

# 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

**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 ...

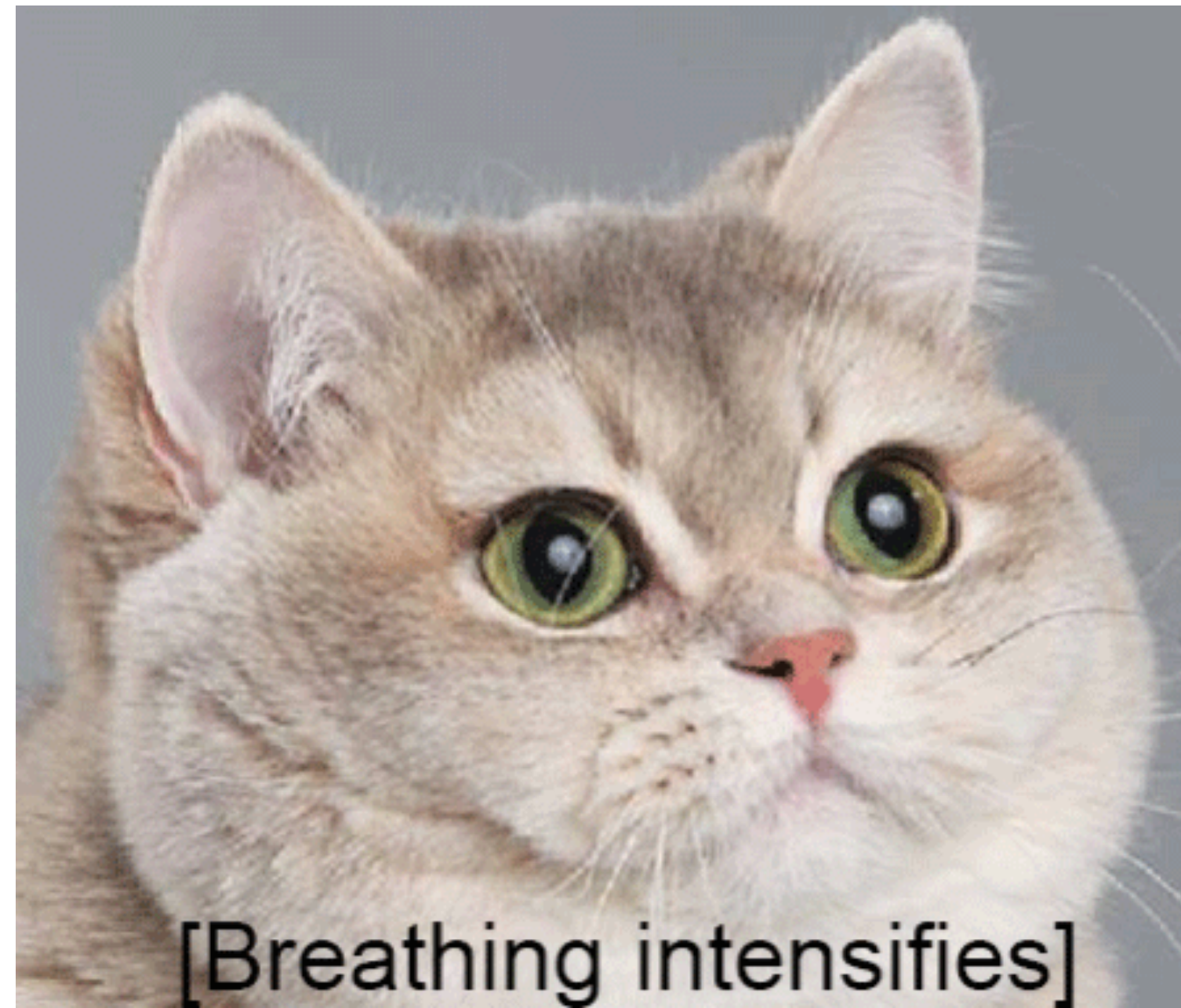
manager : Hey, you are going to work on a new project.

you : what is it about?

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

you : wow, IoT?

you :



[Breathing intensifies]

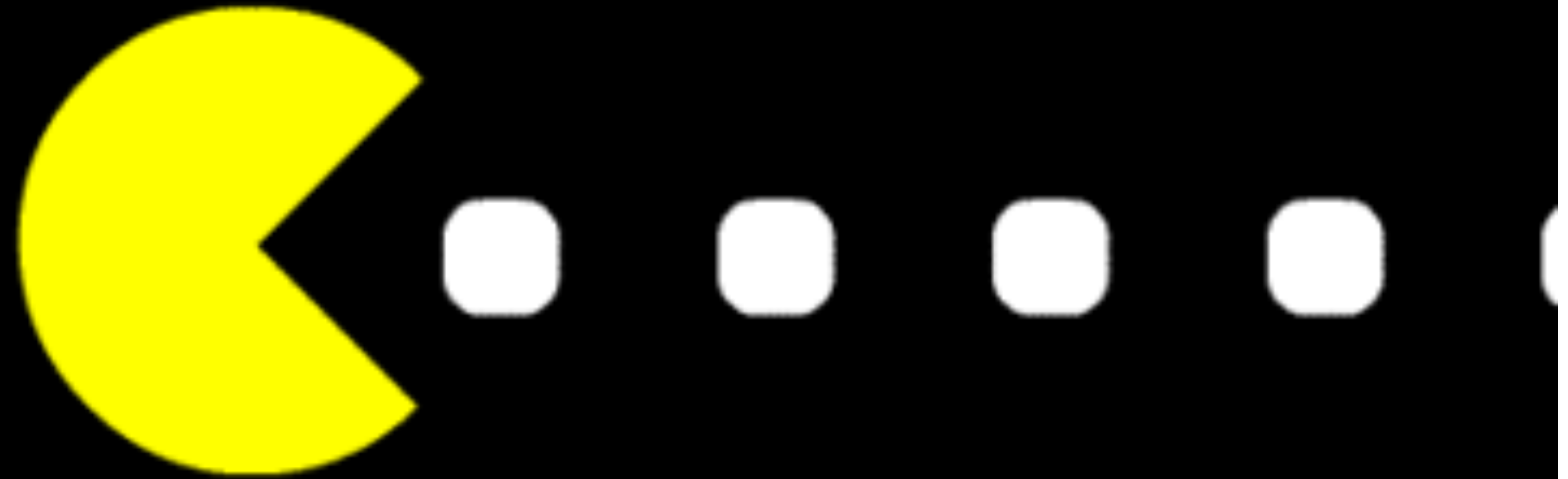
# On one fine day ...

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





# Crunching the IoT data



# 3 tier architecture

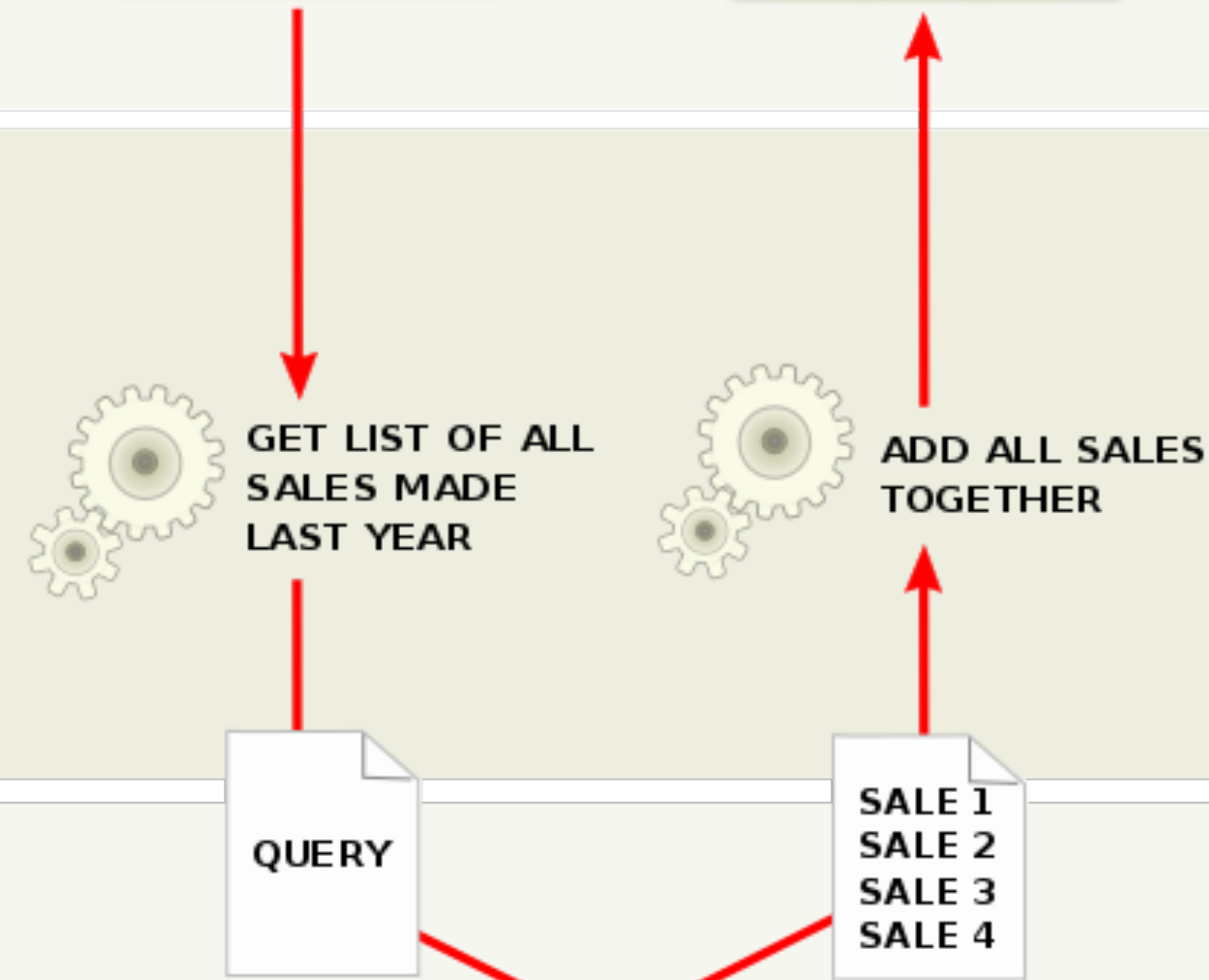
## Presentation tier

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.



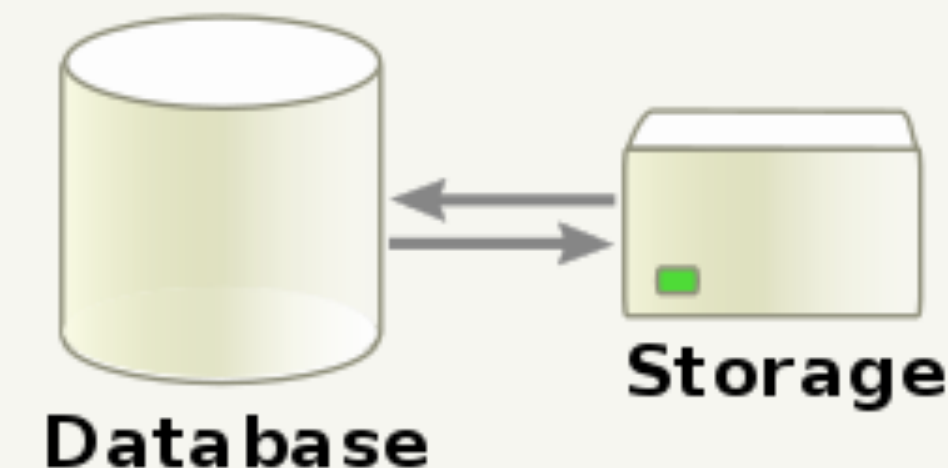
## Logic tier

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.

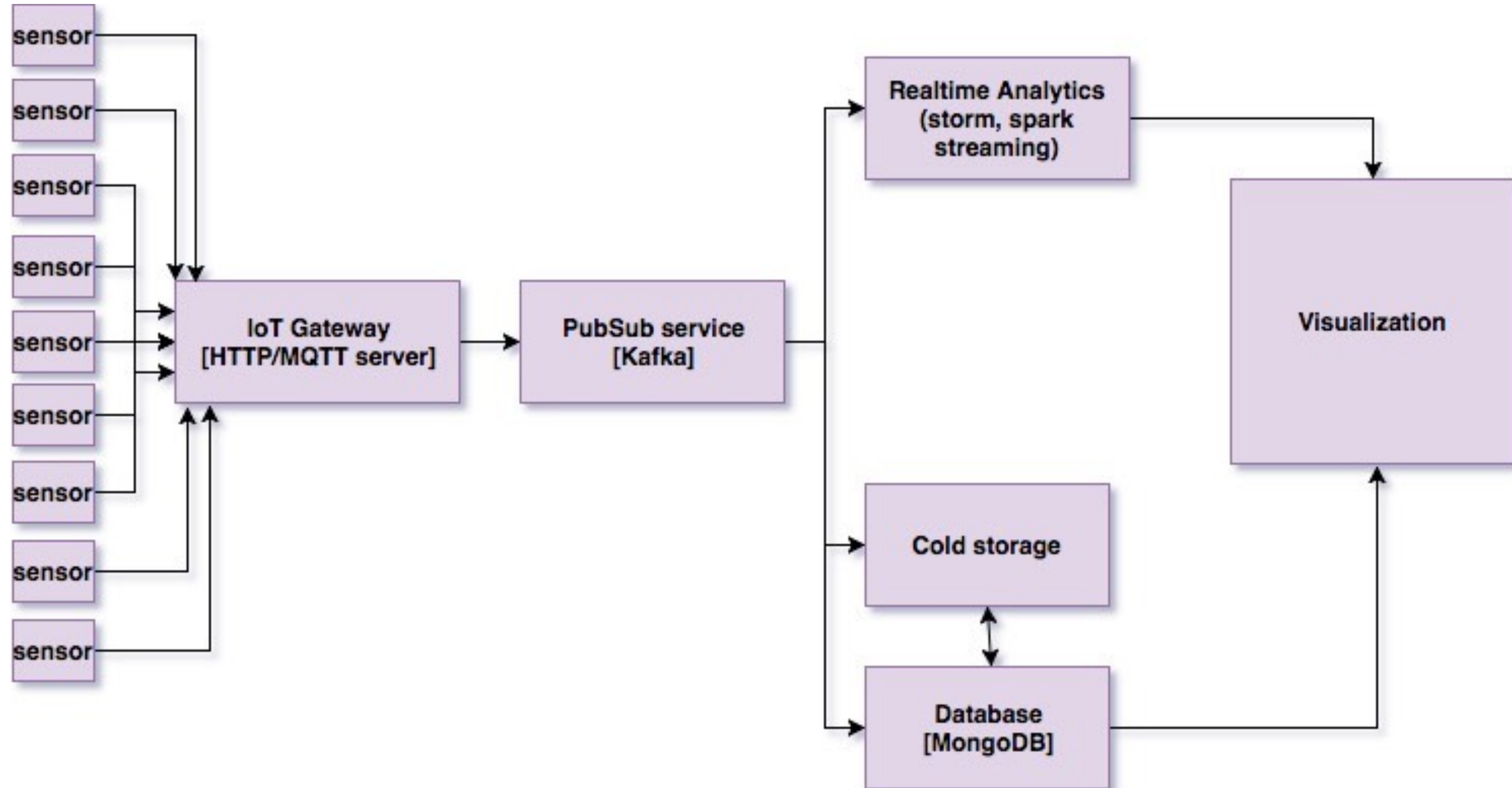


## Data tier

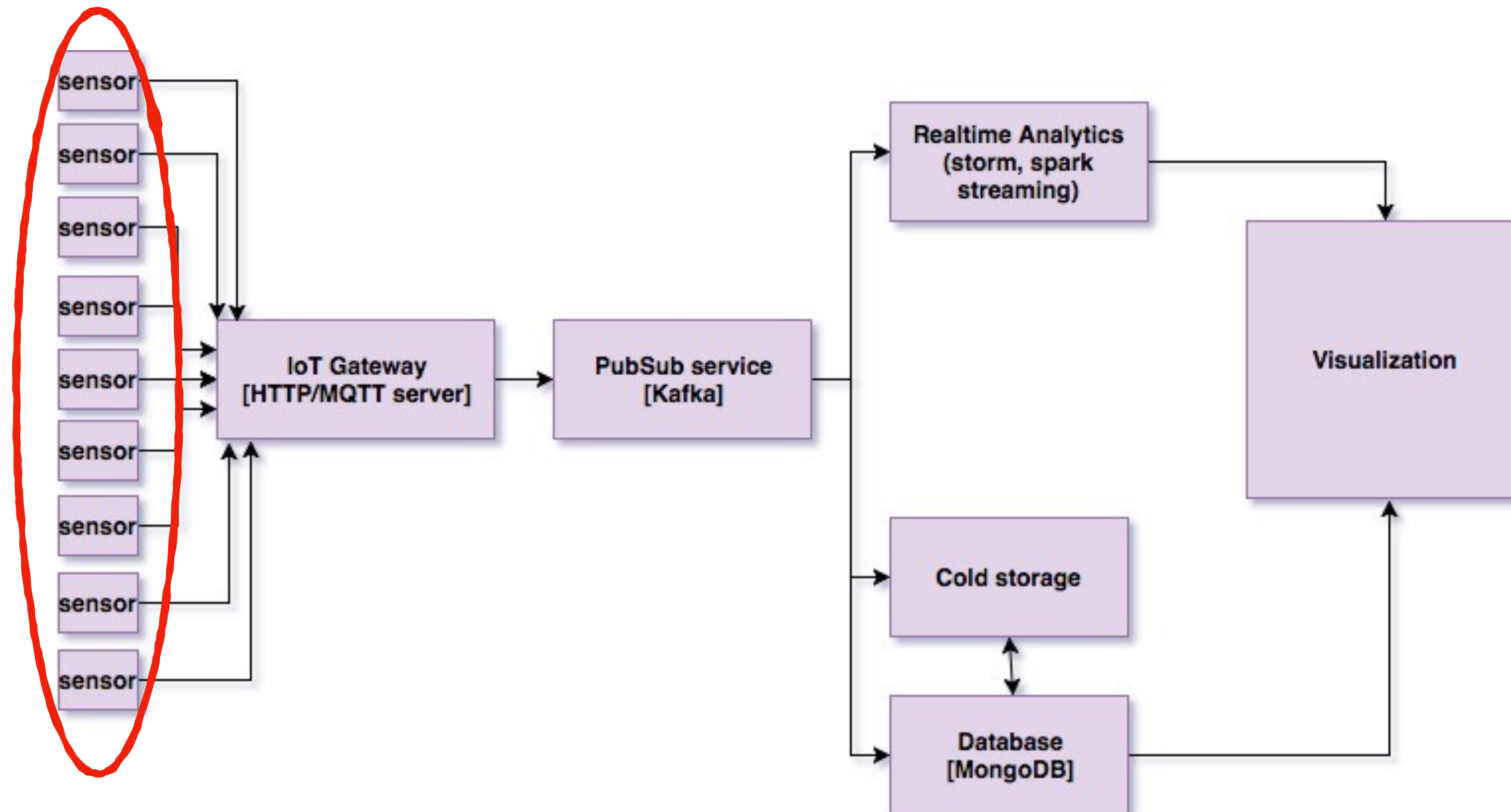
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.



# Traditional Implementation

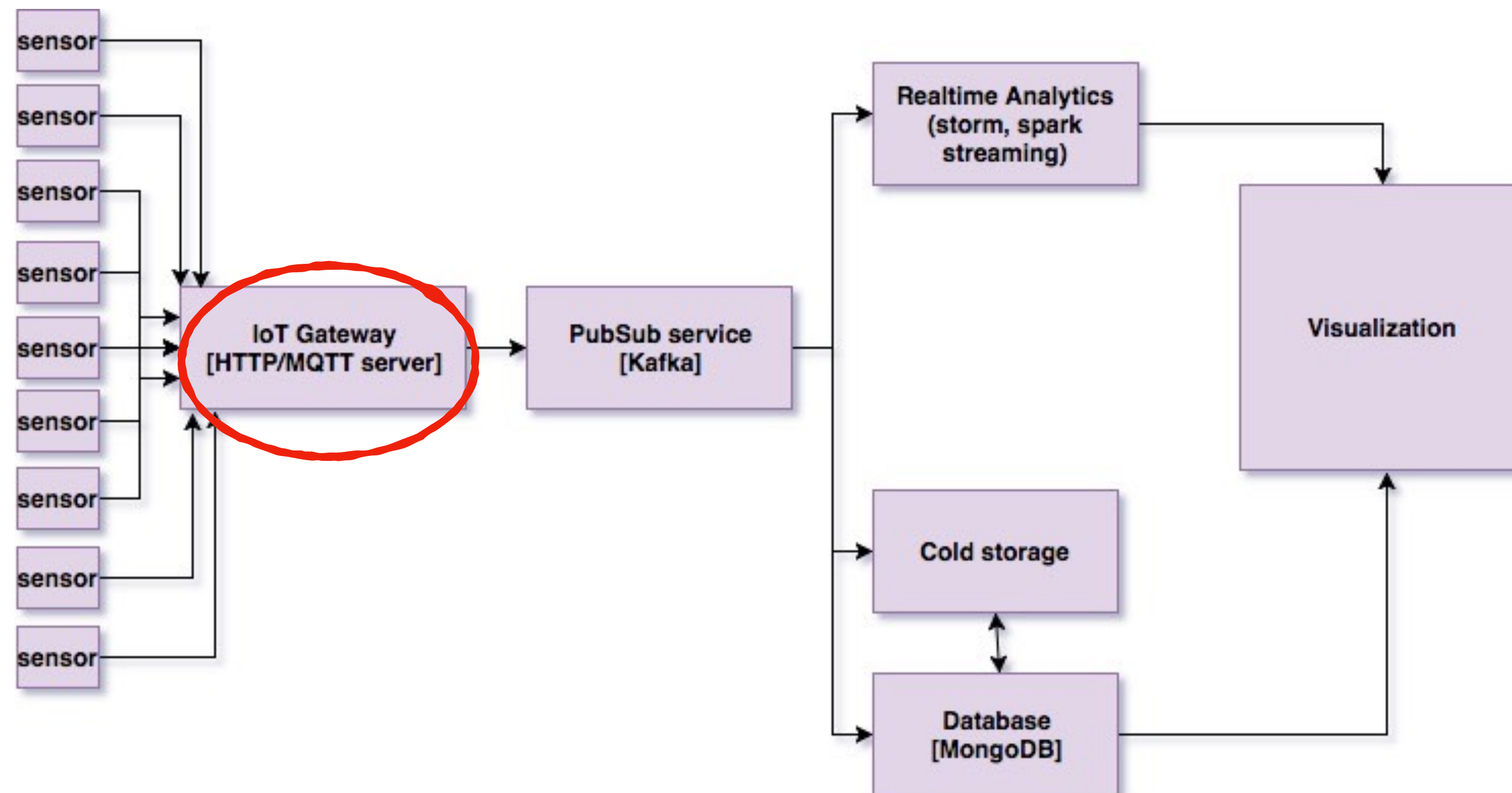


- Connected devices
- Heart rate sensors

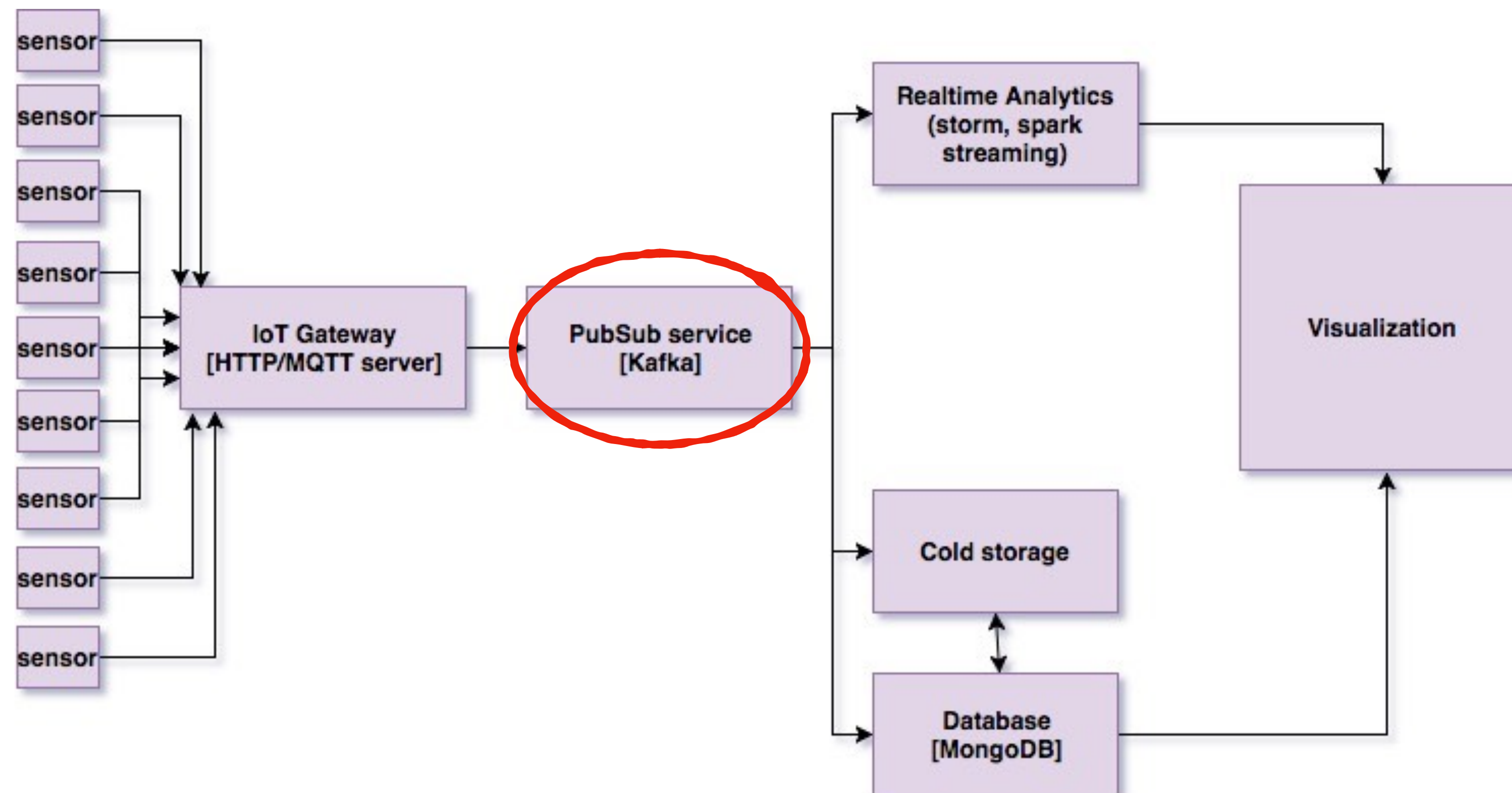




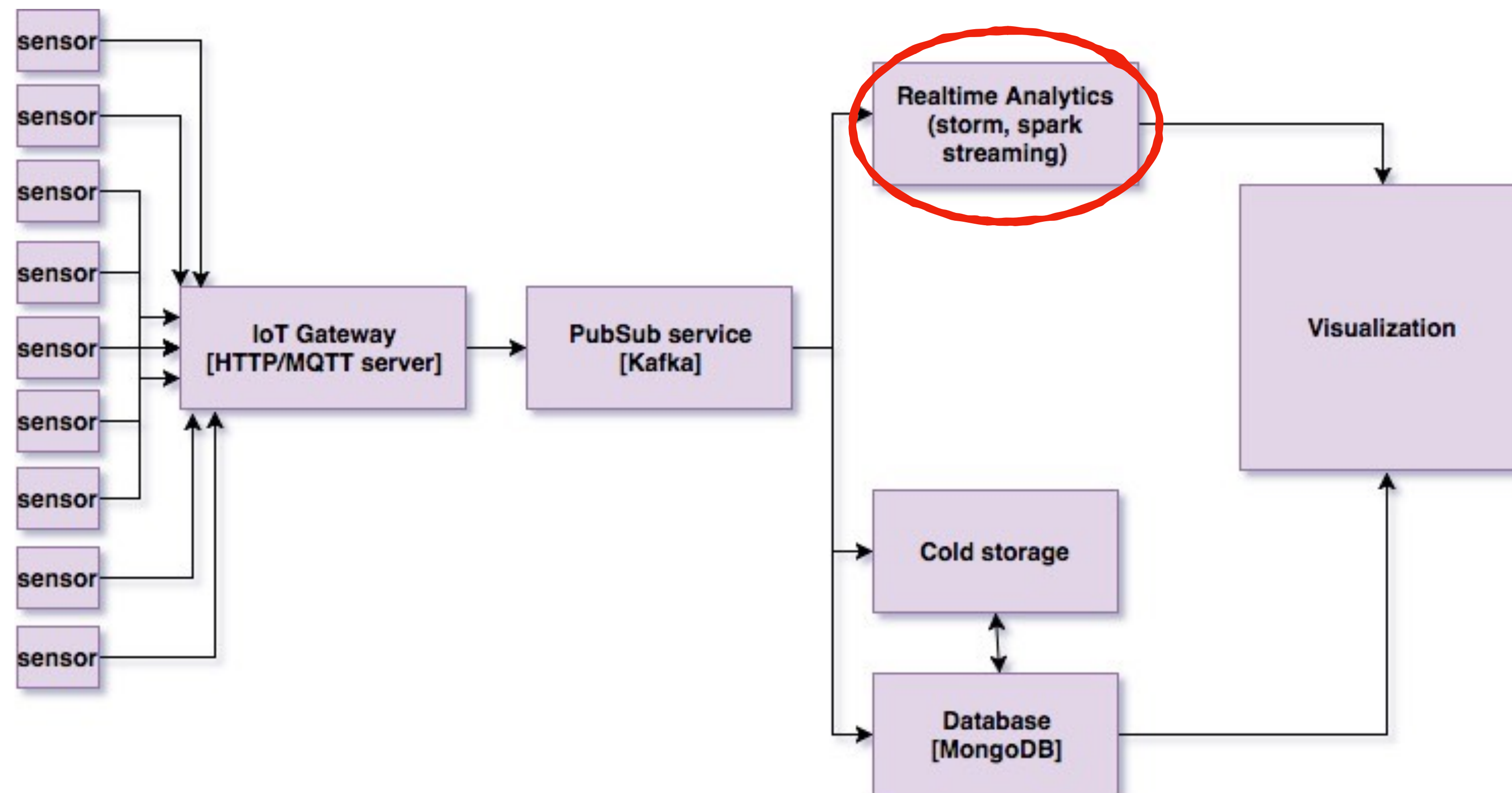
- Django server behind load balancer
- Authentication of registered IoT devices
- Scaling when the throughput or sensor count increases



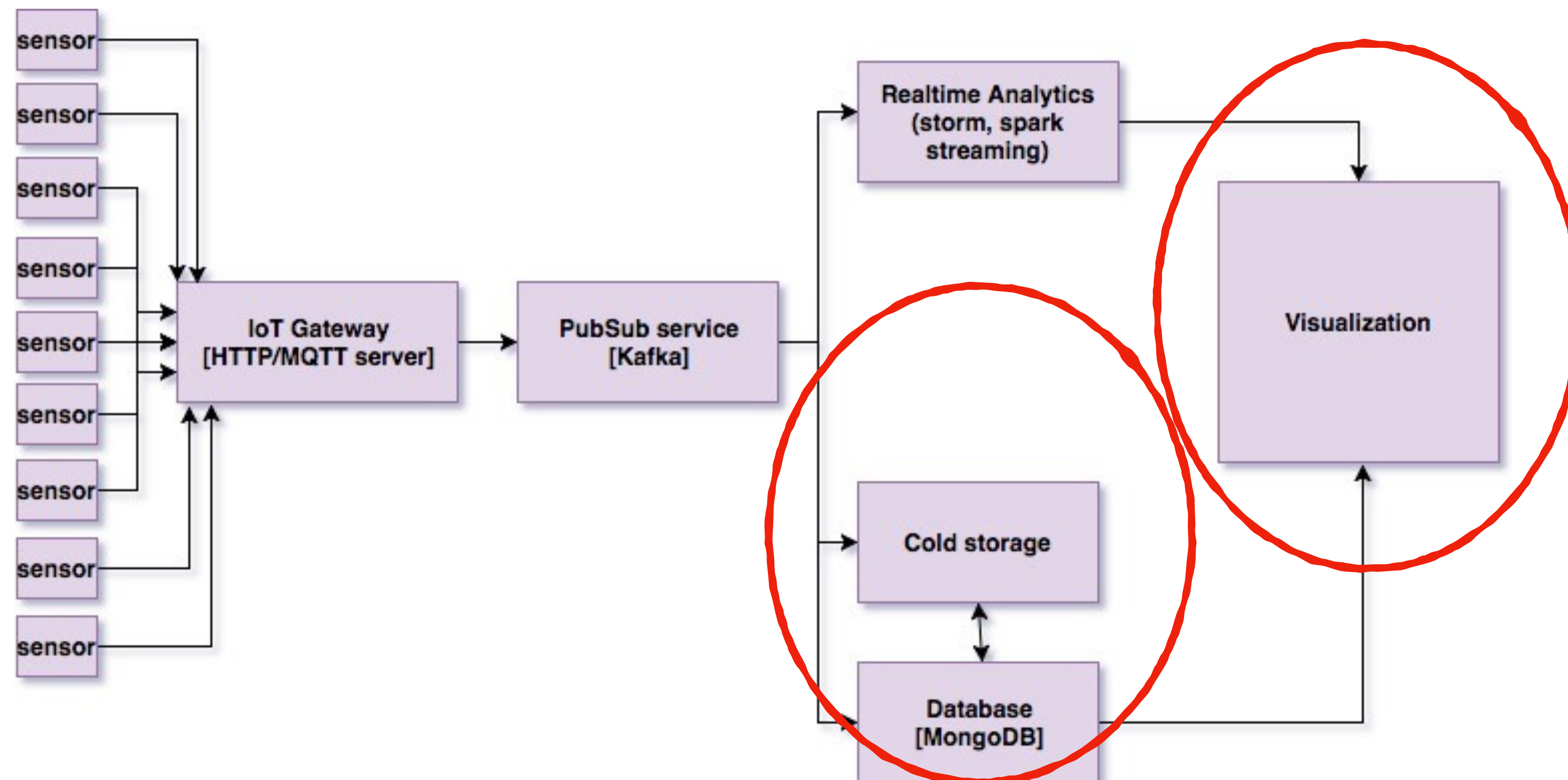
- Kafka
- Cluster/node maintainance
- Producers, Consumers, Nodes, Partitions, ZooKeepers, offsets



- 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

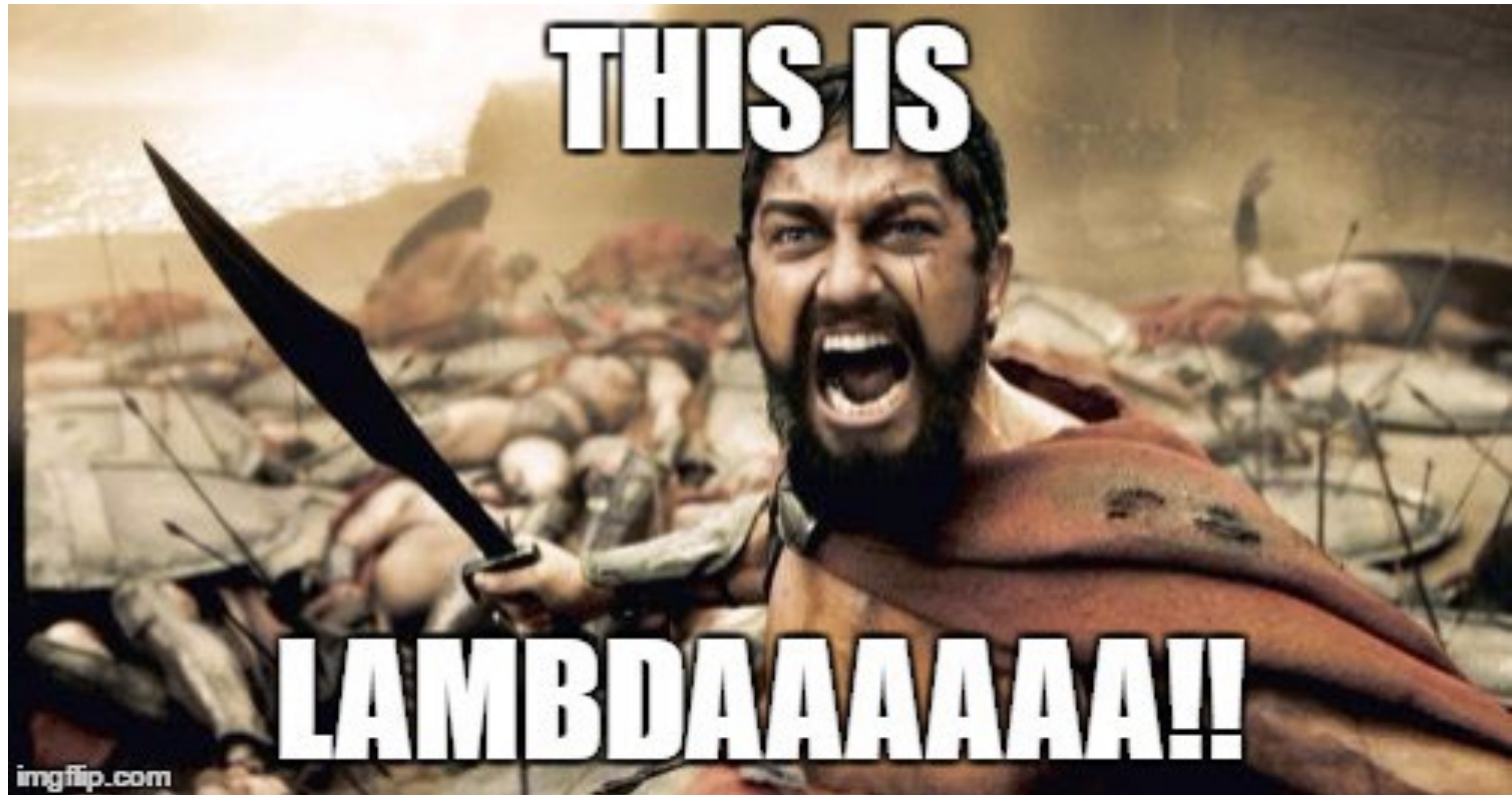
**Lets implement similar architecture  
but without servers**

# Amazon Web Services (AWS)



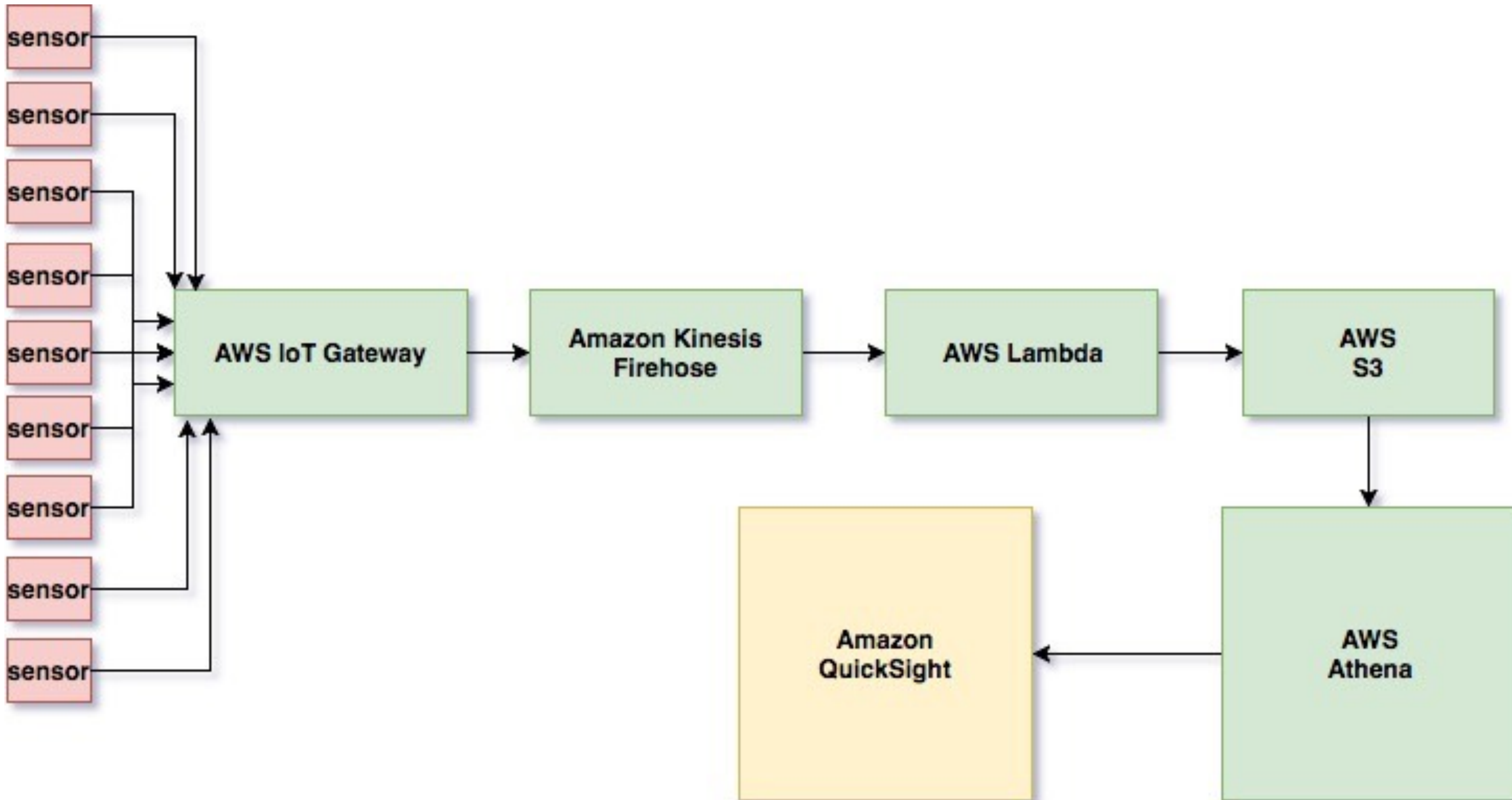
# AWS serverless

# AWS serverless

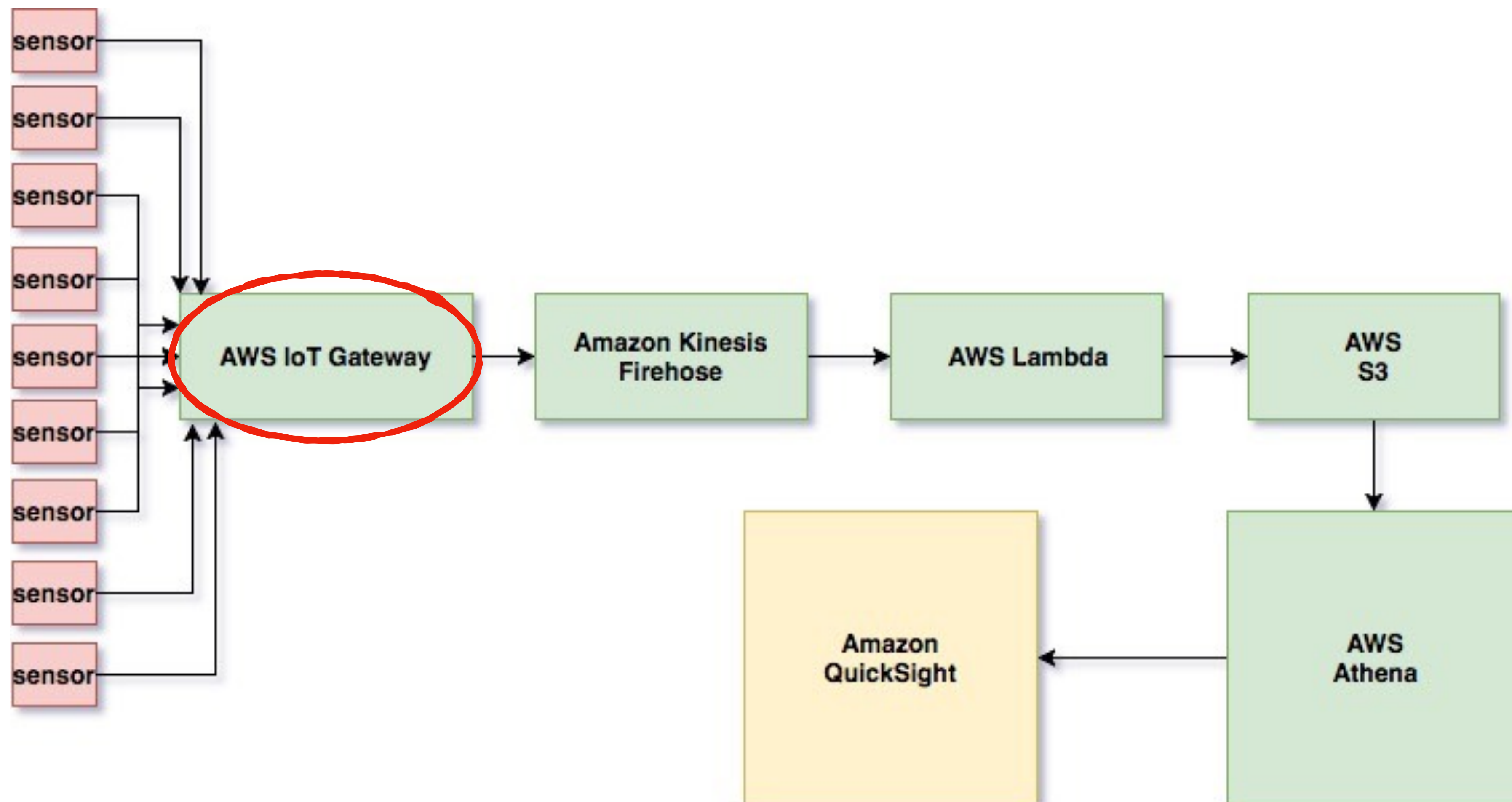




# FaaS

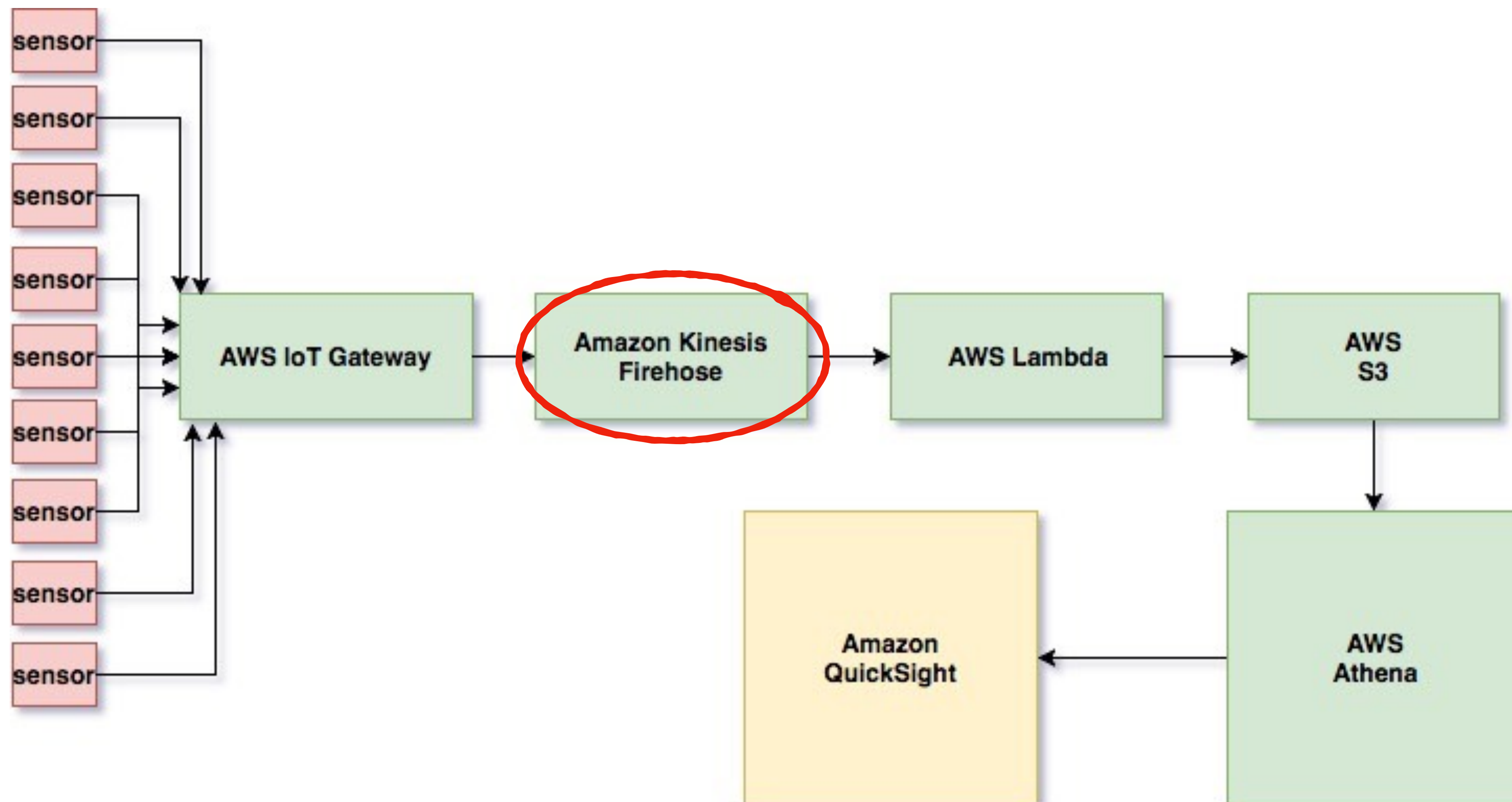


- Securely connect devices and interact with cloud applications
- Device gateway, Message broker, Rules engine

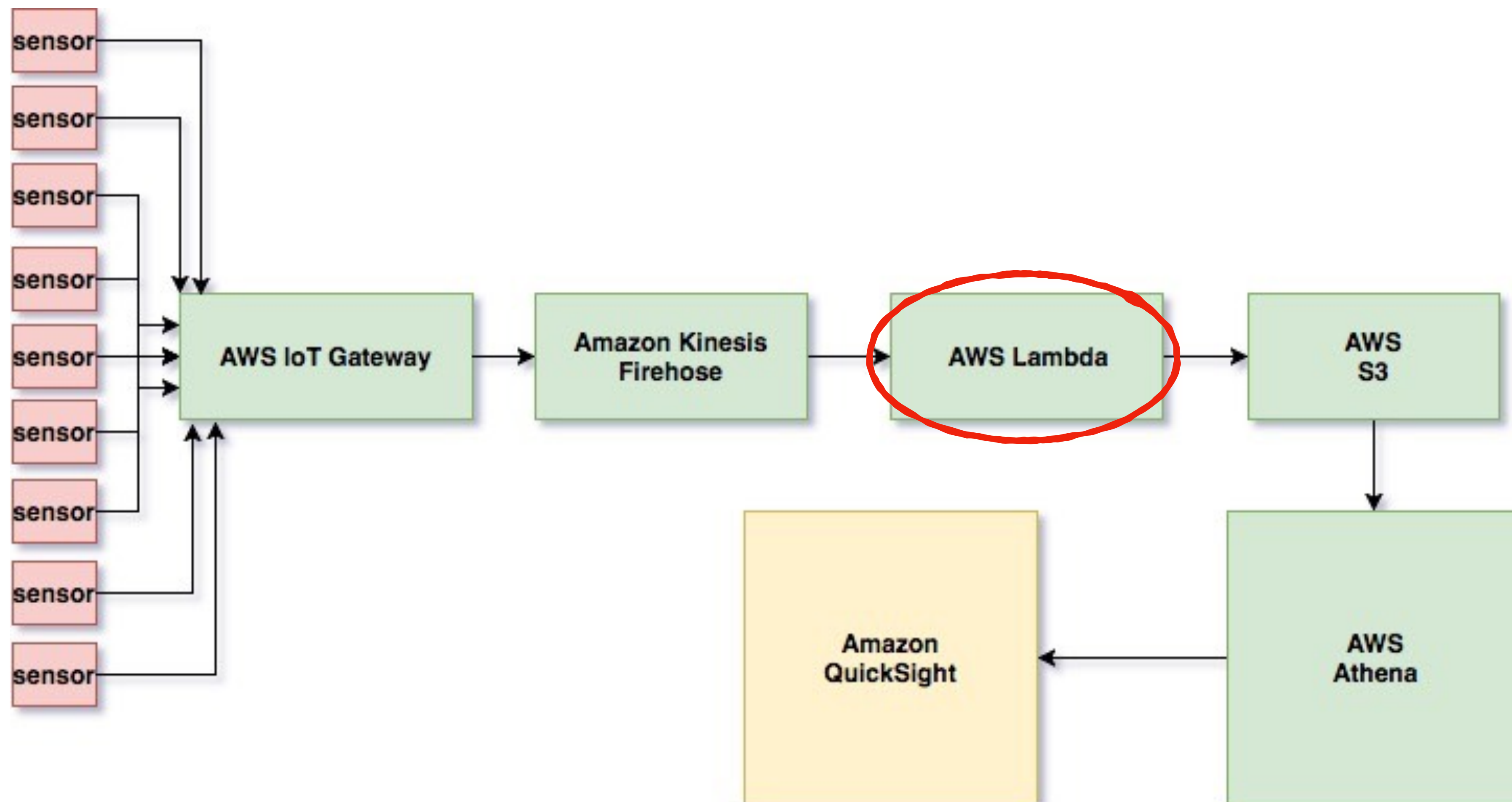




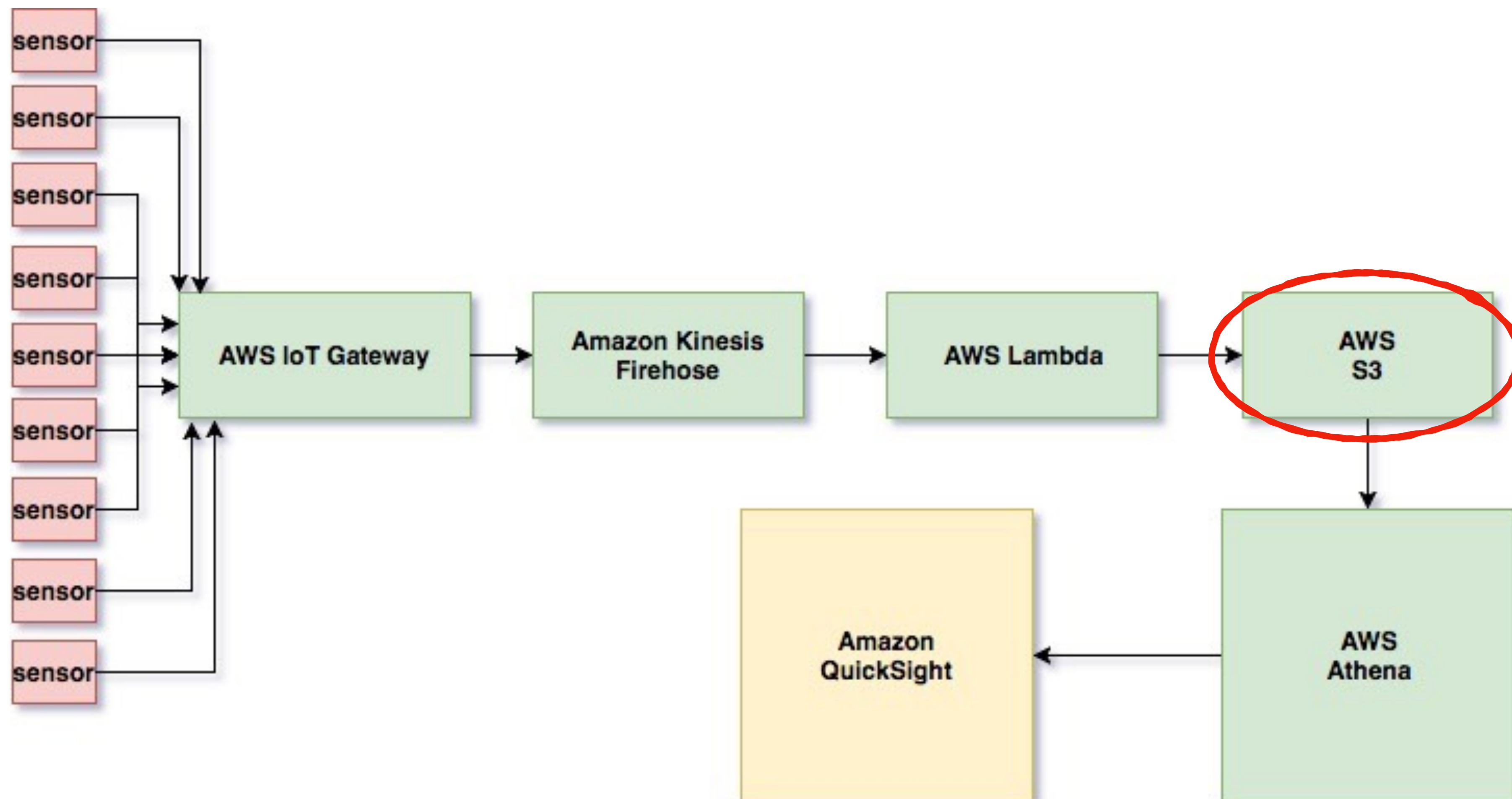
- Delivers real-time streaming data to other services such as Amazon S3, Elastic Search.
- Configurable producers and consumers



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

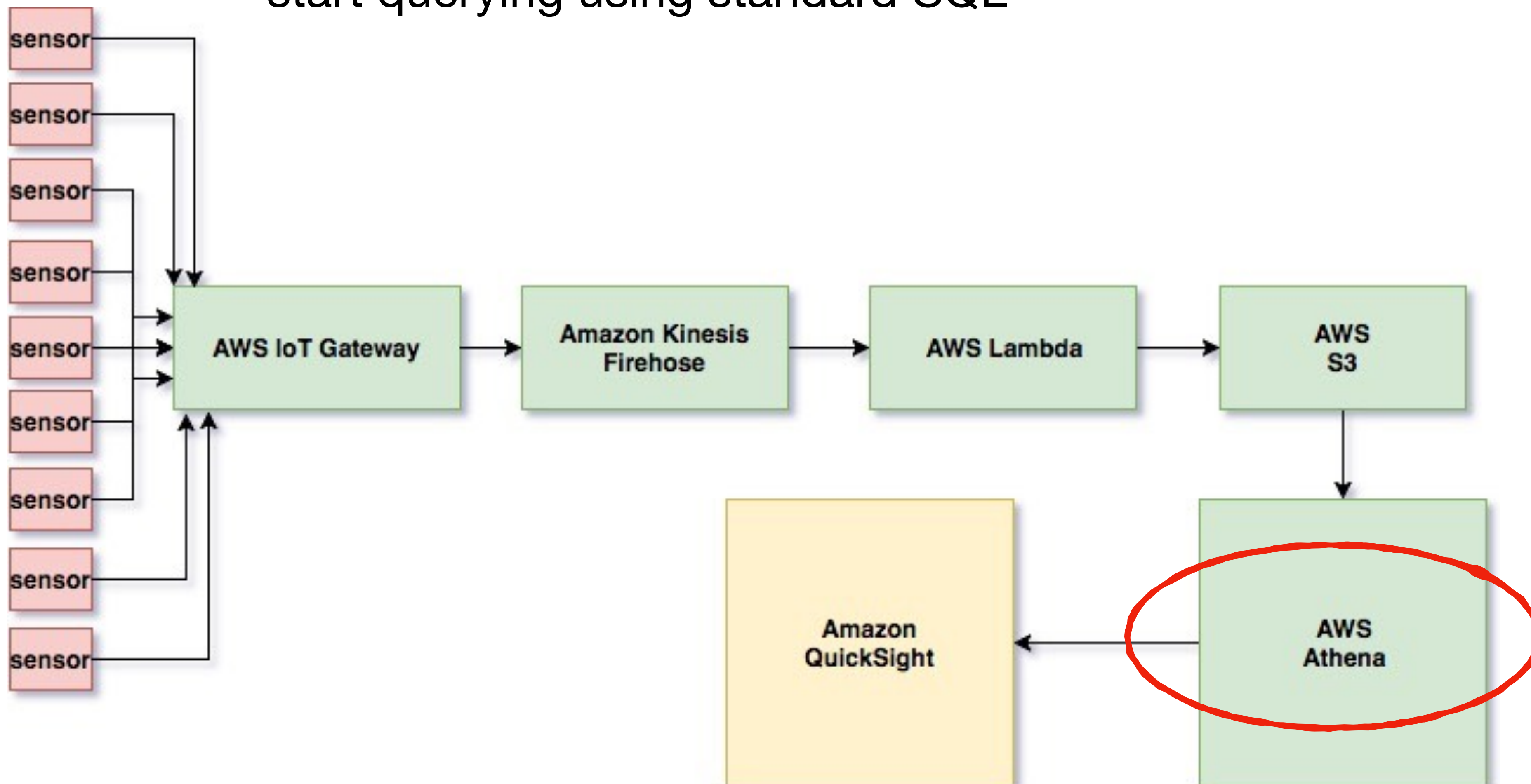


- Store data at massive scale
- Storage tier for many serverless applications



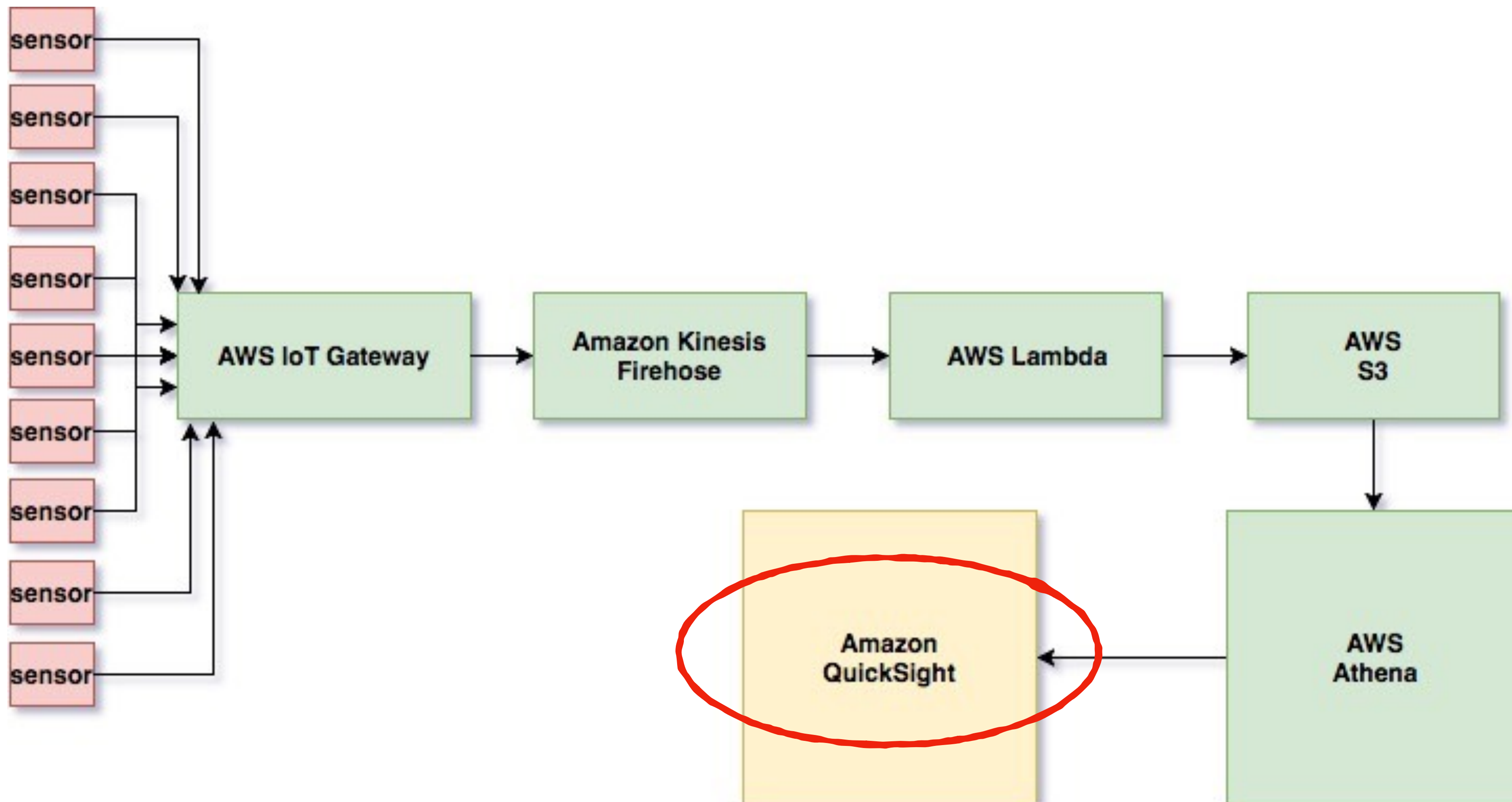


- Serverless interactive query service
- Point to your data source, define the schema, start querying using standard SQL





- 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

Amazon S3 > pydata.iot.pipeline / heartrate\_data / 2017 / 08 / 26 / 07

Overview

Q Type a prefix and press Enter to search. Press ESC to clear.

Upload

Create folder

More

US East (N. Virginia)

Viewing 1 to 13

<input type="checkbox"/>	Name	Last modified	Size	Storage class
<input type="checkbox"/>	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
<input type="checkbox"/>	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
<input type="checkbox"/>	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
<input type="checkbox"/>	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
<input type="checkbox"/>	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
<input type="checkbox"/>	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
<input type="checkbox"/>	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
<input type="checkbox"/>	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

ServicesResource Groups

US East (N. Virginia)

Amazon S3 > pydata.iot.pipeline / heartrate\_data / 2017 / 08 / 26 / 07

Overview

Search

UploadCreate folderMore

	Name	Last modified	Size	Storage class
<input type="checkbox"/>	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
<input type="checkbox"/>	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
<input type="checkbox"/>	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
<input type="checkbox"/>	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
<input type="checkbox"/>	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
<input type="checkbox"/>	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
<input type="checkbox"/>	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
<input type="checkbox"/>	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

# Kinesis Firehose

ServicesResource Groups

N. Virginia

Amazon Kinesis > Firehose delivery streams > pydata.iot.firehose

StreamsFirehoseAnalytics

Test with demo data

Use the tabs below to view, edit and monitor your delivery stream.

DetailsMonitoringS3 Logs

Delete Delivery Stream

Delivery stream name\*

Source

S3 bucket

S3 prefix

IAM role\*

Data transformation\*

Source record backup\*

S3 buffer size (MB)\*

S3 buffer interval (sec)\*

S3 Compression

S3 Encryption

pydata.iot.firehose

Direct PUT

pydata.iot.pipeline

heartrate\_data/

firehose\_delivery\_role

Disabled

Disabled

1

60

UNCOMPRESSED

No Encryption

Edit

# Athena

ServicesResource Groups

N. Virginia

Athena Query Editor

Database: default

Tables: heartrate\_iot\_data

1-- Run an ANSI SQL or Hive Data Definition Language (DDL) statement

2

3-- ANSI SQL Example:

4

5-- SELECT \* FROM default.cloudfront\_logs limit 10;

6

7-- Hive DDL Example:

8

9-- CREATE EXTERNAL TABLE IF NOT EXISTS cloudfront\_logs (

10-- Date Date,

11-- Time STRING,

12-- Location STRING,

13-- Bytes INT,

14-- RequestIP STRING,

15-- Method STRING,

16-- Host STRING,

17-- Uri STRING,

18-- Status INT,

19-- Referrer STRING,

20-- OS String,

21-- ...

Run Query

Save As

Format Query

New Query

Results

Services

Resource Groups

Amazon S3

>

pydata.iot.pipeline

/

heartrate\_data

/

2017

/

08

/

26

/

07

Overview

Q

Type a prefix and press Enter to search. Press ESC to clear.

Upload

Create folder

More

US East (N. Virginia)

Viewing 1 to 13

<input type="checkbox"/>	Name	Last modified	Size	Storage class
<input type="checkbox"/>	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
<input type="checkbox"/>	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
<input type="checkbox"/>	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
<input type="checkbox"/>	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
<input type="checkbox"/>	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
<input type="checkbox"/>	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
<input type="checkbox"/>	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
<input type="checkbox"/>	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

Amazon Kinesis

Streams

Firehose

Analytics

Firehose delivery streams

pydata.iot.firehose

Test with demo data

Use the tabs below to view, edit and monitor your delivery stream.

Details

Monitoring

S3 Logs

Delete Delivery Stream

Delivery stream name\*

pydata.iot.firehose

Source

Direct PUT

S3 bucket

[pydata.iot.pipeline](#)

S3 prefix

heartrate\_data/

IAM role\*

firehose\_delivery\_role

Data transformation\*

Disabled

Source record backup\*

Disabled

S3 buffer size (MB)\*

1

S3 buffer interval (sec)\*

60

S3 Compression

UNCOMPRESSED

S3 Encryption

No Encryption

Edit

Services

Resource Groups

Athena

Query Editor

Saved Queries

History

Catalog Manager

Settings

Tutorial

Help

What's new

DATABASE

default

TABLES

Filter Tables...

heartrate\_iot\_data

To use the AWS Glue Data Catalog with Amazon Athena and Amazon Redshift Spectrum, you must upgrade your Athena Data Catalog to the AWS Glue Data Catalog. Without the upgrade, tables and partitions created by AWS Glue cannot be queried with Amazon Athena or Redshift Spectrum. Click [here](#) to upgrade.

-- Run an ANSI SQL or Hive Data Definition Language (DDL) statement

-- ANSI SQL Example:

-- SELECT \* FROM default.cloudfront\_logs limit 10;

-- Hive DDL Example:

-- CREATE EXTERNAL TABLE IF NOT EXISTS cloudfront\_logs (  
-- Date Date,  
-- Time STRING,  
-- Location STRING,  
-- Bytes INT,  
-- RequestIP STRING,  
-- Method STRING,  
-- Host STRING,  
-- Uri STRING,  
-- Status INT,  
-- Referrer STRING,  
-- OS String,  
-- ...

Use Ctrl + Enter to run query, Ctrl + Space to autocomplete

Run Query

Save As

Format Query

New Query

Results

Visualize

Fields list

SPICE heartrate\_jot\_data 100%

datetime

ratetype

userid

heartrate

Visual types

Field wells

Sum of Heartrate by Userid and Datetime

Userid	Sum of Heartrate (2017)
Bailey	~1.8K
Ben	~1.2K
Bella	~2.3K
Beau	~2.6K
Beatrice	~2.7K
Beth	~2.2K

# 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

```
1  import boto3
2
3  client = boto3.client('firehose')
4  response = client.create_delivery_stream(
5      DeliveryStreamName='iot_boto3',
6      S3DestinationConfiguration={
7          'RoleARN': 'arn:aws:iam::1234567890:role/firehose_delivery_role',
8          'BucketARN': 'arn:aws:s3:::pydata.iot.pipeline',
9          'Prefix': 'string',
10         'BufferingHints': {
11             'SizeInMBs': 5,
12             'IntervalInSeconds': 300
13         },
14         'CompressionFormat': 'UNCOMPRESSED',
15         'EncryptionConfiguration': {
16             'NoEncryptionConfig': 'NoEncryption',
17         },
18     },
19 )
20 print(response)
21
```



## athena query : create table

```
1 import boto3
2
3 client = boto3.client('athena')
4 response = client.create_named_query(
5     Name='iot_boto3',
6     Description='create table from heart rate data',
7     Database='default',
8     QueryString="""CREATE EXTERNAL TABLE heartrate_iot_data (
9         heartRate int,
10        userId string,
11        rateType string,
12        dateTime timestamp)
13        ROW FORMAT serde 'org.apache.hive.hcatalog.data.JsonSerDe'
14        with serdeproperties( 'ignore.malformed.json' = 'true' )
15        LOCATION 's3://pydata.iot.pipeline/heartrate_data/""",
16 )💡
17 print(response)
18
```

# 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](http://dudewho.codes/talks)

Thank you  
@DudeWhoCode