



Code Camp

Serverless Platforms

Jason Barbee
Solutions Architect
CCIE #18039

Agenda

1. Evolution of a Platform
2. Consumable Platforms today
3. Function as a Service
4. Lab Outline

Compute in 2000's - Physical

Physical Server Farm

Provision Time - days



Compute in 2010 - Virtualization

VMWare Servers on physical hardware, SANs, Networking
Provision Time - ~ 1-2 hours



Compute in ~2014 - Virtual Machines

- Spin up VMs at your favorite provider -
- TekLinks, AWS, Google, Azure, dozens of providers...

Compute in 2017 - Microservices and APIs

- FAAS / Function as a Service Providers.
- AWS, Google, Azure, Webtasks
- Your customer's routers.
- Your customer's compute, or hyperconverged.

Routers = Cloud

Routers can run third party service containers. AKA - your code

Host Your Applications on a Device

Cisco service devices support hosting applications directly on the devices for network management, monitoring and other needs.

IOS-XE

Supports 3rd party KVM container and UCS E Series Server module.

IOS-XR

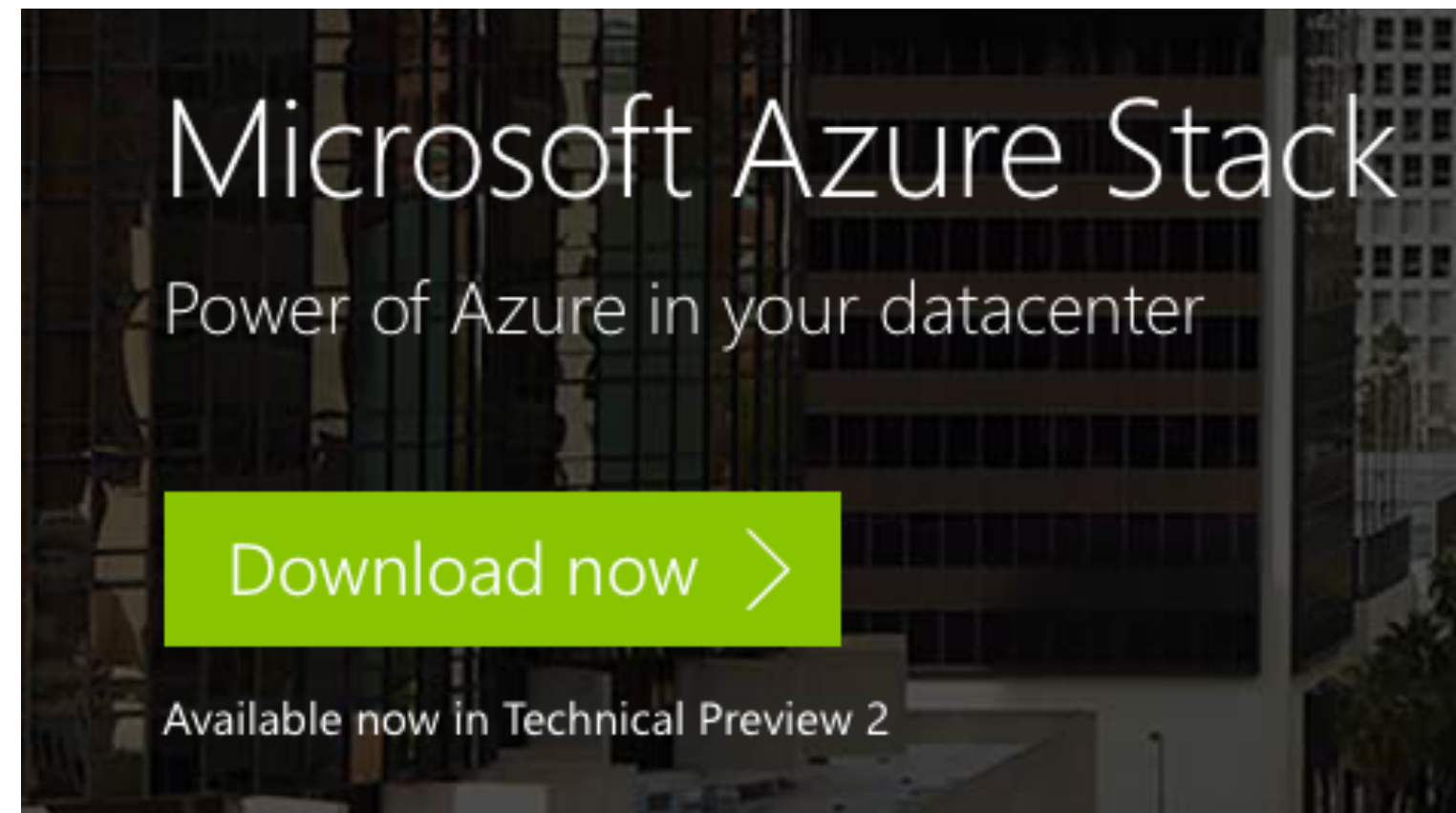
Supports RPM package installation and 3rd Party LXC containers.

Open NX-OS

Supports 3rd party LXC containers and "Guest Shell" container.

Servers = Cloud

Azure wants to run on your bare metal. Same APIs as public cloud



Servers = Cloud

OpenStack wants to run on bare metal and provide APIs for compute, storage and networking.

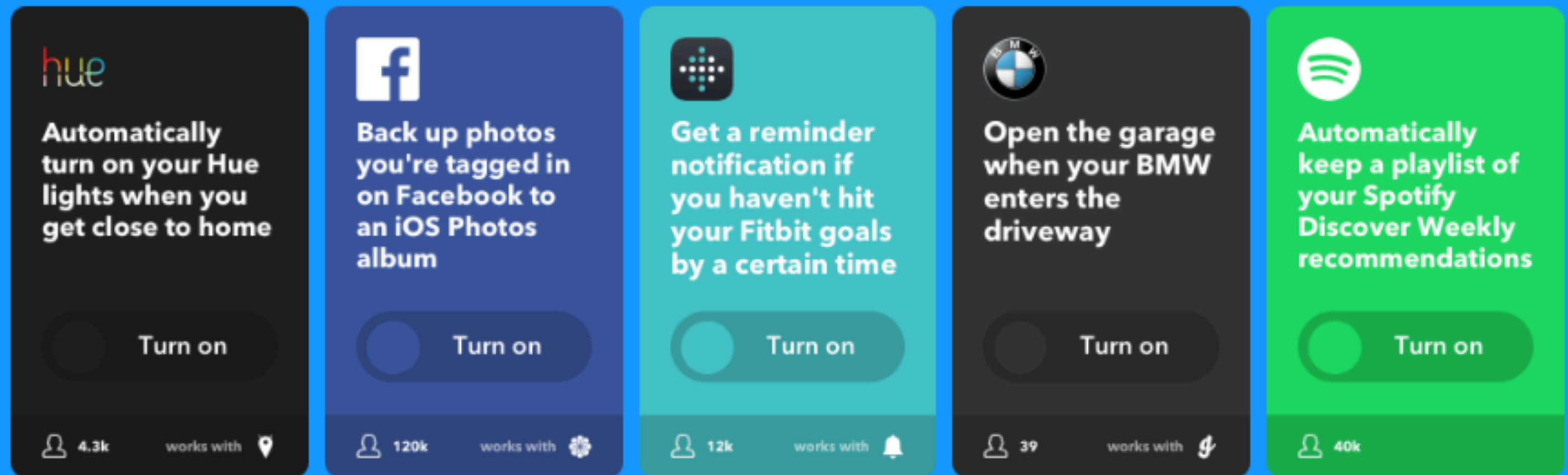


Agenda

1. Evolution of a Platform
2. **Consumable Platforms today**
3. Bot Revolution
4. Function as a Service
5. Lab Outline

API Connector Services - IFTTT

Applets add new experiences to your service



Go beyond IF THIS THEN THAT

IFTT - Custom REST Actions to anything



- Add custom actions with Maker Channel to any URL.
- Send a REST call to any endpoint on the internet.
- Make a Spark call, Tropo Call, Turn on/Off lights.

Zapier.com - Business Logic

- More logic and customization of actions.
- Still End User Friendly. Free and commercial plans.
- 750+ API integrations built in

Drag the fields into the form on the right to customize Mandrill's Send Email. ?

Address City

Address Country

Address Country Code

Address Name

Address State

Address Status

Address Street

Address Zip

Charset

Custom

First Name

Handling Amount

Item Names

Item Name

Item Number

Last Name

Mc Currency

Mc Fee

Mc Gross

Notify Version

Payer Email

Payer Id

Payer Status

Payment Date

Payment Fee

Payment Gross

Payment Status

Payment Type

Protection Eligibility

Quantity

Receiver Email

Receiver Id

Residence Country

Shipping

Tax

Test Ipn

Transaction Subject

Txn Id

Txn Type

Verify Sign

Customize Mandrill's Send Email. You can manually enter information if you want it included in every Send Email. ?

From Email (required)
Choose an email recognizable to your recipients.
me@example.com ✓

To Email (required)
{{payer_email}} ✓

Subject (required)
Thanks for your purchase from My Store, LLC! ✓

From Name (required)
The display name for the sent email.
My Store, LLC ✓

Body HTML (optional)
You must provide either body text or html, or both.

Body Text (optional)
You must provide either body text or html, or both.
Hello from My Store!
You've just purchased {{payment_gross}} worth of stuff from us, including {{item_names}}

DRAG... →

Agenda

1. Evolution of a Platform
2. Consumable Platforms today
3. **Bot Revolution**
4. Function as a Service
5. Lab Outline

Bots the new User interface

Why would I want to talk to a bot?

We all hate calling and navigating IVRs.

Bot Platforms

- Gupshup.com
- Recast.ai
- wit.ai
- Flint Framework for Node
- Microsoft Bot Framework
- Serverless Bots with URL Web Hooks and Actions

Agenda

1. Evolution of a Platform
2. Consumable Platforms today
3. Bot Revolution
4. **Function as a Service**
5. Lab Outline

Function-As-A-Service

- Function as a Service to run small units of code on requests or events
- Read or Write to a database.
- Call a Spark Log
- Call a payment processing and return a value.



Google Cloud Functions

hook.io



webtask

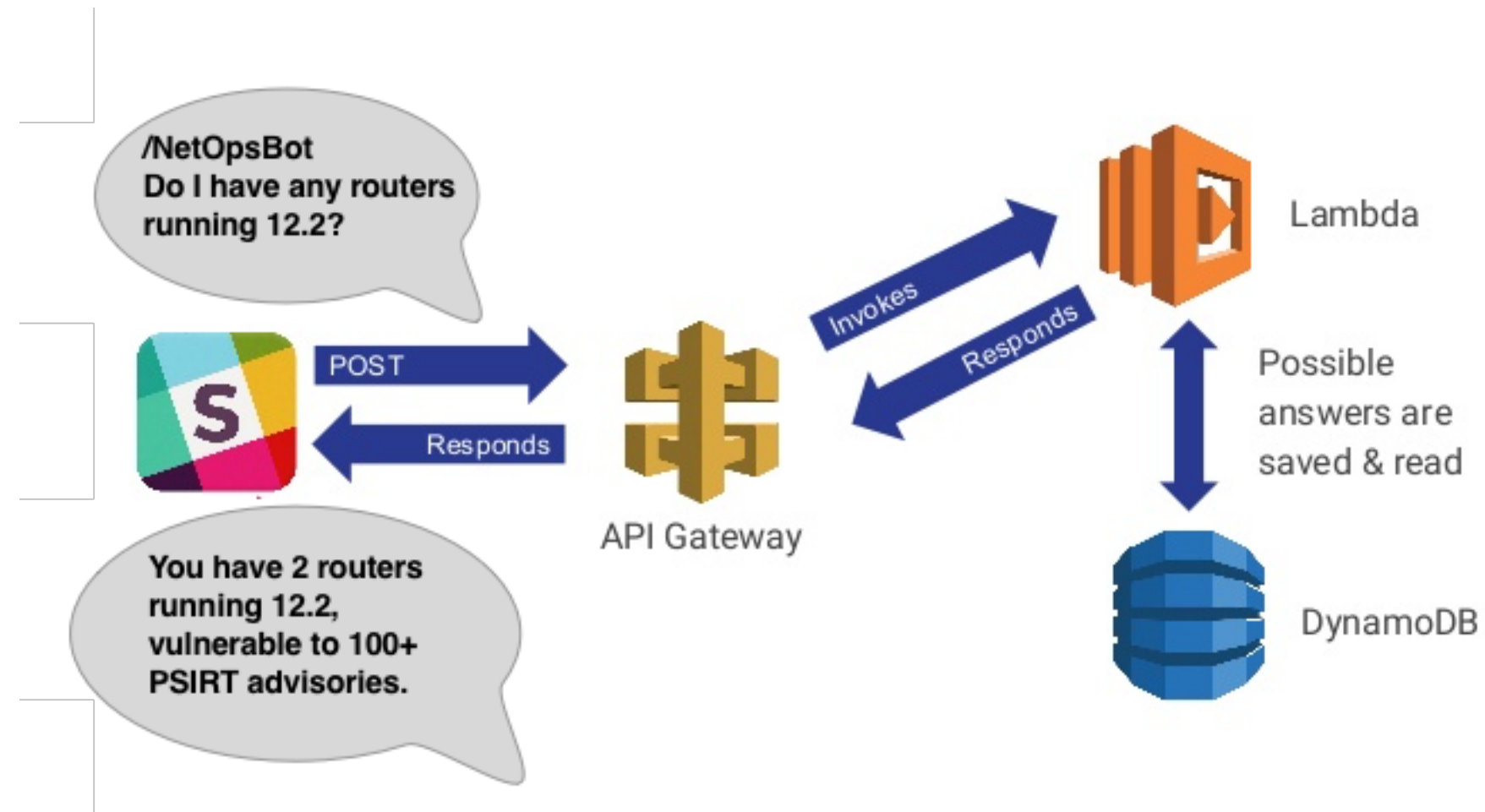


AWS Lambda



Microsoft Azure

AWS Microservice Example



AWS API Gateway

- Accepts GET, POST, PUT all the REST APIs.

`http://mynetopsbot.exampleamazon.com`

`{`

`"Email": "jason.barbee@gmail.com",`

`"Message": "do I have any routers running 12.2"`

`}`

AWS Components

Lambda

Our example : Lambda parses the language and intention, queries a database for security issues for 12.2. Returns results to Lambda for processing.

AWS Microservice Example

Lambda - adds language back to the message and sends to a Spark Web Hook.

<http://spark.cisco.com>

```
{  
  "message" : "You have 2 routers running 12.2 vulnerable to  
100+ PSIRT advisories"  
  "roomID" : "1234123213125125234234"  
}
```

Storing Data

DynamoDB - NoSQL Database

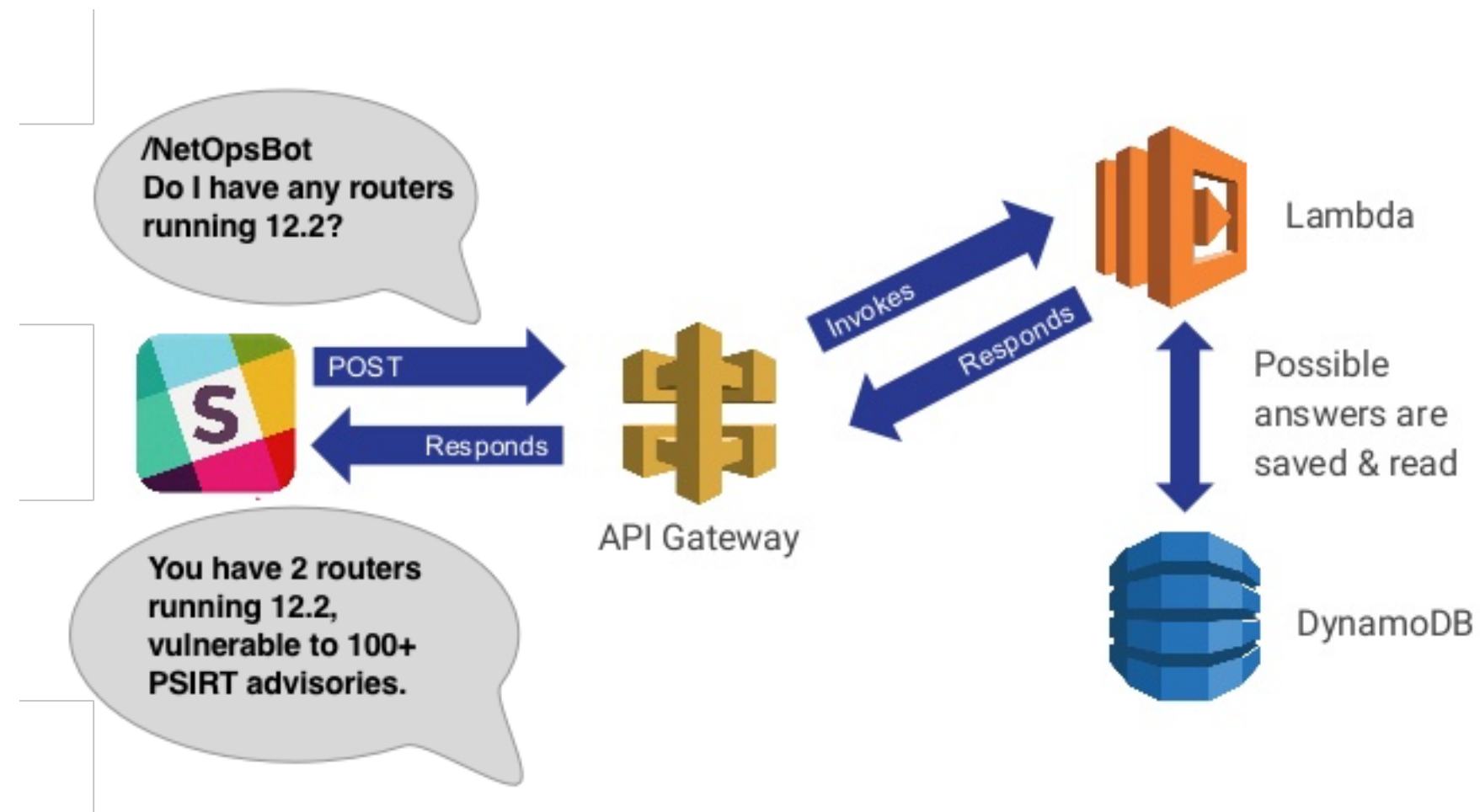
Let's start with SQL and compare to NoSQL

```
INSERT INTO book (  
    `ISBN`, `title`, `author`  
)  
VALUES (  
    '9780992461256',  
    'Mastering Windows NT 4.0',  
    'John Smith'  
);
```


DynamoDB - NoSQL Example

```
db.book.insert({  
  ISBN: "9780992461256",  
  title: "Full Stack JavaScript",  
  author: "Colin Ihrig & Adam Bretz"  
});
```

AWS Microservice Example Review



**What does a
microservice function
look like?**

Hello World at hook.io

Hello World

```
module['exports'] = function helloWorld (hook) {  
  hook.res.end("Hello world!");  
};
```

Make a API Gateway example

Login to AWS Console

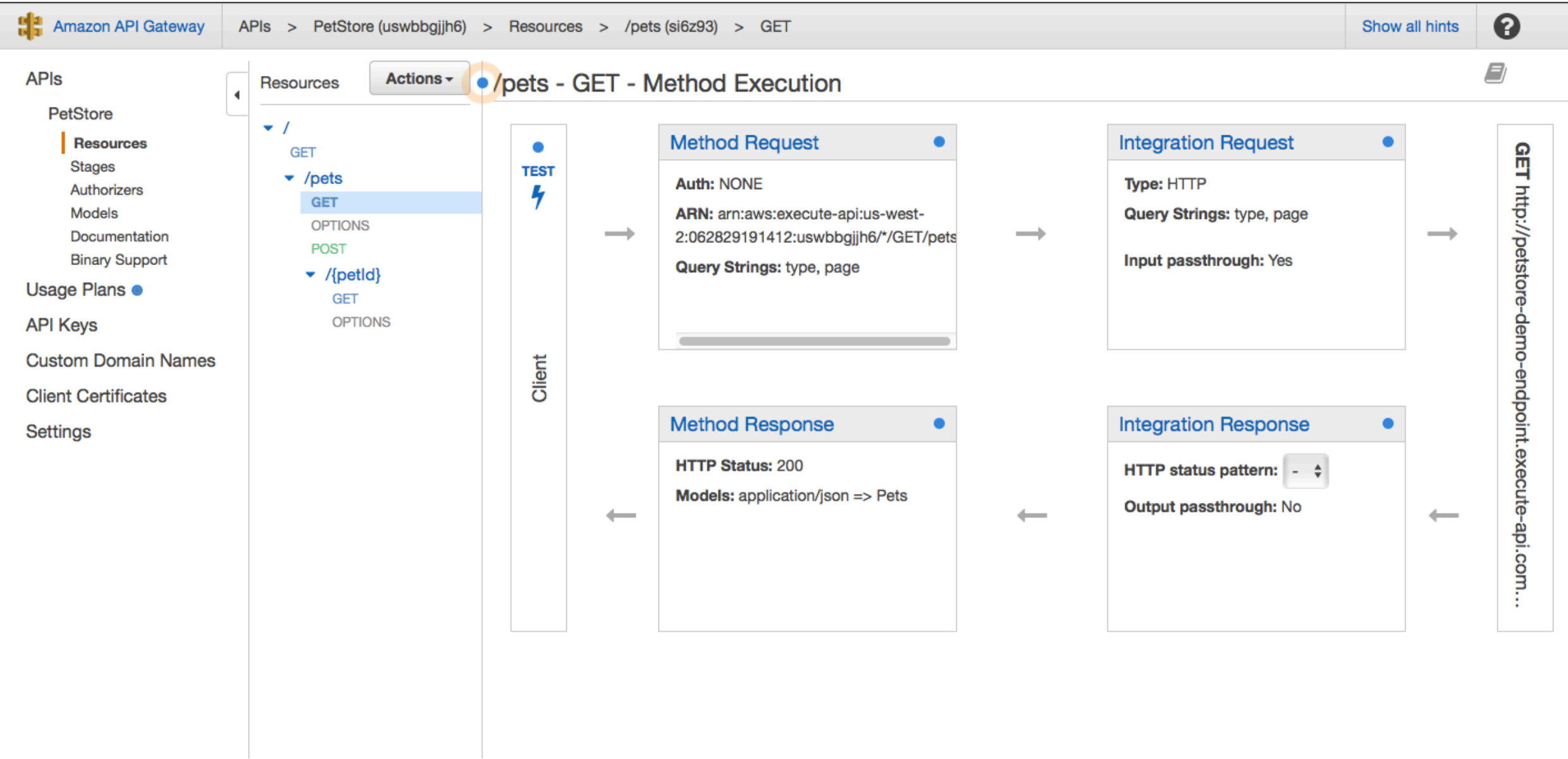
Go to API Gateway. Click Getting Started. You'll get the default PetStore API.

Example API Gateway

AWS API Gateway

Click Getting Started to import the example PetStore API.

My PetStore API - GET



PetStore GET - Response/Test

Amazon API Gateway

APIs > PetStore (uswbbgj6) > Resources > /pets (si6z93) > GET

Show all hints

?

APIs

PetStore

Resources

Stages

Authorizers

Models

Documentation

Binary Support

Usage Plans

API Keys

Custom Domain Names

Client Certificates

Settings

Resources

Actions

Method Execution

/pets - GET - 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

type

Value

page

Value

Headers

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

Stage Variables

No [stage variables](#) exist for this method.

Client Certificate

No client certificates have been generated.

Request Body

Request Body is not supported for GET methods.

Request: /pets

Status: 200

Latency: 244 ms

Response Body

```
[
  {
    "id": 1,
    "type": "dog",
    "price": 249.99
  },
  {
    "id": 2,
    "type": "cat",
    "price": 124.99
  },
  {
    "id": 3,
    "type": "fish",
    "price": 0.99
  }
]
```

Response Headers

```
{"Access-Control-Allow-Origin":"*","X-Amzn-Trace-Id":"Root=1-587e9b95-f366463f2597a5e3f1ba4ea9","Content-Type":"application/json"}
```

Logs

```
Execution log for request test-request
Tue Jan 17 22:32:53 UTC 2017 : Starting execution for request: test-invoke-request
Tue Jan 17 22:32:53 UTC 2017 : HTTP Method: GET, Resource Path: /pets
```


Ok this isn't Fair

I used the template, and PetStoreAPI doesn't actually do anything.

AWS API Gateway

You have to define all the endpoints one by one
create functions, link them to the API one by one
upload files and NPM packages to S3 if you want any imports
Stage dev and production if you want to have a backup copy
hot and ready
Basically there's a lot of stuff to wire together, and a lot of
clicks.

Meet the Serverless Framework - www.serverless.com

**SERVERLESS
FRAMEWORK**
VERSION 1.0

Build auto-scaling, pay-per-execution,
event-driven apps on AWS Lambda

▶ WATCH THE VIDEO

📖 READ THE DOCS

```
# Install serverless globally
$ npm install serverless -g

# Create an AWS Lambda function in Node.js
$ serverless create --template aws-nodejs

# Deploy to live AWS account
$ serverless deploy

# Function deployed!
$ http://api.amazon.com/users/update

-> Read the docs or connect with the community
```

Serverless magic

- Cloudformation to build your environment - API, Lambda, Storage, Databases.
- API Versioning
- Staging to Production

Example of Serverless Deployment

```
vagrant@vagrant:/vagrant/Serverless$ serverless config credentials --provider aws --key mykey --secret mysecret
Serverless: Setting up AWS...
Serverless: Saving your AWS profile in "~/.aws/credentials"...
Serverless: Success! Your AWS access keys were stored under the "default" profile.
vagrant@vagrant:/vagrant/Serverless$ serverless deploy
Serverless: Packaging service...
Serverless: Uploading CloudFormation file to S3...
Serverless: Uploading service .zip file to S3 (3.6 MB)...
Serverless: Updating Stack...
Serverless: Checking Stack update progress...
.....
Serverless: Stack update finished...
Service Information
service: serverless-rest-api-with-dynamodb
stage: dev
region: us-east-1
api keys:
  None
endpoints:
  POST - https://3snrpqj7tj.execute-api.us-east-1.amazonaws.com/dev/routers
  GET - https://3snrpqj7tj.execute-api.us-east-1.amazonaws.com/dev/routers
  GET - https://3snrpqj7tj.execute-api.us-east-1.amazonaws.com/dev/routers/{id}
  PUT - https://3snrpqj7tj.execute-api.us-east-1.amazonaws.com/dev/routers/{id}
  DELETE - https://3snrpqj7tj.execute-api.us-east-1.amazonaws.com/dev/routers/{id}
functions:
  serverless-rest-api-with-dynamodb-dev-update: arn:aws:lambda:us-east-1:062829191412:function:serverless-rest-api-with-dynamodb-dev-update
  serverless-rest-api-with-dynamodb-dev-get: arn:aws:lambda:us-east-1:062829191412:function:serverless-rest-api-with-dynamodb-dev-get
  serverless-rest-api-with-dynamodb-dev-list: arn:aws:lambda:us-east-1:062829191412:function:serverless-rest-api-with-dynamodb-dev-list
  serverless-rest-api-with-dynamodb-dev-create: arn:aws:lambda:us-east-1:062829191412:function:serverless-rest-api-with-dynamodb-dev-create
  serverless-rest-api-with-dynamodb-dev-delete: arn:aws:lambda:us-east-1:062829191412:function:serverless-rest-api-with-dynamodb-dev-delete
vagrant@vagrant:/vagrant/Serverless$
```

Lab 4 - Going Serverless

You will get to build/deploy/play with

- * Hook.io Microservice that logs to a Spark Room.
- * AWS REST API that actually interfaces with a DynamoDB.
- * Using Ansible to push inventory data into that AWS REST API.

Wrapping up

- No we don't all have to write code tomorrow.
- There's never been a more exciting time in our field.
- Engineers are building cool stuff and sharing it.
- Open source is better than ever, and has a lot of corporate backing.

Call to Action:

Actively seek opportunities and Engage us (myself and Jeremy Sanders) to

- Tie collaboration or workflow platforms together
- Add human like interactions to common tasks via chat bots
- Write small apps that integrate business processes together
- Help you deploy and learn Ansible.
- Write scripting to be more efficient on projects if you find yourself facing a large repetitive task set or a chain of processes.

Thank you.

Questions?