Deploy your microservice using AWS S3, AWS API Gateway, AWS Lambda, and Couchbase

Arun Gupta, @arungupta

Docker Captain

Java Champion

JavaOne Rock Star (4 years)

NetBeans Dream Team

Silicon Valley JUG Leader

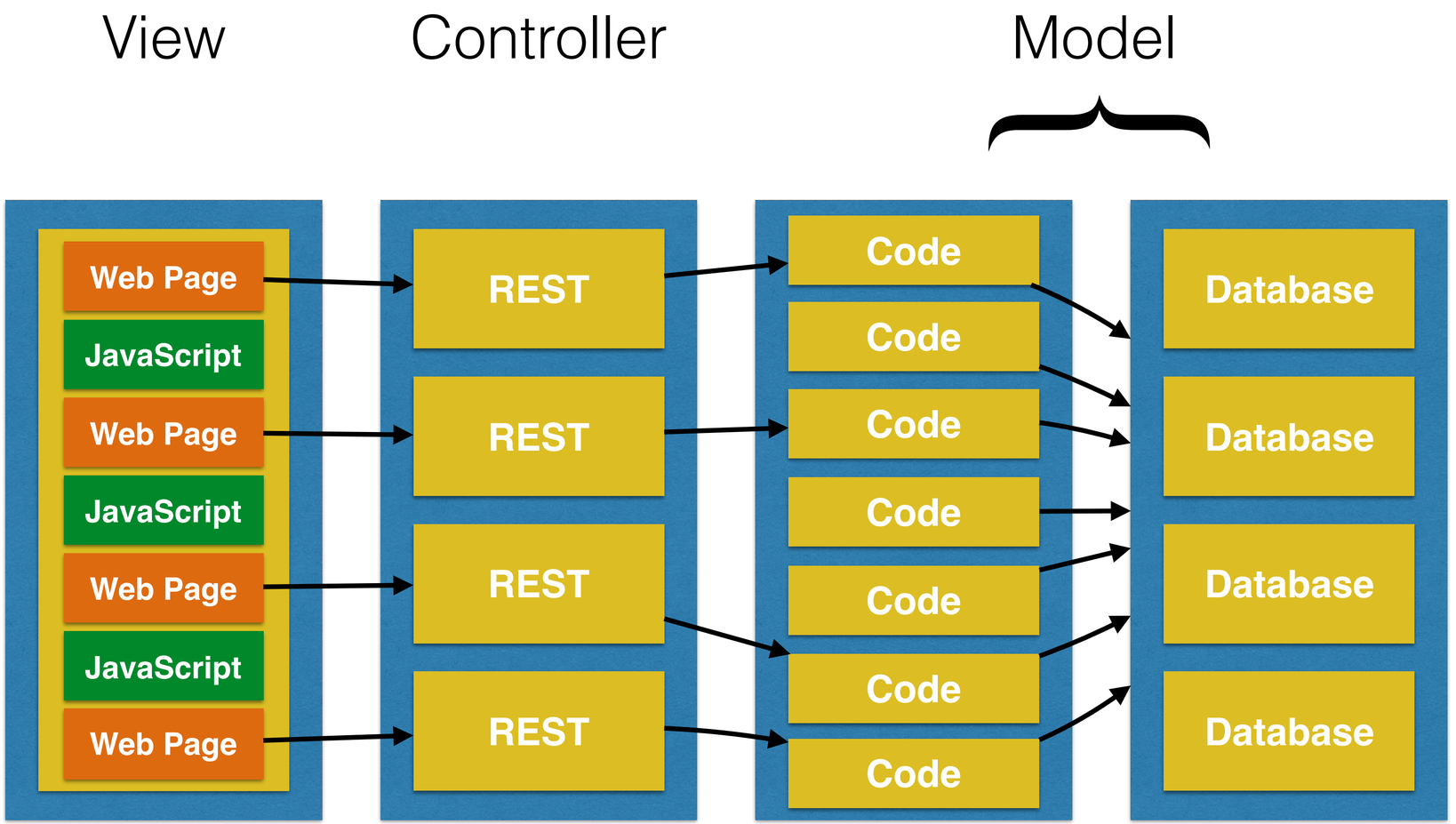
Author

Runner

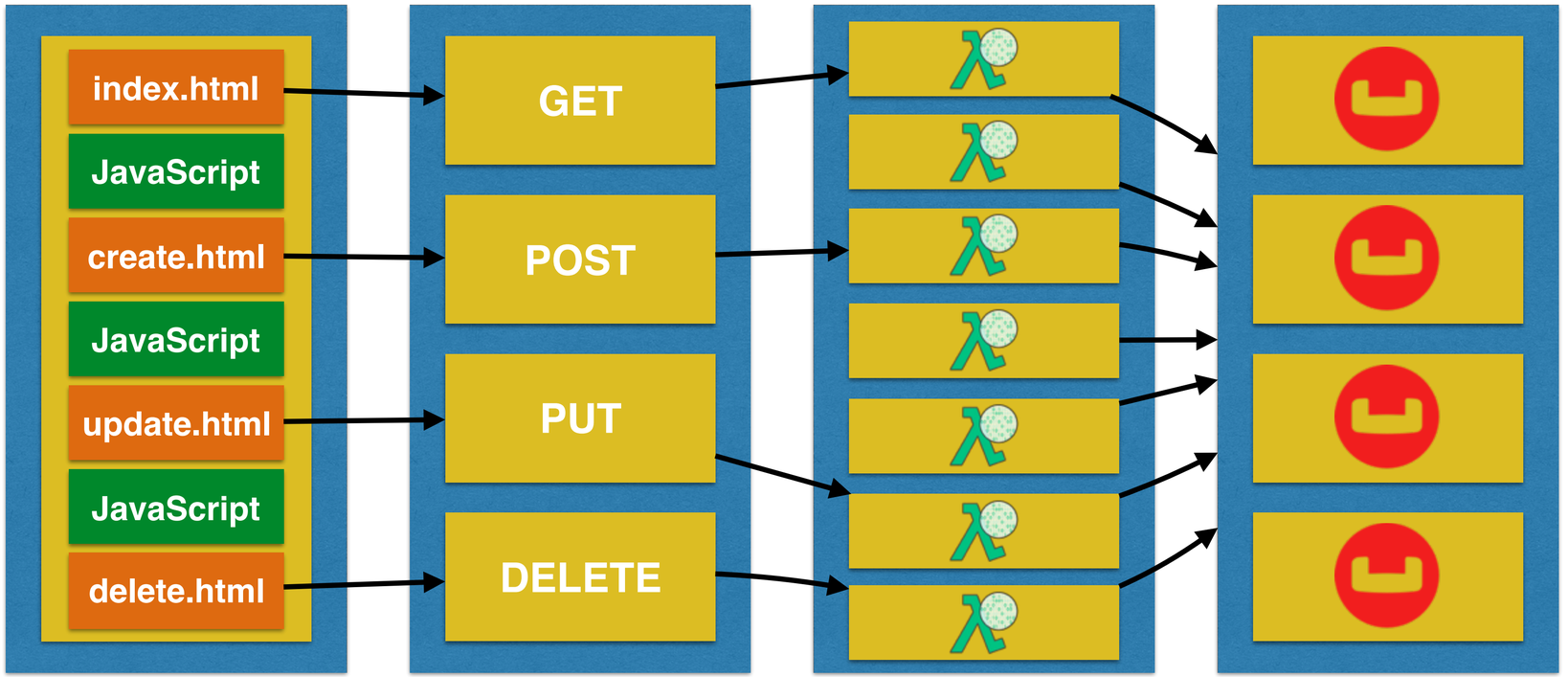
Lifelong learner

the microservice architectural style is an approach to developing a single application as a ***suite of small services****,* ***each running in its own process*** *and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and* ***independently deployable*** *by fully automated deployment machinery. There is a* ***bare minimum of centralized management*** *of these services, which may be written in* ***different programming languages*** *and use* ***different data storage technologies***

https://martinfowler.com/articles/microservices.html


View
Controller
Model
Web Page
JavaScript
Web Page
JavaScript
Web Page
JavaScript
Web Page
REST
REST
REST
REST
Code
Code
Code
Code
Code
Code
Code
Database
Database
Database
Database

S3 API Gateway Lambda Couchbase

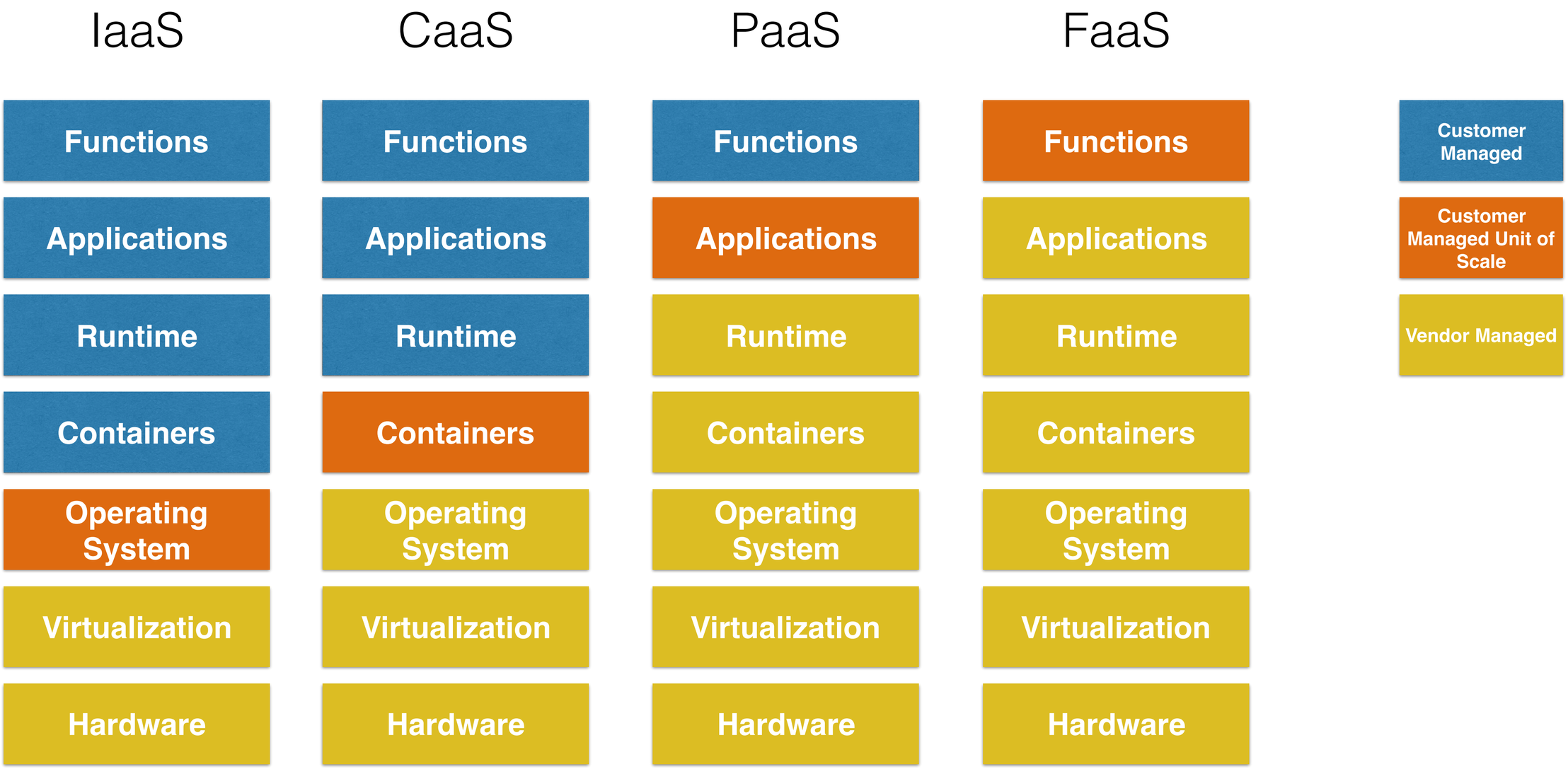


Serverless Computing

Typical Challenges with Server-based Computing

- What size servers for budget/performance?
- Scale servers up/down?
- What O/S?
- O/S settings?
- Patching?
- Control access to servers?
- Deploy new code to server?

	Virtual Machines	Containers	Serverless
Unit of Scale	Machine	Application	Function
Abstraction	Hardware	Operating System	Language Runtime
Packaging	AMI	Container File	Code
Configure	Machine, storage, networking, O/S	Run Servers, configure applications, scaling	Run code when needed
Execution	Multi-threaded, multi-task	Multi-threaded, single task	Single threaded, single task
Runtime	Hours to months	Minutes to days	Microseconds to seconds
Unit of cost	Per VM per hour	Per VM per hour	Per memory/second per request
Amazon	EC2	Docker, Kubernetes, ECS	AWS Lambda


IaaS
CaaS
PaaS
FaaS
Functions
Applications
Runtime
Containers
Operating System
Virtualization
Hardware
Functions
Applications
Runtime
Containers
Operating System
Virtualization
Hardware
Functions
Applications
Runtime
Containers
Operating System
Virtualization
Hardware
Functions
Applications
Runtime
Containers
Operating System
Virtualization
Hardware
Customer Managed
Customer Managed Unit of Scale
Vendor Managed

adrian cockcroft
@adrianco

Following

If your PaaS can efficiently start instances in 20ms that run for half a second, then call it serverless.

> **Julz Friedman** @doctor_julz
> if you think serverless is different than PaaS then either you or I have misunderstood what "serverless" or "PaaS" means

RETWEETS 158 LIKES 207

6:43 AM - 28 May 2016

10 158 207

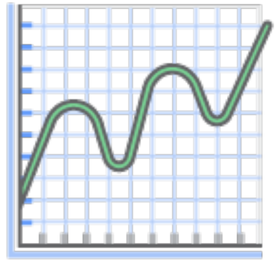
What is AWS Lambda?

Fully Managed

- No provisioning
  - Java, Node, Python, C#
- Zero administration
- High availability

Subsecond Metering

- Charged for every 100ms of execute
- No storage cost



Continuous Scaling

- Automatically
- Scale up and down

How it works?

Upload your code to AWS Lambda

https://aws.amazon.com/lambda/

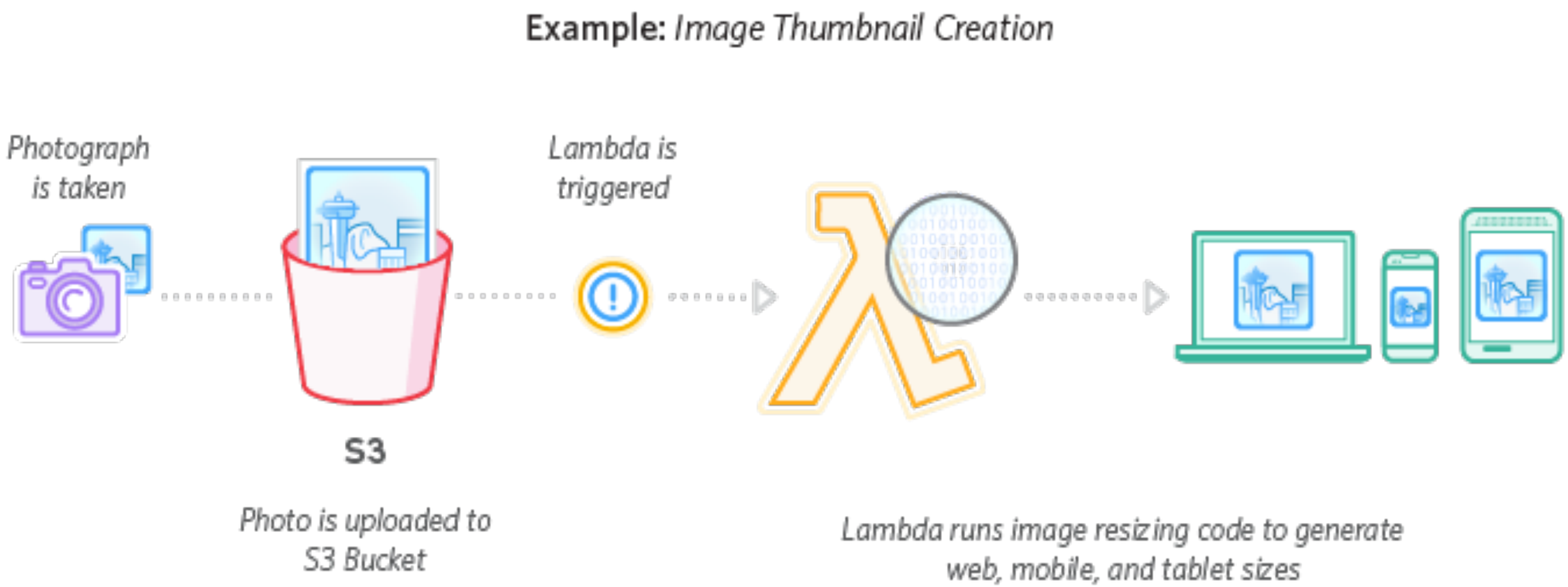
AWS Lambda Pricing

- FREE tier
  - 1M free requests per month
  - 400k GB-seconds of compute time per month
  - CPU and network allocated proportionately
- $0.20 per million requests thereafter

Memory (MB)	Free tier seconds per month	Price per 100ms ($)
128	3,200,000	0.000000208
192	2,133,333	0.000000313
256	1,600,000	0.000000417
320	1,280,000	0.000000521
384	1,066,667	0.000000625
448	914,286	0.000000729
512	800,000	0.000000834

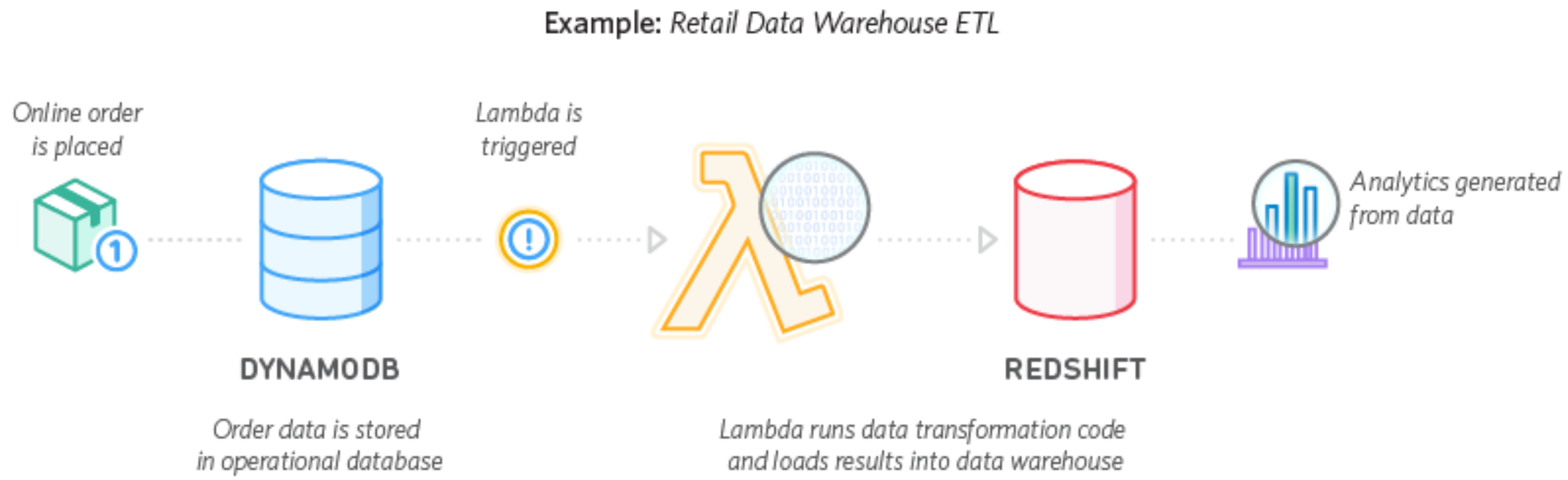
https://aws.amazon.com/lambda/pricing/

AWS Lambda Usecases



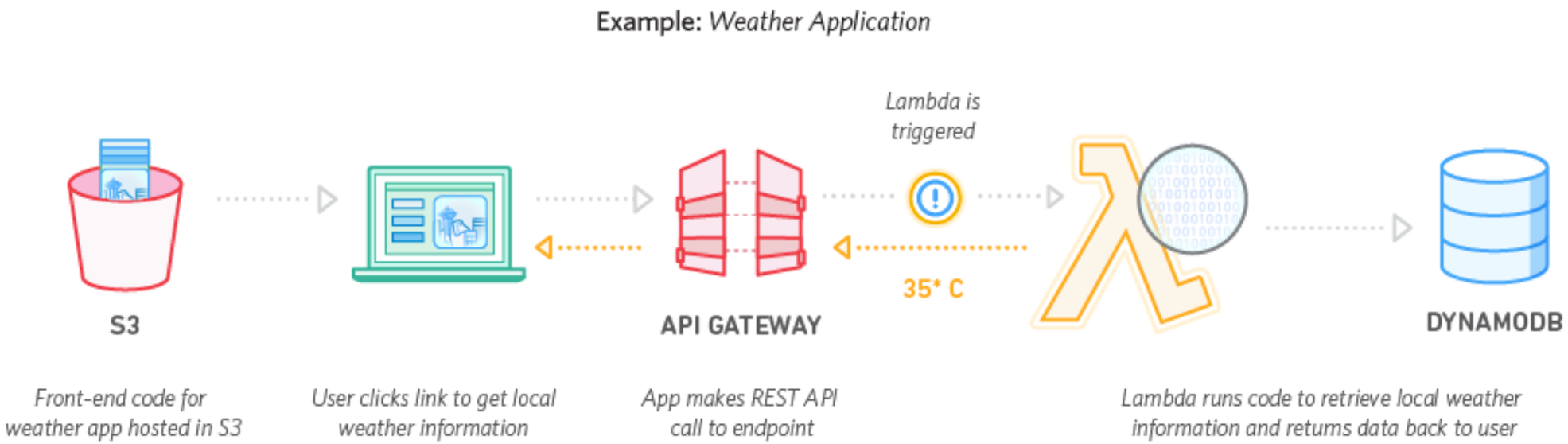
https://aws.amazon.com/lambda/

AWS Lambda Usecases



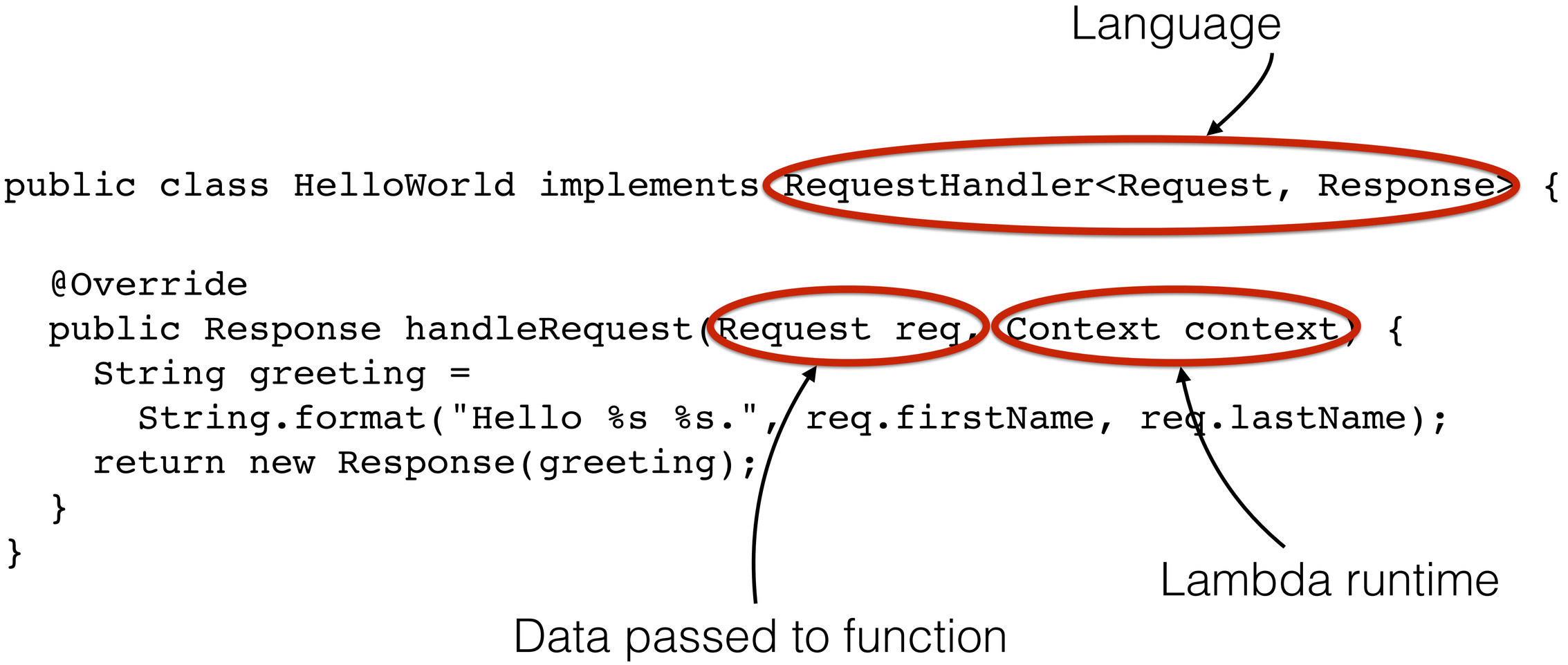
https://aws.amazon.com/lambda/

AWS Lambda Usecases



https://aws.amazon.com/lambda/

Key Components of AWS Lambda



Java + Lambda

Deploy First Java Lambda Function

1

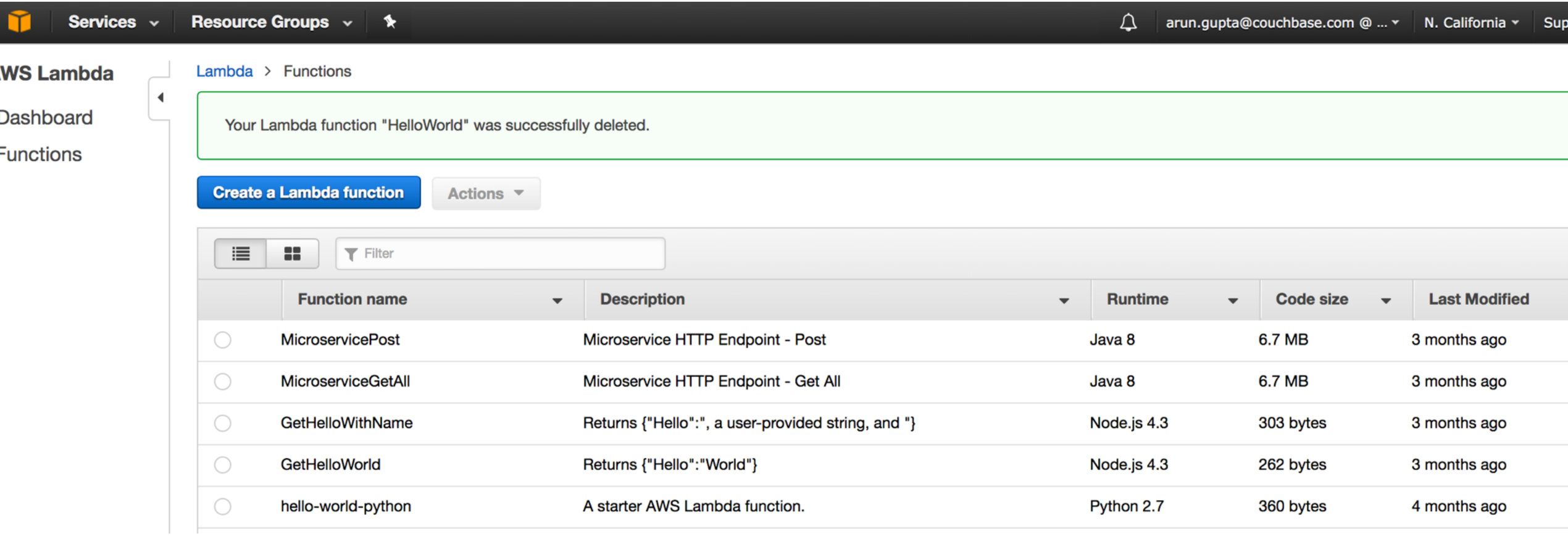
```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>aws-lambda-java-core</artifactId>
  <version>1.1.0</version>
</dependency>
```

2

```
aws lambda create-function \
--function-name HelloWorld \
--role arn:aws:iam::598307997273:role/service-role/myLambdaRole \
--handler org.sample.serverless.aws.helloworld.HelloWorld \
--zip-file fileb:///Users/arungupta/workspaces/serverless/aws/helloworld/
helloworld/target/helloworld-1.0-SNAPSHOT.jar \
--description "Java Hello World" \
--runtime java8  \
--region us-west-1 \
--timeout 30 \
--memory-size 1024 \
--publish
```

3

```
aws lambda invoke \
--function-name HelloWorld \
--region us-west-1 \
--payload '{ "firstName": "John", "lastName": "Smith" }' \
helloworld.out
```


Services
Resource Groups
arun.gupta@couchbase.com @ ...
N. California
AWS Lambda
Dashboard
Functions
Lambda > Functions
Your Lambda function "HelloWorld" was successfully deleted.
Create a Lambda function
Actions
Filter
Function name
Description
Runtime
Code size
Last Modified
MicroservicePost
Microservice HTTP Endpoint - Post
Java 8
6.7 MB
3 months ago
MicroserviceGetAll
Microservice HTTP Endpoint - Get All
Java 8
6.7 MB
3 months ago
GetHelloWithName
Returns {"Hello":", a user-provided string, and "}
Node.js 4.3
303 bytes
3 months ago
GetHelloWorld
Returns {"Hello":"World"}
Node.js 4.3
262 bytes
3 months ago
hello-world-python
A starter AWS Lambda function.
Python 2.7
360 bytes
4 months ago

Couchbase

Data

Query

Index

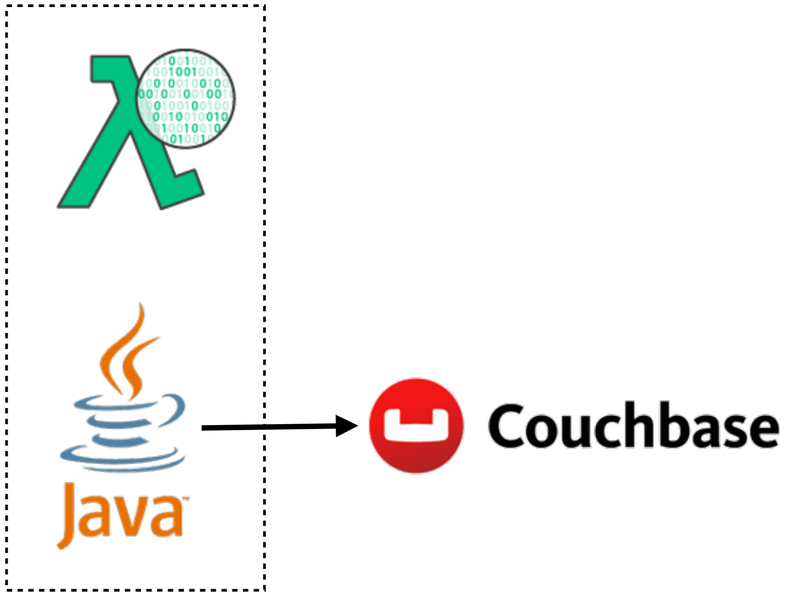
Replication

Mobile

Search

Analytics

Java + Lambda + Couchbase



Java + Couchbase Lambda Function

1

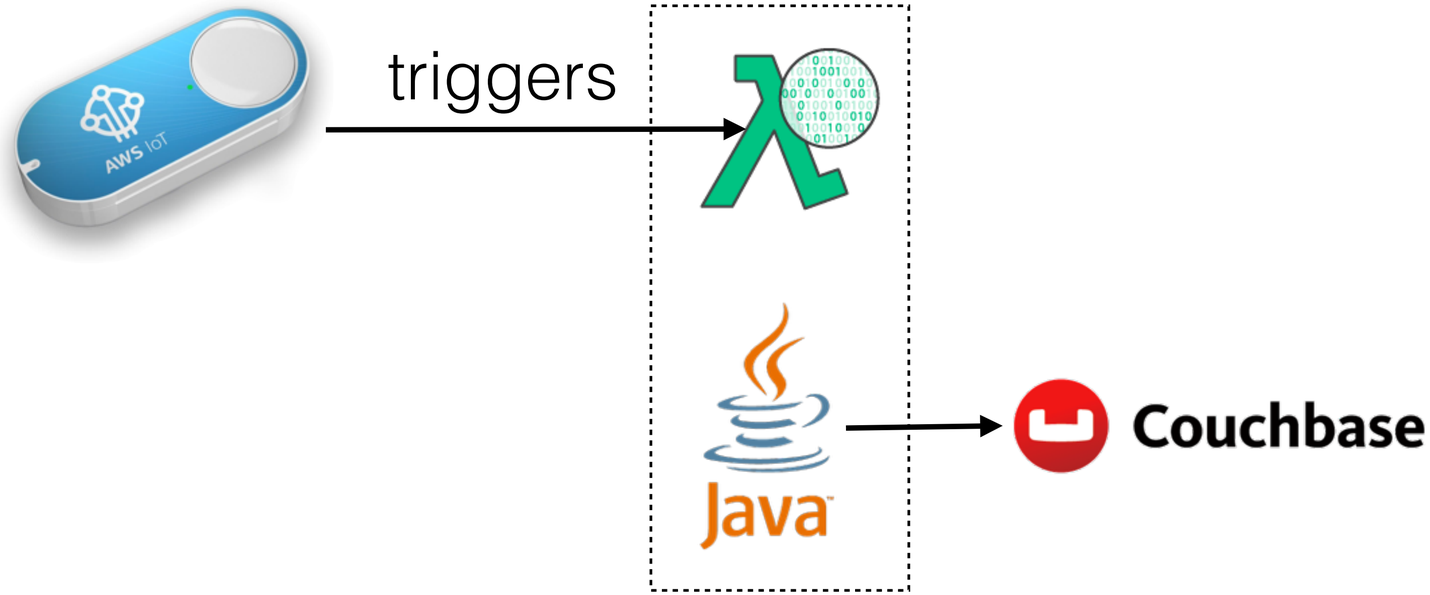
```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>aws-lambda-java-core</artifactId>
  <version>1.1.0</version>
```

```
aws lambda create-function \
--function-name HelloCouchbaseLambda \
--role arn:aws:iam::598307997273:role/service-role/myLambdaRole \
--handler org.sample.serverless.aws.couchbase.HelloCouchbaseLambda \
--zip-file fileb:///Users/arungupta/workspaces/serverless/aws/hellocouchbase/hellocouchbase/
target/hellocouchbase-1.0-SNAPSHOT.jar \
--description "Java Hello Couchbase" \
--runtime java8 \
--region us-west-2 \
--timeout 30 \
--memory-size 1024 \
--environment Variables={COUCHBASE_HOST=ec2-35-165-249-235.us-west-2.compute.amazonaws.com} \
--publish
```

3

```
--function-name HelloWorld \
--region us-west-1 \
--payload '{ "firstName": "John", "lastName": "Smith" }' \
helloworld.out
```

IoT + Java + Lambda + Couchbase



AWS API Gateway

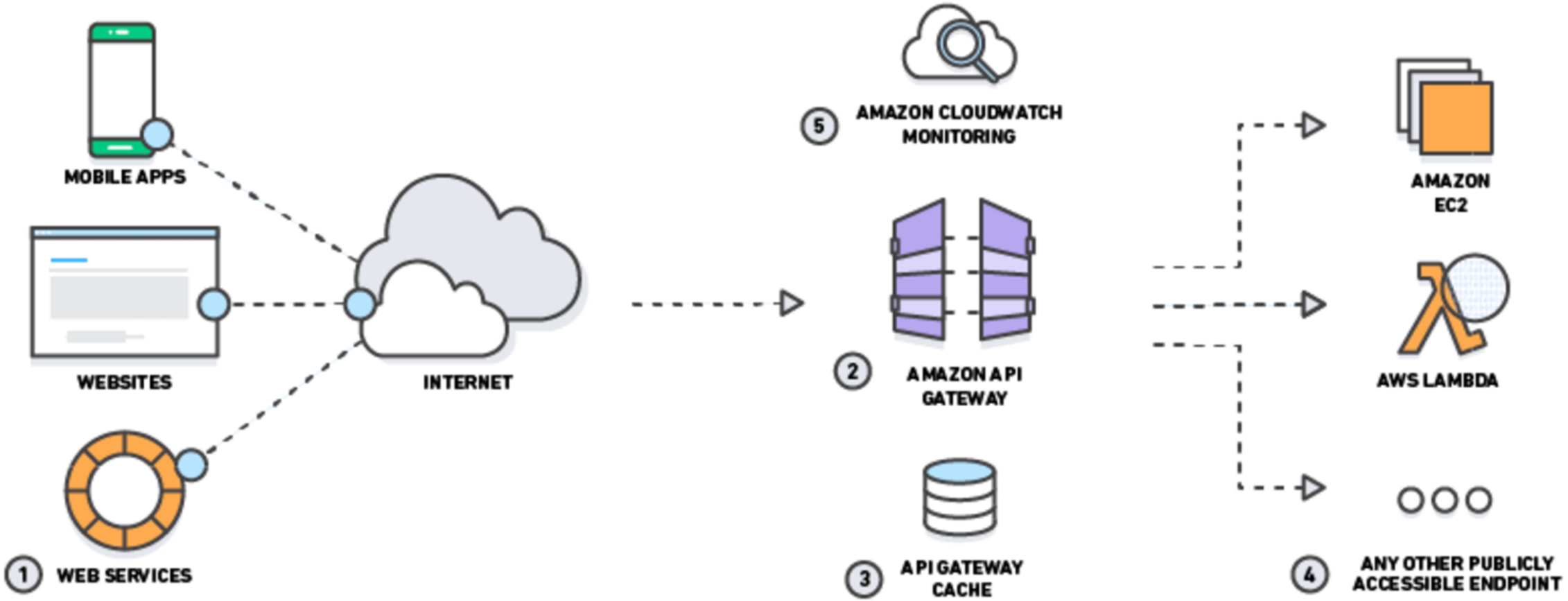
AWS API Gateway

- Create, publish, maintain, monitor and secure RESTful APIs
- Manage multiple stages and version: Iterate, test and release new versions, with backwards compatibiluity
- Operations monitoring: Integrated with CloudWatch
- Low cost and efficient: Only pay for the calls made to APIs and data transfer out

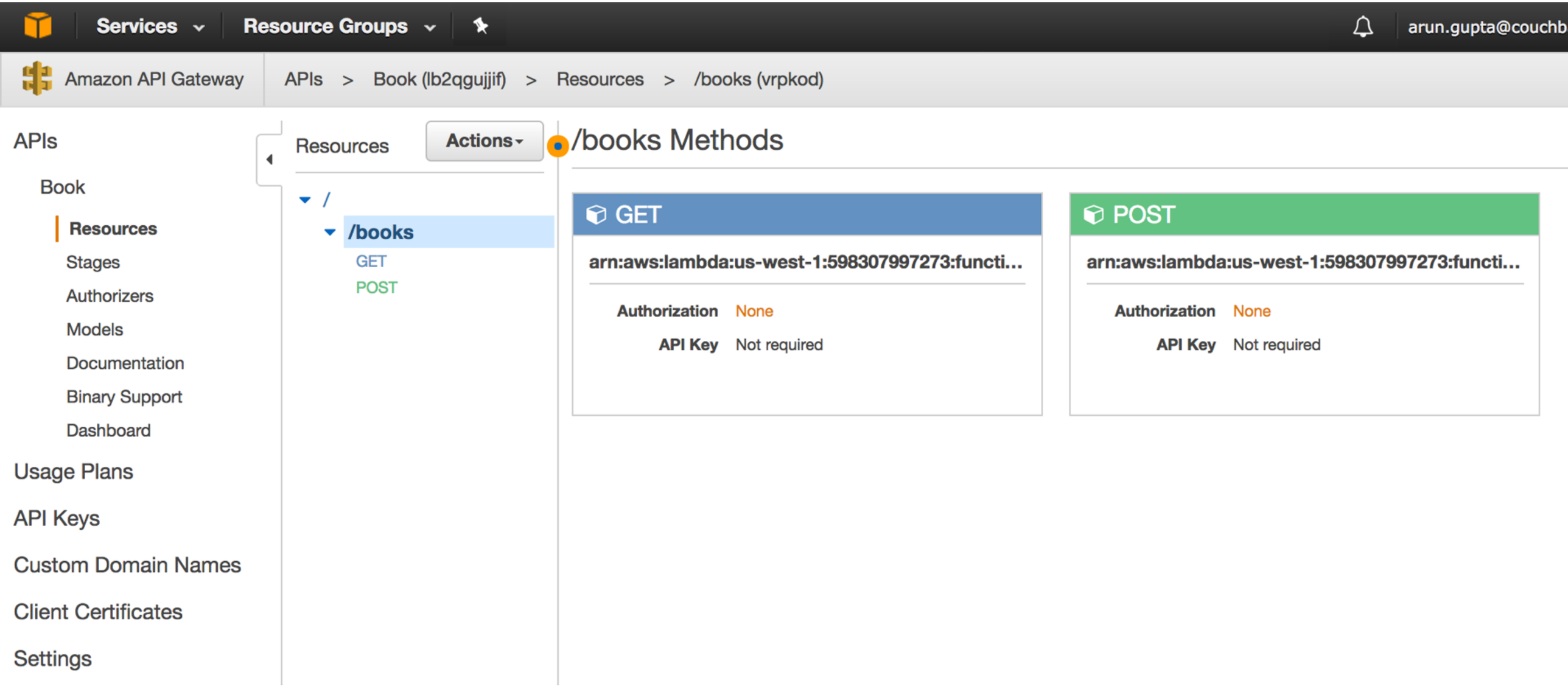
AWS API Gateway

- Traffic management: Set throttle rules
- Authorization and access control: Integrated with AWS IAM and AWS Cognito
- SDK generation
  - JavaScript
  - iOS
  - Android

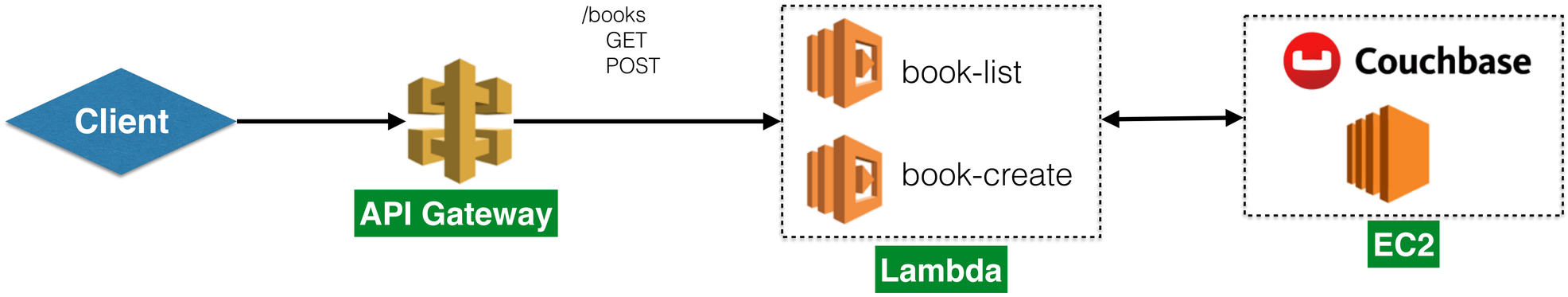
API Call Flow



https://aws.amazon.com/api-gateway/details/

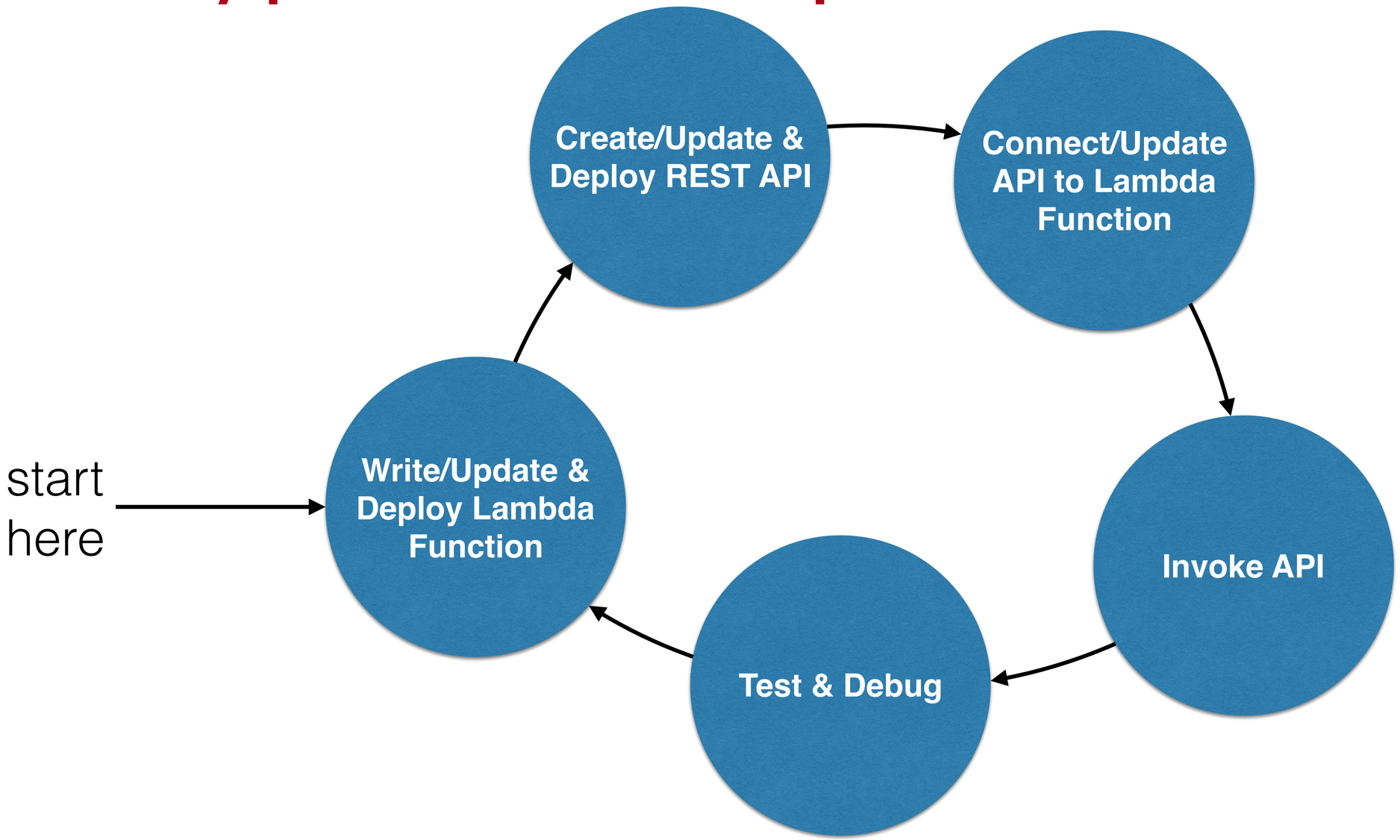

Services
Resource Groups
arun.gupta@couchb
Amazon API Gateway
APIs > Book (lb2qgujjif) > Resources > /books (vrpkod)
APIs
Book
Resources
Stages
Authorizers
Models
Documentation
Binary Support
Dashboard
Usage Plans
API Keys
Custom Domain Names
Client Certificates
Settings
Resources
Actions
/
/books
GET
POST
/books Methods
GET
arn:aws:lambda:us-west-1:598307997273:functi...
Authorization None
API Key Not required
POST
arn:aws:lambda:us-west-1:598307997273:functi...
Authorization None
API Key Not required

Java + Lambda + Couchbase + API



https://blog.couchbase.com/microservice-aws-api-gateway-lambda-couchbase/

Typical development workflow



AWS Serverless Application Model

- Standard application model (SAM) for serverless applications
- Extends CloudFormation
  - New resource types
    - `AWS::Serverless::Function`
    - `AWS::Serverless::Api`
    - `AWS::Serverless::SimpleTable`
  - New event source types: `S3`, `Api`, `Schedule`, …
  - New property types: environment, event source, …

```
AWSTemplateFormatVersion : '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: Microservice using API Gateway, Lambda and Couchbase
Resources:
  MicroserviceGetAll:
    Type: AWS::Serverless::Function
    Properties:
      Handler: org.sample.serverless.aws.couchbase.BucketGetAll
      Runtime: java8
      CodeUri: s3://serverless-microservice/microservice-http-endpoint-1.0-SNAPSHOT.jar
      Timeout: 30
      MemorySize: 1024
      Environment:
        Variables:
          COUCHBASE_HOST: ec2-35-163-21-104.us-west-2.compute.amazonaws.com
      Role: arn:aws:iam::598307997273:role/microserviceRole
```

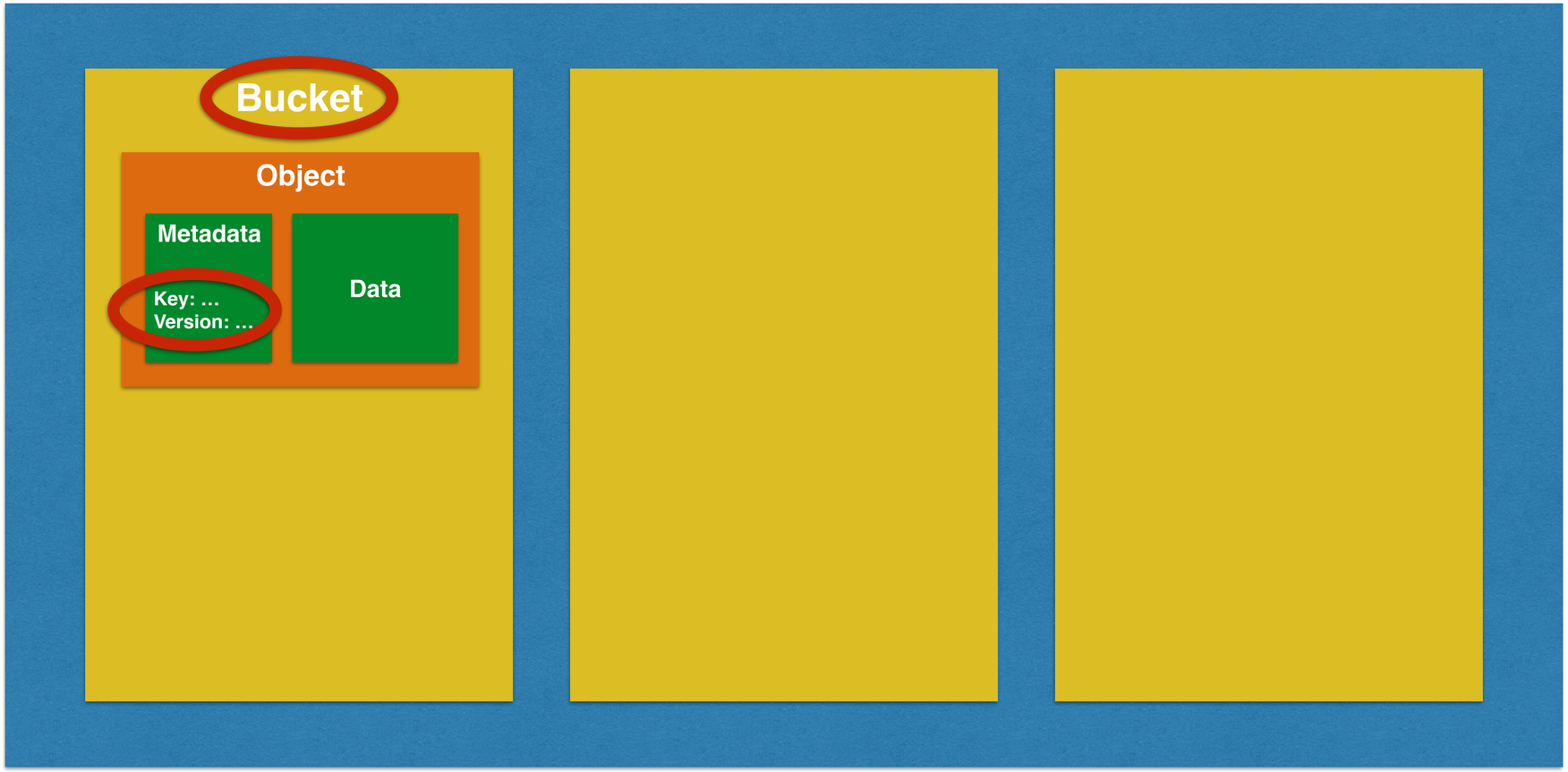

https://github.com/arun-gupta/serverless/blob/master/aws/microservice/template.yml

```
29    MicroserviceGetAllGateway:
30      Type: AWS::Serverless::Function
31      Properties:
32        Handler: org.sample.serverless.aws.couchbase.gateway.BucketGetAll
33        Runtime: java8
34        CodeUri: s3://serverless-microservice/microservice-http-endpoint-1.0-SNAPSHOT.jar
35        Timeout: 30
36        MemorySize: 1024
37        Environment:
38          Variables:
39            COUCHBASE_HOST: ec2-35-163-21-104.us-west-2.compute.amazonaws.com
40        Role: arn:aws:iam::598307997273:role/microserviceRole
41        Events:
42          GetResource:
43            Type: Api
44            Properties:
45              Path: /books
46              Method: get
```

AWS S3 Basics

- **Simple**: Console, REST API and AWS SDKs
- **Durable**: 99.999999999%
- **Scalable**: Gigabytes -> Exabytes
  - Store/retrieve data, any time, anywhere
- **Access control**: type of access (e.g. `READ` and `WRITE`)
- **Authentication**: verify identity of the user

AWS S3 Concepts



Amazon S3 > serverless-microservice

Objects | Properties | Permissions | Management

Versioning

Keep multiple versions of an object in the same bucket.

Learn more

Disabled

Logging

Set up access log records that provide details about access requests.

Learn more

Disabled

Static website hosting

Host a static website, which does not require server-side technologies.

Learn more

Disabled

Advanced settings

Tags

Use tags to track your cost against projects or other criteria.

Learn more

0 Tags

Cross-region replication

Automate copying objects across different AWS Regions.

Learn more

Disabled

Transfer acceleration

Enable fast, easy and secure transfers of files to and from your bucket.

Learn more

Suspended

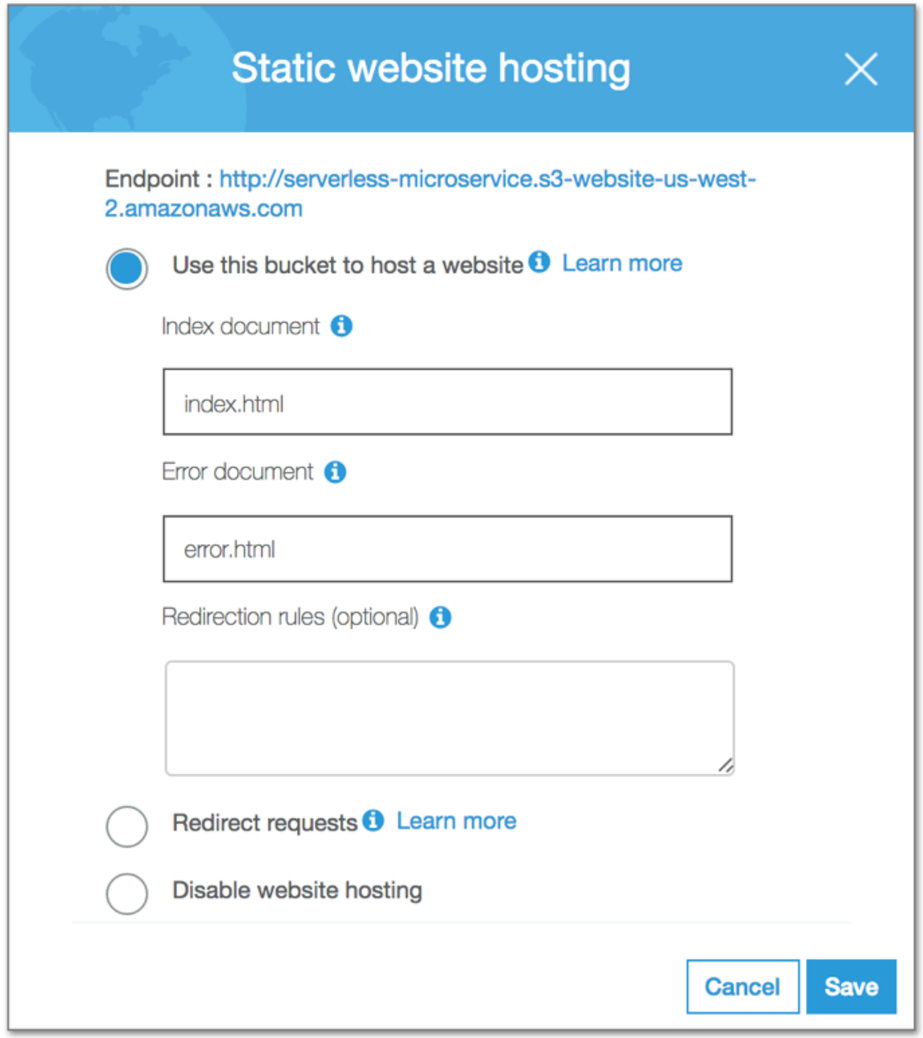
Events

Receive notifications when specific events occur in your bucket.

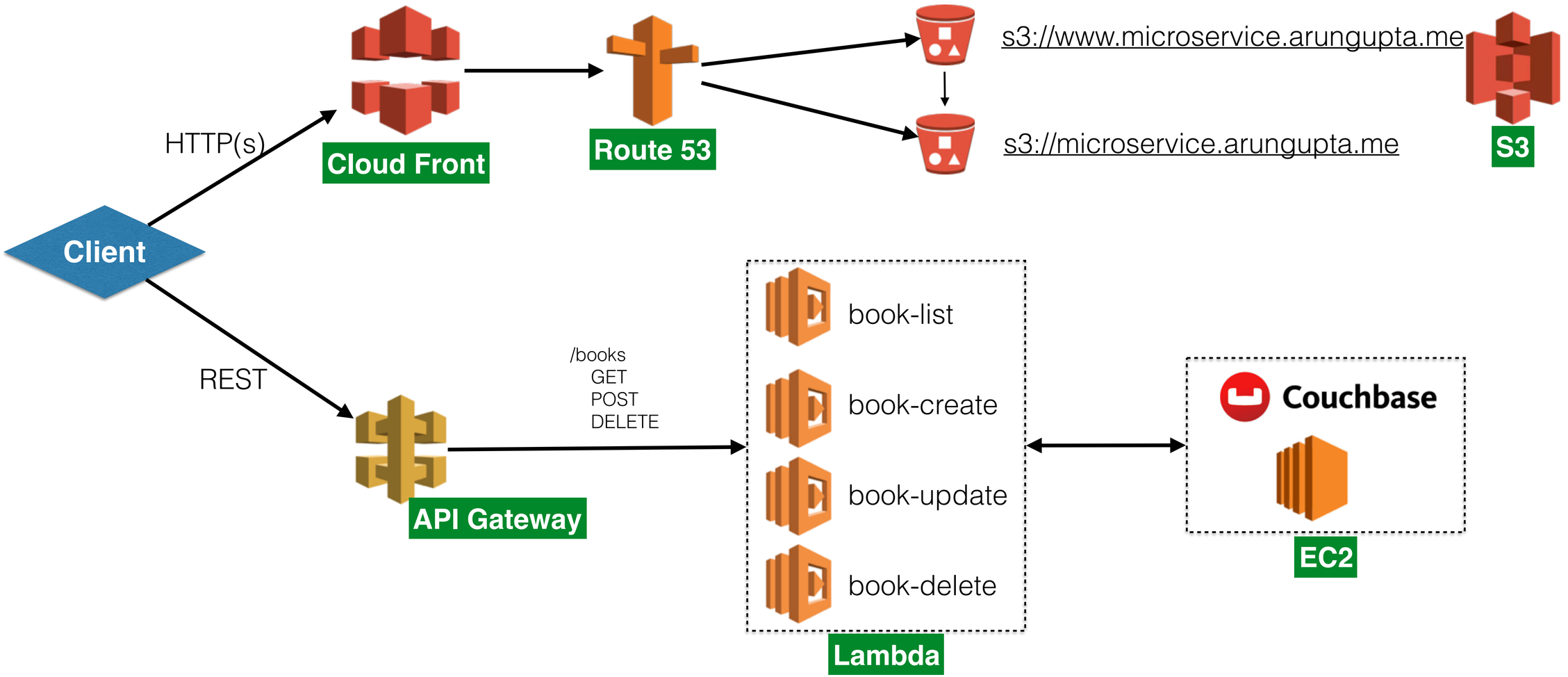
Learn more

0 Active notifications

Hosting Static Websites on S3



Microservice Deployment Architecture



the microservice architectural style is an approach to developing a single application as a ***suite of small services****,* ***each running in its own process*** *and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and* ***independently deployable*** *by fully automated deployment machinery. There is a* ***bare minimum of centralized management*** *of these services, which may be written in* ***different programming languages*** *and use* ***different data storage technologies***

https://martinfowler.com/articles/microservices.html

References

- Amazon Web Services: aws.amazon.com
- Couchbase: couchbase.com
- Slides + Code: github.com/arun-gupta/serverless