» Data Source: newrelic_alert_channel

Use this data source to get information about a specific alert channel in New Relic that already exists. More information on Terraform's data sources can be found here.

» Example Usage

```
# Data source
data "newrelic_alert_channel" "foo" {
  name = "foo@example.com"
}

# Resource
resource "newrelic_alert_policy" "foo" {
  name = "foo"
}

# Using the data source and resource together
resource "newrelic_alert_policy_channel" "foo" {
  policy_id  = newrelic_alert_policy.foo.id
  channel_id = data.newrelic_alert_channel.foo.id
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the alert channel in New Relic.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The ID of the alert channel.
- `type` - Alert channel type, either: `email`, `opsgenie`, `pagerduty`, `slack`, `victorops`, or `webhook`.
- `config` - Alert channel configuration.
- `policy_ids` - A list of policy IDs associated with the alert channel.

» Data Source: newrelic_alert_policy

Use this data source to get information about a specific alert policy in New Relic that already exists. More information on Terraform's data sources can be found here.

» Example Usage

```
data "newrelic_alert_channel" "foo" {
  name = "foo@example.com"
}

data "newrelic_alert_policy" "foo" {
  name = "foo policy"
}

resource "newrelic_alert_policy_channel" "foo" {
  policy_id  = data.newrelic_alert_policy.foo.id
  channel_id = data.newrelic_alert_channel.foo.id
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the alert policy in New Relic.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The ID of the alert policy.
- `incident_preference` - The rollup strategy for the policy. Options include: PER_POLICY, PER_CONDITION, or PER_CONDITION_AND_TARGET. The default is PER_POLICY.
- `created_at` - The time the policy was created.
- `updated_at` - The time the policy was last updated.

» Data Source: newrelic_application

Use this data source to get information about a specific application in New Relic that already exists. More information on Terraform's data sources can be found

here.

» Example Usage

```
data "newrelic_application" "app" {
  name = "my-app"
}

resource "newrelic_alert_policy" "foo" {
  name = "foo"
}

resource "newrelic_alert_condition" "foo" {
  policy_id = newrelic_alert_policy.foo.id

  name        = "foo"
  type        = "apm_app_metric"
  entities    = [data.newrelic_application.app.id]
  metric      = "apdex"
  runbook_url = "https://www.example.com"

  term {
    duration      = 5
    operator      = "below"
    priority      = "critical"
    threshold     = "0.75"
    time_function = "all"
  }
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the application in New Relic.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The ID of the application.
- `instance_ids` - A list of instance IDs associated with the application.
- `host_ids` - A list of host IDs associated with the application.

» Data Source: newrelic_key_transaction

Use this data source to get information about a specific key transaction in New Relic that already exists. More information on Terraform's data sources can be found here.

» Example Usage

```
data "newrelic_key_transaction" "txn" {
  name = "txn"
}

resource "newrelic_alert_policy" "foo" {
  name = "foo"
}

resource "newrelic_alert_condition" "foo" {
  policy_id = newrelic_alert_policy.foo.id

  name        = "foo"
  type        = "apm_kt_metric"
  entities    = [data.newrelic_key_transaction.txn.id]
  metric      = "error_percentage"
  runbook_url = "https://www.example.com"

  term {
    duration      = 5
    operator      = "below"
    priority      = "critical"
    threshold     = "0.75"
    time_function = "all"
  }
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the application in New Relic.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The ID of the application.

» Data Source: newrelic_plugin

Use this data source to get information about a specific installed plugin in New Relic. More information on Terraform's data sources can be found here.

Each plugin published to New Relic's Plugin Central is assigned a GUID. Once you have installed a plugin into your account it is assigned an ID. This account-specific ID is required when creating Plugins alert conditions.

» Example Usage

```
data "newrelic_plugin" "foo" {
  guid = "com.example.my-plugin"
}

resource "newrelic_alert_policy" "foo" {
  name = "foo"
}

resource "newrelic_plugins_alert_condition" "foo" {
  policy_id          = newrelic_alert_policy.foo.id
  name               = "foo"
  metric             = "Component/Summary/Consumers[consumers]"
  plugin_id          = data.newrelic_plugin.foo.id
  plugin_guid        = data.newrelic_plugin.foo.guid
  value_function     = "average"
  metric_description = "Queue consumers"

  term {
    duration      = 5
    operator      = "below"
    priority      = "critical"
    threshold     = "0.75"
    time_function = "all"
  }
}
```

» Argument Reference

The following arguments are supported:

- `guid` - (Required) The GUID of the plugin in New Relic.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The ID of the installed plugin instance.

» Data Source: newrelic_plugin_component

Use this data source to get information about a single plugin component in New Relic that already exists. More information on Terraform's data sources can be found here.

Each plugin component reporting into to New Relic is assigned a unique ID. Once you have a plugin component reporting data into your account, its component ID can be used to create Plugins alert conditions.

» Example Usage

```
data "newrelic_plugin" "foo" {
  guid = "com.example.my-plugin"
}

data "newrelic_plugin_component" "foo" {
  plugin_id = data.newrelic_plugin.foo.id
  name = "My Plugin Component"
}

resource "newrelic_alert_policy" "foo" {
  name = "foo"
}

resource "newrelic_plugins_alert_condition" "foo" {
  policy_id          = newrelic_alert_policy.foo.id
  name               = "foo"
  metric             = "Component/Summary/Consumers[consumers]"
  plugin_id          = data.newrelic_plugin.foo.id
  plugin_guid        = data.newrelic_plugin.foo.guid
  entities           = [data.newrelic_plugin_component.foo.id]
  value_function     = "average"
  metric_description = "Queue consumers"
```

```
  term {
    duration      = 5
    operator      = "below"
    priority      = "critical"
    threshold     = "0.75"
    time_function = "all"
  }
}
```

» Argument Reference

The following arguments are supported:

- `plugin_id` - (Required) The ID of the plugin instance this component belongs to.
- `name` - (Required) The name of the plugin component.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The ID of the plugin component.
- `health_status` - The health status of the plugin component.

» Data Source: newrelic_synthetics_monitor

Use this data source to get information about a specific synthetics monitor in New Relic that already exists. This can be used to set up a Synthetics alert condition.

» Example Usage

```
data "newrelic_synthetics_monitor" "bar" {
  name = "bar"
}

resource "newrelic_synthetics_alert_condition" "baz" {
  policy_id = newrelic_alert_policy.foo.id

  name        = "baz"
  monitor_id  = data.newrelic_synthetics_monitor.bar.id
  runbook_url = "https://www.example.com"
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the synthetics monitor in New Relic.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `monitor_id` - The ID of the synthetics monitor.

» Resource: newrelic_alert_channel

Use this resource to create and manage New Relic alert policies.

» Example Usage

» Email

```
resource "newrelic_alert_channel" "foo" {
  name = "foo"
  type = "email"

  config {
    recipients              = "foo@example.com"
    include_json_attachment = "1"
  }
}
```

See additional examples.

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the channel.
- `type` - (Required) The type of channel. One of: `email`, `slack`, `opsgenie`, `pagerduty`, `victorops`, or `webhook`.
- `config` - (Optional) A nested block that describes an alert channel configuration. Only one config block is permitted per alert channel definition. See Nested config blocks below for details.

- `configuration` - **Deprecated** (Optional) A map of key/value pairs with channel type specific values. This argument is deprecated. Use the `config` argument instead.

» Nested `config` blocks

Each alert channel type supports a specific set of arguments for the `config` block:

- `email`
  - `recipients` - (Required) Comma delimited list of email addresses.
  - `include_json_attachment` - (Optional) `0` or `1`. Flag for whether or not to attach a JSON document containing information about the associated alert to the email that is sent to recipients.
- `webhook`
  - `base_url` - (Required) The base URL of the webhook destination.
  - `auth_password` - (Optional) Specifies an authentication password for use with a channel. Supported by the `webhook` channel type.
  - `auth_type` - (Optional) Specifies an authentication method for use with a channel. Supported by the `webhook` channel type. Only HTTP basic authentication is currently supported via the value `BASIC`.
  - `auth_username` - (Optional) Specifies an authentication username for use with a channel. Supported by the `webhook` channel type.
  - `headers` - (Optional) A map of key/value pairs that represents extra HTTP headers to be sent along with the webhook payload
  - `payload` - (Optional) A map of key/value pairs that represents the webhook payload.
- `pagerduty`
  - `service_key` - (Required) Specifies the service key for integrating with Pagerduty.
- `victorops`
  - `key` - (Required) The key for integrating with VictorOps.
  - `route_key` - (Required) The route key for integrating with VictorOps.
- `slack`
  - `url` - (Required) Your organization's Slack URL.
  - `channel` - (Optional) The Slack channel to send notifications to.
- `opsgenie`
  - `api_key` - (Required) The API key for integrating with OpsGenie.
  - `region` - (Required) The data center region to store your data. Valid values are `US` and `EU`. Default is `US`.
  - `teams` - (Optional) A set of teams for targeting notifications. Multiple values are comma separated.
  - `tags` - (Optional) A set of tags for targeting notifications. Multiple values are comma separated.

- `recipients` - (Optional) A set of recipients for targeting notifications. Multiple values are comma separated.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The ID of the channel.

» Additional Examples

» Slack

```
resource "newrelic_alert_channel" "foo" {
  name = "slack-example"
  type = "slack"

  config {
    url     = "https://<YourOrganization>.slack.com"
    channel = "example-alerts-channel"
  }
}
```

» OpsGenie

```
resource "newrelic_alert_channel" "foo" {
  name = "opsgenie-example"
  type = "opsgenie"

  config {
    api_key    = "abc123"
    teams      = "team1, team2"
    tags       = "tag1, tag2"
    recipients = "user1@domain.com, user2@domain.com"
  }
}
```

» PagerDuty

```
resource "newrelic_alert_channel" "foo" {
  name = "pagerduty-example"
  type = "pagerduty"

  config {
    service_key = "abc123"
```

```
  }
}
```

» VictorOps

```
resource "newrelic_alert_channel" "foo" {
  name = "victorops-example"
  type = "victorops"

  config {
    key       = "abc123"
    route_key = "/example"
  }
}
```

» Import

Alert channels can be imported using the `id`, e.g.

```
$ terraform import newrelic_alert_channel.main <id>
```

» Resource: newrelic_alert_condition

Use this resource to create and manage alert conditions for APM, Browser, and Mobile in New Relic.

» Example Usage

```
data "newrelic_application" "app" {
  name = "my-app"
}

resource "newrelic_alert_policy" "foo" {
  name = "foo"
}

resource "newrelic_alert_condition" "foo" {
  policy_id = newrelic_alert_policy.foo.id

  name        = "foo"
  type        = "apm_app_metric"
  entities    = [data.newrelic_application.app.id]
  metric      = "apdex"
```

```
  runbook_url = "https://www.example.com"
  condition_scope = "application"

  term {
    duration      = 5
    operator      = "below"
    priority      = "critical"
    threshold     = "0.75"
    time_function = "all"
  }
}
```

» Argument Reference

The following arguments are supported:

- `policy_id` - (Required) The ID of the policy where this condition should be used.
- `name` - (Required) The title of the condition. Must be between 1 and 64 characters, inclusive.
- `type` - (Required) The type of condition. One of: `apm_app_metric`, `apm_jvm_metric`, `apm_kt_metric`, `servers_metric`, `browser_metric`, `mobile_metric`
- `entities` - (Required) The instance IDS associated with this condition.
- `metric` - (Required) The metric field accepts parameters based on the `type` set.
- `condition_scope` - (Required) `application` or `instance`. Choose `application` for most scenarios. If you are using the JVM plugin in New Relic, the `instance` setting allows your condition to trigger for specific app instances.
- `gc_metric` - (Optional) A valid Garbage Collection metric e.g. `GC/G1 Young Generation`. This is required if you are using `apm_jvm_metric` with `gc_cpu_time` condition type.
- `violation_close_timer` - (Optional) Automatically close instance-based violations, including JVM health metric violations, after the number of hours specified. Must be: `1`, `2`, `4`, `8`, `12` or `24`.
- `runbook_url` - (Optional) Runbook URL to display in notifications.
- `term` - (Required) A list of terms for this condition. See Terms below for details.
- `user_defined_metric` - (Optional) A custom metric to be evaluated.
- `user_defined_value_function` - (Optional) One of: `average`, `min`, `max`, `total`, or `sample_size`.

» Terms

The `term` mapping supports the following arguments:

- `duration` - (Required) In minutes, must be in the range of `5` to `120`, inclusive.
- `operator` - (Optional) `above`, `below`, or `equal`. Defaults to `equal`.
- `priority` - (Optional) `critical` or `warning`. Defaults to `critical`.
- `threshold` - (Required) Must be 0 or greater.
- `time_function` - (Required) `all` or `any`.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The ID of the alert condition.

» Import

Alert conditions can be imported using notation `alert_policy_id:alert_condition_id`, e.g.

```
$ terraform import newrelic_alert_condition.main 123456:6789012345
```

» Resource: newrelic_alert_policy

Use this resource to create and manage New Relic alert policies.

» Example Usage

```
resource "newrelic_alert_policy" "foo" {
  name = "foo"
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the policy.
- `incident_preference` - (Optional) The rollup strategy for the policy. Options include: `PER_POLICY`, `PER_CONDITION`, or `PER_CONDITION_AND_TARGET`. The default is `PER_POLICY`.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The ID of the policy.
- `created_at` - The time the policy was created.
- `updated_at` - The time the policy was last updated.

» Import

Alert policies can be imported using the `id`, e.g.

```
$ terraform import newrelic_alert_policy.main 12345
```

» Resource: newrelic_alert_policy_channel

Use this resource to map alert policies to alert channels in New Relic.

» Example Usage

```
resource "newrelic_alert_policy" "foo" {
  name = "foo"
}

resource "newrelic_alert_channel" "foo" {
  name = "foo"
  type = "email"

  config {
    recipients              = "foo@example.com"
    include_json_attachment = "1"
  }
}

resource "newrelic_alert_policy_channel" "foo" {
  policy_id  = newrelic_alert_policy.foo.id
  channel_id = newrelic_alert_channel.foo.id
}
```

» Argument Reference

The following arguments are supported:

- `policy_id` - (Required) The ID of the policy.
- `channel_id` - (Required) The ID of the channel.

» Resource: newrelic_dashboard

Use this resource to create and manage New Relic dashboards.

» Example Usage: Create a New Relic Dashboard

```
data "newrelic_application" "my_application" {
  name = "My Application"
}

resource "newrelic_dashboard" "exampledash" {
  title = "New Relic Terraform Example"

  filter {
    event_types = [
        "Transaction"
    ]
    attributes = [
        "appName",
        "name"
    ]
  }

  widget {
    title = "Requests per minute"
    visualization = "billboard"
    nrql = "SELECT rate(count(*), 1 minute) FROM Transaction"
    row = 1
    column = 1
  }

  widget {
    title = "Error rate"
    visualization = "gauge"
    nrql = "SELECT percentage(count(*), WHERE error IS True) FROM Transaction"
    threshold_red = 2.5
    row = 1
    column = 2
  }
```

```
  widget {
    title = "Average transaction duration, by application"
    visualization = "facet_bar_chart"
    nrql = "SELECT average(duration) FROM Transaction FACET appName"
    row = 1
    column = 3
  }

  widget {
    title = "Apdex, top 5 by host"
    duration = 1800000
    visualization = "metric_line_chart"
    entity_ids = [
      data.newrelic_application.my_application.id,
    ]
    metric {
        name = "Apdex"
        values = [ "score" ]
    }
    facet = "host"
    limit = 5
    row = 2
    column = 1
  }

  widget {
    title = "Requests per minute, by transaction"
    visualization = "facet_table"
    nrql = "SELECT rate(count(*), 1 minute) FROM Transaction FACET name"
    row = 2
    column = 2
  }

  widget {
    title = "Dashboard Note"
    visualization = "markdown"
    source = "### Helpful Links\n\n* [New Relic One](https://one.newrelic.com)\n* [Developer
    row = 2
    column = 3
  }
}
```

» Argument Reference

The following arguments are supported:

- `title` - (Required) The title of the dashboard.
- `icon` - (Optional) The icon for the dashboard. Valid values are `adjust`, `archive`, `bar-chart`, `bell`, `bolt`, `bug`, `bullhorn`, `bullseye`, `clock-o`, `cloud`, `cog`, `comments-o`, `crosshairs`, `dashboard`, `envelope`, `fire`, `flag`, `flask`, `globe`, `heart`, `leaf`, `legal`, `life-ring`, `line-chart`, `magic`, `mobile`, `money`, `none`, `paper-plane`, `pie-chart`, `puzzle-piece`, `road`, `rocket`, `shopping-cart`, `sitemap`, `sliders`, `tablet`, `thumbs-down`, `thumbs-up`, `trophy`, `usd`, `user`, and `users`. Defaults to `bar-chart`.
- `visibility` - (Optional) Determines who can see the dashboard in an account. Valid values are `all` or `owner`. Defaults to `all`.
- `editable` - (Optional) Determines who can edit the dashboard in an account. Valid values are `all`, `editable_by_all`, `editable_by_owner`, or `read_only`. Defaults to `editable_by_all`.
- `widget` - (Optional) A nested block that describes a visualization. Up to 300 `widget` blocks are allowed in a dashboard definition. See Nested widget blocks below for details.
- `filter` - (Optional) A nested block that describes a dashboard filter. Exactly one nested `filter` block is allowed. See Nested filter block below for details.

» Attribute Refence

In addition to all arguments above, the following attributes are exported:

- `dashboard_url` - The URL for viewing the dashboard.

» Nested `widget` blocks

All nested `widget` blocks support the following common arguments:

- `title` - (Required) A title for the widget.
- `visualization` - (Required) How the widget visualizes data. Valid values are `billboard`, `gauge`, `billboard_comparison`, `facet_bar_chart`, `faceted_line_chart`, `facet_pie_chart`, `facet_table`, `faceted_area_chart`, `heatmap`, `attribute_sheet`, `single_event`, `histogram`, `funnel`, `raw_json`, `event_feed`, `event_table`, `uniques_list`, `line_chart`, `comparison_line_chart`, `markdown`, and `metric_line_chart`.
- `row` - (Required) Row position of widget from top left, starting at `1`.
- `column` - (Required) Column position of widget from top left, starting at `1`.
- `width` - (Optional) Width of the widget. Valid values are `1` to `3` inclusive. Defaults to `1`.
- `height` - (Optional) Height of the widget. Valid values are `1` to `3` inclusive. Defaults to `1`.
- `notes` - (Optional) Description of the widget.

Each `visualization` type supports an additional set of arguments:

- `billboard`, `billboard_comparison`:
  - `nrql` - (Required) Valid NRQL query string. See Writing NRQL Queries for help.
  - `threshold_red` - (Optional) Threshold above which the displayed value will be styled with a red color.
  - `threshold_yellow` - (Optional) Threshold above which the displayed value will be styled with a yellow color.
- `gauge`:
  - `nrql` - (Required) Valid NRQL query string. See Writing NRQL Queries for help.
  - `threshold_red` - (Required) Threshold above which the displayed value will be styled with a red color.
  - `threshold_yellow` - (Optional) Threshold above which the displayed value will be styled with a yellow color.
- `facet_bar_chart`, `facet_pie_chart`, `facet_table`, `faceted_area_chart`, `faceted_line_chart`, or `heatmap`:
  - `nrql` - (Required) Valid NRQL query string. See Writing NRQL Queries for help.
  - `drilldown_dashboard_id` - (Optional) The ID of a dashboard to link to from the widget's facets.
- `attribute_sheet`, `comparison_line_chart`, `event_feed`, `event_table`, `funnel`, `histogram`, `line_chart`, `raw_json`, `single_event`, or `uniques_list`:
  - `nrql` - (Required) Valid NRQL query string. See Writing NRQL Queries for help.
- `markdown`:
  - `source` - (Required) The markdown source to be rendered in the widget.
- `metric_line_chart`:
  - `entity_ids` - (Required) A collection of entity ids to display data for. These are typically application IDs.
  - `metric` - (Required) A nested block that describes a metric. Nested `metric` blocks support the following arguments:
    - `name` - (Required) The metric name to display.
    - `values` - (Required) The metric values to display.
  - `duration` - (Required) The duration, in ms, of the time window represented in the chart.
  - `end_time` - (Optional) The end time of the time window represented in the chart in epoch time. When not set, the time window will end at the current time.
  - `facet` - (Optional) Can be set to "host" to facet the metric data by host.
  - `limit` - (Optional) The limit of distinct data series to display.
- `application_breakdown`:

– `entity_ids` - (Required) A collection of entity IDs to display data. These are typically application IDs.

» Nested `filter` block

The optional filter block supports the following arguments: * `event_types` - (Optional) A list of event types to enable filtering for. * `attributes` - (Optional) A list of attributes belonging to the specified event types to enable filtering for.

» Import

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

```
$ terraform import newrelic_dashboard.my_dashboard 8675309
```

» Resource: newrelic_infra_alert_condition

Use this resource to create and manage Infrastructure alert conditions in New Relic.

» Example Usage

```
resource "newrelic_alert_policy" "foo" {
  name = "foo"
}

resource "newrelic_infra_alert_condition" "high_disk_usage" {
  policy_id = newrelic_alert_policy.foo.id

  name       = "High disk usage"
  type       = "infra_metric"
  event      = "StorageSample"
  select     = "diskUsedPercent"
  comparison = "above"
  where      = "(`hostname` LIKE '%frontend%')"

  critical {
    duration      = 25
    value         = 90
    time_function = "all"
  }
```

```
  warning {
    duration      = 10
    value         = 90
    time_function = "all"
  }
}

resource "newrelic_infra_alert_condition" "high_db_conn_count" {
  policy_id = newrelic_alert_policy.foo.id

  name       = "High database connection count"
  type       = "infra_metric"
  event      = "DatastoreSample"
  select     = "provider.databaseConnections.Average"
  comparison = "above"
  where      = "(`hostname` LIKE '%db%')"
  integration_provider = "RdsDbInstance"

  critical {
    duration      = 25
    value         = 90
    time_function = "all"
  }
}

resource "newrelic_infra_alert_condition" "process_not_running" {
  policy_id = newrelic_alert_policy.foo.id

  name             = "Process not running (/usr/bin/ruby)"
  type             = "infra_process_running"
  comparison       = "equal"
  process_where    = "`commandName` = '/usr/bin/ruby'"

  critical {
    duration      = 5
    value         = 0
  }
}

resource "newrelic_infra_alert_condition" "host_not_reporting" {
  policy_id = newrelic_alert_policy.foo.id

  name       = "Host not reporting"
  type       = "infra_host_not_reporting"
  event      = "StorageSample"
  select     = "diskUsedPercent"
```

```
  where      = "(`hostname` LIKE '%frontend%')"

  critical {
    duration = 5
  }
}
```

» Argument Reference

The following arguments are supported:

- `policy_id` - (Required) The ID of the alert policy where this condition should be used.
- `name` - (Required) The Infrastructure alert condition's name.
- `type` - (Required) The type of Infrastructure alert condition. Valid values are `infra_process_running`, `infra_metric`, and `infra_host_not_reporting`.
- `event` - (Required) The metric event; for example, `SystemSample` or `StorageSample`. Supported by the `infra_metric` condition type.
- `select` - (Required) The attribute name to identify the metric being targeted; for example, `cpuPercent`, `diskFreePercent`, or `memoryResidentSizeBytes`. The underlying API will automatically populate this value for Infrastructure integrations (for example `diskFreePercent`), so make sure to explicitly include this value to avoid diff issues. Supported by the `infra_metric` condition type.
- `comparison` - (Required) The operator used to evaluate the threshold value. Valid values are `above`, `below`, and `equal`. Supported by the `infra_metric` and `infra_process_running` condition types.
- `critical` - (Required) Identifies the threshold parameters for opening a critial alert violation. See Thresholds below for details.
- `warning` - (Optional) Identifies the threshold parameters for opening a warning alert violation. See Thresholds below for details.
- `enabled` - (Optional) Whether the condition is turned on or off. Valid values are `true` and `false`. Defaults to `true`.
- `where` - (Optional) If applicable, this identifies any Infrastructure host filters used; for example: `hostname LIKE '%cassandra%'`.
- `process_where` - (Optional) Any filters applied to processes; for example: `commandName = 'java'`. Supported by the `infra_process_running` condition type.
- `integration_provider` - (Optional) For alerts on integrations, use this instead of `event`. Supported by the `infra_metric` condition type.
- `runbook_url` - (Optional) Runbook URL to display in notifications.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The ID of the Infrastructure alert condition.

» Thresholds

The `critical` and `warning` threshold mapping supports the following arguments:

- `duration` - (Required) Identifies the number of minutes the threshold must be passed or met for the alert to trigger. Threshold durations must be between 1 and 60 minutes (inclusive).
- `value` - (Optional) Threshold value, computed against the `comparison` operator. Supported by `infra_metric` and `infra_process_running` alert condition types.
- `time_function` - (Optional) Indicates if the condition needs to be sustained or to just break the threshold once; `all` or `any`. Supported by the `infra_metric` alert condition type.

» Resource: newrelic_insights_event

Use this resource to create one or more Insights events during a terraform run.

» Example Usage

```
resource "newrelic_insights_event" "foo" {
  event {
    type = "MyEvent"

    timestamp = 1232471100

    attribute {
      key   = "a_string_attribute"
      value = "a string"
    }
    attribute {
      key   = "an_integer_attribute"
      value = 42
      type  = "int"
    }
    attribute {
```

```
      key   = "a_float_attribute"
      value = 101.1
      type  = "float"
    }
  }
}
```

» Argument Reference

The following arguments are supported:

- `event` - (Required) An event to insert into Insights. Multiple event blocks can be defined. See Events below for details.

» Events

The `event` mapping supports the following arguments:

- `type` - (Required) The event's name. Can be a combination of alphanumeric characters, underscores, and colons.
- `timestamp` - (Optional) Must be a Unix epoch timestamp. You can define timestamps either in seconds or in milliseconds.
- `attribute` - (Required) An attribute to include in your event payload. Multiple attribute blocks can be defined for an event. See Attributes below for details.

» Attributes

The `attribute` mapping supports the following arguments:

- `key` - (Required) The name of the attribute.
- `value` - (Required) The value of the attribute.
- `type` - (Optional) Specify the type for the attribute value. This is useful when passing integer or float values to Insights. Allowed values are `string`, `int`, or `float`. Defaults to `string`.

» Resource: newrelic_nrql_alert_condition

Use this resource to create and manage NRQL alert conditions in New Relic.

» Example Usage

» Type: `static` (default)

```
resource "newrelic_alert_policy" "foo" {
  name = "foo"
}

resource "newrelic_nrql_alert_condition" "foo" {
  policy_id = newrelic_alert_policy.foo.id

  name        = "foo"
  type        = "static"
  runbook_url = "https://www.example.com"
  enabled     = true

  term {
    duration      = 5
    operator      = "below"
    priority      = "critical"
    threshold     = "1"
    time_function = "all"
  }

  nrql {
    query       = "SELECT count(*) FROM SyntheticCheck WHERE monitorId = '<monitorId>'"
    since_value = "3"
  }

  value_function = "single_value"
}
```

See additional examples.

» Argument Reference

The following arguments are supported:

- `policy_id` - (Required) The ID of the policy where this condition should be used.
- `name` - (Required) The title of the condition
- `type` - (Optional) The type of the condition. Valid values are `static` or `outlier`. Defaults to `static`.
- `runbook_url` - (Optional) Runbook URL to display in notifications.
- `enabled` - (Optional) Whether to enable the alert condition. Valid values are `true` and `false`. Defaults to `true`.

- `term` - (Required) A list of terms for this condition. See Terms below for details.
- `nrql` - (Required) A NRQL query. See NRQL below for details.
- `value_function` - (Optional) Possible values are `single_value`, `sum`.
- `expected_groups` - (Optional) Number of expected groups when using `outlier` detection.
- `ignore_overlap` - (Optional) Whether to look for a convergence of groups when using `outlier` detection.
- `violation_time_limit_seconds` - (Optional) Sets a time limit, in seconds, that will automatically force-close a long-lasting violation after the time limit you select. Possible values are `3600`, `7200`, `14400`, `28800`, `43200`, and `86400`.

» Terms

The `term` mapping supports the following arguments:

- `duration` - (Required) In minutes, must be in the range of `1` to `120`, inclusive.
- `operator` - (Optional) `above`, `below`, or `equal`. Defaults to `equal`.
- `priority` - (Optional) `critical` or `warning`. Defaults to `critical`.
- `threshold` - (Required) Must be 0 or greater.
- `time_function` - (Required) `all` or `any`.

» NRQL

The `nrql` attribute supports the following arguments:

- `query` - (Required) The NRQL query to execute for the condition.
- `since_value` - (Required) The value to be used in the `SINCE <X> MINUTES AGO` clause for the NRQL query. Must be between `1` and `20`.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The ID of the NRQL alert condition.

» Additional Examples

» Type: `outlier`

```
resource "newrelic_alert_policy" "foo" {
  name = "foo"
```

```
}

resource "newrelic_nrql_alert_condition" "foo" {
  policy_id = newrelic_alert_policy.foo.id

  name        = "outlier-example"
  runbook_url = "https://bar.example.com"
  enabled     = true

  term {
    duration      = 10
    operator      = "above"
    priority      = "critical"
    threshold     = "0.65"
    time_function = "all"
  }
  nrql {
    query       = "SELECT percentile(duration, 99) FROM Transaction FACET remote_ip"
    since_value = "3"
  }
  type            = "outlier"
  expected_groups = 2
  ignore_overlap  = true
}
```

» Import

Alert conditions can be imported using a composite ID of `<policy_id>:<condition_id>`, e.g.

```
$ terraform import newrelic_nrql_alert_condition.main 12345:67890
```

The actual values for `policy_id` and `condition_id` can be retrieved from the following URL when looking at the alert condition:

https://alerts.newrelic.com/accounts/<account_id>/policies/**<policy_id>**/conditions/**<condition_id>**/

» Resource: newrelic_plugins_alert_condition

Use this resource to create and manage plugins alert conditions in New Relic.

» Example Usage

```
data "newrelic_plugin" "foo" {
```

```
  guid = "com.example.my-plugin"
}

resource "newrelic_alert_policy" "foo" {
  name = "foo"
}

resource "newrelic_plugins_alert_condition" "foo" {
  policy_id          = newrelic_alert_policy.foo.id
  name               = "foo"
  metric             = "Component/Summary/Consumers[consumers]"
  plugin_id          = data.newrelic_plugin.foo.id
  plugin_guid        = data.newrelic_plugin.foo.guid
  value_function     = "average"
  metric_description = "Queue consumers"

  term {
    duration      = 5
    operator      = "below"
    priority      = "critical"
    threshold     = "0.75"
    time_function = "all"
  }
}
```

» Argument Reference

The following arguments are supported:

- `policy_id` - (Required) The ID of the policy where this condition should be used.
- `name` - (Required) The title of the condition. Must be between 1 and 64 characters, inclusive.
- `metric` - (Required) The metric field accepts parameters based on the `type` set.
- `plugin_id` - (Required) The ID of the installed plugin instance which produces the metric.
- `plugin_guid` - (Required) The GUID of the plugin which produces the metric.
- `runbook_url` - (Optional) Runbook URL to display in notifications.
- `term` - (Required) A list of terms for this condition. See Terms below for details.

» Terms

The `term` mapping supports the following arguments:

- `duration` - (Required) In minutes, must be in the range of `5` to `120`, inclusive.
- `operator` - (Optional) `above`, `below`, or `equal`. Defaults to `equal`.
- `priority` - (Optional) `critical` or `warning`. Defaults to `critical`.
- `threshold` - (Required) Must be 0 or greater.
- `time_function` - (Required) `all` or `any`.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The ID of the alert condition.

» Import

Alert conditions can be imported using the `id`, e.g.

```
$ terraform import newrelic_plugins_alert_condition.main 12345
```

» Resource: newrelic_synthetics_alert_condition

Use this resource to create and manage synthetics alert conditions in New Relic.

» Example Usage

```
data "newrelic_synthetics_monitor" "foo" {
  name = "foo"
}

resource "newrelic_synthetics_alert_condition" "foo" {
  policy_id = newrelic_alert_policy.foo.id

  name        = "foo"
  monitor_id  = data.newrelic_synthetics_monitor.foo.id
  runbook_url = "https://www.example.com"
}
```

» Argument Reference

The following arguments are supported:

- `policy_id` - (Required) The ID of the policy where this condition should be used.
- `name` - (Required) The title of this condition.
- `monitor_id` - (Required) The ID of the Synthetics monitor to be referenced in the alert condition.
- `runbook_url` - (Optional) Runbook URL to display in notifications.
- `enabled` - (Optional) Set whether to enable the alert condition. Defaults to `true`.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The ID of the Synthetics alert condition.

» newrelic_synthetics_monitor

Use this resource to create, update, and delete a synthetics monitor in New Relic.

» Example Usage

» Type: `SIMPLE`

```
resource "newrelic_synthetics_monitor" "foo" {
  name = "foo"
  type = "SIMPLE"
  frequency = 5
  status = "ENABLED"
  locations = ["AWS_US_EAST_1", "AWS_US_EAST_2"]

  uri                       = "https://example.com"               # Required for type "SIMPL
  validation_string         = "add example validation check here" # Optional for type "SIMPL
  verify_ssl                = true                                # Optional for type "SIMPL
}
```

See additional examples.

» Argument Reference

The following arguments are supported:

- `name` - (Required) The title of this monitor.
- `type` - (Required) The monitor type. Valid values are `SIMPLE`, `BROWSER`, `SCRIPT_BROWSER`, and `SCRIPT_API`.
- `frequency` - (Required) The interval (in minutes) at which this monitor should run.
- `status` - (Required) The monitor status (i.e. `ENABLED`, `MUTED`, `DISABLED`)
- `locations` - (Required) The locations in which this monitor should be run.
- `sla_threshold` - (Optional) The base threshold for the SLA report.

The `SIMPLE` monitor type supports the following additional arguments:

- `uri` - (Required) The URI for the monitor to hit.
- `validation_string` - (Optional) The string to validate against in the response.
- `verify_ssl` - (Optional) Verify SSL.
- `bypass_head_request` - (Optional) Bypass HEAD request.
- `treat_redirect_as_failure` - (Optional) Fail the monitor check if redirected.

The `BROWSER` monitor type supports the following additional arguments:

- `uri` - (Required) The URI for the monitor to hit.
- `validation_string` - (Optional) The string to validate against in the response.
- `verify_ssl` - (Optional) Verify SSL.

» Attributes Reference

The following attributes are exported:

- `id` - The ID of the Synthetics monitor.

» Additional Examples

Type: `BROWSER`

```
resource "newrelic_synthetics_monitor" "foo" {
  name = "foo"
  type = "BROWSER"
  frequency = 5
  status = "ENABLED"
  locations = ["AWS_US_EAST_1"]
```

```
  uri                       = "https://example.com"               # required for type "SIMPL
  validation_string         = "add example validation check here" # optional for type "SIMPL
  verify_ssl                = true                                # optional for type "SIMPL
  bypass_head_request       = true                                # Note: optional for type
  treat_redirect_as_failure = true                                # Note: optional for type
}
```

Type: `SCRIPT_BROWSER`

```
resource "newrelic_synthetics_monitor" "foo" {
  name = "foo"
  type = "SCRIPT_BROWSER"
  frequency = 5
  status = "ENABLED"
  locations = ["AWS_US_EAST_1"]
}
```

Type: `SCRIPT_API`

```
resource "newrelic_synthetics_monitor" "foo" {
  name = "foo"
  type = "SCRIPT_API"
  frequency = 5
  status = "ENABLED"
  locations = ["AWS_US_EAST_1"]
}
```

» Resource: newrelic_synthetics_monitor_script

Use this resource to update a synthetics monitor script in New Relic.

» Example Usage

```
resource "newrelic_synthetics_monitor" "foo" {
  name = "foo"
  type = "SCRIPT_BROWSER"
  frequency = 5
  status = "ENABLED"
  locations = ["AWS_US_EAST_1"]
}

data "template_file" "foo_script" {
  template = file("${path.module}/foo_script.tpl")
}
```

```
resource "newrelic_synthetics_monitor_script" "foo_script" {
  monitor_id = newrelic_synthetics_monitor.foo.id
  text = data.template_file.foo_script.rendered
}
```

» Argument Reference

The following arguments are supported:

- `monitor_id` - (Required) The ID of the monitor to attach the script to.
- `text` - (Required) plaintext of the monitor script.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The ID of the Synthetics monitor that the script is attached to.