Brewing beer with Python

Chesco Igual
@chescales

ELEMENTS

What shalt thou learn?

- Building an IoT backend -

- Technologies, Protocols and Tools -

- Backend considerations -

- Meet a full running architecture -

- Learn to brew beer -

What shalt thou learn?

- Building an IoT backend -

- Technologies, Protocols and Tools -

- Backend considerations -

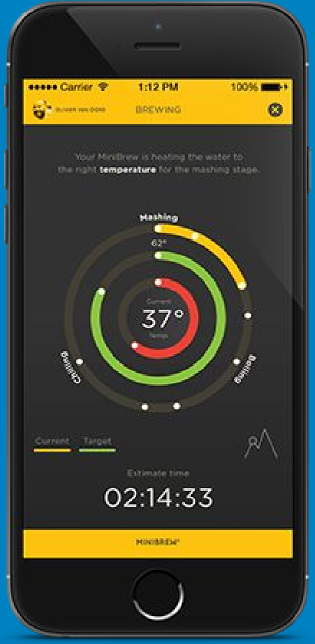
- Meet a full running architecture -

~~- Learn to brew beer -~~

MiniBrew

So what is it?

MINIBREW
BRAVE BEERS



••••• Carrier 1:12 PM 100%

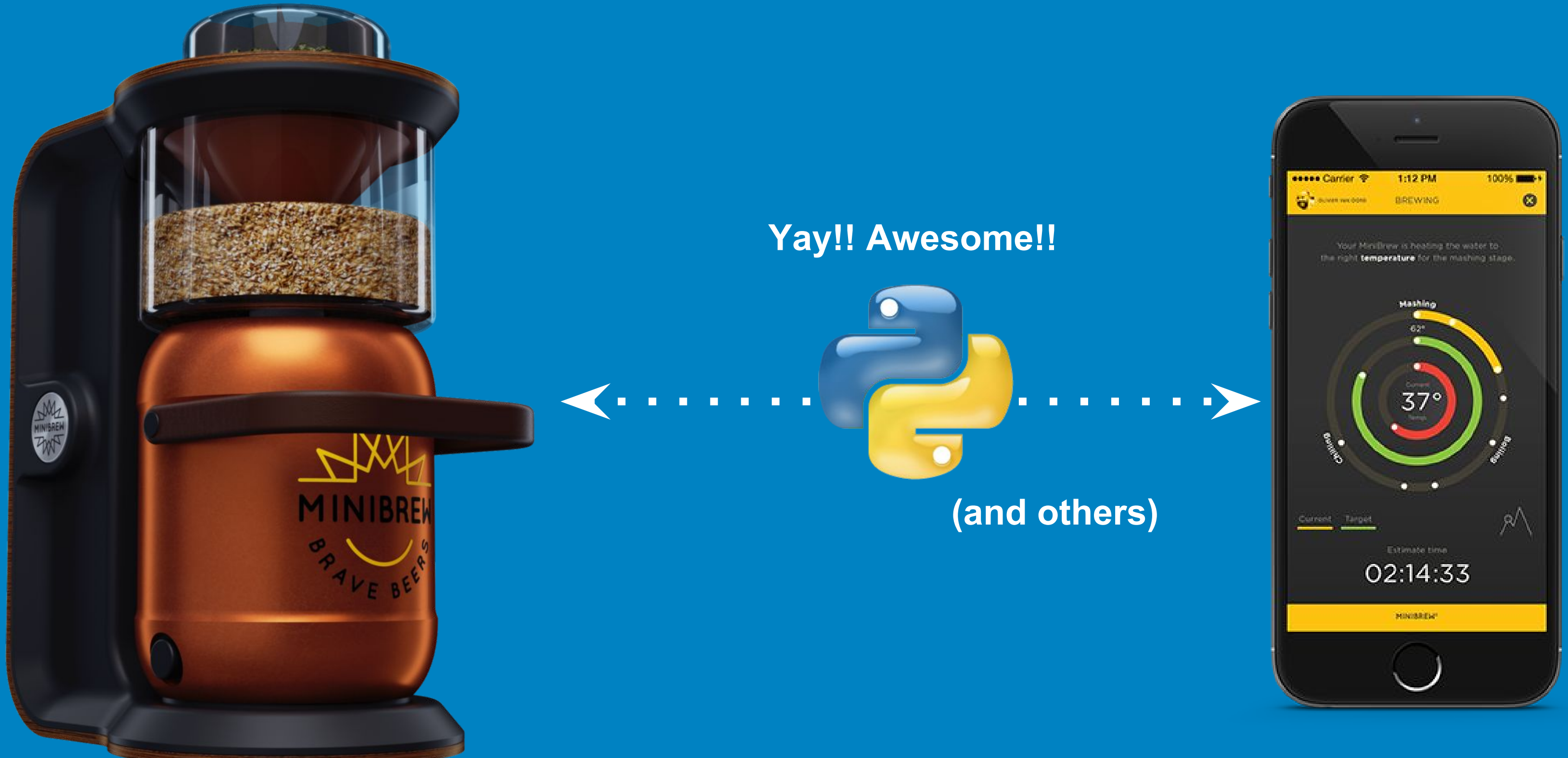
BREWING

Your MiniBrew is heating the water to the right **temperature** for the mashing stage.

Estimate time

02:14:33

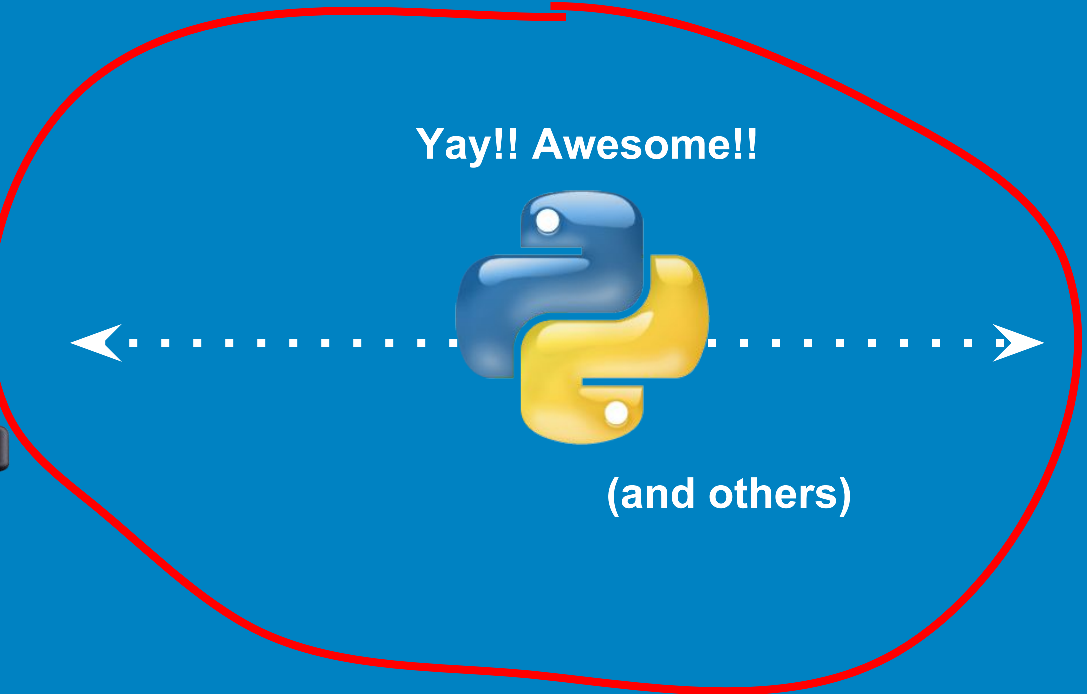
So what is it?



Yay!! Awesome!!

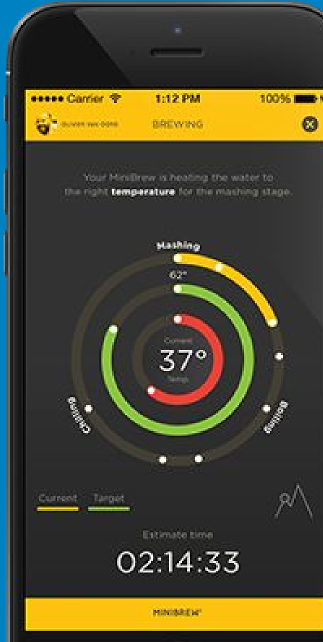
(and others)

So what is it?



Yay!! Awesome!!

(and others)



Your MiniBrew is heating the water to the right **temperature** for the mashing stage.

Let’s go technical

Project Requirements

Real-time data

Project Requirements

- Real-time data -

Security

Project Requirements

- Real-time data -

- Security -

Obfuscation

Project Requirements

- Real-time data -

- Security -

- Obfuscation -

Authentication

Project Requirements

- Real-time data -

- Security -

- Obfuscation -

- Authentication -

Two-way communication

Project Requirements

- Real-time data -

- Security -

- Obfuscation -

- Authentication -

- 2-way communication -

Resiliency

Project Requirements

- Real-time data -

- Security -

- Obfuscation -

- Authentication -

- 2-way communication -

- Resiliency -

Lightweight

Project Requirements

- Real-time data -
- Security -
- Obfuscation -
- Authentication -
- 2-way communication -
- Resiliency -
- Lightweight -

- Last known status -
- Debugging -
- Admin site -
- Mobile app api -
- rainbows, etc. -

What thou shalt take care about too...

- Scalability -

- Proven technologies -

- small tech stack -

- Error tracking -

- Reduce data transfer -

- Documentation -

What thou shalt take care about too...

- Scalability -

- Proven technologies -

- small tech stack -

- Error tracking -

- Reduce data transfer -

~~- Documentation -~~

when I read the project specs for the first time

Source: thecodinglove.com

Step by Step

Communications Protocol

Communication Protocols

~~AMQP~~

~~HTTP~~

MQTT

COAP

~~XMPP~~

~~DDS~~

Communication Protocols

~~AMQP~~

~~HTTP~~

MQTT

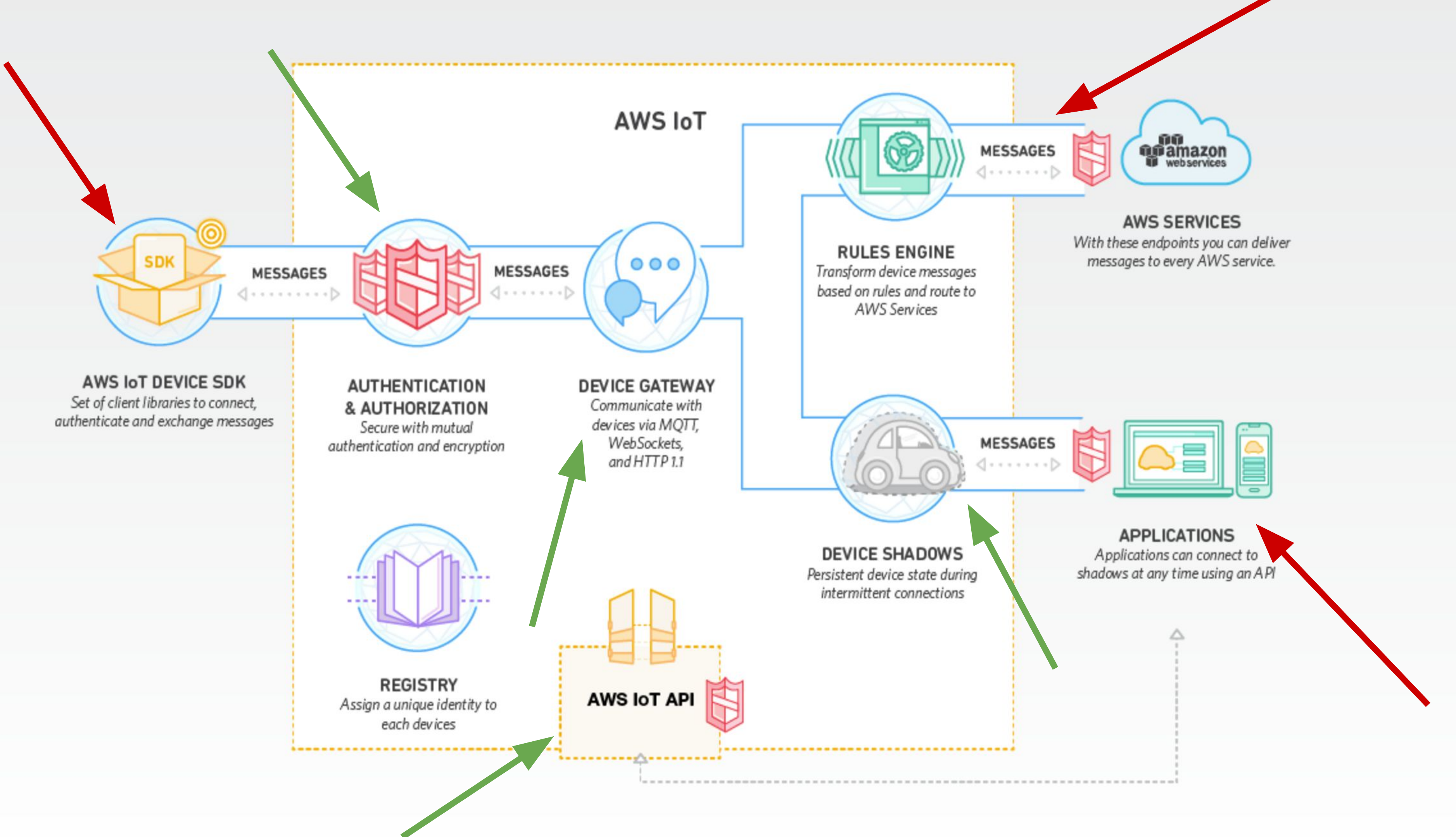
COAP

~~XMPP~~

~~DDS~~

Comprehensive IoT Backend Solutions

Amazon IoT



AWS IoT

AWS IoT DEVICE SDK
Set of client libraries to connect, authenticate and exchange messages

MESSAGES

AUTHENTICATION & AUTHORIZATION
Secure with mutual authentication and encryption

MESSAGES

DEVICE GATEWAY
Communicate with devices via MQTT, WebSockets, and HTTP 1.1

RULES ENGINE
Transform device messages based on rules and route to AWS Services

MESSAGES

AWS SERVICES
With these endpoints you can deliver messages to every AWS service.

DEVICE SHADOWS
Persistent device state during intermittent connections

MESSAGES

APPLICATIONS
Applications can connect to shadows at any time using an API

REGISTRY
Assign a unique identity to each devices

AWS IoT API

So then...?

<u>PROJECT REQUIREMENTS</u>

- Real-time data - ✗
- Security ✓
- Obfuscation - ✗
- Authentication - ✓
- 2-way communication - ✓
- Resiliency - ✓✓

- Lightweight - ✗
- Last known status - ✓
- Debugging -
- Admin site -
- Mobile app api - ✓
- rainbows, etc. - ✗

<u>EXTRAS</u>

- Scalability - ✓✓✓
- Proven technologies - ✓
- small tech stack - ✗
- Error tracking -
- Reduce data transfer - ✗
- Documentation - ✓

Let’s set up our own

(get a broker…)

Options?

ActiveMQ™

eMQ

EMQTT

CloudMQTT

HiveMQ

ENTERPRISE MQTT BROKER

Mosquitto

verneMQ

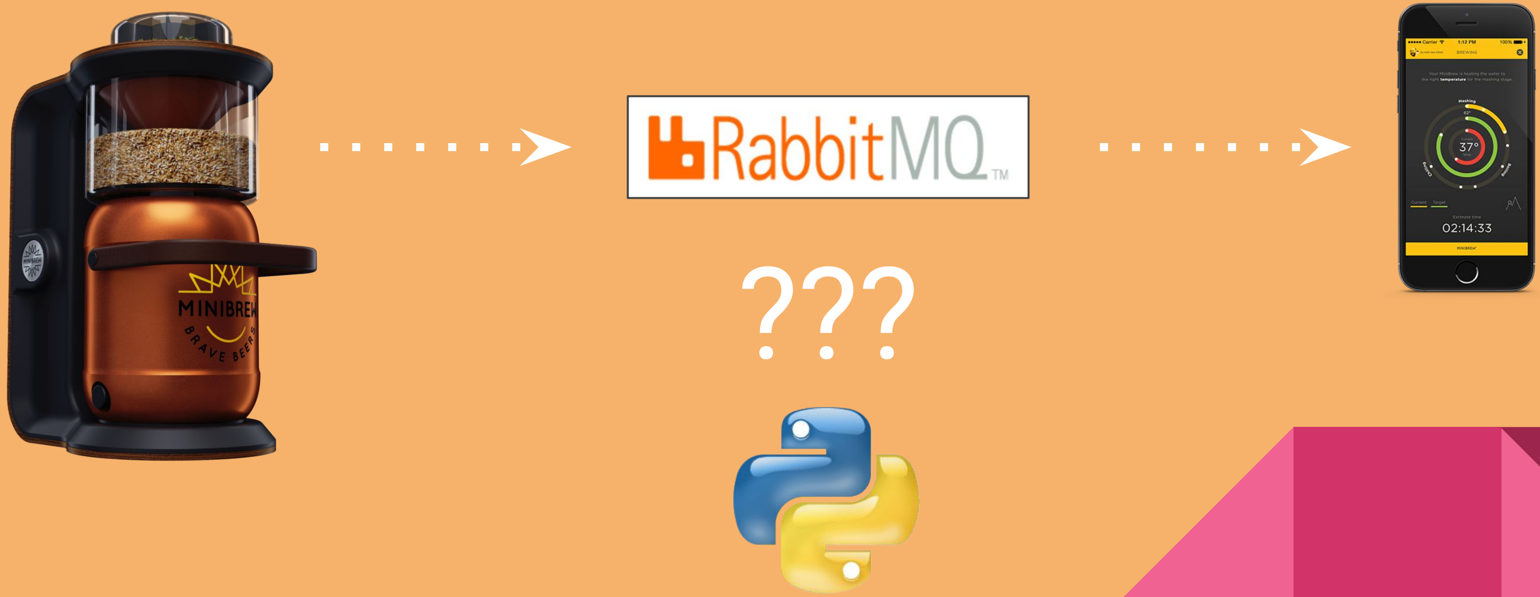
RabbitMQ™

Options?

- ❤️ Top player for many years
- ❤️ Scalability proven (vertical and horizontal)
- ❤️ Can convert from MQTT to other protocols (AMQP)
- ❤️ No payment per use
- ❤️ Familiarity

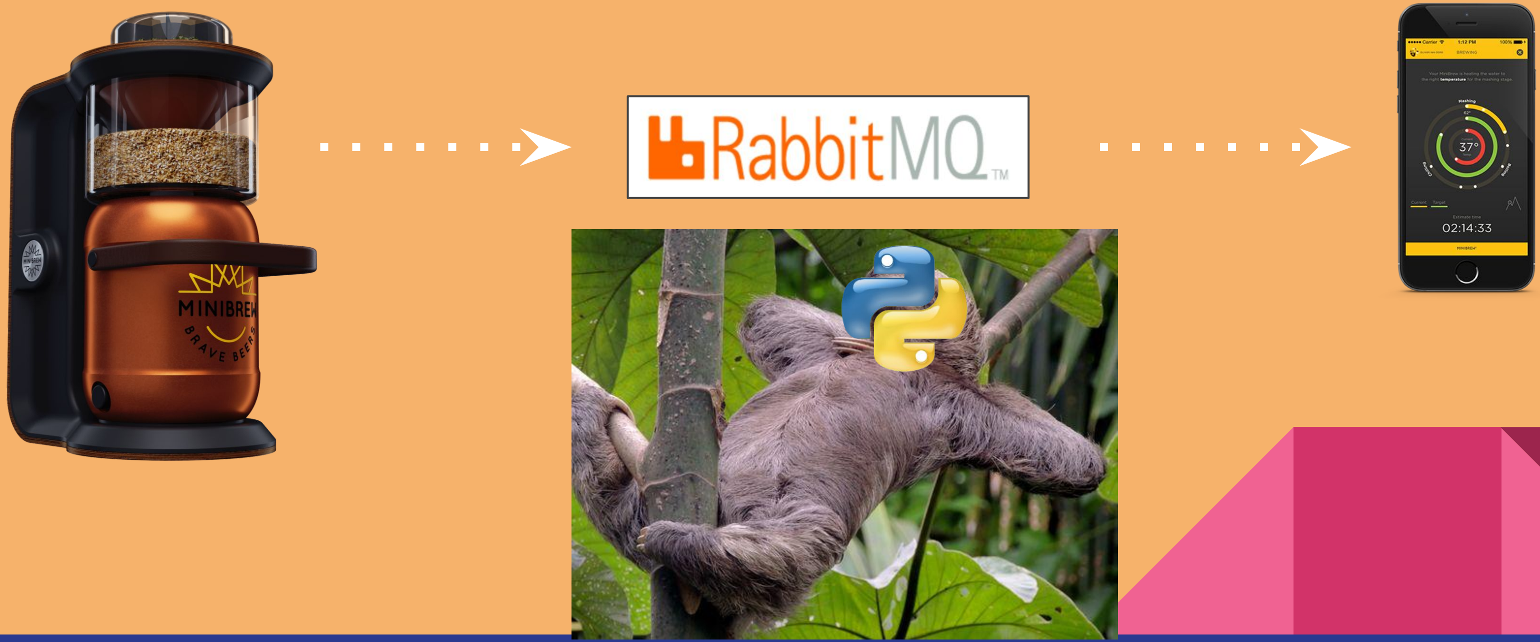
RabbitMQ™

Extra bonus!



???

Extra bonus!



Now what?

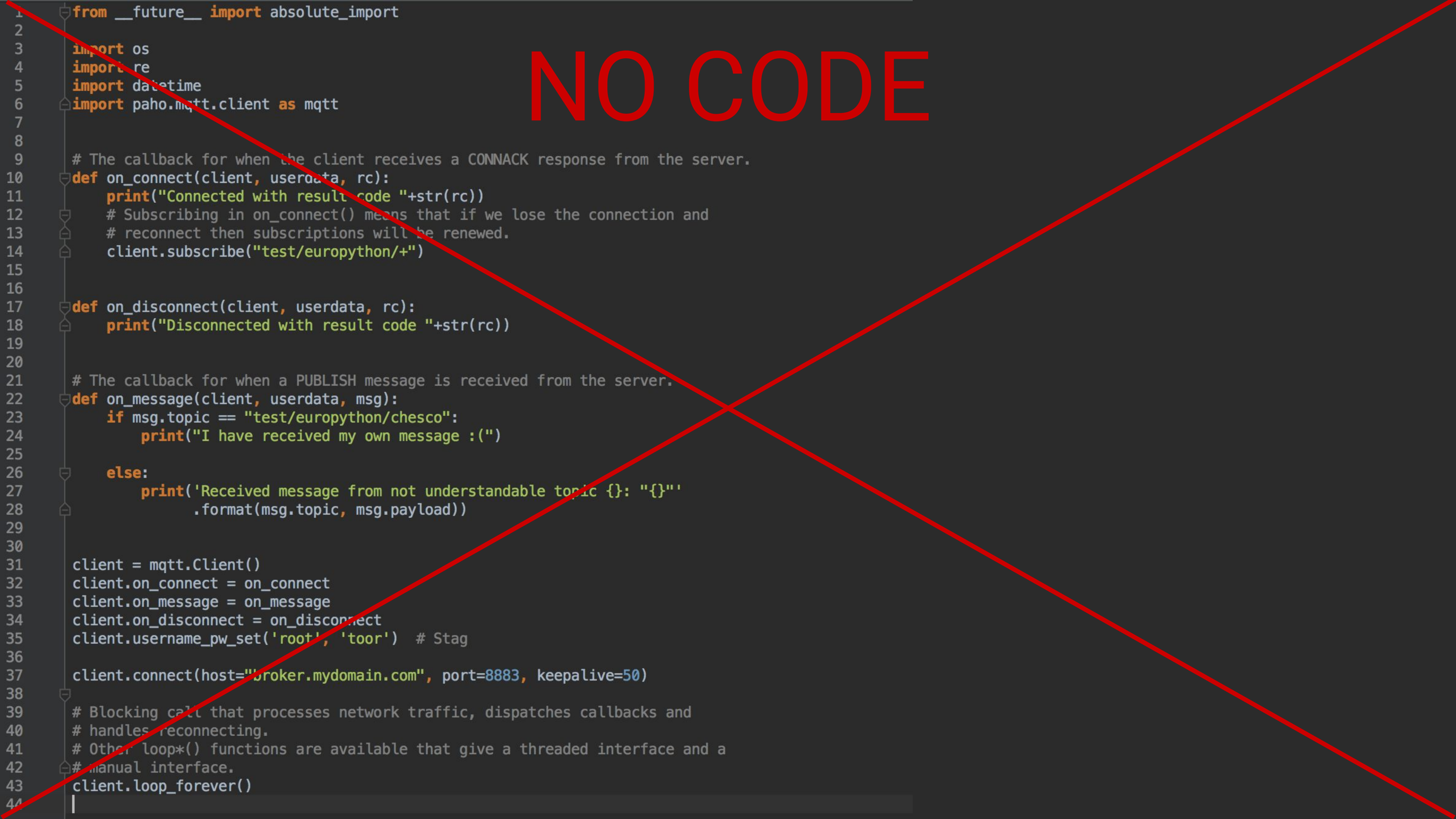
Let’s talk to that broker

❤️ 🐍 ❤️

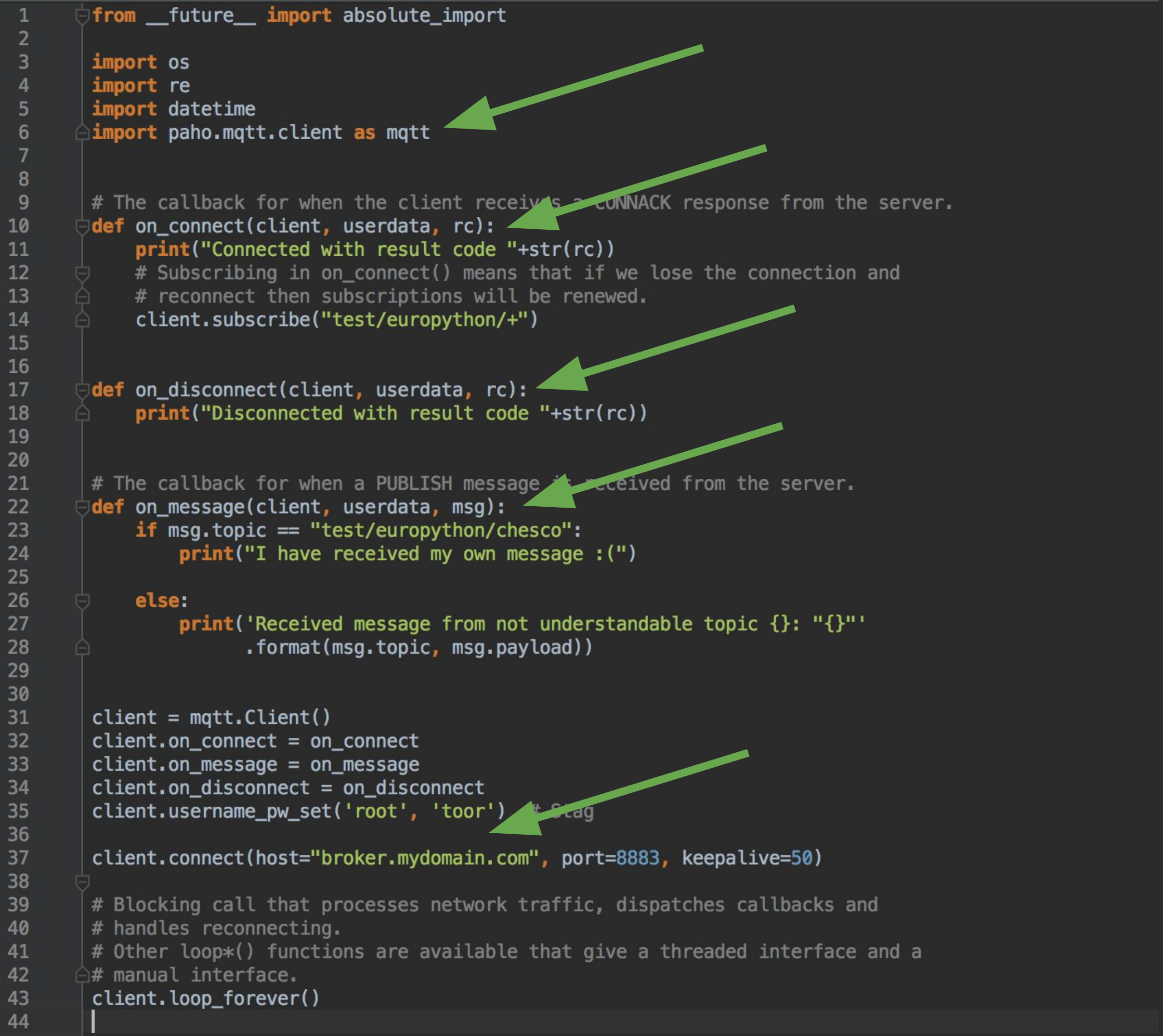
Eclipse library

https://github.com/eclipse/paho.mqtt.python

NO CODE



```python
from __future__ import absolute_import

import os
import re
import datetime
import paho.mqtt.client as mqtt


# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, rc):
    print("Connected with result code "+str(rc))
    # Subscribing in on_connect() means that if we lose the connection and
    # reconnect then subscriptions will be renewed.
    client.subscribe("test/europython/+")


def on_disconnect(client, userdata, rc):
    print("Disconnected with result code "+str(rc))


# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    if msg.topic == "test/europython/chesco":
        print("I have received my own message :(")

    else:
        print('Received message from not understandable topic {}: "{}"'
              .format(msg.topic, msg.payload))


client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.on_disconnect = on_disconnect
client.username_pw_set('root', 'toor')  # Stag

client.connect(host="broker.mydomain.com", port=8883, keepalive=50)

# Blocking call that processes network traffic, dispatches callbacks and
# handles reconnecting.
# Other loop*() functions are available that give a threaded interface and a
# manual interface.
client.loop_forever()
```



```python
from __future__ import absolute_import

import os
import re
import datetime
import paho.mqtt.client as mqtt


# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, rc):
    print("Connected with result code "+str(rc))
    # Subscribing in on_connect() means that if we lose the connection and
    # reconnect then subscriptions will be renewed.
    client.subscribe("test/europython/+")


def on_disconnect(client, userdata, rc):
    print("Disconnected with result code "+str(rc))


# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    if msg.topic == "test/europython/chesco":
        print("I have received my own message :(")

    else:
        print('Received message from not understandable topic {}: "{}"'
              .format(msg.topic, msg.payload))


client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.on_disconnect = on_disconnect
client.username_pw_set('root', 'toor')  # Stag

client.connect(host="broker.mydomain.com", port=8883, keepalive=50)

# Blocking call that processes network traffic, dispatches callbacks and
# handles reconnecting.
# Other loop*() functions are available that give a threaded interface and a
# manual interface.
client.loop_forever()
```

API time

❤️ 🐍 ❤️

Again, options?

Bottle

TASTYPIE

Flask
web development,
one drop at a time

Falcon

Symfony

django
REST
framework

How are we doing?

project requirements

- Real-time data - ✓
- Security ✓
- Obfuscation - ✗
- Authentication - ✗
- 2-way communication - ✓
- Resiliency - ✓

- Lightweight - ✗
- Last known status - ✓
- Debugging - ✓
- Admin site - ✓
- Mobile app api - ✓
- rainbows, etc. - ✓

extras

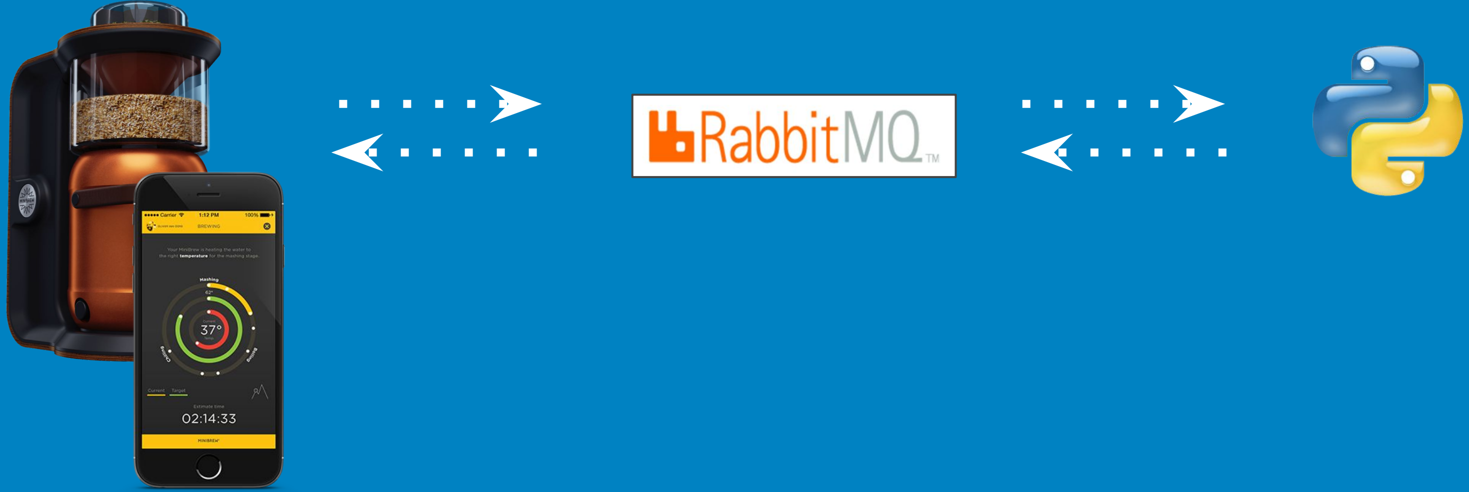
- Scalability - ✓
- Proven technologies - ✓
- small tech stack - ✓
- Error tracking - ✓
- Reduce data transfer - ✓
- Documentation - ✓

❤️ 🐍 ❤️

Authentication

Let Python decide

https://github.com/rabbitmq/rabbitmq-auth-backend-http



How are we doing?

<u>project requirements</u>

- Real-time data ✓
- Security ✓
- Obfuscation ✗
- Authentication ✗
- 2-way communication ✓
- Resiliency ✓

- Lightweight ✗
- Last known status ✓
- Debugging ✓
- Admin site ✓
- Mobile app api ✓
- rainbows, etc. ✓

<u>extras</u>

- Scalability ✓
- Proven technologies ✓
- small tech stack ✓
- Error tracking ✓
- Reduce data transfer ✓
- Documentation ✓

How are we doing?

PROJECT REQUIREMENTS

- Real-time data ✓
- Security ✓
- Obfuscation ✗
- Authentication ✓
- 2-way communication ✓
- Resiliency ✓

- Lightweight ✗
- Last known status ✓
- Debugging ✓
- Admin site ✓
- Mobile app api ✓
- rainbows, etc. ✓

EXTRAS

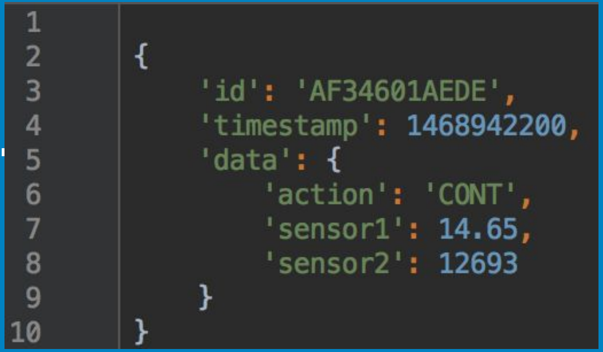
- Scalability ✓
- Proven technologies ✓
- small tech stack ✓
- Error tracking ✓
- Reduce data transfer ✓
- Documentation ✓

Obfuscated and Lightweight messages

Google

Protocol Buffers

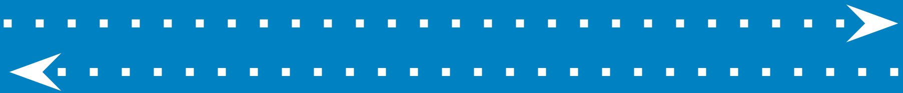
https://github.com/google/protobuf

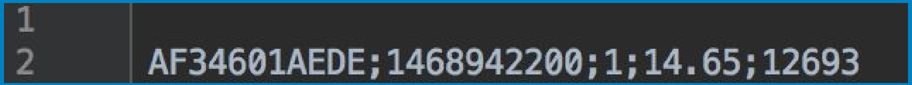


```

{
    'id': 'AF34601AEDE',
    'timestamp': 1468942200,
    'data': {
        'action': 'CONT',
        'sensor1': 14.65,
        'sensor2': 12693
    }
}
```

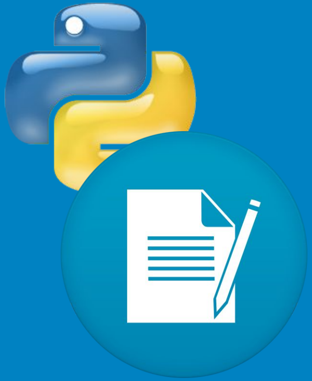
Protocol Buffers

https://github.com/google/protobuf





```

AF34601AEDE;1468942200;1;14.65;12693
```



```
message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;

  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    required string number = 1;
    optional PhoneType type = 2 [default = HOME];
  }

  repeated PhoneNumber phone = 4;
}
```

How are we doing?

PROJECT REQUIREMENTS

- Real-time data - ✓
- Security ✓
- Obfuscation - ✗
- Authentication - ✓
- 2-way communication - ✓
- Resiliency - ✓

- Lightweight - ✗
- Last known status - ✓
- Debugging - ✓
- Admin site - ✓
- Mobile app api - ✓
- rainbows, etc. - ✓

EXTRAS

- Scalability - ✓
- Proven technologies - ✓
- small tech stack - ✓
- Error tracking - ✓
- Reduce data transfer - ✓
- Documentation - ✓

How are we doing?

project requirements

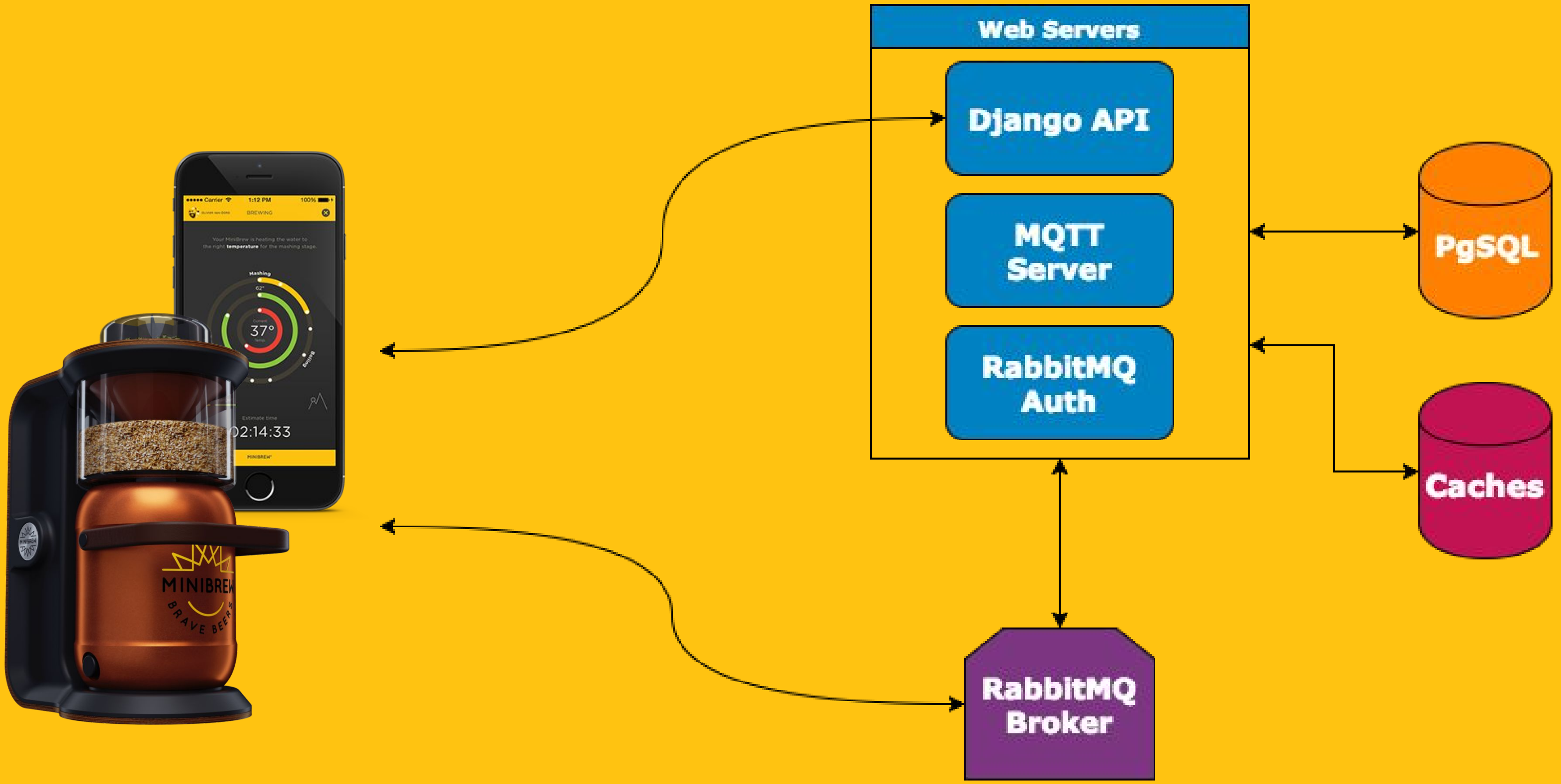
- Real-time data - ✓
- Security ✓
- Obfuscation - ✓
- Authentication - ✓
- 2-way communication - ✓
- Resiliency - ✓

- Lightweight - ✓
- Last known status - ✓
- Debugging - ✓
- Admin site - ✓
- Mobile app api - ✓
- rainbows, etc. - ✓

extras

- Scalability - ✓
- Proven technologies - ✓
- small tech stack - ✓
- Error tracking - ✓
- Reduce data transfer - ✓
- Documentation - ✓

Final Architecture



Wanna roll with us?

Barcelona (ES)

Almere (NL)

elements.nl/careers

Thank You

Questions?