Serverless

Use case: Image resizing

¿Qué es *serverless*?

- **Foco en el código, no en los servidores:** la esencia de serverless es la ausencia del concepto de servidor durante el desarrollo del software.
- **Pagá por lo que realmente usás:** El usuario final sólo paga por el tiempo y recursos usados en si.
- **Aplicaciones que dependen de servicios de terceros.**

¿Por qué usar serverless?

- **Bajo mantenimientos:** no hay que ocuparse de nada respecto al servidor (actualizaciones de seguridad, que esté levantado, etc.)
- **Bajos costos:** se paga por request. Si la aplicación no se usa, no se cobra.
- **Fácil de escalar:** el proveedor del servicio escala en base a la demanda.

Ejemplo

- Usuarios activos diarios: 1000
- Requests por día (por usuario) a la API: 20
- Almacenando alrededor de 10 MB en archivos en S3.

Cálculo de costos

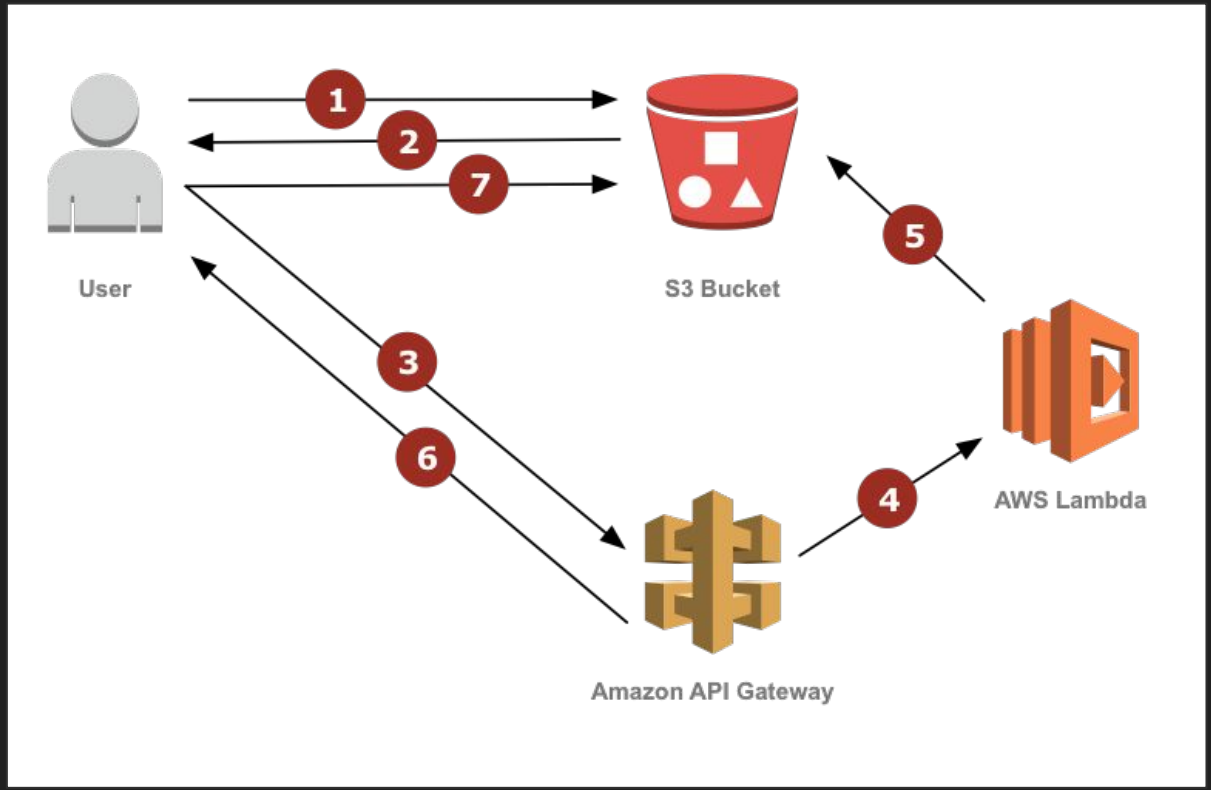
Service	Rate	Cost
Cognito	Free[1]	$0.00
API Gateway	$3.5/M reqs + $0.09/GB transfer	$2.20
Lambda	Free[2]	$0.00
DynamoDB	$0.0065/hr 10 write units, $0.0065/hr 50 read units[3]	$2.80
S3	$0.023/GB storage, $0.005/K PUT, $0.004/10K GET, $0.0025/M objects[4]	$0.24
CloudFront	$0.085/GB transfer + $0.01/10K reqs	$0.86
Route53	$0.50 per hosted zone + $0.40/M queries	$0.50
Certificate Manager	Free	$0.00
Total		**$6.10**

[1] Cognito is free for < 50K MAUs and $0.00550/MAU onwards.
[2] Lambda is free for < 1M requests and 400000GB-secs of compute.
[3] DynamoDB gives 25GB of free storage.
[4] S3 gives 1GB of free transfer.

¿Por qué no usar serverless?

- **En la práctica, toma un tiempo para un serverless escalable en responder a la primer request de la función.**
  - Problema “cold start”
    - E.g. Lambda y sus contenedores (al comienzo, y luego de un tiempo sin uso)
- **Algunas implementaciones Faas (como AWS Lambda) no proveen herramientas para testear funciones localmente.**
- **Arquitecturas serverless (y microservice) introducen overhead adicional a las llamadas de funciones.**
  - No hay operaciones “locales”; no se puede asumir que dos funciones que se comunican se encuentran en el mismo servidor.
- **Vendor lock-in**
  - Se depende del proveedor, y no se tiene control total de la aplicación.

Arquitectura AWS Lambda / S3 / API Gateway



Servidor Local / DigitalOcean

- Aplicación Node.js
- 2 rutas:
  - POST /upload
    - Para subir imágenes nuevas al filesystem del servidor
  - GET /images/ANCHOxALTO/image.jpg
    - Para descargar imágenes en el tamaño deseado
- 2 deploys:
  - Servidor local: http://localhost:8080
  - Droplet en DigitalOcean: http://lucasapps.tk:8080
- Se utilizaron las mismas librerías que en el ejemplo de AWS S3 + Lambda, para una mejor comparación

Resultados

Resize	localhost	DigitalOcean	Amazon AWS
100 x 100			
Espacio en disco	1811 KB	1801 KB	1800 KB
Tiempo de resizeo	177 s	97 s	30.7 s
Tiempo de descarga sin cacheo	3.3 s	89 s	181 s
Tiempo de descarga con cacheo	0.3 s	89 s *	82 s
200 x 200			
Espacio en disco	6027 KB	5967 KB	5960 KB
Tiempo de resizeo	170 s	233 s	37.3 s
Tiempo de descarga sin cacheo	5.1 s	136 s	241 s
Tiempo de descarga con cacheo	0.3 s	118 s	121 s
500 x 500			
Espacio en disco	28239 KB	28103 KB	28003 KB
Tiempo de resizeo	208 s	1308 s	48.5 s
Tiempo de descarga sin cacheo	14 s	289 s	364 s
Tiempo de descarga con cacheo	435 ms	292 s *	222 s

Now

- Fácil:
  - No necesitas instalar varias aplicaciones para correrlo.
  - No necesitas instalar git ni SVN
  - No te tenes que guardar tokens ni nada por el estilo.
  - No te tenes que registrar en ningún cloud service provider.
- Ilimitado
  - Cada vez que ejecutas now, se te da una nueva URL que representa el estado actual de tu aplicación.
  - No tenes que borrar los anteriores.
  - No tenes que setear proyectos ni aplicaciones.
  - Las URLs duran para siempre
- En tiempo real
- Todo el tráfico se sirve por HTTP/2.

Tipos de *deployments*

- Static projects, link
- Docker projects, link
- Node.js projects, link

¿Qué sucede con la escalabilidad?

- **Dinámico:** no se reservan recursos de más ni de menos. No hace falta configurar nada.
- **Multi-cloud:** no depende de un proveedor específico, sino que se abstrae de ellos.
- **Todo se sirve por HTTP/2.** Cada request es como si fuera una micro funcion desde un punto de vista de eficiencia y de ancho de banda.
    - Multiplexing. HTTP/2 mejora la latencia y el uso de ancho de banda al reutilizar la misma conexión TCP para todas las comunicaciones. great latency characteristics and bandwidth use by re-utilizing the same TCP connection for all communications. Esto otorga una gran ventaja sobre todo para usuarios mobile.
    - Header compression. Los headers que son siempre los mismos durante la vida de una conexión se mandan una sola vez.

zeit.co/now
realtime global deployments

- Nuestros static websites:
  - https://tpe-redes-eljbxdnetn.now.sh/
  - https://now-static-website-ormmuhwwvk.now.sh
- GitHub repository: https://github.com/zeit/now-cli
- `npm install -g now`
- Para más información sobre *features*: https://zeit.co/docs#features

Ejemplos de uso de now

```
Examples:

- Deploys the current directory

  $ now

- Deploys a custom path `/usr/src/project`

  $ now /usr/src/project

- Deploys a GitHub repository

  $ now user/repo#ref

- Deploys a GitHub, GitLab or Bitbucket repo using its URL

  $ now https://gitlab.com/user/repo

- Deploys with ENV vars

  $ now -e NODE_ENV=production -e MYSQL_PASSWORD=@mysql-password

- Displays comprehensive help for the subcommand `list`

  $ now help list
```

Algunos comandos incluidos en now

```
Δ now [options] <command | path>

Commands:

  Cloud

    deploy               [path]      Performs a deployment (default)
    ls | list            [app]       List deployments
    rm | remove          [id]        Remove a deployment
    ln | alias           [id] [url]  Configures aliases for deployments
    domains              [name]      Manages your domain names
    certs                [cmd]       Manages your SSL certificates
    secrets              [name]      Manages your secret environment variables
    dns                  [name]      Manages your DNS records
    logs                 [url]       Displays the logs for a deployment
    scale                [args]      Scales the instance count of a deployment
    help                 [cmd]       Displays complete help for [cmd]

  Administrative

    billing | cc         [cmd]       Manages your credit cards and billing methods
    upgrade | downgrade  [plan]      Upgrades or downgrades your plan
    teams                [team]      Manages your teams
    switch                           Switches between teams and your account
    login                            Login into your account or creates a new one
    logout                           Logout from your account
```