Microservices and FaaS for Offensive Security

Ryan Baxendale

BSides Las Vegas (2017)

$ whoami

Ryan Baxendale

Penetration Tester
Centurion Information Security Pte Ltd - www.centurioninfosec.sg
Singapore

twitter.com/ryancancomputer
github.com/ryanbaxendale
linkedin.com/in/ryanbaxendale

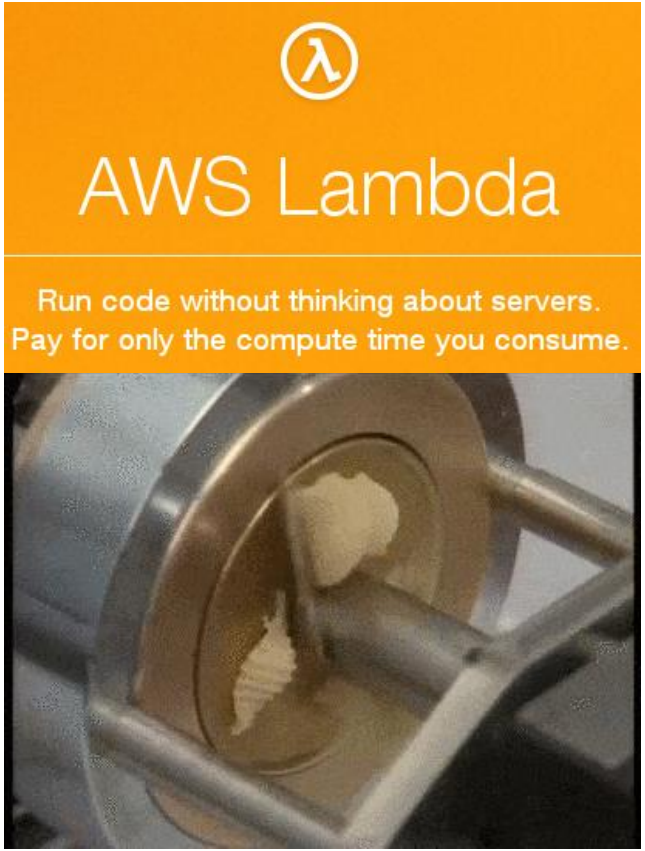
AWS Lambda

Jan 2015 - AWS Lambda Preview
Open to all AWS Customers

Upload your code and trigger it to run
Scales quickly
1,000,000 code runs for free every month

Only supported node.js

lambdash: AWS Lambda Shell Hack
By Eric Hammond
https://github.com/alestic/lambdash
Run shell commands using node.js



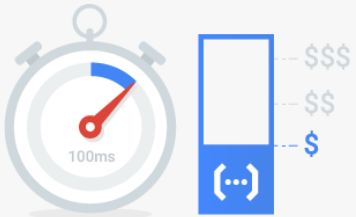
Servers are dead...

“Serverless”

Run Code, Not Servers - Serverless Computing
Ad aws.amazon.com/lambda ▾
Use AWS Lambda To Run Code Without Managing Servers. Get Started Today!
Continuous Scaling · Subsecond Metering · Real-Time Data Processing · No Servers to Manage
Product Details FAQs
One-Year Free Account Getting Started Guide

No Servers to Manage

AWS Lambda automatically runs your code without requiring you to provision or manage servers. Just write the code and upload it to Lambda.



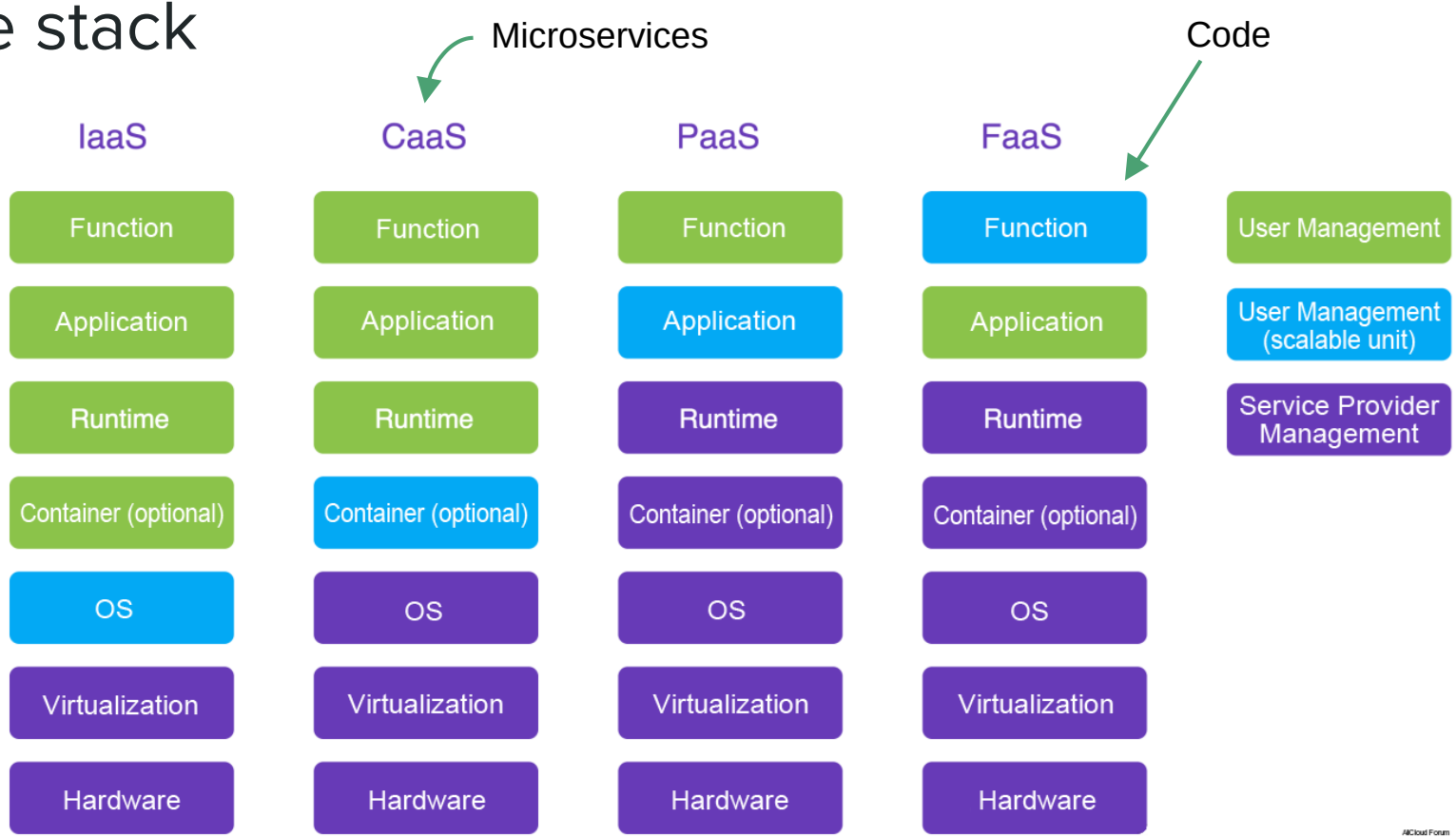
Serverless Economics

Cloud Functions are ephemeral, spinning up on-demand and back down in response to events in the environment. Pay only while your function is executing, metered to the nearest 100 milliseconds, and pay nothing after your function finishes.

Just Add Code

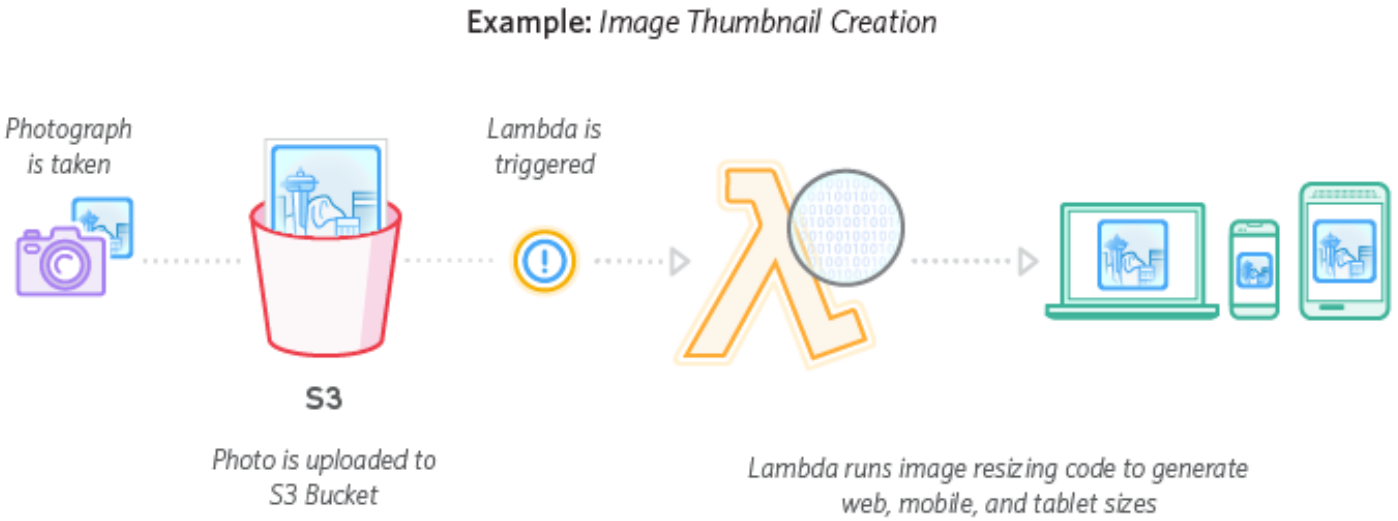
Run in a fully-managed, serverless environment where Google handles infrastructure, operating systems, and runtime environments completely on your behalf. Each Cloud Function runs in its own isolated secure execution context, scales automatically, and has a lifecycle independent from other functions.

The stack



Source: https://intl.aliyun.com/forum/read-499

Real-time File Processing



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

Scale

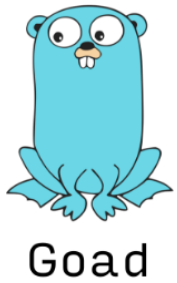
https://github.com/airbnb/streamalert
StreamAlert *is a serverless, realtime data analysis framework which empowers you to ingest, analyze, and alert on data from any environment, using datasources and alerting logic you define.*

https://github.com/0x4D31/honeyLambda
honeyλ *- a simple serverless application designed to create and monitor URL {honey}tokens, on top of AWS Lambda and Amazon API Gateway*

https://github.com/goadapp/goad
Goad *is an AWS Lambda powered, highly distributed, load testing tool built in Go*

https://github.com/davbo/lambda-csp-report-uri
Simple python application which runs on AWS Lambda and writes ***CSP*** *reports into S3 for later processing*

https://github.com/therefromhere/csp_lambda
AWS Lambda function to store ***Content Security Policy*** *reports in ElasticSearch*

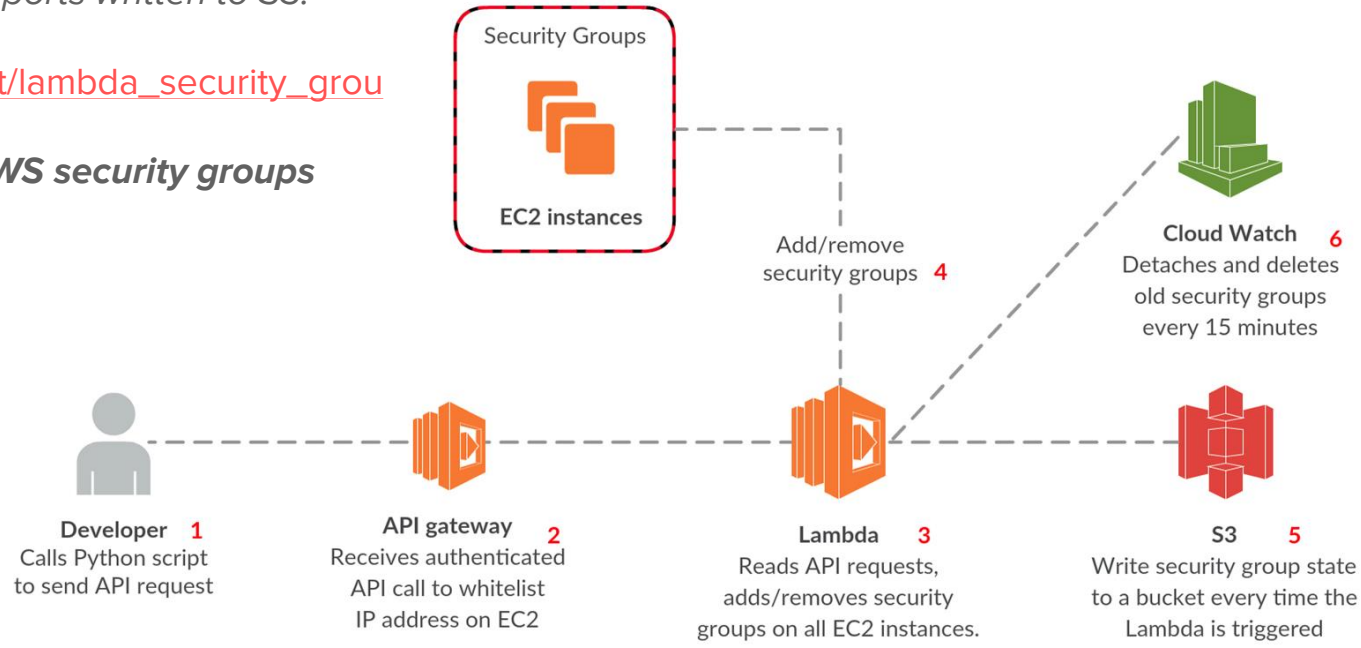


Automate

https://github.com/marekq/aws-lambda-firewall
*Create **temporary security groups on your EC2 instances** through a simple API call. In addition, audit your security groups easily by the use of automated reports written to S3.*

https://github.com/ilijamt/lambda_security_group_manager
*Auto **managing your AWS security groups** with Lambda*

https://github.com/johnmccuk/cloudflare-ip-security-group-update
*Lambda function to retrieve **Cloudflare's IP address** list and update the specified security group*

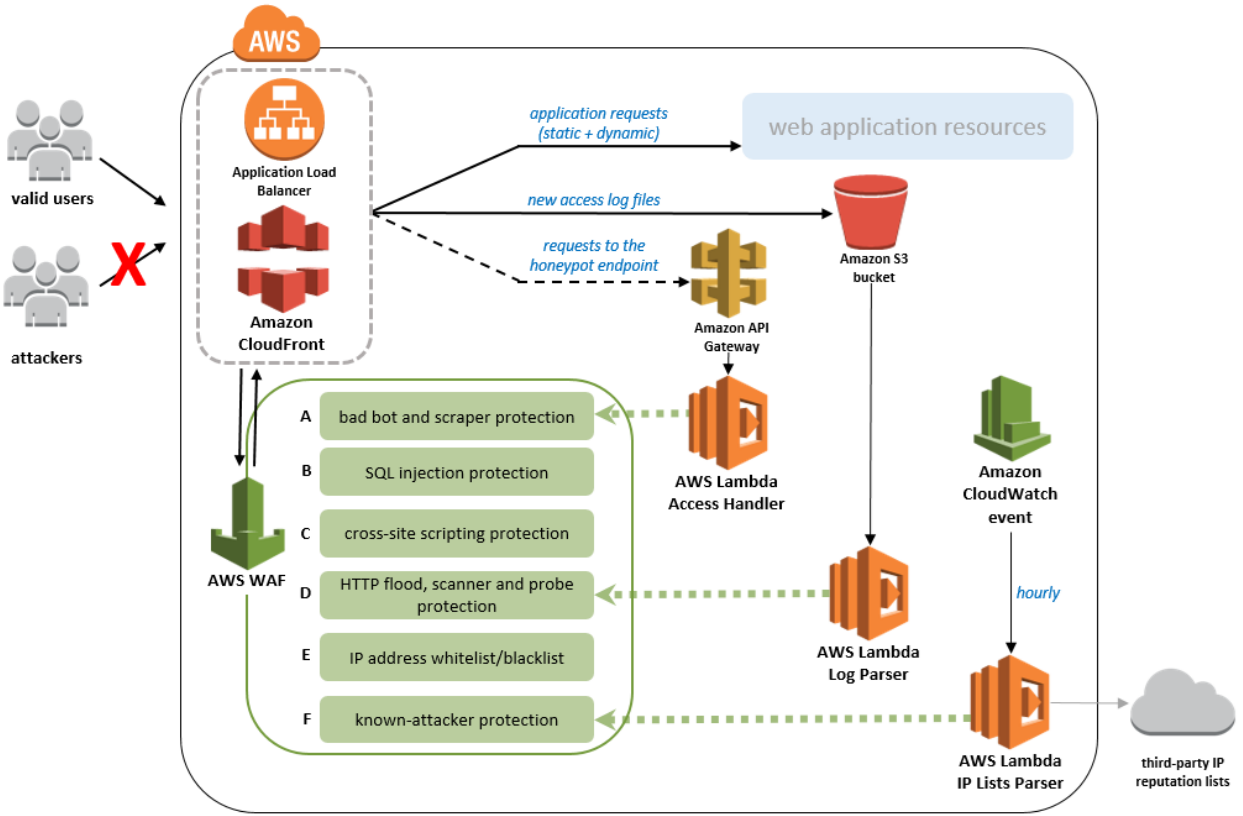


AWS WAF Automation

https://aws.amazon.com/answers/security/aws-waf-security-automations/

Parse application logs and trigger WAF rules

Honeypot
Log parsing (db scraping)
Use third party IP reputation lists

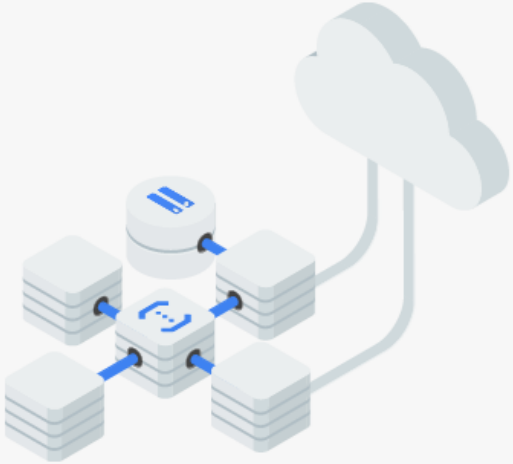


Hello World from the Serverless cloud

Google Cloud Platform

Serverless Applications on Google's Infrastructure

Cloud computing has made possible fully serverless models of computing where logic can be spun up on-demand in response to events originating from anywhere. Construct applications from bite-sized business logic billed to the nearest 100 milliseconds, only while your code is running. Serve users from zero to planet-scale, all without managing any infrastructure.

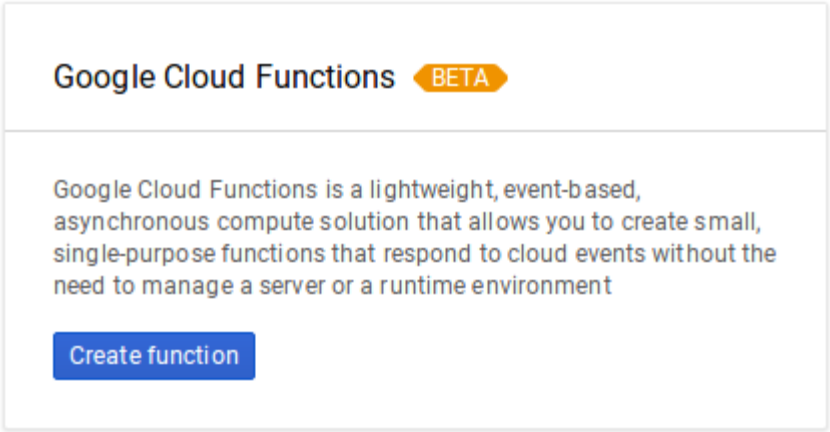


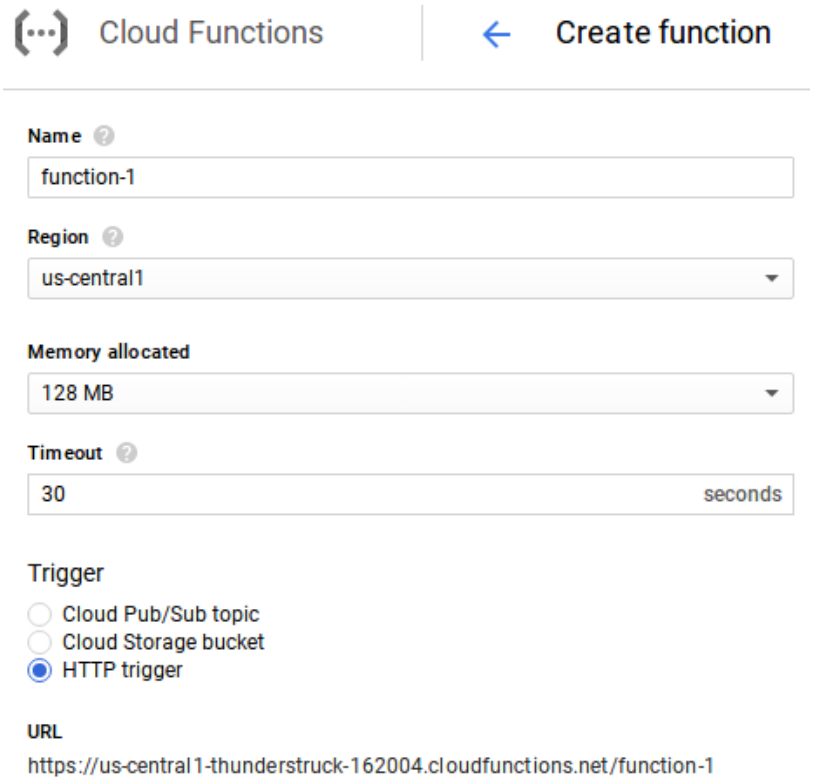
Hello Serverless World

Hello World on Google Cloud Functions (1/4)

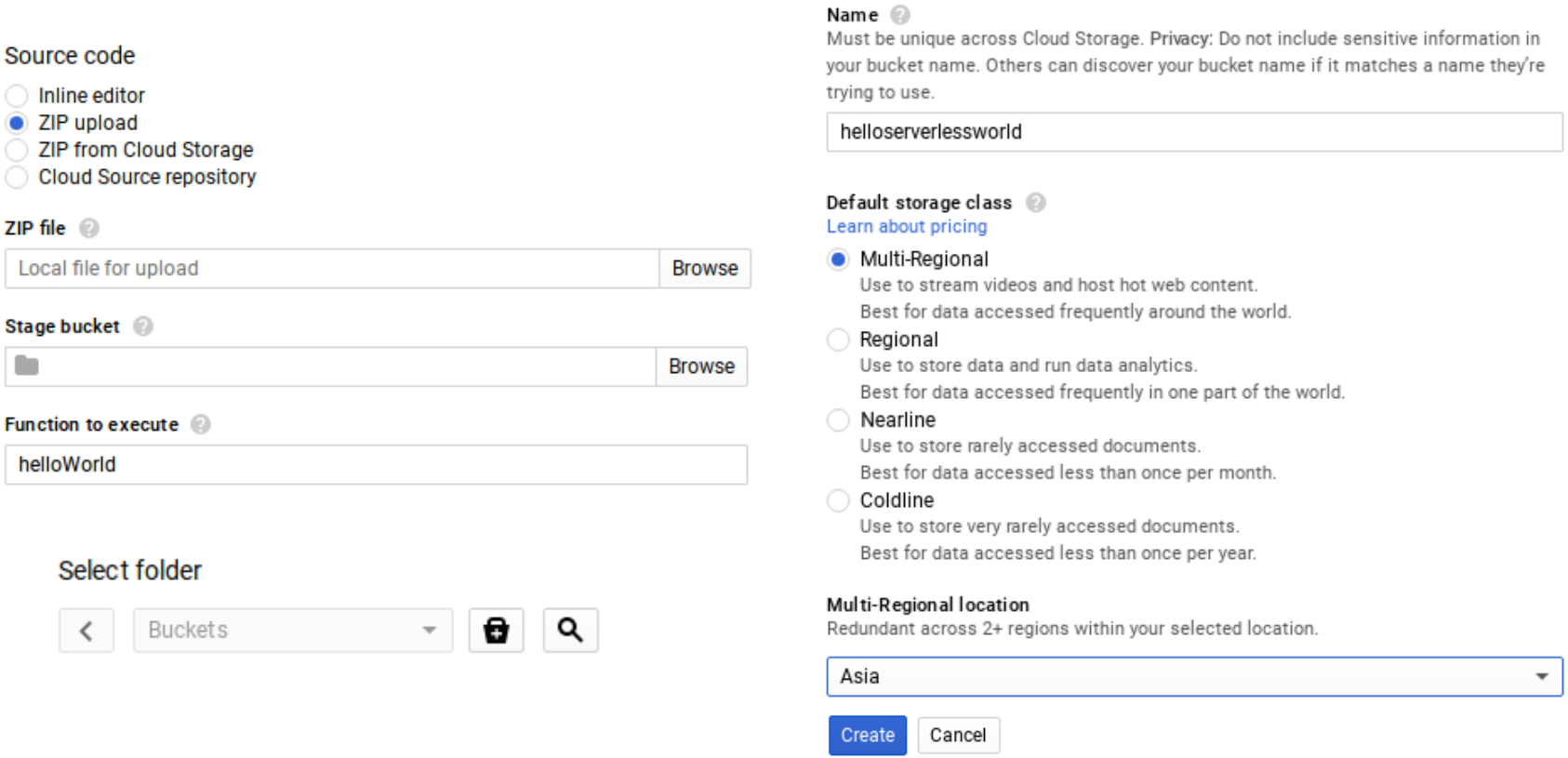
Support for Node.js

Run Python scripts from Node.js with simple (but efficient) inter-process communication through stdio
https://github.com/extrabacon/python-shell

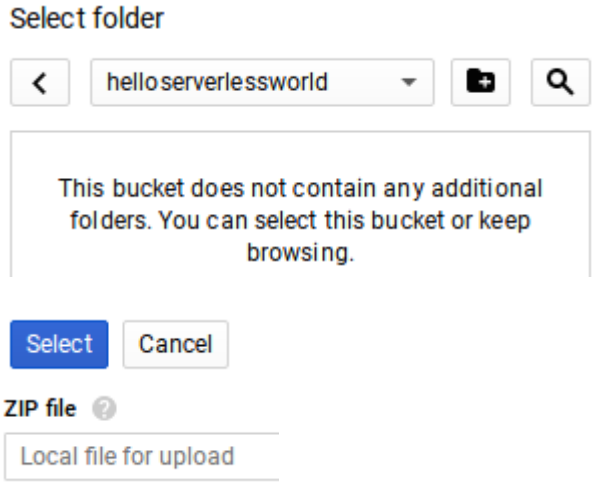




Hello World on Google Cloud Functions (2/4)



Hello World on Google Cloud Functions (3/4)




```
$ npm install --save python-shell
```

package.json

```
{
  "dependencies": {
    "python-shell": "0.0.1"
  }
}
```

```
$ mkdir python
$ touch python/worker.py
```

worker.py

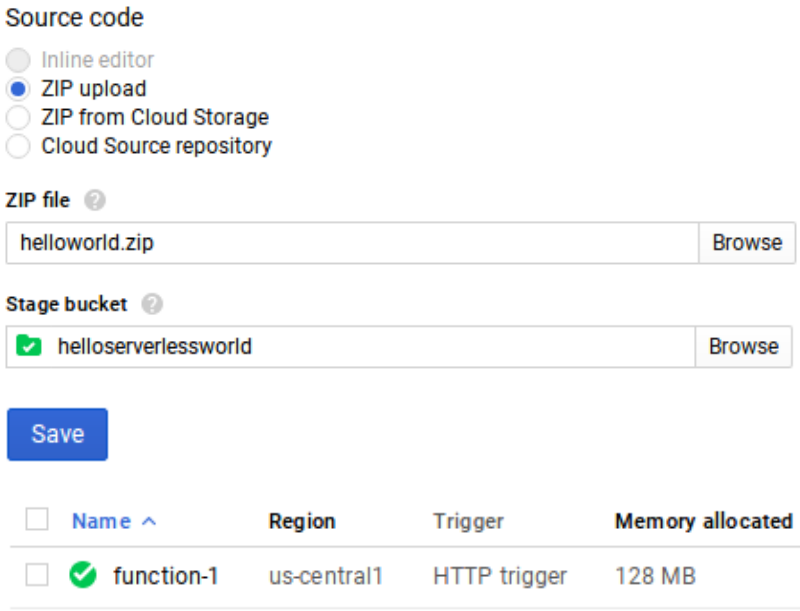
```
import urllib2
response = urllib2.urlopen('http://diagnostic.opendns.com/myip').read()
print 'Hello from google functions '+str(response)
```

index.js

```
const pythonshell = require('python-shell');
exports.helloWorld = function helloWorld(req, res) {
    pythonshell.run('worker.py', function (err, results) {
    if (err) throw err;
    res.status(200).send(results);
    });
};
```

```
$ zip helloworld.zip -r
python/ index.js
package.json
node_modules/
```

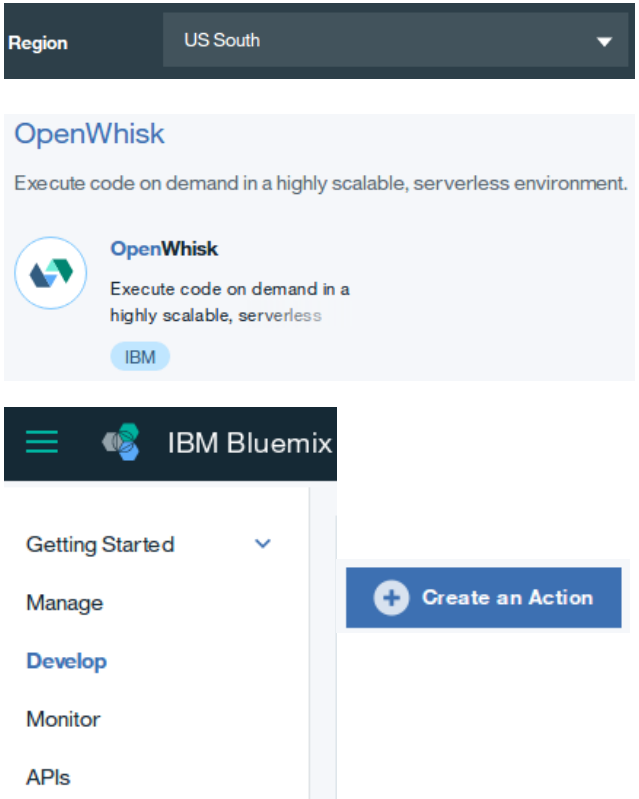
Hello World on Google Cloud Functions (4/4)

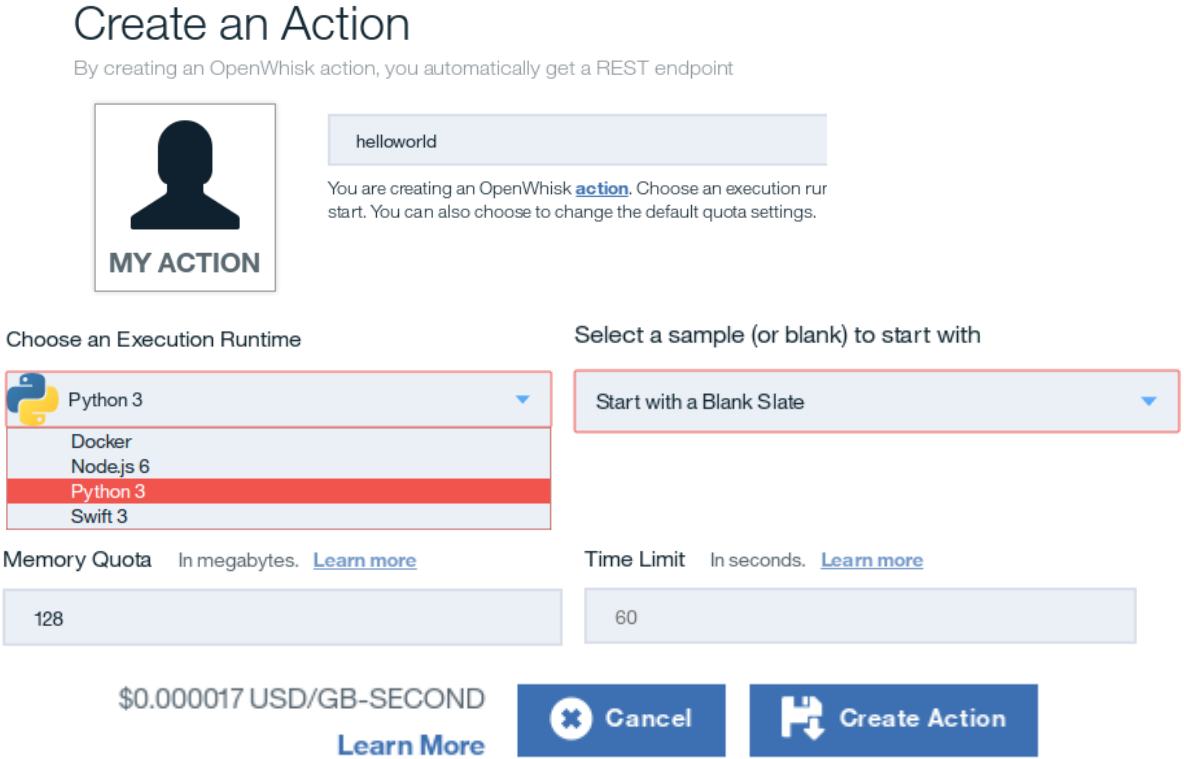


IP address is
2600:1900:2000:1c:400::10

Hello Serverless World

Hello World on IBM OpenWhisk (1/2)





Hello World on IBM OpenWhisk (2/2)

Python3

helloworld

```
1  import sys
2  from urllib.request import urlopen
3  def main(dict):
4      my_ip = urlopen('http://diagnostic.opendns.com/myip').read()
5      print("Hello from azure functions "+str(my_ip))
6      return {'ip': str(my_ip)}
```

Make It Live Run with this Value

Invocation Console

Invoked
helloworld

Invoked at
4:17:02 PM

Completed in
903ms

Billed for
100ms

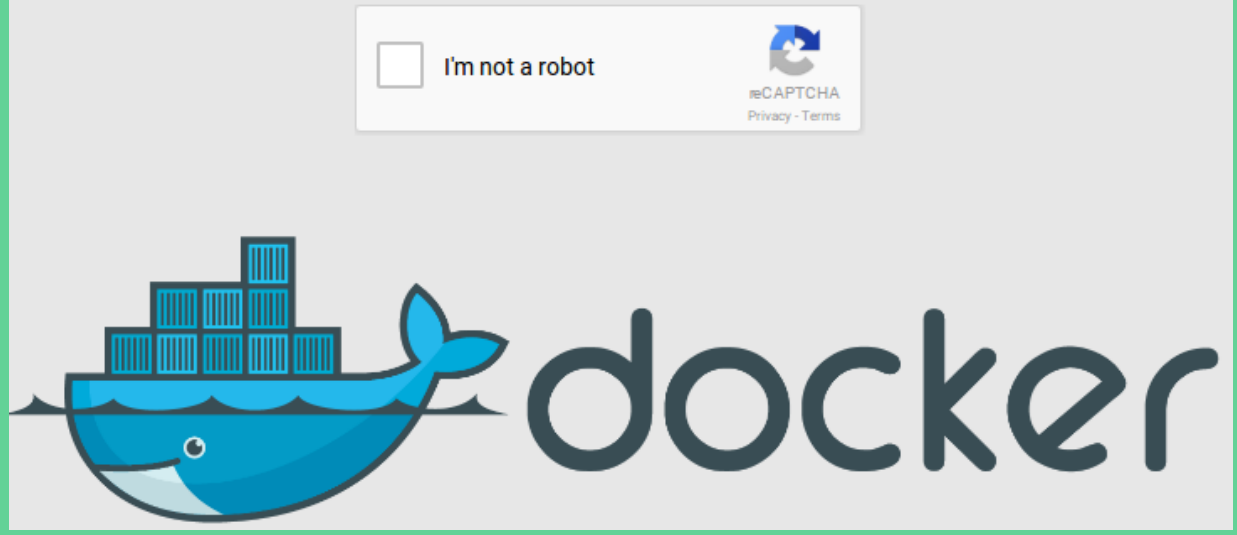
```
{
  "ip": "b'169.46.101.173'"
}
```

2017-06-24T08:17:03.26416996Z stdout: Hello from azure functions b'169.46.101.173'

IP address is
169.46.101.173

Welcome!

Before starting we need to verify you are a human

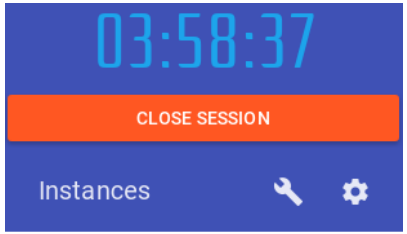


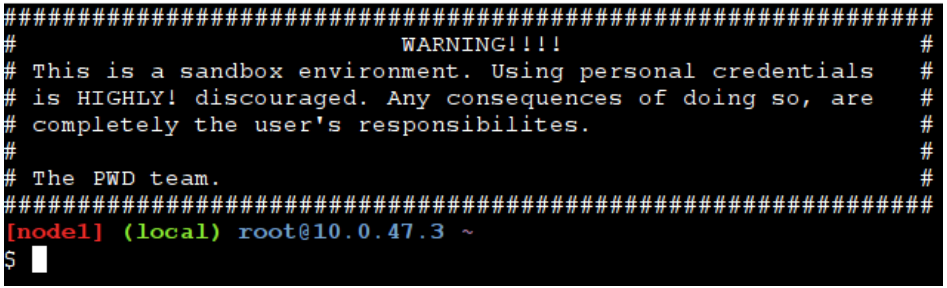
Hello Serverless World

Sessions and all their instances are deleted after 03:59:55 hours.

Hello World on Play with Docker

Hosted: http://www.play-with-docker.com/
Build your own: https://github.com/alexellis/faas
A serverless framework for Docker






```
$ curl ifconfig.co
34.206.199.2
```

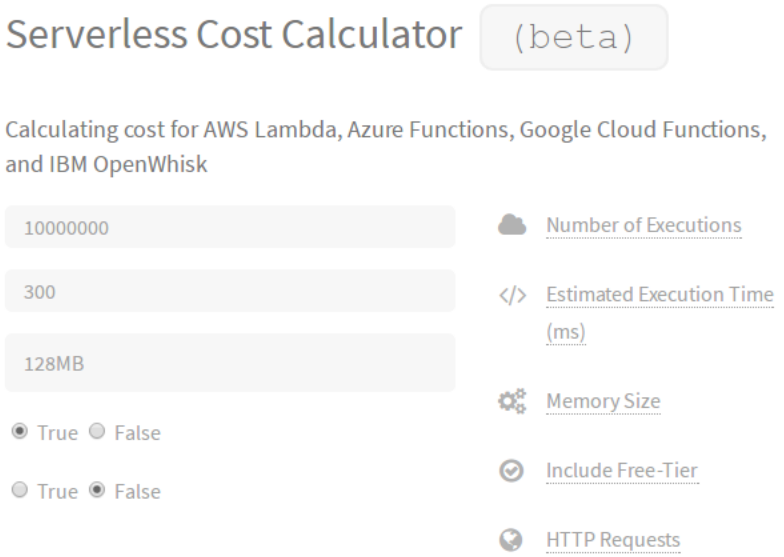
```
$ python -V
Python 2.7.13
```

$ dig +short -x 34.206.199.2
ec2-34-206-199-2.compute-1.amazonaws.com.

+Anonymous (no account)
-time limited
-captcha

Cost

Simple Hello World function to get current IP
http://serverlesscalc.com/



AWS: "**1M free requests per month and 400,000 GB-seconds of compute time per month**"

128 MB = 3,200,000 free seconds per month
Then $0.000000208 per 100ms

300ms - 10 million executions for $1.80

Vendor	Request Cost	Compute Cost	Total
AWS Lambda	$1.80	$0.00	$1.80
Azure Functions	$1.80	$0.00	$1.80
Google Cloud Functions	$3.20	$4.00	$7.20
IBM OpenWhisk	$0.00	$0.00	$0.00

FaaS support by region

AWS

1. US East (N. Virginia)
2. US East (Ohio)
3. US West (N. California)
4. US West (Oregon)
5. Canada (Central)
6. EU (Ireland)
7. EU (Frankfurt)
8. EU (London)
9. Asia Pacific (Singapore)
10. Asia Pacific (Sydney)
11. Asia Pacific (Seoul)
12. Asia Pacific (Tokyo)
13. Asia Pacific (Mumbai)
14. South America (São Paulo)

Azure

1. East US
2. East US 2
3. West US
4. West US 2
5. South Central US
6. North Central US
7. Central US
8. Canada Central
9. Canada East
10. North Europe
11. West Europe
12. UK West
13. UK South
14. Southeast Asia
15. East Asia
16. Japan West

Azure

17. Japan East
18. Brazil South
19. Australia East
20. Australia Southeast
22. Central India
23. South India

IBM

1. US South

Google

1. IOWA (us-central1)

Play with Docker

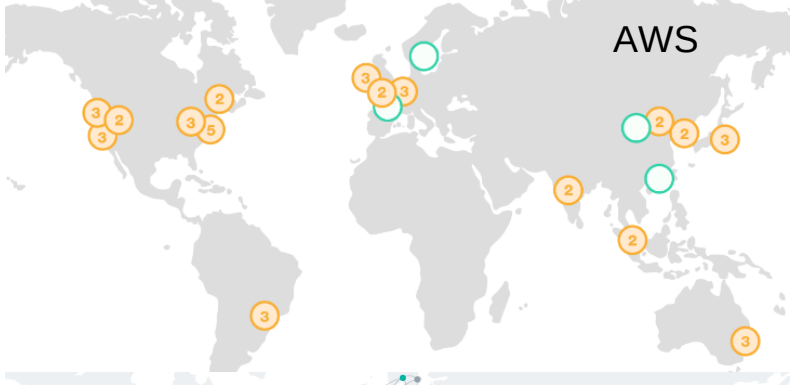
1. (AWS)

Overview

	Google	IBM	AWS	Azure
Regions	1	1	**14**	**23**
Language	Node.js (Python)	**Docker** Node.js 6 Python 3 Swift 3	Edge Node.js 4.3 Node.js 4.3 Node.js 6.10 Python 2.7 Python 3.6	Bash, Batch C#, F# JavaScript Php, PowerShell Python, TypeScript
OS (Python)	Linux Debian 8.8	Linux Ubuntu 14.04.1	Linux 4.4.51-40.60.amzn1.x86_64	**Windows Server 2012**
Network	**IPv6**	IPv4	IPv4	IPv4

Advantages

1. Low cost (“free”)
    a. Sign up credit
2. Unspecified source IP addresses
    a. Possibly low attribution
3. Global data centers
    a. China



Project Thunderstruck

Finding use cases for FaaS in offensive security

Project Thunderstruck

Finding use cases for FaaS in offensive security

Explore different cloud service providers

Try to get *supercomputer* resources without paying supercomputer prices

BSidesLV 2017

1. Searching in IPv6

Searching in IPv6
NOERROR but keep looking

Past work

You can -j REJECT but you can not hide: Global scanning of the IPv6 Internet
- Tobias Fiebig

December 2016
33C3: 33rd Chaos Communication Congress, Hamburg, Presenting first steps towards scanning IPv6 globally.

https://media.ccc.de/v/33c3-8061-you_can_-j_reject_but_you_can_not_hide_global_scanning_of_the_ipv6_internet

https://www.cs.ucsb.edu/~vigna/publications/2017_PAM_CollectingIPv6.pdf

https://gitlab.inet.tu-berlin.de/ptr6scan/toolchain

Finding v6 hosts by efficiently mapping ip6.arpa
- Peter van Dijk

March 2012
http://7bits.nl/blog/posts/finding-v6-hosts-by-efficiently-mapping-ip6-arpa

https://tools.ietf.org/html/rfc7707

Standard-Compliant Nameservers (DNS)

Something From Nothing (There): Collecting Global IPv6 Datasets From DNS

Tobias Fiebig, Kevin Borgolte, Shuang Hao, Christopher Kruegel, Giovanni Vigna

PAM (Passiv and Active Measurement) Conference 2017, Sydney

NOERROR

Keep looking

NXDOMAIN

Give up there's nothing here

“Complementary to prior approaches, van Dijk enumerates IPv6 reverse records… When correctly implementing **RFC1034**, as clarified in **RFC8020**, the Name Error response code (**NXDOMAIN** in practice) has the semantic of **there is nothing here or anywhere thereunder in the name tree**.”

“Combined with the IPv6 PTR DNS tree, where each subzone has 16 (0-f, one for each IPv6 nibble) children up to a depth of 32 levels, provides the possibility to **exploit standard-compliant nameservers to enumerate the zone”**

Let’s dig over here

```
$ dig
1.0.0.0.0.0.0.0.0.0.2.0.1.0.0.2.ip6.arpa.
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status:
NOERROR, id: 18347
;; flags: qr rd ra; QUERY: 1, ANSWER: 0,
AUTHORITY: 0, ADDITIONAL: 1
```

There are records to be found within:

1.0.0.0.0.0.0.0.0.0.2.0.1.0.0.2.ip6.arpa.

2001:0200:0000:0001:...

```
$ dig
5.0.0.0.0.0.0.0.0.0.2.0.1.0.0.2.ip6.arpa.
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status:
NXDOMAIN, id: 58924
;; flags: qr rd ra; QUERY: 1, ANSWER: 0,
AUTHORITY: 0, ADDITIONAL: 1
```

Nothing to be found within:

5.0.0.0.0.0.0.0.0.0.2.0.1.0.0.2.ip6.arpa.

2001:0200:0000:0005:...

Super Computer

Tobias Fiebig, University computer used for Machine Learning

My Measurement Setup

- 80 Thread HP DL580g7
- 512GB Ram
- 2TB Fast storage
- 1gE Network to DFN
- Local resolver (unbound) on the machine

(I bet that people being better with code can run this on their notebook ;-))

Results and Limitations

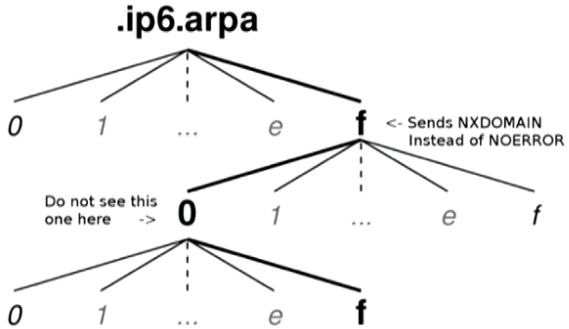
1st Attempt
Breadth first search
1 week
70M+ records

Received complaint email from large ISP
Dynamic reverse zones

2nd Attempt
Detect and ignore auto-gen zones
3 days (65.53 hours)
1.6M records

Non RFC8020 Compliance

- Servers that sent NXDOMAIN instead of NOERROR for non-existing nodes with children:



Not all DNS servers reply with NOERROR

Seeding

Non RFC8020 Compliance

- Solution: Seeding, using an aggregated BGP view
- Algorithm:
  - In: 2001:1:2:3::/64 as seed from BGP
  - Out:

```
3.0.0.0.2.0.0.0.1.0.0.0.1.0.0.2.ip6.arpa.
        2.0.0.0.1.0.0.0.1.0.0.2.ip6.arpa.
                1.0.0.0.1.0.0.2.ip6.arpa.
                        1.0.0.2.ip6.arpa.
```

- Other possible sources:
  - Czyz et al.'s PTRv4 -> AAAA lookups
  - All v6 Datasets you can get your hands on!

Scale to Fail

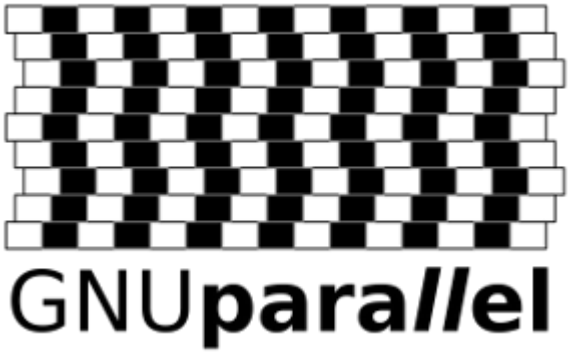
3rd Attempt
70.5 hours
80 threads
5.3M records

4th Attempt
22.2 hours
400 threads
2.2M records <--- :'(

"Lesson: use more than one IP address"
"**Server ran out of sockets**"

Good use case for FaaS?

Bash script
- curl/wget routing data
- BGPdump
- sort --parallel | uniq
- parallel cook_down.py
- parallel extract_terminals.py



The Plan

Question to Tobias Fiebig during 33c3:

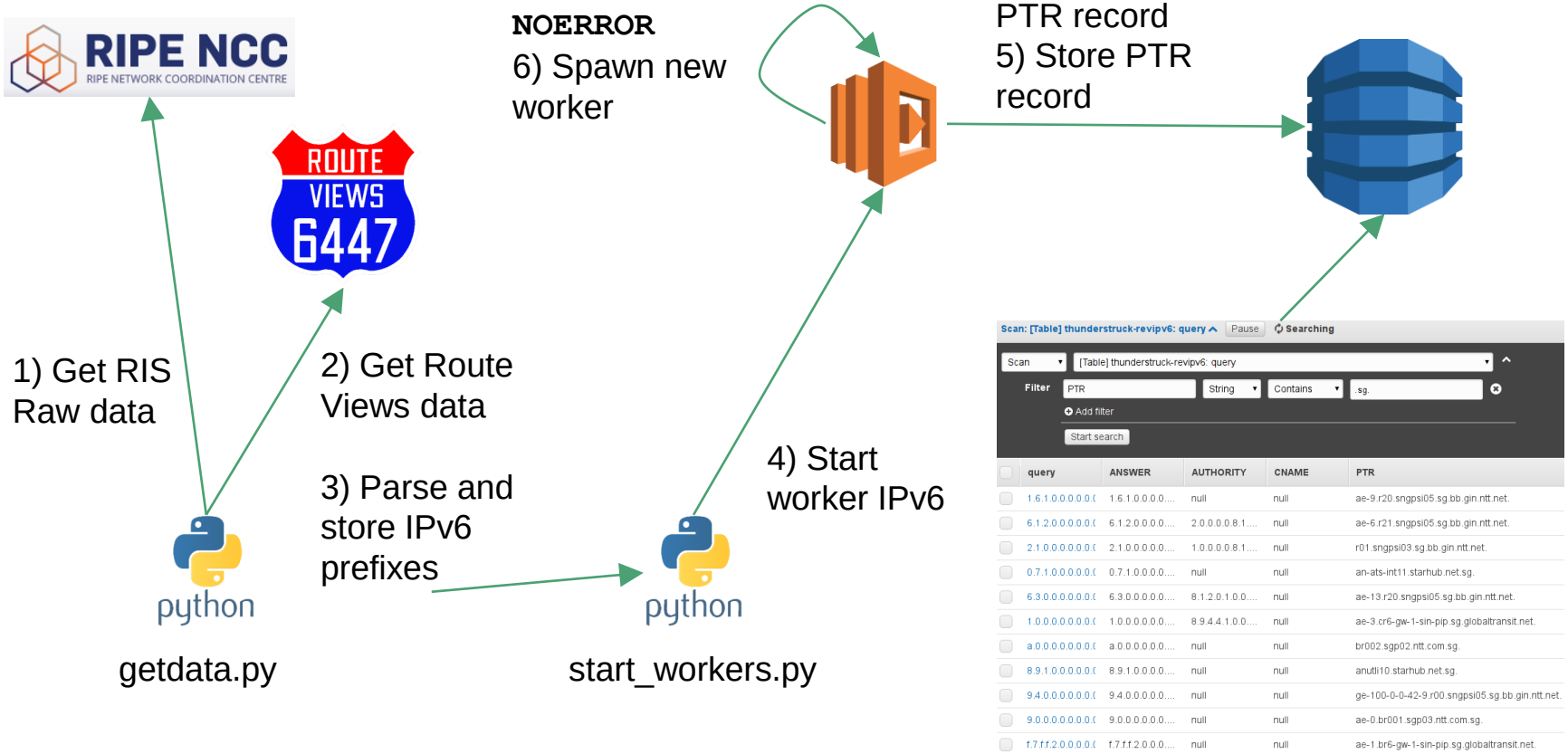
Q: “Where is the dataset?”
A: “**It’s actually stored in a distributed manner on a lot of DNS servers**”

DynamoDB

AWS Lambda

1. Download public routing information (BGP) for advertised IPv6 networks
2. Parse and create a list of starting IPv6 addresses
3. Run worker to enumerate each starting address
4. Write any discovered PTR records to DynamoDB
5. Look for interesting stuff in DynamoDB web interface

Architecture

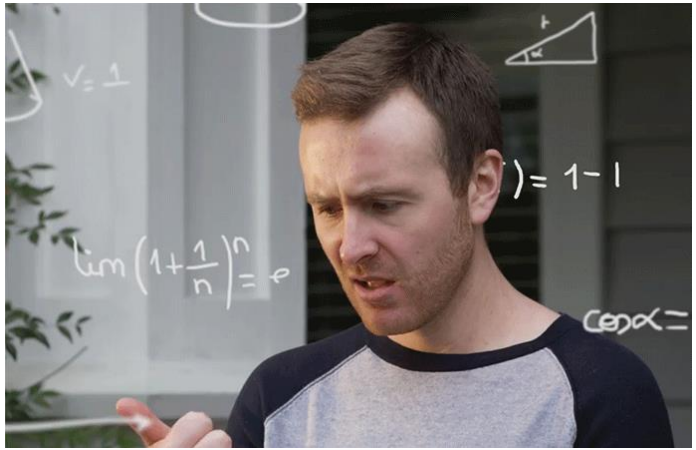


Start the workers!

$ ./getdata.py
$ ./start_workers.py
Need to spawn 55548.0 workers to lookup
55548 with 1 ipv6 address per worker
02:59:47.796904 - starting revipv6
02:59:47.865480 - starting workers
03:05:36.023268 - done starting workers
finished starting workers in 0:0**5:48.157793**

Provisioned read capacity units	10 (Auto Scaling Disabled)
Provisioned write capacity units	100 (Auto Scaling Disabled)
Last decrease time	-
Last increase time	July 14, 2017 at 8:46:35 AM UTC+8
Storage size (in bytes)	114.82 MB
Item count	250,679

250k+ records found

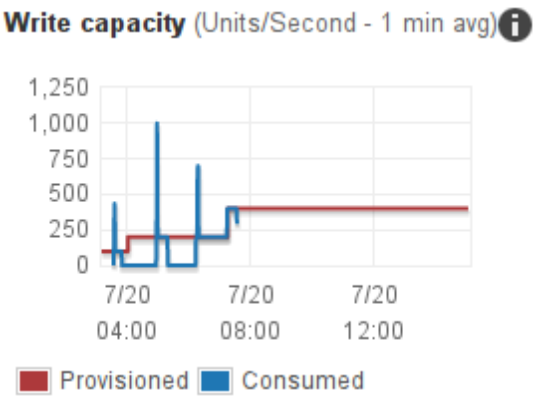


That was fast, but where are all the records?

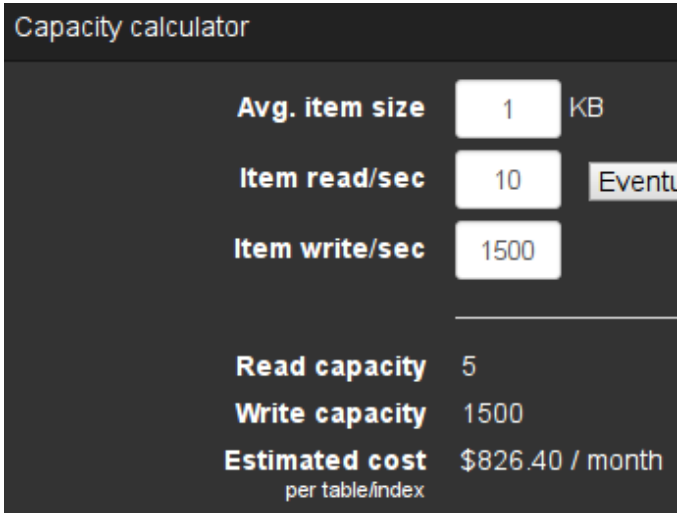
Learning to scale

CloudWatch logs:

```
An error occurred
(ProvisionedThroughputExceededExcep
tion) when calling the PutItem
operation (reached max retries: 9):
The level of configured provisioned
throughput for the table was
exceeded.
```



Too many concurrent attempts to write to DynamoDB



Need to scale the DynamoDB storage backend USD$ 826/month!

Improvements

Remove the recursive function call
- Scales too quickly for too little work

Create depth search limit
- Only look 4, 8, 16, 24, 32 addresses deep from each starting point

Start workers slowly
- Don’t try to look up all possible reverse DNS entries for every IPv6 address in under 5 minutes

Scale up DynamoDB when needed
Scale down when idle

DynamoDB web interface is slow and clunky at search (“scan”)

Let’s look at some results

Scan: [Table] thunderstruck-revipv6: query ^ Pause Searching

Scan | [Table] thunderstruck-revipv6: query

Filter | PTR | String | Contains | .sg.

Add filter

Start search

query	ANSWER	AUTHORITY	CNAME	PTR
1.6.1.0.0.0.0.0.(	1.6.1.0.0.0.0....	null	null	ae-9.r20.sngpsi05.sg.bb.gin.ntt.net.
6.1.2.0.0.0.0.0.(	6.1.2.0.0.0.0....	2.0.0.0.8.1....	null	ae-6.r21.sngpsi05.sg.bb.gin.ntt.net.
2.1.0.0.0.0.0.0.(	2.1.0.0.0.0.0....	1.0.0.0.8.1....	null	r01.sngpsi03.sg.bb.gin.ntt.net.
0.7.1.0.0.0.0.0.(	0.7.1.0.0.0.0....	null	null	an-ats-int11.starhub.net.sg.
6.3.0.0.0.0.0.0.(	6.3.0.0.0.0.0....	8.1.2.0.1.0.0....	null	ae-13.r20.sngpsi05.sg.bb.gin.ntt.net.
1.0.0.0.0.0.0.0.(	1.0.0.0.0.0.0....	8.9.4.4.1.0.0....	null	ae-3.cr6-gw-1-sin-pip.sg.globaltransit.net.
a.0.0.0.0.0.0.0.(	a.0.0.0.0.0.0....	null	null	br002.sgp02.ntt.com.sg.
8.9.1.0.0.0.0.0.(	8.9.1.0.0.0.0....	null	null	anutli10.starhub.net.sg.
9.4.0.0.0.0.0.0.(	9.4.0.0.0.0.0....	null	null	ge-100-0-0-42-9.r00.sngpsi05.sg.bb.gin.ntt.net.
9.0.0.0.0.0.0.0.(	9.0.0.0.0.0.0....	null	null	ae-0.br001.sgp03.ntt.com.sg.
f.7.f.f.2.0.0.0.(	f.7.f.f.2.0.0.0....	null	null	ae-1.br6-gw-1-sin-pip.sg.globaltransit.net.

KVM

Intelligent Platform Management Interface (IPMI)
ipmi.k7-a14-lt1.**kvm**.serveriai.**lt**
ash-ipmi.bootc.**eu**
esx3-ipmi.sgef.**hu**
ipmi.origo.nmugroup.**se**
meter5-test-ipmi.meter.cesnet.**cz**
d**fw**1-ipmi.psg.com
vm0-ipmi.iad.rg.net
hippo-ipmi.**nist.gov**
monitoring-ipmi.dataix.ru

Integrated Lights-Out (iLO)
ilo.app9.mia2.afilias-nst.info
vrrp.ilo.cpr1.lax1.afilias-nst.info
vm11-ilo-gs1-gcn.tgsa.co.uk
tur-**esxfw1-ilo**.massey.ac.nz

OOB
gate-2-oob.belval.restena.**lu**
oob.core.**nrc.ca**
interface-3--hoc1-m15-01-hnd.oob.sfdc.net
oob-dc2.ksrv.**fr**
xrc1-oob.1d12.xs4.mendix.net
internal-240.**fw-oob**.workbone.pironet-ndh.com
eth0.**oob-pix**.egh.uk.mdnx.com

Integrated Dell Remote Access Controller (iDRAC)
insphl-idrac.spawar**.navy.mil**
djs-sandiego-proto-idrac.**mgmt.dren.mil**
idrac-vmhost6.copernica.nl
perez-drac.**lusem.edu**
drac1.softlab.ece.**ntua.gr**
pc19-drac.otemachi.wide.ad.jp

Gov

.mil

djs-sandiego-proto-**idrac**.mgmt.dren.mil

fw.sd.spawar.navy.mil

sentinel.**tni-au.mil.id**
blackbox.tni-au.mil.id
incognito.tni-au.mil.id

www-**largedata-proj**.cmf.nrl.navy.mil

remote.mda.mil
stafflan.svcs.dren.mil

.gov

sec1sgw1.internet.gov.sa
gcca-new-transit-core-gcc.fnal.gov

VRRPv6.s3.bnl.gov

ir2-n1--vlan501.**hsrp**.lbl.gov

bluecoat-sg-cadev-3.ca.sandia.gov

Infrastructure

EQIX-EMA-SV-IPV6.softbanktelecom.com
EQIX-EMA-NY-IPV6.bankofnymellon.com
EQIX-EMA-**NY**-IPV6.**centrilogic.com**
EQIX-EMA-**LA**-IPV6.**cloudflare.com**
EQIX-EMA-**DA**-IPV6.**oraclecorporation.com**
EQIX-EMA-**SV**-IPV6.**evernote.com**
EQIX-EMA-**NY**-IPV6-**HSRP-gateway**.ny.equinix.com
EQIX-V6-MA-dc.**box.com**
EQIX-MA-**CH**-IPV6.**twitter.com**
EQIX-MA-**SV**-IPV6.**vmware.com**
EQIX-MA-**DA2**-IPV6.**NETFLIX.com**
lo0.gw102.me1.ap.equinix.com
xe-0-0-0.gw603.**hk3**.ap.equinix.com
xe-0-0-0.gw101.**ty4**.ap.equinix.com
xe-0-1-0.gw401.**sg1**.ap.equinix.com

loop0.gw01.stjh.nf.aliant.net
loop0.cmts01-cres.mql.ncable.net.au
loop0.ubr10k.omnitel.biz
loop0.**core1**.camphill.dcsi.net.au
loop0.pling.v6.ngdc.net
loop0.23w.ba07.sydn.ns.bellaliant.net
loop0.**hkg**-mgi-**mgmt**-1.megaport.com
loop0.**sin**-sg2-**mgmt**-1.megaport.com

backbone-v6-r23.ictvalleumbra.it

Quantum Computing

NASA
dwave-**fw**.nas.nasa.gov
dwave-**pp2**.nas.nasa.gov
dwave-**qubist**.nas.nasa.gov
dwave-**monitor**.nas.nasa.gov
dwave-**qc**.nas.nasa.gov

heliophysicsdata.sci.gsfc.nasa.gov
rtr-s28-40g-v6.sci.gsfc.nasa.gov
oceanpost.sci.gsfc.nasa.gov
voyager.sci.gsfc.nasa.gov

Summary

Received zero abuse emails

Lessons learned:

Recursive functions that call themself quickly multiple beyond control
- Limit recursive calls

Always make sure your backend data storage will scale
- Elasticsearch is expensive to scale and likely can't handle the workload

Going further

More data sources for seeding
https://github.com/trustedsec/hardcidr

Public datasets, Certificate transparency, Forward DNS dataset, DNS, scans.io
https://github.com/appsecco/bugcrowd-levelup-subdomain-enumeration

Create workflow with DynamoDB Triggers to run another Lambda function (portscan, find matching ipv4, etc)

AWS Lambda free tier is per region
- Reduce cost even further by allocating workload to regions with free compute

Further work

Interesting talks

33c3 2016
Gone in 60 Milliseconds
Intrusion and Exfiltration in Server-less Architectures
Rich Jones

Blackhat US 2016
Account Jumping Post Infection Persistency & Lateral Movement In AWS
Dan Amiga & Dor Knafo

Blackhat US 2017
Hacking Serverless Runtimes: Profiling AWS Lambda Azure Functions and more
Wednesday, July 26 - 1:30pm - 2:20pm
Andrew Krug & Graham Jones

DEF CON 25
Starting the Avalanche: Application DoS In Microservice Architectures
Friday, July 28 - 13:00 in Track 3
Scott Behrens, Jeremy Heffner

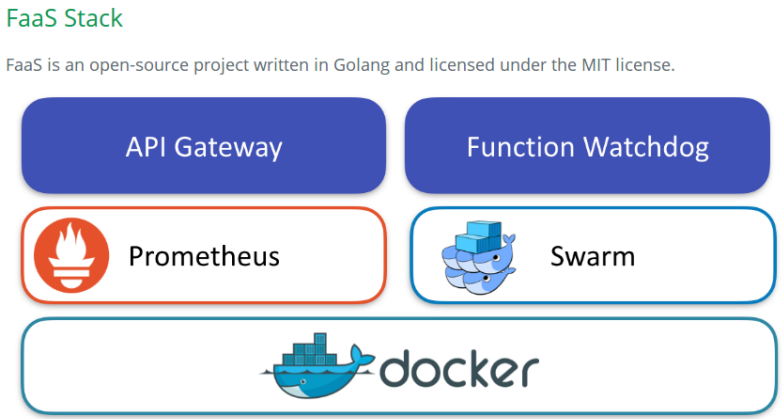
DEF CON 25
Microservices and FaaS for Offensive Security
Saturday July 29 - 11:00 in 101 Track
1. DDoS without Servers
2. SMS OTP Brute Force

Going further

AWS Lambda - High mem: 1536 MB
266,667/seconds/month free

Aliyun / Alibaba Cloud - China
Need to register with +86 mobile number

IBM OpenWhisk
Docker



Build your own FaaS infrastructure
https://github.com/alexellis/faas
UI portal
Setup with one script
Any process that can run in Docker can be a serverless function
Prometheus metrics and logging
Auto-scales as demand increases

github.com/ryanbaxendale/thunderstruck-demo