Serverless Location Tracking

Using Lambda, S3, and Athena

Jonathan LaCour
CTO, Reliam | Stratalux | G2 Tech Group
Hello, there!

I’m Jonathan LaCour, CTO of Reliam

- Technologist
- Programmer
- Cloud Strategist
- Bourbon Junkie

Nice to meet you!
Connected by a shared vision of the power of the AWS cloud, and deep cultural alignment, AWS Advanced Consulting Partners Reliam, Stratalux and G2 Tech Group have joined forces as a single AWS powerhouse.

United as a single team, the three companies can now offer their customers even greater technical expertise, service and support, and a seamless managed services experience stretching from Boston to L.A.
Agenda

What have I built?

Step 1: Building a Collection Web Service using AWS Lambda
Step 2: Preparing for Amazon Athena with AWS Glue
Step 3: Creating Data Rollups with Scheduled Functions
Step 4: Integrating with My Website

Conclusion
What have I built?
“Current Status Bug”

- Phone battery level and charge state
- Wi-Fi status
- Motion – stationary, walking, driving, biking?
“Now” Page
Shows same detail as current status bug, plus a live map with my current location.
“Map” Page

Map my movement on a single day on a live map. Click on individual data points to show:

- Phone battery level and charge state
- Wi-Fi status
- Motion Type
- Speed
- Altitude

Historical data is only available to me when authenticated.
Step 1: Building a Data Collection Web Service
Overland

Location tracking app for iOS, available in the App Store.

- Collects data about current location, battery state, etc.
- Sends data in configurable batches
- JSON formatted data, easy to understand
- Buffers while you’re offline, sends to server later
- Well documented HTTP API.

https://overland.p3k.app
Overland has a simple, JSON based HTTP API:

- JSON object for each “batch” sent via HTTP POST
- Batch contains a “locations” array.

Now, we need to build a web service to consume these messages and store the structured data in an Amazon S3 bucket.
Introducing Zappa

Zappa is a toolkit to deploy standard Python WSGI apps as web services on top of AWS Lambda and Amazon API Gateway.

https://www.zappa.io
Punytrack is the open source web service I built to power this project.

CORS configuration for my website.

API endpoint for accepting the POST JSON and storing it.
Punytrack: Storage

```python
def store(token, payload):
    # validate
    if token not in conf.tokens:
        return False

    # normalize all location data
    locations = []
    for location in payload.get('locations', []):
        locations.append(normalize(location))

    # determine the current location
    if 'current' in payload:
        current = normalize(payload['current'])
    else:
        current = locations[-1]

    # store the current location
    key = 'current/token-%s/current.json' % token
    s3.put_object(
        Bucket=conf.bucket,
        Key=key,
        Body=json.dumps(current)
    )

    # store the normalized payload
    body = '
'.join(['json.dumps(location) for location in locations]
    key = 'history/token=%s/%s.json' % (token, datetime.utcnow().isoformat())
    s3.put_object(
        Bucket=conf.bucket,
        Key=key,
        Body=body
    )

    return True
```

Validate that the user has a known token.

Store the current location in S3. Note data partitioning key name.

Normalize data into Athena-compatible format. One line of JSON per record.
Store the normalized location data in S3. Note data partitioning key name.
Punytrack: Zappa Deployment

Zappa deployment is easy:

1. Create `zappa_settings.json`
   - Specify WSGI app
   - Configure domain / cert
   - Optionally “keep warm”
   - Scheduled functions
   - IAM permissions

2. Run `zappa deploy production`

Then, you’re done!
Step 2: Preparing for Amazon Athena with AWS Glue
Create an AWS Glue Crawler

AWS Glue Crawler will inspect the data and create a “table” in the data catalog.

Crawler configuration:
- S3 data store path
- Data type (JSON, CSV)
- Target database

Can run once, or on a scheduled basis.
We now have a table with a well-defined schema based upon our structured JSON data.

Note, the partition key ("token") has been discovered.
Amazon Athena: Querying our Table

```
select x, y, altitude, battery_state, battery_level
from cleverdevil_history
where year = 2018
and month = 8
and day = 1;
```

Results:

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>altitude</th>
<th>battery_state</th>
<th>battery_level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-117</td>
<td>33.61208361323396</td>
<td>12</td>
<td>unplugged</td>
<td>0.23000000417232513</td>
</tr>
<tr>
<td>2</td>
<td>-117</td>
<td>33.61203111564477</td>
<td>15</td>
<td>unplugged</td>
<td>0.23000000417232513</td>
</tr>
<tr>
<td>3</td>
<td>-117</td>
<td>33.61208361323396</td>
<td>15</td>
<td>unplugged</td>
<td>0.23000000417232513</td>
</tr>
<tr>
<td>4</td>
<td>-117</td>
<td>33.61022100533983</td>
<td>22</td>
<td>unplugged</td>
<td>0.29999999946029584</td>
</tr>
<tr>
<td>5</td>
<td>-117</td>
<td>33.61030384533983</td>
<td>23</td>
<td>unplugged</td>
<td>0.29999999946029584</td>
</tr>
<tr>
<td>6</td>
<td>-117</td>
<td>33.61406806132651</td>
<td>21</td>
<td>unplugged</td>
<td>0.29999999946029584</td>
</tr>
<tr>
<td>7</td>
<td>-117</td>
<td>33.61004312085057</td>
<td>22</td>
<td>unplugged</td>
<td>0.29999999946029584</td>
</tr>
<tr>
<td>8</td>
<td>-117</td>
<td>33.61406806132651</td>
<td>21</td>
<td>unplugged</td>
<td>0.29999999946029584</td>
</tr>
<tr>
<td>9</td>
<td>-117</td>
<td>33.61789818144384</td>
<td>4</td>
<td>unplugged</td>
<td>0.82999999946029584</td>
</tr>
<tr>
<td>10</td>
<td>-117</td>
<td>33.61789818144384</td>
<td>3</td>
<td>unplugged</td>
<td>0.82999999946029584</td>
</tr>
<tr>
<td>11</td>
<td>-117</td>
<td>33.61789818144384</td>
<td>3</td>
<td>unplugged</td>
<td>0.82999999946029584</td>
</tr>
<tr>
<td>12</td>
<td>-117</td>
<td>33.61789818144384</td>
<td>3</td>
<td>unplugged</td>
<td>0.82999999946029584</td>
</tr>
<tr>
<td>13</td>
<td>-117</td>
<td>33.61789818144384</td>
<td>3</td>
<td>unplugged</td>
<td>0.82999999946029584</td>
</tr>
<tr>
<td>14</td>
<td>-117</td>
<td>33.61789818144384</td>
<td>3</td>
<td>unplugged</td>
<td>0.82999999946029584</td>
</tr>
<tr>
<td>15</td>
<td>-117</td>
<td>33.61789818144384</td>
<td>3</td>
<td>unplugged</td>
<td>0.82999999946029584</td>
</tr>
<tr>
<td>16</td>
<td>-117</td>
<td>33.61789818144384</td>
<td>3</td>
<td>unplugged</td>
<td>0.82999999946029584</td>
</tr>
<tr>
<td>17</td>
<td>-117</td>
<td>33.61789818144384</td>
<td>3</td>
<td>unplugged</td>
<td>0.82999999946029584</td>
</tr>
<tr>
<td>18</td>
<td>-117</td>
<td>33.61789818144384</td>
<td>3</td>
<td>unplugged</td>
<td>0.82999999946029584</td>
</tr>
<tr>
<td>19</td>
<td>-117</td>
<td>33.61789818144384</td>
<td>3</td>
<td>unplugged</td>
<td>0.82999999946029584</td>
</tr>
<tr>
<td>20</td>
<td>-117</td>
<td>33.61789818144384</td>
<td>3</td>
<td>unplugged</td>
<td>0.82999999946029584</td>
</tr>
<tr>
<td>21</td>
<td>-117</td>
<td>33.61789818144384</td>
<td>3</td>
<td>unplugged</td>
<td>0.82999999946029584</td>
</tr>
</tbody>
</table>

Running Query...

Estimated time elapsed: 0 seconds

You can run another query by clicking on the New Query button. The current query will continue to run in the background with the status of all queries in the History Tab.
Step 3:
Creating Data Rollups with Scheduled Functions
Daily Rollups: Improving Performance

Rather than querying using Amazon Athena on every request, we can create a daily rollup. A few easy steps:

- Create function
- Edit `zappa_settings.json`
- `zappa schedule production`

Then, we add an endpoint that fetches static daily rollups from Amazon S3.
Step 4: Website Integration
Website Integration: MapKitJS

Website uses a PHP-based CMS. Simple HTTP calls to the API fetch the data points in JSON format.

Once data points are captured server-side, they are injected into the page as JavaScript for rendering with Apple MapKitJS.
Conclusion
Conclusions

- Serverless web services are easy to build using Lambda and API Gateway, especially with Zappa.
- Amazon S3, Glue, and Athena are a powerful platform for data storage, transformation, and analytics.
- Source code for this project is available on GitHub:
  - github.com/cleverdevil/punytrack
  - github.com/cleverdevil/CleverCustomize/blob/master/Pages/Map.php
  - github.com/cleverdevil/CleverCustomize/blob/master/templates/default/pages/map.tpl.php
Please complete the session survey in the summit mobile app.