# HERRAMIENTAS SERVERLESS PARA PENTESTING


Expositor:

Mgtr. José Moreno

Docente

UNIVERSIDAD TECNOLÓGICA DE PANAMÁ
FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

# WHOAMI

- Docente Universidad Tecnológica de Panamá (FISC)
- Miembro Owasp Panamá
- Squad Member de DevSecOps Latam – Panamá
- Embajador Comunidad Dojo Panamá
- Miembro Fundador APPIF
- Cisco Certified CyberOps Associate
- HTB player: jam620
- Articulos de interes:

https://cutt.ly/GdmhAVT

- Twitter:

@utp_team

# TEMAS

1. Historia
   - ¿Qué es eso de serverless?
   - Definición
2. Arquitectura Serverless
   - ¿Cuándo usar arquitectura Serverless?
   - Ventajas y Desventajas
3. Ejemplos
4. Modelos Clouds
5. Soluciones Disponibles
6. Demostración
7. Conclusión
8. ¿Preguntas?

# ¿QUÉ ESPERAR?

1. Comprender como implementar serverless en las fases del pentesting
2. Técnicas de ataques utilizadas comúnmente en privesc utilizando serverless
3. Ventajas de utilizar serverless para pentesting

# HISTORIA

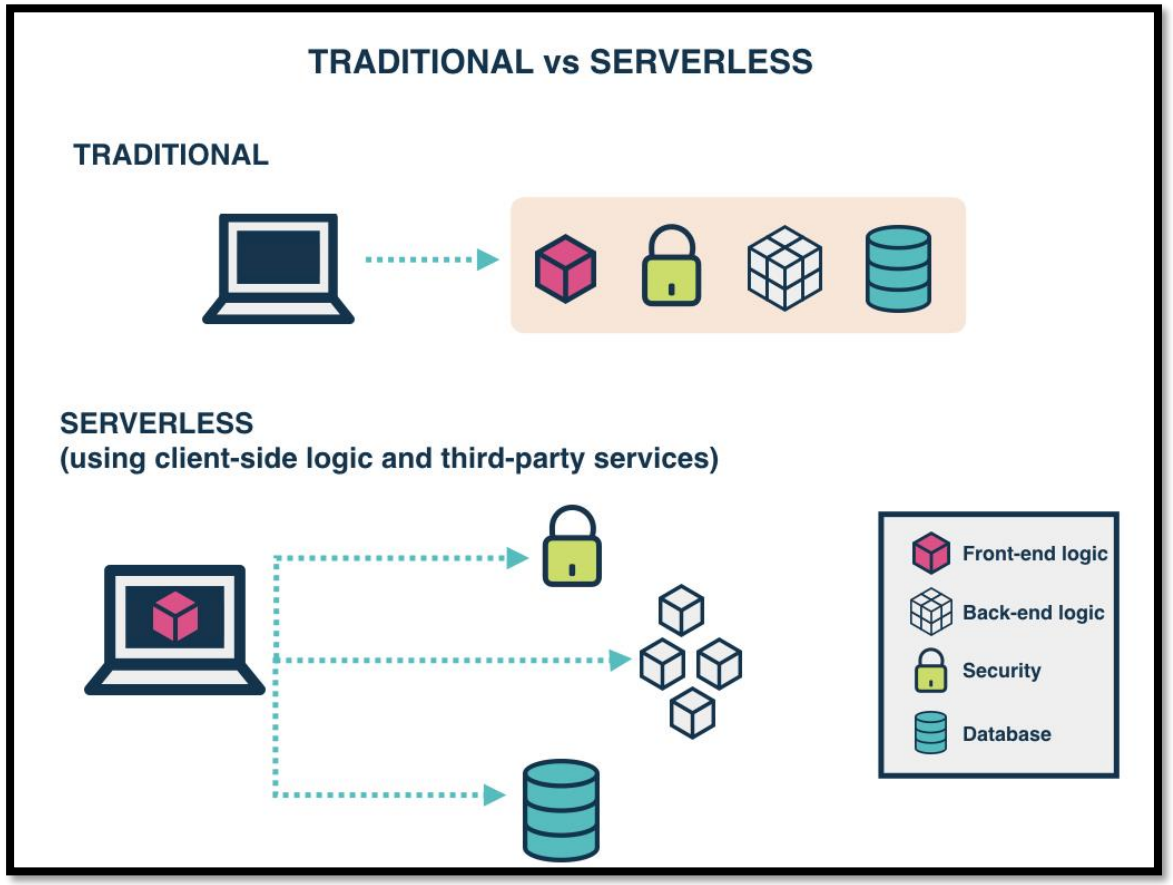
- 2012 - se asoció este tipo de enfoque sobretodo al uso de sistemas de integración continua y control de versiones como servicio, sin la necesidad de ser provisionados on-premises.
- 2014 Amazon lanzó su servicio AWS lambda, permitía desplegar porciones de código sin tener que hacernos cargo de la infraestructura subyacente.
- Julio de 2015 Amazon lanzó su API Gateway, que permitía además realizar peticiones HTTP sobre estas funciones desplegadas
- 2015 Surgen artículos que hablan del futuro de los servidores “Servers are dead”
- 2016 luego de la Serverless Conf, nada ha vuelto a ser igual en el mundo de la computación en el cloud.

# ¿QUÉ ES ESO DE SERVERLESS?

@secvalve "The next time you try and use the word **serverless** just remember it's like calling takeout **kitchenless**"

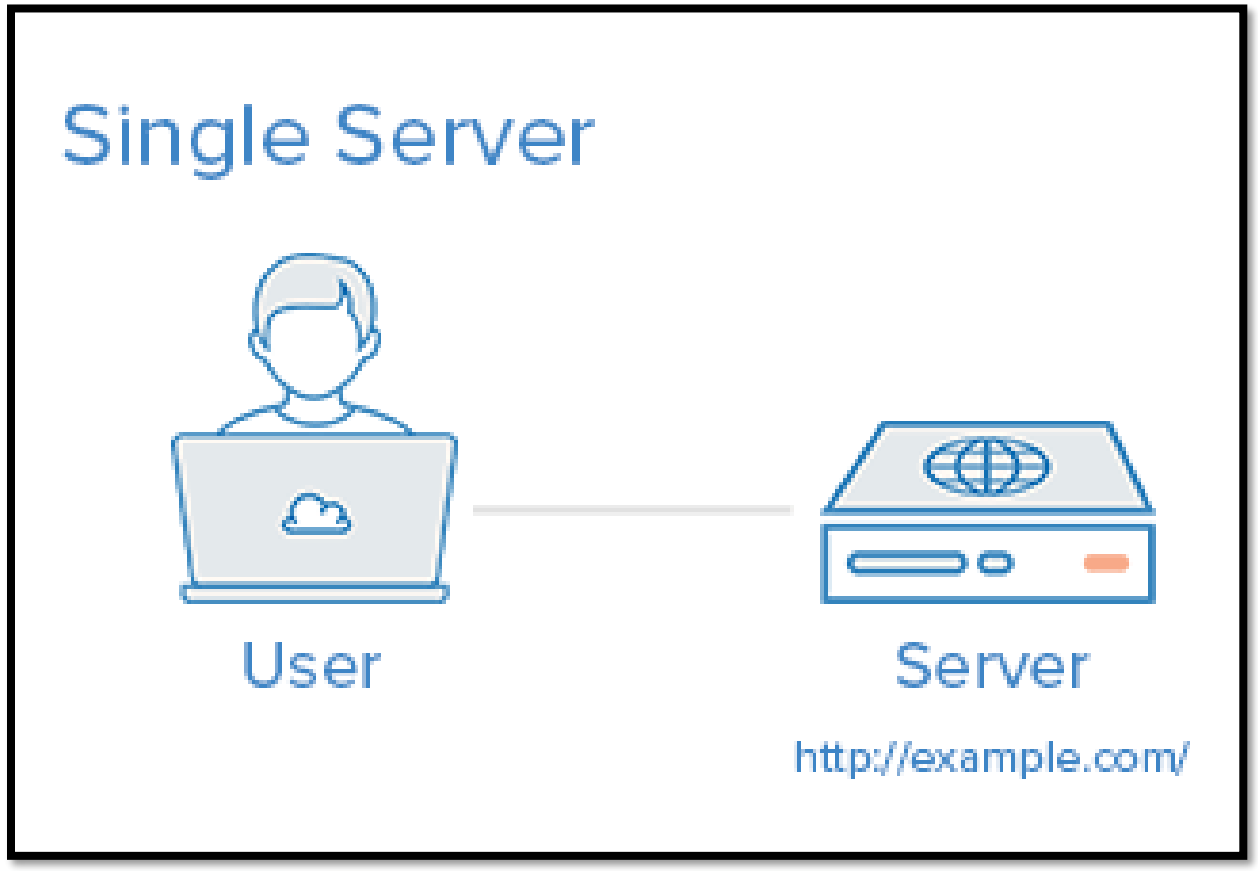
# DEFINICIÓN

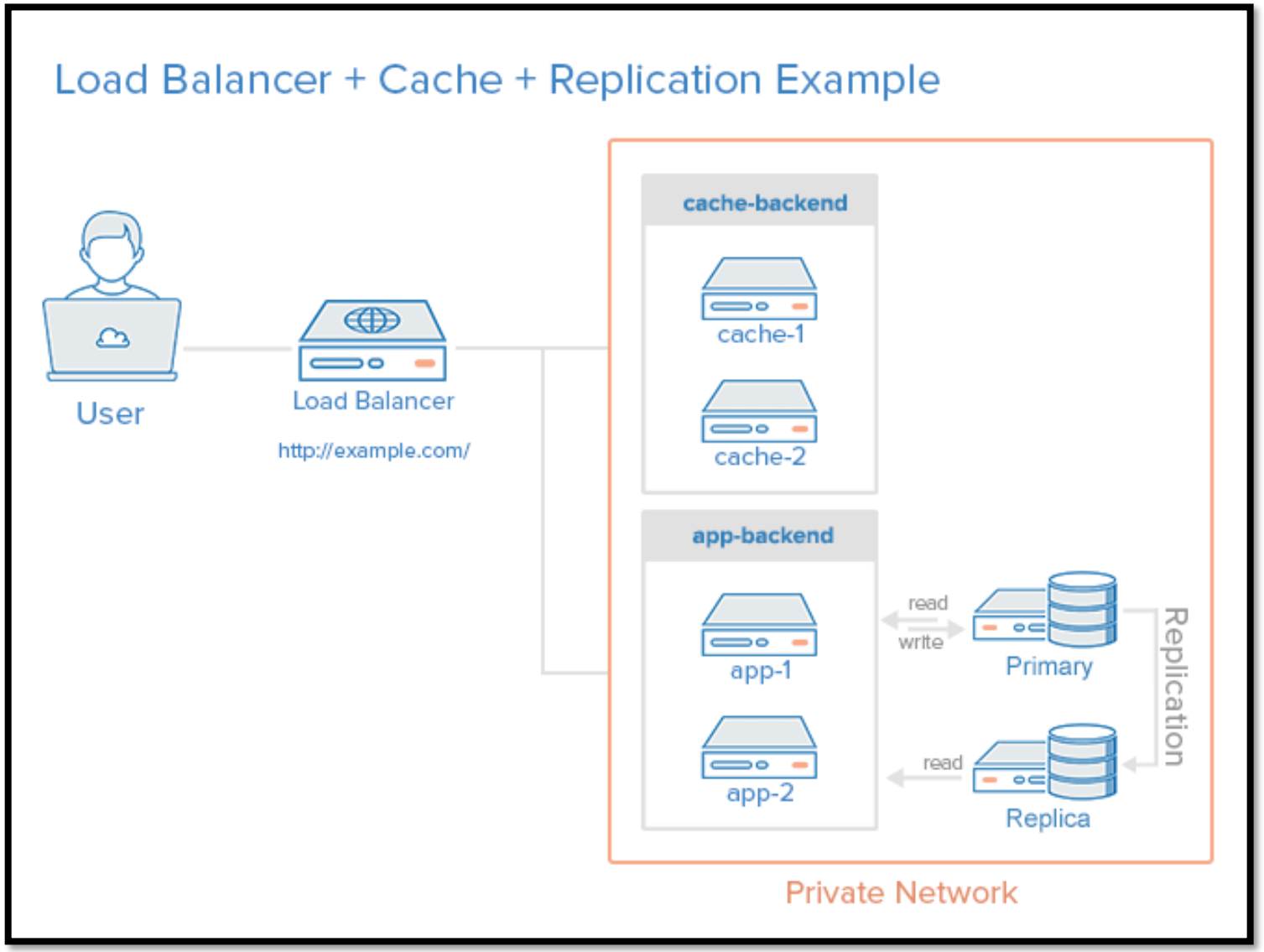
Serverless es un tipo de arquitectura donde los servidores (físicos o en la nube) dejan de existir para el desarrollador y en cambio el código corre en “ambientes de ejecución” que administran proveedores como Amazon, Google, IBM, etc.



# APLICACIONES TRADICIONALES (MONOLÍTICAS)

- Daño colateral
- Todo para uno y uno para el fallo
- Problemas Operacionales
- Líneas de tiempo desfasadas
- Se implementa con menos frecuencia
- Menos interrupciones
- Mayor tiempo en planificación




Load Balancer + Cache + Replication Example
User
Load Balancer
http://example.com/
cache-backend
cache-1
cache-2
app-backend
app-1
app-2
read
write
Primary
read
Replica
Replication
Private Network

# Dudas sobre Servidores (AAAHHHHHHH?)

- ¿Qué tamaño de servidores son los adecuados para mi presupuesto?
- ¿Cuántos usuarios crean demasiada sobrecarga para mis servidores?
- ¿Cuanta capacidad restante tiene mi servidor?
- ¿Cómo puedo detectar si un servidor ha sido comprometido?
- ¿Cuál Sistema Operativo debería tener mi servidor?
- ¿Cuáles usuarios deberían tener acceso a mi servidores?

¿Cómo puedo controlar el acceso a mis servidores ?

¿Cómo mantener mi sistema operativo parchado?

¿Cómo desplegaré nuevo código a mis servidores?

¿Puedo incrementar el uso de mis servidores?

¿Debería escalar mis servidores?

¿Cuál es el tamaño correcto para mejorar mi rendimiento?

Muchas dudas adicionales

# ARQUITECTURA SERVERLESS

Las funciones serverless son sencillas de usar cuando no se require guardar estado en memoria. Debido a que no se tiene control acerca de cuando los ambientes de ejecución son creados o destruidos, no se puede asumir que al guardar un dato en la memoria de la función, este se mantenga allí cuando la función sea nuevamente invocada.

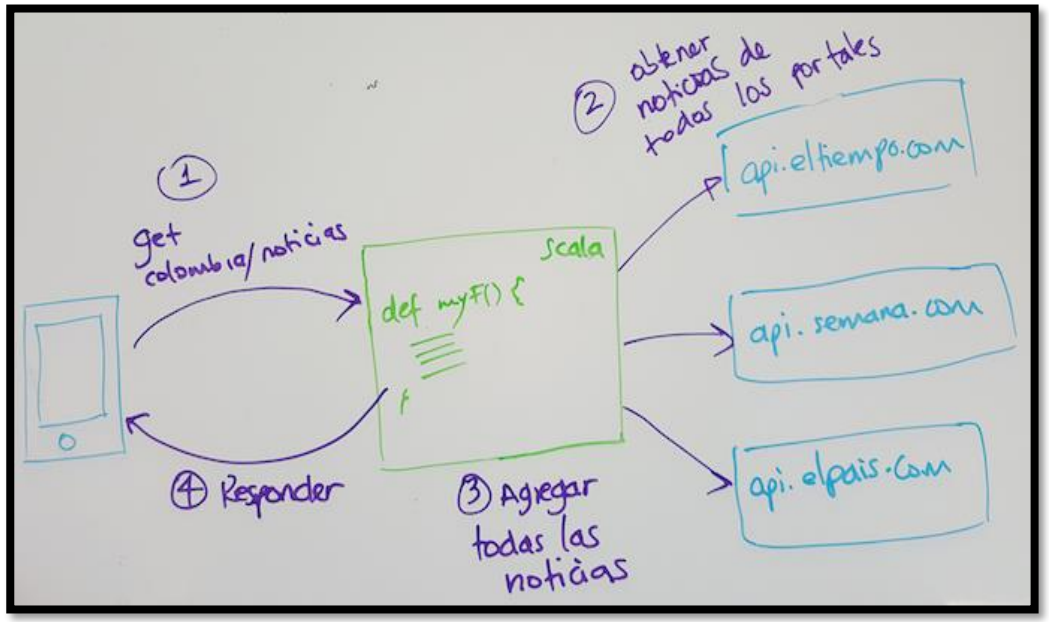
# VENTAJAS

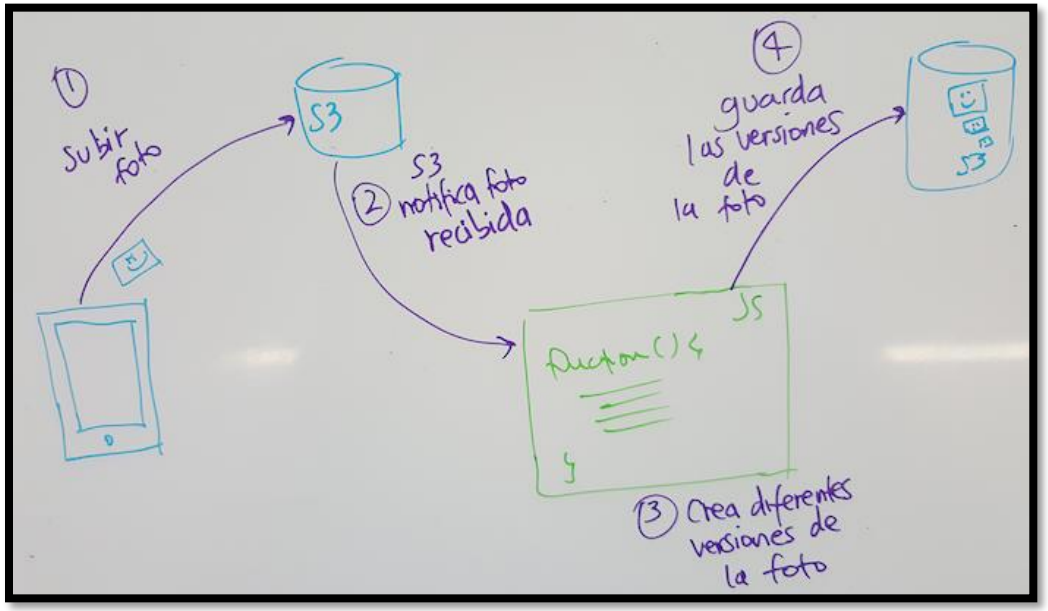
- **Completamente Administrada**
  - Sin aprovisionamiento
  - Cero Administración (a nivel de hardware)
  - Alta disponibilidad
- **Productividad del Desarrollador**
  - Enfoca en el código
  - Reduce el tiempo al mercado
  - Innova rapidamente
- **Escalamiento Continuo**
  - Automaticamente
  - Aumenta o disminuye

# DESVENTAJAS

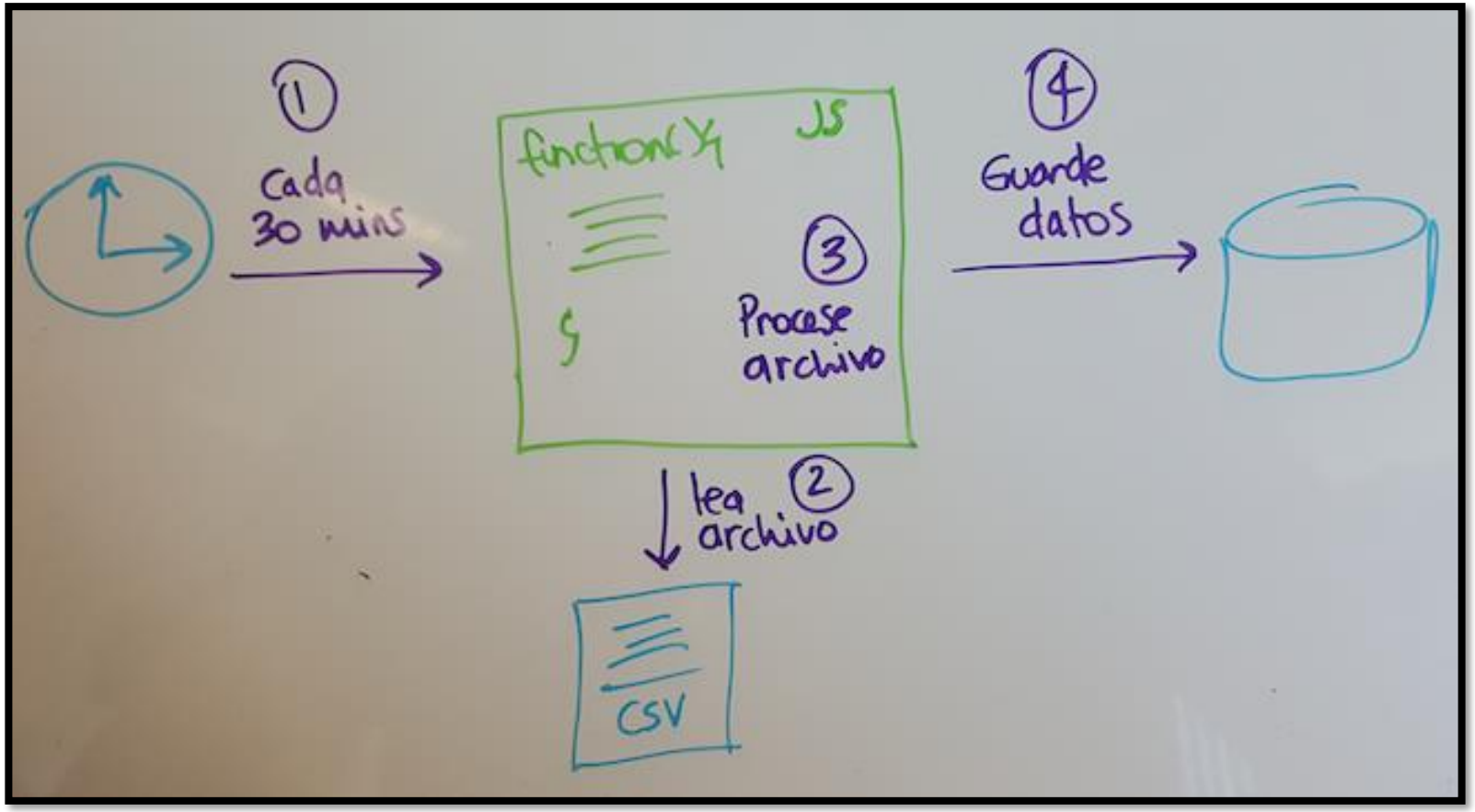
- Si no se desarrolla con cuidado, su código puede terminar bastante acoplado al proveedor.
- Al ser un servicio tan reciente, los lenguajes que se pueden usar para implementar las funciones están limitados por lo que esté soportado por el proveedor.
- Desplegar y monitorear el comportamiento de multiples funciones es mucho más complicado que monitorear un monolito.
- Se requiere esfuerzo extra para poder desarrollar localmente sin necesidad de desplegar el código a los ambientes de ejecución cada que se realice un cambio, ya que puede ser demorado y tedioso.
- Las herramientas alrededor de la automatización del despliegue de funciones serverless son aún muy inmaduras.

# Ejemplos Serverless

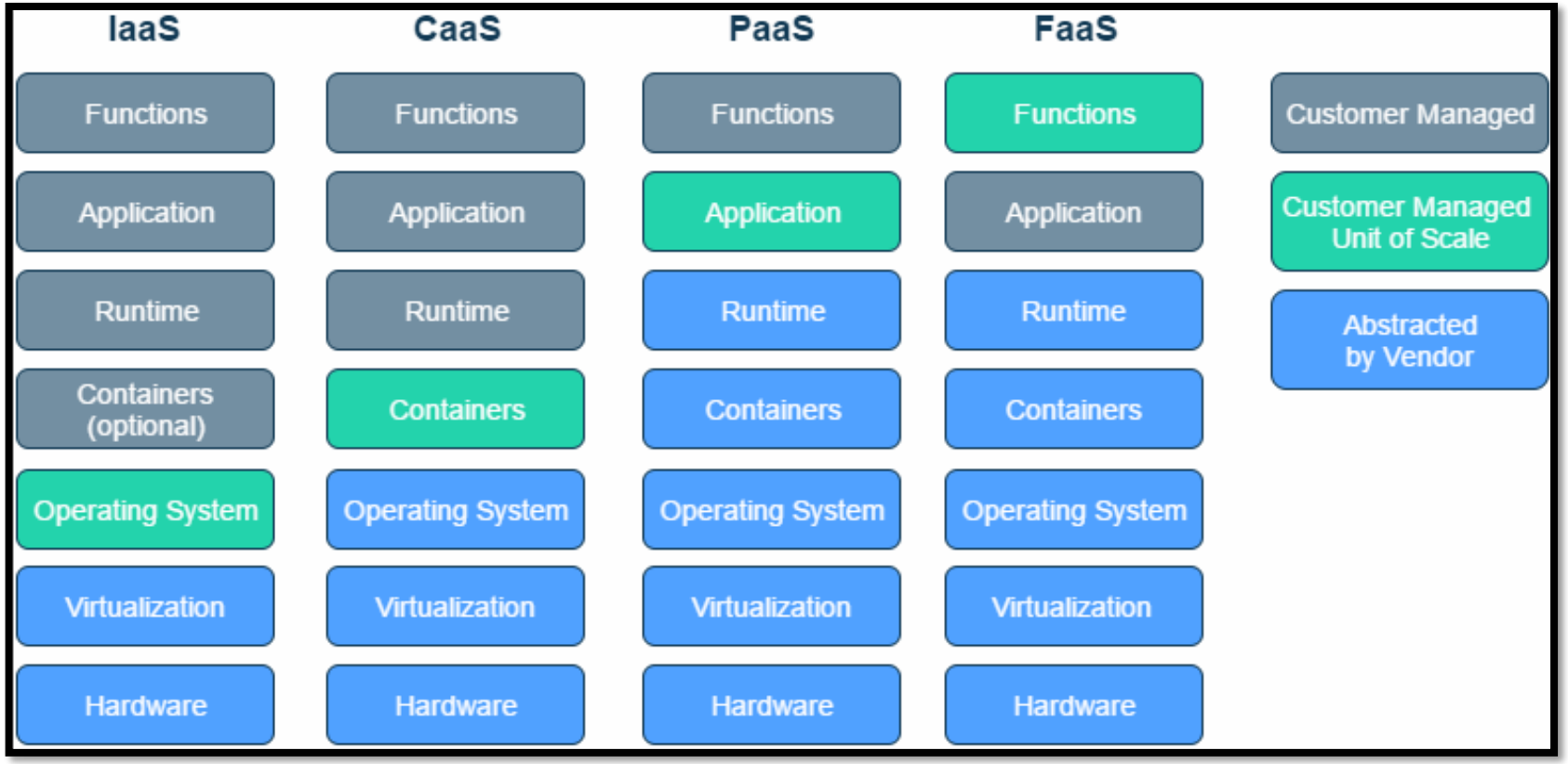


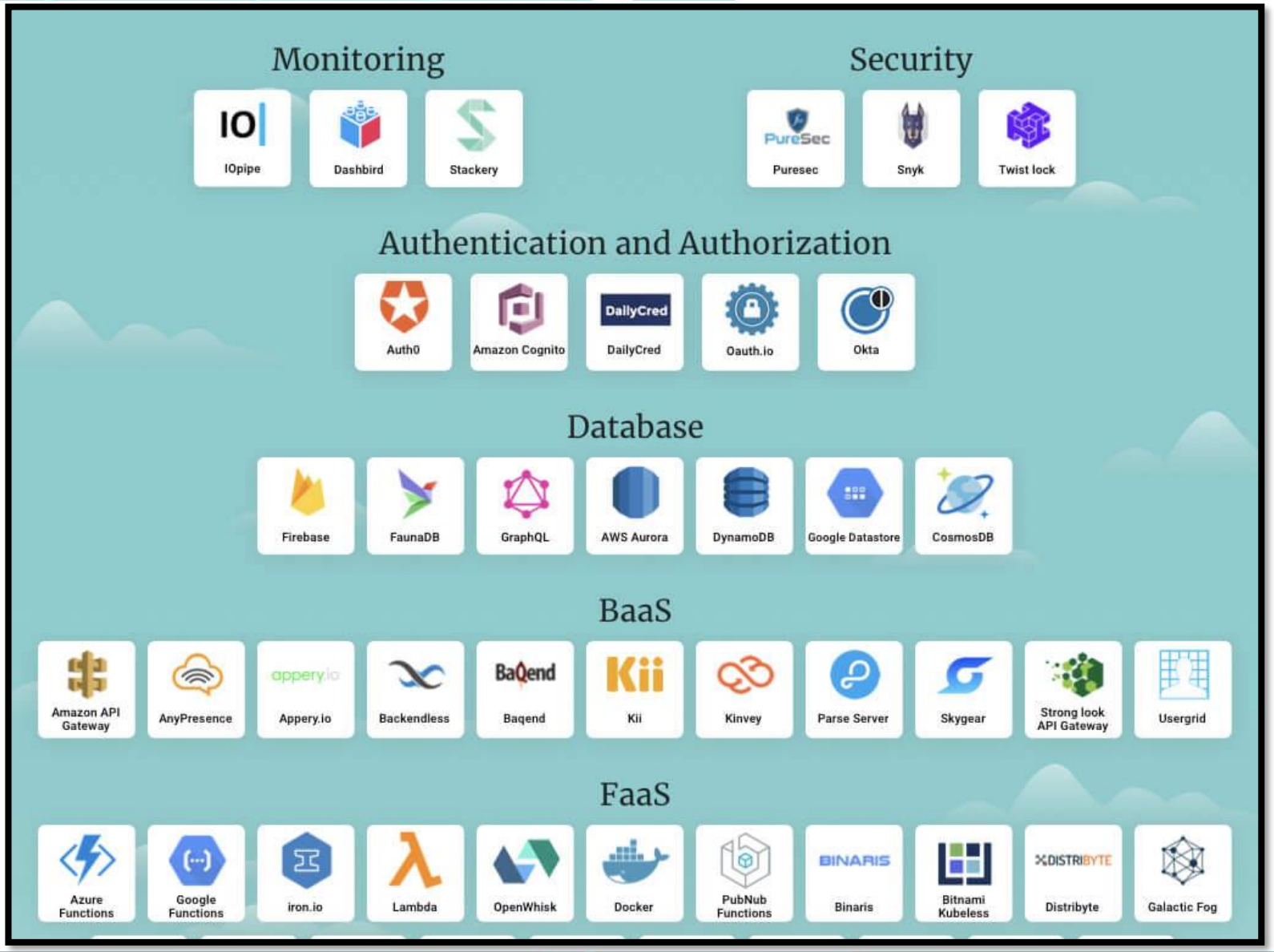
# Cont...


1
Cada
30 mins
function Y
JS
3
Procese
archivo
4
Guarde
datos
2
lea
archivo
CSV

# Modelos Cloud



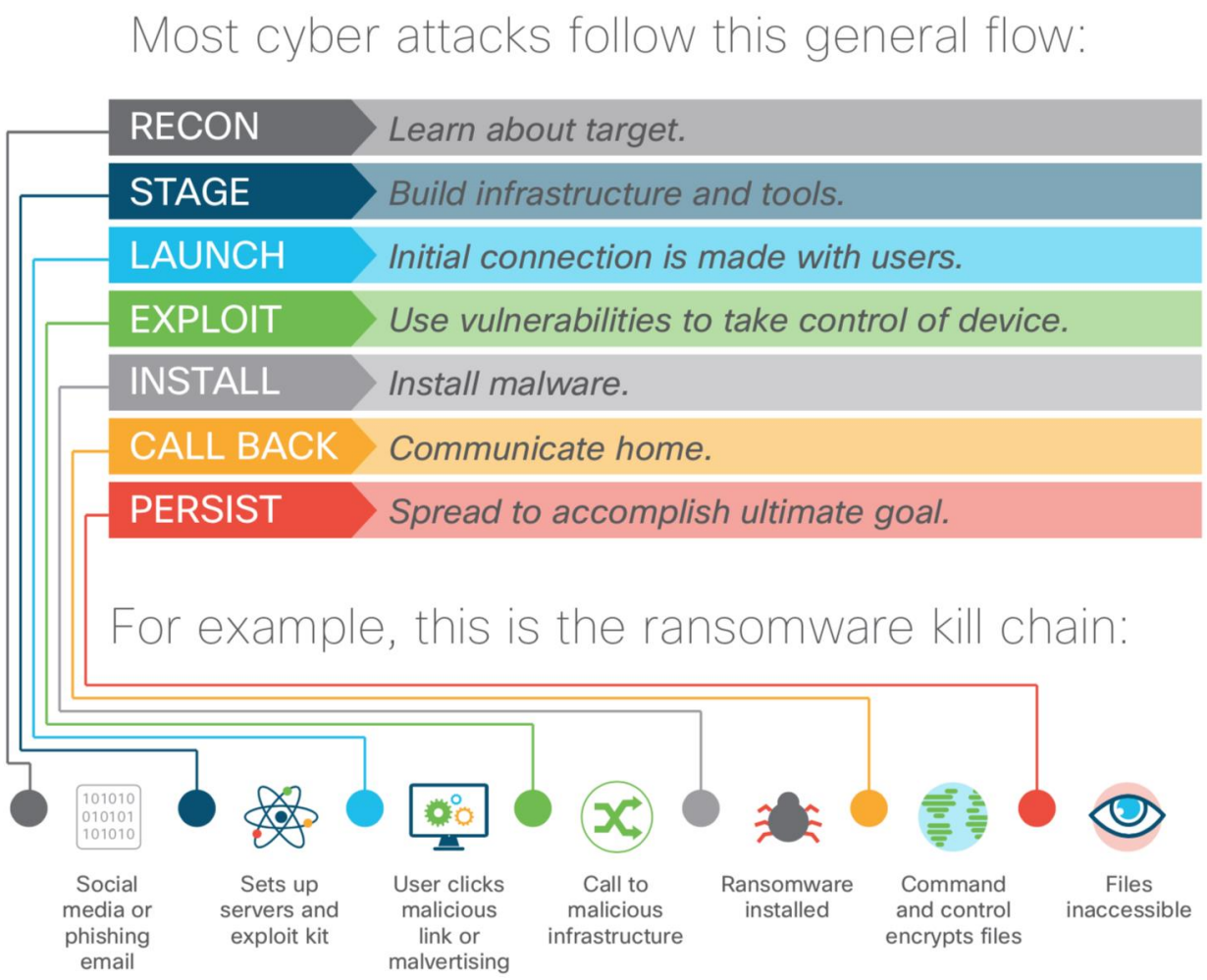
# Soluciones Diponibles



# Cont…


Azure Functions
Google Functions
iron.io
Lambda
OpenWhisk
Docker
PubNub Functions
Binaris
Bitnami Kubeless
Distribyte
Galactic Fog
Iguaz.IO Nuclio
Nano Lambda
OpenFaaS
OpenLambda
OVH Functions
Platform9 Fission
PubNub Blocks
Red Hat Apache OpenWhisk
Spotinst Functions
Syncano
Twilio Functions
weblab.io
Pivotal Spring Cloud Function
Frameworks
Apex
AWS Chalice
AWS SAM
Claudiajs
Dawson
Deep
EFFE
JRestless
Lambder
Lambda Forest
Lambdify
Lambdoku
Lambda-restify
Scar
Sparta
Squeezer
Zappa
Fission.io
Serverless Inc.
Up
CIM
Middy
Modofun
Shep
Turtle
Vandium
Galactic Fog
Tools/Platforms
Clay
LambCI
Node Lambda
Gordon
Kappa
Lambda-uploader
OpenLambda
Back&
Libraries
FDB
Python
StdLib

# Cyber kill chain



# Reconocimiento

```
$ curl https://nmapscan-qi[redacted]s.now.sh\?host\=blog.ropnop.com
Starting Nmap 7.70 ( https://nmap.org ) at 2018-11-11 00:35 UTC
Nmap scan report for blog.ropnop.com (104.18.42.134)
Host is up (0.24s latency).
Other addresses for blog.ropnop.com (not scanned): 104.18.43.134 2606:4700:30::6812:2b86 2606:4700
:30::6812:2a86
Not shown: 96 filtered ports
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
8080/tcp open  http-proxy
8443/tcp open  https-alt

Nmap done: 1 IP address (1 host up) scanned in 14.98 seconds
```

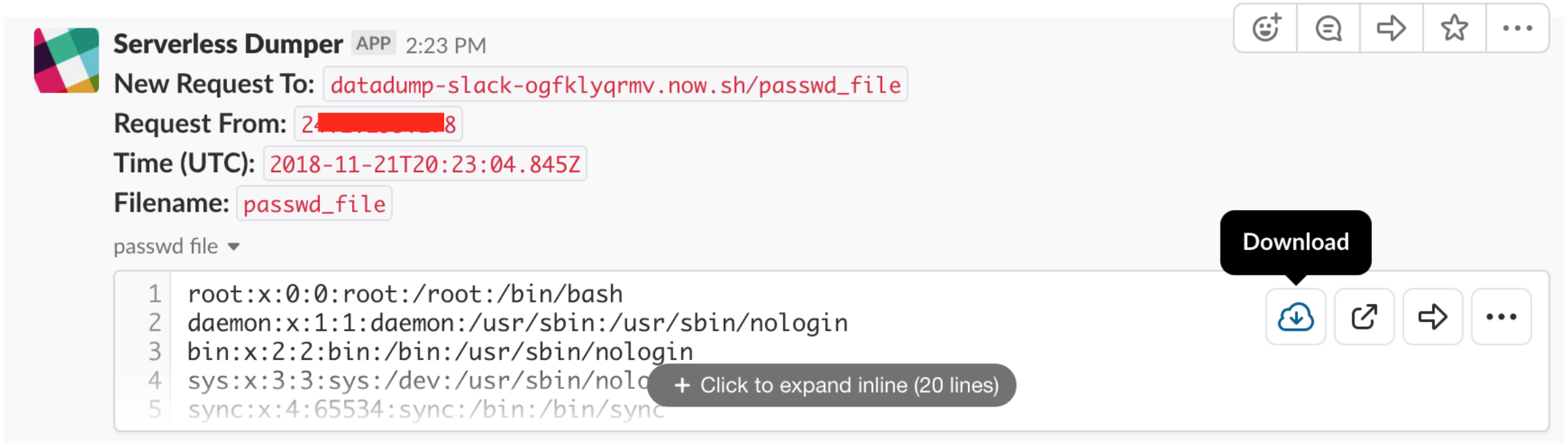
https://github.com/ropnop/serverless_toolkit

# Cont...

```
$ curl -s -X POST --data-binary "@100domains.txt" https://massdns-gingahvmrz.now.sh | jq .
[
  {
    "query_name": "crashlytics.com.",
    "query_type": "A",
    "resp_name": "crashlytics.com.",
    "resp_type": "A",
    "data": "23.21.125.136"
  },
  {
    "query_name": "crashlytics.com.",
    "query_type": "A",
    "resp_name": "crashlytics.com.",
    "resp_type": "A",
    "data": "174.129.250.71"
  },
  {
    "query_name": "crashlytics.com.",
    "query_type": "A",
    "resp_name": "crashlytics.com.",
    "resp_type": "A",
    "data": "23.21.91.79"
  },
```

https://github.com/ropnop/serverless_toolkit

# Command and control




https://github.com/ropnop/serverless_toolkit

# Demostración

# Conclusiones

- Serverless es genial.
- Se realizan implementaciones rápidamente.
- Muchos usos potenciales. Cero gastos generales en mantenimiento.
- Podemos realizar pentesting con herramientas automatizadas sin tener que desplegar un servidor vps.
- De igual manera podemos crear nuestra infraestructura propia cuando se requiera.

# ¿Preguntas?

# Referencias


- Borillo, R. (2020). Qué es serverless y por qué adoptarlo en el desarrollo de tu próxima aplicación. Retrieved 14 April 2020, from https://www.genbeta.com/desarrollo/que-serverless-que-adoptarlo-desarrollo-tu-proxima-aplicacion
- Qué es eso de serverless?. (2020). Retrieved 14 April 2020, from https://medium.com/@PamRucinque/qu%C3%A9-es-eso-de-serverless-f4f6c8949b87
- Thahir , S. (2017, marzo 29). 7 Reasons Serverless Computing Revolution Cloud. Recuperado 14 de abril de 2020, de https://datafloq.com/read/7-reasons-serverless-computing-revolution-cloud/2871
- What Is Serverless Computing? | Serverless Definition. (s. f.). Recuperado 14 de abril de 2020, de https://www.cloudflare.com/
- Solanki, J. (s. f.). Exploring the Ecosystem of Serverless Technologies. Recuperado 14 de abril de 2020, de https://www.simform.com/ecosystem-serverless-technologies/