Blow up the monolith?

Serverless computing, nanofunctions, & Amazon Lambda
Disclaimer

This is a concept and code that I'm still experimenting with. It is not live in production.
"Serverless Computing"

- How can you have your code executed without a computer?
- There is some hype here...
"Serverless Computing"

Not "there is no server"
But "I don't know anything about the server"
You pay someone else to set it up, configure it, patch it, etc.
Can't even tell what OS is running
All-in-one program
  • that's hard to scale,
  • hard to add functionality to due its complexity,
  • or both

Recurring pattern that tends to arise naturally
Scaling a monolith

In 2017, well-established approach is to break the monolith into microservices which can be scaled and enhanced independently, by different teams.

- have had very good experience with this
- still my first recommendation
Finding the places to separate the existing code into microservices can be difficult. Refactoring can help a lot by letting you identify the candidates for microservices and refine them into independent entities (http://stevecorwin.com/blog/2016/09/05/steps-towards-breaking-up-a-monolith/)
Separating can still be difficult

Sometimes no matter what you do you find yourself looking at something like this:

<table>
<thead>
<tr>
<th>ShiftServices</th>
</tr>
</thead>
<tbody>
<tr>
<td>fetchShiftById</td>
</tr>
<tr>
<td>fetchShiftResults</td>
</tr>
<tr>
<td>fetchOpenShifts</td>
</tr>
<tr>
<td>fetchPendingRequestedShifts</td>
</tr>
<tr>
<td>fetchUrgentShifts</td>
</tr>
<tr>
<td>fetchAwardedShiftsByUser</td>
</tr>
<tr>
<td>fetchAwardedShiftsByDate</td>
</tr>
<tr>
<td>fetchNotAwardedToBidderShifts</td>
</tr>
<tr>
<td>fetchAllAwardedShiftsByUser</td>
</tr>
<tr>
<td>fetchAllAwardedShiftsByAdmin</td>
</tr>
<tr>
<td>fetchExportableScheduledShifts</td>
</tr>
<tr>
<td>fetchScheduledShiftsByUser</td>
</tr>
<tr>
<td>fetchOnCallShifts</td>
</tr>
<tr>
<td>fetchScheduledShifts</td>
</tr>
<tr>
<td>fetchNonDutyShifts</td>
</tr>
</tbody>
</table>
GetOrders function is separate from the CreateOrder function, which is separate from the DeleteOrder function

One customer gets a high % of traffic on GetOrders, a different customer gets high % of traffic on CreateOrder

Can scale functions independently
The Vision

ShiftServices

fetchShiftById
fetchOpenShifts
fetchShiftResults
fetchUrgentShifts

hPendingRequestedShifts

fetchAwardedShiftsByUser
fetchAwardedShiftsByDate
fetchNotAwardedToBidderShifts

KABOOM
What do you call something that's broken into even smaller pieces than a microservice?

Independent of vendor

Microservice: Order service has GetOrders endpoint, CreateOrder endpoint, DeleteOrder endpoint

Separate, smaller functions → “nanofunction”
Why Amazon Lambda for this?

It was available
Amazon AWS is widely used and has a solid track record
Amazon Lambda was launched November 2014
You can start for free
Amazon Lambda

Supports Java, Python, Node.js (JavaScript), C#
Suggested for lots of different use cases
• Data Processing
• Real-time File Processing
• Real-time Stream Processing
• Extract, Transform, Load
• IoT Backends
• Mobile Backends
• Web Applications
"Hello world" Lamda

```java
public class Hello {
    public String myHandler(int myCount, Context context) {
        LambdaLogger logger = context.getLogger();
        logger.log("received : " + myCount);
        return String.valueOf(myCount);
    }
}
```

http://docs.aws.amazon.com/lambda/latest/dg/get-started-step4-optional.html
Realistic PoC

- Java
- Takes a couple of inputs
- Returns results based on a database query
- Then add a second nanofunction that stores data into the database
Shared Database - project structure

(project directory)
├── model (maps directly to tables)
├── core (ex: DAOs)
├── nanofunction (orchestrates core)
└── lambda (Amazon-specific)

Dependencies go up not down
Lambda as thin as possible
Multiple nanofunctions share core
Amazon technologies needed

- IAM (creating users and roles to use for accessing the AWS web UI and using the CLI)
- S3 (for storing Java code for Lambda)
- AWS CLI (for deploying and testing my nanofunction)
- RDS (for MySQL database)
- VPC (so Lambda can connect to database)
- Subnets & Security Groups
- EC2 (to have a server where I can run a MySQL client)
- Lambda
- Amazon API Gateway (for the RESTful endpoint)
Sign up for AWS

https://aws.amazon.com/

PoC all done using free resources

Set up billing alert just in case
IAM admin user

http://docs.aws.amazon.com/lambda/latest/dg/setting-up.html

Users: adminuser

<table>
<thead>
<tr>
<th>User ARN</th>
<th>arn:aws:iam::629829016304:user/adminuser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>/</td>
</tr>
<tr>
<td>Creation time</td>
<td>2017-01-24 17:46 PST</td>
</tr>
</tbody>
</table>

Number of attached policies 1

AdministratorAccess - AWS Managed policy from group FullAccessGroup

+ Add inline policy
IAM role for S3 storage

Because "AWS Lambda requires an Amazon S3 bucket to store your Java project when you upload it."

IAM -> Role -> Create Role

1. In Role Name, use a name that is unique within your AWS account (for example, steve-s3-full-access-role).

2. In Select Role Type, choose AWS Service Roles, and then choose AWS Lambda. This grants the AWS Lambda service permissions to assume the role.

3. In Attach Policy, choose AmazonS3FullAccess.
IAM role for VPC access

VPC = virtual private cloud, a virtual network dedicated to your AWS account. Contains AWS resources such as RDS instances.

IAM -> Role -> Create New Role

1. In Role Name, use a name that is unique within your AWS account (for example, steve-lambda-vpc-access-role).
2. In Select Role Type, choose AWS Service Roles, and then choose AWS Lambda. This grants the AWS Lambda service permissions to assume the role.
3. In Attach Policy, choose AWSLambdaVPCAccessExecutionRole.
Create an S3 bucket

AWS regions:
http://docs.aws.amazon.com/general/latest/gr/rand e.html#lambda_region

Put everything in the same region for performance and simplicity
In this presentation everything is in us-west-2 (Oregon)
Install AWS CLI

Much more concise than the web UI
Easier to document
Easier to repeat

http://docs.aws.amazon.com/cli/latest/userguide/installing.html

$ pip install --upgrade --user awscli

$ aws help
$ aws lambda help
Configure AWS CLI

http://docs.aws.amazon.com/lambda/latest/dg/setup-awscli.html

$ aws configure --profile adminuser
    AWS Access Key ID [None]: AKIAJG2JLCQGQCLLW1234
    AWS Secret Access Key [None]: 3B/61xAOpjTvLjCgKuqu78NWfbKJJI/6Y123456
    Default region name [None]: us-west-2
    Default output format [None]: json
“A security group acts as a virtual firewall that controls the traffic for one or more instances. When you launch an instance, you associate one or more security groups with the instance. You add rules to each security group that allow traffic to or from its associated instances.”

Much easier if everything is in the same security group
AWS Subnet

Subnet = subset of the IP address range to a VPC. Some instances get more than one subnet, some only one.

Development is easier if everything is in the same subnet.

Different subnets are possible, outside scope of this presentation.
$ aws rds create-db-instance --db-instance-identifier SteveMySQLForLambdaTest --db-instance-class db.t2.micro --engine MySQL --allocated-storage 5 --no-publicly-accessible --db-name SteveTestDB --master-username steve --master-user-password supersecret --backup-retention-period 3 --profile adminuser
{ "DBInstance": {   "VpcSecurityGroups": [     { "Status": "active", "VpcSecurityGroupId": "sg-8cc7c812" }   ],   ...   "DBSubnetGroup": { "Subnets": [     { "SubnetStatus": "Active", "SubnetIdentifier": "subnet-19617512", "SubnetAvailabilityZone": { "Name": "us-west-2a" }     }, ...   ] }   ...  } }
EC2 (for MySQL client)

Need to choose suitable image; I used "Ubuntu Server 16.04 LTS (HVM), SSD Volume Type"
For size I chose t2.micro "Free tier eligible"
Use VPC, subnet, and security group to match RDS for MySQL
EC2 – allow ssh

Security group sg-8cc7c812
Inbound rule: allow ssh (port 22) from 0.0.0.0/0

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>sg-8cc7c8f5 (default)</td>
</tr>
<tr>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>sg-a39383da (launch-wizard-1)</td>
</tr>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>0.0.0.0/0</td>
</tr>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>::/0</td>
</tr>
</tbody>
</table>
EC2 – install MySQL client

$ sudo apt-get update
$ sudo apt-get install mysql-client-core-5.7

$ mysql -v
   --host=stevemysqlforlambdatest.clmy82gfefkm.us-west-2.rds.amazonaws.com --user=steve
   --password=supersecret SteveTestDB

Create tables as needed
Goal: nanofunctions that connect to a database

Code we need:
- request handler (called for every request)
- Amazon-specific code
- code to connect to the database
@Override
public String handleRequest(HelloInput helloInput, Context context) {

    return "Hello " + helloInput.getFirstName()
    + ", shift data is " +
    fetchShiftsNanofunction.fetchAllShiftsAsString();
}
package com.apihealthcare.lamba;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.apihealthcare.dao.ShiftDAO;
import com.apihealthcare.lambda.HelloInput;

public class FetchShiftsLambda implements RequestHandler<HelloInput, String> {

    @Override
    public String handleRequest(HelloInput helloInput, Context context) {

    }
}
private static FetchShiftsNanofunction fetchShiftsNanofunction;

public FetchShiftsLambda() throws SQLException
{
    fetchShiftsNanofunction = new FetchShiftsNanofunction();
    LOGGER.debug("FetchShiftsLambda constructor done");
}
Database connections

As of 3/20/17, Amazon Lambda does not support connection pooling

Other languages: creating the equivalent of a static member variable

```java
// JDBC_URL = "jdbc:mysql://stevemysqlforlambdatest.clmy82gfefkm.us-west-2.rds.amazonaws.com:3306/SteveTestDB";

public FetchShiftsNanofunction() throws SQLException {
    shiftDao = new ShiftDAO( JDBC_URL, USERNAME, PASSWORD );
    LOGGER.debug( "FetchShiftsNanofunction constructor done" );
}
```
Database connections - continued

Write code that checks for a valid DB connection and creates one if needed
One connection per Lambda instance

In my testing, only lost connection after rebooting RDS MySQL
One request every 5 minutes sufficient

Constructor log output not written anywhere I can find
Build with Maven

Starter pom:
- aws-lambda-java-core (required)
- aws-lambda-java-log4j (optional)

$ mvn clean package → target/lamba-java-example-1.0-SNAPSHOT.jar
Uploading the jar – first time

```
$ aws lambda create-function --function-name FetchShiftsLambda --zip-file fileb://home/steve/swdev/aws/HelloWorldJava/target/lambda-java-example-1.0-SNAPSHOT.jar --role arn:aws:iam::629829011234:role/steve-lambda-vpc-access-role --handler com.apihealthcare.lambda.FetchShiftsLambda::handleRequest --runtime java8 --vpc-config 'SubnetIds=subnet-19617512,SecurityGroupId=sg-8cc7c812' --timeout 60 --region us-west-2 --profile adminuser
```
--function-name: used to identify Lambda
--zip-file: “fileb://” + path to your jar
--role: full arn of role that allows the Lambda VPC access
--handler: full package and class name + “::” + name of function that Lambda framework should call
--vpc-config 'SubnetIds=subnet-19617512,SecurityGroupIds=sg-8cc7c812' → use the subnet Id and security group from RDS instance
--timeout: needs to allow time for DB connection
Uploading the jar – second time

$ aws lambda update-function-code --function-name FetchShiftsLambda --zip-file fileb:///home/steve/swdev/aws/HelloWorldJava/target/lambda-java-example-1.0-SNAPSHOT.jar --region us-west-2 --profile adminuser
Testing the jar – Web UI

"Test" button → supply JSON for input
Output: response, log output, time/resources consumed
Testing the jar – AWS CLI

$ aws lambda invoke --function-name FetchShiftsLambda --log-type Tail --payload '{"firstName": "lambda-invoke", "lastName": "Command Line"}' --region us-west-2 --profile adminuser lambdaOutput.txt

lambdaOutput.txt has response

LogResult is the base64-encoded logs for the Lambda function invocation.

$ base64 --decode <<< {contents of double quotes}
Demo


Sample test event:
{
  "firstName":"Gipsy",
  "lastName":"Kings"
}
RESTful API

GET
POST
etc.

curl https://9k196bl123.execute-api.us-west-2.amazonaws.com/prod/ShiftMgr2?
firstName=GetViaCurl&lastName=Calling
Amazon API Gateway

Capable of creating RESTful endpoints for many things, including Lambda functions
Lots of flexibility
Not so simple...

Use the same region as everything else, at least in development
Create REST API

Create the API
Get the rest-api-id

$ aws apigateway create-rest-api --name ShiftOps2
   --region us-west-2 --profile adminuser

{
   "name": "ShiftOps2",
   "id": "9k196bl123",
   "createdDate": 1488577949
}

9k196bl123 is your rest-api-id
Get the Root Resource ID

Creation doesn't return the new root resource ID

$ aws apigateway get-resources --rest-api-id 9k196bl1k1 --region us-west-2 --profile adminuser

```
{
    "items": [
        {
            "path": "/",
            "id": "j9vz8sl123"
        }
    ]
}
```

j9vz8sl123 = root-id = parent-id
Create a Resource in the API

Resources can have GET, POST, etc. methods
path-part becomes part of the URL

$ aws apigateway create-resource --rest-api-id 9k196bl123 --parent-id j9vz8sl123 --path-part ShiftMgr2 --region us-west-2 --profile adminuser

```json
{
    "path": "/ShiftMgr2", "pathPart": "ShiftMgr2",
    "id": "56a212", "parentId": "j9vz8sl123"
}
```

56a212 = resource-id
API Resource flow

/ShiftManager - POST - Method Execution

**Method Request**
- **Auth**: NONE
- **ARN**: arn:aws:execute-api:us-west-2:629829016304:530vpyyxz1/*
  /POST/ShiftManager

**Integration Request**
- **Type**: LAMBDA
- **Region**: us-west-2

**Method Response**
- **HTTP Status**: 200
- **Models**: application/json => Empty

**Integration Response**
- **HTTP status pattern**: --
- **Output passthrough**: Yes
Implement POST first

For each HTTP method there is a Method Request and an Integration Request
Method Request may be HTTP GET or HTTP POST
Integration Request for a Lambda is always a POST, even when Method Request is a GET
Tutorials don't always make it clear which one they're referring to
Create a POST method on the resource

```
$ aws apigateway put-method --rest-api-id 9k196bl123
    --resource-id 56a212 --http-method POST
    --authorization-type NONE --region us-west-2
    --profile adminuser
```
Create the POST destination

Set the FetchShiftsLambda as the destination for the POST

```
```
Request and Response Mappings

For each HTTP method
- Method Request
- Integration Request
- Integration Response
- Method Response

Mappings can do a variety of transformations

JSON in, JSON out → leave at default (passthrough, empty)
API Resources can be deployed to stages, which you name.

Typical stage names are “test” and “prod”.
Deploy API to “prod” Stage

$ aws apigateway create-deployment --rest-api-id 9k196bl123 --stage-name prod --region us-west-2 --profile adminuser

The “prod” stage was automatically created since it didn't exist
Grant permission to invoke

Grant the Amazon API Gateway service principal (apigateway.amazonaws.com) permissions to invoke FetchShiftsNanofunction


statement-id value must be unique
Grant permission to invoke – part 2

Grant to your deployed API (ShiftMgr2) permissions to invoke FetchShiftsLambda


statement-id value must be unique
This is for the “prod” stage
Test with Web UI

/ShiftManager - POST - Method Test

Make a test call to your method with the provided input

Path

No path parameters exist for this resource. You can define path parameters by using the syntax `{myPathParam}` in a resource path.

Query Strings

No query string parameters exist for this method. You can add them via Method Request.

Headers

No header parameters exist for this method. You can add them via Method Request.

Stage Variables

No stage variables exist for this method.

Request Body

```
1   { "firstName":"AWS Web UI", "lastName":"Calling"}
```
Test with AWS CLI

$ aws apigateway test-involve-method --rest-api-id 9k196bl123 --resource-id 56a212 --http-method POST --path-with-query-string "" --body "{ "firstName": "AWS CLI", "lastName": "Calling"}" --region us-west-2 --profile adminuser

Response is large
- "status": 200
- body
- Endpoint request body after transformations
- Endpoint response body before transformations
$ curl -X POST -d "\{"firstName":"Curl","lastName":"Calling\}" https://9k196bl123.execute-api.us-west-2.amazonaws.com/prod/ShiftMgr2

Web browser should work now too
Implement GET

Can be same Lambda as the POST
Query parameters will require input mapping template
Create a GET method on the resource

```bash
$ aws apigateway put-method --rest-api-id 9k196bl123 --resource-id 56a212 --http-method GET --authorization-type NONE --region us-west-2 --profile adminuser
```
Create the GET destination

Set the FetchShiftsLambda as the destination for the POST

Deploy API to “prod” Stage

$ aws apigateway create-deployment --rest-api-id 9k196bl123 --stage-name prod --region us-west-2 --profile adminuser

Uses the “prod” stage that already exists
Grant permission to invoke

Grant the Amazon API Gateway service principal (apigateway.amazonaws.com) permissions to invoke FetchShiftsLambda


statement-id value must be unique

This is for the “prod” stage
Request and Response Mappings

For each HTTP method
- Method Request
- Integration Request
- Integration Response
- Method Response

For GET, Integration Request can map query parameters to Lambda inputs

JSON out → leave at default (passthrough, empty)
GET Resource Flow

/ShiftMgr2 - GET - Method Execution

**Method Request**

**Auth:** NONE  
**ARN:** arn:aws:execute-api:us-west-2:629820016304:9k196bfllk1*/GET/ShiftMgr2

**Integration Request**

**Type:** LAMBDA  
**Region:** us-west-2

**Method Response**

**HTTP Status:** 200  
**Models:** application/json => Empty

**Integration Response**

**HTTP status pattern:**  
**Output passthrough:** No
Integration Request Body Mapping

Body Mapping Templates

- Request body passthrough
  - When no template matches the request Content-Type header
  - When there are no templates defined (recommended)
  - Never

Content-Type

- application/json

Add mapping template

Generate template:

```
{ "firstName": "$input.params('firstName')", "lastName": "$input.params('lastName')" }
```
API Resource flow (needs responses)

/shift - POST - Method Execution

Method Request
- **Auth**: NONE
- **ARN**: arn:aws:execute-api:us-west-2:629829016304:9k196bl1k1/*
  /POST/shift

Integration Request
- **Type**: LAMBDA
- **Region**: us-west-2

Method Response
- Select an integration response.

Integration Response
- No integration responses set up
Test with AWS CLI

$ aws apigateway test-invoke-method --rest-api-id 9k196bl123 --resource-id 56a212 --http-method GET -path-with-query-string "?firstName=GetViaCurl&lastName=Calling" --region us-west-2 --profile adminuser

Response is large

- "status": 200
- body
- Endpoint request body after transformations
- Endpoint response body before transformations
$ curl --insecure https://9k196bl123.execute-api.us-west-2.amazonaws.com/prod/ShiftMgr2?
firstName=GetViaCurl&lastName=Calling

GET output always shows up on a separate line

Web browser should work now too
Demo RESTful store and retrieve

$ curl --insecure https://9k196bl123.execute-api.us-west-2.amazonaws.com/prod/ShiftMgr2?
  firstName=GetViaCurl&lastName=Calling

curl --insecure -X POST -d
  \{"startDateMillis": "1486047600000", \"endDateMillis": "1486090800000"\}"
https://9k196bl123.execute-api.us-west-2.amazonaws.com/prod/shift

GET output always shows up on a separate line
What has this shown?

It does work – RESTful endpoints are possible

Allows outsourcing much of hosting and operations
Caveats

Learning curve
Every so often a response will be very slow
• JVM start up
• connecting to database
Nanofunctions will be integrating at the database, with all the known downsides
If Amazon Lambda closed down most of your code would still work but you'd have to find a new way of hosting it
Alternatives to AWS operations

- serverless.com
- Ansible
"The Serverless Platform is coming"
"Serverless is an MIT open-source project, actively maintained by a vibrant and engaged community of developers."
Node.js version 4 or greater
Can also deploy Python and Java
Supports AWS, Azure, Google Cloud Platform, and IBM OpenWhisk
"Forum Posts – 100+"

google ""serverless framework"" java' → lots of hits
Lambda support new in version 2.2

API Gateway separate github project
- "v0.0.0a Build Status Unstable work in Progress"
- https://github.com/pjodouin/ansible-api-gateway
Potential open source project

Nanofunction hosting inside Java container