EMQ

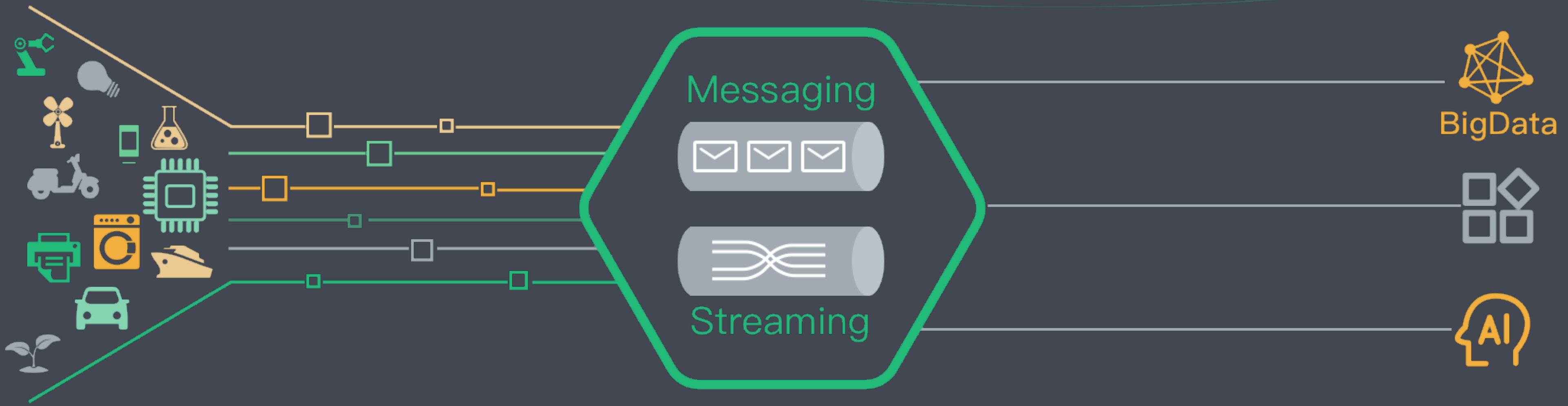
www.emqx.io

EdgeX Geneva 规则引擎介绍

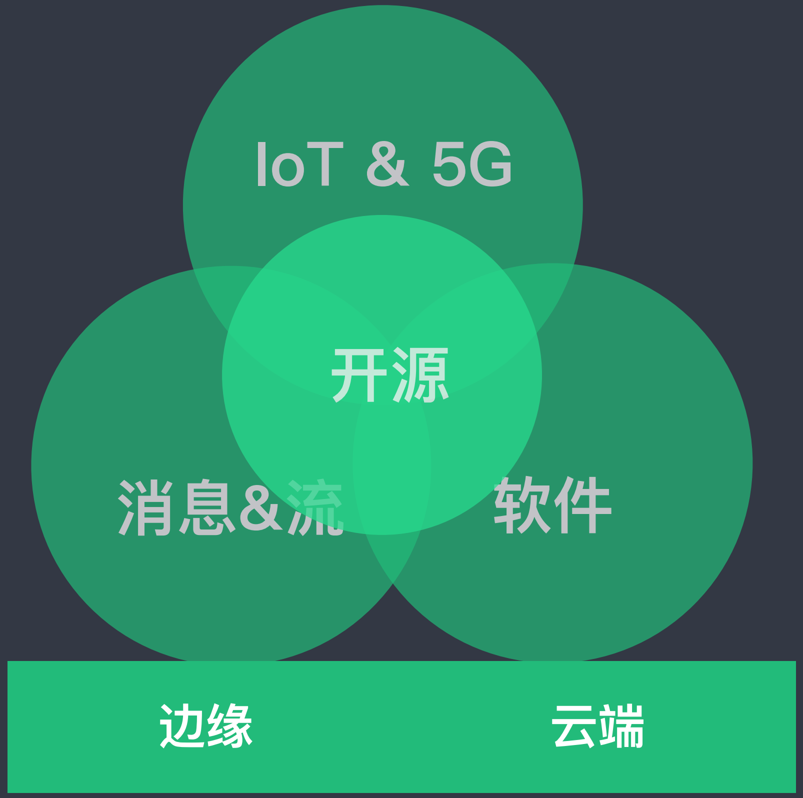
— 基于 EMQ X Kuiper 的超轻量级 IoT 流式数据处理

Apr, 2020

Rocky@emqx.io
杭州映云科技有限公司



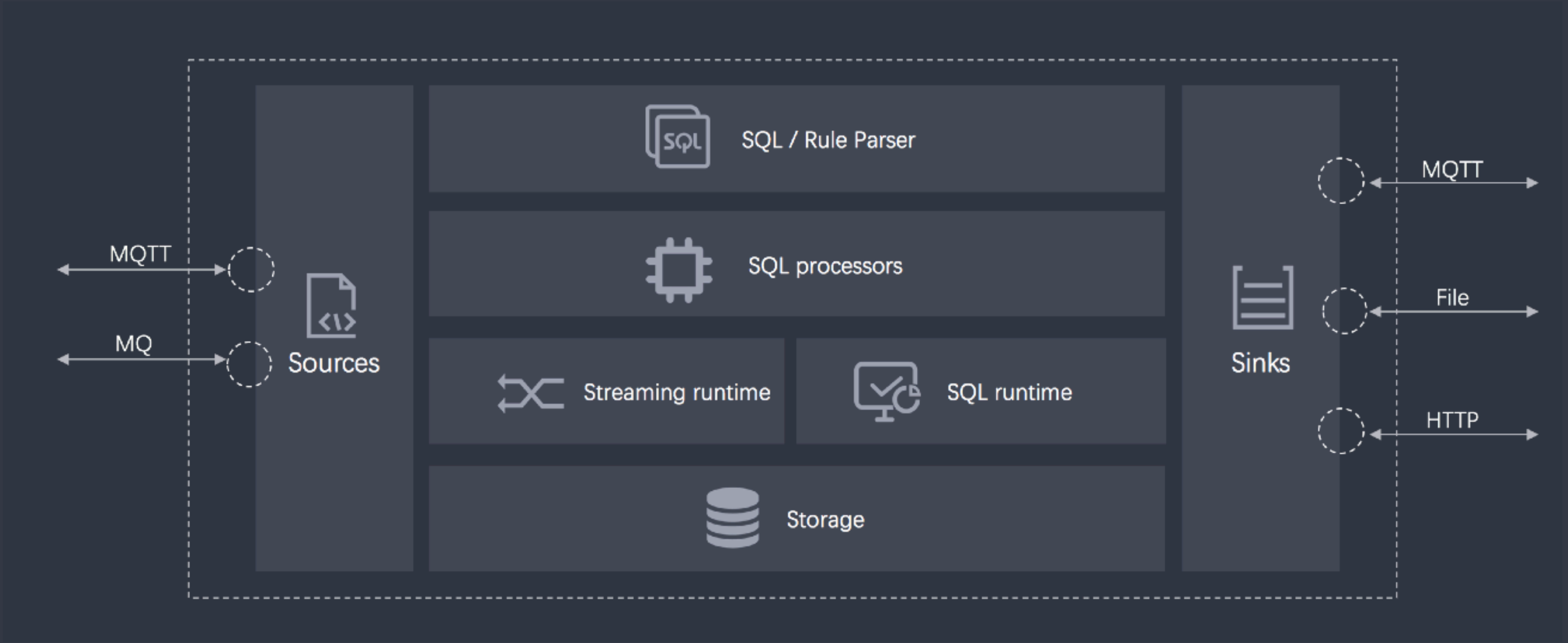
EMQ – 开源物联网消息接入与数据处理领导者



1. 商业化开源软件
2. 服务于 5G 时代的 IoT 产业
3. 消息与流处理
4. 全球 5000+ 企业用户
5. 全球运营: 中国、美国硅谷、欧洲

EMQ X Kuiper

- 将 Apache Flink、Spark 运行在边缘端!
- 原生二进制可运行包
- 基于 SQL 的业务逻辑实现方式
- 基于 Golang 的可扩展框架
  - 数据源
  - 数据目标 (Sinks)
  - 函数
- 规则管理的 API 与 CLI
- 基于 Apache 2.0 开源协议



项目地址：https://github.com/emqx/kuiper

Kuiper 使用过程

- 定义流
  - 类似于数据库中表格的定义

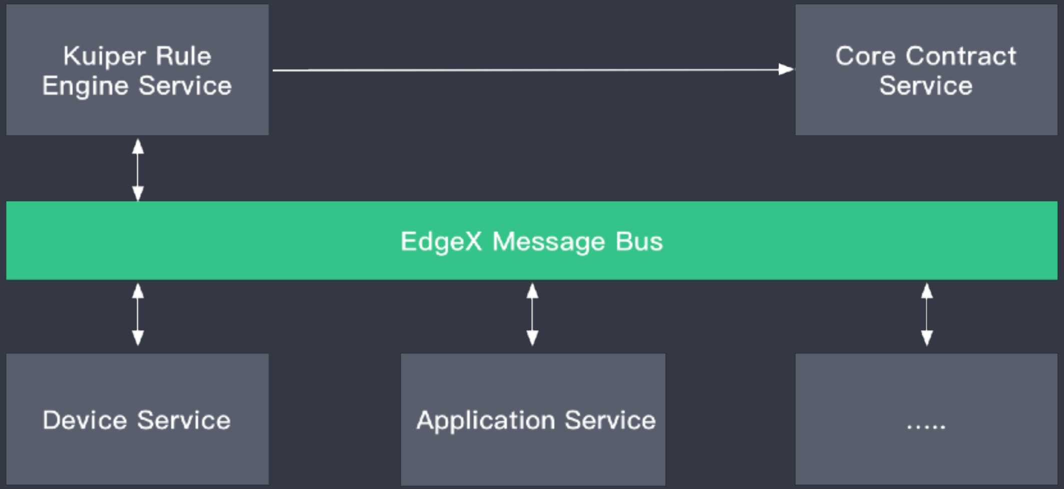
```
# bin/cli create stream demo '(temperature float, humidity bigint)
WITH (FORMAT="JSON", DATASOURCE="devices/+/messages")'
```

- 定义并提交规则
  - 用 SQL 实现业务逻辑，并将运行结果发送到指定目标
  - 支持的 SQL
    - SELECT/FROM/WHERE/ORDER
    - JOIN/GROUP/HAVING
    - 4类时间窗口
    - 60+ SQL 函数
- 规则运行

```
{
  "sql": "SELECT avg(temperature) AS t_av, max(temperature) AS
t_max, min(temperature) AS t_min, COUNT(*) As t_count,
split_value(mqtt(topic), \"/\", 1) AS device_id FROM demo GROUP BY
device_id, TUMBLINGWINDOW(ss, 10)",
  "actions": [
    {
      "log": {}
    },
    {
      "mqtt": {
        "server": "ssl://xyz-ats.iot.us-east-1.amazonaws.com:8883",
        "topic": "devices/result",
        "qos": 1,
        "clientId": "demo_001",
        "certificationPath": "/var/aws/d3807d9fa5-certificate.pem",
        "privateKeyPath": "/var/aws/d3807d9fa5-private.pem.key"
      }
    }
  ]
}
```

EdgeX 微服务之间的数据交互

- EdgeX 数据总线
  - 微服务之间的数据交流总线
  - 支持 ZeroMQ 与 MQTT
- 数据类型的处理
  - 从 Core Contract Service 中读取数据类型
  - 将数据转换为 Kuiper 的数据类型
  - EdgeX 中无需定义字段的类型，使用 schema-less 流定义
- 支持：Bool、String、Uint8、Uint16、Uint32、Uint64、Int8、Int16、Int32、Int64、Float32、Float64
- Binary, Array – 不支持



EdgeX Event & Kuiper 数据映射

创建流

```
create stream events() WITH (FORMAT="JSON", TYPE="edgex")
```

models.Event 数据样例

```
Events:
{Device: "demo", Created: 000, ...}
    reading[0]: {Name: "Temperature", value: “30”,“Created”:123 ...}
    reading[1]: {Name: "Humidity", value: “20”,“Created”:456 ...}
```

1) SELECT temperature, humidity FROM events WHERE meta(device) = “demo”

temperature	humidity
30	20

2) 使用 meta 函数的几个例子:

- meta(created): 000 //Get ‘created’ metadata from Event structure
- meta(temperature -> created): 123 //Get ‘created’ metadata from reading[0], key with ‘temperature’
- meta(humidity -> created): 456 //Get ‘created’ metadata from reading[1], key with ‘humidity’

EdgeX 源使用例子

1) 使用 ZeroMQ 消息总线:

create stream events() WITH (FORMAT="JSON", TYPE="edgex")

etc/sources/edgex.yaml:

```
default:
  protocol: tcp
  server: edgex-app-service-configurable-rules
  port: 5566
  topic: events
  serviceServer: http://localhost:48080
```

2) 使用 MQTT 消息总线:

create stream demo () WITH (FORMAT="JSON", TYPE="edgex" Conf_key="mqtt_conf")

etc/sources/edgex.yaml:

```
mqtt_conf: #Conf_key
  protocol: tcp
  server: 127.0.0.1
  port: 1883
  topic: events
  type: mqtt
  optional:
    ClientId: "client1"
```

EdgeX sink

- 往 EdgeX 消息总线发送符合规格的数据
- 元数据保留方式
  - 发布结果到 EdgeX 消息总线，而不保留原有的元数据：Kuiper 在此情况下作为 EdgeX 的一个单独微服务，它有自己的 device name。 提供了属性 deviceName， 该属性允许用户指定 Kuiper 的设备名称
  - 发布结果到 EdgeX 消息总线，并保留原有的元数据：在此情况下，Kuiper 更像是一个过滤器 – 将不关心的数据过滤掉，但是依然保留原有的数据。

EdgeX sink 使用例子

```
{
  "id": "rule1",
  "sql": "SELECT temperature, humidity, meta(*) AS metadt FROM 
demo WHERE temperature = 72",
  "actions": [
    {
      "edgex": {
        "protocol": "tcp",
        "host": "*",
        "port": 5571,
        "topic": "application",
        "metadata": "metadt",
        "contentType": "application/json"
      }
    }
  ]
}
```

```
{
  "id": "rule1",
  "sql": "SELECT meta(*) AS edgex_meta, temperature, humidity, 
humidity*2 as h1 FROM demo WHERE temperature = 20",
  "actions": [
    {
      "edgex": {
        "protocol": "tcp",
        "host": "${mqtt_srv}",
        "port": 1883,
        "topic": "result",
        "type": "mqtt",
        "metadata": "edgex_meta",
        "contentType": "application/json",
        "optional": {
                "ClientId": "edgex_message_bus_001"
        }
      }
    },
    {
      "log":{}
    }
  ]
}
```

Kuiper 性能 – 支持规则数目的测试

- 8000 规则 * 0.1 消息/秒/规则, 共计的 TPS 为 800 条/秒

- 规则定义
  - 源: MQTT
  - SQL: select temperature from source where temperature > 20 (90% 数据被过滤)
  - 目标: 日志

- 配置
  - AWS: 2core * 4GB
  - Ubuntu

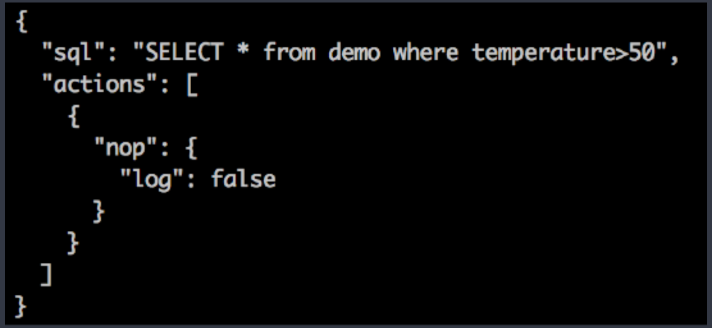
- 资源使用
  - Memory: 89% ~ 72%; 0.4 MB/rule
  - CPU: 25%

性能基准测试

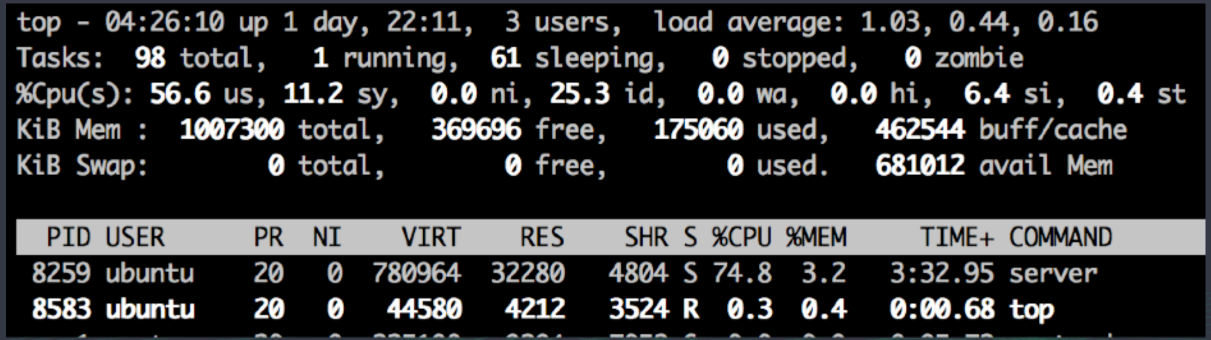
- 配置
  - AWS t2.micro(1 Core * 1 GB)
  - Ubuntu 18.04

- 场景
  - 一个 Go 应用用于往 ZeroMQ 消息总线发送数据，格式如下：
    { "device":"demo",
      "readings": [
        {"device":"Temperature device","name":"Temperature","value":"10"},
        {"device":"Humidity device”,”name":"Humidity","value":"15"}]}
    }
  - 规则如下
    - A nop sink – 忽略发送到该 sink 的所有数据
    - 过滤 90% 数据

- 结果
  - 11434 message/秒
  - 75% CPU (user + sys)
  - 内存: 32MB



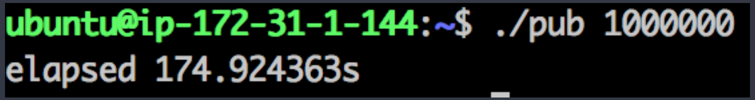
```
{
  "sql": "SELECT * from demo where temperature>50",
  "actions": [
    {
      "nop": {
        "log": false
      }
    }
  ]
}
```



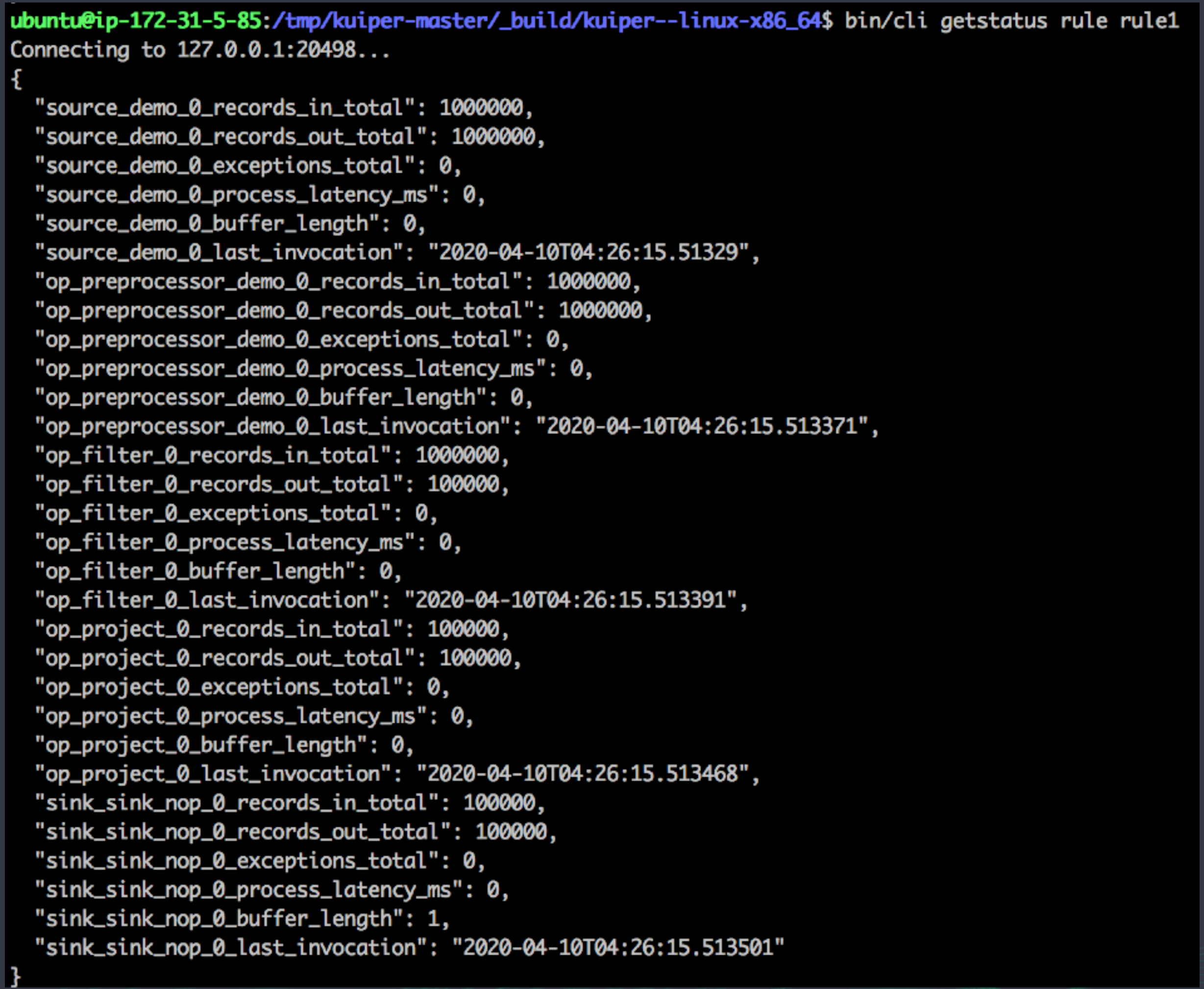
```
top - 04:26:10 up 1 day, 22:11,  3 users,  load average: 1.03, 0.44, 0.16
Tasks:  98 total,   1 running,  61 sleeping,   0 stopped,   0 zombie
%Cpu(s): 56.6 us, 11.2 sy,  0.0 ni, 25.3 id,  0.0 wa,  0.0 hi,  6.4 si,  0.4 st
KiB Mem :  1007300 total,   369696 free,   175060 used,   462544 buff/cache
KiB Swap:        0 total,        0 free,        0 used.   681012 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 8259 ubuntu    20   0  780964  32280   4804 S 74.8  3.2   3:32.95 server
 8583 ubuntu    20   0   44580   4212   3524 R  0.3  0.4   0:00.68 top
```

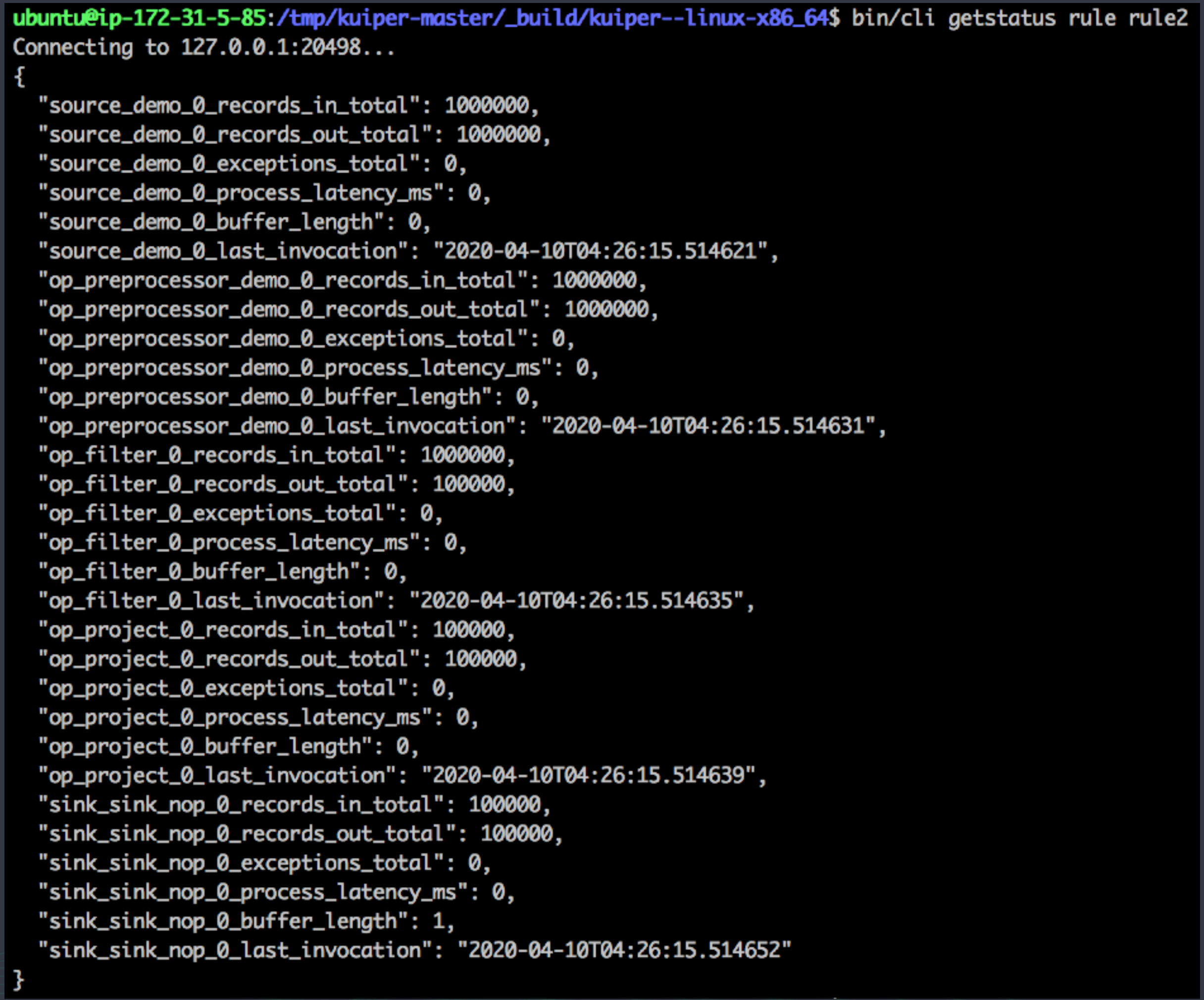
测试状态截屏



```
ubuntu@ip-172-31-1-144:~$ ./pub 1000000
elapsed 174.924363s
```



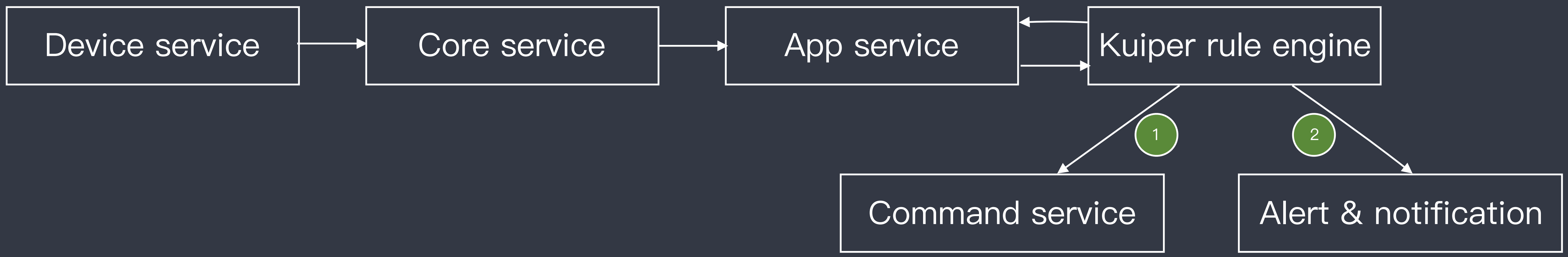
```
ubuntu@ip-172-31-5-85:/tmp/kuiper-master/_build/kuiper--linux-x86_64$ bin/cli getstatus rule rule1
Connecting to 127.0.0.1:20498...
{
  "source_demo_0_records_in_total": 1000000,
  "source_demo_0_records_out_total": 1000000,
  "source_demo_0_exceptions_total": 0,
  "source_demo_0_process_latency_ms": 0,
  "source_demo_0_buffer_length": 0,
  "source_demo_0_last_invocation": "2020-04-10T04:26:15.51329",
  "op_preprocessor_demo_0_records_in_total": 1000000,
  "op_preprocessor_demo_0_records_out_total": 1000000,
  "op_preprocessor_demo_0_exceptions_total": 0,
  "op_preprocessor_demo_0_process_latency_ms": 0,
  "op_preprocessor_demo_0_buffer_length": 0,
  "op_preprocessor_demo_0_last_invocation": "2020-04-10T04:26:15.513371",
  "op_filter_0_records_in_total": 1000000,
  "op_filter_0_records_out_total": 100000,
  "op_filter_0_exceptions_total": 0,
  "op_filter_0_process_latency_ms": 0,
  "op_filter_0_buffer_length": 0,
  "op_filter_0_last_invocation": "2020-04-10T04:26:15.513391",
  "op_project_0_records_in_total": 100000,
  "op_project_0_records_out_total": 100000,
  "op_project_0_exceptions_total": 0,
  "op_project_0_process_latency_ms": 0,
  "op_project_0_buffer_length": 0,
  "op_project_0_last_invocation": "2020-04-10T04:26:15.513468",
  "sink_sink_nop_0_records_in_total": 100000,
  "sink_sink_nop_0_records_out_total": 100000,
  "sink_sink_nop_0_exceptions_total": 0,
  "sink_sink_nop_0_process_latency_ms": 0,
  "sink_sink_nop_0_buffer_length": 1,
  "sink_sink_nop_0_last_invocation": "2020-04-10T04:26:15.513501"
}
```



```
ubuntu@ip-172-31-5-85:/tmp/kuiper-master/_build/kuiper--linux-x86_64$ bin/cli getstatus rule rule2
Connecting to 127.0.0.1:20498...
{
  "source_demo_0_records_in_total": 1000000,
  "source_demo_0_records_out_total": 1000000,
  "source_demo_0_exceptions_total": 0,
  "source_demo_0_process_latency_ms": 0,
  "source_demo_0_buffer_length": 0,
  "source_demo_0_last_invocation": "2020-04-10T04:26:15.514621",
  "op_preprocessor_demo_0_records_in_total": 1000000,
  "op_preprocessor_demo_0_records_out_total": 1000000,
  "op_preprocessor_demo_0_exceptions_total": 0,
  "op_preprocessor_demo_0_process_latency_ms": 0,
  "op_preprocessor_demo_0_buffer_length": 0,
  "op_preprocessor_demo_0_last_invocation": "2020-04-10T04:26:15.514631",
  "op_filter_0_records_in_total": 1000000,
  "op_filter_0_records_out_total": 100000,
  "op_filter_0_exceptions_total": 0,
  "op_filter_0_process_latency_ms": 0,
  "op_filter_0_buffer_length": 0,
  "op_filter_0_last_invocation": "2020-04-10T04:26:15.514635",
  "op_project_0_records_in_total": 100000,
  "op_project_0_records_out_total": 100000,
  "op_project_0_exceptions_total": 0,
  "op_project_0_process_latency_ms": 0,
  "op_project_0_buffer_length": 0,
  "op_project_0_last_invocation": "2020-04-10T04:26:15.514639",
  "sink_sink_nop_0_records_in_total": 100000,
  "sink_sink_nop_0_records_out_total": 100000,
  "sink_sink_nop_0_exceptions_total": 0,
  "sink_sink_nop_0_process_latency_ms": 0,
  "sink_sink_nop_0_buffer_length": 1,
  "sink_sink_nop_0_last_invocation": "2020-04-10T04:26:15.514652"
}
```

EdgeX Kuiper 规则使用场景

- Kuiper 规则引擎的可能输出
  - Command service: send control command (rest API)
  - Alert & notification: send alert & notification (rest API)



2020 规划

- EdgeX Hanoi Release
  - 数组类型支持
  - Sink 数据发送的时候提供更多的灵活性（Golang data template）
  - UI 提升
- 与 KubeEdge 集成
- 我们非常乐意看到来自于社区的新需求！
- 欢迎加入我们的项目！

- 规划链接：https://github.com/emqx/kuiper/projects/1

文档

- 项目网址：https://github.com/emqx/kuiper
- Docker：https://hub.docker.com/r/emqx/kuiper
- EdgeX Kuiper 使用教程：https://github.com/emqx/kuiper/blob/master/docs/zh_CN/edgex/edgex_rule_engine_tutorial.md
- 插件开发教程：https://github.com/emqx/kuiper/blob/master/docs/zh_CN/plugins/plugins_tutorial.md

Thank You

contact@emqx.io