Brewing beer with Python

Chesco Igual
@chescales
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?
So what is it?

Yay!! Awesome!!

(and others)
So what is it?

Yay!! Awesome!!

(and others)
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 -
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
Step by Step
Communications Protocol
Communication Protocols

HTTP
MQTT
XMPP
AMQP
COAP
DDS
Communication Protocols

HTTP

MQTT

XMPP

AMQP

COAP

DDS
Comprehensive IoT Backend Solutions
So then...?

**PROJECT REQUIREMENTS**
- Real-time data
- Security
- Obfuscation
- Authentication
- Two-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
Let’s set up our own
(get a broker... )
Options?

- ActiveMQ
- eMQTT
- CloudMQTT
- HiveMQ
- 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
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_disconnect = on_disconnect
client.on_message = on_message
client.username_pw_set('root', 'toor')

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

# Blocking call that processes network traffic, dispatches callbacks and
# handles reconnecting. Other loops() functions are available that give a threaded interface and a
# manual interface.
client.loop_forever()
```
API time 🏷️❤️❤️
Again, options?

- Bottle
- TASTYPie
- Flask
- Falcon
- Symfony
- REST Framework (highlighted)
How are we doing?

<table>
<thead>
<tr>
<th>PROJECT REQUIREMENTS</th>
<th>EXTRAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Real-time data</td>
<td>- Scalability</td>
</tr>
<tr>
<td>- Security</td>
<td>- Proven technologies</td>
</tr>
<tr>
<td>- Obfuscation</td>
<td>- Small tech stack</td>
</tr>
<tr>
<td>- Authentication</td>
<td>- Error tracking</td>
</tr>
<tr>
<td>- 2-way communication</td>
<td>- Reduce data transfer</td>
</tr>
<tr>
<td>- Resiliency</td>
<td>- Documentation</td>
</tr>
</tbody>
</table>

- Last known status
- Debugging
- Admin site
- Mobile app api
- Rainbows, etc.
Authentication
Let Python decide

https://github.com/rabbitmq/rabbitmq-auth-backend-http
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
Obfuscated and Lightweight messages
Protocol Buffers

https://github.com/google/protobuf
Protocol Buffers

https://github.com/google/protobuf
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 ✔️
- Two-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)

[link](elements.nl/careers)
Thank You
Questions?