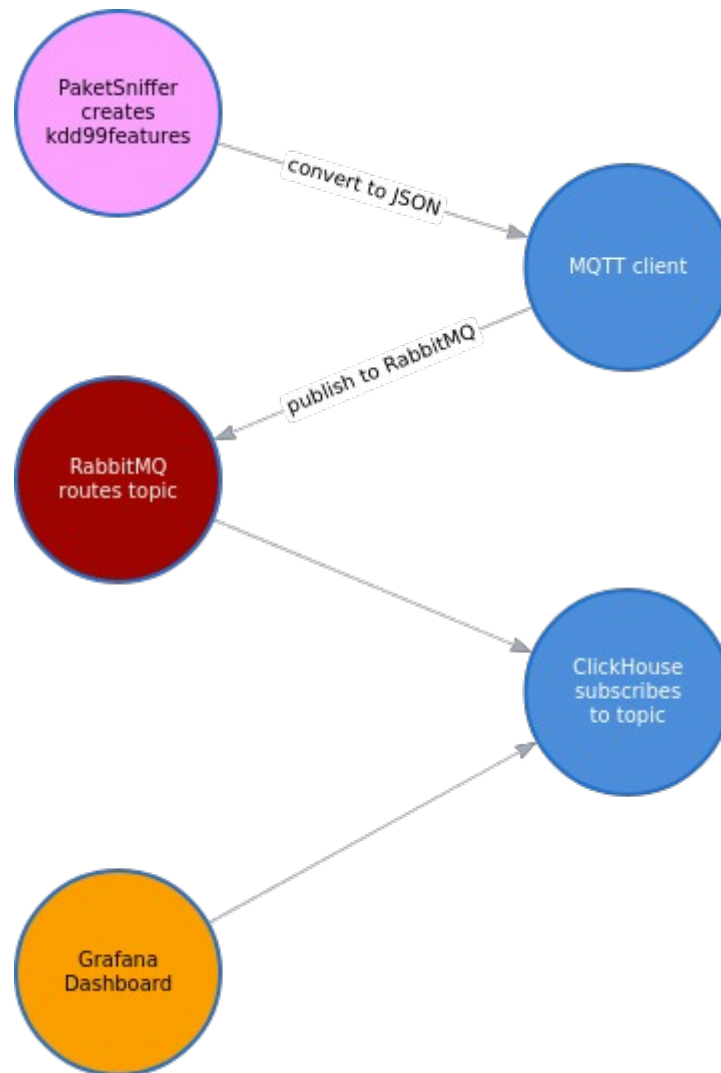


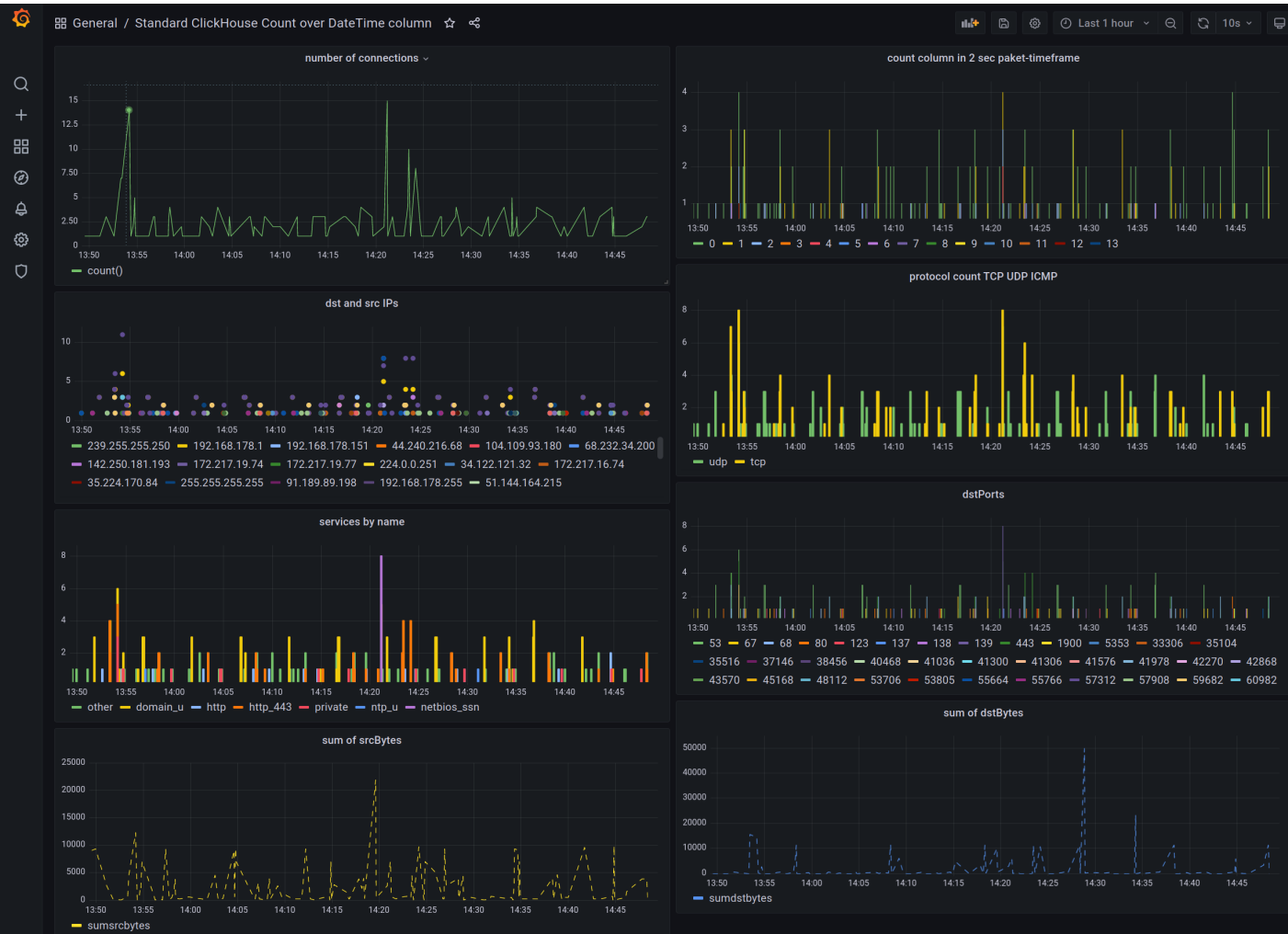
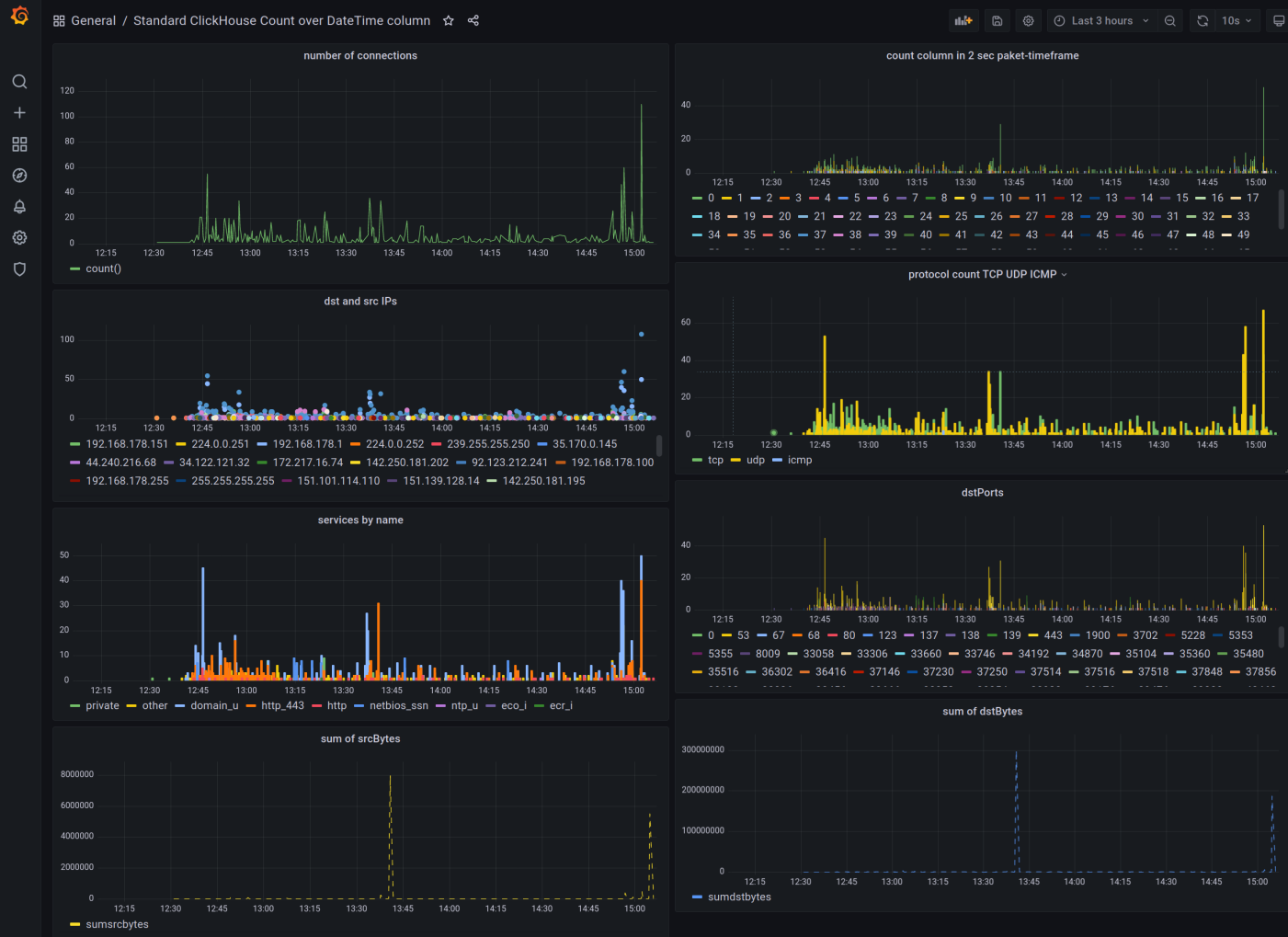
## “almost live” KDD99 features to Grafana Dashboard

Workflow:

- collect via libpcap
- make kdd99 features
- send via MQTT client to RabbitMQ
- ClickHouse Database connects to RabbitMQ and subscribes to kdd99 “topic”
- Grafana visualizes metrics via ClickHouse plugin



# Endresult: Grafana Dashboard of locally collected kdd99 features



## Packet Sniffer with features found in the famous kdd99 Dataset:

- add MQTT capability to kdd99 feature extractor from [https://github.com/AI-IDS/kdd99\\_feature\\_extractor](https://github.com/AI-IDS/kdd99_feature_extractor)
- in a thread send a json string of selected features
- which the Rabbitmq broker with MQTT Plugin will route to ClickHouse column-based Database
- add ClickHouse RabbitMQ Engine with tables with columns according to features
- add Grafana for Dashboarding via ClickHouse plugin (<https://grafana.com/grafana/plugins/vertamedia-clickhouse-datasource/>)

### ClickHouse config:

#### run commands with "clickhouse-client --password -m" CLI

```
=====
CREATE TABLE incomingKDD99
(
  `time` DateTime,
  `srcIP` String,
  `srcPort` UInt16,
  `dstIP` String,
  `dstPort` UInt16,
  `protocol` String,
  `srcbytes` UInt64,
  `dstbytes` UInt64,
  `count` UInt64,
  `service` String
)
ENGINE = RabbitMQ SETTINGS rabbitmq_host_port = 'localhost:5672', rabbitmq_exchange_name =
'toClickHouse', rabbitmq_exchange_type = 'direct', rabbitmq_routing_key_list = 'rabbitkdd99',
rabbitmq_format = 'JSONEachRow', rabbitmq_num_consumers = 1, date_time_input_format =
'best_effort';

CREATE TABLE kdd99
(
  `time` DateTime,
  `srcIP` String,
  `srcPort` UInt16,
  `dstIP` String,
  `dstPort` UInt16,
  `protocol` String,
  `srcbytes` UInt64,
  `dstbytes` UInt64,
  `count` UInt64,
  `service` String
)
ENGINE = MergeTree
ORDER BY time;

CREATE MATERIALIZED VIEW kdd99_material TO kdd99
AS SELECT * FROM incomingKDD99;
=====
```

```

CREATE TABLE incomingKDD99
(
    `time` DateTime,
    `srcIP` String,
    `srcPort` UInt16,
    `dstIP` String,
    `dstPort` UInt16,
    `protocol` String,
    `srcbytes` UInt64,
    `dstbytes` UInt64,
    `count` UInt64,
    `service` String
)
ENGINE = RabbitMQ
SETTINGS rabbitmq_host_port = 'localhost:5672', rabbitmq_exchange_name = 'toClickHouse', rabbitmq_exchange_type = 'direct', rabbitmq_routing_key_list = 'rabbitkdd99', rabbitmq_format = 'JSONEachRow', rabbitmq_num_consumers = 1, date_time_input_format = 'best_effort'

Query id: a512c16e-4b15-4371-b443-2f35d96d4c63

Ok.

0 rows in set. Elapsed: 1.437 sec.

ubu :) CREATE TABLE kdd99
:-] (
:-] `time` DateTime,
:-] `srcIP` String,
:-] `srcPort` UInt16,
:-] `dstIP` String,
:-] `dstPort` UInt16,
:-] `protocol` String,
:-] `srcbytes` UInt64,
:-] `dstbytes` UInt64,
:-] `count` UInt64,
:-] `service` String
:-] )
:-] ENGINE = MergeTree
:-] ORDER BY time;

CREATE TABLE kdd99
(
    `time` DateTime,
    `srcIP` String,
    `srcPort` UInt16,
    `dstIP` String,
    `dstPort` UInt16,
    `protocol` String,
    `srcbytes` UInt64,
    `dstbytes` UInt64,
    `count` UInt64,
    `service` String
)
ENGINE = MergeTree
ORDER BY time

Query id: d91e9082-9b15-4754-9167-b29698c7602d

Ok.

0 rows in set. Elapsed: 0.014 sec.

ubu :)
ubu :) CREATE MATERIALIZED VIEW kdd99_material TO kdd99
:-] AS SELECT * FROM incomingKDD99;
:-] ;

CREATE MATERIALIZED VIEW kdd99_material TO kdd99 AS
SELECT *
FROM incomingKDD99

Query id: 75275476-7275-470b-a40f-eb24a3da21f8

Ok.

0 rows in set. Elapsed: 0.022 sec.

ubu :) 

```

- will add RabbitMQ Engine and materialized view to access persistent table

## RABBITMQ config:

enable MQTT plugin & enable anonymous publishing

- add "mqtt.allow\_anonymous = true" to "/etc/rabbitmq/rabbitmq.conf"
- "[rabbitmq\_management,rabbitmq\_mqtt]." to "/etc/rabbitmq/enabled\_plugins"

- binding from amq.topic Exchange, which stores all MQTT messages, to ClickHouse queue

**RabbitMQ** ™ RabbitMQ 3.8.18 Erlang 24.0.3


Refreshed 2021-07-05 12:43:02 Refresh every 5 seconds

Virtual host All

Cluster rabbit@ubu

User guest Log out

Overview Connections Channels Exchanges **Queues** Admin



0.0 /s

0.0 /s

0.0 /s

0.0 /s

0.0 /s

0.0 /s

11:45 11:50 11:55 12:00 12:05 12:10 12:15 12:20 12:25 12:30 12:35 12:40

Publish 0.00/s

Deliver (manual ack) 0.00/s

Deliver (auto ack) 0.00/s

Consumer ack 0.00/s

Redelivered (manual ack) 0.00/s

Get (auto ack) 0.00/s

Get (empty) 0.00/s

Details

Features x-max-length: 1048545

x-overflow: reject-publish

State idle

Consumers 1

Consumer capacity 100%

Messages 0

Message body bytes 0 B

Process memory 14 kiB

Policy Operator policy

Effective policy definition

From (Default exchange binding)

Routing key

Arguments

toClickHouse\_default\_incomingKDD99\_bridge 1 Unbind

↓

This queue

Add binding to this queue

From exchange: amq.topic \*

Routing key: kdd99features

Arguments: = String

Bind