ENABLELING FLEXIBLE WORK HABITS WITH LEX CHATBOTS

Gavin Perrie | Solutions Architect
GAVIN PERRIE

Solutions Architect @ Data Reply

- Cloud
- Big Data
- Automation
- Chatbot / Voice Assistants
DATA REPLY DE
SERVICE OFFERING

<table>
<thead>
<tr>
<th>DATA INCUBATOR REPLY</th>
<th>DATA SCIENCE</th>
<th>AB INITIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIG DATA DEVELOPMENT (DEVOPS)</td>
<td>DATA STRATEGY</td>
<td>MICROSTRATEGY</td>
</tr>
</tbody>
</table>
### BIG DATA
**DEVELOPMENT AND IMPLEMENTATION**

<table>
<thead>
<tr>
<th>SPECIALIZATION</th>
<th>USED TECHNOLOGIES</th>
<th>ADVANTAGES</th>
</tr>
</thead>
</table>
| • Specialized in Distributed Open Source & Cloud Technologies | Open Source and Cloud technologies like:  
  o Kubernetes  
  o NoSQL  
  o Hadoop  
  o Spark  
  o Kafka  
  o AWS | • Highly automated solutions  
• Scalable solutions  
• Special customized solutions  
• Solution-finding together with the customer |
| • Knowledge of Architecture: Platform and Application |                     |            |
| • Development and Implementation of Platforms and Applications |                     |            |
FROM MONOLYTIC TO ELASTIC FULLY AUTOMATED ARCHITECTURE
**EMR OVERVIEW**

Amazon EMR cluster

- **Master node**
  - Runs tasks
  - Stores data on HDFS
  - Is optional

- **Core node**
  - Runs tasks
  - Stores data on HDFS

- **Task node**
  - Runs tasks only

**OVERVIEW**

- Manages the cluster
- Distributes jobs and data
- Tracks tasks and monitors health

---

**Master Node**

**Core Node**

**Task Node**
EMR CREATION

Quick Options Go to advanced options

General Configuration

Cluster name [Any EMR cluster]

- Logging

S3 folder s3://aws-logs-eu-west-1/elasticmapreduce/

Launch mode

Cluster Step execution

Software configuration

Release

- core-image-2.8.5-with-ganglia-3.7.2,
- Hue 3.3.4, Hue 4.5.0, Mahout 0.13.0, Pig 0.17.0,
- and Tez 0.6.1

Applications

- HBase: 1.4.8 with Ganglia 3.7.2, Hadoop 2.8.5, Hue 3.3.4, Hue 4.5.0, Phoenix 4.14.0,
- and ZooKeeper 3.4.13

Presto: Presto 0.214 with Hadoop 2.8.5 HDFS and Hive 2.3.4 Metastores

Spark: Spark 2.4.0 on Hadoop 2.8.5 YARN with Ganglia 3.7.2 and Zeppelin 0.8.0

- Use AWS Glue Data Catalog for table metadata

Hardware configuration

Instance type m5.large

Number of instances 3 (1 master and 2 core nodes)

Security and access

EC2 key pair Choose an option

Permissions

- Default

- Custom

Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.

EMR role

- EMR_DefaultRole

EC2 instance profile

- EMR_EC2_DefaultRole
# EMR Creation

## Software Configuration

<table>
<thead>
<tr>
<th>Release</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadoop 2.8.5</td>
<td>Zeppelin 0.8.0</td>
<td>Livy 0.5.0</td>
<td></td>
</tr>
<tr>
<td>JupyterHub 0.9.4</td>
<td>Tez 0.9.1</td>
<td>Flink 1.6.2</td>
<td></td>
</tr>
<tr>
<td>Ganglia 3.7.2</td>
<td>HBase 1.4.8</td>
<td>Pig 0.17.0</td>
<td></td>
</tr>
<tr>
<td>Hive 2.3.4</td>
<td>Presto 0.214</td>
<td>ZooKeeper 3.4.13</td>
<td></td>
</tr>
<tr>
<td>MXNet 1.3.1</td>
<td>Sqoop 1.4.7</td>
<td>Mahout 0.13.0</td>
<td></td>
</tr>
<tr>
<td>Hue 4.3.0</td>
<td>Phoenix 4.14.0</td>
<td>Oozie 5.0.0</td>
<td></td>
</tr>
<tr>
<td>Spark 2.4.0</td>
<td>HCatalog 2.3.4</td>
<td>TensorFlow 1.12.0</td>
<td></td>
</tr>
</tbody>
</table>

**AWS Glue Data Catalog settings (optional)**
- Use for Hive table metadata

**Edit software settings**
- Enter configuration
- Load JSON from S3

```
classification=config-file-name.properties=[myKey1=myValue1,myKey2=myValue2]
```

**Add steps (optional)**

- Auto-terminate cluster after the last step is completed
EMR
FLEXIBLE SCALING AND SIZING

<table>
<thead>
<tr>
<th>Cluster</th>
<th>T-Shirt size</th>
<th>Slaves</th>
</tr>
</thead>
<tbody>
<tr>
<td>small</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>medium</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>large</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>xlarge</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>RAM</th>
<th>vCPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>c5.2xlarge</td>
<td>16</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>c5.xlarge</td>
<td>8</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>m5.2xlarge</td>
<td>32</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>m5.xlarge*</td>
<td>16</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>r5.2xlarge</td>
<td>64</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>r5.xlarge</td>
<td>32</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
EMR
FLEXIBLE SCALING AND SIZING

Scale my cluster

Scale my cluster

Scale my cluster

Scale my cluster

Scale my cluster

Scale my cluster
Why can’t Slack just answer these requests itself?
EMR
FLEXIBLE SCALING AND SIZING

We need a chatbot!!
What do we need to prepare:

- Define what actions the users can do
- Expose the actions through API Gateway
- Decide how to authenticate the requests
- Decide the frontend
Define what actions the users can do

- Start Cluster
- Stop Cluster
- Scale Up/Down
- Scale Out/In
- Extend cluster runtime
Expose the actions through API Gateway
- Decide how to authenticate the requests

```python
def get_mfa_token(self, requester):
    """
    Return the MFA code corresponding to the user
    :param requester: string containing the user username
    :return: 6 digit string
    """

    self.log.info("Getting MFA token")
    dynamo_table = self.dynamodb.Table(self.environ['DYNAMO_MFA_TABLE'])
    ddb_item = dynamo_table.get_item(Key = {'requester': requester})
    secret = ddb_item['Item']['google_authenticator_status'].splitlines()[0]
    secret_pad = secret.ljust(32, '=')

    intervals_no = int(time.time()) // 30
    key = base64.b32decode(secret_pad, True)
    msg = struct.pack(">Q", intervals_no)
    h = hmac.new(key, msg, hashlib.sha1).digest()
    o = h[19] & 15
    h = (struct.unpack(">I", h[o:o+4])[0] & 0x7fffffff) % 1000000
    return h
```
Decide the frontend
CHATBOT
NICE TO HAVE

- Serverless
- Task focused
- AWS Service
The chatbot can understand user requests. It can reply with answers, perform actions, ask for more inputs, or respond with error-handling prompts.

The chatbot fulfills user requests by triggering AWS Lambda. Lambda retrieves the requested information or performs other types of actions.
CHATBOT

AMAZON LEX
Deep learning functionalities
Natural language understanding
Automatic speech recognition

AWS LAMBDA
Runs bot code with serverless compute
Executes code only when triggered by Lex
Scales automatically per request
CHATBOT
BUILDING ONE

- Intents
  The users intentions. What they mean

- Utterances
  What they say

- Slots
  Variables
CHATBOT BUILDING ONE
CHATBOT
BUILDING ONE

def authenticateUserIntent(self, intent_request):
    self.log.info("Authenticating user")
    session_attributes = intent_request['sessionAttributes']
    if intent_request['SessionAttributes'] is not None else ()

    table = self.dynamodb.Table(self.env["DYNAMO_TABLE"])
    requester = self.get_slots(intent_request)["username"].lower()
    password_given = self.get_slots(intent_request)["password"]

    self.log.info(session_attributes)

    if self.check_mfa_enabled(requester):
        if len(password_given) == 6:
            response = table.get_item(Key = 
                "requester": requester)
            self.log.info(response)

            if "Item" not in response:
                user_response = "I didn’t find any lab for “ + requester
            else:
                password_stored = self.get_mfa_token(requester)
                if len(password_stored) == len(password_given):
                    user_response = "Naja " + requester + " thanks for authenticating, how can I help?"
                    session_attributes["authenticated_user"] = requester
                else:
                    user_response = "Sorry, I wasn’t able to authenticate you "

            self.log.info(user_response)
CHATBOT
ENOUGH TALK
SHOW ME A DEMO!!!
REPLY BOOTH G07

At the Reply booth you will get to know more about the company’s portfolio in the fields of:

- MACHINE LEARNING
- IOT
- DATA SCIENCE
- CLOUD
- DEVOPS
- LEAN & AGILE
BENEFITS

Chatbot enabled
TAKEAWAYS
LESSONS LEARNED

- Build multiple, dedicated chatbots – not one huge one
- Security by design – include it from the beginning
- Expose management services through APIs – gives flexibility
- Cost wasn’t our main driver
QUESTIONS?
THANK YOU

g.perrie@reply.de
@gavinperrie

www.reply.com