Building Your First Security App in the Cloud

powered by aws

Thank You!

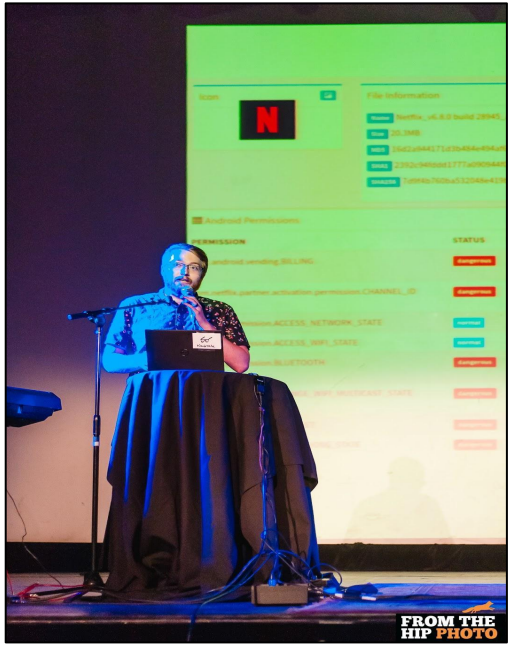
BSides OK

Conference Sponsors

SUGAR SECURITY

Logan Evans @Sugar_Security

Our Cloud Hacks Your Cloud (if you want)



SUGAR SECURITY

Who is this talk for?

If you like cybersecurity…

And you want to try making something in the cloud

- honeypots
- password crackers

This is for you!

SUGAR SECURITY

Post in Chat!

Do you use the cloud?

What do you do in the cloud? How do you do it?

Tech? (EC2, Kubernetes, Lambda?)

Questions (win a book!)

Agenda - 25 minutes

→ **Primer:** AWS Serverless Cloud 101

→ **Getting Practical:** “Hello {you}” API in the cloud

→ **Lab:** Building a Serverless Security Toolkit

→ ***Instant OSINT***: *https://{api}.com/osint/{example.com}*

powered by aws

We won’t be covering:

Non-AWS Clouds

- Google Cloud 💔, Azure, Oracle, Kubernetes

Development Best Practices

- Well-Architected Framework

There is always more to learn!

Building Software: The Non-Cloud Way

Local Software: .exe / scripts

Web application / API:

- App Server (Apache, PHP, ASP.NET)
- Database Server (MySQL, MSSQL)
- Build Server?
- Log Server?

Networking, Patching, Hardware...

Cloud VMs

Largely the same… (released in 2006)

AWS Services Edit Vedang Shah Oregon

Launch Instance Connect Actions

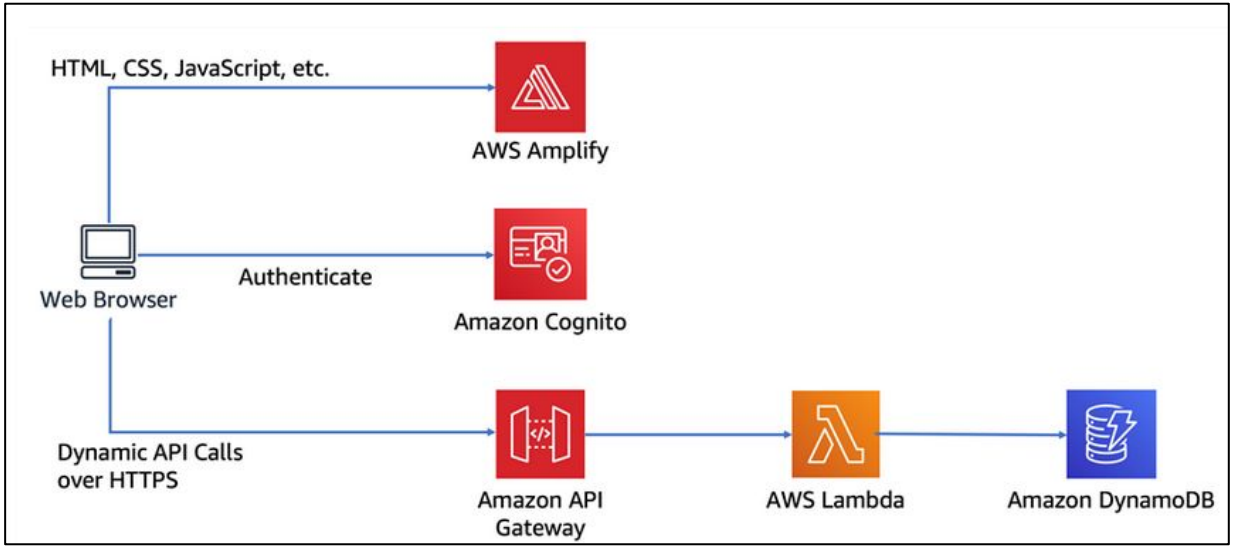
Filter by tags and attributes or search by keyword 1 to 6 of 6

Name	Instance ID	Instance	Availability	Instance State	Status	Alarr	Public DNS	Public IP
	i-234803d5	t2.micro	us-west-2b	running	2/...	...	ec2-52-27-...	52.27.52.89
	i-620f8095	t2.micro	us-west-2a	running	2/...	...	ec2-52-27-...	52.27.47....
	i-214902d7	t2.micro	us-west-2b	running	2/...	...	ec2-52-27-...	52.27.53.66
	i-f8dd9af4	t2.micro	us-west-2b	running	2/...	...	ec2-52-27-...	52.27.52.90
	i-e7befaeb	t2.micro	us-west-2b	running	2/...	...	ec2-52-27-...	52.27.52.96
DDCLB	i-50bb37a7	t2.micro	us-west-2a	running	2/...	...	ec2-52-24-...	52.24.105...

Cloud 2.0: Serverless (c. 2014ish)

(yes, there are still servers)

- **Easier to maintain** (no patching, networking, etc)
- **Cheaper** (only pay for what you use)



Pictured: Serverless Web Application

What can you do **serverless**ly?

Store Files: S3 (Simple Storage Service)

Run Code (*Python, JS, Go, C#*): Lambda

Host an API: https://api.sugarsecurity.com

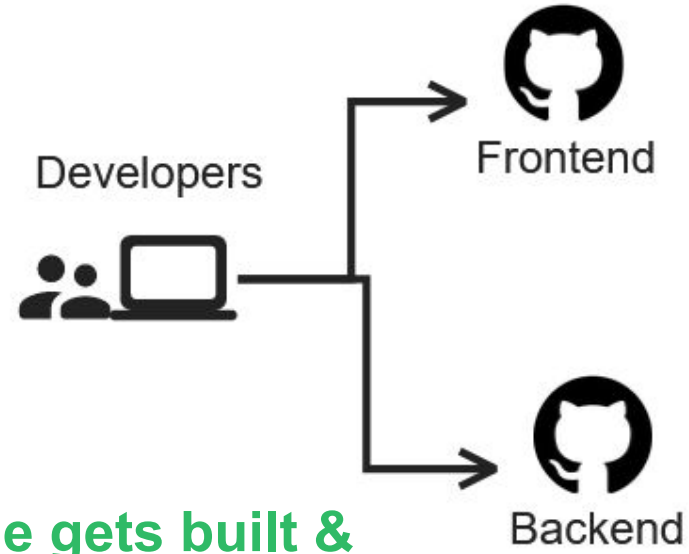
Host a Database: RDS, DynamoDB, Redshift

Authenticate Users: AWS Cognito / IAM

and a lot more!

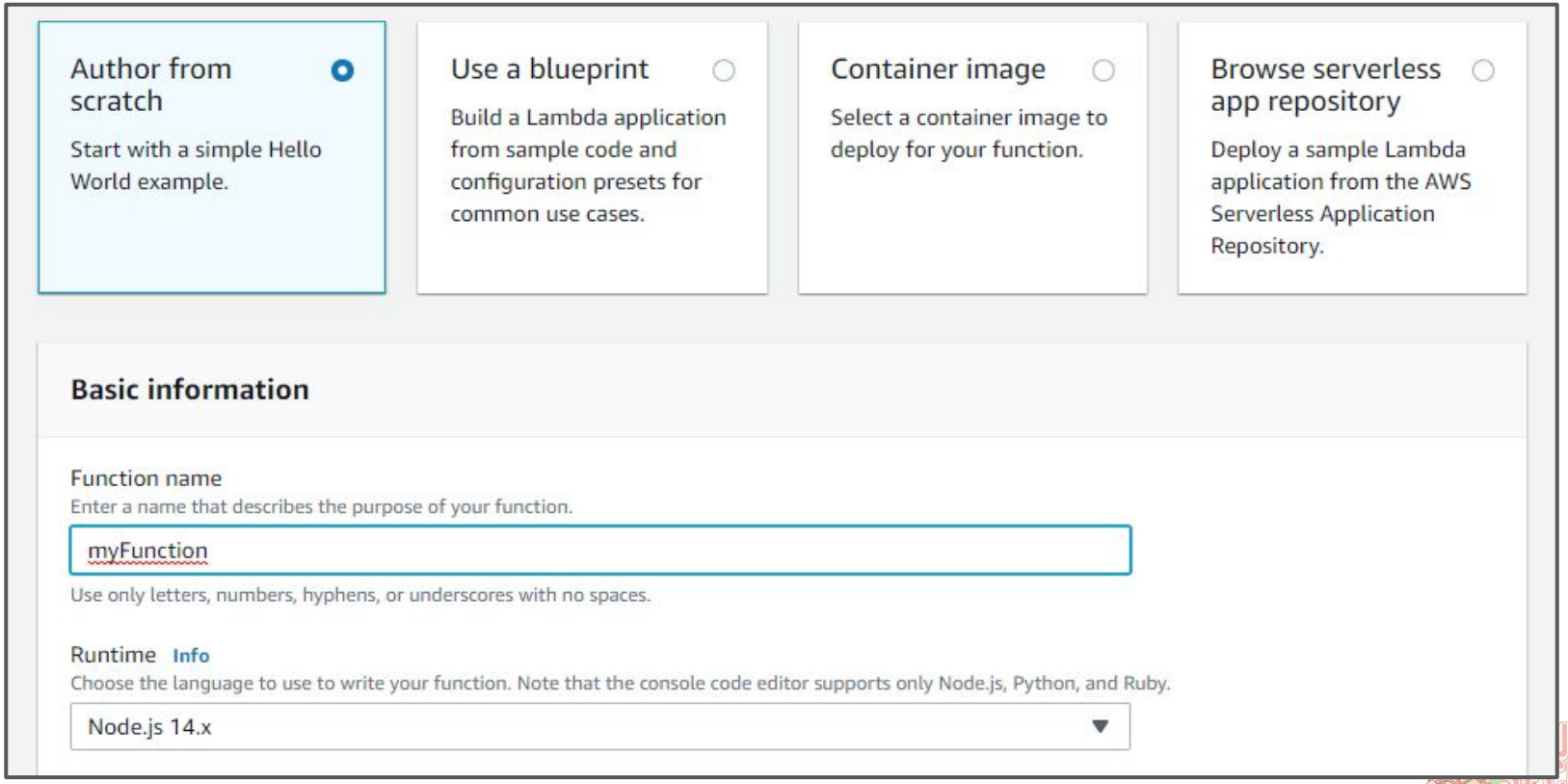
SUGAR SECURITY

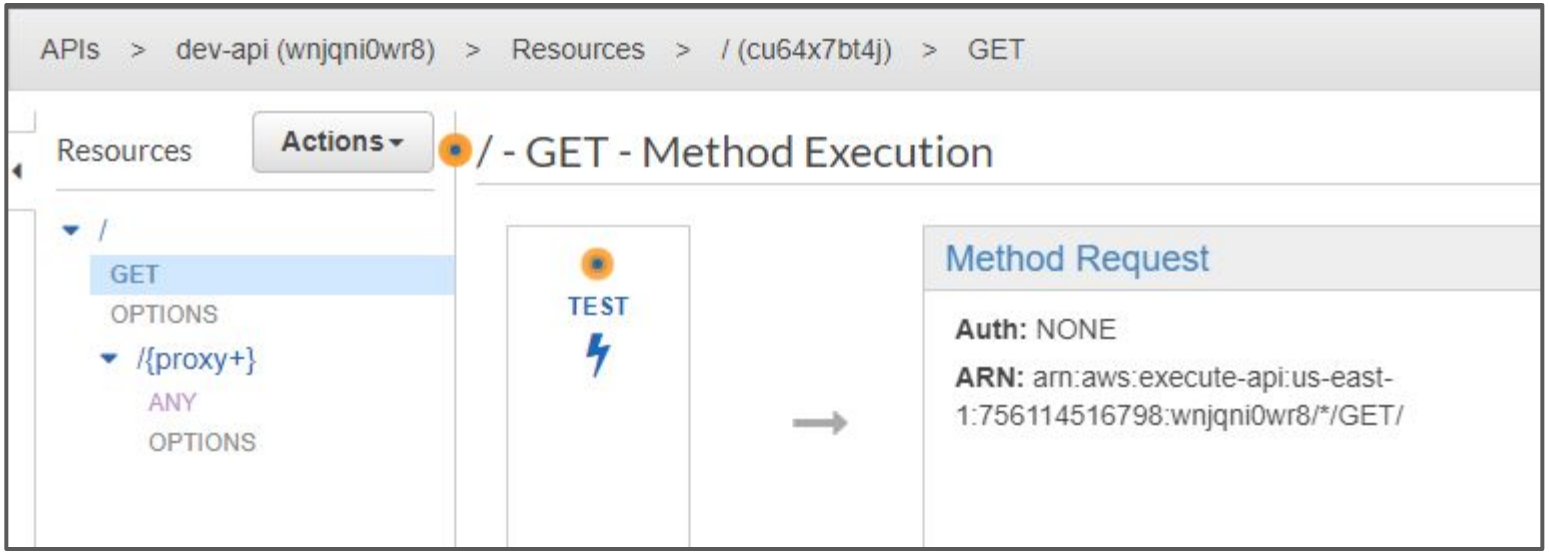
Dev Perspective: Cloud 2.0 / Serverless



Code gets built & deployed on an AWS server

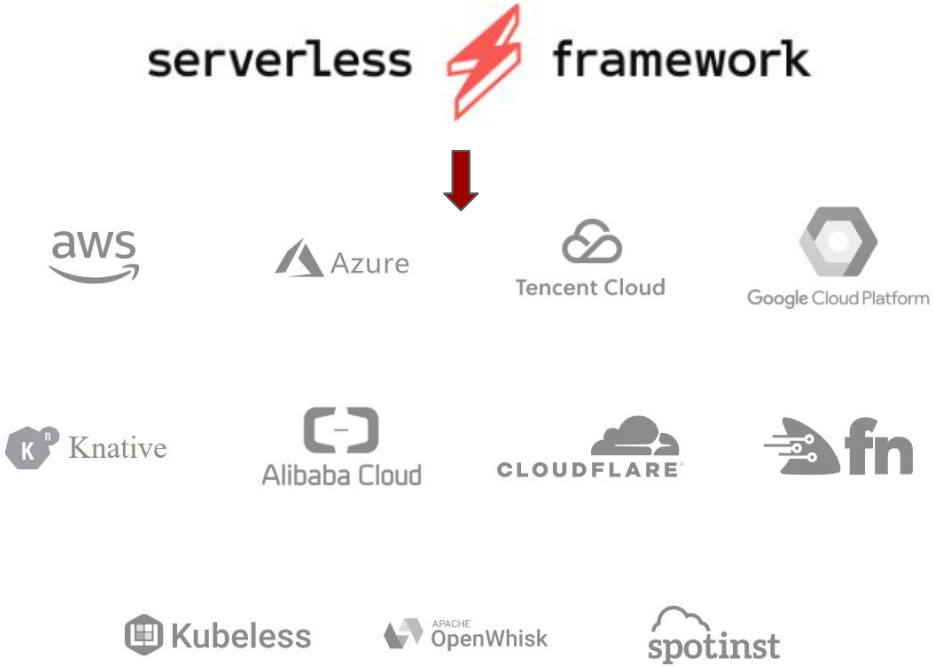
Doing it manually sucks




APIs > dev-api (wnjqni0wr8) > Resources > / (cu64x7bt4j) > GET
Resources
Actions
/ - GET - Method Execution
/
GET
OPTIONS
/{proxy+}
ANY
OPTIONS
TEST
Method Request
Auth: NONE
ARN: arn:aws:execute-api:us-east-1:756114516798:wnjqni0wr8/*/GET/

This is where frameworks come in

Turns hours of manual setup into a few dozen lines of code



Serverless Framework

Helps someone with minimal AWS and Coding knowledge get started

Functions:

- Code that does stuff (add user, create record, run job)

Events:

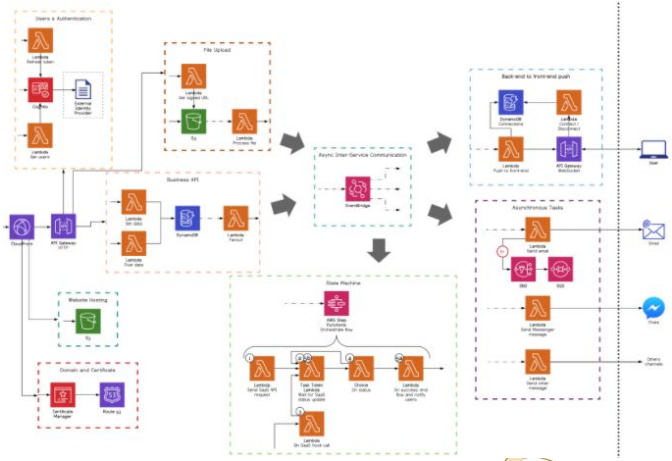
- Triggers a function ^ (user added, record created, job done)

Resources:

- Other AWS services (S3 Buckets, DB Tables, Robots)

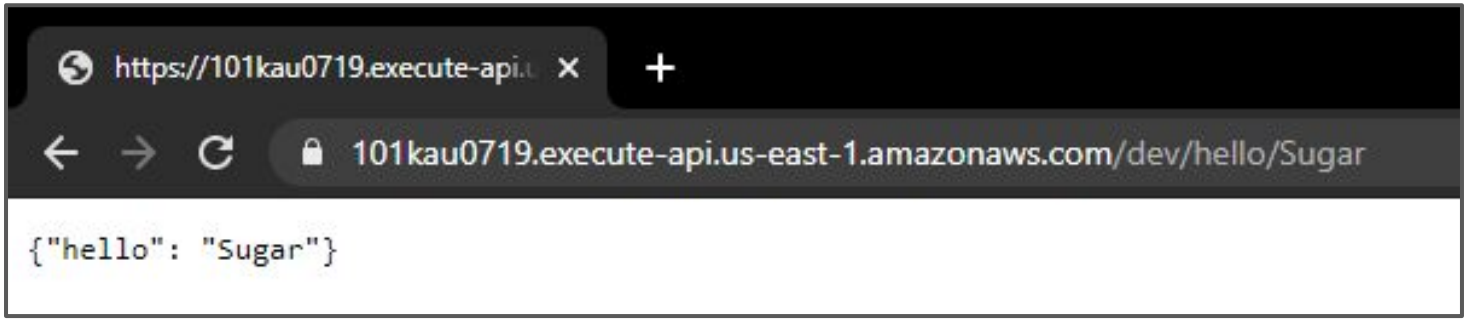
AWS events

- api gateway
- http api
- websocket
- kinesis & dynamodb
- s3
- schedule
- sns
- sqs
- application load
- alexa skill
- alexa smart home
- iot
- cloudwatch event
- cloudwatch log
- eventbridge event
- cloudfront
- cognito user pool
- iot fleet provisioning
- kafka
- msk



SUGAR SECURITY

Lab 1 - Hello {you}!



https://your-api.com/**hello/bsides**

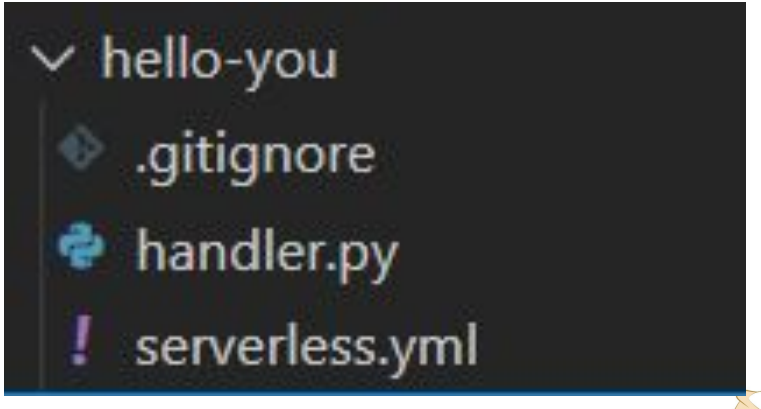
hello("bsides")

Python / Java / etc.

https://github.com/SugarSecurity/bsides-ok-2021

Start a Serverless Project


```
`serverless create
--template aws-python3
--path hello-you`
```



SUGAR SECURITY

serverless.yml

```yaml
service: hello-you
frameworkVersion: '2'

provider:
  name: aws
  runtime: python3.8
  lambdaHashingVersion: 20201221

functions:
  hello:
    handler: handler.hello
```

hello()

handler.py
hello()

hello(“anyone?”)

hello()

AWS events

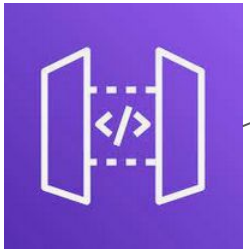
- api gateway
- http api
- websocket
- kinesis & dynamodb
- s3
- schedule
- sns
- sqs
- application load
- alexa skill
- alexa smart home
- iot
- cloudwatch event
- cloudwatch log
- eventbridge event
- cloudfront
- cognito user pool
- iot fleet provisioning
- kafka
- msk

SUGAR SECURITY

serverless.yml

Lambda hello(name)

API GET /hello/{name}



```yaml
functions:
  hello:
    handler: handler.hello
    events:
      - http:
          path: hello/{name}
          method: get
```

handler.py

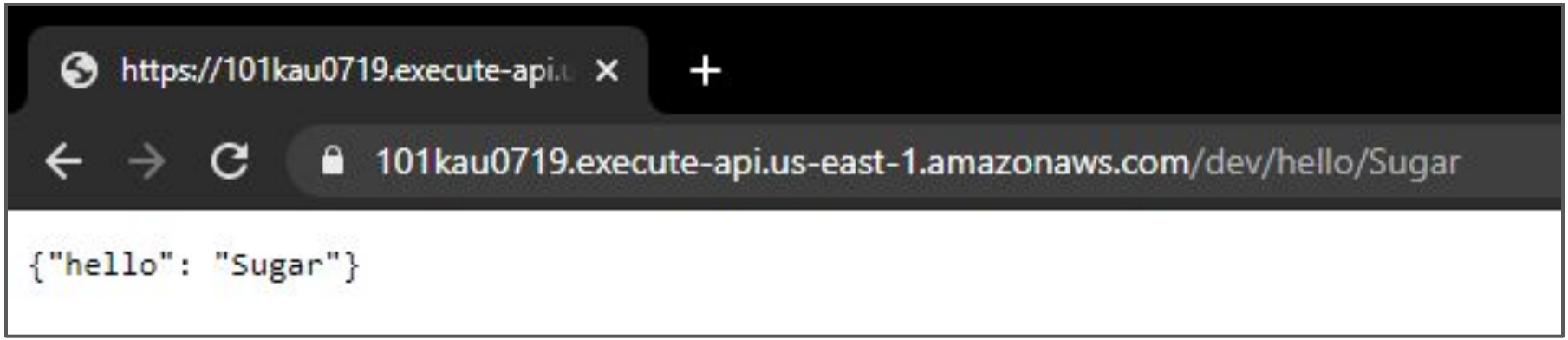
Modify python to take {name} from the event

```
name = event['pathParameters']['name']

http_response = {
    "statusCode”: 200,
    “body”: “{‘hello’: name}”
}

return http_response
```

Deploying to the Cloud

```
serverless deploy
```

```
Serverless: Stack update finished...
Service Information
service: hello-you
stage: dev
region: us-east-1
stack: hello-you-dev
resources: 12
api keys:
  None
endpoints:
  GET - https://101kau0719.execute-api.us-east-1.amazonaws.com/dev/hello/{name}
functions:
  hello: hello-you-dev-hello
layers:
  None
```



https://github.com/SugarSecurity/bsides-ok-2021

What does this have to do with hacker stuff?

You could:

- Write cybersecurity tools
  - Honeypot, password cracker, etc...
- WITHOUT maintaining a server / VM
  - or a Kubernetes cluster

HACKER TOOLKIT - OSINT DEMO

Open **S**ource **Int**elligence

- *https://<your-api>/osint/{example.com}*

serverless framework

get_osint({domain}) returns:

- WHOIS info
- Subdomains

requirements.txt
serverless.yml
toolkit.py

SUGAR SECURITY

serverless.yml

```yaml
functions:
  get_osint:
    handler: toolkit.get_osint
    events:
      - http:
          path: osint/{domain}
          method: get
```

get_osint(domain)

toolkit.py

/osint/example.com

OSINT Toolkit

input: {domain} e.g. sugarsecurity.com

output:

- **subdomains:** HackerTarget search sugarsecurity.com
- **whois:** ARIN.net search sugarsecurity.com

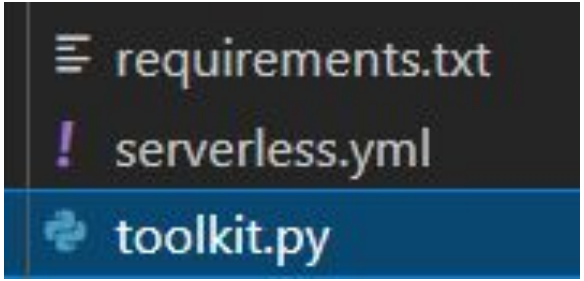
toolkit.py

get_osint()

```
import json
import whois
import requests
```

...

toolkit.py




```
def get_osint(event, context):
    domain = event['pathParameters']['domain']
    whois_results = whois.whois(domain)
    subdomain_results = requests.get(f"{hackertarget}?q={domain}")
    http_response = {
        "statusCode": 200,
        "body": {
                "whois": whois_results,
                "subdomains": subdomain_results
            }
    }
    return http_response
```

https://github.com/SugarSecurity/bsides-ok-2021

Get Subdomains in Python

Begin by importing the Requests module:

```
>>> import requests
```

Now, let's try to get a webpage. For this example, let's get GitHub's public timeline:

```
>>> r = requests.get(https://api.hackertarget.com/hostsearch/?q=example.com
```

thanks hackertarget.com!

Whois in Python

```
>>> import whois
>>> w = whois.whois('webscraping.com')
```

python-whois 0.7.3

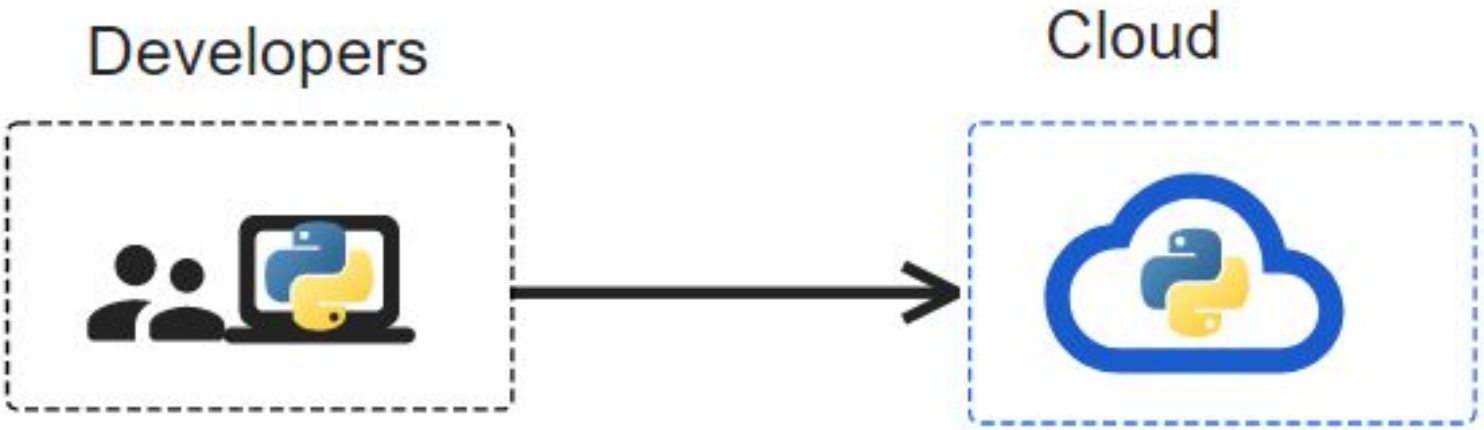
```
pip install python-whois
```

requirements.txt U

02_hacker-toolkit > requirements.txt

```
1  python-whois
2  requests
```

Cloud Problem: Local vs Cloud Environment



`serverless plugin install -n serverless-python-requirements`

“serverless deploy”

```
endpoints:
  GET - https://2ojqa6olff.execute-api.us-east-1.amazonaws.com/dev/osint/{domain}
functions:
  get_osint: hacker-toolkit-dev-get_osint
```

SUGAR SECURITY

OSINT Dashboard Result

2ojqa6olff.execute-api.us-east-1.an

https://2ojqa6olff.execute-api.us-east-1.amazonaws.com/dev/osint/sugarsecurity.com

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON


```
▼ whois:
  ▶ domain_name:      […]
    registrar:        "Amazon Registrar, Inc."
    whois_server:     "whois.registrar.amazon.com"
    referral_url:     null
  ▶ updated_date:     […]
    state:            "WA"
    zipcode:          "98108-1226"
    country:          "US"
  subdomains:         "sugarsecurity.com,99.84.219.89\n"
```

fp89w0uiml.execute-api.us-east-1.a

https://fp89w0uiml.execute-api.us-east-1.amazonaws.com/dev/osint/mcdonalds.com

JSON Raw Data Headers

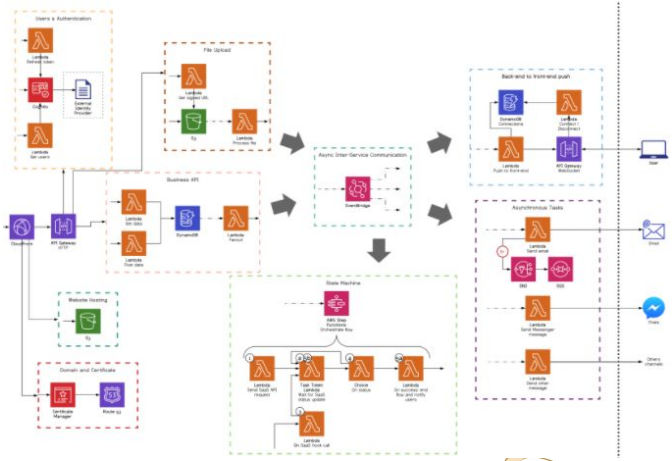
Save Copy Collapse All Expand All Filter JSON


```
▼ whois:
  ▼ domain_name:
      0:          "MCDONALDS.COM"
      1:          "mcdonalds.com"
    registrar:    "CSC CORPORATE DOMAINS, INC."

▼ subdomains:     "author1.mcdonalds.com,66.111.188.47\nwebtrends02.mcdonalds.com,164.109.144.105\nwww.dna.mcdonalds.com,216.255.64.202\nr207.b
                  \nmcjob.mcdonalds.com,66.252.76.91\nr210.c.mcdonalds.com,172.82.218.210\nr211.c.mcdonalds.com,172.82.218.211\nr212.c.mcdonalc
                  \nr213.c.mcdonalds.com,172.82.218.213\nr204.c.mcdonalds.com,172.82.218.204\nr214.c.mcdonalds.com,172.82.218.214\nr205.c.mcdor
                  \nr206.c.mcdonalds.com,172.82.218.206\nr216.c.mcdonalds.com,172.82.218.216\nr207.c.mcdonalds.com,172.82.218.207\nr217.c.mcdor
                  \nr218.c.mcdonalds.com,172.82.218.218\nr209.c.mcdonalds.com,172.82.218.209\nmcworld.mcdonalds.com,208.193.67.135\ne.mcdonalds
                  \nr53.e.mcdonalds.com,192.243.225.53\nr54.e.mcdonalds.com,192.243.225.54\nr55.e.mcdonalds.com,192.243.225.55\nr56.e.mcdonalds
                  \nr58.e.mcdonalds.com,192.243.225.58\nr59.e.mcdonalds.com,192.243.225.59\ndep-stag.mcdonalds.com,216.255.65.195\nstaging.mcdc
                  a12.staging.mcdonalds.com,216.255.66.225\na13.staging.mcdonalds.com,216.255.66.226\nwww3.staging.mcdonalds.com,216.255.65.214
                  \nwmmadmin.mcdonalds.com,52.22.31.17\nuknutritionadmin.mcdonalds.com,52.22.31.17\nfun.mcdonalds.com,52.6.111.61\nsvn.mcdonalc
                  \navatar.mcdonalds.com,204.232.232.140\nbrandcenter.mcdonalds.com,216.255.66.164\ntest-intlfr.mcdonalds.com,52.6.111.61\nauth
                  \nwww.investor.mcdonalds.com,52.6.111.61\nfeelgoodawards.mcdonalds.com,67.225.129.91\nbestpractices.mcdonalds.com,216.255.66.
                  \nbeta.happymealchefs.mcdonalds.com,74.39.191.247\ncsr.blogs.mcdonalds.com,69.46.111.201\nwww.blackhawks.mcdonalds.com,74.39.
                  \nr69.alerts.mcdonalds.com,192.243.225.69\nm2.development.mcdonalds.com,216.255.67.42\nwcapp.dev.mcdonalds.com,216.255.65.203
```


What Next?

AWS events

- api gateway
- http api
- websocket
- kinesis & dynamodb
- s3
- schedule
- sns
- sqs
- application load
- alexa skill
- alexa smart home
- iot
- cloudwatch event
- cloudwatch log
- eventbridge event
- cloudfront
- cognito user pool
- iot fleet provisioning
- kafka
- msk



SUGAR SECURITY

More Services == More Possibilities

ECS / EKS / Fargate:

- Containers (Docker) for Long-Running Jobs
  - (honeypots, password crackers)

Futuristic Stuff:

- AI / ML / Blockchain
- Satellites, Robots, Quantum

Serverless CTF

https://app.cloud-logon.com/dev/calculator

sugar security unhackable calculator

what would you like us to calculate?

800/(41*170)

result: 0.114

Tell BSides you love Sugar Security!

email me logan.evans@sugarsecurity.com

Conclusion

Everybody gets something different out of these talks!

No AWS background? Learned some cloud stuff

Some AWS / Serverless background? Learned practical applications

Already using Serverless Framework? Talk to me l8r please