Training Part 2
Action, Passion, Integrity
AWS API

- RESTful
- Generally, AWS is API first
- Well supported in several languages
  - Today we will focus on Python = Boto
  - https://aws.amazon.com/sdk-for-python/
- Boto (and other libraries, handle things for you)
- And even if they don’t:
  - Signing = https://github.com/boto/botocore/blob/develop/botocore/auth.py
**API Credentials**

- Protect at all costs!

1. Passing credentials as parameters in the `boto.client()` method
2. Passing credentials as parameters when creating a `Session` object
3. Environment variables
4. Shared credential file (``~/.aws/credentials``)
5. AWS config file (``~/.aws/config``)
6. Assume Role provider
7. Boto2 config file (``/etc/boto.cfg`` and ``~/.boto``)
8. Instance metadata service on an Amazon EC2 instance that has an IAM role configured.
Lambda (tm)
What is AWS Lambda?

- AWS’s “serverless” compute
- Function-as-a-service
- You pay in increments of 200ms and provisioned compute power
- Several supported languages
  - Python
  - Node
  - Go, Java, C#
- Several execution triggers
  - Manual (Provide parameters, get response)
  - AWS Events (Different events in AWS can trigger a Lambda with the event as parameter, CRON)
  - API gateway (Make a web service!)
Shhh... Lambda is just containers
But containers done very well.
What’s the big deal?

- Server no longer matters (that much)
- Massively parallel
- Usually pretty cheap
- The glue for event-based actions in AWS

- Let’s talk about trust again ...
<table>
<thead>
<tr>
<th>OSI Model</th>
<th>Layer</th>
<th>Protocol data unit (PDU)</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host layers</strong></td>
<td>7. Application</td>
<td>Data</td>
<td>High-level APIs, including resource sharing, remote file access</td>
</tr>
<tr>
<td></td>
<td>6. Presentation</td>
<td></td>
<td>Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption</td>
</tr>
<tr>
<td></td>
<td>5. Session</td>
<td>Segment, Datagram</td>
<td>Managing communication sessions, i.e. continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes</td>
</tr>
<tr>
<td></td>
<td>4. Transport</td>
<td>Segment, Datagram</td>
<td>Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing</td>
</tr>
<tr>
<td><strong>Media layers</strong></td>
<td>3. Network</td>
<td>Packet</td>
<td>Structuring and managing a multi-node network, including addressing, routing and traffic control</td>
</tr>
<tr>
<td></td>
<td>2. Data link</td>
<td>Frame</td>
<td>Reliable transmission of data frames between two nodes connected by a physical layer</td>
</tr>
<tr>
<td></td>
<td>1. Physical</td>
<td>Symbol</td>
<td>Transmission and reception of raw bit streams over a physical medium</td>
</tr>
</tbody>
</table>
What to be aware of ...

- Container reuse
- Networking
  - VPC placement / securitygroups
  - Consumes IP space (pre-provisioned)
- Remember, it’s a container
  - Operating System
  - Environment
  - Local storage
- Libraries
  - Must compile binaries on the same version of Amazon Linux
  - Deployment package must be self-contained
Deployment

- IAM
- Packaging
  - An archived (ZIP) directory structure
  - This gets unzipped on the container and is the root of execution
- Upload your package
  - S3
  - Direct (size limitation)
- Versionable
- Logging
  - Cloudwatch Logs
Let’s build some things
Thank you!