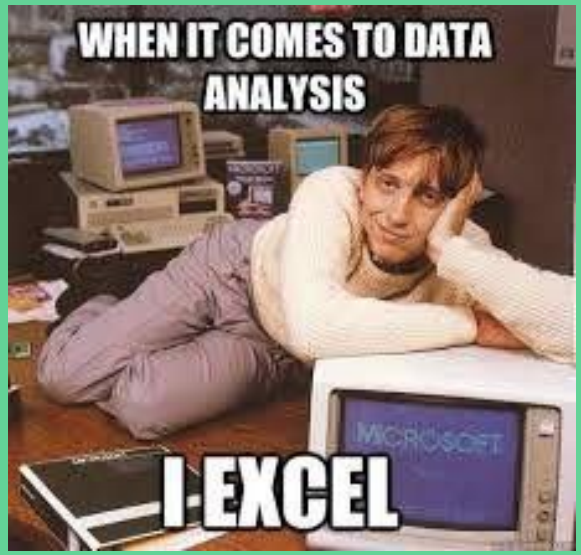
Wireless Analysis with Kismet and ELK

Chase Peterson
@developingchase
github.com/developingchase/wawkelk

I want to understand the wireless environment around me --

1) Capture all the RFs with Kismet
2) Dive deeper - Spatial, temporal, trend analysis



Overview

Goal: Present lessons learned and a quick start guide to help others and generate interest and involvement

1. Kismet Collection
2. Elasticsearch
3. Logstash
4. Python
5. Kibana
6. Analysis

Kismet

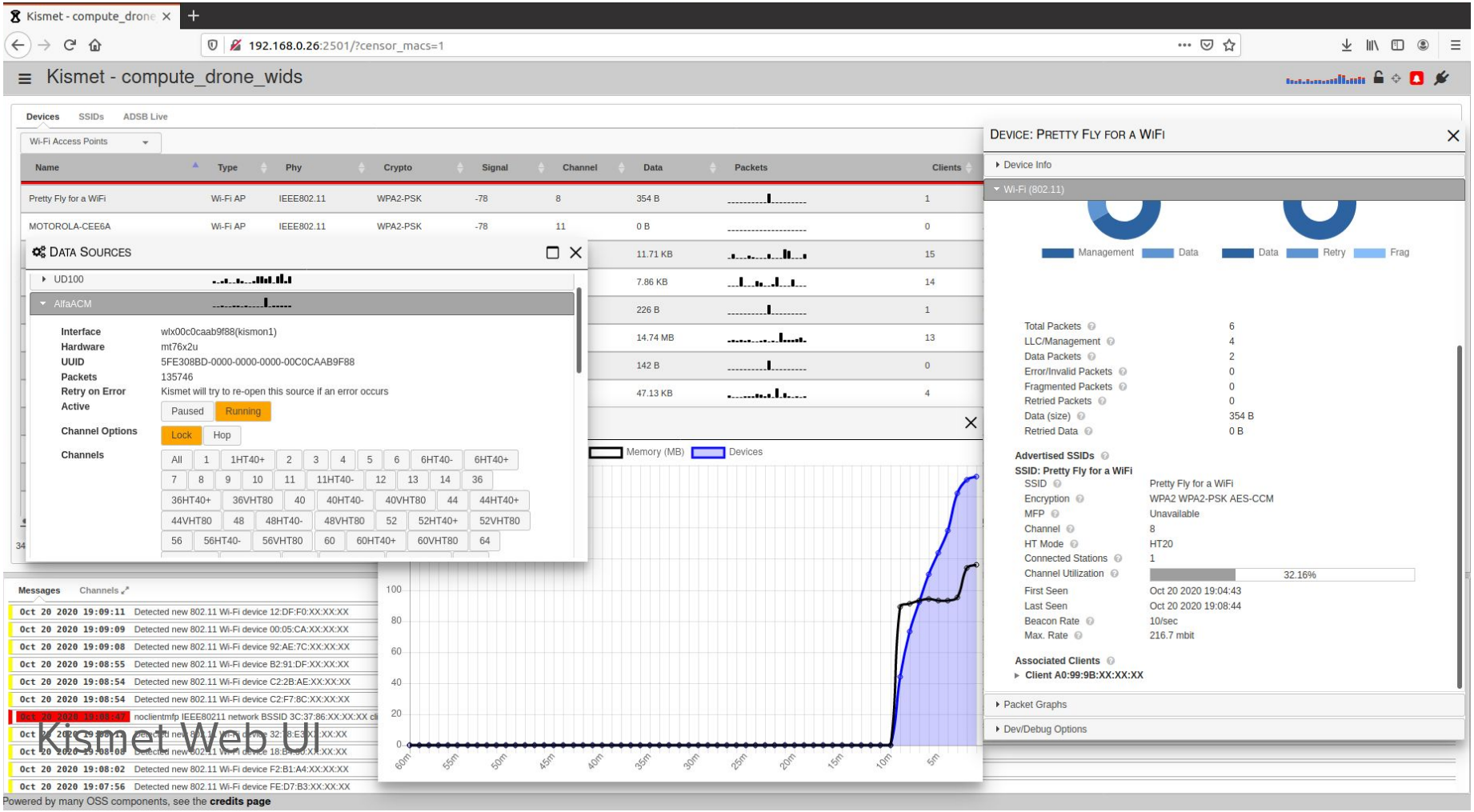
https://kismetwireless.net

https://kismetwireless.net/docs

@kismetwireless

Discord: https://discord.gg/5N4ME9a

Dragorn's recent SharkFest Presentation:
https://www.youtube.com/watch?v=z6MzlDwjUmc


Kismet - compute_drone
192.168.0.26:2501/?censor_macs=1
Kismet - compute_drone_wids
Devices
SSIDs
ADSB Live
Wi-Fi Access Points
Name
Type
Phy
Crypto
Signal
Channel
Data
Packets
Clients
Pretty Fly for a WiFi
Wi-Fi AP
IEEE802.11
WPA2-PSK
-78
8
354 B
1
MOTOROLA-CEE6A
Wi-Fi AP
IEEE802.11
WPA2-PSK
-78
11
0 B
0
11.71 KB
15
7.86 KB
14
226 B
1
14.74 MB
13
142 B
0
47.13 KB
4
DATA SOURCES
UD100
AlfaACM
Interface
wlx00c0caab9f88(kismon1)
Hardware
mt76x2u
UUID
5FE308BD-0000-0000-0000-00C0CAAB9F88
Packets
135746
Retry on Error
Kismet will try to re-open this source if an error occurs
Active
Paused
Running
Channel Options
Lock
Hop
Channels
All
1
1HT40+
2
3
4
5
6
6HT40-
6HT40+
7
8
9
10
11
11HT40-
12
13
14
36
36HT40+
36VHT80
40
40HT40-
40VHT80
44
44HT40+
44VHT80
48
48HT40-
48VHT80
52
52HT40+
52VHT80
56
56HT40-
56VHT80
60
60HT40+
60VHT80
64
Memory (MB)
Devices
100
80
60
40
20
0
60m
55m
50m
45m
40m
35m
30m
25m
20m
15m
10m
5m
DEVICE: PRETTY FLY FOR A WIFI
Device Info
Wi-Fi (802.11)
Management
Data
Data
Retry
Frag
Total Packets
6
LLC/Management
4
Data Packets
2
Error/Invalid Packets
0
Fragmented Packets
0
Retried Packets
0
Data (size)
354 B
Retried Data
0 B
Advertised SSIDs
SSID: Pretty Fly for a WiFi
SSID
Pretty Fly for a WiFi
Encryption
WPA2 WPA2-PSK AES-CCM
MFP
Unavailable
Channel
8
HT Mode
HT20
Connected Stations
1
Channel Utilization
32.16%
First Seen
Oct 20 2020 19:04:43
Last Seen
Oct 20 2020 19:08:44
Beacon Rate
10/sec
Max. Rate
216.7 mbit
Associated Clients
Client A0:99:9B:XX:XX:XX
Packet Graphs
Dev/Debug Options
Messages
Channels
Oct 20 2020 19:09:11 Detected new 802.11 Wi-Fi device 12:DF:F0:XX:XX:XX
Oct 20 2020 19:09:09 Detected new 802.11 Wi-Fi device 00:05:CA:XX:XX:XX
Oct 20 2020 19:09:08 Detected new 802.11 Wi-Fi device 92:AE:7C:XX:XX:XX
Oct 20 2020 19:08:55 Detected new 802.11 Wi-Fi device B2:91:DF:XX:XX:XX
Oct 20 2020 19:08:54 Detected new 802.11 Wi-Fi device C2:2B:AE:XX:XX:XX
Oct 20 2020 19:08:54 Detected new 802.11 Wi-Fi device C2:F7:8C:XX:XX:XX
Oct 20 2020 19:08:47 noclientmfp IEEE80211 network BSSID 3C:37:86:XX:XX:XX
Oct 20 2020 19:08:02 Detected new 802.11 Wi-Fi device F2:B1:A4:XX:XX:XX
Oct 20 2020 19:07:56 Detected new 802.11 Wi-Fi device FE:D7:B3:XX:XX:XX
Kismet Web UI
Powered by many OSS components, see the credits page

Kismet Collection Platforms

“War-Box”/Static Collection Platform

- Intel Compute Stick 325
  - Ubuntu 20.04.01
- 16 Port USB Hub
- Panda PAU09 x 1 (Channel locked)
- Alfa AWUS036ACM x 1 (Channel locked)
- Rosewill x 2 (Scanning)
- Sena UD100
- Ubertooth
- Internal Bluetooth
- NooElec SDR (ADS-B)
- RTL-SDR v3 (RTL-433)
  - rtl-433 dependency
- USB VFAN BG-803 (Ublox) GPS x 1

Kismet Collection Platforms

“War-Pack”

- Raspberry Pi 3
  - Rasbian
  - Re4son Kernel
- Panda PAU09 x 2
- NooElec SDR x 1
- Anker 20000 MHa Battery
- GlobalSat BU-353 S4
- Bagsmart Travel Bag

Kismet Config Tips

Config files - kismet_site.conf

server_name=compute_drone

log_prefix=/home/compute/kismet/
log_title=homecaptures
log_types=kismet,pcapppi

gps=gpsd:host=localhost,port=2947
source=wlx00c0caab9f88:name=AlfaACM,channels="1,6,11,36,44,149,153,157,161",vht_channels=false,ht_channels=false
source=rtl433-00009002:name=RTLSDRv3Int
source_stagger_threshold=6

hidedata=true
kis_log_device_filter=IEEE802.11,aa:bb:cc:dd:ee:ff,block
kis_log_packet_filter=IEEE802.11,aa:bb:cc:dd:ee:ff,block

Kismet Data

Logging Data Output Types

Foo.kismetdb - sqlitedb; RDB with JSON blob
Foo.pcapppi - PPI has GPS and signal details
Foo.pcapng - Types combined, Radiotap

Data processing tools

kismetdb_clean	kismetdb_dump_devices
kismetdb_statistics	kismetdb_strip_packets
kismetdb_to_gpx	kismetdb_to_kml
kismetdb_to_pcap	kismetdb_to_wiglecsv

API

curl http://username:password@localhost:2501/system/status.prettyjson
http://localhost:2501/system/tracked_fields.html
Serialization: JSON vs EKJSON

Example Data Fields

[kismet.device.base] - Fields common to all devices

- kismet.device.base.macaddr - mac address
- kismet.device.base.type - printable device type
- kismet.device.base.manuf - manufacturer name

[dot11] - Fields specific to 802.11 devices; lots of nested fields

- dot11.probedssid.ssid - probed SSID
- dot11.device.dot11.device.last.probed.ssid.record.dot11.probedssid.ssid - Actual nested field result

[rtl433] - Fields for rtl433 devices

- rtl433.device.lightning_strike_count - Strike count

Elasticsearch, Logstash, and Kibana (ELK) Stack

Elasticsearch - Document database built for scaling

Logstash - Data pipeline tool that parses files/logs and pushes them into Elasticsearch

Kibana - Web based visualization tool that displays Elasticsearch data

Windows/Mac/Linux

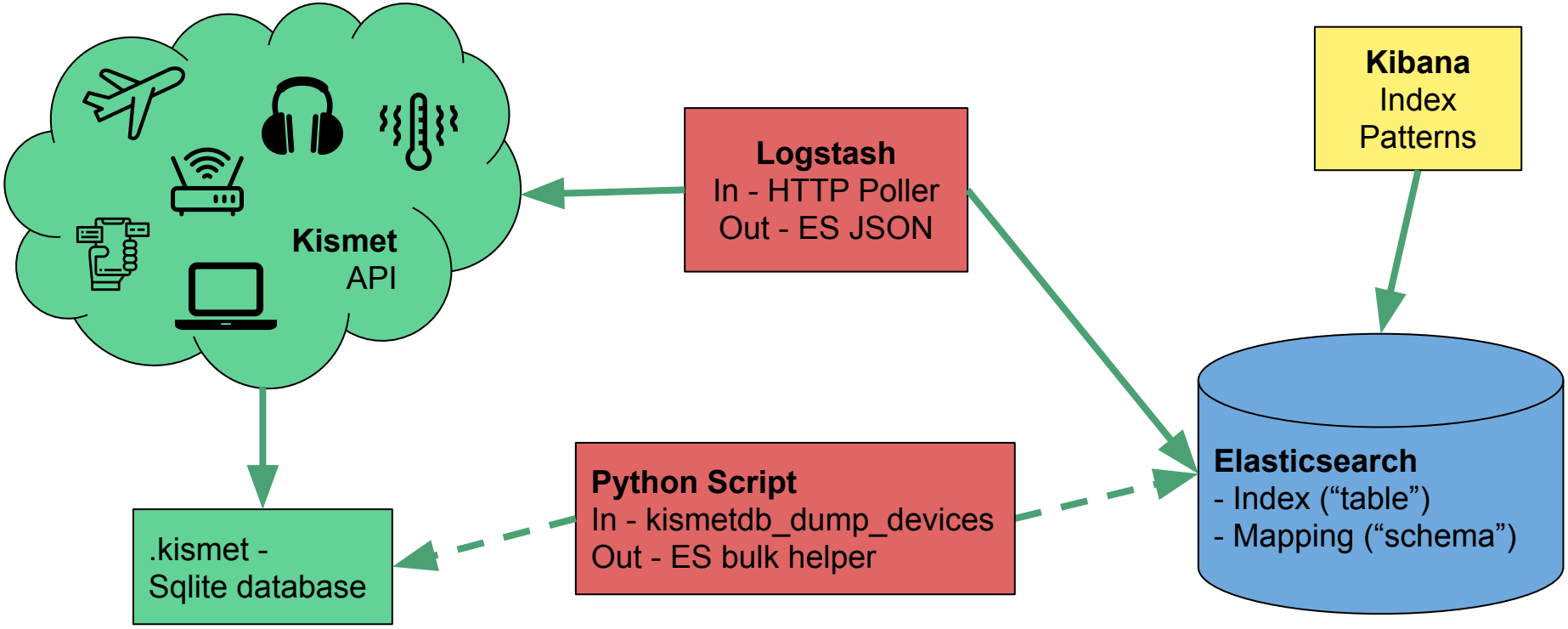
ELK Software Options

- FOSS Package (Limited)
- Basic License (Better)
- Gold, Platinum, Etc...

Deployment Options

- Computer/VM Self Managed
- **Bitnami Virtual Machine Images**
- AWS Cloud
- Elastic.co Cloud (Managed/Self Managed)

WAWKELK Service Diagram



Elasticsearch

Defining Indices
- Lots of optimization settings; great documentation

Defining Mappings
- Static versus Dynamic Mapping
- Must be defined in advance; once data is in the index, it cannot be changed*

Use the Dev Console in Kibana
Test, test, and test again; once happy, blow away everything and start fresh
ES suffers when fields exceed 1000
Nested fields are *terrific*...
Kibana won't detect nested geo_point fields, so you have to pull them into the base document

```
PUT kismet_device_activity
{
  "settings": {
    "index.mapping.ignore_malformed": true
  },
  "mappings": {
    "dynamic": true,
    "numeric_detection": true,
    "properties": {
      "kismet_device_base_location_avg_geopoint": {
        "type": "geo_point"
      },
      "kismet_device_base_location": {
        "type": "nested",
        "properties": {
          "kismet_common_location_avg_loc": {
            "type": "nested",
            "properties": {
              "kismet_common_location_geopoint": {
                "type": "geo_point"
              }
            }
          }
        }
      },
      "kismet_device_base_tags": {
        "type": "nested",
        "properties": {
          "notes": {
            "type":"text"
          }
        }
      }
    }
  }
}
```

Logstash

Runs as a service

Inputs

- Lots of options; HTTP Poller best met this use case

Filters

- Data mangling

Outputs

- Elasticsearch

Kismet-isms

- EKJSON - 1 document per line
- Request fields in “.” notation
- Receive fields in “_” notation

Logstash - Inputs

```
input {
  http_poller {
            urls => {
            device_activity => {
            url => "http://192.168.0.100:2501/devices/last-time/-305/devices.ekjson"
            method => post
            auth => { user => "kismet"    password => "password"     }
            body =>
'json={"fields":["kismet.device.base.type","kismet.device.base.location","kismet.device.base.tags","kismet.device.base.manuf","kismet.device.base.num_alerts","kismet.device.base.server_uuid","kismet.device.base.last_time","kismet.device.base.frequency","kismet.device.base.channel","kismet.device.base.crypt","kismet.device.base.commonname","kismet.device.base.name","kismet.device.base.phyname","kismet.device.base.macaddr","kismet.device.base.key","kismet.device.base.key","kismet.device.base.packets.rx","kismet.device.base.packets.tx"]}'
            headers => {
            "Accept" => "application/json"
            "Content-Type" => "application/x-www-form-urlencoded"
            }
            }
            }
            add_field => { "[@metadata][dest_index]" => "kismet_device_activity" }
            add_field => { "kismet_host_inline" => "system-one" }
            add_field => { "kismet_location_inline" => "home" }
            request_timeout => 8
            #schedule => { cron => "* * * * * UTC"}
            schedule => { every => "5m" }
            codec => "json_lines"
  }
}
```

1
2
3
4
5
6

Logstash - Filters

```
filter {
  if ([@metadata][dest_index] == "kismet_device_activity") {
        date {
                match => ["kismet_device_base_last_time","UNIX"] target => "kismetdtg"
        }
        if [kismet_device_base_channel] == "FHSS" {
                prune { remove_field => ["[kismet_device_base_channel]"] }
                mutate { add_field => { "[kismet_device_base_channel_fhss]" => "FHSS" } }
                mutate { add_field => { "[kismet_device_base_channel]" => 0 } }
        }
        if [kismet_device_base_tags] == 0 {
                prune { remove_field => ["[kismet_device_base_tags]"] }
                mutate {
                add_field => { "[kismet_device_base_tags][notes]" => "NA" }
                }
        }
        prune {
                remove_field => ["[kismet_device_base_location][kismet_common_location_last]"]
                remove_field => ["[kismet_device_base_location][kismet_common_location_max_loc]"]
                remove_field => ["[kismet_device_base_location][kismet_common_location_min_loc]"]
        }
        mutate {
                add_field => {
                        "my_fingerprint" => "devact_%{[kismet_device_base_key]}_%{[kismet_device_base_last_time]}"
                }
        }
  }
}
```

1
2
3
4
5

Logstash - Outputs

```
output {
  elasticsearch {
        hosts => ["localhost:9200"]
    1   index => "%{[@metadata][dest_index]}"
        document_id => "%{[my_fingerprint]}"
  }                                              2
  #disable this once in prod stdout
3 #stdout { codec => rubydebug } # output any left overs
}
```

Tips

Read the (current) docs/SO/boards

Trial and error

Use repeatable test data

Get paranoid about {}'s

Python

MUCH more customizable and powerful

Kismet and Elasticsearch libraries

Can check for and push indices

Can filter JSONs, conduct interim analysis, and create new fields

Can be scripted

Kibana Visualizations

Create an Index Pattern from an ES Index (Wildcards to match sub ES Indices)

Refresh field indexes when new fields appear

Creating a Visualization

- Types - Line/Bar, Heat Map, Coordinate Map
- Select Index Pattern
- Metrics (Y-Axis) - Count, Unique Count
- Buckets
  - X-Axis
  - Split Series
  - Split Chart
- Aggregations - Date Histogram, Terms
- Adjust “Metrics & Axis” & “Panel Settings”

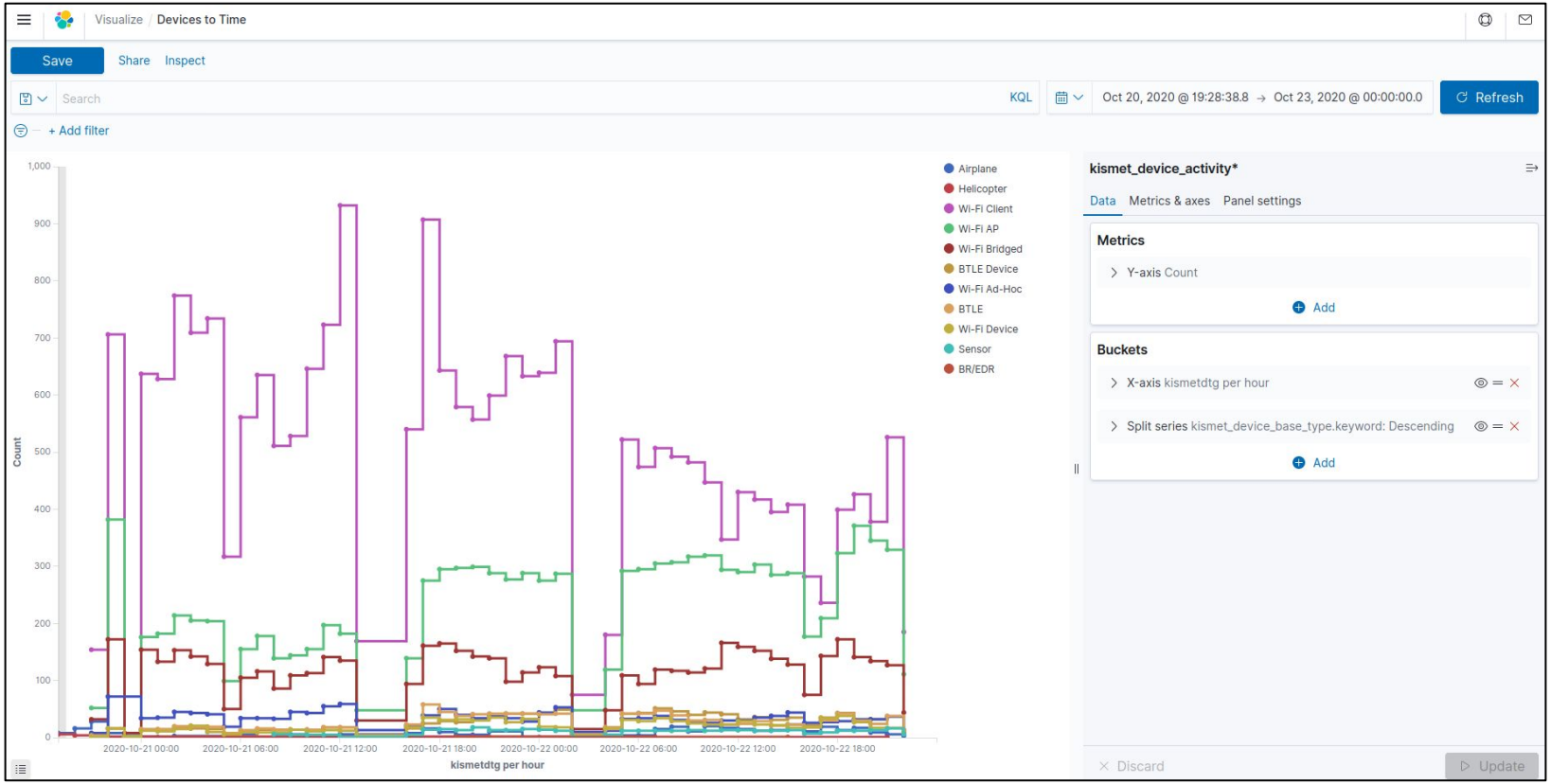
Key Fields

- kismet_device_base_key.keyword - Uniquely ID’s a device
- kismet_device_base_type.keyword - Technology

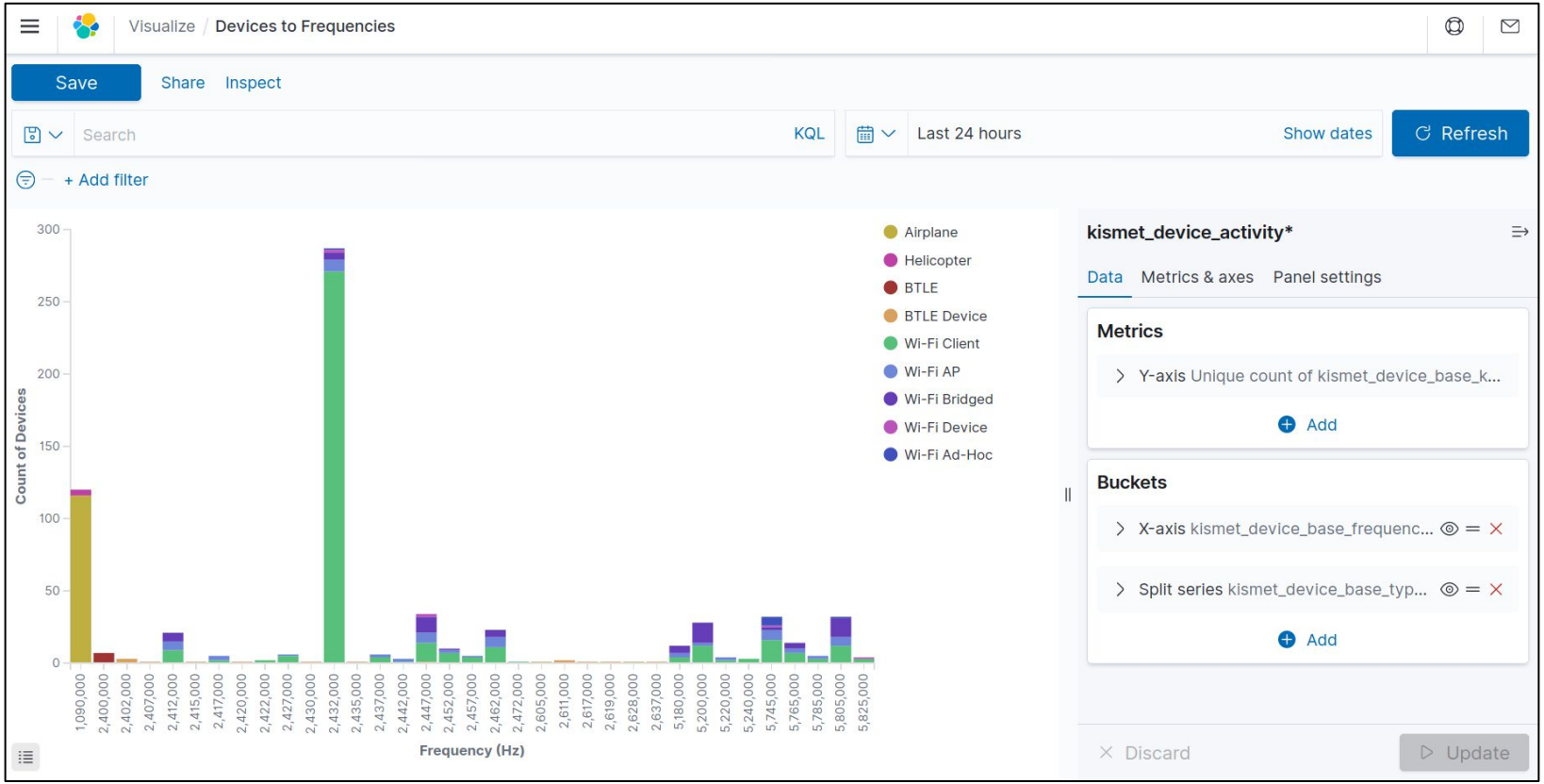
Tips

- Build in filters (but don’t forget you did)
- Visualizations are tied to the Index Pattern’s UUID...

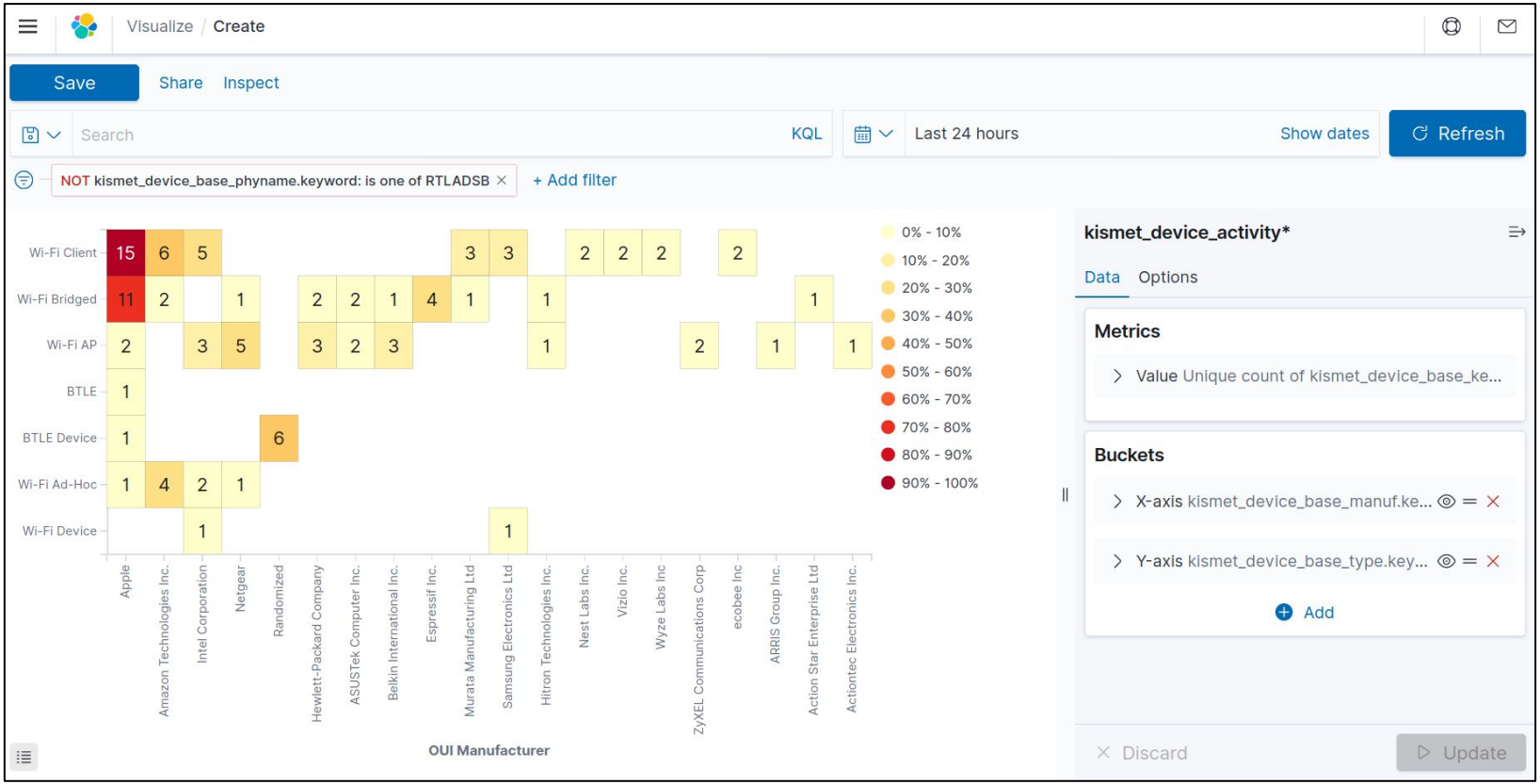
Line Graph - Device Types to Time



Bar Graph - Device Types to Frequency



Heat Map - Device Types to Manufacturer



Coordinate Map - Device Locations

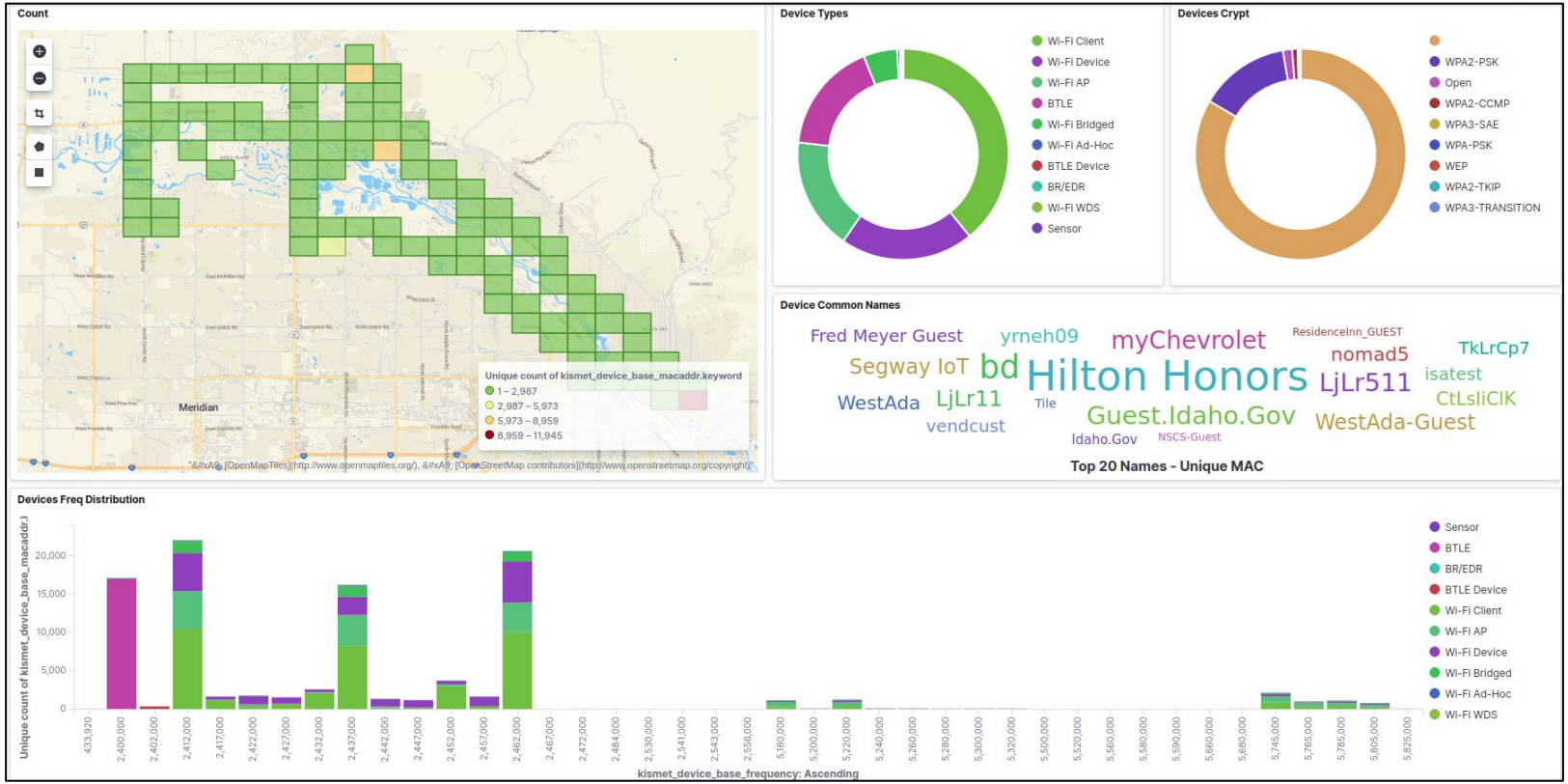
Elasticsearch Organic WMS is limited to a zoom depth of 12

Custom WMS:

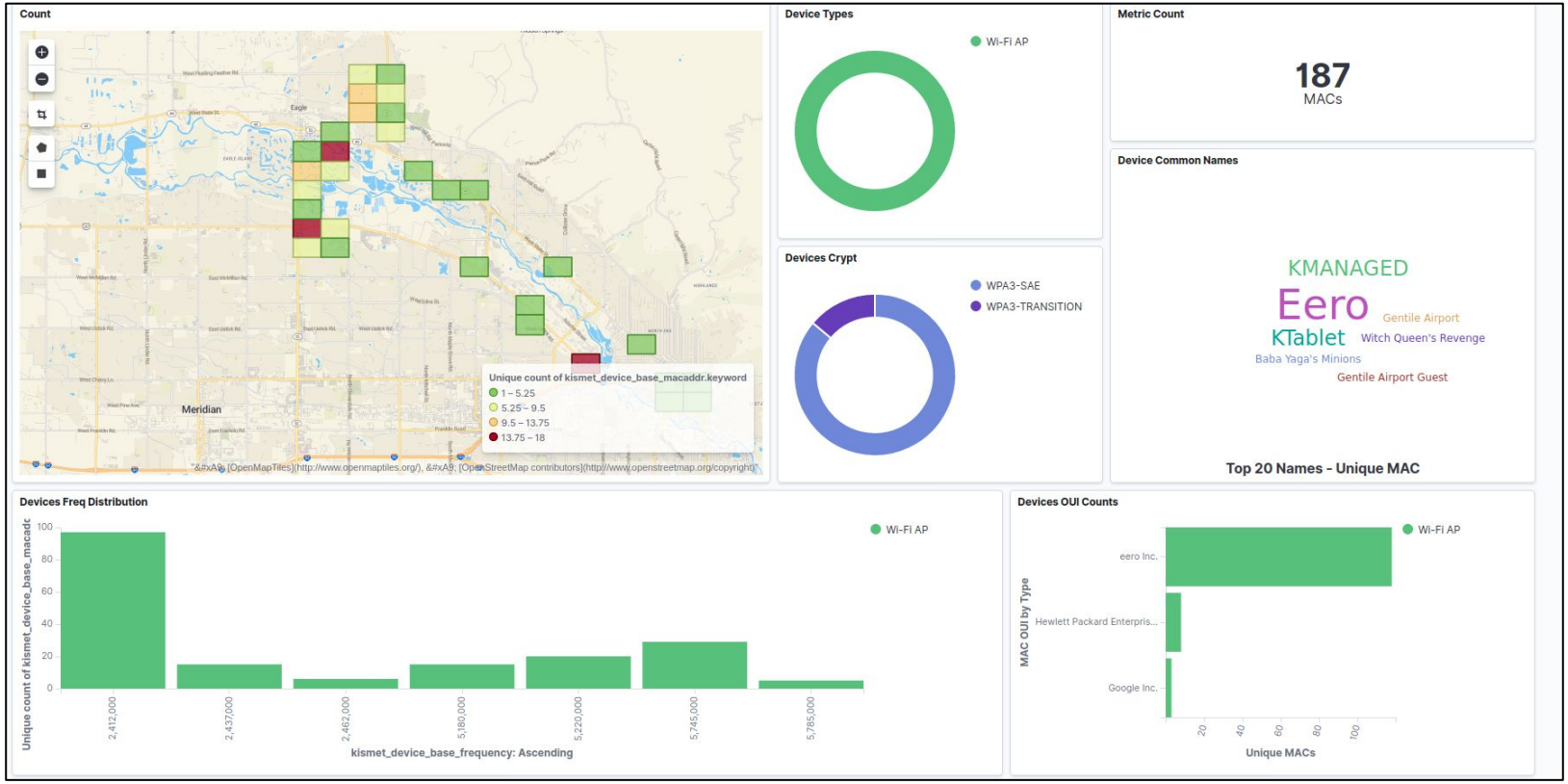
- https://www.maptiler.com
- WMS:
- Streets:
  - https://api.maptiler.com/maps/streets/{z}/{x}/{y}.png?key=<your key>
  - image/jpg
- Satellite:
  - WMS: https://api.maptiler.com/maps/hybrid/{z}/{x}/{y}.jpg?key=<your key>
  - image/png

Dashboards - Home Monitoring

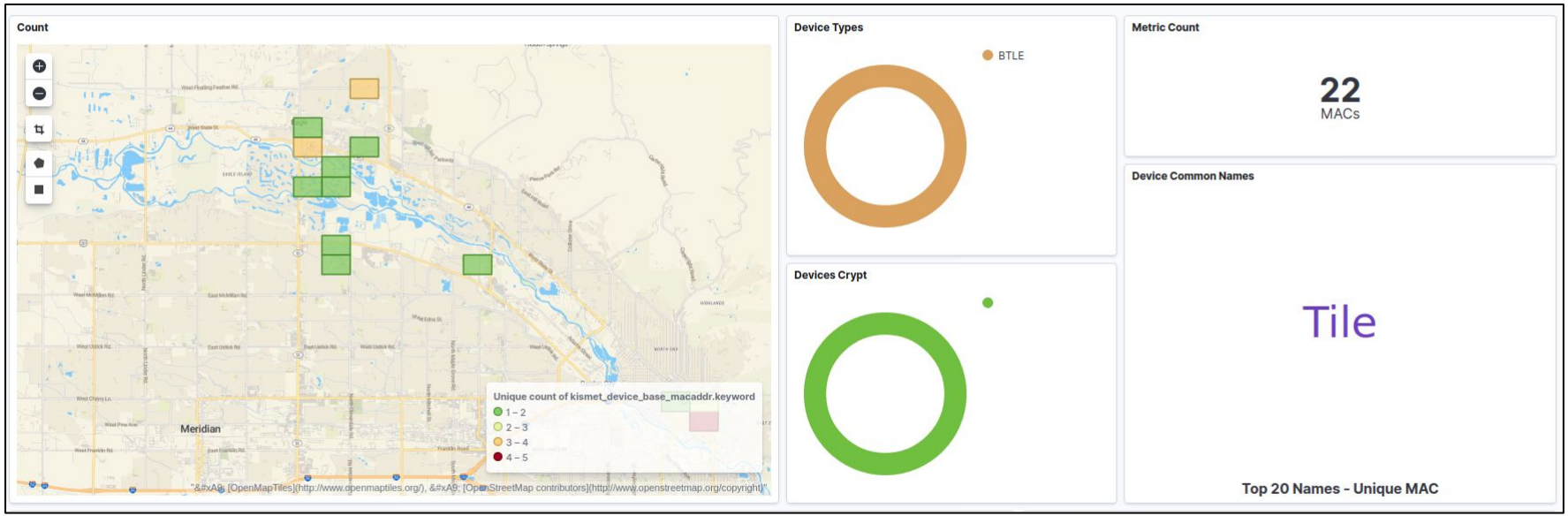
Dashboards - Spatial Trends



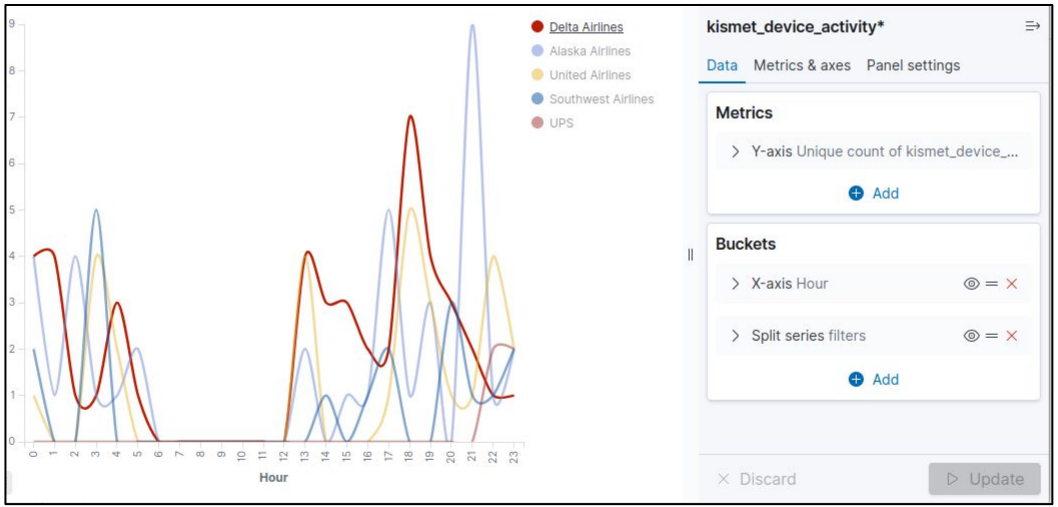
WPA3-SAE/Transition Distribution



Tile Devices



ADS-B: Airline Traffic Patterns



Scripted field in Kibana

- doc['@timestamp'].value.getHour()

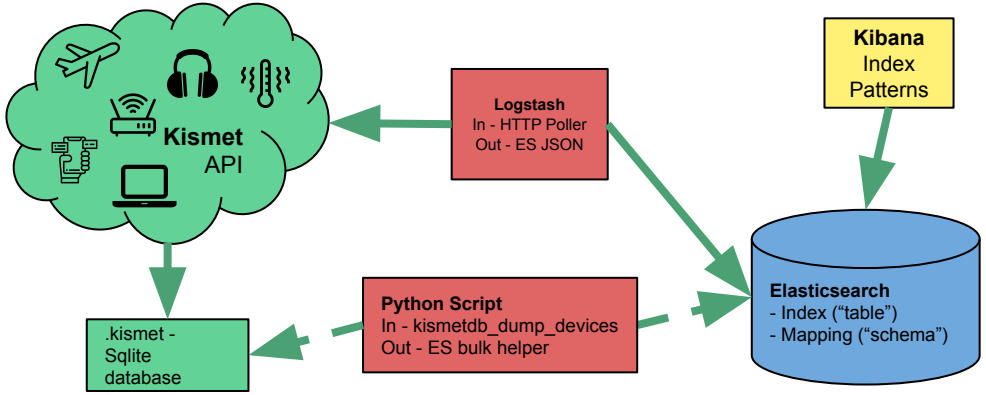
Filters

- Lucene not KQL
- {"bool":{"should":[{"match_phrase":{"kismet_device_base_commonname":"*DELTA*"}}]}}

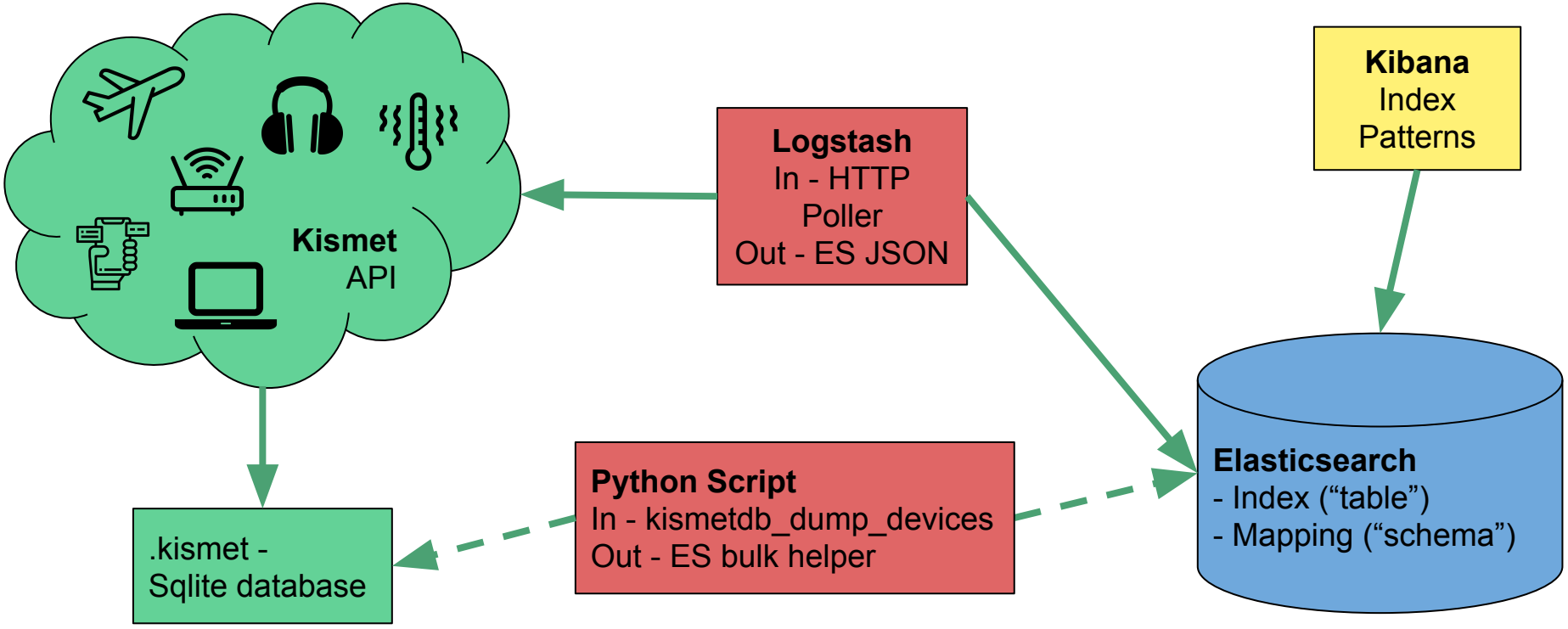
Order of Operations

1. Start Elasticsearch and Kibana
2. Create index and mapping in Elasticsearch
3. Configure Logstash pipeline
4. Start Kismet
5. Start Logstash
6. Once data is in ES, create a Kibana Index Pattern
7. Verify data in Kibana
8. Create visualizations and dashboards
9. Profit

WAWKELK Service Diagram



WAWKELK Service Diagram



Final Thoughts

Security - Careful how you expose/access this data

Github - Adding examples, lessons learned, and quick start guides

https://github.com/developingchase/wawkelk

Please contribute/add suggestions

Feel free to reach out with any questions:

- @developingchase
- Discord: chase#1342
- Via Github's Issue tracker