AWS Lambda and Endless Serverless Possibilities

Abhishek Tiwari

Citation: *A. Tiwari*, ”AWS Lambda and Endless Serverless Possibilities”, Abhishek Tiwari, 2015. doi:10.59350/aekh9-abz94

Published on: October 26, 2015

Ever since the Amazon introduced the AWS Lambda service at AWS re:Invent 2014, a variety of new applications for the service has emerged which highlights tremendous potential and traction for the AWS Lambda service. Over last one year, Amazon has been actively working towards integrating other AWS services with AWS Lambda. You can consider recently released Amazon ElasticSearch service as a good example which heavily relies on AWS Lambda to stream the data from other AWS services such as Amazon CloudWatch, Amazon S3, etc. into ElasticSearch. With recent buzz around microservices and launch of Amazon API Gateway, AWS Lambda has emerged as preferred choice to develop serverless microservices. On top of that, in recent re:Invent AWS Lambda was further expanded with new features such as versioning and scheduled jobs, and introduced support for other popular languages including Python and PHP.

Serverless architecture

Just to be clear, a serverless architecture doesn't mean servers are no longer involved. It simply means we as the developer have no longer to think about provisioning, scaling and operating the servers but under the hood this is a something managed by cloud service provider. Honestly, the concept of serverless architecture is not something new. Heroku started supporting serverless architecture way back in 2010. Having said that, AWS Lambda offers a very different pricing (subsecond billing) and computing model (event driven). In terms of completeness, with the introduction of AWS Mobile Hub AWS serverless stack can be considered as par with or better than Parse's serverless stack.

A typical 3-tier business application can be developed using a serverless architecture on AWS. Let's take 3-Tier application as an example, a serverless architecture on AWS will need following features

- Client tier: Content Delivery via Amazon CloudFront. In this case, a Client-Side UI developed using a JavaScript based MVC framework such as Angular which interacts to business tier using the APIs.
- Business tier: Amazon API Gateway with published APIs with AWS Lambda as the backend service, authentication using AWS Lambda with AWS IAM, business logic execution using AWS Lambda, task queue processing using Amazon SQS, etc. Depending on the application requirements, this layer can include Amazon Elastic Transcoder for video encoding, Amazon SNS for push notification, Amazon SES for outbound and inbound email delivery, and Amazon Cognito for mobile app identity management and user data syncing, Amazon Machine Learning for predictions. Having said that, AWS Lambda sits in the core of all these possibilities.
- Data tier: SQL Datastore using Amazon RDS/Aurora, NoSQL Datastore using Amazon DynamoDB, Cache service using Amazon ElasticCache, Blob store using Amazon S3 - all of these using AWS Lambda functions.

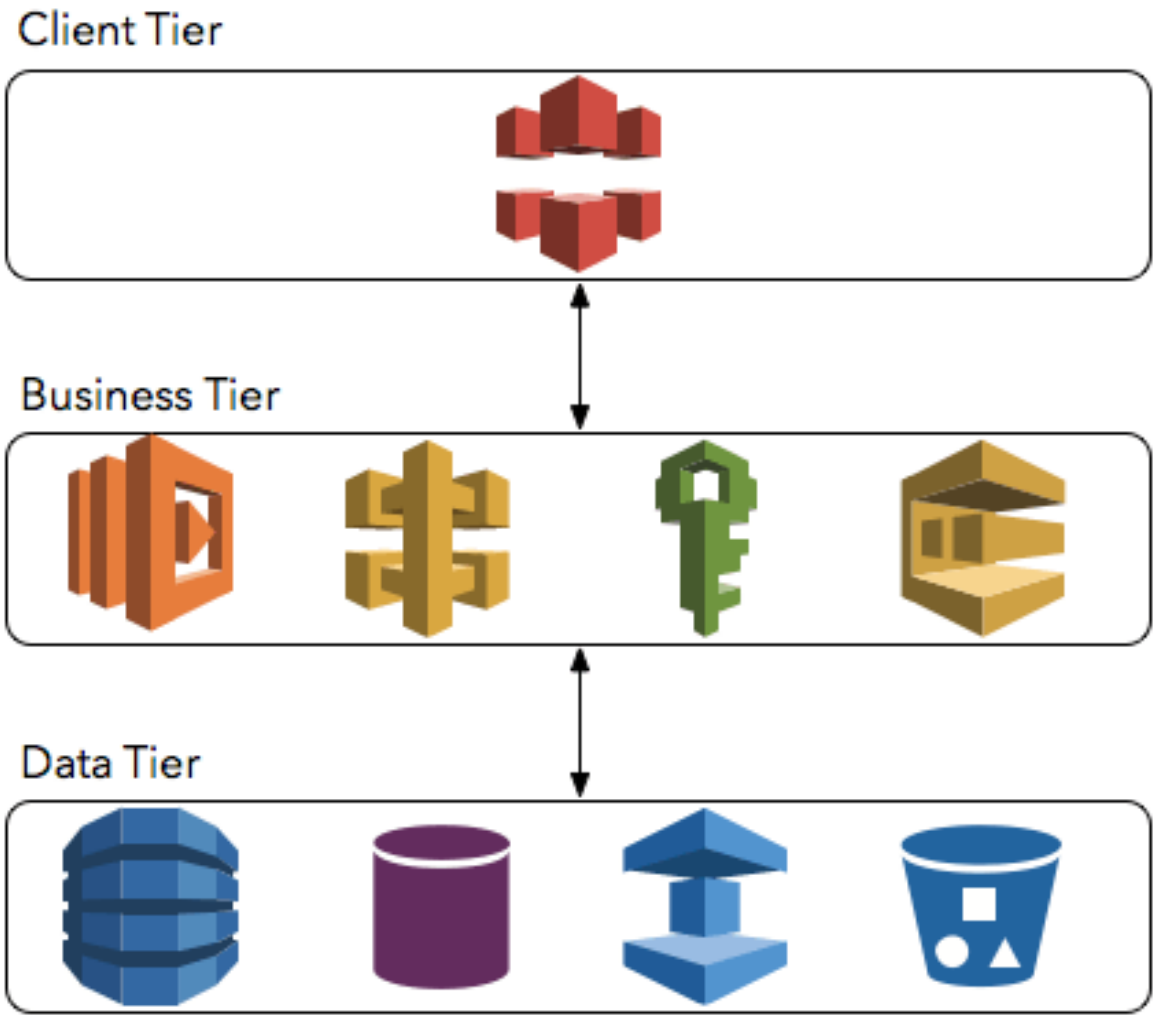

Figure 1: 3-tier business application using AWS Lambda and other AWS services

Event-driven serverless computing

AWS Lambda can be considered as a perfect example for the event-driven serverless computing which means no more upfront capacity provisioning. Lambda executes your code in response to events. Within milliseconds after an event trigger, Lambda automatically provisions compute resources for you and runs you code or Lambda function. An event can be a change to an Amazon S3 bucket, an update to an Amazon DynamoDB table, or custom events generated by various applications or devices. Lambda to eliminates the issue related to unused server capacity without sacrificing scalability or responsiveness.

AWS Lambda Blueprints

AWS Lambda blueprints is a collection of curated but reusable and extendable lambda functions. You can consider blueprints more like recipes with sample configurations of event sources and Lambda functions. A range of scenarios are covered in these blueprints, hence a good starting point to learn how AWS Lambda actually works.

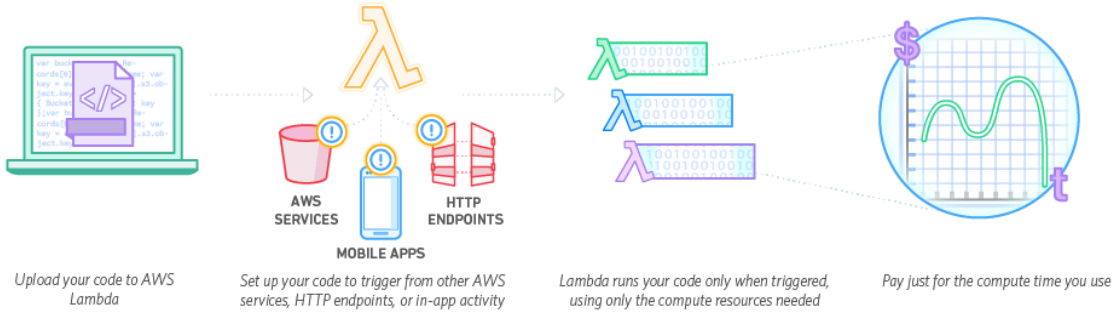

Figure 2: How AWS Lambda works? Image credits AWS.

Endless serverless possibilities

One thing really amazes me is the endless possibilities when you decide to use the AWS Lambda.

Serverless startups

AWS Lambda is a perfect match for startups who are either in early stage and don't want waste a lot of money on in the infrastructure cost or startups those are focusing on solving the problem and less interested in infrastructure management. Startups such as Gousto has migrated from their monolithic application architecture to microservices driven architecture by heavily utilising the AWS Lambda. Ability to scale and be responsive during traffic surges are quite critical for a high-growth startups. With AWS Lambda, startups are not compromising anything on the scalability, or on the availability, or on the performance.

Serverless microservices

Using AWS Lambda, you can easily build scalable microservices for mobile, the web, and IoT applications without managing infrastructure. With Amazon API Gateway and newly introduced versioning feature, building microservices has become a lot easier than last year. Take an example of a dynamic

image thumbnailing application, a range of image manipulations can be performed over the API exposed via Amazon API Gateway with AWS Lambda as image manipulation engine.

In the following setup, a client sends a request to Amazon CloudFront for an image manipulation. If image manipulation response is available in the CloudFront cache with valid cache control header then it will respond back with cache image. Otherwise, the request is passed to API Gateway which maps path based parameter to Lambda functions which perform the image manipulation on the fly and responds back with a new image. This image will be cached for future CloudFront requests.

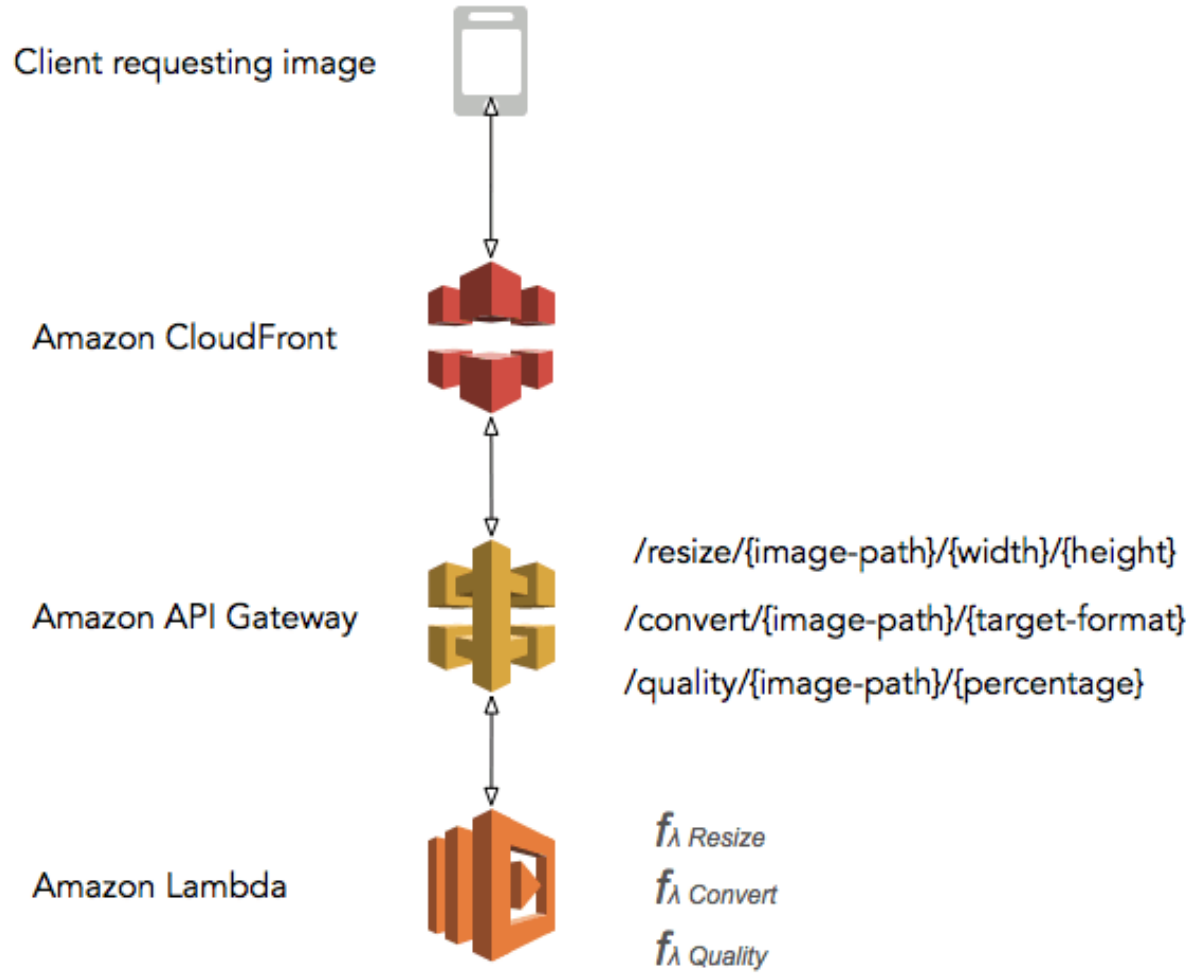

Figure 3: Image Manipulation as Service using AWS Lambda. A path based parameter mapping is used here for API gateway to avoid query string based caching in Amazon CloudFront.

Serverless testing

AWS Lambda includes the blueprints for running unit and load testing. In fact, blueprint library includes a simple framework for conducting various tests of your Lambda functions. As illustrated below, test functions are just like any other normal AWS Lambda functions and they accept function name which is required to be tested as the parameter when test functions are invoked.

```
var load = function(event, context) {
    var payload = event.event;
    asyncAll({
        times: event.iterations,
        fn: function(next, i) {
            payload.iteration = i;
            var lambdaParams = {
                FunctionName: event.function,
                InvocationType: 'Event',
                Payload: JSON.stringify(payload)
            };
            lambda.invoke(lambdaParams, function(err, data) {
                next();
            });
        },
        done: function() {
            context.succeed('Load test complete');
        }
    });
};
```

Serverless analytics

AWS Lambda can be an excellent utility to build low-cost analytics solution. For example, my blog currently runs Snowplow web analytics tracker which sends data to Amazon S3 via CloudFront logging function. Once data arrives in S3, it triggers an AWS Lambda function to immediately process the raw log shipped to S3 bucket by CloudFront logging function. This Lambda function extracts raw data from the log files, transforms them into a canonical format, perform enrichment and loads final data into the DynamoDB table which is connected to a D3.js dashboard served from another S3 bucket. All of this cost me less than 5$ a month. Alternatively, I can use Amazon Elasticsearch Service as dashboard but it will cost me a bit more.

Serverless stream processing

Using AWS Lambda you can send the clickstream data to Amazon Kinesis Stream and from there we can analyse the data in the real-time. In this case, AWS Lambda acts as highly-scalable data collector

API, sitting behind the Amazon API Gateway proxy. Lambda function accepts the clickstream payload sent by the client via pixel request and then put this payload into the Amazon Kinesis Stream. Data from Amazon Kinesis Stream is either picked by Amazon EMR or the Amazon Elasticsearch Service for analysis and displayed as the dashboard.

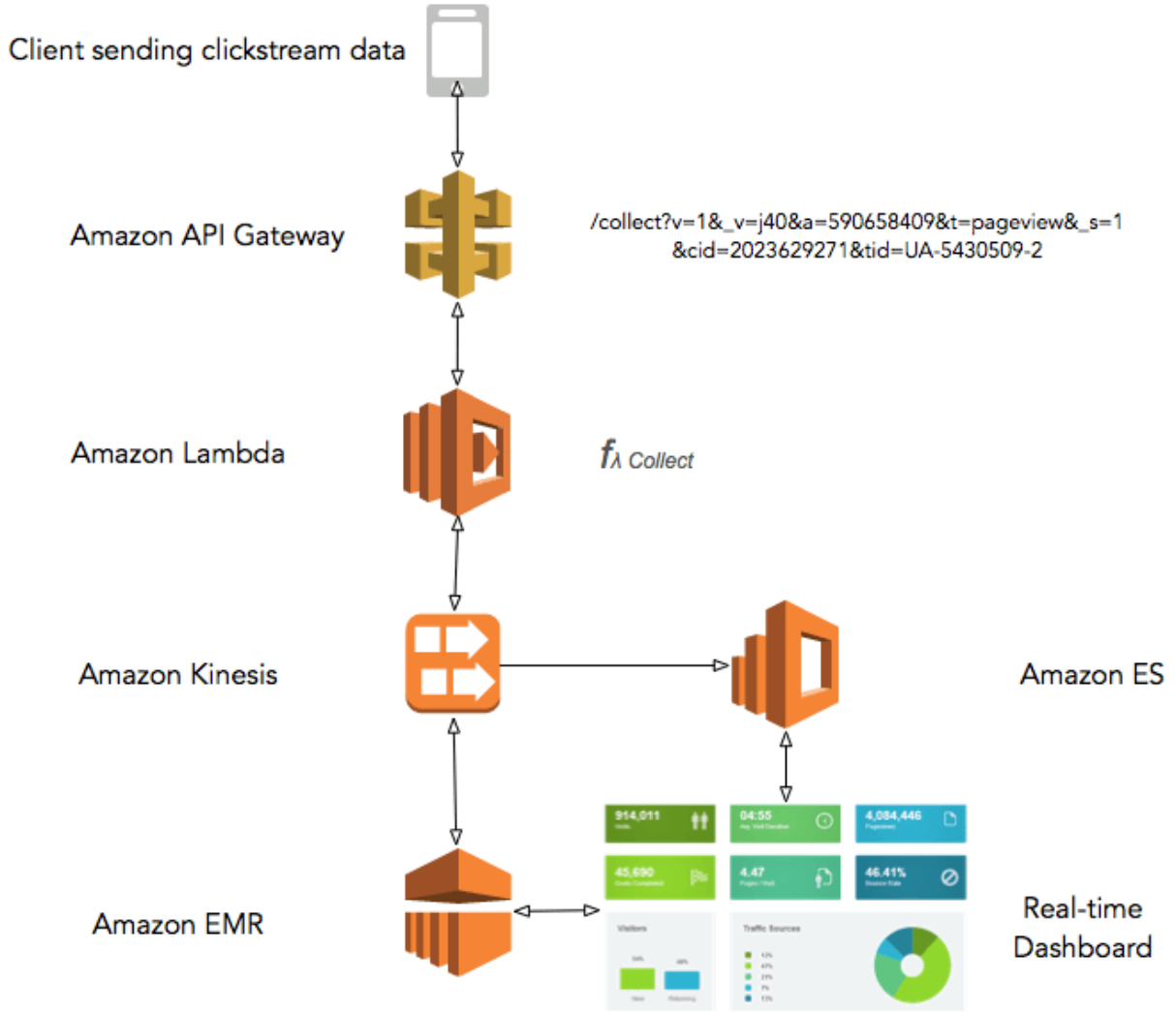

Figure 4: AWS Lambda and Streaming Analytics

Serverless ETL

Using AWS Lambda one can create the serverless ETL pipeline. For example, if you are storing your clickstream log files to Amazon S3 bucket then an AWS Lambda function can be triggered whenever a new log file is added to your bucket. This AWS Lambda function (`Extract`) will extract metadata of the log file and put as a record into Amazon Kinesis Stream which will be picked by another AWS

Lambda function (`Transform`). The AWS Lambda function (`Transform`) will then transform the data and push transformed record in another Amazon Kinesis Stream which. This Amazon Kinesis Stream is attached to Amazon Kinesis Firehose which loads transformed records into the Amazon RedShift. Alternatively, once Amazon Kinesis Analytics is publically available you can use it instead of AWS Lambda do transformation part by running the standard SQL queries against Amazon Kinesis Stream data.

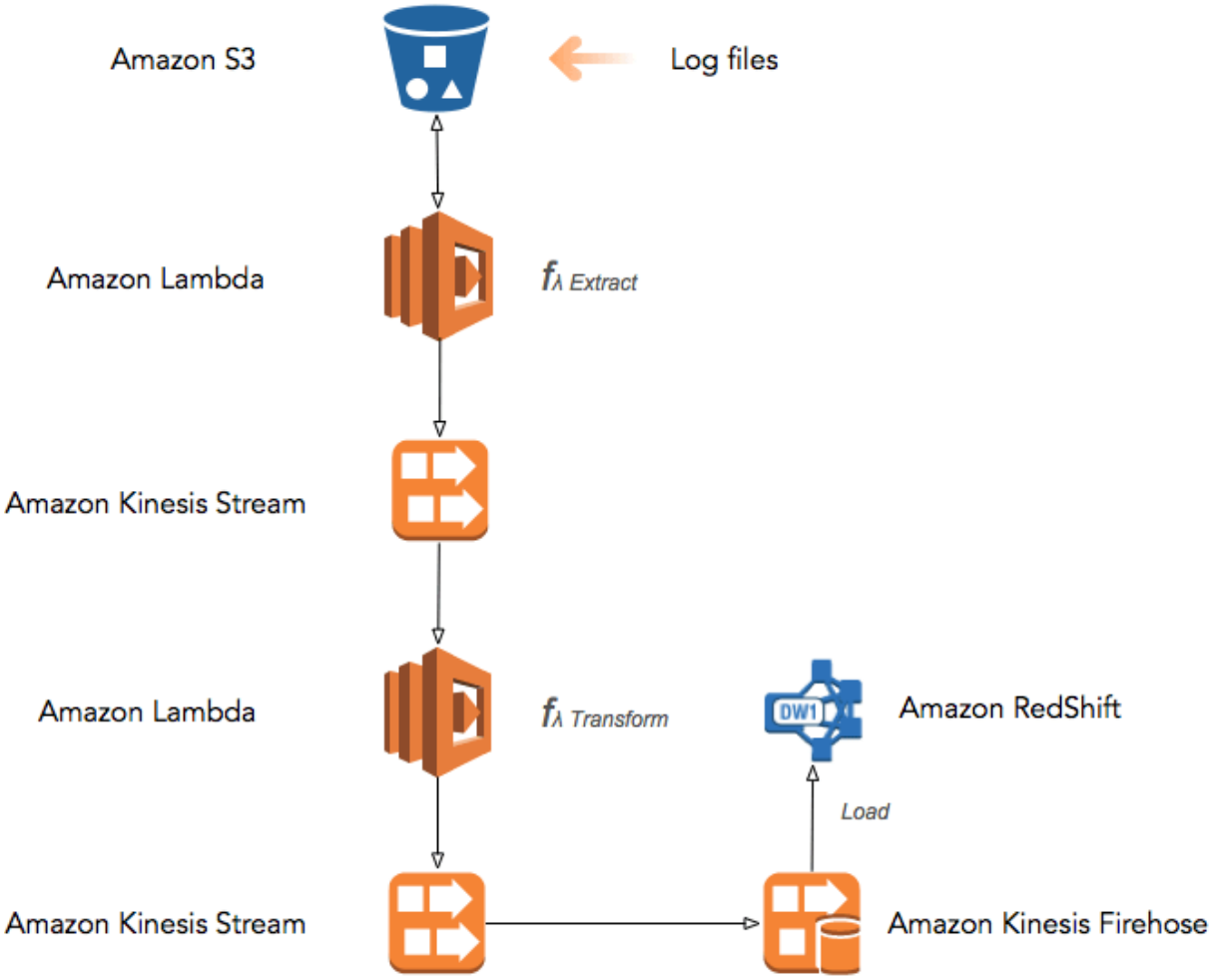

Figure 5: AWS Lambda Serverless ETL

Serverless statistics

AWS Lambda allows us to run arbitrary executables which means one can basically run a portable and lightweight version of R wrapped by a Lambda function written in any supported language. This Lambda function will be the entry point for the R executable and can be called as API if you are using Amazon API Gateway

Serverless machine learning

Last but not least, you can always use AWS Lambda in conjunction with Amazon Machine Learning to either do real-time predictions for streaming data or on-demand prediction by exposing a Lambda function wrapping Amazon Machine Learning as API.

Closing remarks

So there you have it, endless serverless possibilities once you mix the AWS Lambda with other AWS services.