10 Things I Learned Using Serverless

Paulo Andrade, 23rd March

What is Serverless ?

- Serverless is a way of running application code without provisioning or managing the underlying servers
- The user writes functions that are executed by the cloud providers
- The functions automatically scale based on the traffic
- The user only pays for the time consumed, memory and the invocation count

AWS Lambda

- “AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers, creating workload-aware cluster scaling logic, maintaining event integrations, or managing runtimes”
- The developer writes the function code and AWS is responsible for running them
- AWS Lambda provides an interface the function code needs to implement to receive the payload and return a response

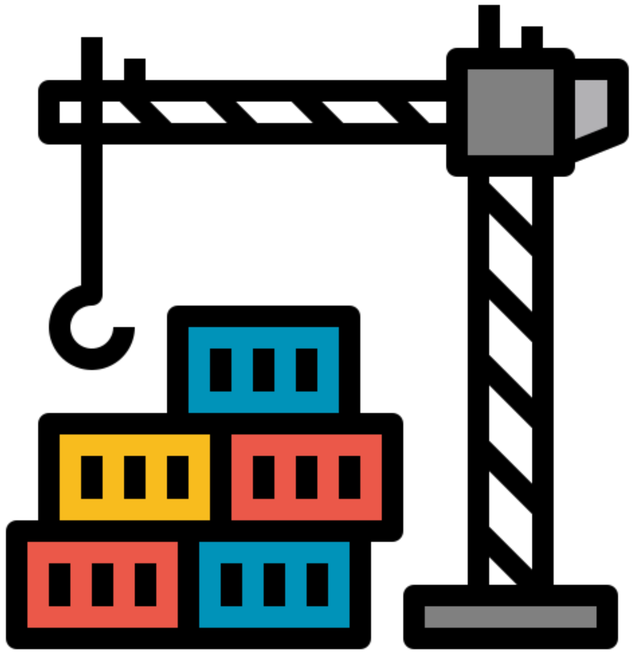
10 Things I Learned Using Serverless

1. Serverless applications run on containers
2. Functions can be triggered by different events
3. Cold starts might impact several users
4. Load stuff you need on the startup
5. Use a module bundler
6. Load the minimum dependencies into memory
7. Use middlewares and focus more on the business logic
8. Use a tool to manage deploys
9. Be careful when managing database connections
10. Use monitoring tools

1. Serverless applications run on containers

1. Serverless applications run on containers

- Each execution requires a server to handle the request
- The applications run on containers under the hood
- Similar to running an application using docker
- Isolation from another executions
- Helpful to know the developer's responsibilities



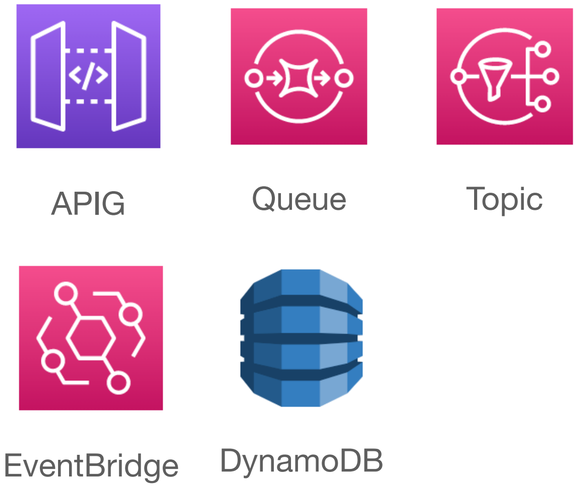
Icon taken from

2. Functions can be triggered by different events

Events

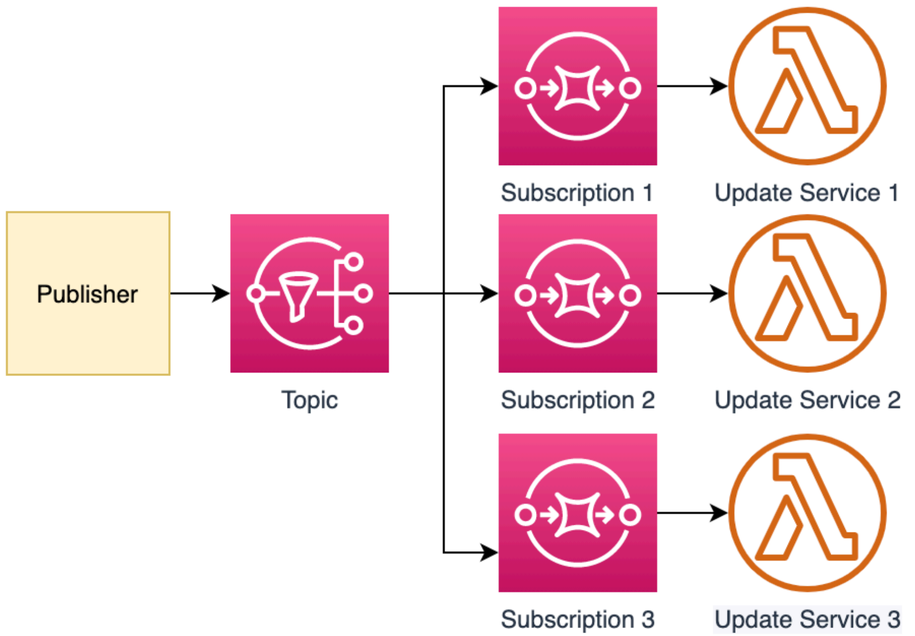
- HTTP request
- Queue
- Topic
- Stream
- CronJob

AWS Lambda



Google Cloud Functions

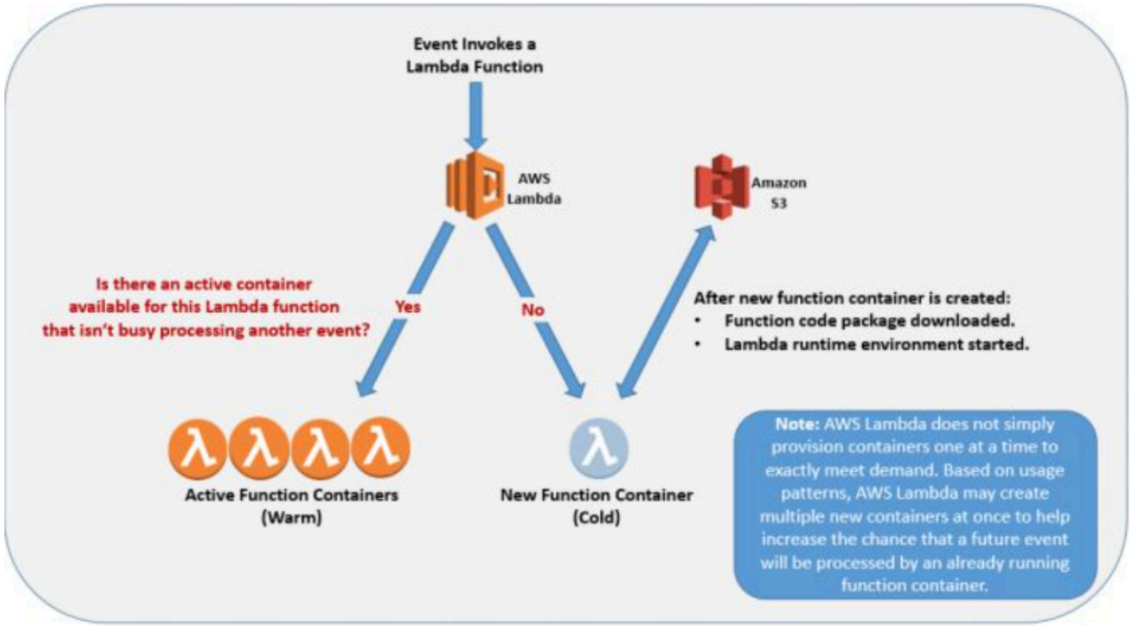
- The functions only execute after receiving an event
- The cloud providers manage the logic to trigger the functions
- The events have different behaviour in terms of scaling the functions
- The user only pays when the functions are requested



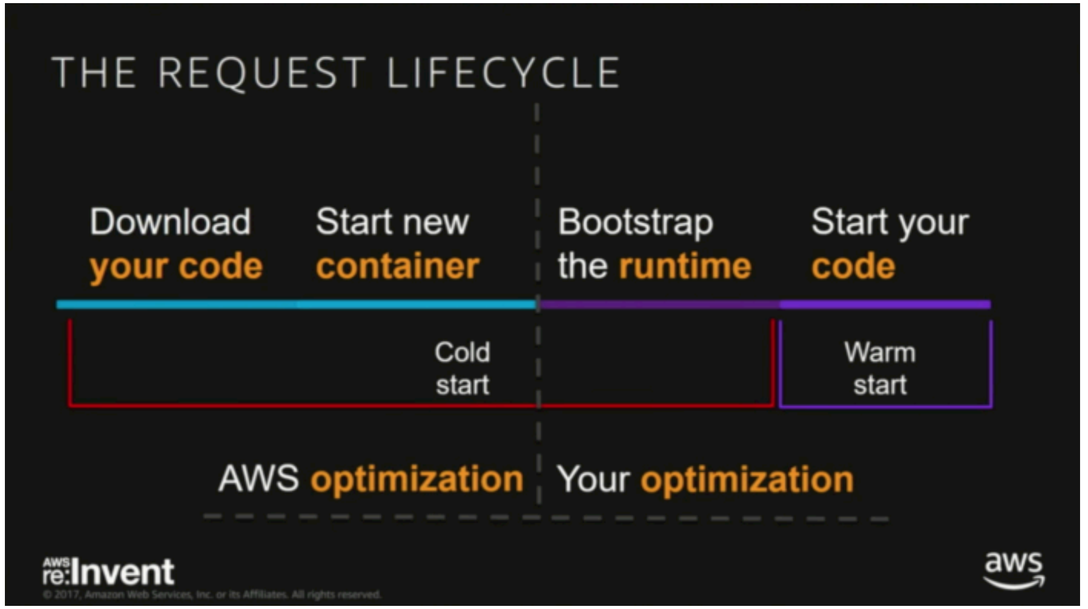
Messaging Fanout Pattern

3. Cold starts might impact several users

Cold starts



Cold Start



Request Lifecycle

Cold starts in AWS Lambda

- One function can only handle one request at a time
- Concurrent requests creates one cold start for each function that is not running
- Cold starts can have a great impact on the application performance

4. Load stuff you need on the startup

4. Load stuff you need on the startup

- The AWS Lambda does not charge the user by the cold start time
- Trick to optimize and reduce the function costs
- Lambda uses a lot of resources initialising the functions to reduce the cold start time
- The functions save the static variables and /tmp folder to following executions

5. Use a module bundler

5. Use a module bundler

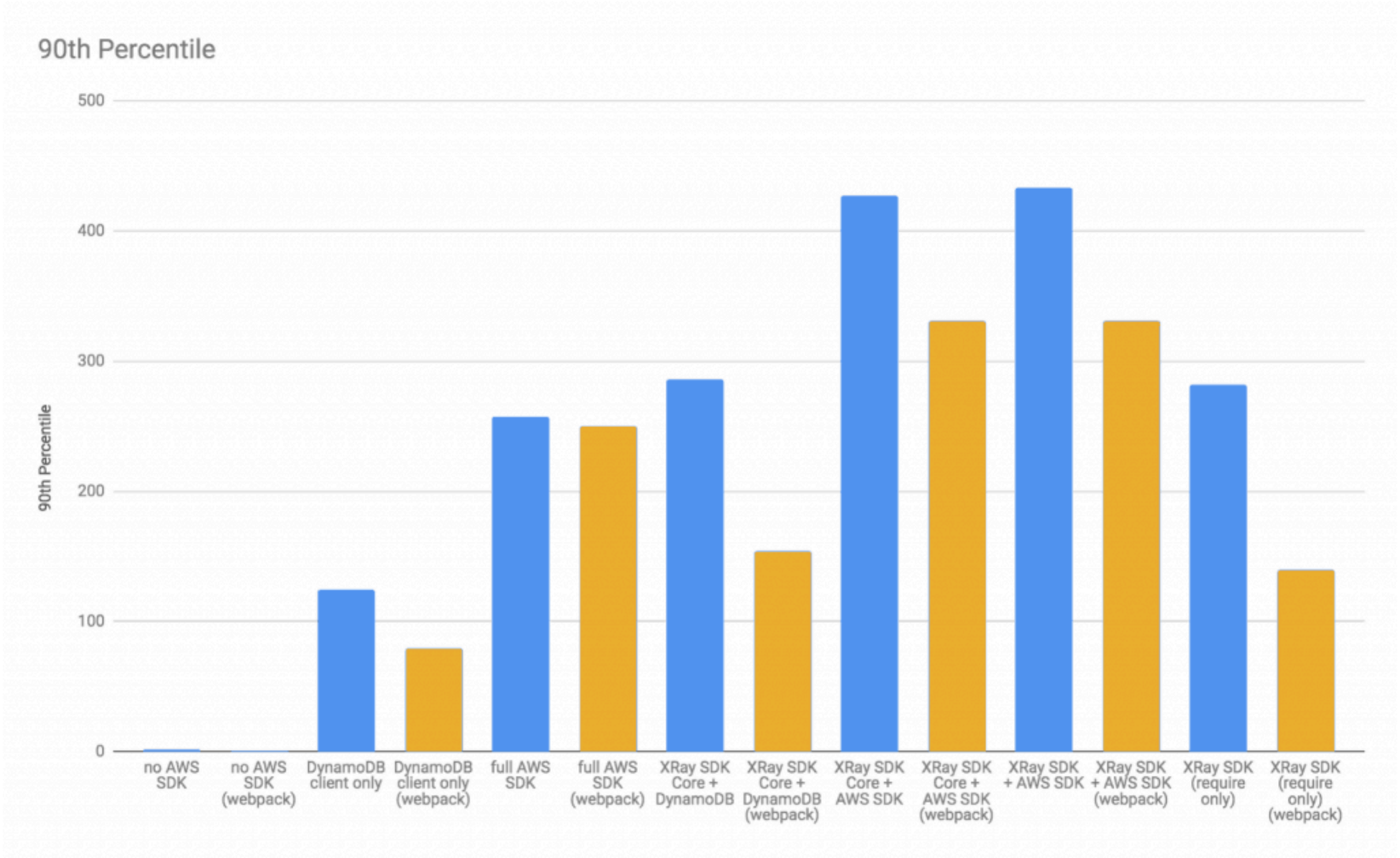
- Use a module bundler to create a bundle only with the required code
- Reduce the download size
- Improve the cold start of the function
- Reduce the memory required and reduce costs

6. Load the minimum dependencies into memory

6. Load the minimum dependencies into memory

- Reduce the amount of dependencies load into memory
- Import only the required modules in the application code
- Improve the cold start of the function
- Reduce the memory required and reduce costs

Performance Improvements



Just how expensive is the full AWS SDK by theburningmonk

7. Use middlewares and focus more on the business logic

7. Use middlewares focusing only on the business logic

- Use middleware to share the same code with another functions
- Use middlewares for correlationIds, logging, error handling, request parsing…
- Develop only on the business logic
- Examples: middyjs, lambda powertools or custom middleware

8. Use a tool to manage deploys

8. Use a tool to manage deploys

- Infrastructure as code
- Ease of deployment
- Run the application code locally
- Examples: Serverless Framework, Serverless Application Model (SAM), Cloud Development Kit (CDK)

9. Be careful when managing database connections

9. Be careful when managing database connections

- Hosted databases have a limited number of database connections
- Use database services managed by the cloud providers
- Serverless packages that consider the database connections numbers
- AWS provides a cloud proxy that manages the database connections

10. Use monitoring tools

10. Use monitoring tools

- In a production environment usually there’s a lot of components involved in the business flows
- Increase the debug capabilities by using monitoring tools
- The tools provide monitoring, tracing and alerting
- Examples: Amazon X-Ray, Epsagon, Datadog and Lumigo

Resources

- Demo project: https://github.com/belezebu/10-things-I-learned-using-serverless
- Talk resources:
  - https://www.martinfowler.com/articles/serverless.html
  - https://theburningmonk.com/2019/03/just-how-expensive-is-the-full-aws-sdk/
  - https://lumigo.io/blog/this-is-all-you-need-to-know-about-lambda-cold-starts/
  - https://hichaelmart.medium.com/shave-99-93-off-your-lambda-bill-with-this-one-weird-trick-33c0acebb2ea
  - https://docs.aws.amazon.com/whitepapers/latest/security-overview-aws-lambda/security-overview-aws-lambda.html

Thank you