





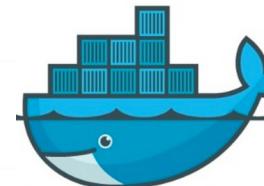
Serverless Architectures with AWS Lambda and MongoDB Atlas



Sig Narváez
Sr. Solutions Architect
sig@mongodb.com
@SigNarvaez

Serverless

Where will my code run?



Big Iron

Commodity
Hardware

Virtualized

Containers

Functions

Forbes: 5 factors fuelling Serverless Computing

1. Cloud services have matured
 2. BaaS → “*SaaS-ification*”
 3. API's are the glue
 4. Containers – *now per function*
- 5. SysOps → DevOps → NoOps**
Less Ops, More Engineering

<https://www.forbes.com/sites/janakirammsv/2016/02/28/five-factors-that-are-fueling-serverless-computing-part-1>



WE LIVE IN A SOFTWARE DEFINED ECONOMY

EVERY COMPANY IS A SOFTWARE COMPANY

DEVELOPERS ARE VALUABLE RESOURCE

BUILD VALUE – NOT OPERATIONS

Serverless architecture

TRIAL ?

Serverless architecture is an approach that replaces long-running virtual machines with ephemeral compute power that comes into existence on request and disappears immediately after use. Since the last Radar, we have had several teams put applications into production using a "serverless" style. Our teams like the approach, it's working well for them and we consider it a valid architectural choice. Note that serverless doesn't have to be an all-or-nothing approach: some of our teams have deployed a new chunk of their systems using serverless while sticking to a traditional architectural approach for other pieces.

Worth pursuing. It is important to understand how to build up this capability. Enterprises should try this technology on a project that can handle the risk.

NOV
2016

TRIAL ?

Serverless architecture is an approach that replaces long-running virtual machines with ephemeral compute power that comes into existence on request and disappears immediately after use. Since the last Radar, we have had several teams put applications into production using a "serverless" style. Our teams like the approach, it's working well for them and we consider it a valid architectural choice. Note that serverless doesn't have to be an all-or-nothing approach: some of our teams have deployed a new chunk of their systems using serverless while sticking to a traditional architectural approach for other pieces.

APR
2016

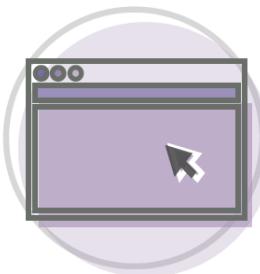
ASSESS ?

Serverless architecture replaces long-running virtual machines with ephemeral compute power that comes into existence on request and disappears immediately after use. Examples include [Firebase](#) and [AWS Lambda](#). Use of this architecture can mitigate some security concerns such as security patching and SSH access control, and can make much more efficient use of compute resources. These systems cost very little to operate and

Thoughtworks Technology Radar

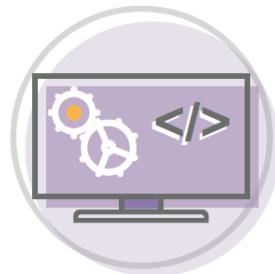


Common use cases



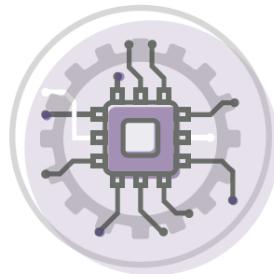
Web applications

- Static websites
- Dynamic web apps
- Packages for Flask and Express



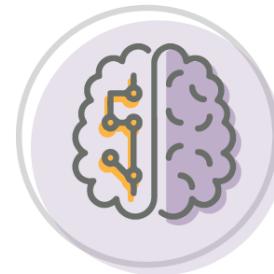
Back ends

- Apps & services
- Mobile
- IoT



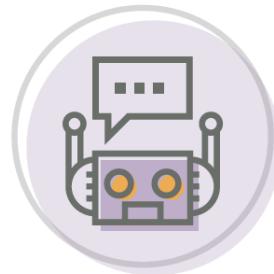
Data processing

- Real time
- MapReduce
- Batch



Amazon Alexa

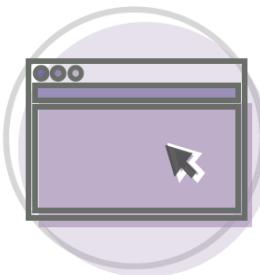
- Powering voice-enabled apps
- Alexa Skills Kit



Chatbots

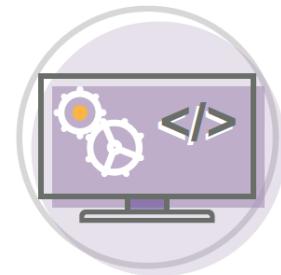
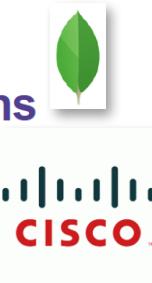
- Powering chatbot logic

Common use cases



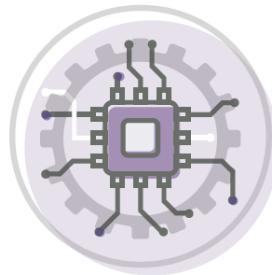
Web applications

- Static websites
- Dynamic web apps
- Packages for Flask and Express



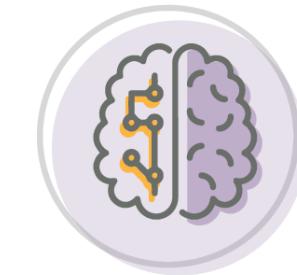
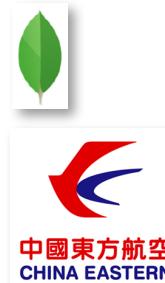
Back ends

- Apps & services
- Mobile
- IoT



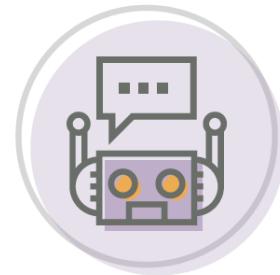
Data processing

- Real time
- MapReduce
- Batch



Amazon Alexa

- Powering voice-enabled apps
- Alexa Skills Kit



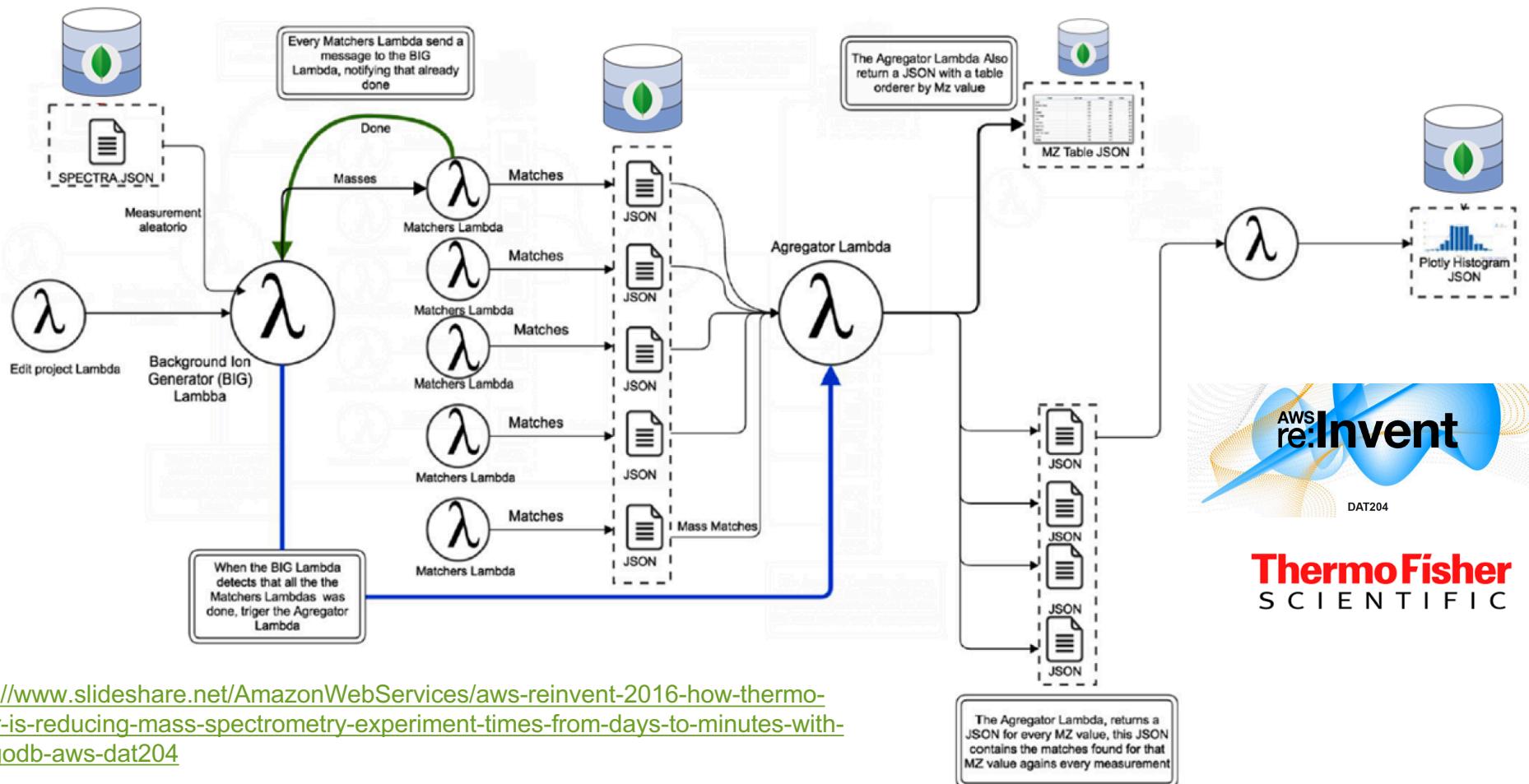
Chatbots

- Powering chatbot logic



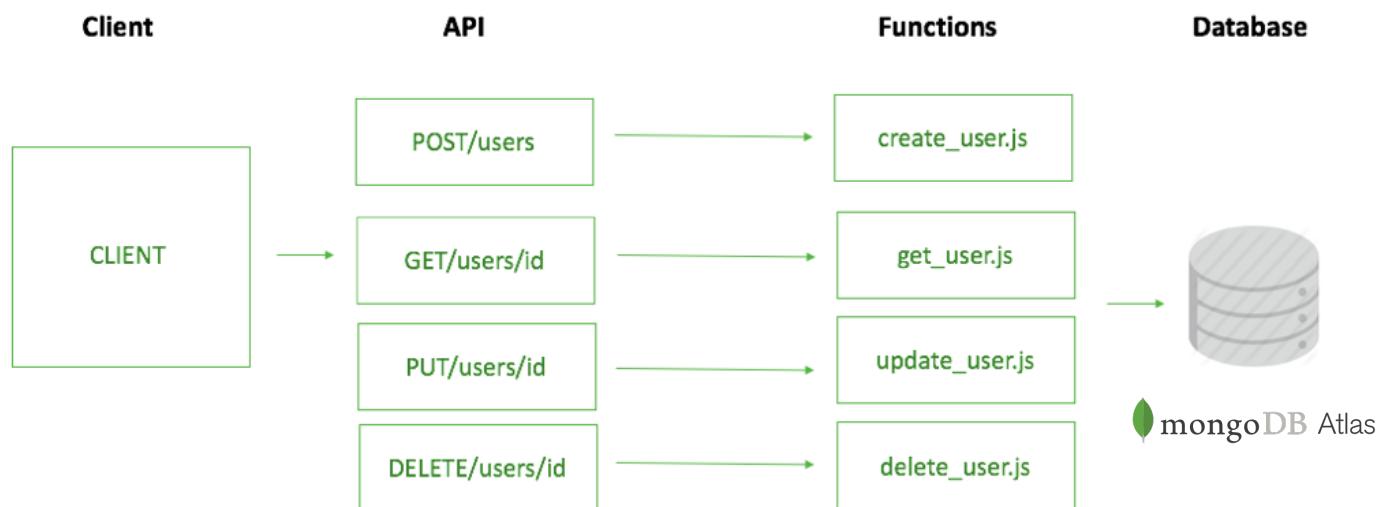
Parallel data processing

@SigNarvaez | @MongoDB | @BigDataDayLA

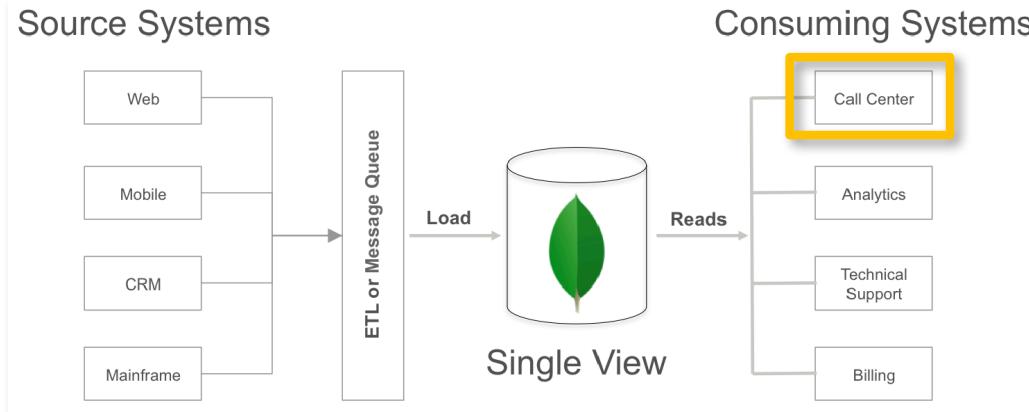


Microservices

Serverless Microservice



Customer Single View - Insurance Industry (hypothetical)



High-level architecture of a single view platform

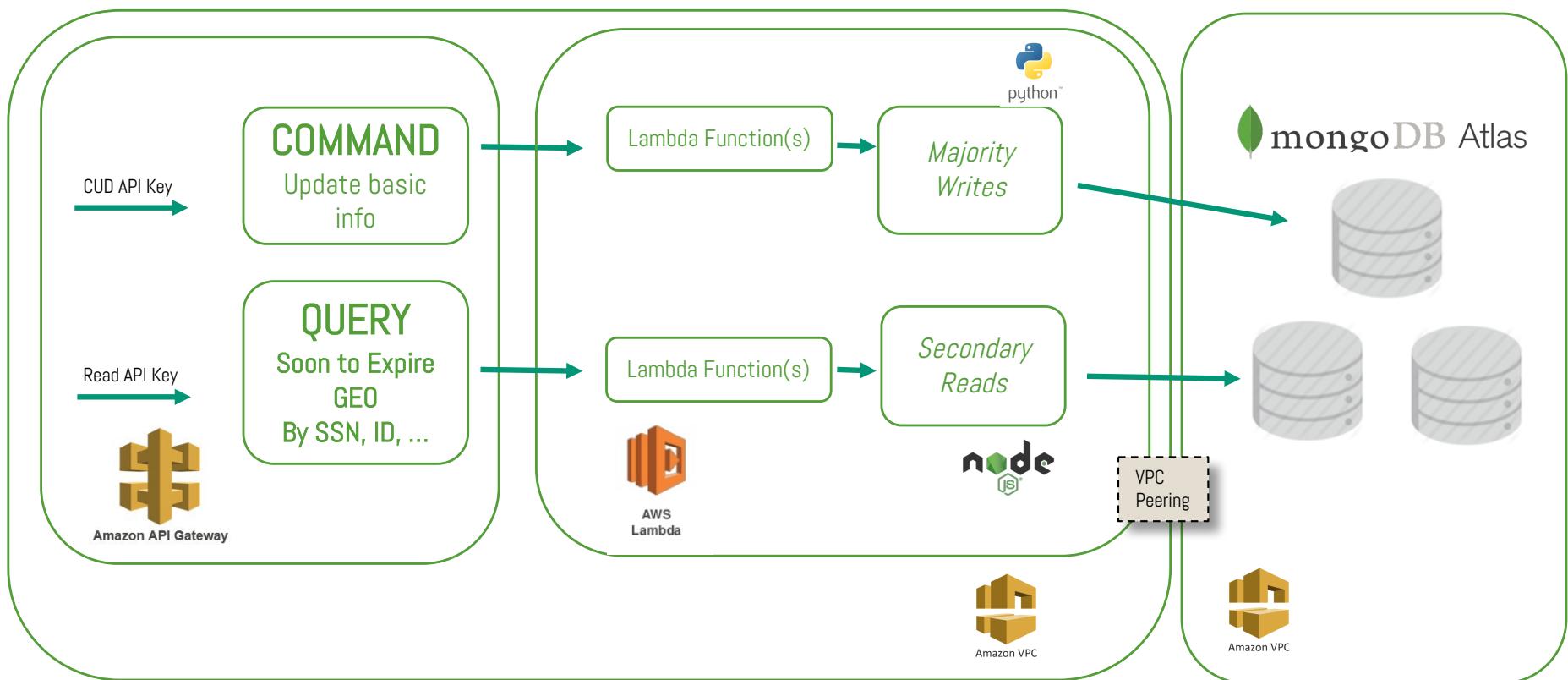
Shape

- Person
- Insurance Policies
 - Shape changes per policy type
- Addresses

Operations via API

- GET Customers with soon-to-expire policies, within a geo radius
- GET Customers / by SSN, id, etc.
- PATCH Update basic contact info (cell, email, ...)

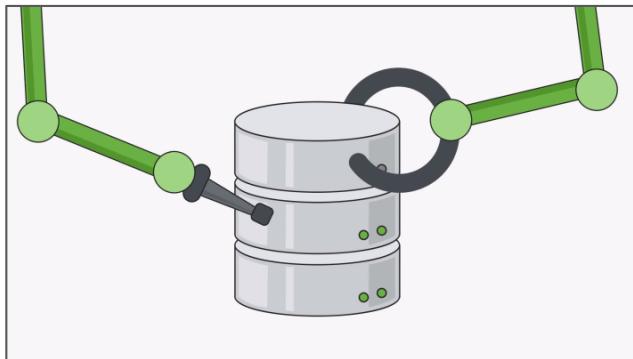
CQRS on Serverless Microservices



MongoDB Atlas & AWS

Build it!

MongoDB Atlas



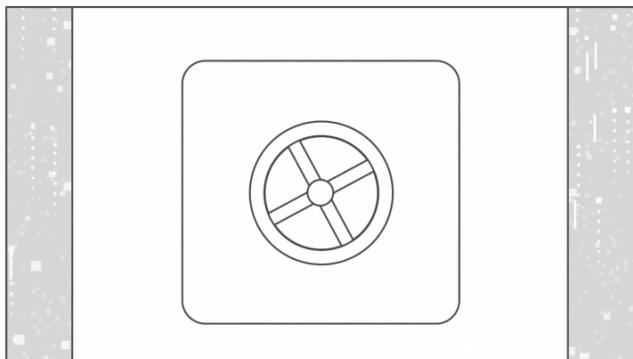
Automated Service



On-demand DBaaS



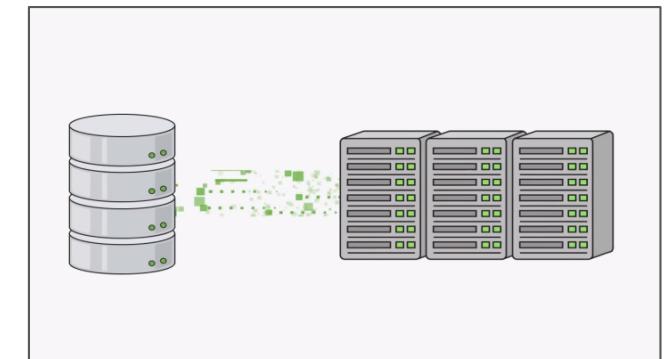
Elastic Scalability



Secure



Highly Available



Disaster Recovery

Setup MongoDB Atlas - M10+ *Need assigned AWS region*

The screenshot shows the MongoDB Atlas web interface. At the top, there's a navigation bar with a back arrow, forward arrow, refresh button, and a URL field showing <https://cloud.mongodb.com/v2/579785aae4b0a40a0c2465ec#clusters>. Below the bar, the title "Clusters" is displayed, along with a "Build a New Cluster" button. On the left, a sidebar lists various sections: GROUP (with "SigfridoAtMongoDB!"), Clusters (selected), Stitch Apps, Alerts (1), Backup, Users, Settings, Docs, and Support. The main content area shows a cluster named "BDDLA17Customers". It includes details like Version 3.4.6, BACKUPS (Inactive), INSTANCE SIZE (M10), REGION (AWS / Oregon (us-west-2)), TYPE (Replica Set - 3 nodes), and LINKED APP (None Linked - [Link Application](#)). To the right of these details are four performance metrics: Operations (R: 2.0 W: 0, Last 6 Hours), Disk Usage (4.3 GB, Last 30 Days), Connections (22, Last 6 Hours), and Disk IOPS (0.8, Last 6 Hours). At the bottom of the page, there's a footer with links for System Status, Last Login, and various MongoDB services.

System Status: All Good Last Login: 68.4.234.46
©2017 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

Generate dataset (optional)

mgeneratejs

- <https://github.com/rueckstiess/mgeneratejs>
- npm install -g mgeneratejs
- Create template – generate data
- Upload to Atlas via mongoimport
 - Hint: get connection string from Atlas UI!
- Browse with Compass

```
mgeneratejs -n 100 CustomerSingleView.json | mongoimport --host "YOUR  
ATLAS CLUSTER" --numInsertionWorkers 4  
--db SingleView --collection Customers --authenticationDatabase admin --  
ssl --username YOURUSER --password YOURPASSWORD
```

Template (CustomerSingleView.json)

```
{  
  "firstname": "$first",  
  "lastname": "$last",  
  "cell": { "$number": { "min": 1111111111, "max": 9999999999 }},  
  "email": "$email",  
  "yob": { "$date": { "min": "1930-01-01", "max": "2016-12-31" }},  
  "gender": "$gender",  
  "address": {  
    "number": { "$number": { "min": 1, "max": 9999 }},  
    "street": { "$street": { "country": "us" }},  
    "city": "$city",  
    "state": { "$state": { "country": "us" }},  
    "zip": "$zip",  
    "location": { "$point": { "long_lim": [-118.668469, -82.062023], "lat_lim": [
```

Required AWS Services

IAM

- Role with Lambda execute policies

VPC

- VPC Peering Connection
- Security Group

Lambda

- Set VPC, Security Group and IAM role
- Upload deployment package (.zip)

API Gateway

- API definition (Resources & HTTP Methods)
- Map Routes to Lambda functions
- API Keys & Usage Plans

VPC Peering

VPC Peering

USERS | IP WHITELIST | PEERING

+ NEW PEERING CONNECTION ATLAS CIDR BLOCK: 192.168.248.0/21

VPC ID	Status	Peering ID	VPC CIDR	Action
vpc-854200e3	Available	pcx-33c3595a	172.31.0.0/16	<button>TERMINATE</button>



USERS | IP WHITELIST | PEERING

+ ADD IP ADDRESS

You will only be able to connect to your cluster from the following list of IP Addresses:

IP Address/Security Group	Comment	Status	Actions
[REDACTED] 32 (includes your current IP address)	Home	Active	<button>EDIT</button> <button>DELETE</button>
172.31.0.0/16		Active	<button>EDIT</button> <button>DELETE</button>

VPC Peering

USERS | IP WHITELIST | PEERING

+ NEW PEERING CONNECTION

ATLAS CIDR BLOCK: 192.168.248.0/21

VPC ID	Status	Peering ID	VPC CIDR	Action
vpc-854200e3	Available	pcx-33c3595a	172.31.0.0/16	<button>TERMINATE</button>



USERS | IP WHITELIST | PEERING

+ ADD IP ADDRESS

You will only be able to connect to your cluster from the following list of IP Addresses:

IP Address/Security Group	Comment	Status	Actions
[REDACTED] 32 (includes your current IP address)	Home	Active	<button>EDIT</button> <button>DELETE</button>
172.31.0.0/16		Active	<button>EDIT</button> <button>DELETE</button>

vpc-854200e3

Summary Flow Logs Tags

VPC ID: vpc-854200e3 | Default VPC
State: available
IPv4 CIDR: 172.31.0.0/16
IPv6 CIDR:
DHCP options set: dopt-e682cb81
Route table: rtb-b336a1ca

rtb-b336a1ca

Summary Routes Subnet Associations

Edit

View: All rules

Destination	Target
172.31.0.0/16	local
0.0.0.0/0	igw-25717742
192.168.248.0/21	pcx-33c3595a

VPC Peering

ATLAS CIDR BLOCK: 192.168.248.0/21

VPC ID	Status	Peering ID	VPC CIDR	Action
vpc-854200e3	Available	pcx-33c3595a	172.31.0.0/16	TERMINATE

Atlas

You will only be able to connect to your cluster from the following list of IP Addresses:

IP Address/Security Group	Comment	Status	Actions
[REDACTED] 32 (includes your current IP address)	Home	Active	<button>EDIT</button> <button>DELETE</button>
172.31.0.0/16		Active	<button>EDIT</button> <button>DELETE</button>

AWS

Name	VPC ID	State	IPv4 CIDR
Default VPC	vpc-854200e3	available	172.31.0.0/16

vpc-854200e3

Summary Flow Logs Tags

VPC ID: vpc-854200e3 | Default VPC
State: available
IPv4 CIDR: 172.31.0.0/16
IPv6 CIDR:
DHCP options set: dopt-e682cb81
Route table: rtb-b336a1ca

rtb-b336a1ca

Summary Routes Subnet Associations

Edit

View: All rules

Destination	Target
172.31.0.0/16	local
0.0.0.0/0	igw-25717742
192.168.248.0/21	pcx-33c3595a

VPC Peering

The diagram illustrates the configuration of a VPC peering connection between two environments: **Atlas** and **AWS**.

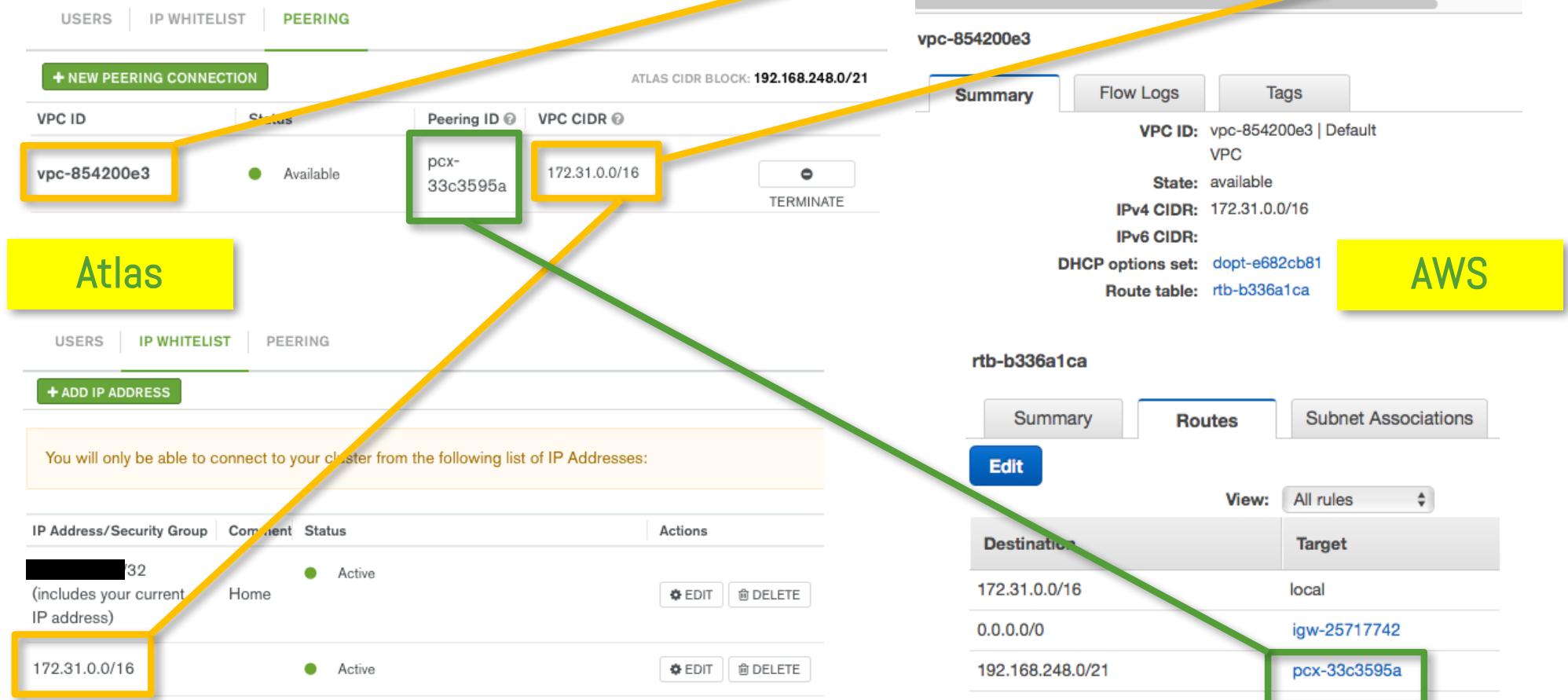
Atlas (Left):

- Peering:** A table shows a single peering connection with the following details:
 - VPC ID:** vpc-854200e3 (highlighted)
 - Status:** Available
 - Peering ID:** pcx-33c3595a
 - VPC CIDR:** 172.31.0.0/16 (highlighted)
 - ATLAS CIDR BLOCK:** 192.168.248.0/21
- IP Whitelist:** A table lists IP addresses and security groups:
 - IP Address/Security Group: 32 (includes your current IP address) - Home, Active
 - IP Address/Security Group: 172.31.0.0/16 - Active

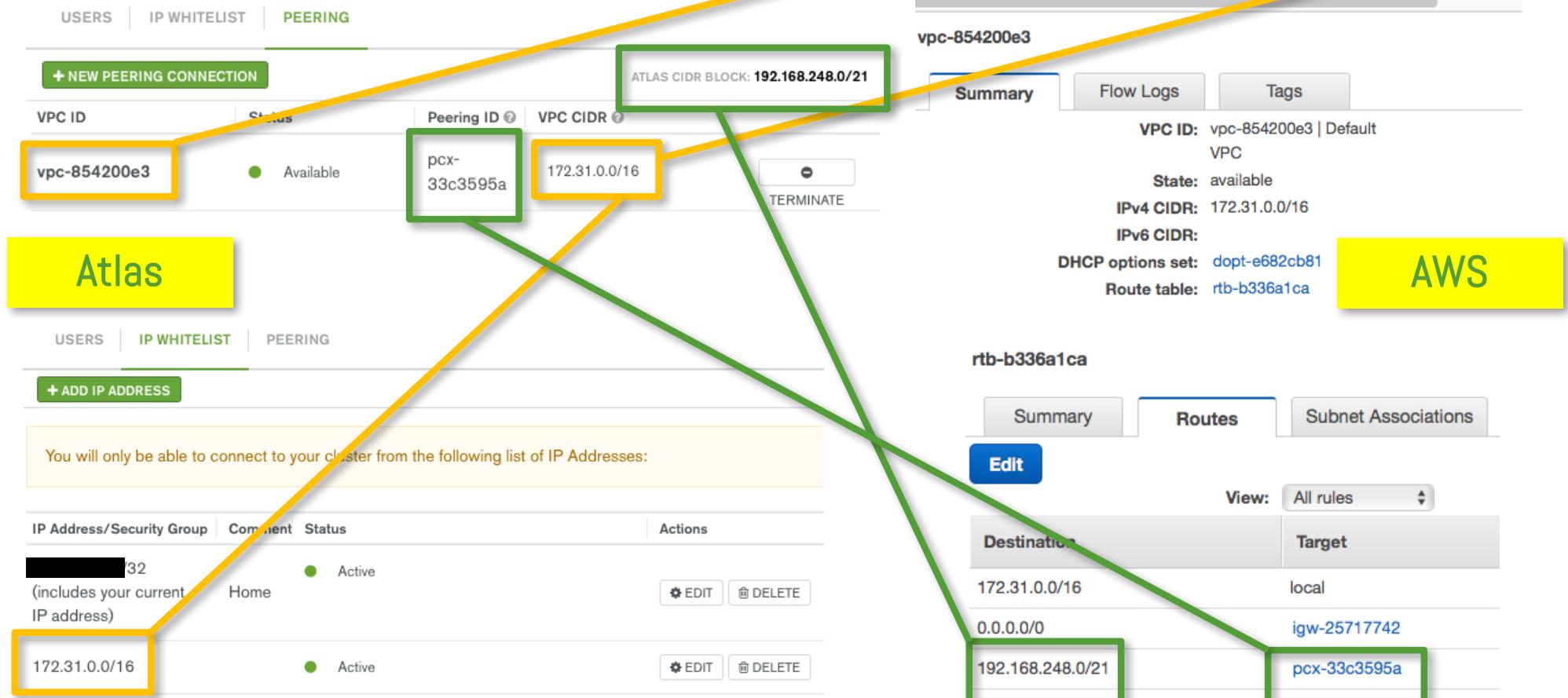
AWS (Right):

- VPC Summary:** A table shows the Default VPC with the following details:
 - Name:** Default VPC
 - VPC ID:** vpc-854200e3 (highlighted)
 - State:** available (highlighted)
 - IPv4 CIDR:** 172.31.0.0/16 (highlighted)
- Route Table:** rtb-b336a1ca (highlighted)
 - Summary:** Shows the VPC ID, State, IPv4 CIDR, and Route table.
 - Routes:** A table lists network routes:
 - Destination: 172.31.0.0/16, Target: local
 - Destination: 0.0.0.0/0, Target: igw-25717742
 - Destination: 192.168.248.0/21, Target: pcx-33c3595a

VPC Peering



VPC Peering



Lambda

Connections & Containers

... AWS Lambda maintains the container for some time in anticipation of another Lambda function invocation. ... the service freezes the container after a function completes, and thaws the container for reuse. If AWS Lambda chooses to reuse the container, this has the following implications:

- Any declarations in your Lambda function code (outside the handler code, see Programming Model) remains initialized, providing additional optimization when the function is invoked again. For example, if your Lambda function establishes a database connection, instead of reestablishing the connection, the original connection is used in subsequent invocations. You can add logic in your code to check if a connection already exists before creating one.

Python lambda function skeleton

```
from __future__ import print_function
import json
import pymongo

# Atlas connection outside of Lambda handler(s)
print('==> CONNECTING TO MONGODB ATLAS ==>') # logs to CloudWatch
atlasUri = os.environ['MONGODB_ATLAS_CLUSTER_URI']

MongoClient = pymongo.MongoClient(atlas_uri)
print("==> DONE - DB NAME: " + MongoClient.get_default_database().name + " ==>")

# =====
def lambda_handler(event, context):
# =====

    # get handle to collection
    coll = MongoClient.MyDB.MyColl
    ...
```

Node.js lambda function skeleton

```
'use strict';
var MongoClient = require('mongodb').MongoClient;
var assert = require('assert');
var ObjectId = require('mongodb').ObjectID;

// Atlas connection outside of Lambda handler
const atlas_uri = ">>> Atlas conn string here <<<"; // to-do read from AWS keys

let cachedDb = null;

exports.handler = (event, context, callback) => {
    // Set to false to allow re-use of database connections across lambda calls
    // see http://docs.aws.amazon.com/lambda/latest/dg/nodejs-prog-model-context.html
    context.callbackWaitsForEmptyEventLoop = false;

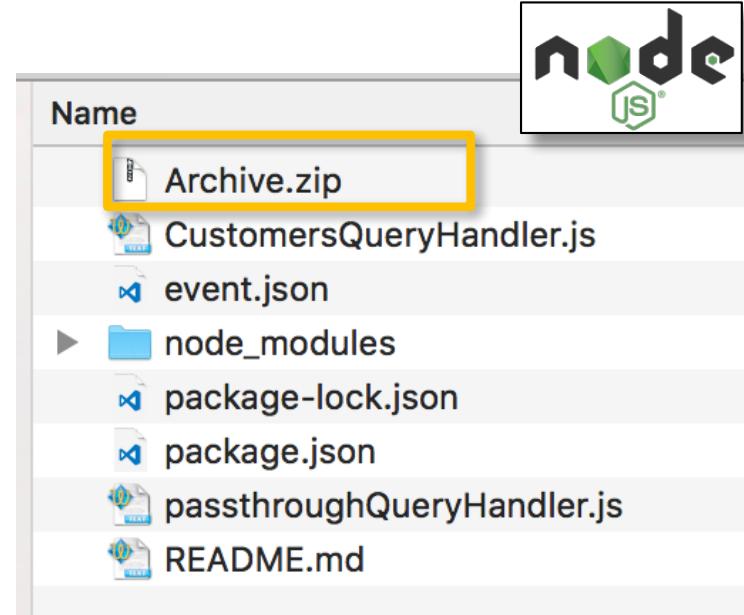
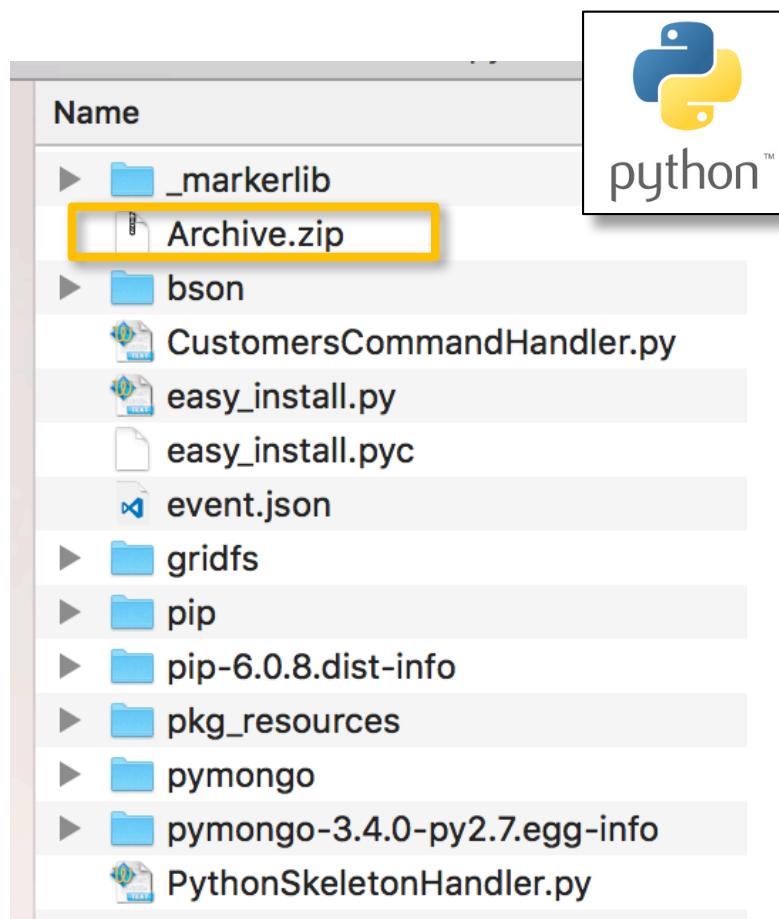
    try {
        if (cachedDb && cachedDb.serverConfig.isConnected()) {
            //Execute Query
            executeQuery(cachedDb, {}, callback);
        }
        else {
            console.log("== CONNECTING TO MONGODB ATLAS =="); // Log to CloudWatch
            MongoClient.connect(atlas_uri, function (err, db) {
                if (err) {
                    console.log(`the error is ${err}.`, err);
                    process.exit(1);
                }
                cachedDb = db;
            });
        }
    }
}
```

Local Emulators : Test on EC2 instance against Atlas

python-lambda-local at <https://pypi.python.org/pypi/python-lambda-local>

lambda-local (node.js) at <https://www.npmjs.com/package/lambda-local>

Code packaging



<http://docs.aws.amazon.com/lambda/latest/dg/deployment-package-v2.html>

Lambda functions

Lambda > Functions

The screenshot shows the AWS Lambda Functions dashboard. At the top, there are buttons for "Create a Lambda function" and "Actions". Below this is a search bar with filters for tags and attributes, and a keyword search field. The main area displays a table of Lambda functions with the following data:

	Function name	Description	Runtime	Code size
<input type="radio"/>	BDDLA17CustomersPATCH	PATCH handlers for update basic info	Python 2.7	3.8 MB
<input type="radio"/>	BDDLA17CustomersGET	GET handlers for feeds and single queries	Node.js 6.10	620.3 kB
<input type="radio"/>	BDDLA17CustomersRenewalsGET	GET handlers for GEO & Date queries	Node.js 6.10	620.3 kB

Upload & configure function

Lambda > Functions > BDDLA17CustomersGET ARN - arn:aws:lambda:us-west-2:853990252524:function:BDDLA17CustomersGET

Qualifiers ▾ Test Actions ▾

This function contains external libraries. Uploading a new file will override these libraries.

Code Configuration Triggers Tags Monitoring ?

Runtime Node.js 6.10 Handler CustomersQueryHandler.GET_hai The handler function

Role Choose an existing role The role with lambda permissions

Existing role MyLambdaRole

Description GET handlers for feeds and single

Advanced settings These settings allow you to control the code execution performance and costs for your Lambda function. Changing your resource settings (by selecting memory) or changing the timeout may impact your function cost. [Learn more](#) about how Lambda pricing works.

Memory (MB)* 128 MB

Timeout 1 min 0 sec

AWS Lambda will automatically retry failed executions for asynchronous invocations. You can additionally optionally configure Lambda to forward payloads that were not processed to a dead-letter queue (DLQ), such as an SQS queue or an SNS topic. Learn more about Lambda's [retry policy](#) and [DLQs](#). Please ensure your role has appropriate permissions to access the DLQ resource.

DLQ Resource Select resource

All AWS Lambda functions run securely inside a default system-managed VPC. However, you can optionally configure Lambda to access resources, such as databases, within your custom VPC. Learn more about accessing VPCs within Lambda. Please ensure your role has appropriate permissions to configure VPC.

VPC Default vpc-51cdb636 (172.31.0.0/16) | Default VPC The VPC (peered with Atlas)

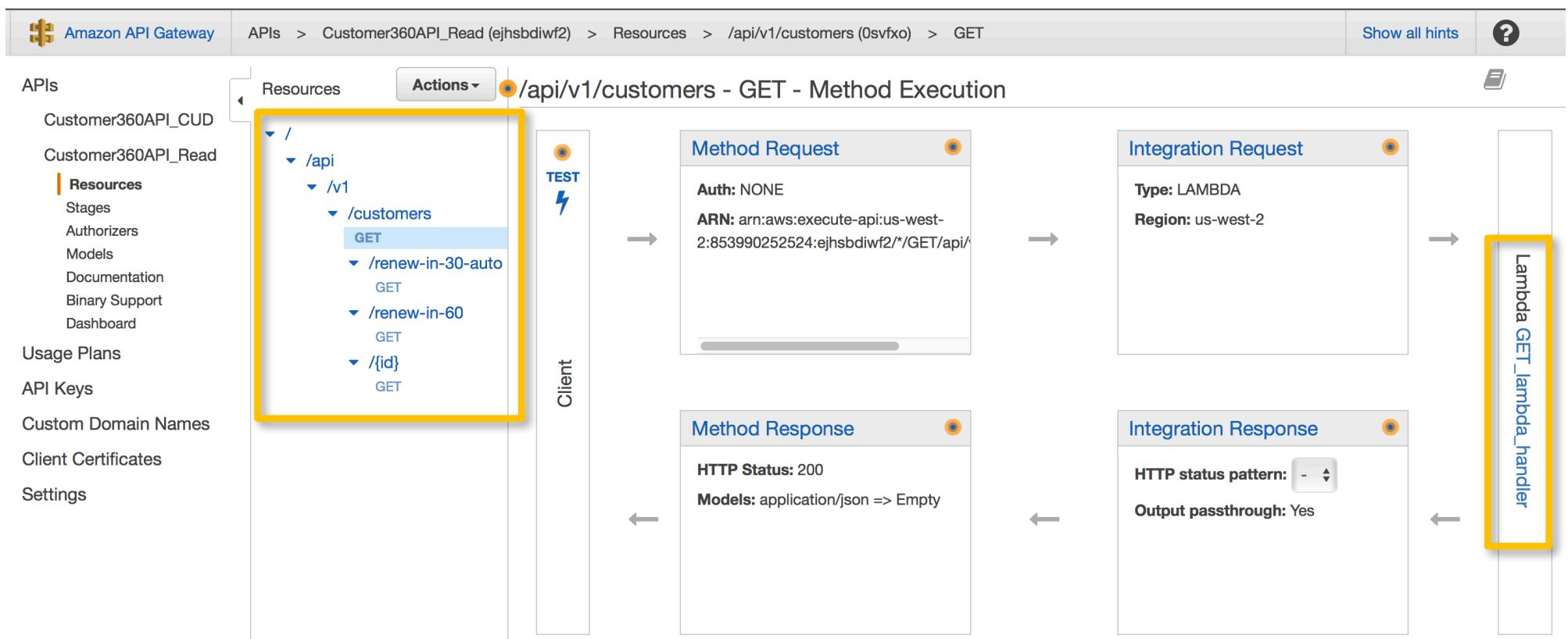
Subnets* subnet-ad3298... subnet-2435a54... subnet-8f222bd... At least 2 subnets

Security Groups* sg-b5c8c8cd (al...) The security group that allows traffic

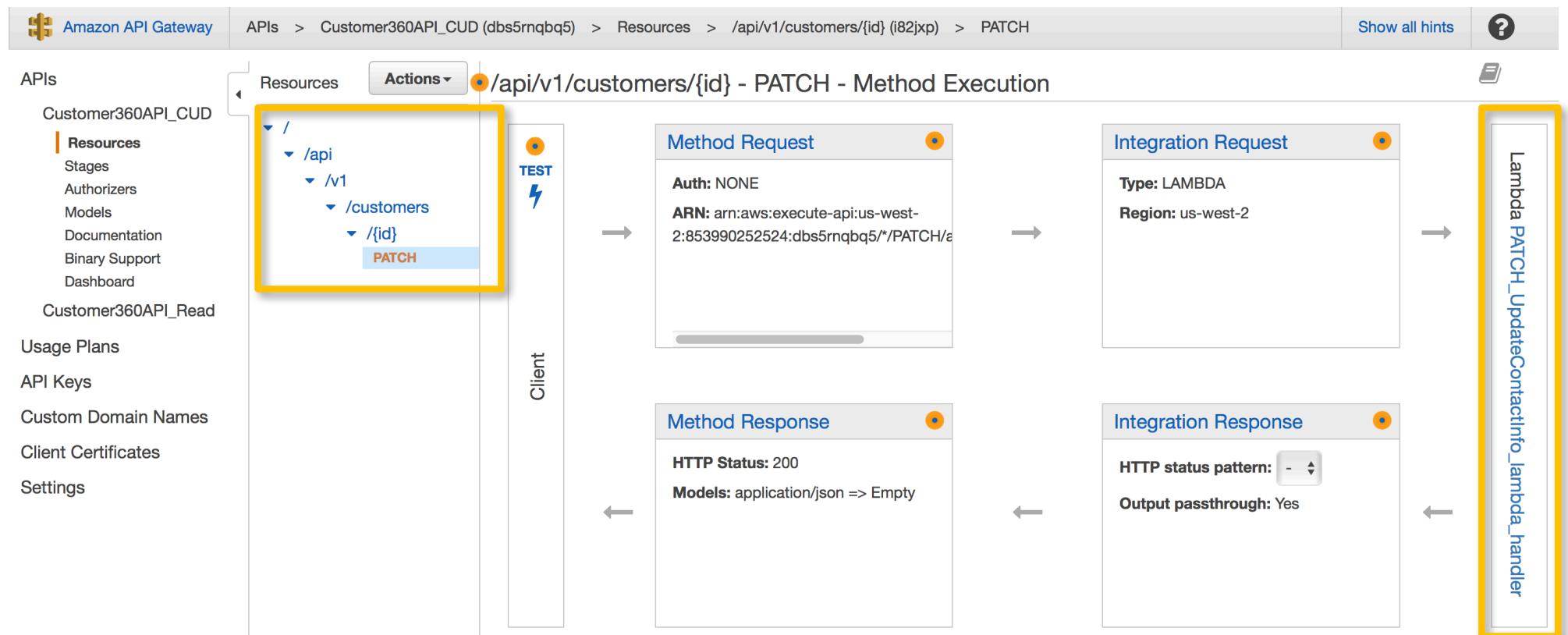
When you enable VPC, your Lambda function will lose default internet

API Gateway

Read API – GET /api/v1/customers



CUD API - PATCH /api/v1/customers



Deploying the API

The screenshot shows the Amazon API Gateway console with the following details:

- APIs:** Customer360API_CUD, Customer360API_Read
- Stages:** prod (selected), /
- Resources:** /api, /api/v1, /api/v1/customers (GET), /api/v1/customers/re (GET), /api/v1/customers/re (GET), /api/v1/customers/{id} (GET)
- Method:** GET - /api/v1/customers
- Invoke URL:** https://ejhsbdiwf2.execute-api.us-west-2.amazonaws.com/prod/api/v1/customers (highlighted with a yellow box)
- Settings:** Override for this method selected.
- CloudWatch Settings:** Enable CloudWatch Logs (unchecked), Enable Detailed CloudWatch Metrics (unchecked)
- Method Throttling:** Enabled (checked).
 - Rate: 1000 requests per second
 - Burst: 2000 requests

Access and throttling via API Keys

The screenshot shows the Amazon API Gateway interface for managing API keys. The left sidebar has a menu with 'Usage Plans' and 'API Keys' highlighted by a yellow box. The main content area shows a 'GET key' configuration. The key details are:

- ID:** pfp5uy7zi9
- Name:** GET key
- API key:** CbcZ9HTuET1VGPNIeool98GXICNpMI1dTWM7F610
- Description:** No description.
- Enabled:** Enabled

Below this, under 'Associated Usage Plans', there is a table:

Usage Plan	API	Stage	X
Read plan	Customer360API_Read	prod	X

Test!

Test with Postman

The screenshot shows the Postman application interface. The main window displays a successful API request for `GET /customers/id` with the URL `https://ejhsbdifw2.execute-api.us-west-2.amazonaws.com/prod/api/v1/customers/58cedd698957f9f0f2b37863`. The response status is `200 OK`, time is `333 ms`, and size is `2.3 KB`. The response body is a JSON object containing customer details.

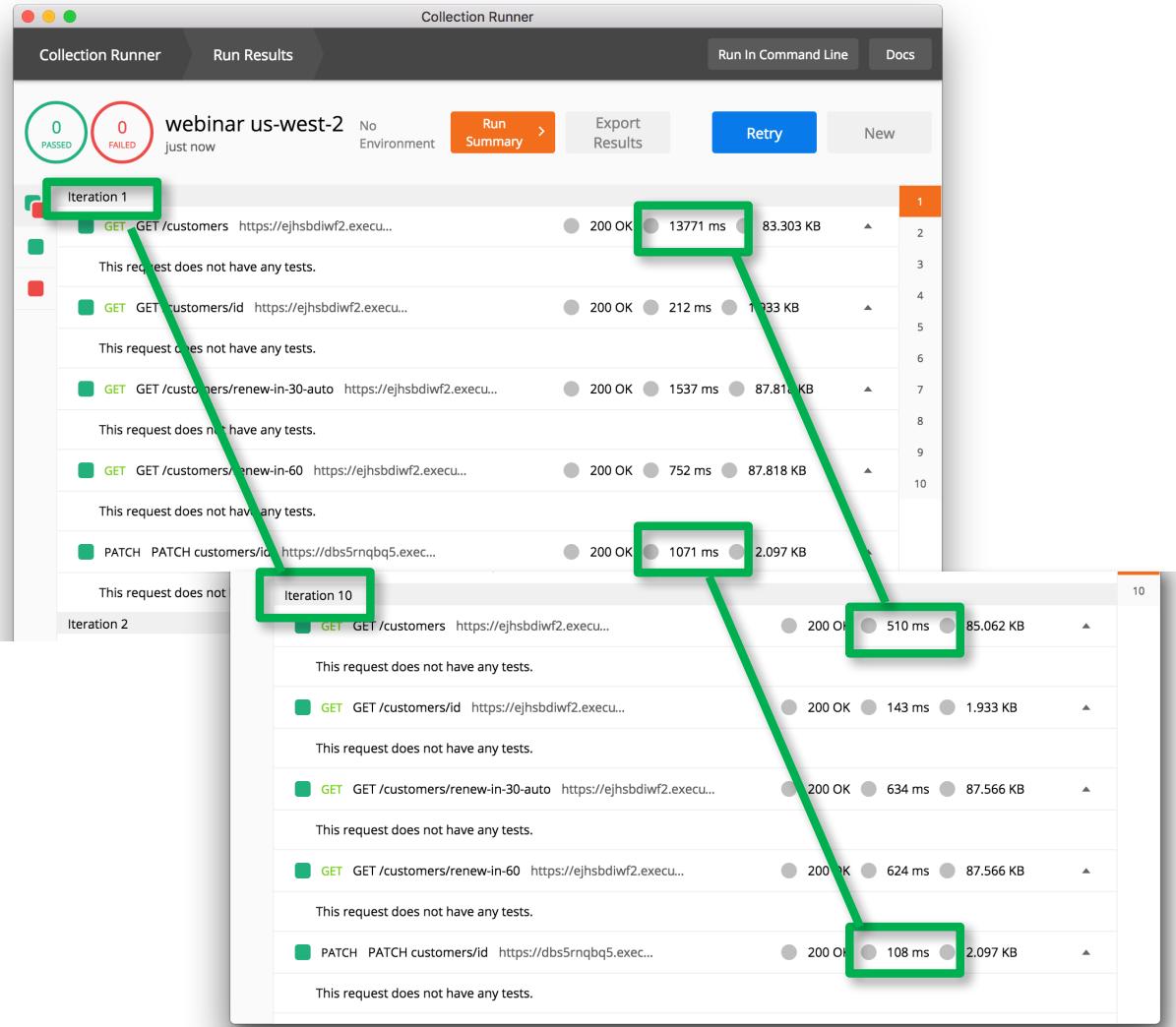
Left Sidebar: Shows a collection named "webinar us-west-2" which contains five requests: `GET /customers`, `GET /customers/id` (highlighted with a yellow box), `GET /customers/renew-in-30-auto`, `GET /customers/renew-in-60`, and `PATCH /customers/:id`.

Request Details:

- Method:** GET
- URL:** `https://ejhsbdifw2.execute-api.us-west-2.amazonaws.com/prod/api/v1/customers/58cedd698957f9f0f2b37863`
- Headers:** (1) `x-api-key` set to `CbcZ9HTuET1VGPNIleooI98GXICNpMI1dTWM7F610`
- Body:** (Raw) JSON response (shown below)

```
[{"firstname": "Michael", "lastname": "Rivera", "address": {"city": "Dujajkir", "zip": "82388", "number": 23, "state": "NE", "street": "Pilap Street", "location": {"type": "Point", "coordinates": [-108.9023, 39.9752]}, "yob": "$date": -631276860142}, "cell": 2206229691, "policies": [{"nextRenewalDt": {"$date": 1490241425151}, "insured_person": {"lastname": "Ruiz", "yob": "$date": -1257835299071}, {"firstname": "Olive", "smoking": false}, "policyType": "life", "policyNum": "U&tN8WF8KdTGNuZuhvlt*MdUECjD"}, {"address": {"city": "Heotale", "zip": "13008", "number": 9768, "state": "MO", "street": "Lecaw Way", "location": {"type": "Point", "coordinates": [94.88439, 40.46987]}, "value": 5872779, "policyType": "home", "year": 2002, "nextRenewalDt": {"$date": 1484064552076}, "policyNum": "JbgY2s4HxzJ(nfJHYzd5A76yc@t"}, {"nextRenewalDt": {"$date": 1485999675800}, "insured_person": {"lastname": "Wise", "yob": {"$date": 132707825345}, "firstname": "Celia", "smoking": true}, "policyType": "life", "policyNum": "3H#xXkbf5zVSecIVjH(TMxApGI%"), {"address": {"city": "Bakupaci", "zip": "61035", "number": 8354, "state": "IA", "street": "Uziumo Square", "location": {"type": "Point", "coordinates": [-108.02976, 41.55441]}, "value": 9060751, "policyType": "home", "year": 1978, "nextRenewalDt": {"$date": 1492011740291}, "policyNum": "NEGILz3rYFEpR02E4t!mh[n72KFpr"], {"nextRenewalDt": {"$date": 1485358013824}, "insured_person": {"lastname": "Little", "yob": {"$date": -175927072774}, "firstname": "Curtis", "smoking": true}, "policyType": "life", "policyNum": "2SjkbaNz058")/yWI/GaurcwBT09]^([]], "gender": "Female", "id": {"$oid": "58cedd698957f9f0f2b37863"}, "email": "mos@averoh.9g"}]
```

Load test too!



AWS CloudWatch

CloudWatch > Log Groups > /aws/lambda/GET_lambda_handler > 2017/03/21/[&LATEST]f5fc39dc71f647ea9b3d9f41753fe4ce

Filter events		<input checked="" type="radio"/> Row <input type="radio"/> Text			
Time (UTC +00:00)	Message	all 30s 5m 1h 6h 1d 1w custom ▾			
2017-03-21					No older events found at the moment. Retry.
▶ 00:13:35	Loading function				
▶ 00:13:35	==== CONNECTING TO MONGODB ATLAS ====				
▶ 00:13:35	==== DONE - DB NAME: WebinarCustomerSingleView ====				
▶ 00:13:35	START RequestId: 31fd10bf-0dcb-11e7-928e-518cbd547b30 version: \$LATEST				
▶ 00:13:35	filter = {}				
▶ 00:13:36	END RequestId: 31fd10bf-0dcb-11e7-928e-518cbd547b30				
▶ 00:13:36	REPORT RequestId: 31fd10bf-0dcb-11e7-928e-518cbd547b30 Duration: 735.96 ms Billed Duration: 800 ms Memory Size: 128 MB Max Memory Used: 26				
▶ 00:13:36	START RequestId: 3a2a91f8-0dcb-11e7-9ccc-4ffa9bba1c06 Version: \$LATEST				
▶ 00:13:36	filter = {"_id": {"\$oid": "58cedd698957f9f0f2b37863"}}				
▶ 00:13:36	END RequestId: 3a2a91f8-0dcb-11e7-9ccc-4ffa9bba1c06				
▶ 00:13:36	REPORT RequestId: 3a2a91f8-0dcb-11e7-9ccc-4ffa9bba1c06 Duration: 3.98 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 26 M				
▶ 00:13:40	START RequestId: 3c59ce72-0dcb-11e7-9592-85950a4fcfc2 Version: \$LATEST				
▶ 00:13:40	filter = {}				
▶ 00:13:40	END RequestId: 3c59ce72-0dcb-11e7-9592-85950a4fcfc2				
▶ 00:13:40	REPORT RequestId: 3c59ce72-0dcb-11e7-9592-85950a4fcfc2 Duration: 511.89 ms Billed Duration: 600 ms Memory Size: 128 MB Max Memory Used: 26				
▶ 00:13:40	START RequestId: 3ce17479-0dcb-11e7-8a1f-13c38df468a9 Version: \$LATEST				
▶ 00:13:40	filter = {"_id": {"\$oid": "58cedd698957f9f0f2b37863"}}				
▶ 00:13:40	END RequestId: 3ce17479-0dcb-11e7-8a1f-13c38df468a9				
▶ 00:13:40	REPORT RequestId: 3ce17479-0dcb-11e7-8a1f-13c38df468a9 Duration: 2.34 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 26 M				
▶ 00:13:42	START RequestId: 3dea58ec-0dcb-11e7-b370-afe0f068a543 Version: \$LATEST				
▶ 00:13:42	filter = {}				
▶ 00:13:43	END RequestId: 3dea58ec-0dcb-11e7-b370-afe0f068a543				
▶ 00:13:43	REPORT RequestId: 3dea58ec-0dcb-11e7-b370-afe0f068a543 Duration: 329.55 ms Billed Duration: 400 ms Memory Size: 128 MB Max Memory Used: 26				
▶ 00:13:43	START RequestId: 3e4571db-0dcb-11e7-8d1e-e79450dc4d9c Version: \$LATEST				
▶ 00:13:43	filter = {"_id": {"\$oid": "58cedd698957f9f0f2b37863"}}				

MongoDB Atlas Monitoring and Alerts

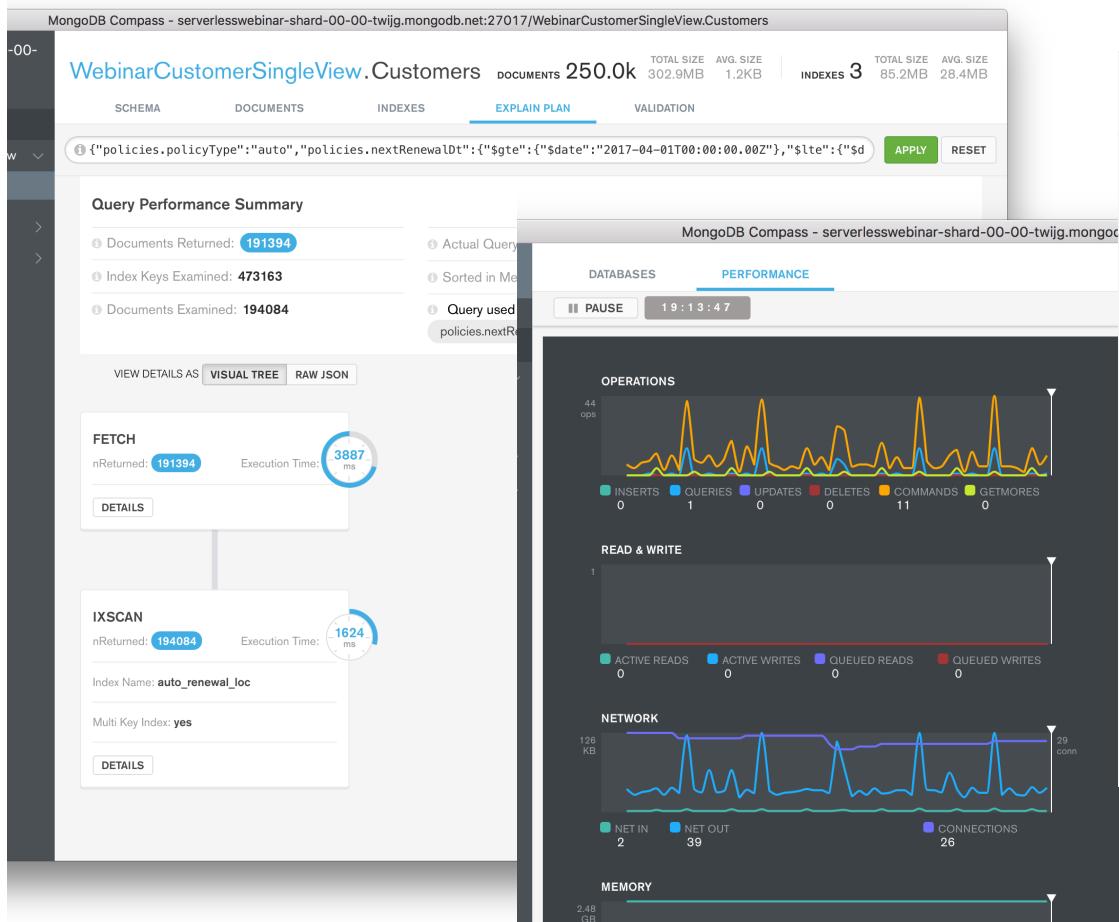
The screenshot shows the MongoDB Atlas Monitoring interface. On the left, there are two charts: 'Connections' and 'Opcounters'. The 'Connections' chart shows the number of connections over time, with a significant spike around 18:50. The 'Opcounters' chart shows various operation counts, with one series showing a peak around 18:50.

On the right, a modal window titled 'Create a New Alert' is open. It has three main steps:

- 1 Alert if**:
 - TARGET**: Host > of any type
 - HOST TYPE**: of type Standalone > of type Primary
 - CONDITION/METRIC**: Asserts: Warning is...
 - Average Execution Time: Commands is...
 - Average Execution Time: Reads is...
 - Average Execution Time: Writes is...
 - Background Flush Average is...
 - ABOVE**: 100 MSEC
- 2 For**: Any Host Hosts where...
- 3 Send to**:
 - Group**: All Roles
 - send if condition lasts at least 0 mins, resend after 60 mins
 - Email SMS PagerDuty
 - Atlas User**
 - Group owner**

SAVE

MongoDB Compass



WebinarCustomerSingleView.Customers

SCHEMA DOCUMENTS INDEXES EXPLAIN PLAN VALIDATION

Query returned 191394 documents. Displaying documents 1-40

INSERT DOCUMENT

```
{
  "policies.policyType": "auto",
  "policies.nextRenewalDt": {
    "$gte": "2017-04-01T00:00:00.002Z",
    "$lte": "2017-04-01T00:00:00.002Z"
  }
}
```

APPLY

DOCUMENTS

_id: ObjectId('58cedd698957f9f0f2b37856')
 firstname: "Arthur"
 lastname: "McLaughlin"
 cell: 3791578356
 email: "focasta@vusubi.bq"
 yob: 1975-10-25 19:09:55.109
 gender: "Male"
 address: Object
 number: 5161
 street: "Fuah Center"
 city: "Mahanzi"
 state: "NH"
 zip: "56334"
 location: Object
 policies: Array
 0: Object
 policyType: "home"
 policyNum: "Scg!(0L[vct6rvnUm]UWLV)\$Ry09iB"
 nextRenewalDt: 2017-03-11 20:12:38.226
 address: Object
 year: 1971
 value: 881753

Serverless Web
too!!

RetroPie – Old-school games, killer tiny hardware



Retro Web App based on aws-serverless-express

The screenshot shows a web browser window with a title bar reading "4ibap7jn0.execute-api.us-west-2.amazonaws.com". The main content is a table titled "SingleView Customers" with a subtitle "MongoDB + Express + Node.js gone Serverless!!". The table has columns for First Name, Last Name, Email, Total Policies, and Details. The "Details" column contains hyperlinks labeled "details". The data is a list of 50 customer entries, each with a unique first name and last name, a generated email address, a total number of policies (ranging from 3 to 6), and a "details" link.

First Name	Last Name	Email	Total Policies	Details
Adeline	Brewer	nohru@mid.by	6	details
Ann	Byrd	sai@hu.am	4	details
Delia	Torres	kokum@vob.nc	5	details
Mae	Kelley	ni@dog.bg	3	details
Isaac	Perkins	tihe@ejpac.cz	4	details
Willie	Price	puega@dog.tt	3	details
Roy	Lawson	ke@du.ci	4	details
Mike	Frank	palopoqo@izuri.mz	5	details
Andre	Powell	jura@co.nobebhum.gu	4	details
Allie	Marshall	hi@gup.bv	6	details
Andrew	Newton	ku@ehhu.uy	5	details
Lucinda	Weaver	hi@uvefi.qg	5	details
Beatrice	Lucas	guwpamlez@jaka.tr	4	details
Kyle	Reeves	to@on.ss	5	details
Jeffrey	Austin	dag@pzow.aq	3	details
Duane	Holland	radjariv@keefi.uk	5	details
Daisy	Hawkins	zo@ipacilz.co.uk	6	details
Emma	Mendoza	cifewu@fohekeupu.pa	4	details
Frances	Francis	vorozda@eganog.bi	6	details
Rhoda	Peters	ibzec@riagozo.tw	6	details
Lydia	Tucker	jawfilip@edwojos.it	5	details
Gordon	Burton	vejufi@hunal.sj	4	details
Ernest	Miles	so@eruon.gr	5	details
Nora	Thompson	luwufalin@sudjiwbo.br	5	details
Harvey	Williamson	figredfa@ew.ag	3	details
Polly	Moss	katu@mel.su	5	details
Alexander	Saunders	bo@ew.mz	5	details
Virginia	Hernandez	lukuf@uhomep.pe	6	details
Marcus	Watts	nover@ada.gn	3	details
Duane	Mitchell	rimmutcb@pof.ma	6	details
Vera	Garrett	oki@rog.tz	6	details
Lucas	Johnston	sogo@be.dz	5	details
Raymond	Hart	fuh@ta.mm	6	details
Belle	Fleming	itzusrov@ifacki.gi	5	details
Cody	Rogers	rattodgop@mjil.il	3	details
Dominic	Martinez	ebuline@dovlo.gf	6	details
Pearl	Morales	muc@nericka.sv	4	details
Dollie	Walsh	cohut@uppowo.az	3	details
Winfred	Hayes	vuj@afi.py	6	details
Edna	Watts	zo@hic.gv	4	details
Jay	Bates	fiuli@pefbu.bn	4	details
Cynthia	Maldonado	joiku@bo.fm	3	details
Pearl	Sherman	huzzbeh@co.nw	3	details

<https://github.com/snarvaez/bddla17/tree/master/singleViewWebApp>

<https://github.com/awslabs/aws-serverless-express>



Done!

...

But what about?

Scaling?

Scaling Lambda

No user intervention required - Default safety throttle of **100 concurrent executions per account per region.**

Functions invoked **synchronously** throw 429 error code.
Functions invoked **asynchronously** can absorb reasonable bursts for approx. 15-30 minutes. If exhausted, consider using Simple Queue Service (SQS) or Simple Notification Service (SNS) as the Dead Letter Queue (DLQ).

Read more at <https://aws.amazon.com/lambda/faqs/>

Scaling MongoDB Atlas

- On-Demand
Zero downtime
Upscale/Downscale:
- Instance size
 - Storage size
 - IOPS
 - Replication factor.

Replication Factor

Select how many copies of your data should exist in your cluster. You can easily change your deployment, with no down-time.



3 Nodes



5 Nodes



7 Nodes

Instance Size

Select the size of your servers. You can easily upgrade your servers at any time after deployment.

Great for development environments and low traffic websites.		
M0	included: shared RAM	512 MB storage
M10	included: 2 GB RAM	10 GB storage
M20	included: 4 GB RAM	20 GB storage
Perfect for production environments, supporting high-traffic applications or large data sets.		
M30	included: 8 GB RAM	40 GB storage
M40	included: 16 GB RAM	80 GB storage
	included: 32 GB RAM	160 GB storage



@SigNarvaez | @MongoDB | @BigDataDayLA



Serverless Architectures with AWS Lambda and MongoDB Atlas

Q & A



Sig Narváez
Sr. Solutions Architect
sig@mongodb.com
[@SigNarvaez](https://github.com/snarvaez/bddla17)

<https://github.com/snarvaez/bddla17>

[https://resources.mongodb.com/serverles
s-architectures](https://resources.mongodb.com/serverless-architectures)