» Resource: signalfx_aws_integration

SignalFx AWS CloudWatch integrations. For help with this integration see Monitoring Amazon Web Services.

» Example Usage

```
resource "signalfx_aws_integration" "aws_myteam" {
    name = "AWSFoo"
    enabled = true

        auth_method = "ExternalId"
        role_arn = "arn:aws:iam::XXX:role/SignalFx-Read-Role"
        regions = ["us-east-1"]
        poll_rate = 300
        import_cloud_watch = true
        enable_aws_usage = true

        custom_namespace_sync_rule {
            default_action = "Exclude"
            filter_action = "Include"
            filter_source = "filter('code', '200')"
            namespace = "fart"
        }

        namespace_sync_rule {
            default_action = "Exclude"
            filter_action = "Include"
            filter_source = "filter('code', '200')"
            namespace = "AWS/EC2"
        }
}
```

» Service Names

Fields that expect an AWS service/namespace will work with one of: ”AWS/ApiGateway” ”AWS/AppStream” ”AWS/AutoScaling” ”AWS/Billing” ”AWS/CloudFront” ”AWS/CloudSearch” ”AWS/Events” ”AWS/Logs” ”AWS/Connect” ”AWS/DMS” ”AWS/DX” ”AWS/DynamoDB” ”AWS/EC2” ”AWS/EC2Spot” ”AWS/ECS” ”AWS/ElasticBeanstalk” ”AWS/EBS” ”AWS/EFS” ”AWS/ELB” ”AWS/ApplicationELB” ”AWS/NetworkELB” ”AWS/ElasticTranscoder” ”AWS/ElastiCache” ”AWS/ES” ”AWS/ElasticMapReduce” ”AWS/GameLift” ”AWS/Inspector” ”AWS/IoT” ”AWS/KMS” ”AWS/KinesisAnalytics”

"AWS/Firehose" "AWS/Kinesis" "AWS/KinesisVideo" "AWS/Lambda" "AWS/Lex" "AWS/ML" "AWS/OpsWorks" "AWS/Polly" "AWS/Redshift" "AWS/RDS" "AWS/Route53" "AWS/SageMaker" "AWS/DDoSProtection" "AWS/SES" "AWS/SNS" "AWS/SQS" "AWS/S3" "AWS/SWF" "AWS/States" "AWS/StorageGateway" "AWS/Translate" "AWS/NATGateway" "AWS/VPN (VPN)" "WAF" "AWS/WorkSpaces".

» Argument Reference

- `name` - (Required) Name of the integration.
- `enabled` - (Required) Whether the integration is enabled.
- `auth_method` - (Optional) The mechanism used to authenticate with AWS. The allowed values are "ExternalID" or "SecurityToken".
- `custom_cloudwatch_namespaces` - (Optional) List of custom AWS CloudWatch namespaces to monitor. Custom namespaces contain custom metrics that you define in AWS; SignalFx imports the metrics so you can monitor them.
- `custom_namespace_sync_rule` - (Optional) Each element controls the data collected by SignalFx for the specified namespace. Conflicts with the `custom_cloudwatch_namespaces` property.
  - `default_action` - (Required) Controls the SignalFx default behavior for processing data from an AWS namespace. If you do specify a filter, use this property to control how SignalFx treats data that doesn't match the filter. The available actions are one of `"Include"` or `"Exclude"`.
  - `filter_action` - (Required) Controls how SignalFx processes data from a custom AWS namespace. The available actions are one of `"Include"` or `"Exclude"`.
  - `filter_source` - (Required) Expression that selects the data that SignalFx should sync for the custom namespace associated with this sync rule. The expression uses the syntax defined for the SignalFlow `filter()` function; it can be any valid SignalFlow filter expression.
  - `namespace` - (Required) An AWS custom namespace having custom AWS metrics that you want to sync with SignalFx. See the AWS documentation on publishing metrics for more information.
- `namespace_sync_rule` - (Optional) Each element in the array is an object that contains an AWS namespace name and a filter that controls the data that SignalFx collects for the namespace. Conflicts with the `services` property. If you don't specify either property, SignalFx syncs all data in all AWS namespaces.
  - `default_action` - (Required) Controls the SignalFx default behavior for processing data from an AWS namespace. If you do specify a filter, use this property to control how SignalFx treats data that doesn't match the filter. The available actions are one of `"Include"` or `"Exclude"`.

  - `filter_action` - (Required) Controls how SignalFx processes data from a custom AWS namespace. The available actions are one of `"Include"` or `"Exclude"`.
  - `filter_source` - (Required) Expression that selects the data that SignalFx should sync for the custom namespace associated with this sync rule. The expression uses the syntax defined for the SignalFlow `filter()` function; it can be any valid SignalFlow filter expression.
  - `namespace` - (Required) An AWS custom namespace having custom AWS metrics that you want to sync with SignalFx. See the AWS documentation on publishing metrics for more information.
- `enable_aws_usage` - (Optional) Flag that controls how SignalFx imports usage metrics from AWS to use with AWS Cost Optimizer. If `true`, SignalFx imports the metrics.
- `import_cloud_watch` - (Optional) Flag that controls how SignalFx imports Cloud Watch metrics. If true, SignalFx imports Cloud Watch metrics from AWS.
- `key` - (Optional) If you specify `auth_method = \"SecurityToken\"` in your request to create an AWS integration object, use this property to specify the key.
- `regions` - (Optional) List of AWS regions that SignalFx should monitor.
- `role_arn` - (Optional) Role ARN that you add to an existing AWS integration object.
- `services` - (Optional) List of AWS services that you want SignalFx to monitor. Each element is a string designating an AWS service. Conflicts with `namespace_sync_rule`.
- `poll_rate` - (Oprional) AWS poll rate (in seconds). One of `60` or `300`.

» Resource: signalfx_azure_integration

SignalFx Azure integrations. For help with this integration see Monitoring Microsoft Azure.

» Example Usage

```
resource "signalfx_azure_integration" "azure_myteam" {
    name = "Azure Foo"
    enabled = true

    resource "signalfx_azure_integration" "azure_myteamXX" {
        name = "AzureFoo"
        enabled = false

        environment = "azure"
```

```
            poll_rate = 300

        secret_key = "XXX"

        app_id = "YYY"

        tenant_id = "ZZZ"

        services = [ "microsoft.sql/servers/elasticpools" ]

        subscriptions = [ "microsoft.sql/servers/elasticpools" ]
    }
}
```

» Service Names

Fields that expect an Azure service will work with one of: ”microsoft.sql/servers/elasticpools” ”microsoft.storage/storageaccounts” ”microsoft.storage/storageaccountsservices/tableservices” ”microsoft.storage/storageaccountsservices/blobservices” ”microsoft.storage/storageaccounts/queueservices” ”microsoft.storage/storageaccounts/fileservices” ”microsoft.compute/virtualmachinescalesets” ”microsoft.compute/virtualmachinescalesets/virtualmachines” ”microsoft.compute/virtualmachines” ”microsoft.devices/iothubs” ”microsoft.eventHub/namespaces” ”microsoft.batch/batchaccounts” ”microsoft.sql/servers/databases” ”microsoft.cache/redis” ”microsoft.logic/workflows”.

» Argument Reference

- `name` - (Required) Name of the integration.
- `enabled` - (Required) Whether the integration is enabled.
- `app_id` - (Required) Azure application ID for the SignalFx app. To learn how to get this ID, see the topic Connect to Microsoft Azure in the product documentation.
- `environment` (Optional) What type of Azure integration this is. The allowed values are `\"azure_us_government\"` and `\"azure\"`. Defaults to `\"azure\"`.
- `poll_rate` - (Optional) AWS poll rate (in seconds). One of `60` or `300`.
- `secret_key` - (Required) Azure secret key that associates the SignalFx app in Azure with the Azure tenant ID. To learn how to get this ID, see the topic Connect to Microsoft Azure in the product documentation.
- `services` - (Required) List of Microsoft Azure service names for the Azure services you want SignalFx to monitor.
- `subscriptions` - (Required) List of Azure subscriptions that SignalFx should monitor.

- `tenant_id` (Required) Azure ID of the Azure tenant. To learn how to get this ID, see the topic Connect to Microsoft Azure in the product documentation.

» Resource: signalfx_dashboard

A dashboard is a curated collection of specific charts and supports dimensional filters, dashboard variables and time range options. These options are applied to all charts in the dashboard, providing a consistent view of the data displayed in that dashboard. This also means that when you open a chart to drill down for more details, you are viewing the same data that is visible in the dashboard view.

NOTE: Since every dashboard is included in a dashboard group (SignalFx collection of dashboards), you need to create that first and reference it as shown in the example.

» Example Usage

```
resource "signalfx_dashboard" "mydashboard0" {
    name = "My Dashboard"
    dashboard_group = "${signalfx_dashboard_group.mydashboardgroup0.id}"

    time_range = "-30m"

    filter {
        property = "collector"
        values = ["cpu", "Diamond"]
    }
    variable {
        property = "region"
        alias = "region"
        values = ["uswest-1-"]
    }
    chart {
        chart_id = "${signalfx_time_chart.mychart0.id}"
        width = 12
        height = 1
    }
    chart {
        chart_id = "${signalfx_time_chart.mychart1.id}"
        width = 5
        height = 2
    }
```

```
}
```

» Argument Reference

The following arguments are supported in the resource block:

- `name` - (Required) Name of the dashboard.
- `dashboard_group` - (Required) The ID of the dashboard group that contains the dashboard.
- `description` - (Optional) Description of the dashboard.
- `charts_resolution` - (Optional) Specifies the chart data display resolution for charts in this dashboard. Value can be one of `"default"`, `"low"`, `"high"`, or `"highest"`.
- `time_range` - (Optional) The time range prior to now to visualize. SignalFx time syntax (e.g. `"-5m"`, `"-1h"`).
- `start_time` - (Optional) Seconds since epoch. Used for visualization. You must specify time_span_type = `"absolute"` too.
- `end_time` - (Optional) Seconds since epoch. Used for visualization. You must specify time_span_type = `"absolute"` too.
- `filter` - (Optional) Filter to apply to the charts when displaying the dashboard.
  - `property` - (Required) A metric time series dimension or property name.
  - `negated` - (Optional) Whether this filter should be a not filter. `false` by default.
  - `values` - (Required) List of of strings (which will be treated as an OR filter on the property).
  - `apply_if_exist` - (Optional) If true, this filter will also match data that doesn't have this property at all.
- `variable` - (Optional) Dashboard variable to apply to each chart in the dashboard.
  - `property` - (Required) A metric time series dimension or property name.
  - `alias` - (Required) An alias for the dashboard variable. This text will appear as the label for the dropdown field on the dashboard.
  - `description` - (Optional) Variable description.
  - `values` - (Optional) List of of strings (which will be treated as an OR filter on the property).
  - `value_required` - (Optional) Determines whether a value is required for this variable (and therefore whether it will be possible to view this dashboard without this filter applied). `false` by default.
  - `values_suggested` - (Optional) A list of strings of suggested values for this variable; these suggestions will receive priority when values are autosuggested for this variable.
  - `restricted_suggestions` - (Optional) If `true`, this variable may

only be set to the values listed in `values_suggested` and only these values will appear in autosuggestion menus. `false` by default.
  - `replace_only` - (Optional) If `true`, this variable will only apply to charts that have a filter for the property.
  - `apply_if_exist` - (Optional) If true, this variable will also match data that doesn't have this property at all.
- `chart` - (Optional) Chart ID and layout information for the charts in the dashboard.
  - `chart_id` - (Required) ID of the chart to display.
  - `width` - (Optional) How many columns (out of a total of 12) the chart should take up (between `1` and `12`). `12` by default.
  - `height` - (Optional) How many rows the chart should take up (greater than or equal to `1`). `1` by default.
  - `row` - (Optional) The row to show the chart in (zero-based); if `height > 1`, this value represents the topmost row of the chart (greater than or equal to `0`).
  - `column` - (Optional) The column to show the chart in (zero-based); this value always represents the leftmost column of the chart (between `0` and `11`).
- `grid` - (Optional) Grid dashboard layout. Charts listed will be placed in a grid by row with the same width and height. If a chart cannot fit in a row, it will be placed automatically in the next row.
  - `chart_ids` - (Required) List of IDs of the charts to display.
  - `width` - (Optional) How many columns (out of a total of 12) every chart should take up (between `1` and `12`). `12` by default.
  - `height` - (Optional) How many rows every chart should take up (greater than or equal to `1`). `1` by default.
- `column` - (Optional) Column layout. Charts listed will be placed in a single column with the same width and height.
  - `chart_ids` - (Required) List of IDs of the charts to display.
  - `column` - (Optional) Column number for the layout.
  - `width` - (Optional) How many columns (out of a total of `12`) every chart should take up (between `1` and `12`). `12` by default.
  - `height` - (Optional) How many rows every chart should take up (greater than or equal to 1). 1 by default.
- `event_overlay` - (Optional) Specify a list of event overlays to include in the dashboard. Note: These overlays correspond to the *suggested* event overlays specified in the web UI, and they're not automatically applied as active overlays. To set default active event overlays, use the `selected_event_overlay` property instead.
  - `line` - (Optional) Show a vertical line for the event. `false` by default.
  - `label` - (Optional) Text shown in the dropdown when selecting this overlay from the menu.
  - `color` - (Optional) Color to use : gray, blue, azure, navy, brown, orange, yellow, iris, magenta, pink, purple, violet, lilac, emerald, green, aquamarine.

  - `signal` - Search term used to choose the events shown in the overlay.
  - `type` - (Optional) Can be set to `eventTimeSeries` (the default) to refer to externally reported events, or `detectorEvents` to refer to events from detector triggers.
  - `source` - (Optional) Each element specifies a filter to use against the signal specified in the `signal`.
    * `property` - The name of a dimension to filter against.
    * `values` - A list of values to be used with the `property`, they will be combined via `OR`.
    * `negated` - (Optional) If true, only data that does not match the specified value of the specified property appear in the event overlay. Defaults to `false`.
- `selected_event_overlay` - (Optional) Defines event overlays which are enabled by **default**. Any overlay specified here should have an accompanying entry in `event_overlay`, which are similar to the properties here.
  - `signal` - Search term used to choose the events shown in the overlay.
  - `type` - (Optional) Can be set to `eventTimeSeries` (the default) to refer to externally reported events, or `detectorEvents` to refer to events from detector triggers.
  - `source` - (Optional) Each element specifies a filter to use against the signal specified in the `signal`.
    * `property` - The name of a dimension to filter against.
    * `values` - A list of values to be used with the `property`, they will be combined via `OR`.
    * `negated` - (Optional) If true, only data that does not match the specified value of the specified property appear in the event overlay. Defaults to `false`.

» Dashboard Layout Information

Every SignalFx dashboard is shown as a grid of 12 columns and potentially infinite number of rows. The dimension of the single column depends on the screen resolution.

When you define a dashboard resource, you need to specify which charts (by `chart_id`) should be displayed in the dashboard, along with layout information determining where on the dashboard the charts should be displayed. You have to assign to every chart a **width** in terms of number of column to cover up (from 1 to 12) and a **height** in terms of number of rows (more or equal than 1). You can also assign a position in the dashboard grid where you like the graph to stay. In order to do that, you assign a **row** that represent the topmost row of the chart and a **column** that represent the leftmost column of the chart. If by mistake, you wrote a configuration where there are not enough columns to accommodate your charts in a specific row, they will be split in different rows. In case a **row** was specified with value higher than 1, if all the rows above are

not filled by other charts, the chart will be placed the **first empty row**.

The are a bunch of use cases where this layout makes things too verbose and hard to work with loops. For those you can now use one of these two layouts: grids and columns.

» Grid

The dashboard is divided into equal-sized charts (defined by `width` and `height`). If a chart does not fit in the same row (because the total width > max allowed by the dashboard), this and the next ones will be place in the next row(s).

```
resource "signalfx_dashboard" "grid_example" {
    name = "Grid"
    dashboard_group = "${signalfx_dashboard_group.example.id}"
    time_range = "-15m"

    grid {
        chart_ids = ["${concat(signalfx_time_chart.rps.*.id,
                signalfx_time_chart.50ths.*.id,
                signalfx_time_chart.99ths.*.id,
                signalfx_time_chart.idle_workers.*.id,
                signalfx_time_chart.cpu_idle.*.id)}"]
        width = 3
        height = 1
    }
}
```

» Column

The dashboard is divided into equal-sized charts (defined by `width` and `height`). The charts are placed in the grid by column (column number is called `column`).

```
resource "signalfx_dashboard" "load" {
    name = "Load"
    dashboard_group = "${signalfx_dashboard_group.example.id}"

    column {
        chart_ids = ["${signalfx_single_value_chart.rps.*.id}"]
        width = 2
    }
    column {
        chart_ids = ["${signalfx_time_chart.cpu_capacity.*.id}"]
        column = 2
        width = 4
    }
```

```
    chart {
        chart_id = "${signalfx_single_value_chart.loadbalancer_rps.id}"
        width = 2
        height = 1
        row = 0
        column = 6
    }
    chart {
        chart_id = "${signalfx_time_chart.cpu_idle.id}"
        width = 4
        height = 1
        row = 0
        column = 8
    }
    chart {
        chart_id = "${signalfx_time_chart.network.id}"
        width = 6
        height = 3
        row = 1
        column = 6
    }
    chart {
        chart_id = "${signalfx_single_value_chart.disk.id}"
        width = 2
        height = 1
        row = 5
        column = 6
    }
    chart {
        chart_id = "${signalfx_time_chart.mem.id}"
        width = 4
        height = 1
        row = 5
        column = 8
    }
}
```

» Resource: signalfx_dashboard_group

In the SignalFx web UI, a dashboard group is a collection of dashboards.

NOTE: Dashboard groups cannot be accessed directly, but just via a dashboard contained in them. This is the reason why make show won't show any of yours dashboard groups.

» Example Usage

```
resource "signalfx_dashboard_group" "mydashboardgroup0" {
    name = "My team dashboard group"
    description = "Cool dashboard group"
}
```

» Example Usage With Mirrored Dashboards

```
resource "signalfx_dashboard_group" "mydashboardgroup_withmirrors" {
    name = "My team dashboard group"
    description = "Cool dashboard group"

    // You can add as many of these as you like. Make sure your account
    // supports this feature!
    dashboard {
      dashboard_id = "${signalfx_dashboard.gc_dashboard.id}"
      name_override = "GC For My Service"
      description_override = "Garbage Collection dashboard maintained by JVM team"

      filter_override {
        property = "service"
        values = [ "myservice" ]
        negated = false
      }

      variable_override {
        property = "region"
        values = ["us-west1"]
        values_suggested = ["us-west-1", "us-east-1"]
      }
    }
}
```

» Argument Reference

The following arguments are supported in the resource block:

- `name` - (Required) Name of the dashboard group.
- `description` - (Required) Description of the dashboard group.
- `teams` - (Optional) Team IDs to associate the dashboard group to.
- `dashboard` - (Optional) Mirrored dashboards in this dashboard group. **Note:** This feature is not present in all accounts. Please contact support if you are unsure.

- `dashboard_id` - (Required) The dashboard id to mirror
- `name_override` - (Optional) The name that will override the original dashboards's name.
- `description_override` - (Optional) The description that will override the original dashboards's description.
- `filter_override` - (Optional) The description that will override the original dashboards's description.
- `property` - (Required) The name of a dimension to filter against.
- `values` - (Required) A list of values to be used with the `property`, they will be combined via `OR`.
- `negated` - (Optional) If true, only data that does not match the specified value of the specified property appear in the event overlay. Defaults to `false`.
- `filter_override` - (Optional) The description that will override the original dashboards's description.
- `property` - (Required) A metric time series dimension or property name.
- `values` - (Optional) (Optional) List of of strings (which will be treated as an OR filter on the property).
- `values_suggested` - (Optional) A list of strings of suggested values for this variable; these suggestions will receive priority when values are autosuggested for this variable.

» Resource: signalfx_detector

Provides a SignalFx detector resource. This can be used to create and manage detectors.

» Example Usage

```
resource "signalfx_detector" "application_delay" {
    count = "${length(var.clusters)}"
    name = " max average delay - ${var.clusters[count.index]}"
    description = "your application is slow - ${var.clusters[count.index]}"
    max_delay = 30
    program_text = <<-EOF
        signal = data('app.delay', filter('cluster','${var.clusters[count.index]}'), extrapo
        detect(when(signal > 60, '5m')).publish('Processing old messages 5m')
        detect(when(signal > 60, '30m')).publish('Processing old messages 30m')
    EOF
    rule {
        description = "maximum > 60 for 5m"
        severity = "Warning"
```

```
        detect_label = "Processing old messages 5m"
        notifications = ["Email,foo-alerts@bar.com"]
    }
    rule {
        description = "maximum > 60 for 30m"
        severity = "Critical"
        detect_label = "Processing old messages 30m"
        notifications = ["Email,foo-alerts@bar.com"]
    }
}

provider "signalfx" {}

variable "clusters" {
    default = ["clusterA", "clusterB"]
}
```

» Notification Format

As SignalFx supports different notification mechanisms a comma-delimited string is used to provide inputs. If you'd like to specify multiple notifications, then each should be a member in the list, like so:

```
notifications = ["Email,foo-alerts@example.com", "Slack,credentialId,channel"]
```

This will likely be changed in a future iteration of the provider. See SignalFX Docs for more information. For now, here are some example of how to configure each notification type:

» Email

```
notifications = ["Email,foo-alerts@bar.com"]
```

» Opsgenie

Note that the `credentialId` is the SignalFx-provided ID shown after setting up your Opsgenie integration. `Team` here is hardcoded as the `responderType` as that is the only acceptable type as per the API docs.

```
notifications = ["Opsgenie,credentialId,responderName,responderId,Team"]
```

» PagerDuty

```
notifications = ["PagerDuty,credentialId"]
```

» Slack

Exclude the # on the channel name!

```
notifications = ["Slack,credentialId,channel"]
```

» Team

Sends notifications to a team.

```
notifications = ["Team,teamId"]
```

» Team

Sends an email to every member of a team.

```
notifications = ["TeamEmail,teamId"]
```

» Webhook

```
notifications = ["Webhook,credentialId,secret,url"]
```

» Argument Reference

- `name` - (Required) Name of the detector.
- `program_text` - (Required) Signalflow program text for the detector. More info at https://developers.signalfx.com/docs/signalflow-overview.
- `description` - (Optional) Description of the detector.
- `max_delay` - (Optional) How long (in seconds) to wait for late datapoints. See https://signalfx-product-docs.readthedocs-hosted.com/en/latest/charts/chart-builder.html#delayed-datapoints for more info. Max value is `900` seconds (15 minutes).
- `show_data_markers` - (Optional) When `true`, markers will be drawn for each datapoint within the visualization. `false` by default.
- `show_event_lines` - (Optional) When `true`, the visualization will display a vertical line for each event trigger. `false` by default.
- `disable_sampling` - (Optional) When `false`, the visualization may sample the output timeseries rather than displaying them all. `false` by default.
- `time_range` - (Optional) From when to display data. SignalFx time syntax (e.g. `"-5m"`, `"-1h"`). Conflicts with `start_time` and `end_time`.
- `start_time` - (Optional) Seconds since epoch. Used for visualization. Conflicts with `time_range`.

- `end_time` - (Optional) Seconds since epoch. Used for visualization. Conflicts with `time_range`.
- `teams` - (Optional) Team IDs to associcate the detector to.
- `rule` - (Required) Set of rules used for alerting.
  - `detect_label` - (Required) A detect label which matches a detect label within `program_text`.
  - `severity` - (Required) The severity of the rule, must be one of: `"Critical"`, `"Major"`, `"Minor"`, `"Warning"`, `"Info"`.
  - `disabled` - (Optional) When true, notifications and events will not be generated for the detect label. `false` by default.
  - `notifications` - (Optional) List of strings specifying where notifications will be sent when an incident occurs. See https://developers.signalfx.com/detectors_reference.html#operation/Create%20Single%20Detector for more info.
  - `parameterized_body` - (Optional) Custom notification message body when an alert is triggered. See https://docs.signalfx.com/en/latest/detect-alert/set-up-detectors.html#about-detectors#alert-settings for more info.
  - `parameterized_subject` - (Optional) Custom notification message subject when an alert is triggered. See https://docs.signalfx.com/en/latest/detect-alert/set-up-detectors.html#about-detectors#alert-settings for more info.
  - `runbook_url` - (Optional) URL of page to consult when an alert is triggered. This can be used with custom notification messages.
  - `tip` - (Optional) Plain text suggested first course of action, such as a command line to execute. This can be used with custom notification messages.

Notes

It is highly recommended that you use both `max_delay` in your detector configuration and an `extrapolation` policy in your program text to reduce false positives/negatives.

`max_delay` allows SignalFx to continue with computation if there is a lag in receiving data points.

`extrapolation` allows you to specify how to handle missing data. An extrapolation policy can be added to individual signals by updating the data block in your `program_text`.

See https://signalfx-product-docs.readthedocs-hosted.com/en/latest/charts/chart-builder.html#delayed-datapoints for more info.

» Attributes Reference

The following attributes are exported:

- `id` - ID of the SignalFx detector

» Import

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

```
$ terraform import signalfx_detector.application_delay abc123
```

» Resource: signalfx_event_feed_chart

Displays a listing of events as a widget in a dashboard.

» Example Usage

```
resource "signalfx_event_feed_chart" "mynote0" {
    name = "Important Dashboard Note"
    description = "Lorem ipsum dolor sit amet"
    program_text = "A = events(eventType='Fart Testing').publish(label='A')"

    viz_options {
        label = "A"
        color = "orange"
    }
}
```

» Argument Reference

The following arguments are supported in the resource block:

- `name` - (Required) Name of the text note.
- `program_text` - (Required) Signalflow program text for the chart. More info at https://developers.signalfx.com/docs/signalflow-overview.
- `description` - (Optional) Description of the text note.
- `time_range` - (Optional) From when to display data. SignalFx time syntax (e.g. `"-5m"`, `"-1h"`). Conflicts with `start_time` and `end_time`.
- `start_time` - (Optional) Seconds since epoch. Used for visualization. Conflicts with `time_range`.
- `end_time` - (Optional) Seconds since epoch. Used for visualization. Conflicts with `time_range`.

» Resource: signalfx_gcp_integration

SignalFx GCP Integration

» Example Usage

```
resource "signalfx_gcp_integration" "gcp_myteam" {
    name = "GCP - My Team"
    enabled = true
    poll_rate = 300000
    services = ["compute"]
    project_service_keys {
        project_id = "gcp_project_id_1"
        project_key = "${file("/path/to/gcp_credentials_1.json")}"
    }
    project_service_keys {
        project_id = "gcp_project_id_2"
        project_key = "${file("/path/to/gcp_credentials_2.json")}"
    }
}
```

» Argument Reference

- `name` - (Required) Name of the integration.
- `enabled` - (Required) Whether the integration is enabled.
- `poll_rate` - (Required) GCP integration poll rate in milliseconds. Can be set to either 60000 or 300000 (1 minute or 5 minutes).
- `services` - (Optional) GCP service metrics to import. Can be an empty list, or not included, to import 'All services'.
- `project_service_keys` - (Required) GCP projects to add.

» Resource: signalfx_heatmap_chart

This chart type displays the specified plot in a heatmap fashion. This format is similar to the Infrastructure Navigator, with squares representing each source for the selected metric, and the color of each square representing the value range of the metric.

» Example Usage

```
resource "signalfx_heatmap_chart" "myheatmapchart0" {
```

```
    name = "CPU Total Idle - Heatmap"

    program_text = <<-EOF
        myfilters = filter("cluster_name", "prod") and filter("role", "search")
        data("cpu.total.idle", filter=myfilters).publish()
        EOF

    description = "Very cool Heatmap"

    disable_sampling = true
    sort_by = "+host"
    group_by = ["hostname", "host"]
    hide_timestamp = true
}
```

» Argument Reference

The following arguments are supported in the resource block:

- `name` - (Required) Name of the chart.
- `program_text` - (Required) Signalflow program text for the chart. More info at https://developers.signalfx.com/docs/signalflow-overview.
- `description` - (Optional) Description of the chart.
- `unit_prefix` - (Optional) Must be `"Metric"` or `"Binary"`. `"Metric"` by default.
- `minimum_resolution` - (Optional) The minimum resolution (in seconds) to use for computing the underlying program.
- `max_delay` - (Optional) How long (in seconds) to wait for late datapoints.
- `refresh_interval` - (Optional) How often (in seconds) to refresh the values of the heatmap.
- `disable_sampling` - (Optional) If `false`, samples a subset of the output MTS, which improves UI performance. `false` by default.
- `group_by` - (Optional) Properties to group by in the heatmap (in nesting order).
- `sort_by` - (Optional) The property to use when sorting the elements. Must be prepended with `+` for ascending or `-` for descending (e.g. `-foo`).
- `hide_timestamp` - (Optional) Whether to show the timestamp in the chart. `false` by default.
- `color_range` - (Optional. Conflict with color_scale) Values and color for the color range. Example: `color_range : { min : 0, max : 100, color : blue }`. Look at this link.
  - `min_value` - (Optional) The minimum value within the coloring range.
  - `max_value` - (Optional) The maximum value within the coloring range.

- `color` - (Required) The color range to use. Must be either gray, blue, navy, orange, yellow, magenta, purple, violet, lilac, green, aquamarine.

» Resource: signalfx_list_chart

This chart type displays current data values in a list format.

The name of each value in the chart reflects the name of the plot and any associated dimensions. We recommend you click the Pencil icon and give the plot a meaningful name, as in plot B below. Otherwise, just the raw metric name will be displayed on the chart, as in plot A below.

» Example Usage

```
resource "signalfx_list_chart" "mylistchart0" {
    name = "CPU Total Idle - List"

    program_text = <<-EOF
    myfilters = filter("cluster_name", "prod") and filter("role", "search")
    data("cpu.total.idle", filter=myfilters).publish()
    EOF

    description = "Very cool List Chart"

    color_by = "Metric"
    max_delay = 2
    disable_sampling = true
    refresh_interval = 1

    legend_options_fields {
      property = "collector"
      enabled  = false
    }

    legend_options_fields {
      property = "cluster_name"
      enabled = true
    }
    legend_options_fields {
      property = "role"
      enabled = true
    }
    legend_options_fields {
```

```
    property = "collector"
    enabled = false
  }
  legend_options_fields {
    property = "host"
    enabled = false
  }
  max_precision = 2
  sort_by = "-value"
}
```

» Argument Reference

The following arguments are supported in the resource block:

- `name` - (Required) Name of the chart.
- `program_text` - (Required) Signalflow program text for the chart. More info at https://developers.signalfx.com/docs/signalflow-overview.
- `description` - (Optional) Description of the chart.
- `unit_prefix` - (Optional) Must be `"Metric"` or `"Binary"`. `"Metric"` by default.
- `color_by` - (Optional) Must be `"Dimension"` or `"Metric"`. `"Dimension"` by default.
- `max_delay` - (Optional) How long (in seconds) to wait for late datapoints.
- `disable_sampling` - (Optional) If `false`, samples a subset of the output MTS, which improves UI performance. `false` by default.
- `refresh_interval` - (Optional) How often (in seconds) to refresh the values of the list.
- `viz_options` - (Optional) Plot-level customization options, associated with a publish statement.
  - `label` - (Required) Label used in the publish statement that displays the plot (metric time series data) you want to customize.
  - `display_name` - (Optional) Specifies an alternate value for the Plot Name column of the Data Table associated with the chart.
  - `color` - (Optional) Color to use : gray, blue, azure, navy, brown, orange, yellow, iris, magenta, pink, purple, violet, lilac, emerald, green, aquamarine.
  - `value_unit` - (Optional) A unit to attach to this plot. Units support automatic scaling (eg thousands of bytes will be displayed as kilobytes). Values values are `Bit, Kilobit, Megabit, Gigabit, Terabit, Petabit, Exabit, Zettabit, Yottabit, Byte, Kibibyte, Mebibyte, Gigibyte, Tebibyte, Pebibyte, Exbibyte, Zebibyte, Yobibyte, Nanosecond, Microsecond, Millisecond, Second, Minute, Hour, Day, Week`.
  - `value_prefix`, `value_suffix` - (Optional) Arbitrary prefix/suffix to

display with the value of this plot.

- `legend_fields_to_hide` - (Optional) List of properties that should not be displayed in the chart legend (i.e. dimension names). All the properties are visible by default. Deprecated, please use `legend_options_fields`.
- `legend_options_fields` - (Optional) List of property names and enabled flags that should be displayed in the data table for the chart, in the order provided. This option cannot be used with `legend_fields_to_hide`.
  - `property` The name of the property to display. Note the special values of `sf_metric` which shows the label of the time series `publish()` and `sf_originatingMetric` that shows the name of the metric for the time series being displayed.
  - `enabled` True or False depending on if you want the property to be shown or hidden.
- `max_precision` - (Optional) Maximum number of digits to display when rounding values up or down.
- `secondary_visualization` - (Optional) The type of secondary visualization. Can be `None`, `Radial`, `Linear`, or `Sparkline`. If unset, the SignalFx default is used (`Sparkline`).
- `color_scale` - (Optional. `color_by` must be `"Scale"`) Single color range including both the color to display for that range and the borders of the range. Example: `[{ gt = 60, color = "blue" }, { lte = 60, color = "yellow" }]`. Look at this link.
  - `gt` - (Optional) Indicates the lower threshold non-inclusive value for this range.
  - `gte` - (Optional) Indicates the lower threshold inclusive value for this range.
  - `lt` - (Optional) Indicates the upper threshold non-inculsive value for this range.
  - `lte` - (Optional) Indicates the upper threshold inclusive value for this range.
  - `color` - (Required) The color range to use. Must be either gray, blue, navy, orange, yellow, magenta, purple, violet, lilac, green, aquamarine.
- `sort_by` - (Optional) The property to use when sorting the elements. Use `value` if you want to sort by value. Must be prepended with `+` for ascending or `-` for descending (e.g. `-foo`). Note there are some special values for some of the options provided in the UX: `"value"` for Value, `"sf_originatingMetric"` for Metric, and `"sf_metric"` for plot.

» Resource: signalfx_opsgenie_integration

SignalFx Opsgenie integration.

» Example Usage

```
resource "signalfx_opsgenie_integration" "opgenie_myteam" {
    name = "Opsgenie - My Team"
    enabled = true
    api_key = "farts"
    api_url = "https://api.opsgenie.com"
}
```

» Argument Reference

- `name` - (Required) Name of the integration.
- `enabled` - (Required) Whether the integration is enabled.
- `api_key` - (Required) The API key
- `api_url` - (Optional) Opsgenie API URL. Will default to `https://api.opsgenie.com`. You might also want `https://api.eu.opsgenie.com`.

» Resource: signalfx_pagerduty_integration

SignalFx PagerDuty integrations

» Example Usage

```
resource "signalfx_pagerduty_integration" "pagerduty_myteam" {
    name = "PD - My Team"
    enabled = true
    api_key = "1234567890"
}
```

» Argument Reference

- `name` - (Required) Name of the integration.
- `enabled` - (Required) Whether the integration is enabled.
- `api_key` - (Required) PagerDuty API key.

» Resource: signalfx_slack_integration

SignalFx Slack integration.

» Example Usage

```
resource "signalfx_slack_integration" "slack_myteam" {
    name = "Slack - My Team"
    enabled = true
    webhook_url = "http://example.com"
}
```

» Argument Reference

- `name` - (Required) Name of the integration.
- `enabled` - (Required) Whether the integration is enabled.
- `webhook_url` - (Required) Slack incoming webhook URL.

» Resource: signalfx_org_token

Manage SignalFx org tokens.

» Example Usage

```
resource "signalfx_org_token" "myteamkey0" {
    name = "TeamIDKey"
    description = "My team's rad key"
    notifications = ["Email,foo-alerts@bar.com"]

    host_or_usage_limits {
      host_limit = 100
      host_notification_threshold = 90
      container_limit = 200
      container_notification_threshold = 180
      custom_metrics_limit = 1000
      custom_metrics_notification_threshold = 900
      high_res_metrics_limit = 1000
      high_res_metrics_notification_threshold = 900
    }
}
```

» Argument Reference

The following arguments are supported in the resource block:

- `name` - (Required) Name of the token.

- `description` - (Optional) Description of the token.
- `disabled` - (Optional) Flag that controls enabling the token. If set to `true`, the token is disabled, and you can't use it for authentication. Defaults to `false`.
- `notifications` - (Optional) Where to send notifications about this token's limits. Please consult the Notification Format laid out in detectors.
- `host_or_usage_limits` - (Optional) Specify Usage-based limits for this token.
  - `host_limit` - (Optional) Max number of hosts that can use this token
  - `host_notification_threshold` - (Optional) Notification threshold for hosts
  - `container_limit` - (Optional) Max number of Docker containers that can use this token
  - `container_notification_threshold` - (Optional) Notification threshold for Docker containers
  - `custom_metrics_limit` - (Optional) Max number of custom metrics that can be sent with this token
  - `custom_metrics_notification_threshold` - (Optional) Notification threshold for custom metrics
  - `high_res_metrics_limit` - (Optional) Max number of hi-res metrics that can be sent with this toke
  - `high_res_metrics_notification_threshold` - (Optional) Notification threshold for hi-res metrics
- `dpm_limits` (Optional) Specify DPM-based limits for this token.
  - `dpm_notification_threshold` - (Optional) DPM level at which SignalFx sends the notification for this token. If you don't specify a notification, SignalFx sends the generic notification.
  - `dpm_limit` - (Required) The datapoints per minute (dpm) limit for this token. If you exceed this limit, SignalFx sends out an alert.

» Resource: signalfx_single_value_chart

This chart type displays a single number in a large font, representing the current value of a single metric on a plot line.

If the time period is in the past, the number represents the value of the metric near the end of the time period.

» Example Usage

```
resource "signalfx_single_value_chart" "mysvchart0" {
    name = "CPU Total Idle - Single Value"

    program_text = <<-EOF
```

```
        myfilters = filter("cluster_name", "prod") and filter("role", "search")
        data("cpu.total.idle", filter=myfilters).publish()
        EOF

    description = "Very cool Single Value Chart"

    color_by = "Dimension"

    max_delay = 2
    refresh_interval = 1
    max_precision = 2
    is_timestamp_hidden = true
}
```

» Argument Reference

The following arguments are supported in the resource block:

- `name` - (Required) Name of the chart.
- `program_text` - (Required) Signalflow program text for the chart. More info at https://developers.signalfx.com/docs/signalflow-overview.
- `description` - (Optional) Description of the chart.
- `color_by` - (Optional) Must be `"Dimension"`, `"Scale"` or `"Metric"`. `"Dimension"` by default.
- `color_scale` - (Optional. `color_by` must be `"Scale"`) Single color range including both the color to display for that range and the borders of the range. Example: `[{ gt = 60, color = "blue" }, { lte = 60, color = "yellow" }]`. Look at this link.
  - `gt` - (Optional) Indicates the lower threshold non-inclusive value for this range.
  - `gte` - (Optional) Indicates the lower threshold inclusive value for this range.
  - `lt` - (Optional) Indicates the upper threshold non-inculsive value for this range.
  - `lte` - (Optional) Indicates the upper threshold inclusive value for this range.
  - `color` - (Required) The color range to use. Must be either gray, blue, navy, orange, yellow, magenta, purple, violet, lilac, green, aquamarine.
- `viz_options` - (Optional) Plot-level customization options, associated with a publish statement.
  - `label` - (Required) Label used in the publish statement that displays the plot (metric time series data) you want to customize.
  - `display_name` - (Optional) Specifies an alternate value for the Plot Name column of the Data Table associated with the chart.

  - `color` - (Optional) Color to use : gray, blue, azure, navy, brown, orange, yellow, iris, magenta, pink, purple, violet, lilac, emerald, green, aquamarine.
  - `value_unit` - (Optional) A unit to attach to this plot. Units support automatic scaling (eg thousands of bytes will be displayed as kilobytes). Values values are `Bit, Kilobit, Megabit, Gigabit, Terabit, Petabit, Exabit, Zettabit, Yottabit, Byte, Kibibyte, Mebibyte, Gigibyte, Tebibyte, Pebibyte, Exbibyte, Zebibyte, Yobibyte, Nanosecond, Microsecond, Millisecond, Second, Minute, Hour, Day, Week`.
  - `value_prefix`, `value_suffix` - (Optional) Arbitrary prefix/suffix to display with the value of this plot.
- `unit_prefix` - (Optional) Must be `"Metric"` or `"Binary"`. `"Metric"` by default.
- `max_delay` - (Optional) How long (in seconds) to wait for late datapoints
- `refresh_interval` - (Optional) How often (in seconds) to refresh the value.
- `max_precision` - (Optional) The maximum precision to for value displayed.
- `is_timestamp_hidden` - (Optional) Whether to hide the timestamp in the chart. `false` by default.
- `secondary_visualization` - (Optional) The type of secondary visualization. Can be `None`, `Radial`, `Linear`, or `Sparkline`. If unset, the SignalFx default is used (`None`).
- `show_spark_line` - (Optional) Whether to show a trend line below the current value. `false` by default.

» Resource: signalfx_text_chart

This special type of chart doesn't display any metric data. Rather, it lets you place a text note on the dashboard.

» Example Usage

```
resource "signalfx_text_chart" "mynote0" {
    name = "Important Dashboard Note"
    description = "Lorem ipsum dolor sit amet, laudem tibique iracundia at mea. Nam posse do

    markdown = <<-EOF
    1. First ordered list item
    2. Another item
      * Unordered sub-list.
    1. Actual numbers don't matter, just that it's a number
```

```
      1. Ordered sub-list
    4. And another item.

       You can have properly indented paragraphs within list items. Notice the blank line ab

       To have a line break without a paragraph, you will need to use two trailing spaces.
       Note that this line is separate, but within the same paragraph.
       (This is contrary to the typical GFM line break behaviour, where trailing spaces are

    * Unordered list can use asterisks
    - Or minuses
    + Or pluses
    EOF
}
```

» Argument Reference

The following arguments are supported in the resource block:

- `name` - (Required) Name of the text note.
- `markdown` - (Required) Markdown text to display.
- `description` - (Optional) Description of the text note.

» Resource: signalfx_time_chart

Provides a SignalFx time chart resource. This can be used to create and manage the different types of time charts.

Time charts display data points over a period of time.

» Example Usage

```
resource "signalfx_time_chart" "mychart0" {
    name = "CPU Total Idle"

    program_text = <<-EOF
        myfilters = filter("shc_name", "prod") and filter("role", "splunk_searchhead")
        data("cpu.total.idle", filter=myfilters).publish(label="CPU Idle")
        EOF

    time_range = 3600

    plot_type = "LineChart"
```

```
    show_data_markers = true

    legend_options_fields {
      property = "shc_name"
      enabled = true
    }
    legend_options_fields {
      property = "role"
      enabled = true
    }
    legend_options_fields {
      property = "collector"
      enabled = false
    }
    legend_options_fields {
      property = "prefix"
      enabled = false
    }
    legend_options_fields {
      property = "hostname"
      enabled = false
    }
    viz_options {
        label = "CPU Idle"
        axis = "left"
        color = "orange"
    }

    axis_left {
        label = "CPU Total Idle"
        low_watermark = 1000
    }
}
```

» Argument Reference

The following arguments are supported in the resource block:

- `name` - (Required) Name of the chart.
- `program_text` - (Required) Signalflow program text for the chart. More info at https://developers.signalfx.com/docs/signalflow-overview.
- `plot_type` - (Optional) The default plot display style for the visualization. Must be `"LineChart"`, `"AreaChart"`, `"ColumnChart"`, or `"Histogram"`. Default: `"LineChart"`.
- `description` - (Optional) Description of the chart.

- `axes_precision` - (Optional) Specifies the digits SignalFx displays for values plotted on the chart. Defaults to `3`.
- `unit_prefix` - (Optional) Must be `"Metric"` or `"Binary"`. `"Metric"` by default.
- `color_by` - (Optional) Must be `"Dimension"` or `"Metric"`. `"Dimension"` by default.
- `minimum_resolution` - (Optional) The minimum resolution (in seconds) to use for computing the underlying program.
- `max_delay` - (Optional) How long (in seconds) to wait for late datapoints.
- `timezone` - (Optional) A string denotes the geographic region associated with the time zone.
- `disable_sampling` - (Optional) If `false`, samples a subset of the output MTS, which improves UI performance. `false` by default
- `time_range` - (Optional) How many seconds ago from which to display data. For example, the last hour would be `3600`, etc. Conflicts with `start_time` and `end_time`.
- `start_time` - (Optional) Seconds since epoch. Used for visualization. Conflicts with `time_range`.
- `end_time` - (Optional) Seconds since epoch. Used for visualization. Conflicts with `time_range`.
- `axes_include_zero` - (Optional) Force the chart to display zero on the y-axes, even if none of the data is near zero.
- `axis_left` - (Optional) Set of axis options.
  - `label` - (Optional) Label of the left axis.
  - `min_value` - (Optional) The minimum value for the left axis.
  - `max_value` - (Optional) The maximum value for the left axis.
  - `high_watermark` - (Optional) A line to draw as a high watermark.
  - `high_watermark_label` - (Optional) A label to attach to the high watermark line.
  - `low_watermark` - (Optional) A line to draw as a low watermark.
  - `low_watermark_label` - (Optional) A label to attach to the low watermark line.
- `axis_right` - (Optional) Set of axis options.
  - `label` - (Optional) Label of the right axis.
  - `min_value` - (Optional) The minimum value for the right axis.
  - `max_value` - (Optional) The maximum value for the right axis.
  - `high_watermark` - (Optional) A line to draw as a high watermark.
  - `high_watermark_label` - (Optional) A label to attach to the high watermark line.
  - `low_watermark` - (Optional) A line to draw as a low watermark.
  - `low_watermark_label` - (Optional) A label to attach to the low watermark line.
- `viz_options` - (Optional) Plot-level customization options, associated with a publish statement.
  - `label` - (Required) Label used in the publish statement that displays the plot (metric time series data) you want to customize.

  - `display_name` - (Optional) Specifies an alternate value for the Plot Name column of the Data Table associated with the chart.
  - `color` - (Optional) Color to use : gray, blue, azure, navy, brown, orange, yellow, iris, magenta, pink, purple, violet, lilac, emerald, green, aquamarine.
  - `axis` - (Optional) Y-axis associated with values for this plot. Must be either `right` or `left`.
  - `plot_type` - (Optional) The visualization style to use. Must be `"LineChart"`, `"AreaChart"`, `"ColumnChart"`, or `"Histogram"`. Chart level `plot_type` by default.
  - `value_unit` - (Optional) A unit to attach to this plot. Units support automatic scaling (eg thousands of bytes will be displayed as kilobytes). Values values are `Bit, Kilobit, Megabit, Gigabit, Terabit, Petabit, Exabit, Zettabit, Yottabit, Byte, Kibibyte, Mebibyte, Gigibyte, Tebibyte, Pebibyte, Exbibyte, Zebibyte, Yobibyte, Nanosecond, Microsecond, Millisecond, Second, Minute, Hour, Day, Week`.
  - `value_prefix`, `value_suffix` - (Optional) Arbitrary prefix/suffix to display with the value of this plot.
  - `event_options` - (Optional) Event customization options, associated with a publish statement. You will need to use this to change settings for any `events(…)` statements you use.
    * `label` - (Required) Label used in the publish statement that displays the event query you want to customize.
    * `display_name` - (Optional) Specifies an alternate value for the Plot Name column of the Data Table associated with the chart.
    * `color` - (Optional) Color to use : gray, blue, azure, navy, brown, orange, yellow, iris, magenta, pink, purple, violet, lilac, emerald, green, aquamarine.
- `histogram_options` - (Optional) Only used when `plot_type` is `"Histogram"`. Histogram specific options.
  - `color_theme` - (Optional) Color to use : gray, blue, azure, navy, brown, orange, yellow, iris, magenta, pink, purple, violet, lilac, emerald, green, aquamarine, red, gold, greenyellow, chartreuse, jade
- `legend_fields_to_hide` - (Optional) List of properties that should not be displayed in the chart legend (i.e. dimension names). All the properties are visible by default. Deprecated, please use `legend_options_fields`.
- `legend_options_fields` - (Optional) List of property names and enabled flags that should be displayed in the data table for the chart, in the order provided. This option cannot be used with `legend_fields_to_hide`.
  - `property` The name of the property to display. Note the special values of `sf_metric` which shows the label of the time series `publish()` and `sf_originatingMetric` that shows the name of the metric for the time series being displayed.
  - `enabled` True or False depending on if you want the property to be shown or hidden.

- `on_chart_legend_dimension` - (Optional) Dimensions to show in the on-chart legend. On-chart legend is off unless a dimension is specified. Allowed: `"metric"`, `"plot_label"` and any dimension.
- `show_event_lines` - (Optional) Whether vertical highlight lines should be drawn in the visualizations at times when events occurred. `false` by default.
- `show_data_markers` - (Optional) Show markers (circles) for each datapoint used to draw line or area charts. `false` by default.
- `stacked` - (Optional) Whether area and bar charts in the visualization should be stacked. `false` by default.
- `timezone` - (Optional) Time zone that SignalFlow uses as the basis of calendar window transformation methods. For example, if you set ”timezone”: ”Europe/Paris” and then use the transformation sum(cycle=”week”, cycle_start=”Monday”) in your chart's SignalFlow program, the calendar window starts on Monday, Paris time. See the full list of timezones for more. `"UTC"` by default.