (Required) Your pipeline name.
- `is_enabled` - (Optional, default = false) Boolean value to enable your pipeline.
- `processor` - (Optional) Processors or nested pipelines. See below (/docs/providers/datadog/r/logs_custom_pipeline.html#Processors) for more detailed descriptions.

Note A pipeline or its processors are disabled by default if `is_enabled` is not explicitly set to `true`.

Processors

- arithmetic_processor
  - `expression` - (Required) Arithmetic operation between one or more log attributes.
  - `target` - (Required) Name of the attribute that contains the result of the arithmetic operation.
  - `is_replace_missing` - (Optional, default = false) If true, it replaces all missing attributes of expression by 0, false skips the operation if an attribute is missing.
  - `name` - (Optional) Name of the processor
  - `is_enabled` - (Optional, default = false) If the processor is enabled or not.
- attribute_remapper
  - `sources` - (Required) List of source attributes or tags.
  - `source_type` - (Required) Defines where the sources are from (log `attribute` or `tag`).
  - `target` - (Required) Final `attribute` or `tag` name to remap the sources.
  - `target_type` - (Required) Defines if the target is a log `attribute` or `tag`.

    - `preserve_source` - (Optional, default = false) Remove or preserve the remapped source element.
    - `override_on_conflict` - (Optional, default = false) Override the target element if already set.
    - `name` - (Optional) Name of the processor
    - `is_enabled` - (Optional, default = false) If the processor is enabled or not.
- category_processor
    - `target` - (Required) Name of the target attribute whose value is defined by the matching category.
    - `category` - (Required) List of filters to match or exclude a log with their corresponding name to assign a custom value to the log.
    - `name` - (Required) Name of the cateory.
    - `filter`
        - `query` - (Required) Filter criteria of the category.
    - `name` - (Optional) Name of the processor
    - `is_enabled` - (Optional, default = false) If the processor is enabled or not.
- date_remapper
    - `sources` - (Required) List of source attributes.
    - `name` - (Optional) Name of the processor.
    - `is_enabled` - (Optional, default = false) If the processor is enabled or not.
- grok_parser
    - `source` - (Required) Name of the log attribute to parse.
    - `grok`
    - `support_rules` - (Required) Support rules for your grok parser.
    - `match_rules` - (Required) Match rules for your grok parser.
    - `name` - (Optional) Name of the processor.
    - `is_enabled` - (Optional, default = false) If the processor is enabled or not.
- message_remapper
    - `sources` - (Required) List of source attributes.
    - `name` - (Optional) Name of the processor.
    - `is_enabled` - (Optional, default = false) If the processor is enabled or not.
- pipeline
    - `filter` - (Required) Defines the nested pipeline filter. Only logs that match the filter criteria are processed by this pipeline.
    - `query` - (Required)

    - `processor` - (Optional) Processors (/docs/providers/datadog/r/logs_custom_pipeline.html#Processors). Nested pipeline can't take any other nested pipeline as its processor.
    - `name` - (Optional) Name of the nested pipeline.
    - `is_enabled` - (Optional, default = false) If the processor is enabled or not.
- service_remapper
    - `sources` - (Required) List of source attributes.
    - `name` - (Optional) Name of the processor
    - `is_enabled` - (Optional, default = false) If the processor is enabled or not.
- status_remapper
    - `sources` - (Required) List of source attributes.
    - `name` - (Optional) Name of the processor
    - `is_enabled` - (Optional, default = false) If the processor is enabled or not.
- trace_id_remapper
    - `sources` - (Required) List of source attributes.
    - `name` - (Optional) Name of the processor
    - `is_enabled` - (Optional, default = false) If the processor is enabled or not.
- url_parser
    - `sources` - (Required) List of source attributes.
    - `target` - (Required) Name of the parent attribute that contains all the extracted details from the `sources`.
    - `normalize_ending_slashes` - (Optional, default = false) Normalize the ending slashes or not.
    - `name` - (Optional) Name of the processor
    - `is_enabled` - (Optional, default = false) If the processor is enabled or not.
- user_agent_parser
    - `sources` - (Required) List of source attributes.
    - `target` - (Required) Name of the parent attribute that contains all the extracted details from the sources.
    - `is_encoded` - (Optional, default = false) If the source attribute is URL encoded or not.
    - `name` - (Optional) Name of the processor
    - `is_enabled` - (Optional, default = false) If the processor is enabled or not.

Even though some arguments are optional, we still recommend you to state **all** arguments in the resource to avoid unnecessary confusion.

Import

For the previously created custom pipelines, you can include them in Terraform with the `import` operation. Currently, Terraform requires you to explicitly create resources that match the existing pipelines to import them. Use `terraform import <resource.name> <pipelineID>` for each existing pipeline.

Important Notes

Each `datadog_logs_custom_pipeline` resource defines a complete pipeline. The order of the pipelines is maintained in a different resource datadog_logs_pipeline_order (/docs/providers/datadog/r/logs_pipeline_order.html#datadog_logs_pipeline_order). When creating a new pipeline, you need to **explicitly** add this pipeline to the `datadog_logs_pipeline_order` resource to track the pipeline. Similarly, when a pipeline needs to be destroyed, remove its references from the `datadog_logs_pipeline_order` resource.

datadog_logs_index

Provides a Datadog Logs Index API (https://docs.datadoghq.com/api/?lang=python#logs-indexes) resource. This can be used to create and manage Datadog logs indexes.

Example Usage

A sample Datadog logs index resource definition. Note that at this point, it is not possible to create new logs indexes through Terraform, so the `name` field must match a name of an already existing index. If you want to keep the current state of the index, we suggest importing it (see below).

```
resource "datadog_logs_index" "sample_index" {
    name = "your index"
    filter {
        query = "*"
    }
    exclusion_filter {
        name = "Filter coredns logs"
        is_enabled = true
        filter {
            query = "app:coredns"
            sample_rate = 0.97
        }
    }
    exclusion_filter {
        name = "Kubernetes apiserver"
        is_enabled = true
        filter {
            query = "service:kube_apiserver"
            sample_rate = 1.0
        }
    }
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the index.
- `filter` - (Required) Logs filter.
  - `query` - (Required) Logs filter criteria. Only logs matching this filter criteria are considered for this index.
- `exclusion_filter` - (Optional) List of exclusion filters.
  - `filter` - (Optional)
    - `query` - (Optional) Only logs matching the filter criteria and the query of the parent index will be considered for this exclusion filter.
    - `sample_rate` - (Optional, default = 0.0) The fraction of logs excluded by the exclusion filter, when active.

- `name` - (Optional) The name of the exclusion filter.
- `is_enabled` - (Optional, default = false) A boolean stating if the exclusion is active or not.

Import

The current Datadog Terraform provider version does not support the creation and deletion of indexes. To manage the existing indexes, do `terraform import <datadog_logs_index.name> <indexName>` to import them to Terraform. If you create a resource which does not match the name of any existing index, `terraform apply` will throw `Not Found` error code.

Important Notes

The order of indexes is maintained in the separated resource datadog_logs_index_order (/docs/providers/datadog/r/logs_index_order.html#datadog_logs_index_order).

datadog_logs_index_order

Provides a Datadog Logs Index API (https://docs.datadoghq.com/api/?lang=python#logs-indexes) resource. This can be used to manage the order of Datadog logs indexes.

Example Usage

```
resource "datadog_logs_index_order" "sample_index_order" {
    name = "sample_index_order"
    depends_on = [
        "datadog_logs_index.sample_index"
    ]
    indexes = [
        "${datadog_logs_index.sample_index.id}"
    ]
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The unique name of the index order resource.
- `indexes` - (Required) The index resource list. Logs are tested against the query filter of each index one by one following the order of the list.

Import

The current Datadog Terraform provider version does not support the creation and deletion of index orders. Do `terraform import <datadog_logs_index_order.name> <name>` to import index order to Terraform. There must be at most one `datadog_logs_index_order` resource.

datadog_logs_integration_pipeline

Provides a Datadog Logs Pipeline API (https://docs.datadoghq.com/api/?lang=python#logs-pipelines) resource to manage the integrations (https://docs.datadoghq.com/logs/log_collection/?tab=tcpussite).

Integration pipelines are the pipelines that are automatically installed for your organization when sending the logs with specific sources. You don't need to maintain or update these types of pipelines. Keeping them as resources, however, allows you to manage the order of your pipelines by referencing them in your datadog_logs_pipeline_order (/docs/providers/datadog/r/logs_pipeline_order.html#datadog_logs_pipeline_order) resource. If you don't need the `pipeline_order` feature, this resource declaration can be omitted.

Example Usage

```
resource "datadog_logs_integration_pipeline" "python" {
    is_enabled = true
}
```

Argument Reference

- `is_enabled` - (Required) Boolean value to enable your pipeline.

`is_enabled` is the only value that can be modified for integration pipeline.

Import

```
terraform import <resource.name> <pipelineID>
```

datadog_logs_pipeline_order

Provides a Datadog Logs Pipeline API (https://docs.datadoghq.com/api/?lang=python#logs-pipelines) resource, which is used to manage Datadog log pipelines order.

Example Usage

```
resource "datadog_logs_pipeline_order" "sample_pipeline_order" {
    name = "sample_pipeline_order"
    depends_on = [
        "datadog_logs_custom_pipeline.sample_pipeline",
        "datadog_logs_integration_pipeline.python"
    ]
    pipelines = [
        "${datadog_logs_custom_pipeline.sample_pipeline.id}",
        "${datadog_logs_integration_pipeline.python.id}"
    ]
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name attribute in the resource `datadog_logs_pipeline_order` needs to be unique. It's recommended to use the same value as the resource `NAME`. No related field is available in Logs Pipeline API (https://docs.datadoghq.com/api/?lang=python#get-pipeline-order).
- `pipelines` - (Required) The pipeline IDs list. The order of pipeline IDs in this attribute defines the overall pipeline order for logs.

Attributes Reference

- `pipelines` - The `pipelines` list contains the IDs of resources created and imported by the datadog_logs_custom_pipeline (/docs/providers/datadog/r/logs_custom_pipeline.html#datadog_logs_custom_pipeline) and datadog_logs_integration_pipeline (/docs/providers/datadog/r/logs_integration_pipeline.html#datadog_logs_integration_pipeline). Updating the order of pipelines in this list reflects the application order of the pipelines. You cannot delete or create pipeline by deleting or adding IDs to this list.

Import

There must be at most one `datadog_logs_pipeline_order` resource. Pipeline order creation is not supported from logs config API. You can import the `datadog_logs_pipeline_order` or create a pipeline order (which is actually doing the update operation).

```
terraform import <datadog_logs_pipeline_order.name> <name>
```

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. The mapping from these types to the types found in the Datadog Web UI can be found in the Datadog API documentation (https://docs.datadoghq.com/api/?lang=python#create-a-monitor) page. Available options to choose from are:
    - `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. Defaults to 10 minutes.
- `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 automatically insert its triggering tags into the title. Defaults to true.
- `enable_logs_sample` (Optional) A boolean indicating whether or not to include a list of log values which triggered the alert. Defaults to false. This is only used by log monitors. 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
- `threshold_windows` (Optional) A mapping containing `recovery_window` and `trigger_window` values, e.g. `last_15m`. Can only be used for, and are required for, anomaly monitors.
    - `recovery_window` describes how long an anomalous metric must be normal before the alert recovers.
    - `trigger_window` describes how long a metric must be anomalous before an alert triggers.
- `silenced` (Optional) Each scope will be muted until the given POSIX timestamp or forever if the value is 0. Use `-1` if you want to unmute the scope. **Deprecated** The `silenced` parameter is being deprecated in favor of the downtime resource. This will be removed in the next major version of the Terraform Provider.

  To mute the alert completely:

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

  To mute role:db for a short time:

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

  Note: due to HCL limitations (https://github.com/hashicorp/terraform/issues/2042), it is impossible to use interpolations in keys. For example, the following will result in muting the scope `role:${var:role}` (no interpolation is done):

  ```
  silenced = {
      "role:${var:role}" = 0
  }
  ```

  To workaround this, you can use the map function (https://www.terraform.io/docs/configuration/functions/map.html) of HCL:

```
silenced = ${map("role:${var:role}", 0)}
```

Silencing by Hand and by Downtimes

There are two ways how to silence a single monitor:

- Mute it by hand
- Create a Downtime

Both of these actions add a new value to the `silenced` map. This can be problematic if the `silenced` attribute doesn't contain them in your Terraform, as they would be removed on next `terraform apply` invocation. In order to prevent that from happening, you can add following to your monitor:

```
lifecycle {
  ignore_changes = ["silenced"]
}
```

The above will make sure that any changes to the `silenced` attribute are ignored.

This issue doesn't apply to multi-monitor downtimes (those that don't contain `monitor_id`), as these don't influence contents of the `silenced` attribute.

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
```

Composite Monitors

You can compose monitors of all types in order to define more specific alert conditions (see the doc (https://docs.datadoghq.com/monitors/monitor_types/composite/)). You just need to reuse the ID of your `datadog_monitor` resources. You can also compose any monitor with a `datadog_synthetics_test` by passing the computed `monitor_id` attribute in the query.

```
resource "datadog_monitor" "bar" {
  name = "Composite Monitor"
  type = "composite"
  message = "This is a message"

    query = "${datadog_monitor.foo.id} || ${datadog_synthetics_test.foo.monitor_id}"
}
```

datadog_screenboard

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

Note: This resource is outdated. Use the new `datadog_dashboard` (/docs/providers/datadog/r/dashboard.html) resource instead.

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"
    }

    # NOTE: this will only work with TF >= 0.12; see metadata_json
    # documentation below for example on usage with TF < 0.12
    metadata_json = jsonencode({
      "avg:system.cpu.user{*}": {
        "alias": "CPU Usage"
      }
    })
  }

  request {
    log_query {
      index = "mcnulty"
      compute {
        aggregation = "avg"
        facet = "@duration"
        interval = 5000
      }
      search {
        query = "status:info"
      }
      group_by {
        facet = "host"
        limit = 10
        sort {
          aggregation = "avg"
          order = "desc"
          facet = "@duration"
        }
      }
    }
    type = "area"
  }

  request {
    apm_query {
      index = "apm-search"
      compute {
        aggregation = "avg"
        facet = "@duration"
        interval = 5000
      }
      search {
        query = "type:web"
      }
      group_by {
        facet = "resource_name"
        limit = 50
        sort {
          aggregation = "avg"
          order = "desc"
```

```
            order    desc
            facet = "@string_query.interval"
          }
        }
      }
      type = "bars"
    }

    request {
      process_query {
        metric = "process.stat.cpu.total_pct"
        search_by = "error"
        filter_by = ["active"]
        limit = 50
      }
      type = "area"
    }

    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"
```

```
      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_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_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_hosts` - (Optional, only for widgets of type "hostmap") Boolean indicating whether to show ungrouped nodes.
- `no_metric_hosts` - (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 (exactly only one of `q`, `apm_query`, `log_query` or `process_query` is required within the request block):

- `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.
- `apm_query` - (Optional) The APM query to use in the widget. The structure of this block is described below (/docs/providers/datadog/r/screenboard.html#nested-widget-tile_def-request-apm_query-and-log_query-blocks).
- `log_query` - (Optional) The log query to use in the widget. The structure of this block is described below (/docs/providers/datadog/r/screenboard.html#nested-widget-tile_def-request-apm_query-and-log_query-blocks).
- `process_query` - (Optional) The process query to use in the widget. The structure of this block is described below (/docs/providers/datadog/r/screenboard.html#nested-widget-tile_def-request-process_query-blocks).

- `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.
- `metadata_json` - (Optional) A JSON blob (preferrably created using jsonencode (https://www.terraform.io/docs/configuration/functions/jsonencode.html)) representing mapping of query expressions to alias names. Note that the query expressions in `metadata_json` will be ignored if they're not present in the query. For example, this is how you define `metadata_json` with Terraform >= 0.12: `metadata_json = jsonencode({ "avg:redis.info.latency_ms{$host}": { "alias": "Redis latency" } })` And here's how you define `metadata_json` with Terraform < 0.12: ``` variable "my_metadata" { default = { "avg:redis.info.latency_ms{$host}" = { "alias": "Redis latency" } } }

resource "datadog_screenboard" "SomeScreenboard" { ... metadata_json = "${jsonencode(var.my_metadata)}" } ``` `Note that this has to be a JSON blob because of [limitations] (https://github.com/hashicorp/terraform/issues/6215) of Terraform's handling complex nested structures. This is also why the key is called` metadata_json `even though it sets` metadata` attribute on the API call.

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 apm_query` and `log_query` blocks

Nested `apm_query` and `log_query` blocks have the following structure (Visit the Graph Primer (https://docs.datadoghq.com/graphing/) for more information about these values):

- `index` - (Required)
- `compute` - (Required). Exactly one nested block is required with the following structure:
  - `aggregation` - (Required)
  - `facet` - (Optional)
  - `interval` - (Optional)
- `search` - (Optional). One nested block is allowed with the following structure:
  - `query` - (Optional)
- `group_by` - (Optional). Multiple nested blocks are allowed with the following structure:
  - `facet` - (Optional)
  - `limit` - (Optional)
  - `sort` - (Optional). One nested block is allowed with the following structure:
  - `aggregation` - (Optional)
  - `order` - (Optional)
  - `facet` - (Optional)

Nested `widget tile_def request process_query` blocks

Nested `process_query` blocks have the following structure (Visit the Graph Primer (https://docs.datadoghq.com/graphing/) for more information about these values):

- `metric` - (Required)
- `search_by` - (Required)
- `filter_by` - (Required)

- `limit` - (Required)

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_service_level_objective

Provides a Datadog service level objective resource. This can be used to create and manage Datadog service level objectives.

Example Usage

Metric-Based SLO

```
# Create a new Datadog service level objective
resource "datadog_service_level_objective" "foo" {
  name               = "Example Metric SLO"
  type               = "metric"
  description        = "My custom metric SLO"
  query {
    numerator = "sum:my.custom.count.metric{type:good_events}.as_count()"
    denominator = "sum:my.custom.count.metric{*}.as_count()"
  }

  thresholds {
    timeframe = "7d"
    target = 99.9
    warning = 99.99
    target_display = "99.900"
    warning_display = "99.990"
  }

  thresholds {
    timeframe = "30d"
    target = 99.9
    warning = 99.99
    target_display = "99.900"
    warning_display = "99.990"
  }

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

Monitor-Based SLO

```
# Create a new Datadog service level objective
resource "datadog_service_level_objective" "bar" {
  name               = "Example Monitor SLO"
  type               = "monitor"
  description        = "My custom monitor SLO"
  monitor_ids = [1, 2, 3]

  thresholds {
    timeframe = "7d"
    target = 99.9
    warning = 99.99
  }

  thresholds {
    timeframe = "30d"
    target = 99.9
    warning = 99.99
  }

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

Argument Reference

The following arguments are supported:

- `type` - (Required) The type of the service level objective. The mapping from these types to the types found in the Datadog Web UI can be found in the Datadog API documentation (https://docs.datadoghq.com/api/?lang=python#create-a-service-level-objective) page. Available options to choose from are:
  - `metric`
  - `monitor`
- `name` - (Required) Name of Datadog service level objective
- `description` - (Optional) A description of this service level objective.
- `tags` (Optional) A list of tags to associate with your service level objective. This can help you categorize and filter service level objectives in the service level objectives page of the UI. Note: it's not currently possible to filter by these tags when querying via the API
- `thresholds` - (Required) - A list of thresholds and targets that define the service level objectives from the provided SLIs.
  - `timeframe` (Required) - the time frame for the objective. The mapping from these types to the types found in the Datadog Web UI can be found in the Datadog API documentation (https://docs.datadoghq.com/api/?lang=python#create-a-service-level-objective) page. Available options to choose from are:
    - `7d`
    - `30d`
    - `90d`
  - `target` - (Required) the objective's target `[0,100]`

- `target_display` - (Optional) the string version to specify additional digits in the case of `99` but want 3 digits like `99.000` to display.
- `warning` - (Optional) the objective's warning value `[0,100]`. This must be `> target` value.
- `warning_display` - (Optional) the string version to specify additional digits in the case of `99` but want 3 digits like `99.000` to display.

The following options are specific to the `type` of service level objective:

- `metric` type SLOs:
  - `query` - (Required) The metric query configuration to use for the SLI. This is a dictionary and requires both the `numerator` and `denominator` fields which should be `count` metrics using the `sum` aggregator.
    - `numerator` - (Required) the sum of all the `good` events
    - `denominator` - (Required) the sum of the `total` events
    - Example Usage:

```
query {
    numerator   = "sum:my.custom.count.metric{type:good}.as_count()"
    denominator = "sum:my.custom.count.metric{*}.as_count()"
}
```

- `monitor` type SLOs:
  - `monitor_ids` - (Optional) A list of numeric monitor IDs for which to use as SLIs. Their tags will be auto-imported into `monitor_tags` field in the API resource. At least 1 of `monitor_ids` or `monitor_search` must be provided.
  - `monitor_search` - (Optional) The monitor query search used on the monitor search API to add monitor_ids by searching. Their tags will be auto-imported into `monitor_tags` field in the API resource. At least 1 of `monitor_ids` or `monitor_search` must be provided.
  - `groups` - (Optional) A custom set of groups from the monitor(s) for which to use as the SLI instead of all the groups.

Attributes Reference

The following attributes are exported:

- `id` - ID of the Datadog service level objective

Import

Service Level Objectives can be imported using their string ID, e.g.

```
$ terraform import datadog_service_level_objective.12345678901234567890123456789012 "baz"
```

datadog_synthetics_test

Provides a Datadog synthetics test resource. This can be used to create and manage Datadog synthetics test.

Example Usage (Synthetics API test)

Create a new Datadog Synthetics API/HTTP test on https://www.example.org (https://www.example.org)

```
resource "datadog_synthetics_test" "test_api" {
  type = "api"
  subtype = "http"
  request = {
    method = "GET"
    url = "https://www.example.org"
  }
  request_headers = {
    Content-Type = "application/json"
    Authentication = "Token: 1234566789"
  }
  assertions = [
    {
      type = "statusCode"
      operator = "is"
      target = "200"
    }
  ]
  locations = [ "aws:eu-central-1" ]
  options = {
    tick_every = 900
  }
  name = "An API test on example.org"
  message = "Notify @pagerduty"
  tags = ["foo:bar", "foo", "env:test"]

  status = "live"
}
```

Example Usage (Synthetics SSL test)

Create a new Datadog Synthetics API/SSL test on example.org

```
resource "datadog_synthetics_test" "test_ssl" {
  type = "api"
  subtype = "ssl"
  request = {
    host = "example.org"
    port = 443
  }
  assertions = [
    {
      type = "certificate"
      operator = "isInMoreThan"
      target = 30
    }
  ]
  locations = [ "aws:eu-central-1" ]
  options = {
    tick_every = 900
    accept_self_signed = true
  }
  name = "An API test on example.org"
  message = "Notify @pagerduty"
  tags = ["foo:bar", "foo", "env:test"]

  status = "live"
}
```

Example Usage (Synthetics Browser test)

Support for Synthetics Browser test is limited (see below)

```
# Create a new Datadog Synthetics Browser test starting on https://www.example.org
resource "datadog_synthetics_test" "test_browser" {
  type = "browser"

  request = {
    method = "GET"
    url    = "https://app.datadoghq.com"
  }

  device_ids = ["laptop_large"]
  locations  = ["aws:eu-central-1"]

  options = {
    tick_every = 3600
  }

  name = "A Browser test on example.org"
  message = "Notify @qa"
  tags    = []

  status = "paused"
}
```

Argument Reference

The following arguments are supported:

- `type` - (Required) Synthetics test type (api or browser)
- `subtype` - (Optional) For type=api, http or ssl (Default = http)
- `name` - (Required) Name of Datadog synthetics test
- `message` - (Required) A message to include with notifications for this synthetics test. Email notifications can be sent to specific users by using the same '@username' notation as events.
- `tags` - (Required) A list of tags to associate with your synthetics test. This can help you categorize and filter tests in the manage synthetics page of the UI.
- `request` - (Required) if type=api and subtype=http
  - `method` - (Optional) For type=api and subtype=http, one of DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT
  - `url` - (Required) Any url
  - `timeout` - (Optional) For type=api, any value between 0 and 60 (Default = 60)
  - `body` - (Optional) Request body
- `request` - (Required) if type=api and subtype=ssl
  - `host` - (Required) host name
  - `port` - (Required) port number
  - `timeout` - (Optional) For type=api, any value between 0 and 60 (Default = 60)
- `request` - (Required) if type=browser
  - `method` - (Required) no-op, use GET
  - `url` - (Required) Any url
- `request_headers` - (Optional) Header name and value map
- `assertions` - (Required) Array of 1 to 10 items, only some combinations of type/operator are valid (please refer to Datadog documentation)
  - `type` - (Required) body, header, responseTime, statusCode
  - `operator` - (Required) Please refer to Datadog documentation (https://docs.datadoghq.com/synthetics/api_test/#validation) as operator depend on assertion type
  - `target` - (Required) Expected value, please refer to Datadog documentation (https://docs.datadoghq.com/synthetics/api_test/#validation) as target depend on assertion type
  - `property` - (Optional) if assertion type is "header", this is a the header name
- `options` - (Required)
  - `tick_every` - (Required) How often the test should run (in seconds). Current possible values are 900, 1800, 3600, 21600, 43200, 86400, 604800 plus 60 if type=api or 300 if type=browser
  - `follow_redirects` - (Optional) For type=api, true or false
  - `min_failure_duration` - (Optional) How long the test should be in failure before alerting (integer, number of

seconds, max 7200). Default is 0.

    - `min_location_failed` - (Optional) Threshold below which a synthetics test is allowed to fail before sending notifications

    - `accept_self_signed` - (Optional) For type=ssl, true or false

- `locations` - (Required) Please refer to Datadog documentation (https://docs.datadoghq.com/synthetics/api_test/#request) for available locations (e.g. "aws:eu-central-1")

- `device_ids` - (Optional) "laptop_large", "tablet" or "mobile_small" (only available if type=browser)

- `status` - (Required) "live", "paused"

Attributes Reference

The following attributes are exported:

- `id` - ID (public_id) of the Datadog synthetics test

- `monitor_id` - ID of the monitor associated with the Datadog synthetics test

Import

Synthetics tests can be imported using their public string ID, e.g.

```
$ terraform import datadog_synthetics_test.fizz abc-123-xyz
```

Synthetics Browser test

Support for Synthetics Browser test is limited to creating shallow Synthetics Browser test (cf. example usage below)

You cannot create/edit/delete steps or assertions via Terraform unless you use Datadog WebUI (https://app.datadoghq.com/synthetics/list) in conjunction with Terraform.

We are considering adding support for Synthetics Browser test steps and assertions in the future but can't share any release date on that matter.

datadog_timeboard

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

Note: This resource is outdated. Use the new `datadog_dashboard` (/docs/providers/datadog/r/dashboard.html) resource instead.

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"

      # NOTE: this will only work with TF >= 0.12; see metadata_json
      # documentation below for example on usage with TF < 0.12
      metadata_json = jsonencode({
        "avg:redis.info.latency_ms{$host}": {
          "alias": "Redis latency"
        }
      })
    }

    request {
        log_query {
          index = "mcnulty"
          compute {
            aggregation = "avg"
            facet = "@duration"
            interval = 5000
          }
          search {
            query = "status:info"
          }
          group_by {
            facet = "host"
            limit = 10
            sort {
              aggregation = "avg"
              order = "desc"
              facet = "@duration"
            }
          }
        }
```

```
      type = "area"
    }

    request {
      apm_query {
        index = "apm-search"
        compute {
          aggregation = "avg"
          facet = "@duration"
          interval = 5000
        }
        search {
          query = "type:web"
        }
        group_by {
          facet = "resource_name"
          limit = 50
          sort {
            aggregation = "avg"
            order = "desc"
            facet = "@string_query.interval"
          }
        }
      }
      type = "bars"
    }

    request {
      process_query {
        metric = "process.stat.cpu.total_pct"
        search_by = "error"
        filter_by = ["active"]
        limit = 50
      }
      type = "area"
    }
}

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 (exactly only one of `q`, `apm_query`, `log_query` or `process_query` is required within the request block):

- `q` - (Optional) The query of the request. Pro tip: Use the JSON tab inside the Datadog UI to help build you query strings.
- `apm_query` - (Optional) The APM query to use in the widget. The structure of this block is described below (/docs/providers/datadog/r/timeboard.html#nested-graph-request-apm_query-and-log_query-blocks).
- `log_query` - (Optional) The log query to use in the widget. The structure of this block is described below (/docs/providers/datadog/r/timeboard.html#nested-graph-request-apm_query-and-log_query-blocks).
- `process_query` - (Optional) The process query to use in the widget. The structure of this block is described below (/docs/providers/datadog/r/timeboard.html#nested-graph-request-process_query-blocks).

- `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.
- `extra_col` - (Optional, only for graphs of visualization "change") If set to "present", displays current value. Can be left empty otherwise.
- `metadata_json` - (Optional) A JSON blob (preferrably created using jsonencode (https://www.terraform.io/docs/configuration/functions/jsonencode.html)) representing mapping of query expressions to alias names. Note that the query expressions in `metadata_json` will be ignored if they're not present in the query. For example, this is how you define `metadata_json` with Terraform >= 0.12:

```
metadata_json = jsonencode({
  "avg:redis.info.latency_ms{$host}": {
    "alias": "Redis latency"
  }
})
```

And here's how you define `metadata_json` with Terraform < 0.12:

```
variable "my_metadata" {
  default = {
    "avg:redis.info.latency_ms{$host}" = {
      "alias": "Redis latency"
    }
  }
}

resource "datadog_timeboard" "SomeTimeboard" {
  ...
      metadata_json = "${jsonencode(var.my_metadata)}"
}
```

Note that this has to be a JSON blob because of limitations (https://github.com/hashicorp/terraform/issues/6215) of Terraform's handling complex nested structures. This is also why the key is called `metadata_json` even though it sets `metadata` attribute on the API call.

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 apm_query` and `log_query` blocks

Nested `apm_query` and `log_query` blocks have the following structure (Visit the Graph Primer (https://docs.datadoghq.com/graphing/) for more information about these values):

- `index` - (Required)
- `compute` - (Required). Exactly one nested block is required with the following structure:
  - `aggregation` - (Required)
  - `facet` - (Optional)
  - `interval` - (Optional)
- `search` - (Optional). One nested block is allowed with the following structure:
  - `query` - (Optional)
- `group_by` - (Optional). Multiple nested blocks are allowed with the following structure:
  - `facet` - (Optional)
  - `limit` - (Optional)
  - `sort` - (Optional). One nested block is allowed with the following structure:
  - `aggregation` - (Optional)
  - `order` - (Optional)
  - `facet` - (Optional)

Nested `graph request process_query` blocks

Nested `process_query` blocks have the following structure (Visit the Graph Primer (https://docs.datadoghq.com/graphing/) for more information about these values):

- `metric` - (Required)
- `search_by` - (Required)
- `filter_by` - (Required)
- `limit` - (Required)

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
```

Dynamic Timeboards

Since Terraform 0.12, it's possible to create timeboard graphs dynamically based on contents of a list/map variable. This can be achieved by using the dynamic blocks (https://www.terraform.io/docs/configuration/expressions.html#dynamic-blocks) feature. For example:

```
variable "my_list" {
  default = ["First", "Second", "Third"]
}

variable "my_map" {
  default = {
    "First" = "value1"
    "Second" = "value2"
  }
}

# Create a timeboard with "First", "Second" and "Third" timeseries graphs
resource "datadog_timeboard" "my_timeboard" {
  title       = "My Timeboard"
  description = "My Description"
  read_only   = true

  dynamic "graph" {
    for_each = var.my_list
    content {
      title = "${graph.value}"
      viz = "timeseries"
      request {
        q = "anomalies(sum:mycount{adapter:${graph.value}}.as_count().rollup(sum, 3600), 'robust', 4, dir
ection='below')"
      }
    }
  }
}

# Create a timeboard with "First" and "Second" timeseries graphs, use map keys as titles and map values a
s adapter names
resource "datadog_timeboard" "my_timeboard_map" {
  title       = "My Timeboard From Map"
  description = "My Description"
  read_only   = true

  dynamic "graph" {
    for_each = var.my_map
    content {
      title = "${graph.key}"
      viz = "timeseries"
      request {
        q = "anomalies(sum:mycount{adapter:${graph.value}}.as_count().rollup(sum, 3600), 'robust', 4, dir
ection='below')"
      }
    }
  }
}
```

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:

- `access_role` - (Optional) Role description for user. Can be `st` (standard user), `adm` (admin user) or `ro` (read-only user). Default is `st`.
- `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. **Warning**: the corresponding query parameter is ignored by the Datadog API, thus the argument would always trigger an execution plan.
- `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
```