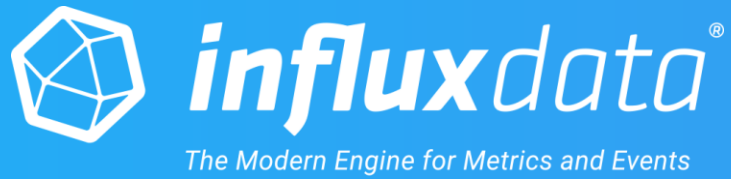
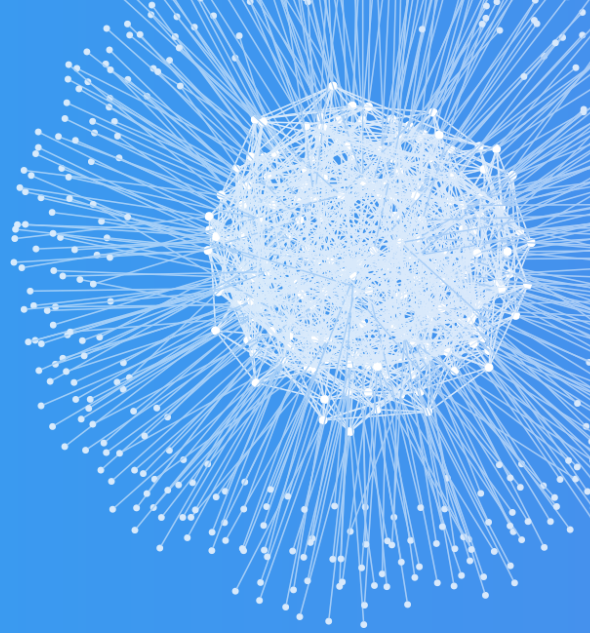


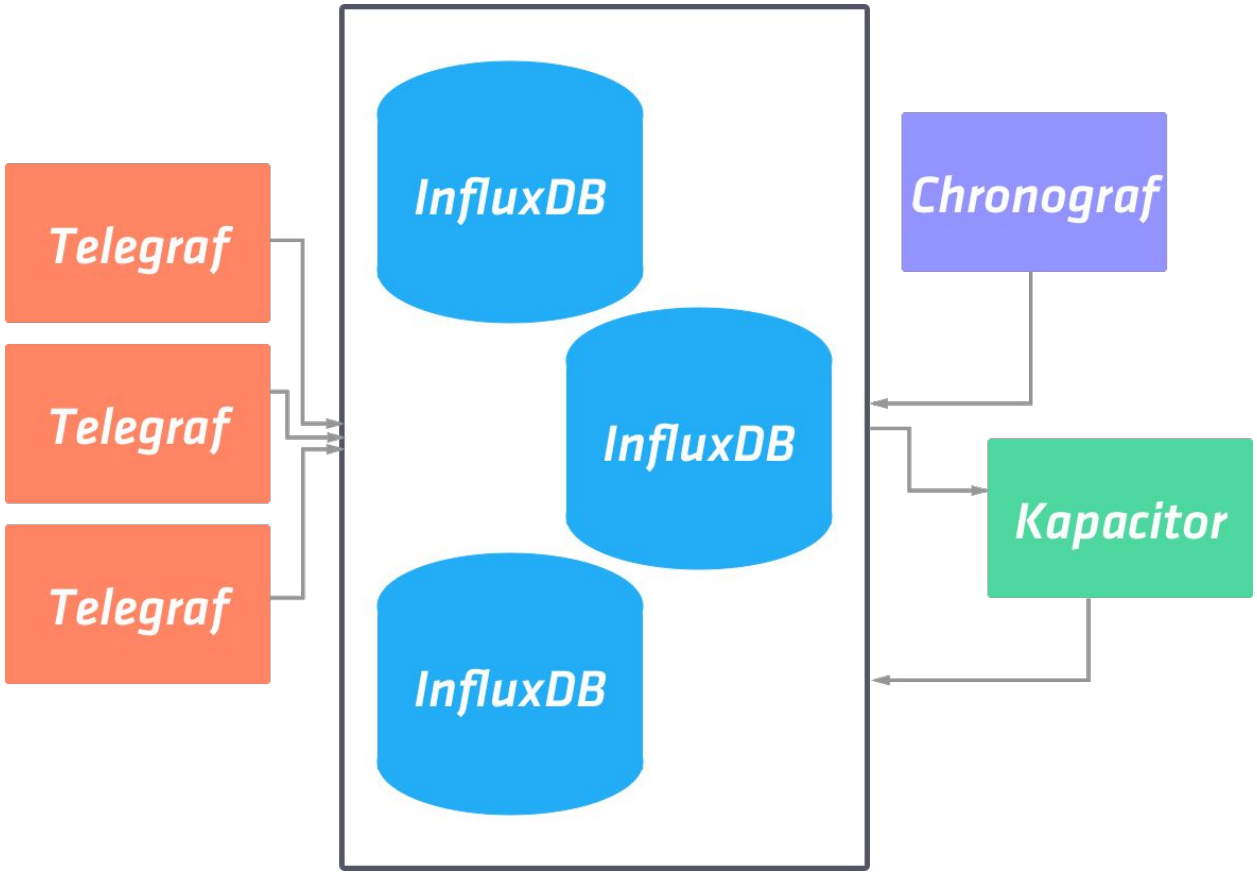


Kapacitor

Introduction to Kapacitor

Agenda

- What is Kapacitor
- Understand TICKscript
  - Quoting rules
- Understand Batch vs Stream
- Use Kapacitor for
  - Downsampling
  - Alerting
- Creating Templated Tasks
- Using Topics
- Debugging TICKscript
- Examples
  - Working with Telegraf Data



Introduction to Kapacitor

What is it?

- Native time-series processing engine
- Process stream and batch data from InfluxDB
- Generates Alerts on dynamic criteria
  - match metrics for patterns
  - compute statistical anomalies
  - perform specific actions
- Integrates with 15+ Alert Providers
  - OpsGenie
  - Sensu
  - PagerDuty
  - Slack
  - and more.

Install

Installation

- Download
  - GitHub
  - https://portal.influxdata.com/downloads
- Pick your favorite package
- Install

Kapacitor Design

Server Daemon (kapacitord)
- Config for inputs/outputs/services/ etc.

Tasks - units of work
- Defined by a TICKscript
- Stream or Batch
- DAG - pipeline

Templates -
- Building many instances of a templated task

CLI (kapacitor)
- Calls for HTTP API or server
- Not interactive

Recordings - saved data
- Useful for isolated testing

Topics & topics handlers
- Generic rules for all alters

Defining Tasks

TICKScript

```
var measurement = 'requests'

var data = stream
    |from()
        .measurement(measurement)
    |where(lambda: "is_up" == TRUE)
    |where(lambda: "my_field" > 10)
    |window()
         .period(5m)
         .every(5m)

// Count number of points in window
data
     |count('value')
       .as('the_count')

// Compute mean of data window
data
     |mean('value')
      .as('the_average')
```

- Chain invocation language
  - | chains together different nodes
  - . refers to specific attributes on a node
- Variables refer to values
  - Strings
  - Ints, Floats
  - Durations
  - Pipelines

TICKScript Syntax - Quoting Rules

```
// ' means the use the literal string value
var measurement = 'requests'

var data = stream
    |from()
        .measurement(measurement)
// " means use the reference value
    |where(lambda: "is_up" == TRUE)
    |where(lambda: "my_field" > 10)
    |window()
        .period(5m)
        .every(5m)

// ' means the use the literal string value
data
     |count('value')
       .as('the_count')
```

- Double Quotes
  - References data in lambda expression
- Single Quotes
  - Literal String value

Create a Stream TICKscript

```
// cpu.tick
stream
    |from()
        .measurement('cpu')
    |log()
```

- Logs all data from the measurement **cpu**

Create a Stream Task

```
// cpu.tick
stream
    |from()
        .measurement('cpu')
    |log()
```

```
$ kapacitor define cpu \
  -tick cpu.tick \
  -type stream   \
  -dbrp telegraf.autogen
```

Create a Stream Task

```
// cpu.tick
dbrp "telegraf"."autogen"
dbrp "telegraf"."not_autogen"


stream
    |from()
        .measurement('cpu')
        .database('telegraf')
        .retentionPolicy('not_autogen')
    |log()
```

```
$ kapacitor define cpu \
  -tick cpu.tick \
```

Show a Stream Task

```
$ kapacitor define cpu \
  -tick cpu.tick \
  -type stream   \
  -dbrp telegraf.autogen

$ kapacitor enable cpu
```

```
$ kapacitor show cpu
ID: cpu
Error:
Template:
Type: stream
Status: enabled
Executing: true
Created: 10 Oct 17 16:05 EDT
Modified: 10 Oct 17 16:05 EDT
LastEnabled: 01 Jan 01 00:00 UTC
Databases Retention Policies: ["telegraf"."autogen"]
TICKscript:
// cpu.tick
stream
    |from()
        .measurement('cpu')
    |log()

DOT:
digraph cpu {
stream0 -> from1;
from1 -> log2;
}
```

Create a More Interesting Stream TICKscript

```
// cpu.tick
stream
    |from()
        .measurement('cpu')
    |window()
        .period(5m)
        .every(1m)
    |mean('usage_user')
        .as('mean_usage_user')
    |log()
```

- Create 5m windows of data that emit every 1m
- Compute the average of the field **usage_user**
- Log the result

An even more interesting TICKscript

```
// cpu.tick
stream
    |from()
        .measurement('cpu')
    |where(lambda: "cpu" == 'cpu-total')
    |window()
        .period(5m)
        .every(1m)
    |mean('usage_user')
        .as('mean_usage_user')
    |log()
```

- Filter on the tag **cpu=cpu-total**
- Create 5m windows of data that emit every 1m
- Compute the average of the field **usage_user**
- Log the result

Adding Alerts

```
// cpu.tick
stream
    |from()
        .measurement('cpu')
    |where(lambda: "cpu" == 'cpu-total')
    |window()
        .period(5m)
        .every(1m)
    |mean('usage_user')
        .as('mean_usage_user')
    |alert()
        .crit(lambda: "mean_usage_user" >  80)
        .message('cpu usage high!')
        .slack()
          .channel('alerts')
        .email('oncall@example.com')
```

- Filter on the tag **cpu=cpu-total**
- Create 5m windows of data that emit every 1m
- Compute the average of the field **usage_user**
- Alert when **mean_usage_user > 80**
- Send alert to
    - Slack channel **alerts**
    - Email **oncall@example.com**

Create a Batch TICKscript

```
// batch_cpu.tick
batch
  |query('''
SELECT mean("usage_user") AS mean_usage_user
FROM "telegraf"."autogen"."cpu"
''')
    .period(5m)
    .every(1m)
  |log()
```

- Query 5m windows of data every 1m
- Compute the average of the field **usage_user**
- Log the result

Create a Batch Task

```
// batch_cpu.tick
batch
  |query('''
SELECT mean("usage_user") AS mean_usage_user
FROM "telegraf"."autogen"."cpu"
''')
    .period(5m)
    .every(1m)
   |log()
```

```
$ kapacitor define batch_cpu \
  -tick batch_cpu.tick \
  -type batch   \
  -dbrp telegraf.autogen
```

Create a Batch TICKscript

```
// batch_cpu.tick
batch
  |query('''
SELECT mean("usage_user") AS mean_usage_user
FROM "telegraf"."autogen"."cpu"
''')
    .period(5m)
    .every(1m)
  |alert()
    .crit(lambda: "mean_usage_user" >  80)
    .message('cpu usage high!')
    .slack()
      .channel('alerts')
    .email('oncall@example.com')
```

- Query 5m windows of data every 1m
- Compute the average of the field **usage_user**
- Alert when **mean_usage_user > 80**
- Send alert to
  - Slack channel **alerts**
  - Email **oncall@example.com**

Batching

- Queries InfluxDB periodically
- Does not buffer as much of data in RAM
- Places additional query load on InfluxDB

Streaming

- Writes are mirrored onto the InfluxDB instance
- Buffers all data in RAM
- Places additional write load on Kapacitor

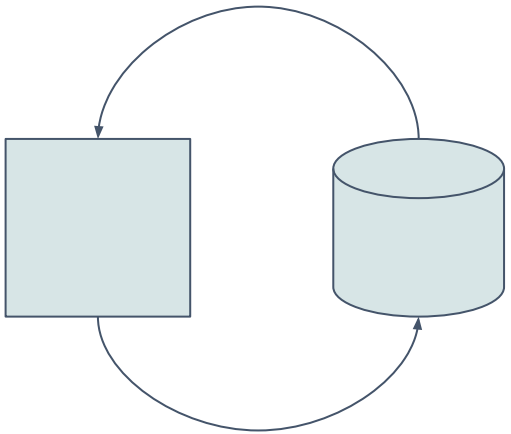
influxdata

Using Kapacitor for Downsampling

What is Downsampling?

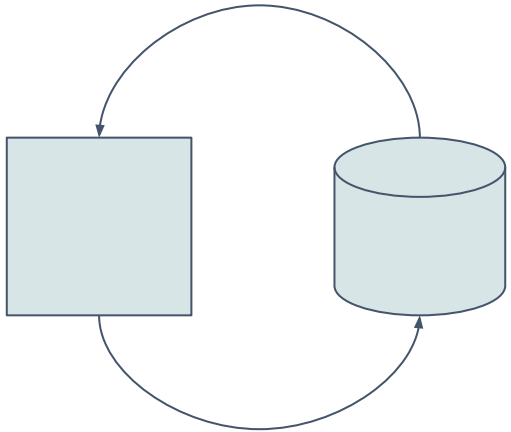
Downsampling is the process of reducing a sequence of points in a series to a single data point

- Computing the average, max, min, etc of a window of data for a particular series



Why Downsample?

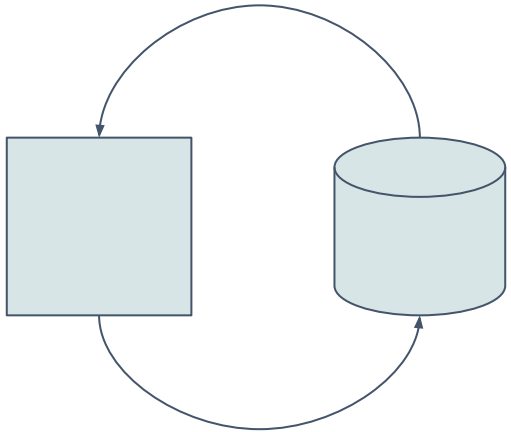
- Faster queries
- Store less data



Downsample using Kapacitor

Offload computation to a separate host

- Create a task that aggregates your data
  - Can be stream or batch depending on your use-case
- Writes data back into InfluxDB
  - Typically back into another retention policy



Example

```
// batch_cpu.tick
batch
  |query('''
SELECT mean("usage_user") AS usage_user
FROM "telegraf"."autogen"."cpu"
''')
    .period(5m)
    .every(5m)
  |influxDBOut()
    .database('telegraf')
    .retenionPolicy('5m')
    .tag('source', 'kapacitor')
```

- Downsample the data into 5m windows
- Store that data back into a the **5m** retention policy in the **telegraf** database

Defining Task Templates

Create a Stream Template

```
// generic.tick
var measurement string
var where_filter = lambda: TRUE
var groups = [*]
var field string
var crit lambda
var window = 5m

stream
    |from()
        .measurement(measurement)
        .where(where_filter)
        .groupBy(groups)
    |window()
        .period(window)
        .every(window)
    |mean(field)
    |alert()
         .crit(crit)
         .slack()
         .channel(slack_channel)
```

```
$ kapacitor define-template generic \
  -tick generic.tick \
  -type stream   \
```

Create a Stream Task from a Template

```
// generic_vars.json
{
  "measurement": {"type" : "string", "value" : "cpu" },
  "where_filter": {"type": "lambda", "value": "\"cpu\" == 'cpu-total'"},
  "groups": {"type": "list", "value": [{"type":"string", "value":"host"},{"type":"string", "value":"dc"}]},
  "field": {"type" : "string", "value" : "usage_idle" },
  "crit": {"type" : "lambda", "value" : "\"mean\" < 10.0" },
  "window": {"type" : "duration", "value" : "1m" },
}
```

```
$ kapacitor define generic_task \
  -template generic \
  -vars generic_vars.json
  -dbrp telegraf.autogen
```

Show a Templated Task

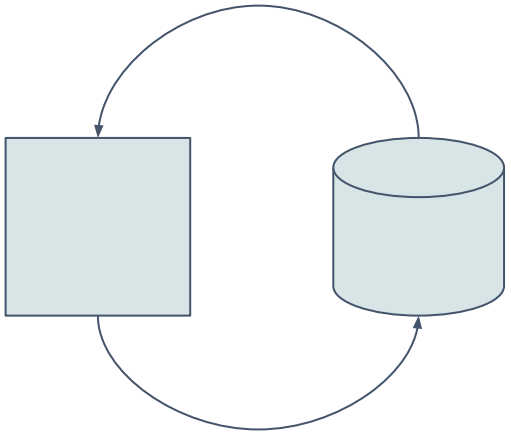
```
stream
    |from()
        .measurement(measurement)
        .where(where_filter)
        .groupBy(groups)
    |window()
        .period(window)
        .every(window)
    |mean(field)
    |alert()
        .warn(warn)
        .crit(crit)
        .slack()
        .channel(slack_channel)

Vars:
Name                            Type      Value
crit                            lambda    "mean" < 10.0
field                           string    usage_idle
groups                          list      [host, dc]
measurement                     string    cpu
warn                            lambda    "mean" < 10.0
where_filter                    lambda    "cpu" ==
'cpu-total'
window                          duration  1m0s
```

```
$  kapacitor show generic_task
```

TICKscript Examples

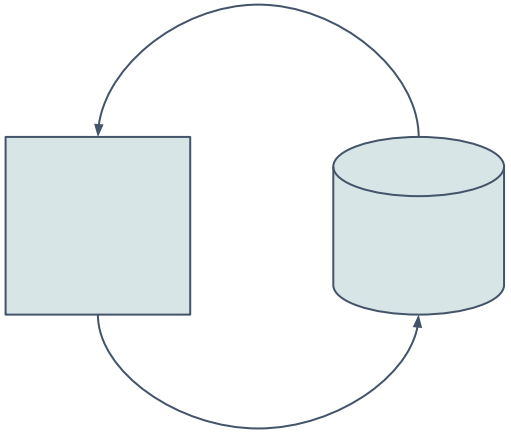
Example

- Compute the rolling average of the fields usage_user, usage_system, and usage_idle
- Create alerts for each



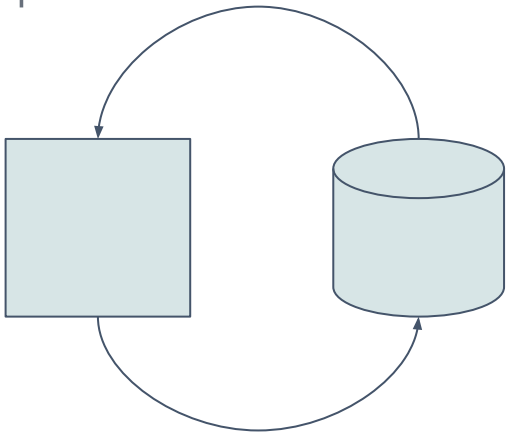
Example

- Compute the rolling average of the fields usage_user, usage_system, and usage_system
- Create alerts for each
- Join each of the computed values together



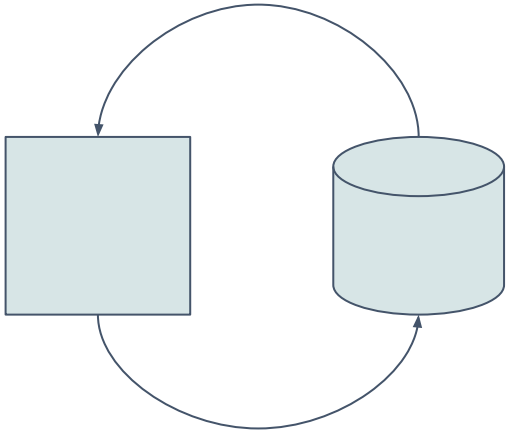
Example

- Create a derived metric under the measurement cpu that contains the cpu usage of a host
- Write the result back into a new database called telegraf_kapacitor



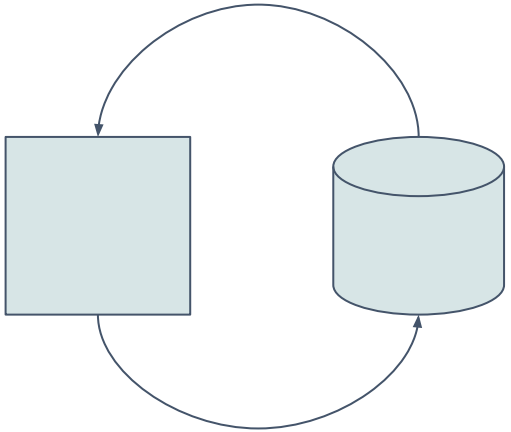
Example

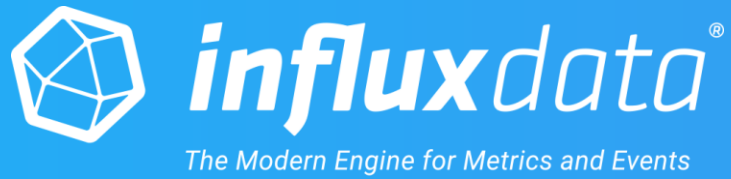
- Change a field into a tag

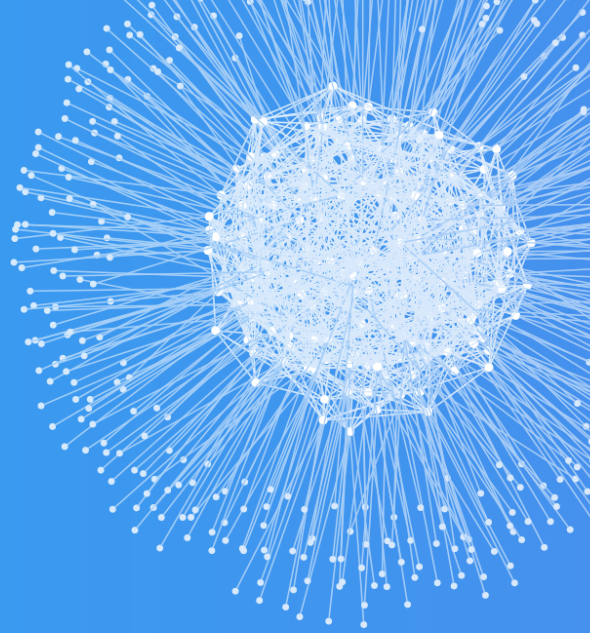


Example

- Gracefully handle data that is missing fields







Advanced Kapacitor: Alert Topic Handlers

Michael DeSa

Agenda

- Describe what *topic* is as it relates to Kapacitor
- Use *topics* effectively in Tickscript
- Explain what a *topic handlers* are used for
- Define their own *topic handlers*
- Use special purpose *topic handlers*

Kapacitor Topics

- Specified in a Tickscript
- Allows users to separate the handling of alerts from the task that generates them
- Gives users the ability to handle alerts in a more sophisticated way
- Follows a publish/subscribe pattern
  - Alerts are published to a topic
  - Handlers subscribe to a topic
- Handlers are scoped to a topic

Without Topics

- All alert handlers had to be defined in tickscript
- Adding a new handler required redefining the task
  - All current task state would be lost
- Couldn't have cascading alerts be triggered
- Simplistic condition matching

```
// example.tick

stream
    |from()
        .measurement('cpu')
        .groupBy(*)
    |alert()
        .warn(lambda: "usage_idle" < 20)
        .crit(lambda: "usage_idle" < 10)
        .slack()
          .channel('#alerts')
        .slack()
          .channel('#on_call')
```

With Topics: Creating/Using a topic

- Remove all handlers in the tickscript
- Add a topic handler

Note: Using a topic in a tickscript creates the topic.

```
// example.tick

stream
    |from()
        .measurement('cpu')
        .groupBy(*)
    |alert()
        .warn(lambda: "usage_idle" < 20)
        .crit(lambda: "usage_idle" < 10)
        .topic('basic')
```

With Topics: List topics

- Shows
  - Topic Name - ID
  - Current Alert Level - Level
  - Number of Alerts Published - Collected

```
$ kapacitor list topics
ID    Level     Collected
basic OK              245
```

With Topics: Show Topic

- Shows
  - Topic Name - ID
  - Current Alert Level - Level
  - Number of Alerts Published - Collected
  - Sample of Collected Events - Events

```
$ kapacitor show-topic basic
ID: basic
Level: OK
Collected: 247
Handlers: []
Events:
Event Level Message  Date
cpu   OK    cpu ...  05 Jul 17 20:55 EDT
cpu   OK    cpu ...  05 Jul 17 20:55 EDT
cpu   OK    cpu ...  05 Jul 17 20:55 EDT
cpu   OK    cpu ...  05 Jul 17 20:55 EDT
cpu   OK    cpu ...  05 Jul 17 20:55 EDT
cpu   OK    cpu ...  05 Jul 17 21:05 EDT
cpu   OK    cpu ...  05 Jul 17 21:05 EDT
cpu   OK    cpu ...  05 Jul 17 21:05 EDT
cpu   OK    cpu ...  05 Jul 17 21:10 EDT
```

Creating Topic Handlers

- Defining a topic handlers requires:
  - Topic Name
  - Handler Name
  - yaml File
- Yaml file specifies
  - Type of handler - kind
    - Typically the property method name from tickscript (e.g. slack, post, pagerduty, etc)
  - Required Parameters - options
    - Typically the property methods from tickscript

```
// alerts.yaml

kind: slack
topic: basic
id: alerts-channel
options:
  channel: "#alerts"
```

```
// on_call.yaml

kind: slack
topic: basic
id: on-call-chan
options:
  channel: "#on_call"
```

```
$ kapacitor define-topic-handler alerts.yaml

$ kapacitor define-topic-handler on_call.yaml
```

With Topics: List topics handlers

- Shows
  - Topic Name - Topic
  - Handler Name - ID
  - Type of handler - Kind

```
$ kapacitor list topic-handlers
Topic      ID             Kind
basic      alerts-channel slack
basic      on-call-chan   slack
```

With Topics: Show Topic Handler

- Shows
  - Handler Name - ID
  - Handler Topic - Topic
  - Type of handler - Kind
  - Matching logic for alerts - Match
  - Handler Options - Options

```
$ kapacitor show-topic-handler basic
alerts-channel
ID: alerts-channel
Topic: basic
Kind: slack
Match:
Options: {"channel":"#alerts"}
```

```
$ kapacitor show-topic-handler basic
on-call-chan
ID: on-call-chan
Topic: basic
Kind: slack
Match:
Options: {"channel":"#on_call"}
```

With Topics: Show Topic (again)

- Shows
  - Topic Name - ID
  - Current Alert Level - Level
  - Number of Alerts Published - Collected
  - Current topic handlers - Handlers
  - Sample of Collected Events - Events

```
$ kapacitor show-topic basic
ID: basic
Level: OK
Collected: 285
Handlers: [alerts-channel, on-call-chan]
Events:
Event Level Message  Date
cpu   OK    cpu ...  05 Jul 17 20:55 EDT
cpu   OK    cpu ...  05 Jul 17 20:55 EDT
cpu   OK    cpu ...  05 Jul 17 20:55 EDT
cpu   OK    cpu ...  05 Jul 17 20:55 EDT
cpu   OK    cpu ...  05 Jul 17 20:55 EDT
cpu   OK    cpu ...  05 Jul 17 21:05 EDT
cpu   OK    cpu ...  05 Jul 17 21:05 EDT
cpu   OK    cpu ...  05 Jul 17 21:05 EDT
cpu   OK    cpu ...  05 Jul 17 21:10 EDT
```

Chaining Topics using Handlers

- Topics can be chained together using a publish action.
  - This allows you to further group your alerts into various topics.

```
// chain.yaml

kind: publish
topic: basic
id: chain
options:
  topics:
    - ops_team
```

```
$ kapacitor define-topic-handler chain.yaml
```

Using Match Conditions in Handlers

- Conditions for matching a handle may be set in the match section of the yaml file
- Must be boolean expression
- Functions
  - Any built-in Kapacitor function
  - changed()
  - level()
  - name()
  - taskName()
  - duration()

```
// chain.yaml

kind: publish
topic: basic
id: chain
match: changed() == TRUE
options:
  topics:
    - ops_team
```

```
$ kapacitor define-topic-handler chain.yaml
```

With Topics: Show Topic Handler

- Shows
  - Handler Name - ID
  - Handler Topic - Topic
  - Type of handler - Kind
  - Matching logic for alerts - Match
  - Handler Options - Options

```
$ kapacitor show-topic-handler basic chain
ID: chain
Topic: basic
Kind: publish
Match: changed() == TRUE
Options: {"topics":["ops_team"]}
```

Summary

- Kapacitor users can utilize **topics** instead of explicitly handling alerts in tickscript
  - Requires use of topic handlers
- Handlers are scoped to a topic
  - Typically referred to as **topic handlers**
- Topics may be chained together using the **publish** handler
- Handlers have additional matching logic to allow for more sophisticated alert event handling
- For more information visit the documentation http://docs.influxdata.com/kapacitor/v1.3/guides/using_alert_topics/

Questions?

Thank you