

CUSTOM YOUTUBE PLUGiN

Project Objective

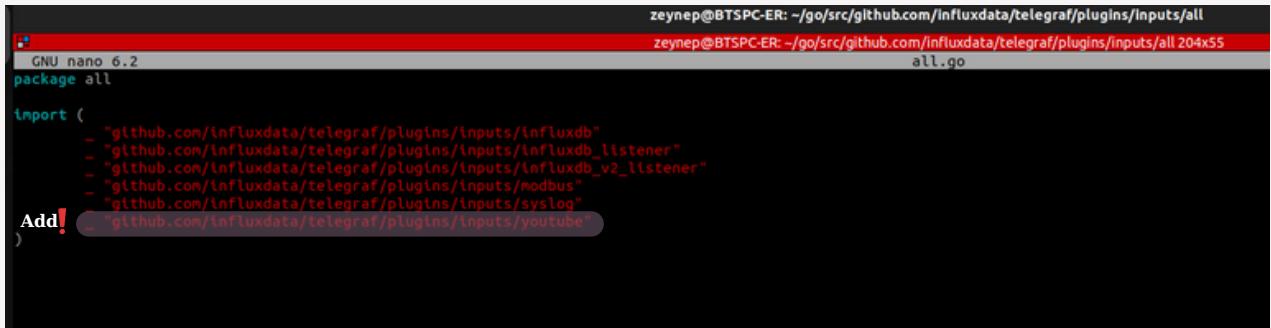
The goal of this project is to automatically collect and analyze key statistics from YouTube channels. Using the YouTube Data API v3, our Telegraf plugin gathers the following data for specified channels:

- Subscriber Count
- Video Count
- Total View Count
- Channel Country
- Channel Name
- Channel Creation Date

```
-GO youtube.go > ...
```

```
1 package youtube
2
3 import [
4     "context"
5     "fmt"
6     "strings"
7     "time"
8
9     "github.com/influxdata/telegraf"
10    "github.com/influxdata/telegraf/plugins/inputs"
11    "google.golang.org/api/option"
12    "google.golang.org/api/youtube/v3"
13 ]
14
15 type Youtube struct {
16     Channels      []string `toml:"channels"`
17     APIKey        string   `toml:"api_key"`
18     youtubeService *youtube.Service
19 }
20
21 const YoutubConfig = `
22     ## List of channels to monitor
23     channels = ["UCLA_DiR1FfKNvjuUpBHmylQ"]
24     ## Google API Key
25     # api_key = ""
26
27
28 func (s *Youtube) SampleConfig() string {
29     return YoutubConfig
30 }
31
32 func (s *Youtube) Description() string {
33     return "Gather youtube channel information from Youtube"
34 }
35
36 func (s *Youtube) createYoutubeService(ctx context.Context) (*youtube.Service, error) {
37     return youtube.NewService(ctx, option.WithAPIKey(s.APIKey))
38 }
39
40 func (s *Youtube) Gather(acc telegraf.Accumulator) error {
41     ctx := context.Background()
42
43     if s.youtubeService == nil {
44         service, err := s.createYoutubeService(ctx)
45         if err != nil {
46             return err
47         }
48         s.youtubeService = service
49     }
50
51     part := strings.Join([]string{"snippet", "statistics"}, ",")
52     call := s.youtubeService.Channels.List([]string(part)).Id(s.Channels...).MaxResults(50)
53
54     resp, err := call.Do()
55     if err != nil {
56         return err
57     }
58
59     now := time.Now()
60
61     for _, item := range resp.Items {
62         tags := getTags(item)
63         fields := getFields(item)
64
65         acc.AddFields("youtube_channel", fields, tags, now)
66     }
67
68     return nil
69 }
70
71 func getTags(channelInfo *youtube.Channel) map[string]string {
72     fmt.Println("channelInfo", channelInfo)
73     return map[string]string{
74         "id":    channelInfo.Id,
75         "title": channelInfo.Snippet.Title,
76     }
77 }
78
79 func getFields(channelInfo *youtube.Channel) map[string]interface{} {
80     return map[string]interface{}{
81         "subscribers": channelInfo.Statistics.SubscriberCount,
82         "videos":      channelInfo.Statistics.VideoCount,
83         "views":       channelInfo.Statistics.ViewCount,
84         "country":     channelInfo.Snippet.Country,
85         "channelName": channelInfo.Snippet.Title,
86         "createdon":   channelInfo.Snippet.PublishedAt,
87     }
88 }
89
90 func init() {
91     inputs.Add("youtube", func() telegraf.Input {
92         return &Youtube{}
93     })
94 }
```

2. Edit All.go File



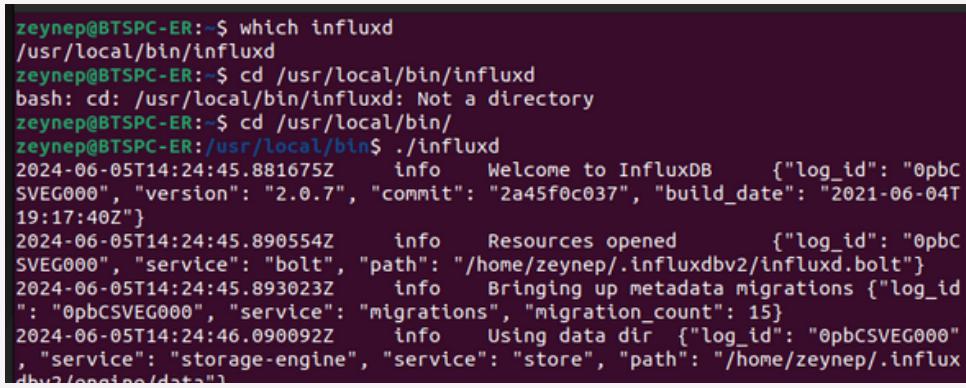
```
GNU nano 6.2
zeynep@BTSPC-ER: ~/go/src/github.com/influxdata/telegraf/plugins/inputs/all
zeynep@BTSPC-ER: ~/go/src/github.com/influxdata/telegraf/plugins/inputs/all 20x55
package all

import (
    "github.com/influxdata/telegraf/plugins/inputs/influxdb"
    "github.com/influxdata/telegraf/plugins/inputs/influxdb_listener"
    "github.com/influxdata/telegraf/plugins/inputs/influxdb_v2_listener"
    "github.com/influxdata/telegraf/plugins/inputs/modbus"
    "github.com/influxdata/telegraf/plugins/inputs/syslog"
)
Add! "github.com/influxdata/telegraf/plugins/inputs/youtube"
```

3. Create telegraf binary file

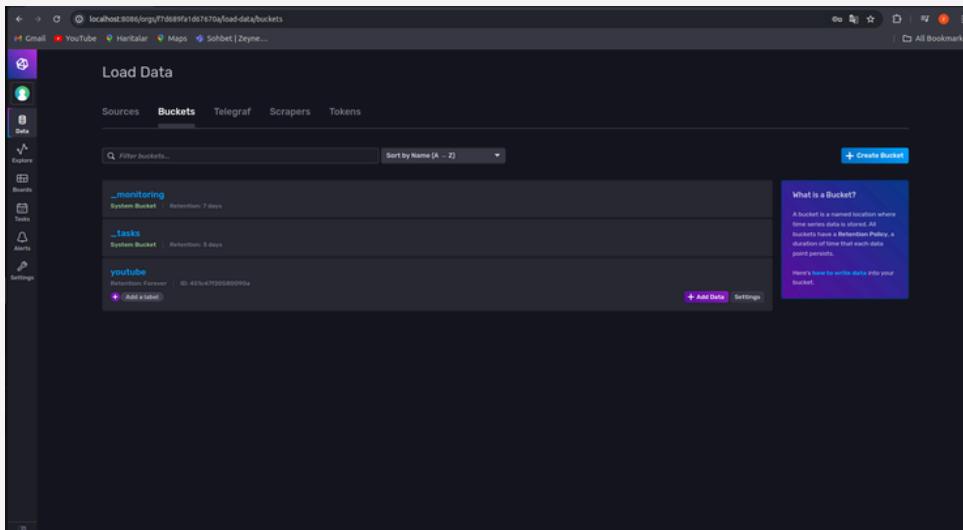
```
zeynep@BTSPC-ER:~/go/src/telegraf$ make telegraf
```

4. Start Influxdb



```
zeynep@BTSPC-ER:~$ which influxd
/usr/local/bin/influxd
zeynep@BTSPC-ER:~$ cd /usr/local/bin/influxd
bash: cd: /usr/local/bin/influxd: Not a directory
zeynep@BTSPC-ER:~$ cd /usr/local/bin/
zeynep@BTSPC-ER:/usr/local/bin$ ./influxd
2024-06-05T14:24:45.881675Z      info      Welcome to InfluxDB      {"log_id": "0pbCSVEG000", "version": "2.0.7", "commit": "2a45f0c037", "build_date": "2021-06-04T19:17:40Z"}
2024-06-05T14:24:45.890554Z      info      Resources opened      {"log_id": "0pbCSVEG000", "service": "bolt", "path": "/home/zeynep/.influxdbv2/influxd.bolt"}
2024-06-05T14:24:45.893023Z      info      Bringing up metadata migrations {"log_id": "0pbCSVEG000", "service": "migrations", "migration_count": 15}
2024-06-05T14:24:46.090092Z      info      Using data dir {"log_id": "0pbCSVEG000", "service": "storage-engine", "service": "store", "path": "/home/zeynep/.influxdbv2/engine/data"}
```

5. Create Youtube Bucket



6.Run token commands

```
export INFLUX_TOKEN=o-qItcD-a8YntwdHZF3Xz3oWX1LWnTzTVIDN5ObV0EqTERbkClhR8ZsUPJDj2XoFYVffa3fDMdXreI55XSD0Uw==
```

```
telegraf --config http://localhost:8086/api/v2/telegrafs/0d2632a78519b000
```

7.Create Youtube conf. File

The screenshot shows the Grafana Load Data interface. On the left sidebar, there are tabs for Sources, Buckets, Telegraf, Scrapers, and Tokens. The Telegraf tab is selected. In the main content area, there is a search bar labeled "Filter telegraf configurations..." and a dropdown menu set to "Sort by Name (A - Z)". Below the search bar, there is a "Telegraf Output Plugin" button and a "+ Create Configuration" button. A modal window titled "Edit Telegraf Configuration" is open, displaying the configuration code for the youtube plugin. The configuration includes settings like `interval` (set to "10s"), `round_interval` (set to "true"), and `metric_batch_size` (set to 1000). The modal also contains a "Download Config" button and a "Save Changes" button at the bottom. To the right of the modal, there is a "What is Telegraf?" help section.

```
1 # Configuration for telegraf agent
2 [agent]
3   ## Default data collection interval for all inputs
4   interval = "10s"
5   ## Set collection interval to "interval"
6   ## i.e. if interval="10s" then always collect on .00, .10, .20, etc.
7   round_interval = true
8
9   ## Telegraf will send metrics to outputs in batches of at most
10  ## metric_batch_size metrics.
11  metric_batch_size = 1000
12
13  ## For failed writes, telegraf will cache metric_buffer_limit metrics for each
14  ## output, and will flush this buffer on a successful write. Oldest metrics
15  ## are dropped first when this buffer fills.
16  ## This can be used to avoid many plugins querying things like sysfs at the
17  ## same time, which may have a measurable effect on the system.
18  metric_buffer_limit = 10000
19
20  ## Collection jitter is used to jitter the collection by a random amount.
21  ## Each plugin will sleep for a random time within jitter before collecting.
22  ## This can be used to avoid many plugins querying things like sysfs at the
23  ## same time, which may have a measurable effect on the system.
24  collection_jitter = "0s"
25
26  ## Default flushing interval for all outputs. Maximum flush_interval will be
27  ## flush_interval + flush_jitter
28  ## Jitter the flush interval by a random amount. This is primarily to avoid
29  ## contention for resources when using a large number of telegraf instances.
30  ## i.e. a jitter of 5s and interval 10s means flushes will happen every 10-15s
31  flush_jitter = "0s"
32
```

8. Google cloud/Youtube Data API v3 Key

The image displays six screenshots from the Google Cloud Platform (GCP) console, arranged in a 3x2 grid. Each screenshot shows a different step in the process of creating and managing a key for the YouTube Data API v3.

- Screenshot 1:** Shows the main GCP dashboard for "My First Project". The "APIs & Services" section is highlighted, showing the "Enabled APIs & services" list. A tooltip indicates that data is available for the selected time frame.
- Screenshot 2:** Shows the "APIs & Services" page with the "Enabled APIs & services" tab selected. It lists "YouTube Data API v3" as enabled. A tooltip indicates that no data is available for the selected time frame.
- Screenshot 3:** Shows the "API Library" page. A search bar at the top has "youtube" typed into it. Below the search bar, "youtube data api v3" is listed in the results. A tooltip indicates that no data is available for the selected time frame.
- Screenshot 4:** Shows the "Product details" page for the "YouTube Data API v3". It includes sections for "Overview", "Additional details", and "Tutorials and documentation".
- Screenshot 5:** Shows the main GCP dashboard again, but this time the "Credentials" section under "APIs & Services" is highlighted. A tooltip indicates that no data is available for the selected time frame.
- Screenshot 6:** Shows the "Credentials" page. It lists "API key" as the selected credential type. Other options like "OAuth client ID" and "Service account" are also shown. A tooltip indicates that no OAuth clients are displayed.

9.Edit Telegraf Conf.

```
youtube.conf
~/Downloads
```

```
## Default flushing interval for all outputs. Maximum flush_interval will be
## flush_interval + flush_jitter
flush_interval = "10s"
## Jitter the flush interval by a random amount. This is primarily to avoid
## large write spikes for users running a large number of telegraf instances.
## ie, a jitter of 5s and interval 10s means flushes will happen every 10-15s
flush_jitter = "0s"

## By default or when set to "0s", precision will be set to the same
## timestamp order as the collection interval, with the maximum being 1s.
## ie, when interval = "10s", precision will be "1s"
## when interval = "250ms", precision will be "1ms"
## Precision will NOT be used for service inputs. It is up to each individual
## service input to set the timestamp at the appropriate precision.
## Valid time units are "ns", "us" (or "μs"), "ms", "s".
precision = ""

## Logging configuration:
## Run telegraf with debug log messages.
debug = false
## Run telegraf in quiet mode (error log messages only).
quiet = false
## Specify the log file name. The empty string means to log to stderr.
logfile = ""

## Override default hostname, if empty use os.Hostname()
hostname = ""
## If set to true, do no set the "host" tag in the telegraf agent.
omit_hostname = false
[[outputs.influxdb_v2]]
## The URLs of the InfluxDB cluster nodes.
##
## Multiple URLs can be specified for a single cluster, only ONE of the
## urls will be written to each interval.
## urls exp: http://127.0.0.1:8086
urls = ["http://localhost:8086"]

## Token for authentication.
token = "o-qItcD-a8YntwdHZF3Xz3oWX1LWnTzTVIDN5ObV0EqTERbkClhR8ZsUPJDj2XoFYVffa3fDMdXreI55XSD0Uw=="

## Organization is the name of the organization you wish to write to; must exist.
organization = "BTS"

## Destination bucket to write into.
bucket = "youtube"
[[inputs.youtube]]
channels = [
    "UCLA_DiR1FfKNvjuUpBHmylQ",
    "UCuJW0p17WofTPFqJZPz8jYg",
    "UCfaSK6K7hB3rAH7MLh8o6pw"
]
api_key = "AIzaSyA1rfp6PC-qMkYC0S6ixU0tK0d8mr5GW24"
```

Influx Token

Youtube channels ID:
Nasa, Hubble, SpaceX

Youtube Data API V3 Key

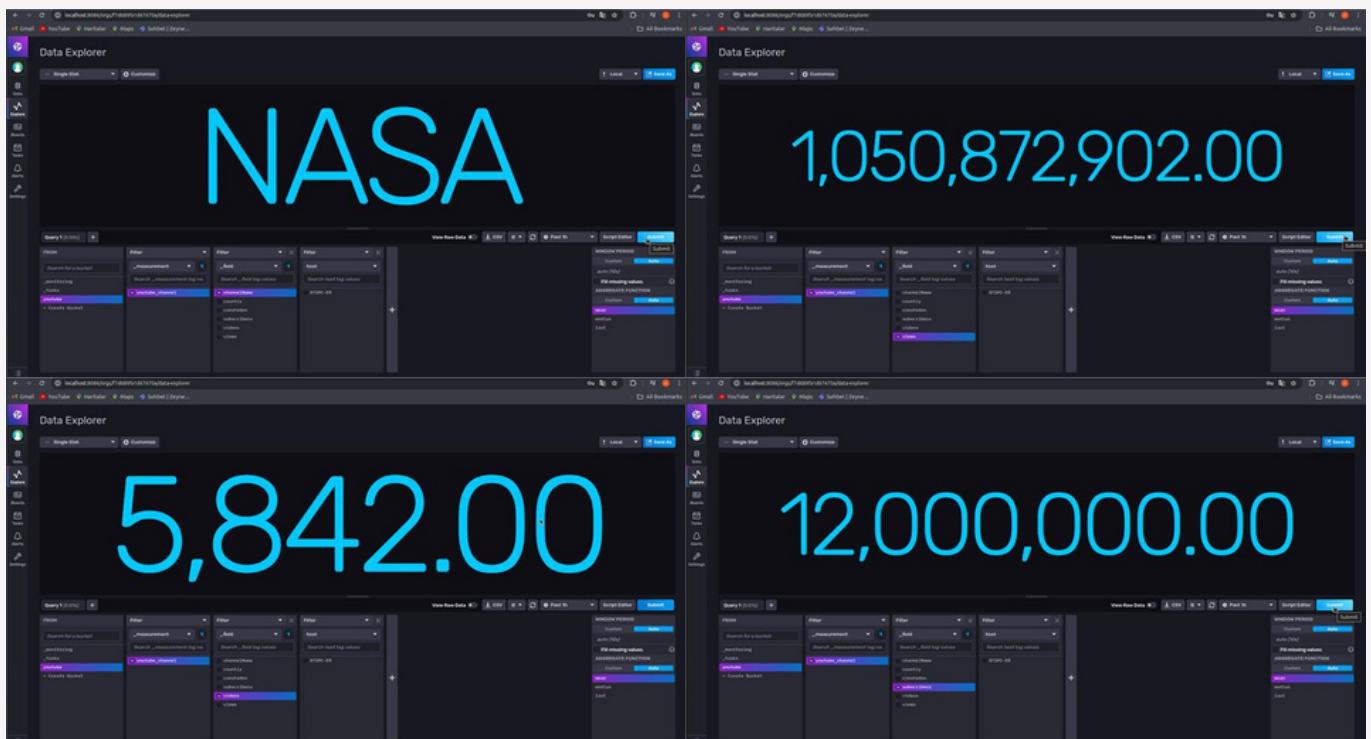
Plain Text ▾ Tab Width: 8 ▾ Ln 65, Col 1 ▾ INS

10. Start Telegraf conf.

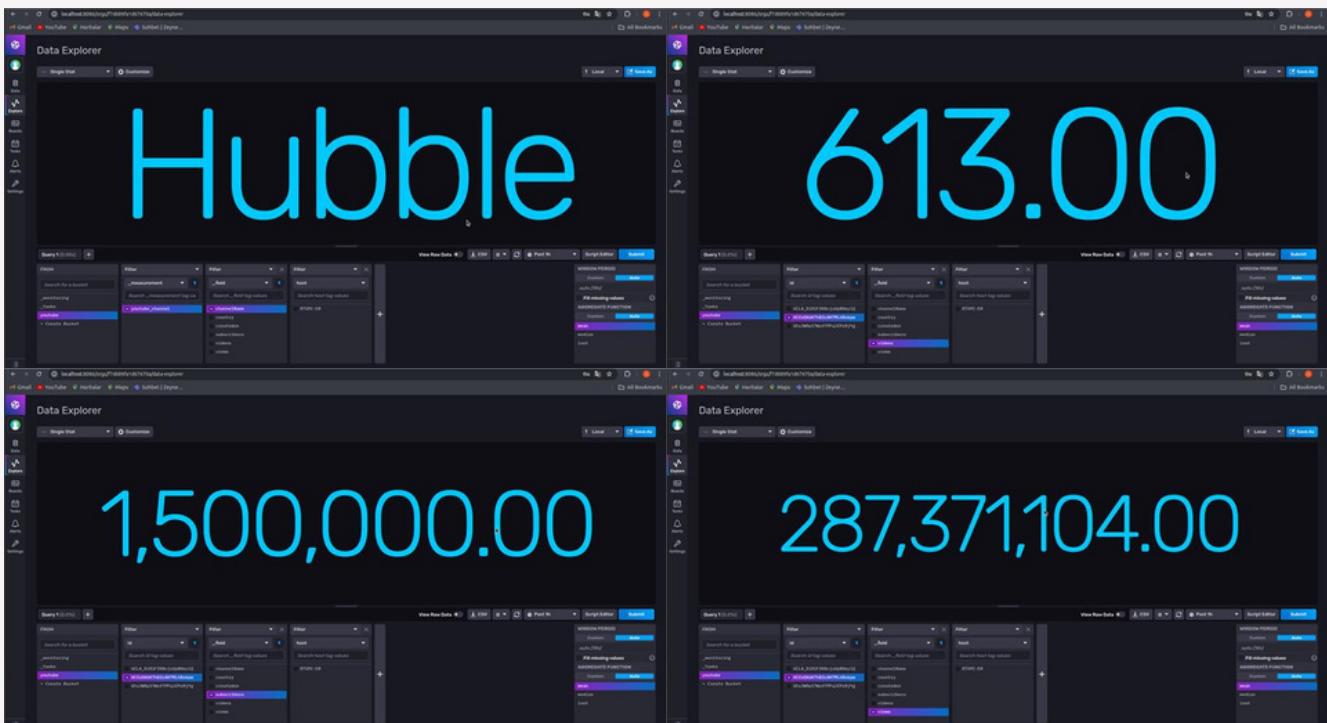
```
zeynep@BTSPC-ER:~/go/src/github.com/influxdata/telegraf$ ./telegraf --config /home/zeynep/Downloads/youtube.conf
2024-06-05T19:25:20Z I! Loading config: /home/zeynep/Downloads/youtube.conf
2024-06-05T19:25:20Z I! Starting Telegraf 1.31.0-c2a67ecd brought to you by InfluxData the makers of InfluxDB
2024-06-05T19:25:20Z I! Available plugins: 235 inputs, 9 aggregators, 32 processors, 26 parsers, 60 outputs, 6 secret-stores
2024-06-05T19:25:20Z I! Loaded inputs: youtube
2024-06-05T19:25:20Z I! Loaded aggregators:
2024-06-05T19:25:20Z I! Loaded processors:
2024-06-05T19:25:20Z I! Loaded secretstores:
2024-06-05T19:25:20Z I! Loaded outputs: influxdb_v2
2024-06-05T19:25:20Z I! Tags enabled: host=BTSPC-ER
2024-06-05T19:25:20Z I! [agent] Config: Interval:10s, Quiet:false, Hostname:"BTSPC-ER", Flush Interval:10s
channelInfo &{<nill> <nill> <nill> <nill> <nill> Z1IfEEX5Kc-bdq38rDpmfqNrH68 UCuJW0p17WoftTPFqJZPz8jYg youtube#channel map[] 0xc00298ac80 0xc002993200 <nill> <nill> {0 map[]} [] []}
channelInfo &{<nill> <nill> <nill> <nill> TxM_PgOV126yDBr6ZxQaUjxH-9w UCfaSK6K7hB3rAH7MLh8o6pw youtube#channel map[] 0xc00298ad20 0xc0029932c0 <nill> <nill> {0 map[]} [] []}
channelInfo &{<nill> <nill> <nill> <nill> U7e20Sd0Rcc0ReuCEM gaUhNvV0 UCLA D1R1EfkNy4uIeBHuwoL youtube#channel map[] 0xc00298ad20 0xc0029932c0 <nill> <nill> {0 map[]} [] []}
```

11. Complete Project

- NASA



- **Hubble**



- **SpaceX**

