Who am I?

- Professional Geek since 1987
- Java/J2EE/Java EE since 1999
- Cloud since 2012
- Roles include:
  - Architect
  - Developer
  - Project Manager
  - DBA
  - System Admin
Discussion Resources

• This slide deck
  – http://www.slideshare.net/derekashmore

• Sample code on my Github
  – https://github.com/Derek-Ashmore/

• Sample Java AWS Lambda Source
  – https://github.com/Derek-Ashmore/AWSLambdaExamples

• Slide deck has hyper-links!
  – Don’t bother writing down URLs
Agenda

- The “What” and “Why” of AWS Lambda
- Developing Lambda
- Deploying Lambda
- Tales from the Field
- Summary / Q&A

©2016 Derek C. Ashmore, All Rights Reserved
What are AWS Lambdas?

• You provide custom code -> AWS runs it
  – Java, Node.js, Python
• Computing power with less management
  – AWS manages the hardware
  – AWS autoscales that hardware
  – AWS maintains that hardware
• Lambdas are event driven
  – API Gateway (e.g. RESTful Web Service call)
  – Many more
• Lambdas are stateless
• Not to be confused with “Lambda Expressions” in Java 8
Lambda Event Sources

- API Gateway
- SNS Messaging Subscriptions
- Schedule
- Storage writes
  - S3, DynamoDB, Kenesis
- SES Email receipt
- Cloudwatch
  - Schedule, Events, log entries
- Cognito (Security)
- CloudFormation
  - Creation script
What’s the Business Benefit

• Less Maintenance Hassle
• Unlimited Parallelism
• Current cost advantage
  – Don’t pay for idle
  – CPU cost currently lower
    • Free tier
      – 1 M executions and 400K compute seconds per month
      – Memory allocated determines allowed free-tier runtime
    • 20 cents per 1 M executions + memory/runtime cost
  – Administration cost
    • No O/S upgrades, server backups, etc.
There’s no free lunch

• Less control over environment
  – Harder to tune
  – Memory and time limits on execution

• Few Environment amenities
  – No connection pooling, session support, caching

• Proprietary Interface
  – Potential Technical Lock-in

• No Guarantee that AWS cost will be constant
  – Potential Business Risk

• Modern version of CGI
Lambda Competitors

• Azure Functions (here)
  – Closest matching feature set
    • Large number of event types
    • Node.js and C# Language Support
      – Claim to support more, but not documented
    • Pricing Model Similar
    • Better developer support (IDE integrated with portal)

• Google Cloud Functions (here)
  – Still Alpha
  – Fewer event types
  – Node.js language support
Lambda API

• Create a Request Handler
  – Inputs are
    • Event input (user input arguments)
    • Context input (info about execution environment)
  – Outputs are user decided
• JSON marshalling of Request and Response
• Execution Context
• Deployment Zip Artifact
  – Typically one RequestHandler per Zip artifact
Lambda Request / Response Sample

• Expose REST API that collects email addresses for a mailing list
  – https://scrubbed/prod/EmailCollector

• Sample request/response

```json
{
  "firstName": "Derek",
  "lastName": "Ashmore",
  "emailAddress": "testme@test.com"
}
```
Java Request Handler Example

```java
public class EmailCollectorLambda implements RequestHandler<EmailCollectionRequest, EmailCollectionResponse> {
  @Override
  public EmailCollectionResponse handleRequest(EmailCollectionRequest input, Context context) {
    try {
      return new EmailCollector().collect(input);
    }
    catch (Exception e) {
      throw new ContextedRuntimeException(e)
        .addContextValue("emailCollectionRequest", input)
        .addContextValue("context", ContextUtils.toString(context));
    }
  }
}
```

- **RequestHandler** interface is generic.
  - POJOs represent the request and response
  - POJOs determine JSON request/response format
  - Execution **Context** class provided by AWS
- This class specified when Lambda defined
- Note the wrapping **try/catch**
  - Portion of the Context provided by AWS -> need more
  - AWS Does marshalling
  - **ContextedRuntimeException** from Apache Commons **Lang3**
- Note that the Lambda is thin -> Business logic is elsewhere
- Sample is on GitHub ([here](#))
- Node.js and Python similar
What’s in the Context?

• Execution Context provided by AWS
• Contains:
  – AWS Request ID -> Get logs for specific request
  – Function name, version, arn
  – Cognito identity
  – Remaining time/memory
  – Mobile client information (AWS Mobile SDK)
    • Environment name/value map
    • Custom name/value map
Resource Support

- **Runtime Support**
  - you can configure memory and max run time available
  - Java → Java 8; AWS SDK jar must be included in your deployment
  - Node.js → AWS SDK for Node.js preinstalled
  - Python → AWS SDK for Python preinstalled
- **All logging viewable/searchable in Cloudwatch logs**
  - Node.js and Python log to the console
  - Java uses customer Log4J Appender
- **Third party jars/libraries can be included**
  - Including AWS SDK for AWS Tasks (executing other Lambdas)
  - Database Drivers
  - Web service libraries
- **Keep in Mind**
  - You create/destroy all database connections
    - No connection pooling (Lambdas are stateless)
  - Caching APIs have limited benefit (Lambdas are stateless)
  - No Remote Debug capability
Agenda

- The “What” and “Why” of AWS Lambda
- Developing Lambda
- Deploying Lambda
- Tales from the Field
- Summary / Q&A
Lambda Deployment Package

• Zip Organization (Java example)
  – Root is in the classpath
  – Lib contains 3rd party jars

• Maven example here
  – Need maven-dependency-plugin, maven-antrun-plugin
API Gateway

- Exposes Lambdas as a RESTful Web Service
  - Can be publicly available or part of a secured private microservice library

```
https://scrubbed/beta/EmailCollector

{
  "firstName": "Derek",
  "lastName": "Ashmore",
  "emailAddress": "testme@test.com"
}
```
Gateway Lambda Integration

- Integrations do basic transformations
  - Map headers and parameters to Lambda request fields
Gateway Integration Example

```
{
    "Name": "API Gateway Test",
    "Email": "customer@money.com",
    "Message": "Please contact me about breaking my monolith!"
}
```

```
#set($inputRoot = $input.path('/$'))
{
    "params": {
        "apiId": "3b676d18-3d7f-494f-8136-fe733b",
        "apiSecretId": "7062b0a0d63a5e348760e100f0"
    },
    "subject": "Break The Monolith Contact Request",
    "message": "This is a customer contact request from: $inputRoot.Name
Name: $inputRoot.Name
Email: $inputRoot.Email
Message: $inputRoot.Message",
    "fromName": "Break The Monolith Site",
    "fromEmail": "derek.ashmore@dvtconsulting.com",
    "toName": "Break The Monolith",
    "toEmail": "sales@breakthemonolith.guru"
}
```
Lambdas and SNS Topics

- Lambdas can subscribe to SNS Publish/subscribe topics
- Request Message is type `SNSEvent`
Lambdas can be scheduled

- Lambda executions can be scheduled through CloudWatch
  - Cron expressions supported
Agenda

The “What” and “Why” of AWS Lambda

Developing Lambda

Deploying Lambda

Tales from the Field

Summary / Q&A
Chief Complaints

• Documentation leaves a lot to be desired
  – This is an understatement.
  – For Java Lambdas, you are almost on your own.

• Lambda start-up time not consistent
  – Sometimes long start-up time for JVM
  – Python is the fastest

• Optimizations that depend on state aren’t as easy
  – You would have to persist that state
    • This would have it’s own concurrency and performance issues
Implementation Tips

• Separation of Concerns
  – Keep Lambda code separate from business logic
  • Might want to change vendors someday
  – Keep AWS SDK code separate from business logic
  • Same reason
  – Invoke other Lambdas through the API Gateway, not directly through the AWS SDK
  • Same reason
  – Keep Business Logic locally runnable/debuggable
  • Remote debug isn’t yet possible

• Ensure you can always tie AWS Request Id to your business transaction
  – Need a way to gather logs from a complete business transactions and the many services it might use
  – All invocations get unique AWS request Ids
  • For example, lambda invokes other lambdas
  – Configure log4j layout (Java) to include AWS Request Id (example)
  – Node.js and Python logs have request id automatically
Common Use Cases

• Processing for uploaded data
  – Image processing
• Low-volume web site features
  – Paying for idle cost prohibitive
• Scheduled Batch Work
  – Break up batch by invoking other Lambdas
  – The resulting scaling is delegated to AWS
Performance

• Start-up Time – Python, Node.js, Java
  – Berezovsky performance test

• Throughput – Java and Node.js
  – Both have a JIT
  – DZone comparison here
  – Close enough that other factors would likely guide your language choice
Lambdas and Microservices
Using Lambdas as Microservices

• Lambda / API Gateway is a deployment option for microservices
  – No differences in design principles
    • Single purpose
    • Self-contained
  – Still design for failure
    • Don’t “assume” that Lambda outages can’t happen
      – A Lambda might need external resources that aren’t available

• Off Limits: Coding patterns that use state
  – Lambdas must be stateless
  – Fail fast patterns
    • Service Call Mediator
    • Circuit Breaker
  – Performance Patterns
    • Expiring Cache (API Gateway allows request caching)
Lambda and the Gartner Hype Cycle

- Peak of Inflated Expectations
- Plateau of Productivity
- Slope of Enlightenment
- Trough of Disillusionment
- Technology Trigger
Further Reading

• This slide deck
  – http://www.slideshare.net/derekashmore

• AWS Lambda Reading List
Questions?

• Derek Ashmore:
  – Blog: www.derekashmore.com
  – LinkedIn: www.linkedin.com/in/derekashmore
  – Twitter: https://twitter.com/Derek_Ashmore
  – GitHub: https://github.com/Derek-Ashmore
  – Book: http://dvtpress.com/