Serverless Tales from the Trenches
Peter Sbarski, PhD
AWS Community Hero
@sbarski

Author
Serverless Architectures on AWS
https://book.acloud.guru

VP Engineering | VP Content
A Cloud Guru
https://acloud.guru

Organizer
Serverlessconf
https://serverlessconf.io
TEACHING THE WORLD TO CLOUD

Join the 500,000+ engineers that have taken courses with A Cloud Guru.

BROWSE CONTENT
### Why Serverless?

<table>
<thead>
<tr>
<th></th>
<th>IaaS</th>
<th>PaaS</th>
<th>Serverless</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unit of Scale</strong></td>
<td>Virtual Machine or Container (Docker)</td>
<td>Application</td>
<td>Function</td>
</tr>
<tr>
<td><strong>Fleet Operational Responsibility</strong></td>
<td>Application developer</td>
<td>Shared between developer and vendor</td>
<td>Vendor only</td>
</tr>
<tr>
<td><strong>Required Management &amp; Maintenance</strong></td>
<td>High – Operating System level</td>
<td>Medium – Application level</td>
<td>Low – function level</td>
</tr>
<tr>
<td></td>
<td>Per VM per minute or hour</td>
<td>Per VM per minute or hour</td>
<td>Per 100 milliseconds (continuous)</td>
</tr>
<tr>
<td><strong>Impact of Idle Time</strong></td>
<td>Economic loss when machines are idle or underused</td>
<td>Economic loss when machines are idle or underused</td>
<td>None – functions execute only when needed</td>
</tr>
<tr>
<td><strong>Integration with other vendor services</strong></td>
<td>Manual</td>
<td>Mixed</td>
<td>Automatic</td>
</tr>
</tbody>
</table>

*Source: Serverless Design Patterns (T. Wagner, Y. Kiriati, P. Sbarski)*
Number of Requests (Millions | Thousands | Not Scaled) Show Details

- **Number of HTTP Requests**: Total: 7,494 K  Average: 0.2575 K  Minimum: 0.086 K  Maximum: 0.501 K
- **All Requests**: Total: 7,549,221 K  Average: 268.865 K  Minimum: 0.086 K  Maximum: 373.999 K

Data Transferred By Protocol (Gigabytes | Megabytes | Kilobytes) Show Details

- **HTTP Data**: Total: 0.0027 GB  Average: 0.0001 GB  Minimum: 0 GB  Maximum: 0.0001 GB
- **HTTPS Data**: Total: 37,190,9819 GB  Average: 1,327.177 GB  Minimum: 977.84 GB  Maximum: 1,642.7816 GB
- **All Data**: Total: 37,190,9846 GB  Average: 1,327.178 GB  Minimum: 0 GB  Maximum: 1,642.7816 GB
## February 2018

<table>
<thead>
<tr>
<th>Service</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>API Gateway</td>
<td>$100.00</td>
</tr>
<tr>
<td>Lambda</td>
<td>$206.66</td>
</tr>
<tr>
<td>DynamoDB</td>
<td>$424.27</td>
</tr>
<tr>
<td>Elastic Transcoder</td>
<td>$169.89</td>
</tr>
<tr>
<td>Kinesis</td>
<td>$20.16</td>
</tr>
<tr>
<td>Redshift</td>
<td>$503.50</td>
</tr>
<tr>
<td>Simple Storage Service</td>
<td>$58.36</td>
</tr>
<tr>
<td>Developer Support</td>
<td>$178.74</td>
</tr>
<tr>
<td>CloudWatch</td>
<td>$586.24</td>
</tr>
<tr>
<td>CloudFront</td>
<td>$3,775.42</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$6,023.24</strong></td>
</tr>
</tbody>
</table>
Serverless Stories

Frontend and API Failover

Making functions & services more resilient

Patterns and architectures
Frontend Failover
Frontend Failover

December 2017

Website Outage
The website has been stable now for some time so we are marking this issue as resolved. We will update this status later...
Dec 13, 10:34 - 14:15 AEDT

November 2017

No incidents reported for this month.

October 2017

No incidents reported for this month.
Frontend Failover

Settings for h5boilerplate
serverlessarchitectures.com
Deploys from GitHub. Owned by Peter Sbarski
Last update at 7:13 pm (25 minutes ago)

404

We've got ourselves all confused.
This page doesn't exist.

Go To Our Home Page
Frontend Failover

You can also type the domain name for the resource. Examples:
- CloudFront distribution domain name: d111111abofef.cloudfront.net
- Elastic Beanstalk environment CNAME: example.elasticbeanstalk.com
- ELB load balancer DNS name: example-1.us-east-1.elb.amazonaws.com
- AWS website endpoint: s3-website.us-east-2.amazonaws.com
- Resource record set in this host zone: www.example.com

Learn More

Routing Policy: Failover

Route 53 responds to queries using primary record sets if any are healthy, or using secondary record sets otherwise. Learn More

Failover Record Type: Primary

Set ID: Primary

Evaluate Target Health: Yes

Associate with Health Check: Yes
Frontend Failover

Danger zone
Irreversible and destructive actions. Tread lightly.

Delete site
Once you delete a site, there is no going back.

Delete this site
API Failover
API Failover

**API Gateway – US East 1**

**Endpoint Configuration**

- **Regional**
  - **Target Domain Name**: d-3gopa9fu5i.execute-api.us-east-1.amazonaws.com
  - **ACM Certificate**: api.serverlessarchitectures.com (6ee3c7fb)

**Base Path Mappings**

<table>
<thead>
<tr>
<th>Path</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>dev-serverlessarchitecture:dev</td>
</tr>
</tbody>
</table>

**Edit**

**API Gateway – US West 1**

**Endpoint Configuration**

- **Regional**
  - **Target Domain Name**: d-lx6d0pjfm0.execute-api.us-west-1.amazonaws.com
  - **ACM Certificate**: api.serverlessarchitectures.com (17c4b4d7)

**Base Path Mappings**

<table>
<thead>
<tr>
<th>Path</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>dev-serverlessarchitecture:dev</td>
</tr>
</tbody>
</table>

**Edit**
## API Failover

### Health Check Details

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>api-us-east</td>
<td>healthy</td>
<td></td>
</tr>
<tr>
<td>com:443/ind...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Network Details

- **IP Address**: 54.68.178.118
- **Country**: United States
- **Region**: Virginia
- **City**: Ashburn
- **ISP**: Amazon Technologies Inc.
- **Latency**: 39.0637 ms
- **Enable SNI**: Yes

### Health Check Regions

- US East (N. Virginia)
- US West (N. California)
- US West (Oregon)
- EU (ireland)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- South America (São Paulo)
## API Failover

### Method Execution

**/ping - GET - Method Request**

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Description</th>
<th>Alarms</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>api</td>
<td>Unhealthy</td>
<td><a href="https://ucsg94rkda.execute-api.us-east-1.amazonaws.com">Link</a></td>
<td>No alarms configured</td>
<td>31f03ed8-c0d3-4d03-8c5f-444a36ce860d</td>
</tr>
<tr>
<td>h5boilerplate</td>
<td></td>
<td><a href="https://h5boilerplate.netlify.com:443">Link</a></td>
<td>No alarms configured</td>
<td>cc0c2152-19fa-4bdf-4b8-f0f65a55e4f</td>
</tr>
</tbody>
</table>

### Health checkers

<table>
<thead>
<tr>
<th>Health checker region</th>
<th>Health checker IP</th>
<th>Last checked</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asia Pacific (Tokyo)</td>
<td>54.250.253.245</td>
<td>Mar 17, 2018 11:39:45 AM</td>
<td>Failure: Status Code 403, Forbidden, Resolved IP: 13.32.52.66</td>
</tr>
<tr>
<td>Asia Pacific (Singapore)</td>
<td>54.251.31.149</td>
<td>Mar 17, 2018 11:39:54 AM</td>
<td>Failure: Status Code 403, Forbidden, Resolved IP: 54.230.15.126</td>
</tr>
<tr>
<td>EU (Ireland)</td>
<td>54.228.16.21</td>
<td>Mar 17, 2018 11:39:52 AM</td>
<td>Failure: Status Code 403, Forbidden, Resolved IP: 52.85.201.126</td>
</tr>
<tr>
<td>EU (Ireland)</td>
<td>176.34.159.245</td>
<td>Mar 17, 2018 11:39:52 AM</td>
<td>Failure: Status Code 403, Forbidden, Resolved IP: 52.85.201.126</td>
</tr>
<tr>
<td>South America (São Paulo)</td>
<td>177.71.207.181</td>
<td>Mar 17, 2018 11:39:37 AM</td>
<td>Failure: Status Code 403, Forbidden, Resolved IP: 52.84.177.126</td>
</tr>
<tr>
<td>US East (N. Virginia)</td>
<td>54.243.31.245</td>
<td>Mar 17, 2018 11:39:52 AM</td>
<td>Failure: Status Code 403, Forbidden, Resolved IP: 52.84.128.126</td>
</tr>
</tbody>
</table>
| US West (N. California) | 54.241.32.117      | Mar 17, 2018 11:39:40 AM   | Failure: Status Code 403, Forbidden, Resolved IP: 52.84.237.5 
API Failover

![API Failover](image-url)
Making functions & services more resilient
Handling Errors

Peter uploads a file

Source S3 Bucket

Lambda

Elastic Transcoder

Firebase (Database)

Destination S3

Lambda

SNS

Lambda

Destination S3
Handling Errors

Peter uploads a file

Source S3 Bucket

Lambda

Elastic Transcoder

Error!

Dead Letter Queue (SNS)

Error!

Firebase (Database)

Lambda

SNS

Destination S3
Burning down the house
Burning down the house

You might still over provision or under provision DB connections

Multiple functions may need DB access with different usage profiles at different times.
Burning down the house

Dynamic Connection Management

Do something interesting
CloudWatch Alarm
Amazon CloudWatch
Amazon DynamoDB
Maintain atomic counter and max count
Helper
Open / close connections
DB operations

https://github.com/aws-samples/aws-lambda-manage-rds-connections
Tips

• 1 function = 1 task (avoid fat monoliths)

• No state (be idempotent)

• Design for failure

• More memory = more CPU and IO

• Set function concurrency to 0 as a kill switch

• Keep permissions and roles tight

• Incremental architecture is not dirty
Patterns and Architectures
### Patterns

#### Primitive
- Periodical (Cron Jobs)

#### API
- Proxy
- Facade

#### Orchestration
- One way chain
- Two way chain
- Fan in
- Fan out

#### Workflows
- Long Running tasks
- Pipes and Filters
- Inline Stream Transform

#### Traditional
- Command
- Singleton

#### Compound
- Backends
- CQRS
- Data processing

Source: Serverless Design Patterns (T. Wagner, Y. Kiriaty, P. Sbarski)
Patterns

Name
API Proxy (also known as wrapper)

Description
Acts as a mediator between two systems that cannot communicate directly. Transforms request and response payloads to facilitate exchange of information.

Motivation
Useful when incompatible systems need to talk. Reduces coupling by removing the need to build direct dependencies between incompatible systems.
Transformer function transforms JSON to XML and back again.
Patterns

Name
Simple fan-out

Description
Allows multiple endpoints to receive a copy of an input event. Turns any single-receiver delivery system into a multiple-receiver system.

Motivation
Event-based systems are often designed to have a single receiver for events, and API calls are by definition single receiver. The simple fan-out pattern asynchronously delivers its triggering event to one or more workers.
How would you design transactional fan-out?
Patterns

Name
Inline stream transform

Description
Transmits data between systems. Can be chained, can multiple and demultiplex at the source or destination. A transformation function can transform a record with the result progressing through the stream.

Motivation
A way to decouple systems that share data. Can offer temporal decoupling by allowing producers and consumers to operate at different rates. A transform function is used to clean, modify, group, analyze data before it gets to the consumer.
A stream can be sharded if order is not important.
How can you get started?

- serverlessconf.io & video.serverlessconf.io
- Follow @serverlessconf for serverlessconf info
- A Cloud Guru acloud.guru/serverless
- Follow @acloudguru and @sbarski
Thanks :-)