IoT meets Serverless

Narendran

A bit about me

Naren

Backend/Product Engineer

Scaling A.I to millions @ MadStreetDen

python, golang, FOSS, cycling, travel

twitter :
@DudeWhoCode

www.dudewho.codes

IoT

(buzzword #1)

“things” in Internet of Things

IoT is really really a

BIG data

(buzzword #2)

- 20 billion connected devices currently

- 400 ZetaBytes of data by 2018

ref : cisco

sensors in Boeing 787 generates

of data in an hour of flight

On one fine day …

manager : Hey, you are going to work on a new project.

On one fine day …

manager : Hey, you are going to work on a new project.

you : what is it about?

On one fine day …

manager : Hey, you are going to work on a new project.

you : what is it about?

manager : You are going to do a PoC to get insights from IoT data

On one fine day …

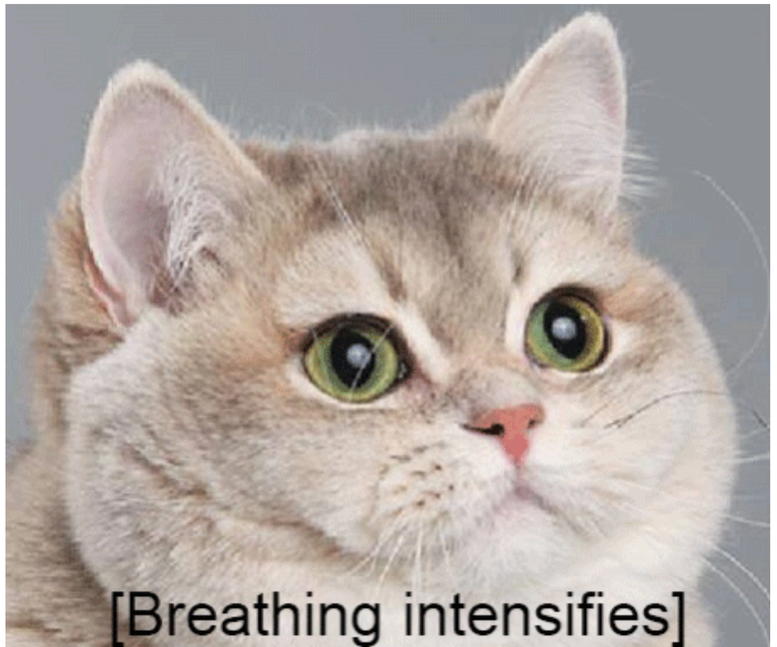
manager : Hey, you are going to work on a new project.

you : what is it about?

manager : You are going to do a PoC to get insights from IoT data

you : wow, IoT?

you :



On one fine day …

manager : The client will sign the contract if we show some insights within few days.

Crunching the IoT data

3 tier architecture

Presentation tier

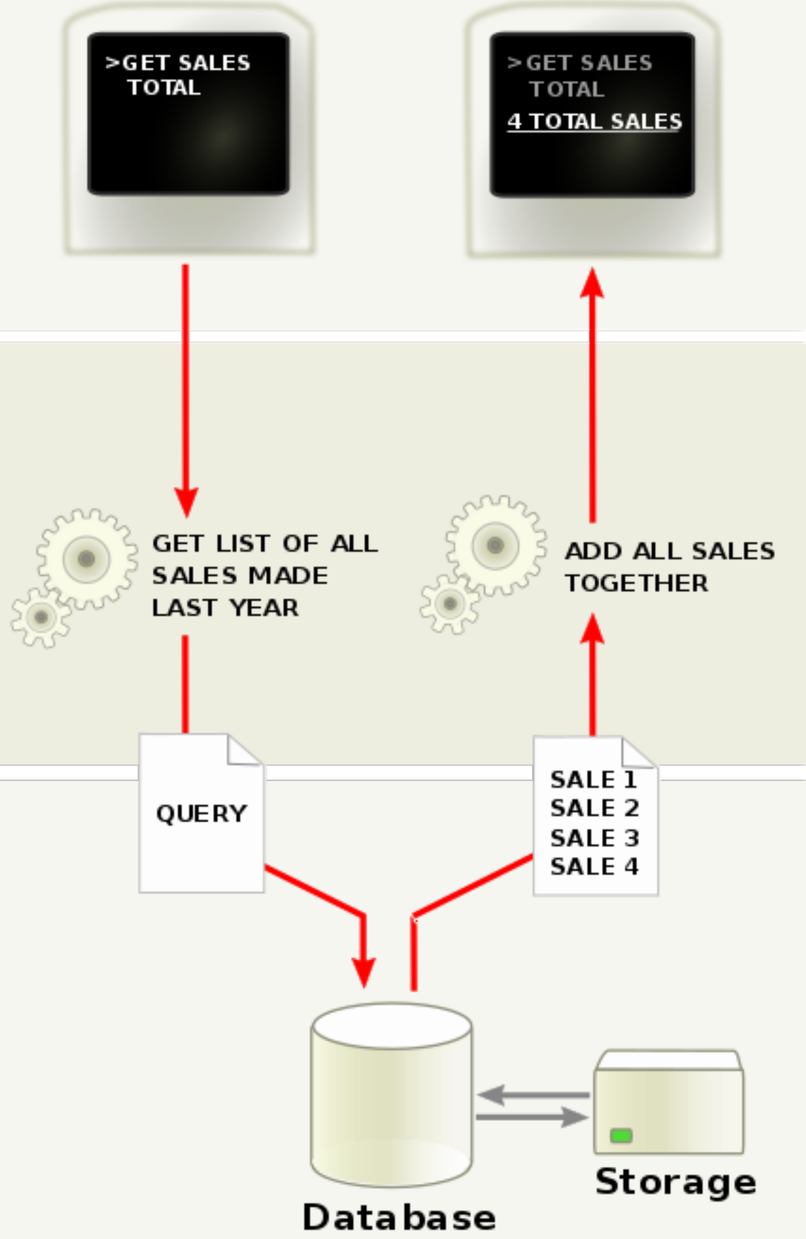
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

Logic tier

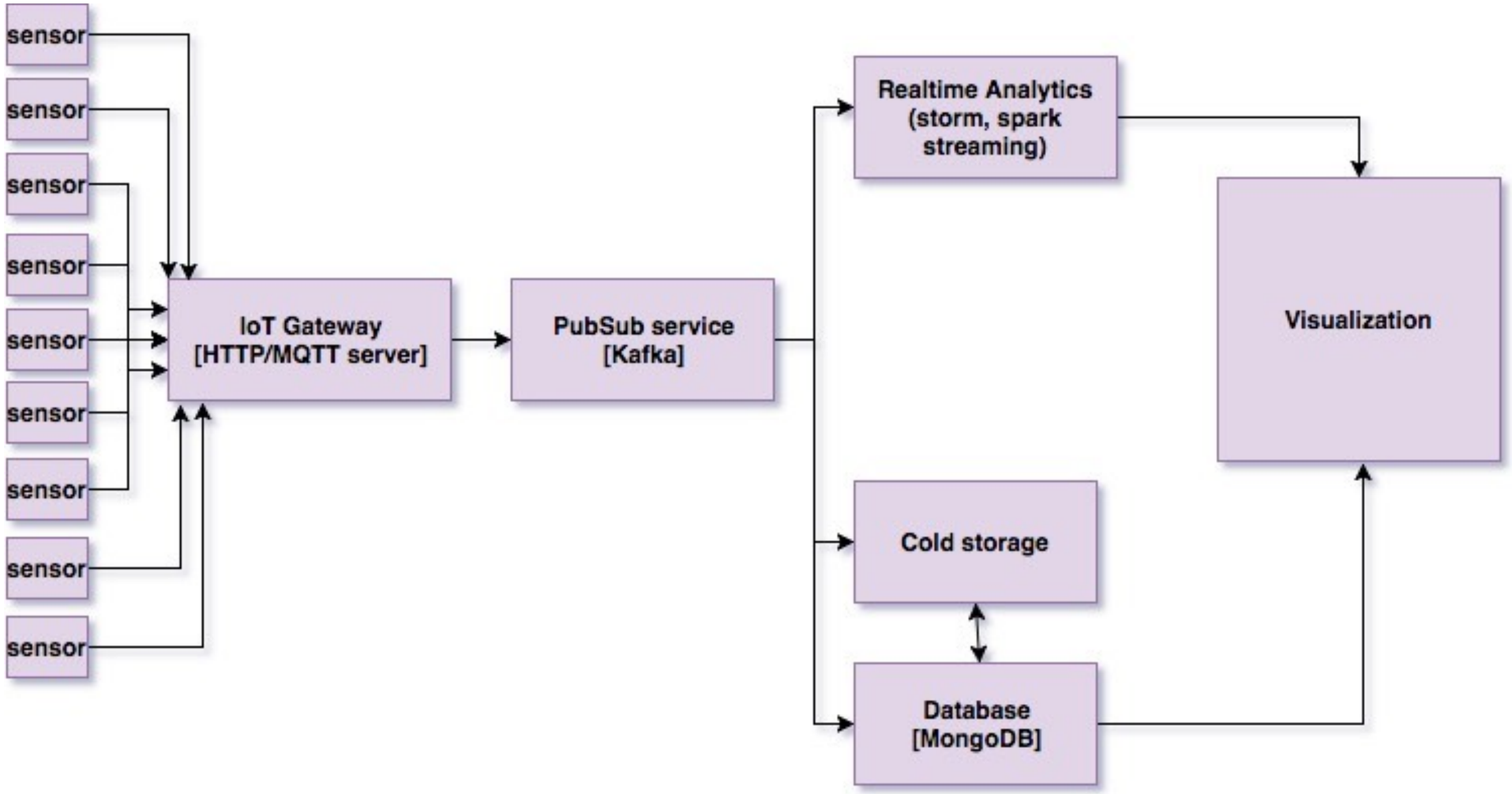
This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier

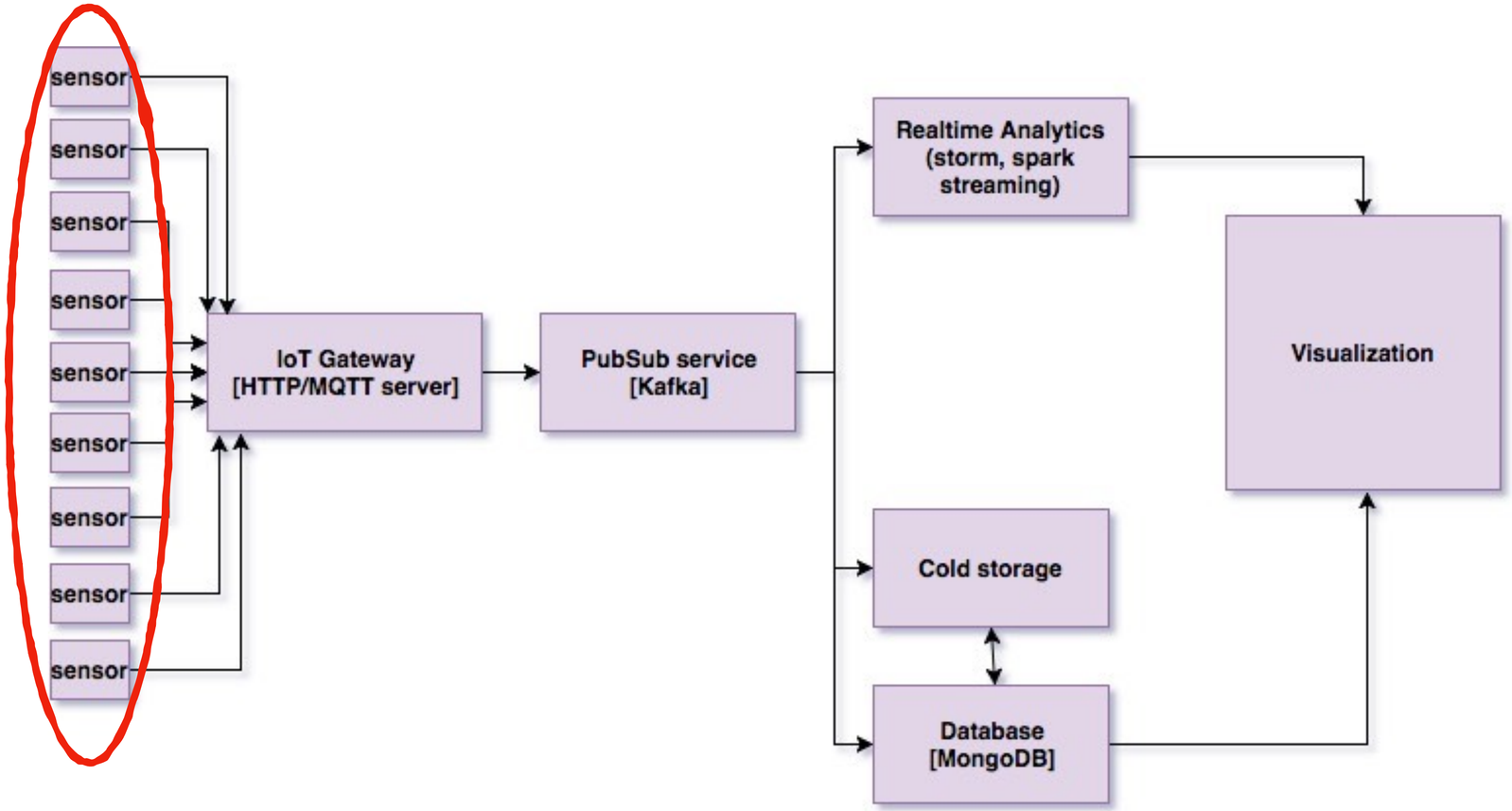
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.



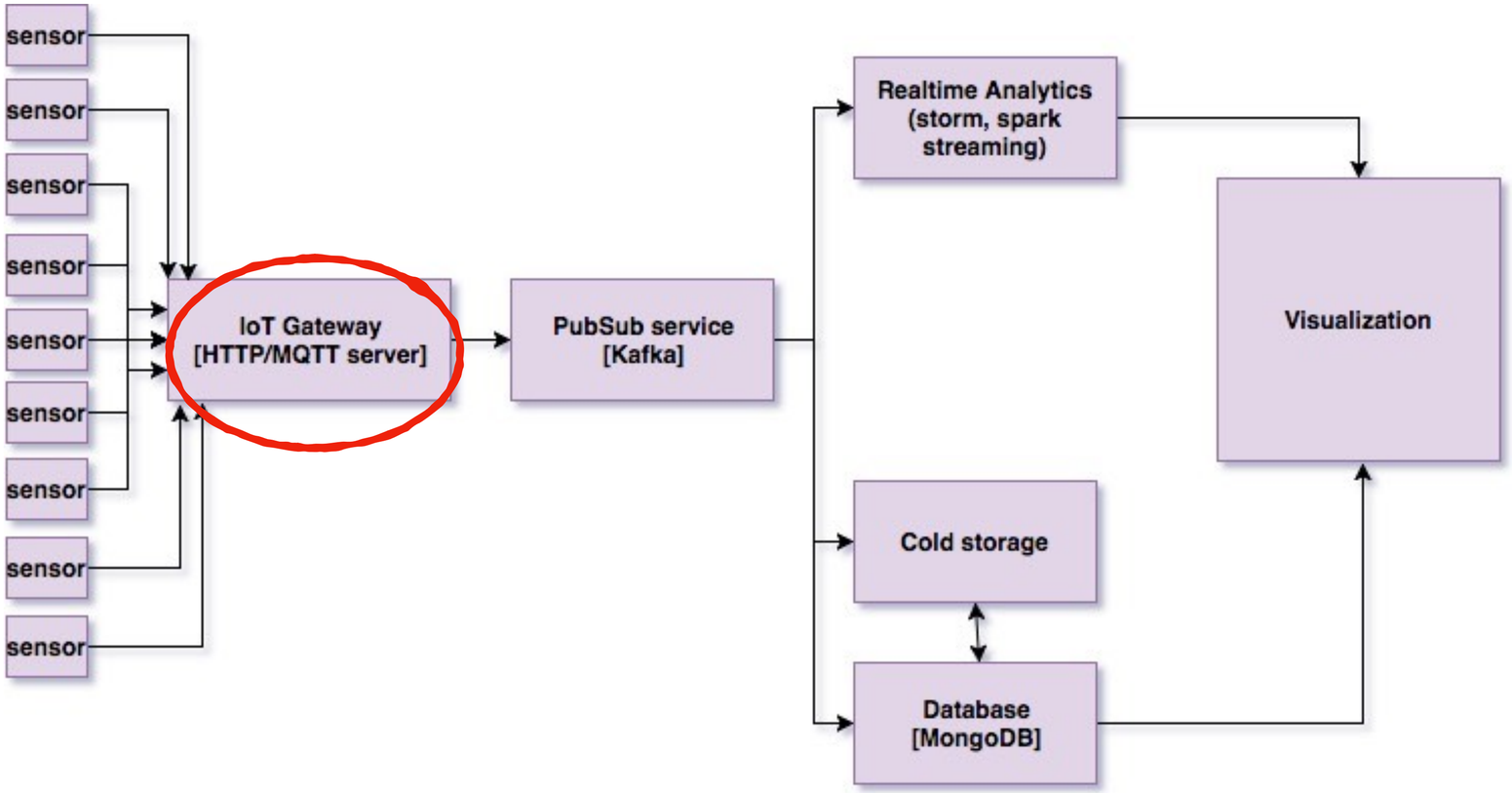
Traditional Implementation



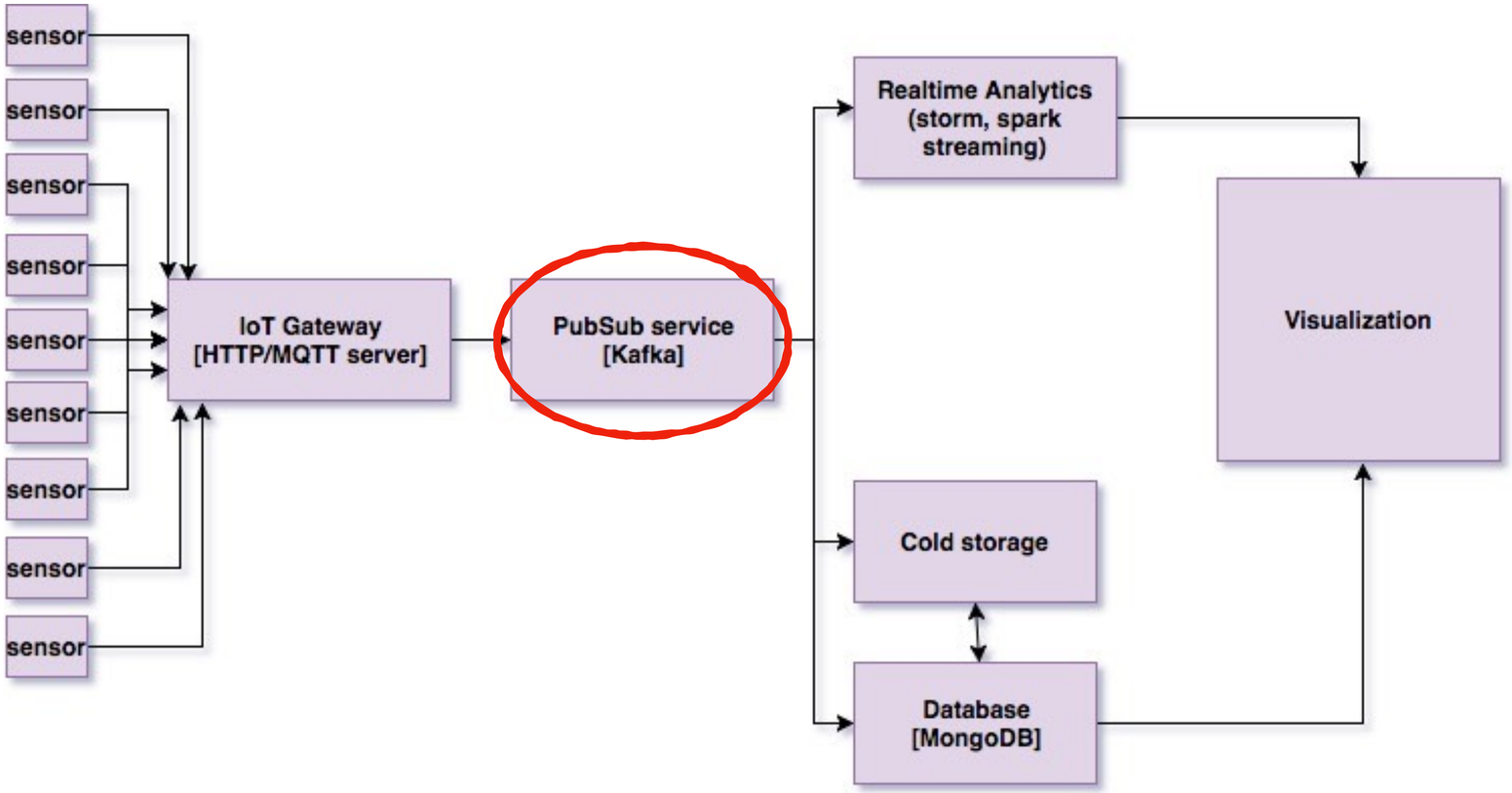
- Connected devices
- Heart rate sensors



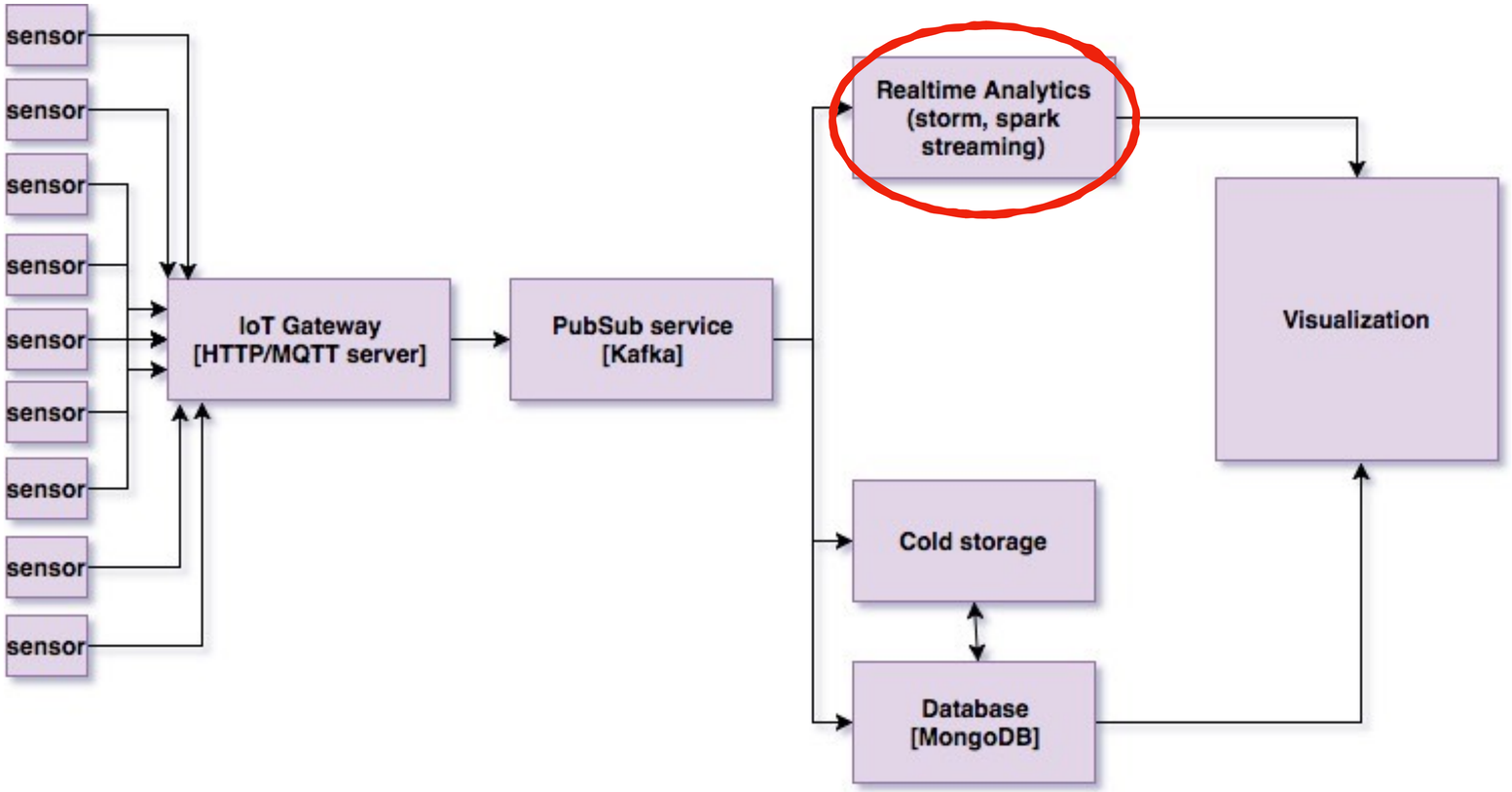
- Django server behind load balancer
- Authentication of registered IoT devices
- Scaling when the throughput or sensor count increases



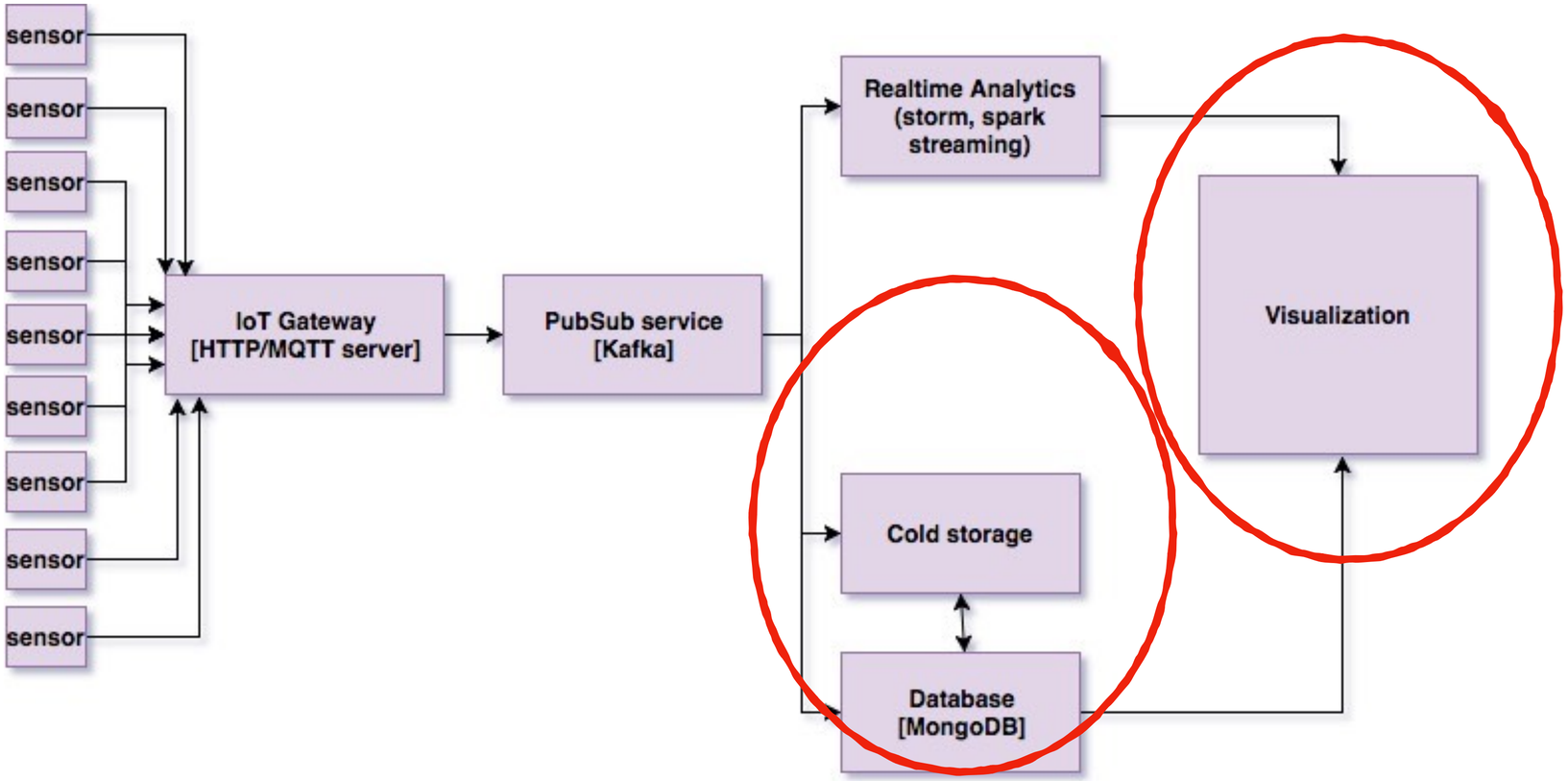
- Kafka
- Cluster/node maintainance
- Producers, Consumers, Nodes, Partitions, ZooKeepers, offsets



- Apache Storm
- Cluster/node maintainance
- Dedicated resources



- ElasticSearch and Kibana
- High physical resource and cluster maintenance for high throughput




ONe
WEEk
LAteR...

manager : We lost our contract ..!

Everything boils down to this …

Everything boils down to this …
servers, servers, servers

Everything boils down to this …
servers, servers, servers

servers, servers, servers

- Operational overhead :
  - create and manage machines/VMs
  - Patching O.S, web servers
- Not so easy while implementing microservices
- Auto scaling, DevOps

Going serverless

(buzzword #3)

No server is easier to manage than “no server”

- Werner Wogels, CTO, Amazon

Going serverless

Why serverless?

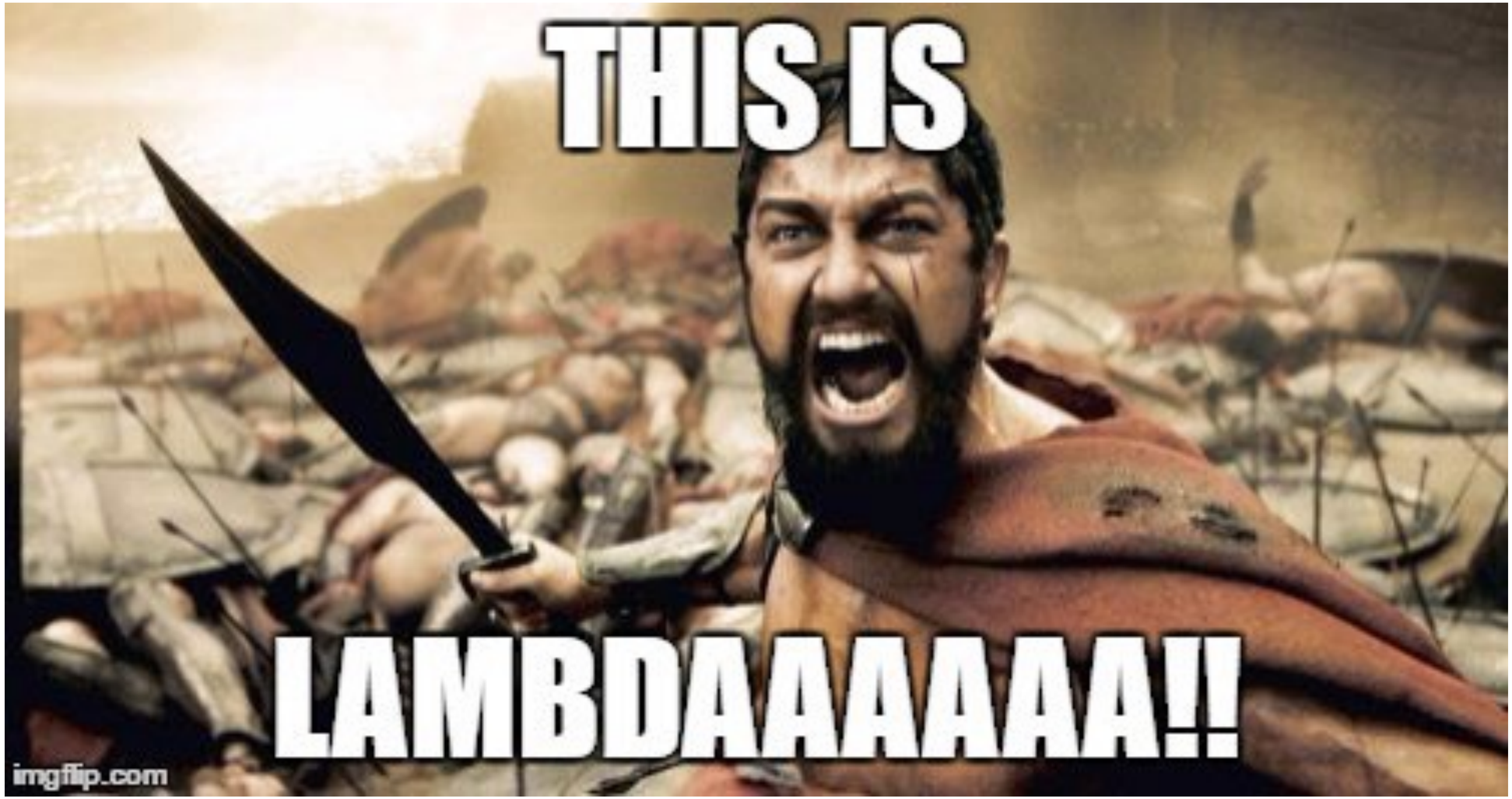
- Low operational overhead
  - No versioning issues
- Event driven microservices
- Stateless
- Don’t pay for idle time

Lets implement similar architecture but without servers

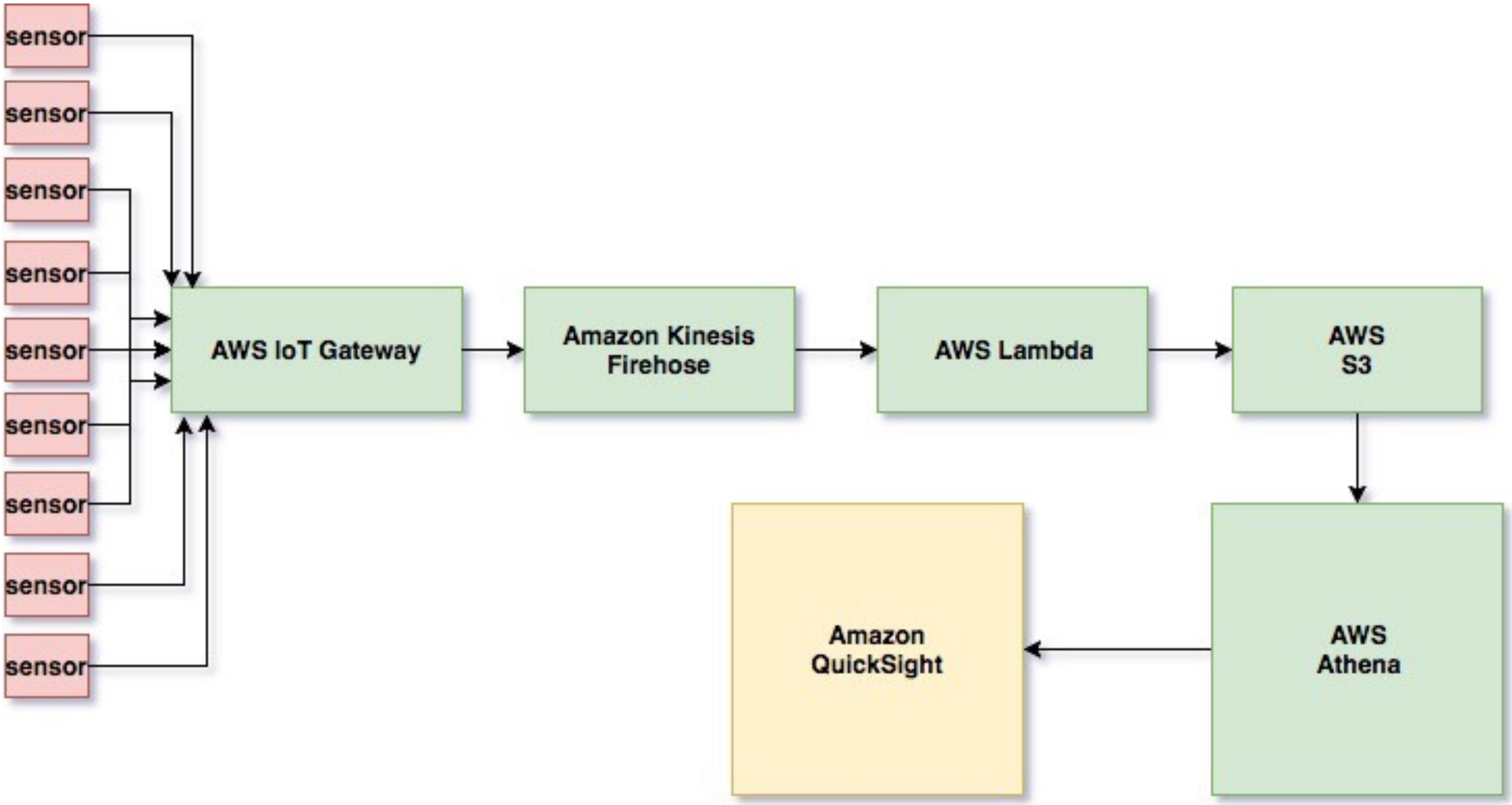
Amazon Web Services (AWS)

AWS serverless

AWS serverless

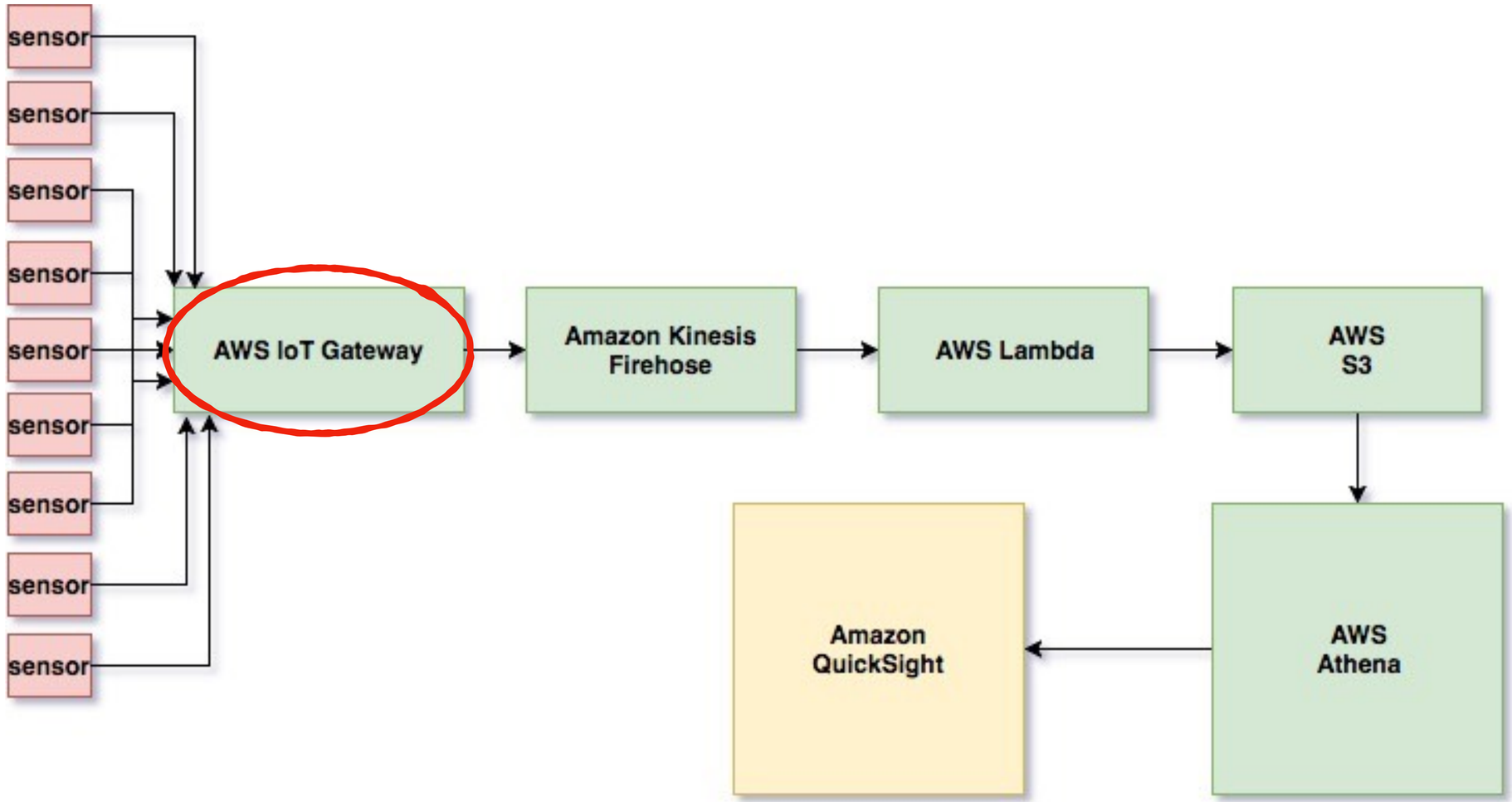


FaaS


sensor
sensor
sensor
sensor
sensor
sensor
sensor
sensor
AWS IoT Gateway
Amazon Kinesis Firehose
AWS Lambda
AWS S3
AWS Athena
Amazon QuickSight

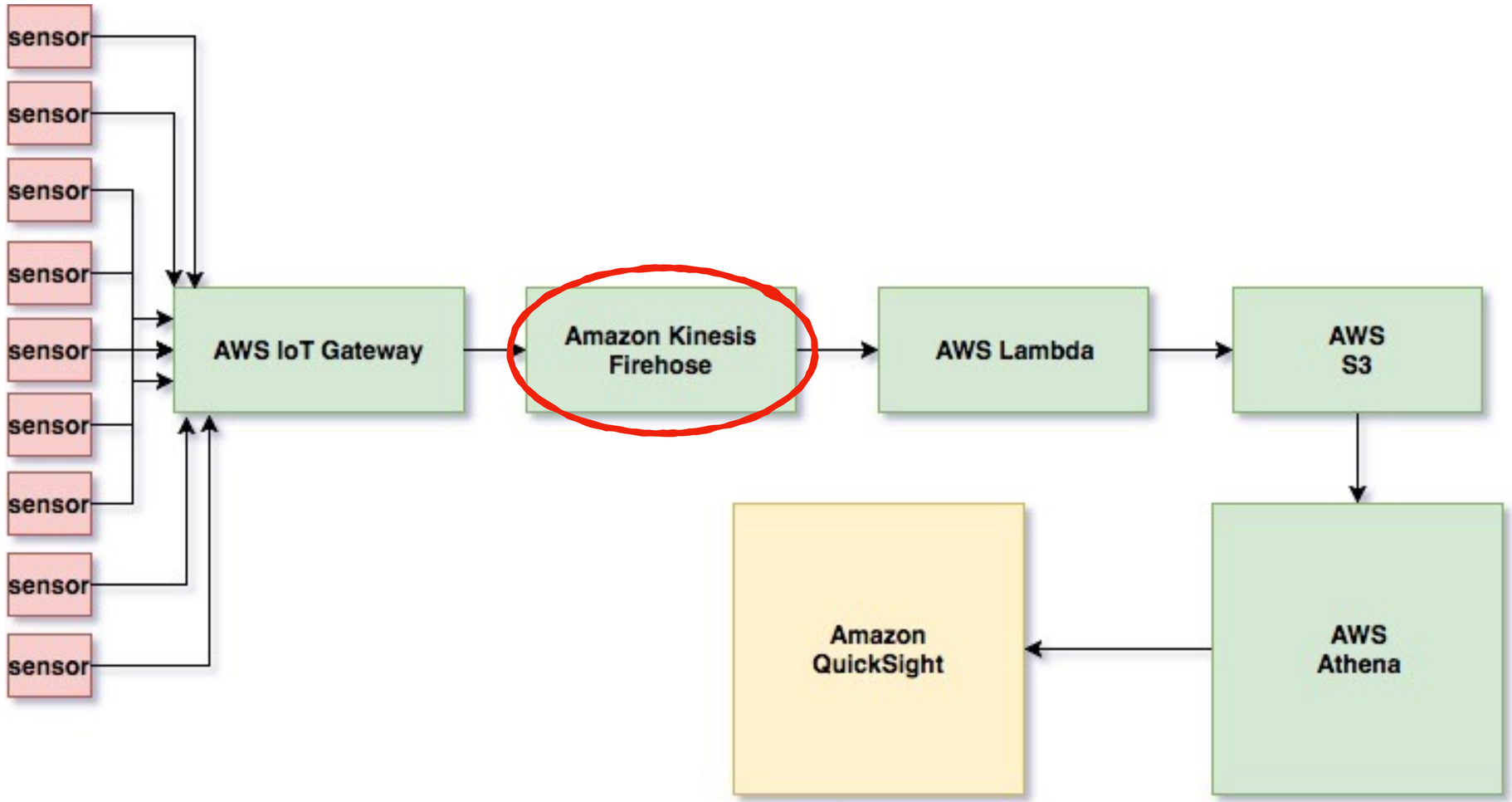
- Securely connect devices and interact with cloud applications

- Device gateway, Message broker, Rules engine

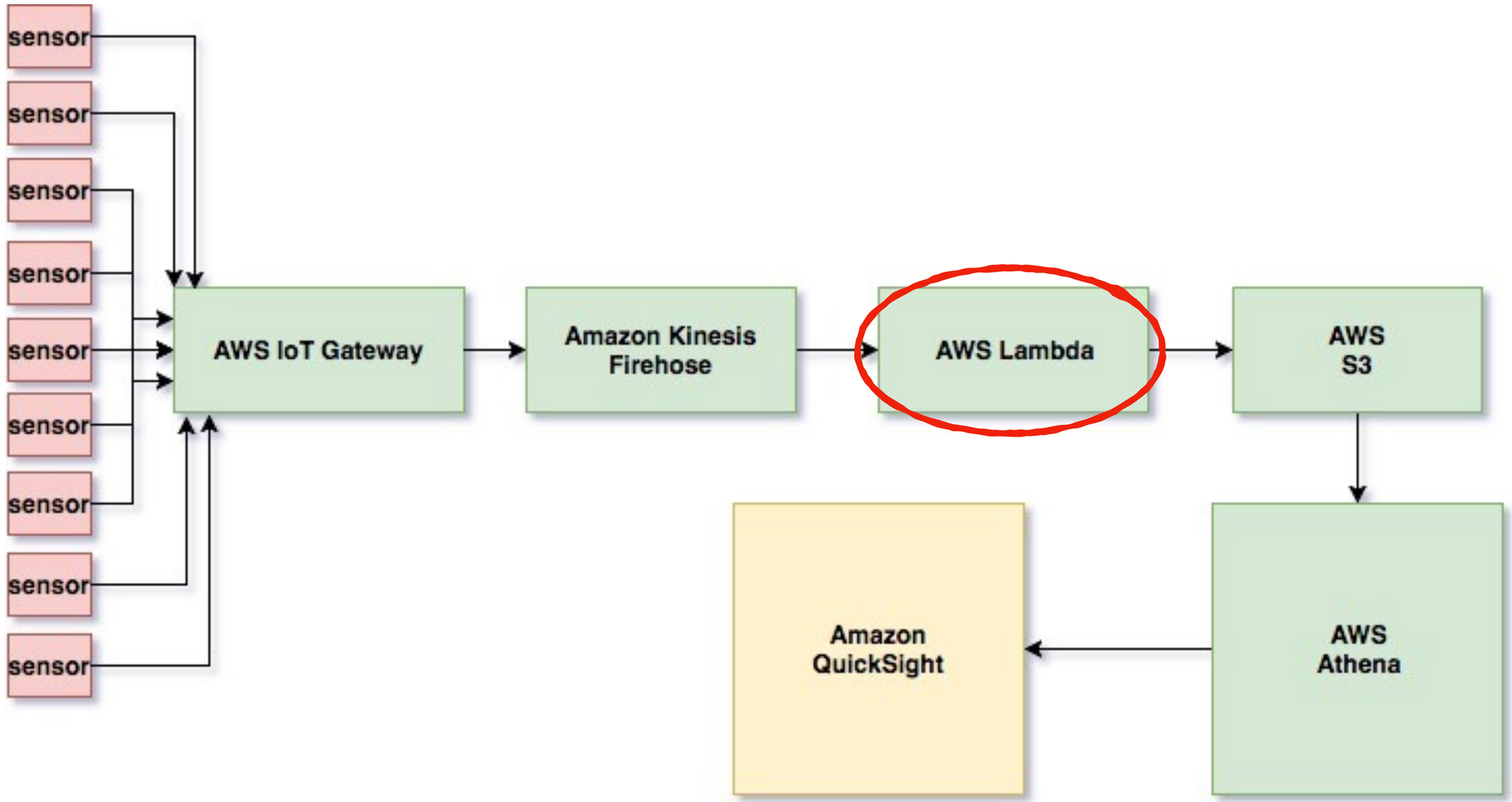


- Delivers real-time streaming data to other services such as Amazon S3, Elastic Search.

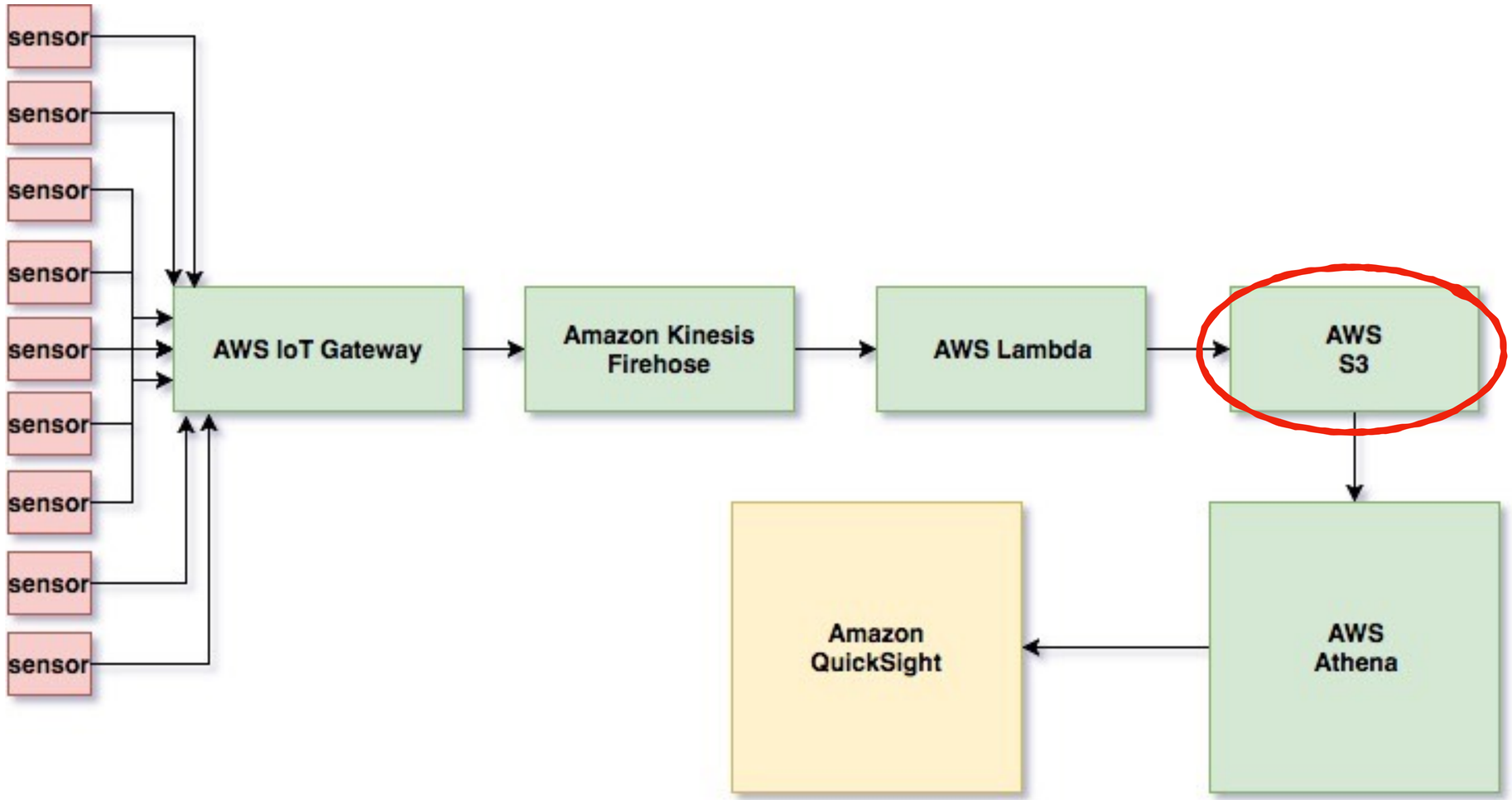
- Configurable producers and consumers



- Run code without provisioning servers
- Event driven, Highly scalable

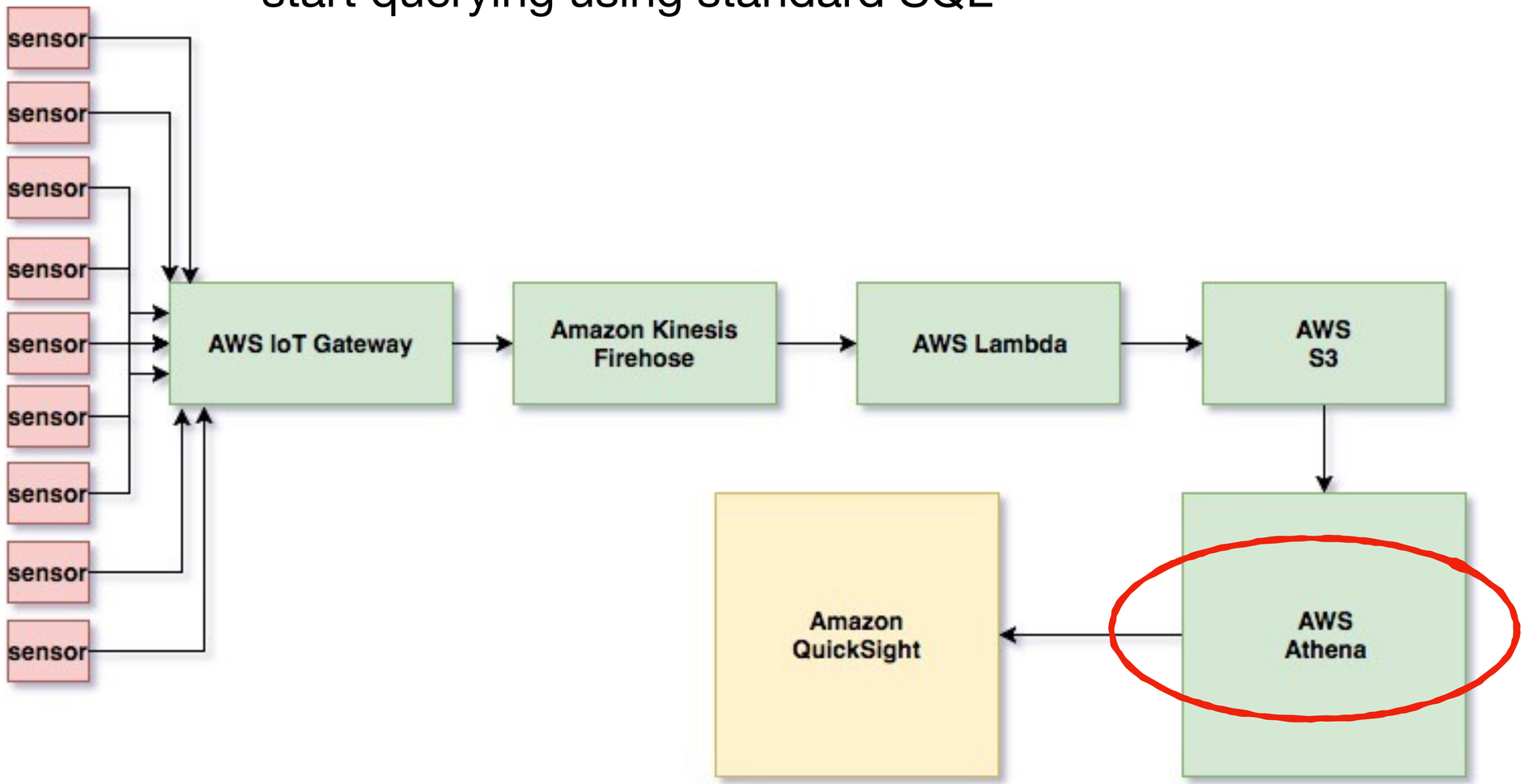


- Store data at massive scale
- Storage tier for many serverless applications

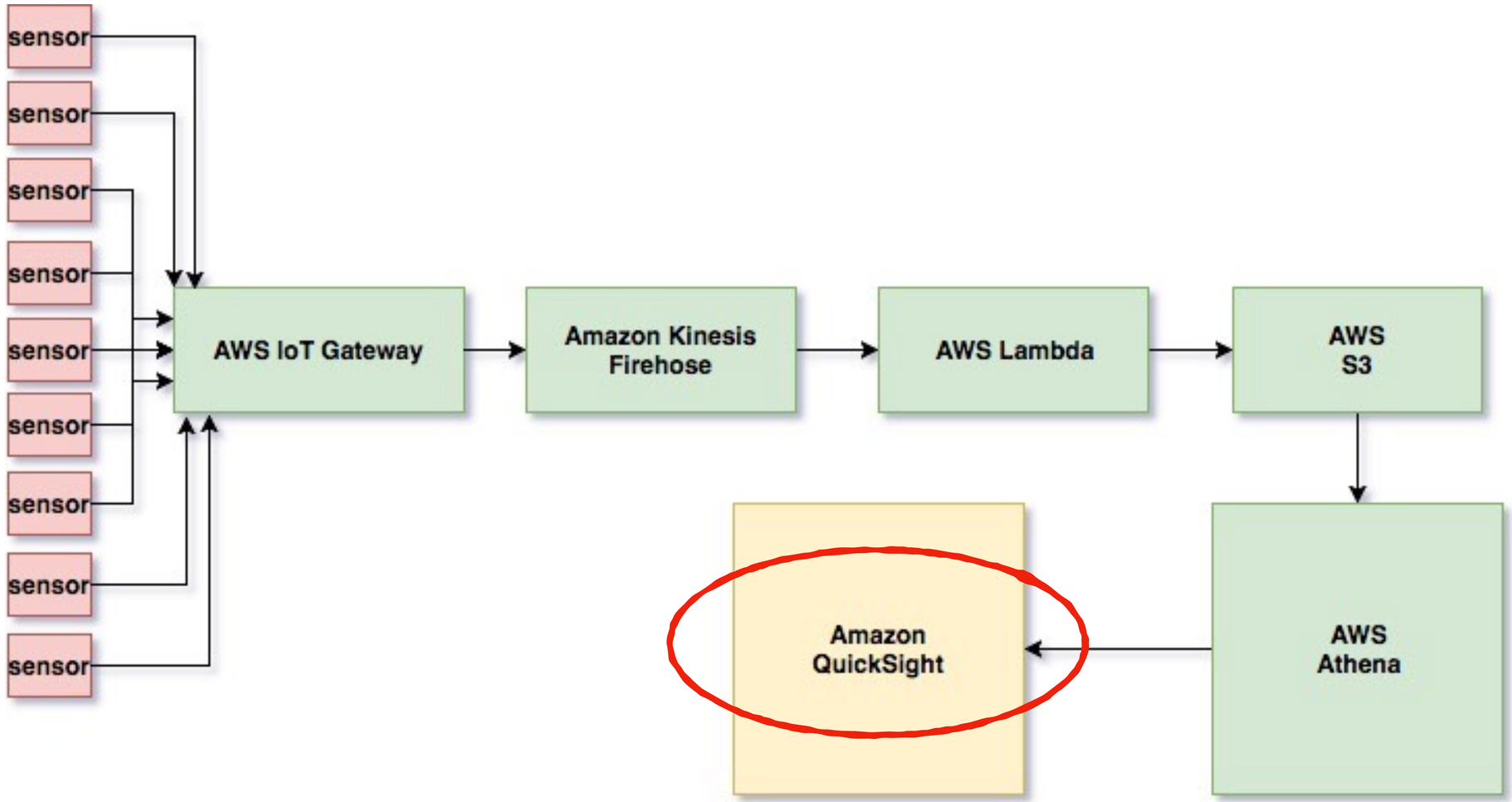


- Serverless interactive query service

- Point to your data source,
  define the schema,
  start querying using standard SQL



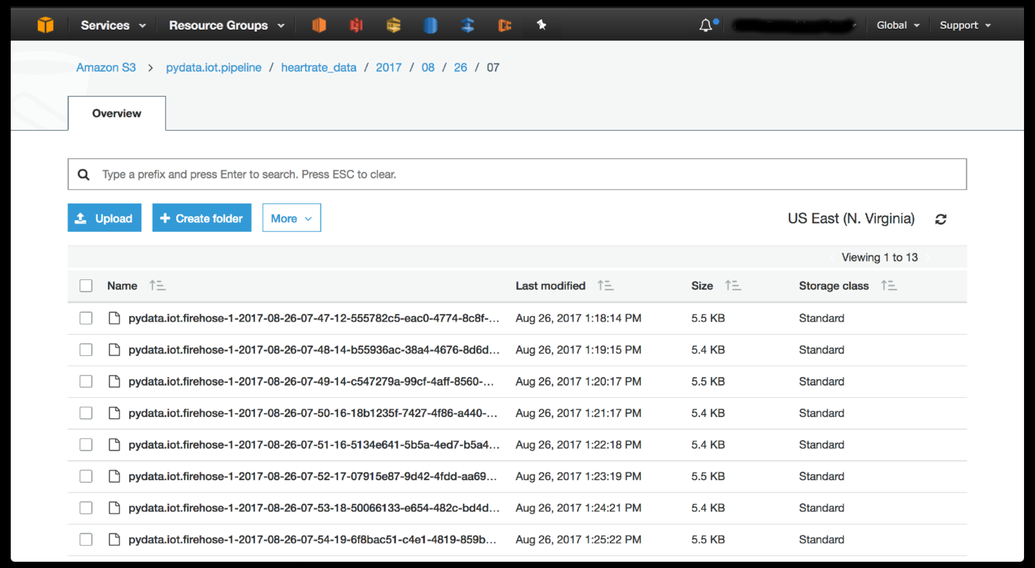
- cloud-powered visualization tool
- Perform ad-hoc analysis and get quick business insights



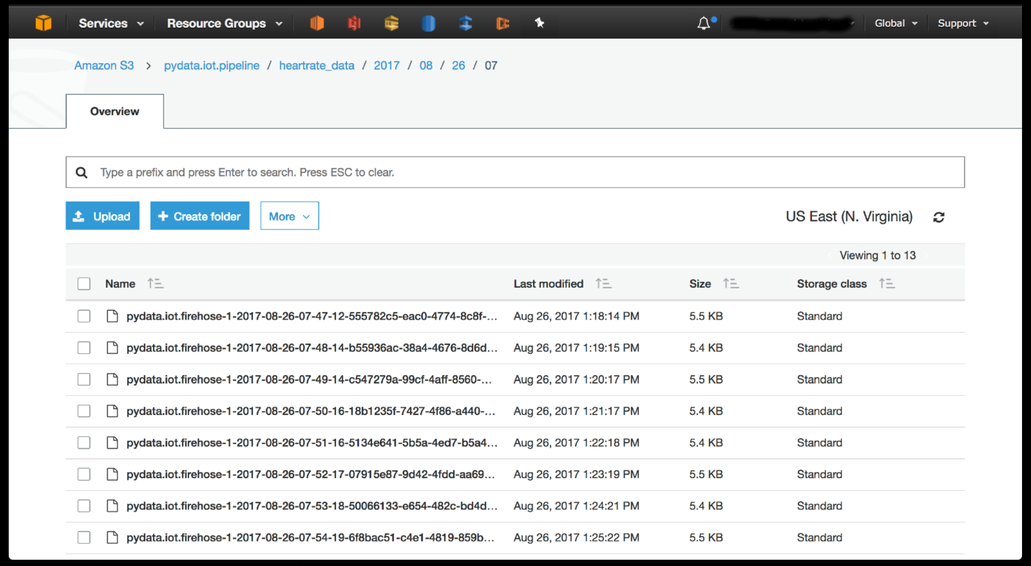
Managing our serverless model

1. Web Console
2. Python code (boto3)

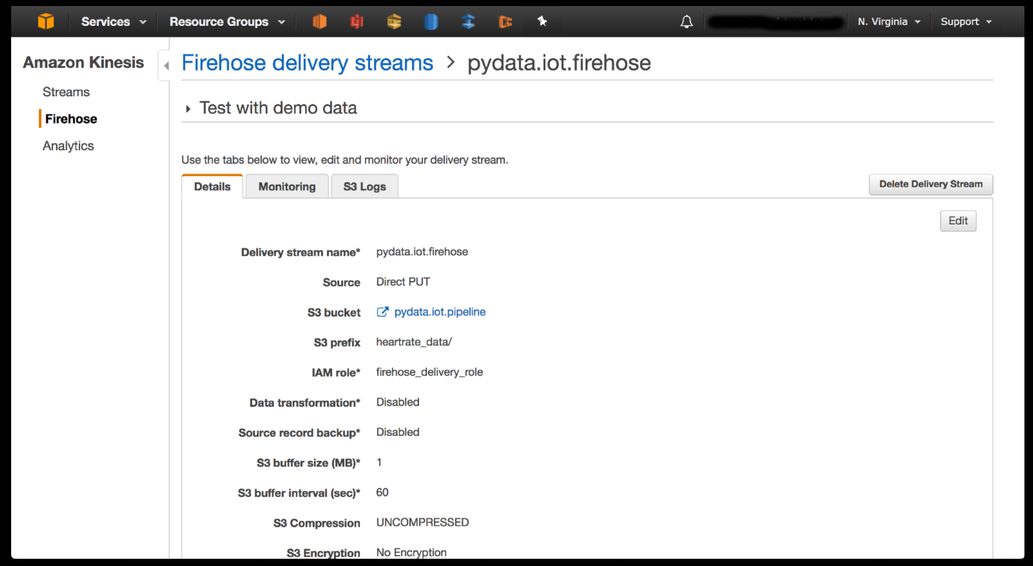
s3


Services
Resource Groups
Global
Support
Amazon S3 > pydata.iot.pipeline / heartrate_data / 2017 / 08 / 26 / 07
Overview
Type a prefix and press Enter to search. Press ESC to clear.
Upload
Create folder
More
US East (N. Virginia)
Viewing 1 to 13
Name
Last modified
Size
Storage class
pydata.iot.firehose-1-2017-08-26-07-47-12-555782c5-eac0-4774-8c8f-... Aug 26, 2017 1:18:14 PM 5.5 KB Standard
pydata.iot.firehose-1-2017-08-26-07-48-14-b55936ac-38a4-4676-8d6d... Aug 26, 2017 1:19:15 PM 5.4 KB Standard
pydata.iot.firehose-1-2017-08-26-07-49-14-c547279a-99cf-4aff-8560-... Aug 26, 2017 1:20:17 PM 5.5 KB Standard
pydata.iot.firehose-1-2017-08-26-07-50-16-18b1235f-7427-4f86-a440-... Aug 26, 2017 1:21:17 PM 5.4 KB Standard
pydata.iot.firehose-1-2017-08-26-07-51-16-5134e641-5b5a-4ed7-b5a4... Aug 26, 2017 1:22:18 PM 5.4 KB Standard
pydata.iot.firehose-1-2017-08-26-07-52-17-07915e87-9d42-4fdd-aa69... Aug 26, 2017 1:23:19 PM 5.5 KB Standard
pydata.iot.firehose-1-2017-08-26-07-53-18-50066133-e654-482c-bd4d... Aug 26, 2017 1:24:21 PM 5.4 KB Standard
pydata.iot.firehose-1-2017-08-26-07-54-19-6f8bac51-c4e1-4819-859b... Aug 26, 2017 1:25:22 PM 5.5 KB Standard

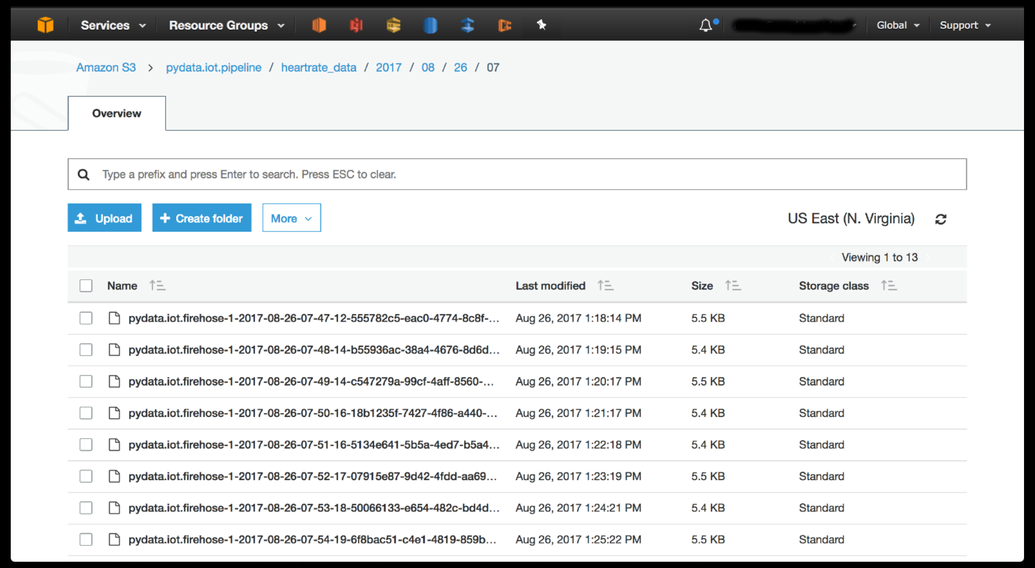
s3



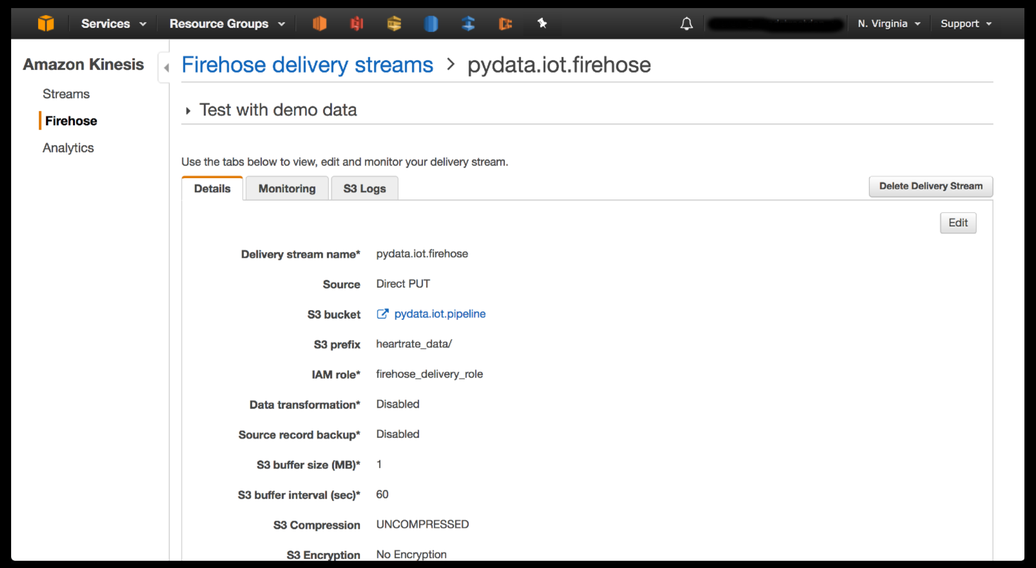
Kinesis Firehose



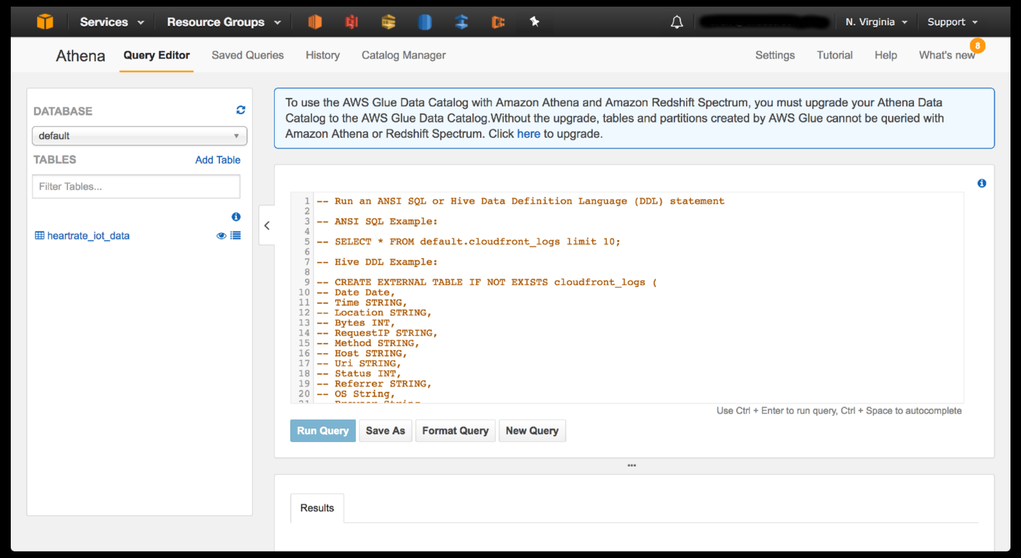
s3



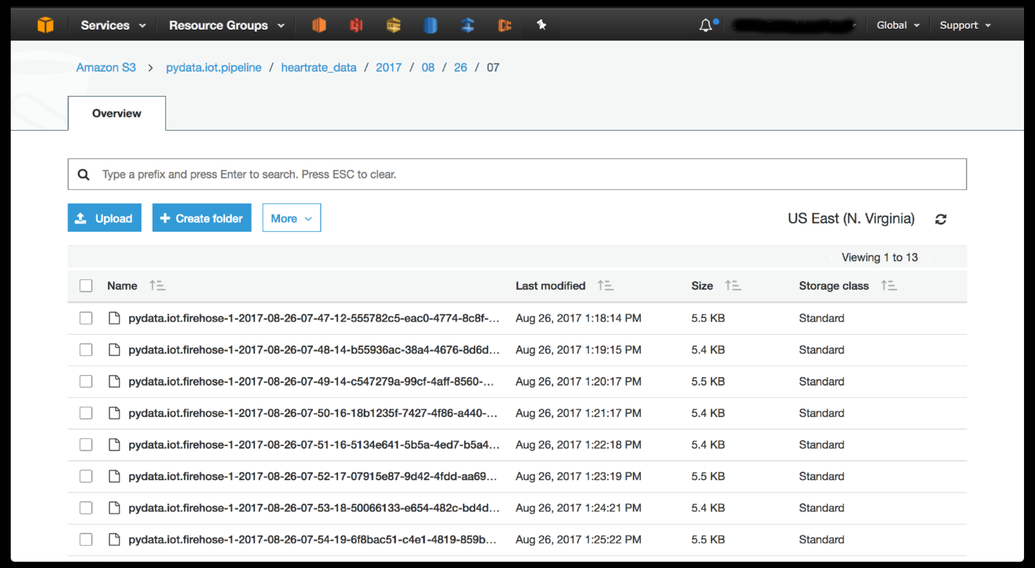
Kinesis Firehose



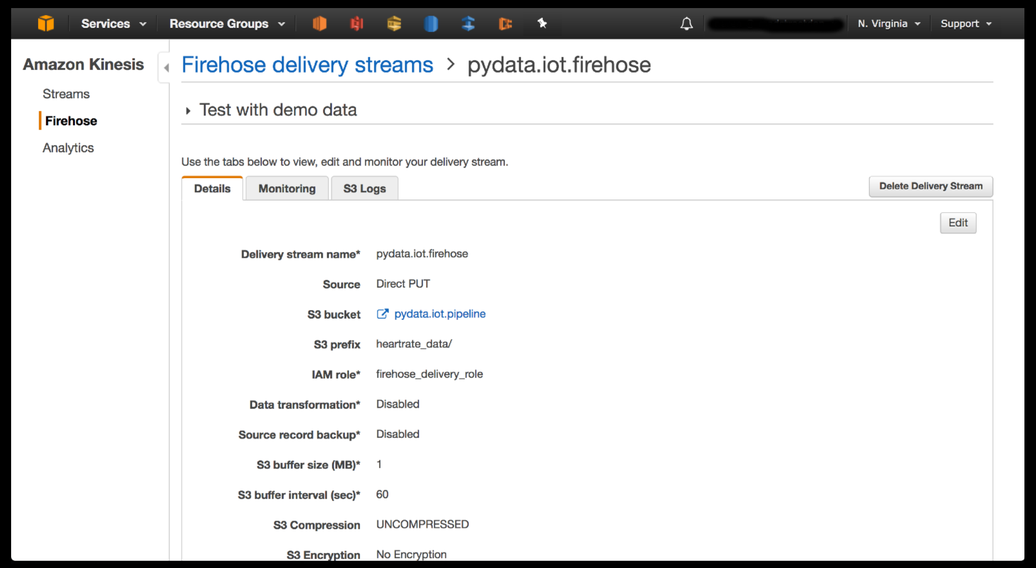
Athena



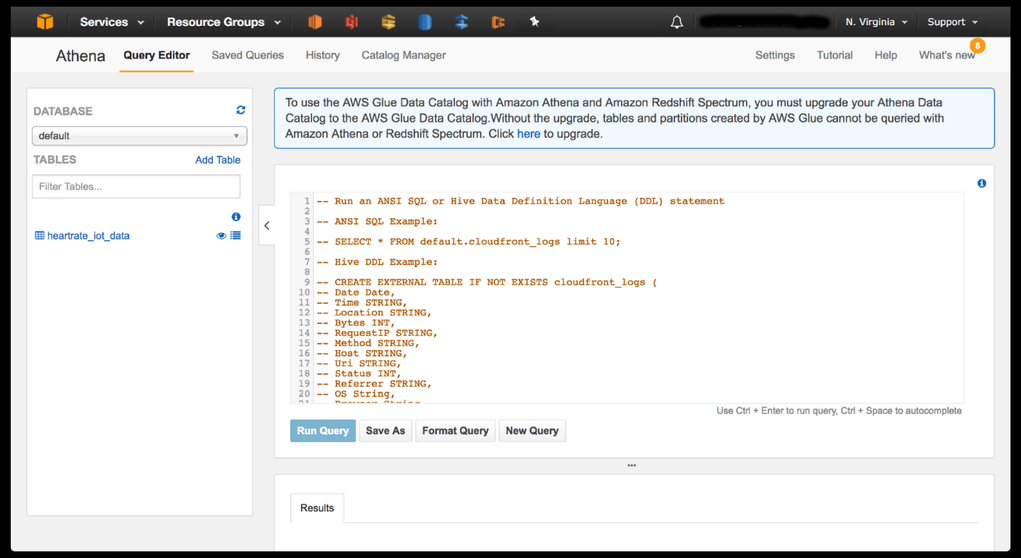
s3



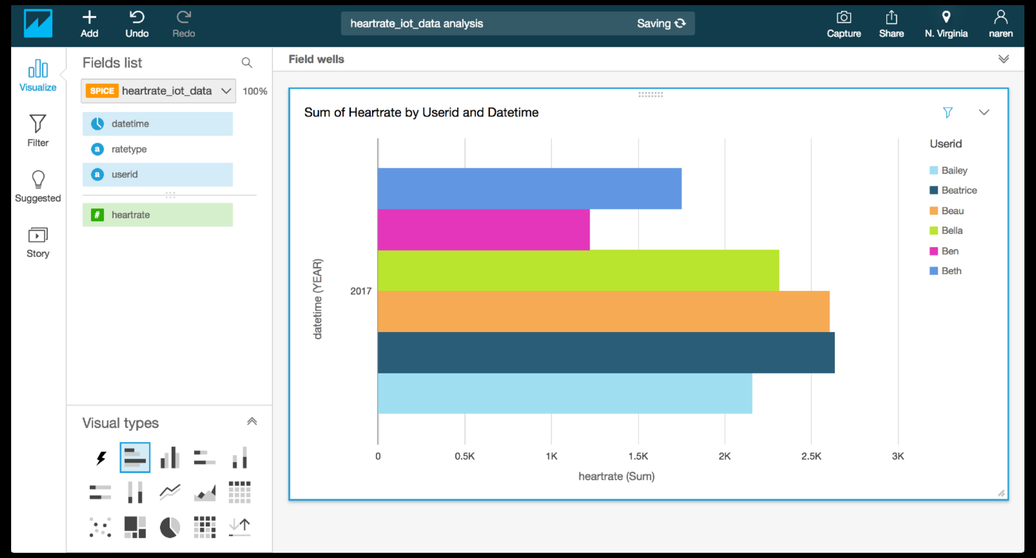
Kinesis Firehose



Athena



QuickSight



Boto3

- Amazon Web Services (AWS) SDK for Python
- Object Oriented API as well as low level access


DEMO

How to do it in code?

create s3 bucket

```
import boto3

s3 = boto3.resource('s3')
bucket = s3.Bucket('pydata.iot.serverless')
response = bucket.create(
    ACL='private',
    CreateBucketConfiguration={
        'LocationConstraint': 'ap-southeast-1'
    },
)
print response
```

create IoT gateway rule

```
import boto3

client = boto3.client('iot')
response = client.create_topic_rule(
    ruleName='IoT_boto3',
    topicRulePayload={
        'sql': 'select * from "/health/#"',
        'description': 'IoT datapipeline',
        'actions': [
            {
                'firehose': {
                    'roleArn': 'arn:aws:iam::1234567890:role/service-role/iot.firehose',
                    'deliveryStreamName': 'iot.kinesis',
                    'separator': '\n'
                }
            },
        ],
        'ruleDisabled': True,
        'awsIotSqlVersion': '2016-03-23'
    }
)
print(response)
```

create firehose delivery stream

```python
import boto3

client = boto3.client('firehose')
response = client.create_delivery_stream(
    DeliveryStreamName='iot_boto3',
    S3DestinationConfiguration={
        'RoleARN': 'arn:aws:iam::1234567890:role/firehose_delivery_role',
        'BucketARN': 'arn:aws:s3:::pydata.iot.pipeline',
        'Prefix': 'string',
        'BufferingHints': {
            'SizeInMBs': 5,
            'IntervalInSeconds': 300
        },
        'CompressionFormat': 'UNCOMPRESSED',
        'EncryptionConfiguration': {
            'NoEncryptionConfig': 'NoEncryption',
        },
    },
)
print(response)

```

athena query : create table

```
import boto3

client = boto3.client('athena')
response = client.create_named_query(
    Name='iot_boto3',
    Description='create table from heart rate data',
    Database='default',
    QueryString="""CREATE EXTERNAL TABLE heartrate_iot_data (
    heartRate int,
    userId string,
    rateType string,
    dateTime timestamp)
    ROW FORMAT  serde 'org.apache.hive.hcatalog.data.JsonSerDe'
    with serdeproperties( 'ignore.malformed.json' = 'true' )
    LOCATION 's3://pydata.iot.pipeline/heartrate_data/'""",
)
print(response)
```

Summary

- Usual approach : takes weeks and certain level of expertise to implement
- Going serverless : Minutes to implement
- AWS services used : AWS IoT, Lambda, Kinesis Firehose, s3, Athena, Quicksight
- Easy to use AWS web console
- Boto3 : the official AWS python SDK

Final thoughts

- Servers :

  DevOps, OnCall, Downtime, Sleepless nights, Angry people

- Serverless :

  Cost efficient, easy to deploy and maintain, all heavy lifting is done by the provider

Caveats

Caveats

Caveats

- Everytime you debug, you need to deploy it in cloud
- Closely packed env, packages/dependencies version are decided by provider
- Local development is difficult
- Vendor lockin

Caveats

-
-
-


I'M OUTTA BULLET POINTS...
ANY QUESTIONS?

slides :
dudewho.codes/talks

Thank you
@DudeWhoCode