Selçuk Cihan

- Boğaziçi Üni. Bilgisayar Müh. Bölümü 2008 mezunuyum
- AirTies, Amazon, Ziraat Teknoloji, Serverless Inc. gibi yerlerde çalıştım
- AWS ve Google Cloud üzerinde web uygulamaları geliştiriyorum
- Sunucu tarafı uygulamaları - backend üzerine uzmanlaştım


SERVERLESS COMPUTING

•••

AWS Serverless Ecosystem


Outline

- What is Serverless?
- Why Serverless?
- Key Serverless Components on AWS
- Serverless Architecture Design
- Limitations & Trade-offs
- Q&A - Discussion

What is Serverless?

- Wait, no servers??

There are servers, but you don't manage them!

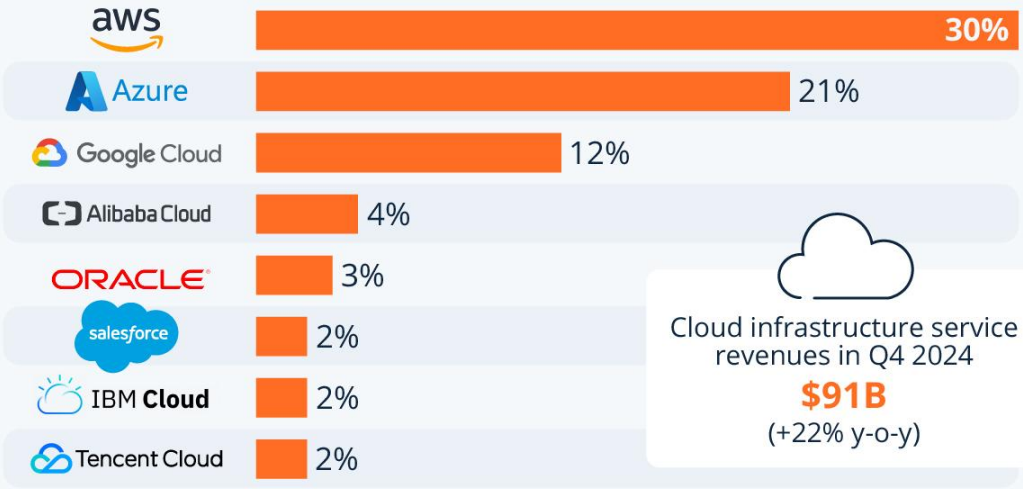
- Managed infrastructure
- Pay per use
- Auto-scaling

Examples of Serverless Platforms

- AWS Lambda (compute)
- Cloudflare Workers(compute)
- Google Cloud Run (compute)
- Azure Functions (compute)
- PlanetScale Database (storage)
- Auth0 (authentication provider)

Amazon and Microsoft Stay Ahead in Global Cloud Market

Worldwide market share of leading cloud infrastructure service providers in Q4 2024*



Cloud infrastructure service revenues in Q4 2024

$91B

(+22% y-o-y)

* Includes platform as a service (PaaS) and infrastructure as a service (IaaS) as well as hosted private cloud services

Source: Synergy Research Group

Characteristics of Serverless Workloads

- Runs infrequently
- Has highly variable (or unknown) scaling requirements
- Small and short-lived (AWS Lambda has a max 15 minute limit)
- Stateless functions

Stateless: As if your code executes in a fresh environment - scales well!

Stateful: Code hangs on to its variables and in-memory data - limited scaling

Why Serverless?

Because: LOWER TCO (Total Cost of Ownership)

- Hidden Costs
  - Security
  - Reliability
  - Scalability
  - Observability
- Visible Costs
  - Low economic barrier to entry
  - Very granular “pay for what you use” model

Bonus: LLMs + Coding assistants => 1-person teams

As our capabilities increase, it’s more crucial to have cloud computing expertise

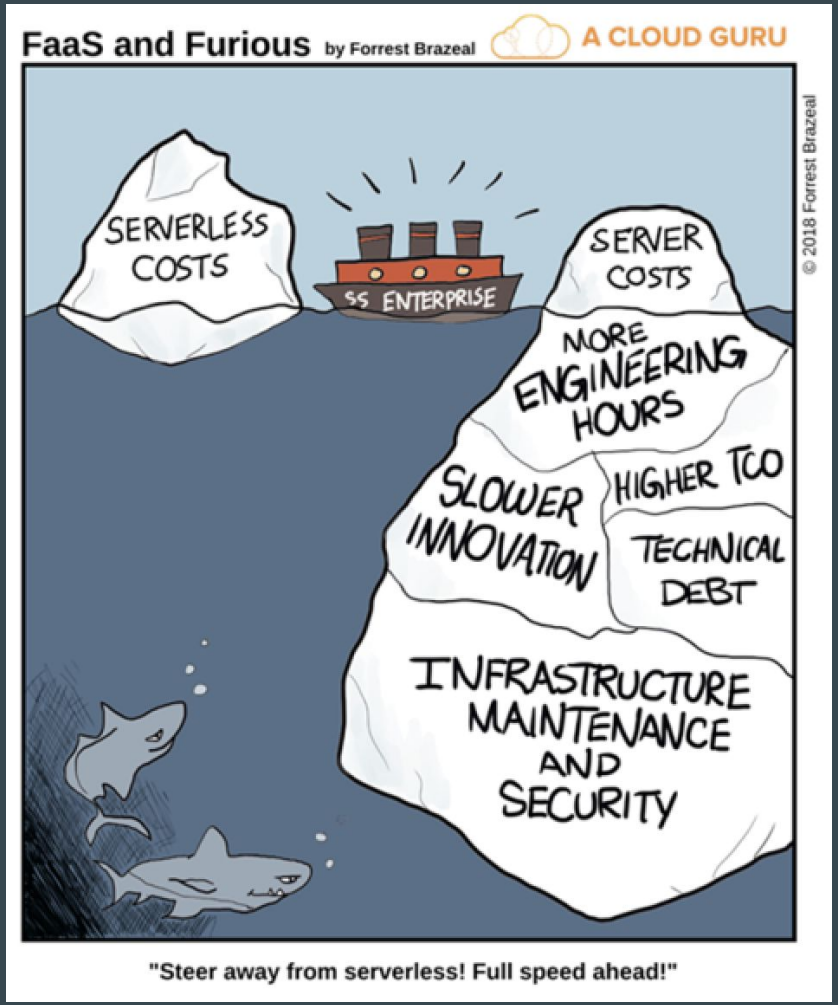
Total Cost of Ownership (TCO)

RENTING a car (serverless)
vs
OWNING a car (serverful)

Hidden costs of owning a car

- Service maintenance
- Insurance
- Tax
- Depreciation (wear and tear)

Push responsibilities and ownership to the cloud as much as possible!



"Steer away from serverless! Full speed ahead!"

Total Cost of Ownership (TCO)

1. Development: Salaries of developers, project management, tools, licenses etc.
2. Deployment: Cloud costs like AWS or Azure or own data center
3. Maintenance & Support: bug fixes, security patches, managing storage etc.
4. Reliability & Robustness: multi AZ, redundancy, backups, disaster recovery
5. Compliance: GDPR, HIPAA, ISO & other regulations
6. Security: Network & system security, vulnerability analysis, encryption
7. Depreciation & Decommissioning: Cost of infra aging over time (both hw & sw)
8. Migrations: Replacing outdated technologies, migrating data etc.

Key Components on AWS: Compute

- Lambda: Take this code and run it!
- API Gateway + AppSync
  - To serve REST or GraphQL APIs
- Step Functions
  - Orchestrate services to create workflows

AWS Lambda Runtimes

- NodeJS
- Python
- Java
- .Net
- Ruby
- Go
- Rust

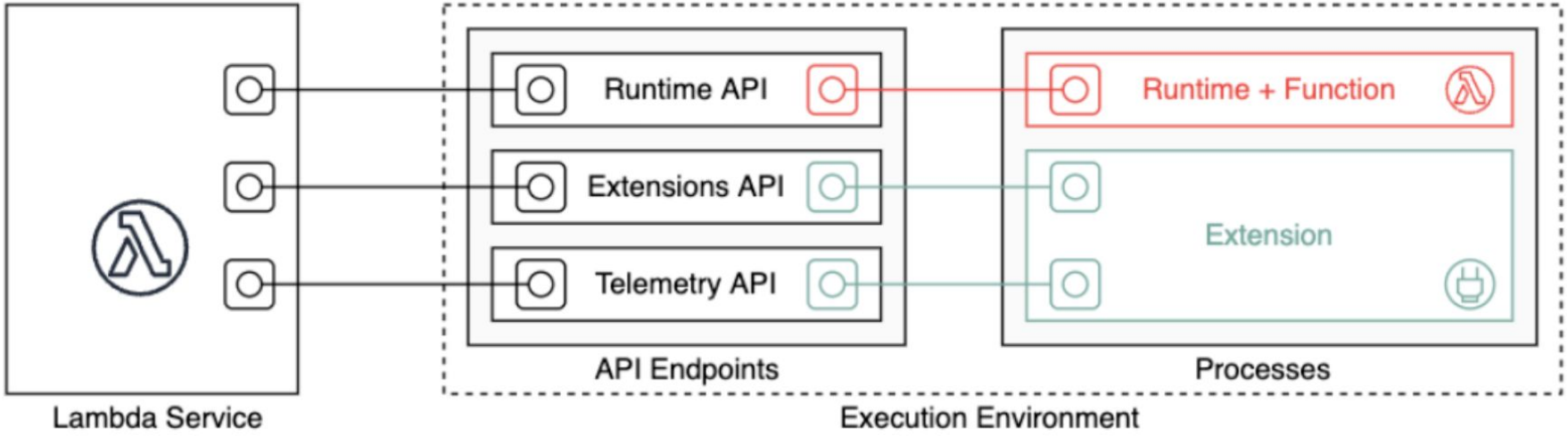
AWS Lambda Packaging

You can package your code in two ways:

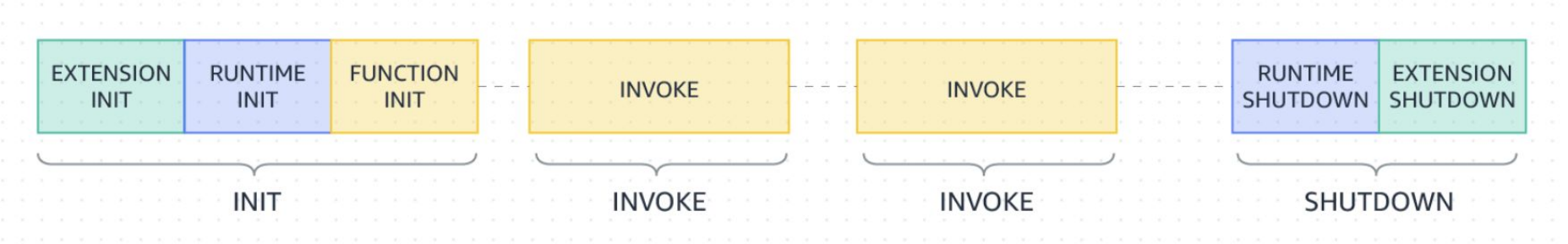
1. As a .zip file: faster, less customizable

2. As a docker container image: slower, flexible

AWS Lambda Model

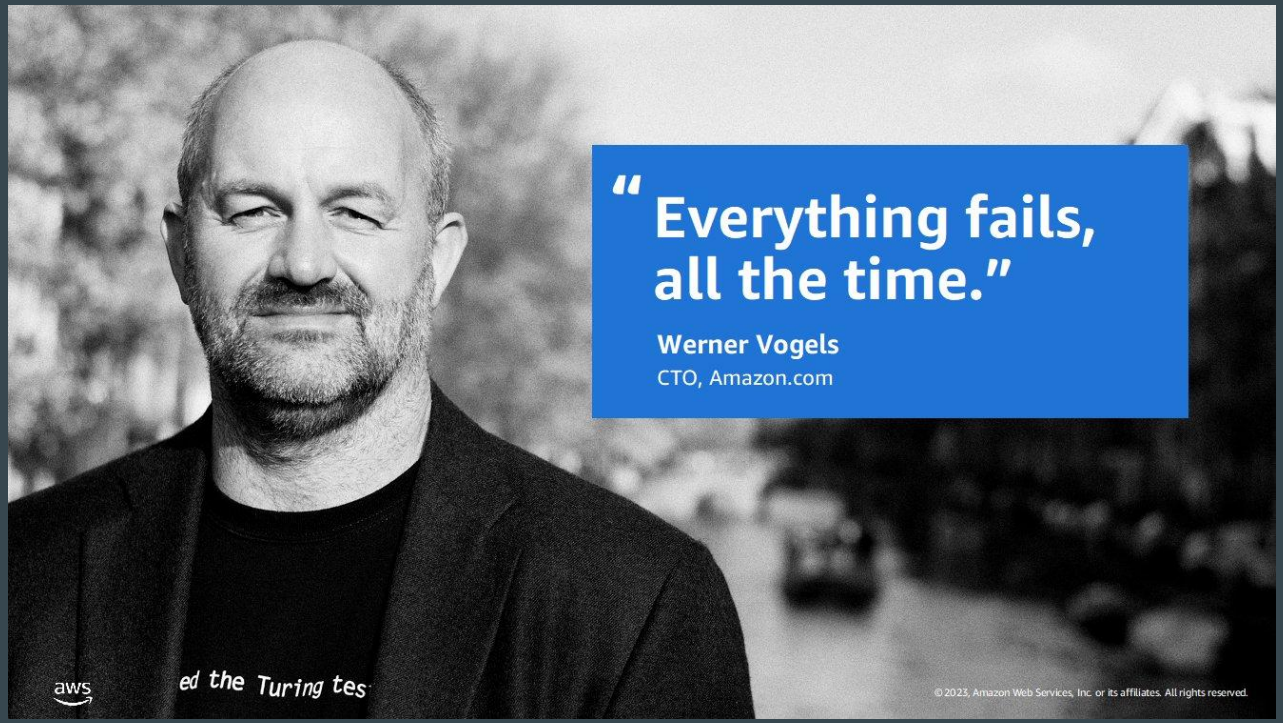


AWS Lambda Lifecycle



1. Init: Runtime & static code initialization
2. Invoke: Your code runs with the input event
3. Shutdown:
    - AWS Lambda maintains execution environment after an invocation for some time.
    - AWS Lambda can terminate environment at any point

AWS Lambda Error Handling



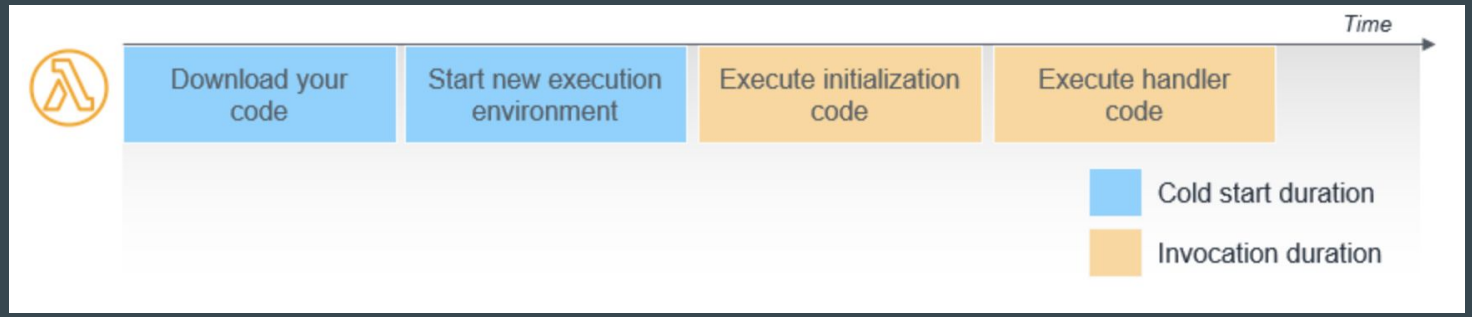
Idempotency & Error Tolerance

- AWS Lambda may retry your function randomly

- Idempotency will guarantee that there are no unintended side-effects (eg. instead of creating a single user, duplicate users might be created)
- Write your systems to be self-healing and error tolerant!

~ERRORS WILL HAPPEN~
BE PREPARED!

AWS Lambda Cold Starts

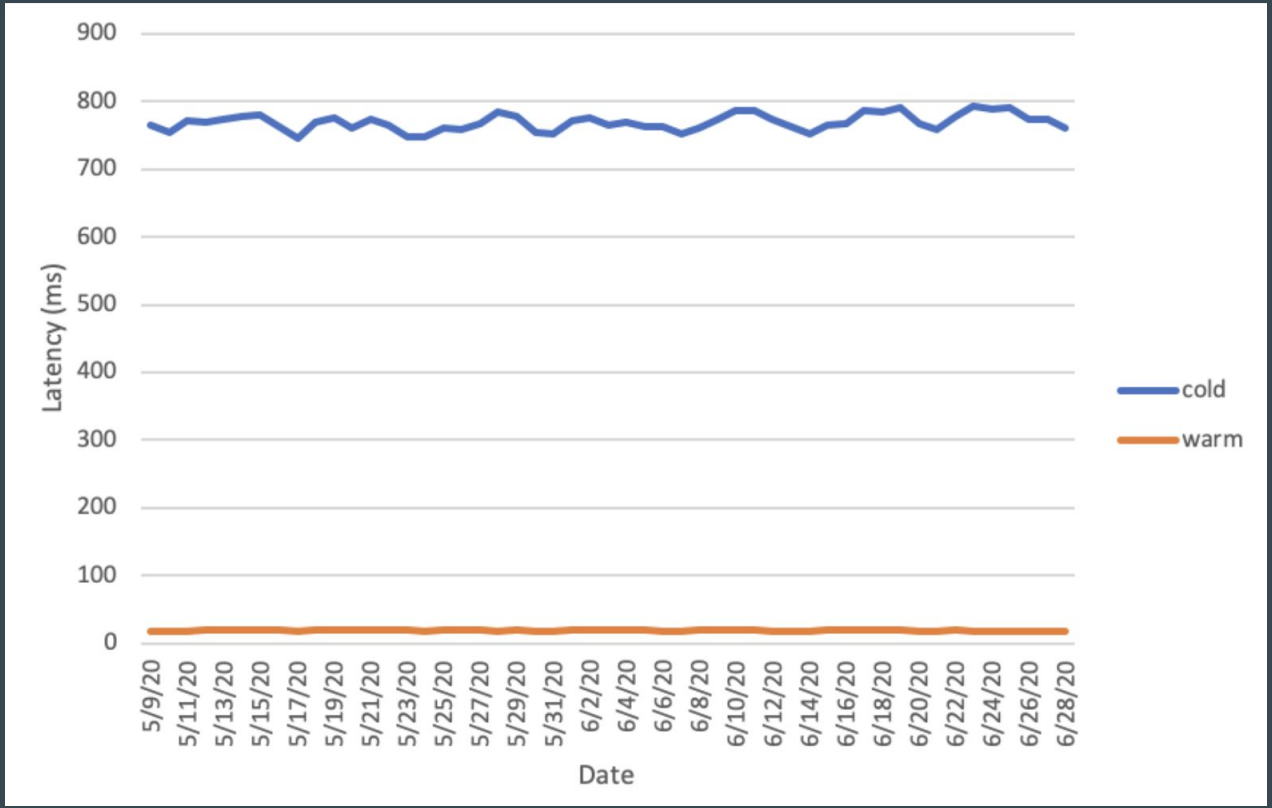


AWS Lambda may recycle your environment at any time.

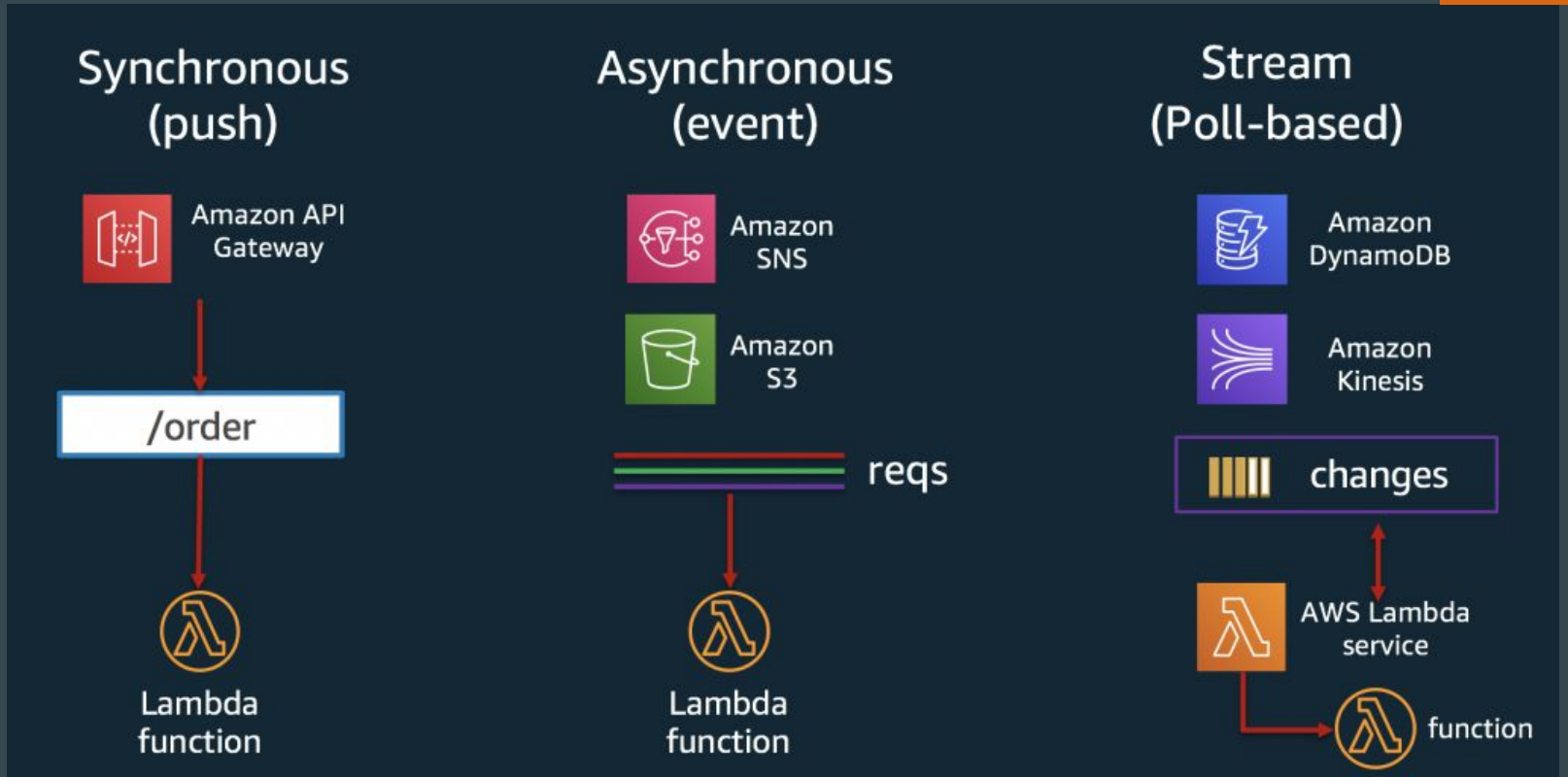
New invocation may need a new environment to be spun up.

It may take anywhere between a few hundred milliseconds to a few seconds.

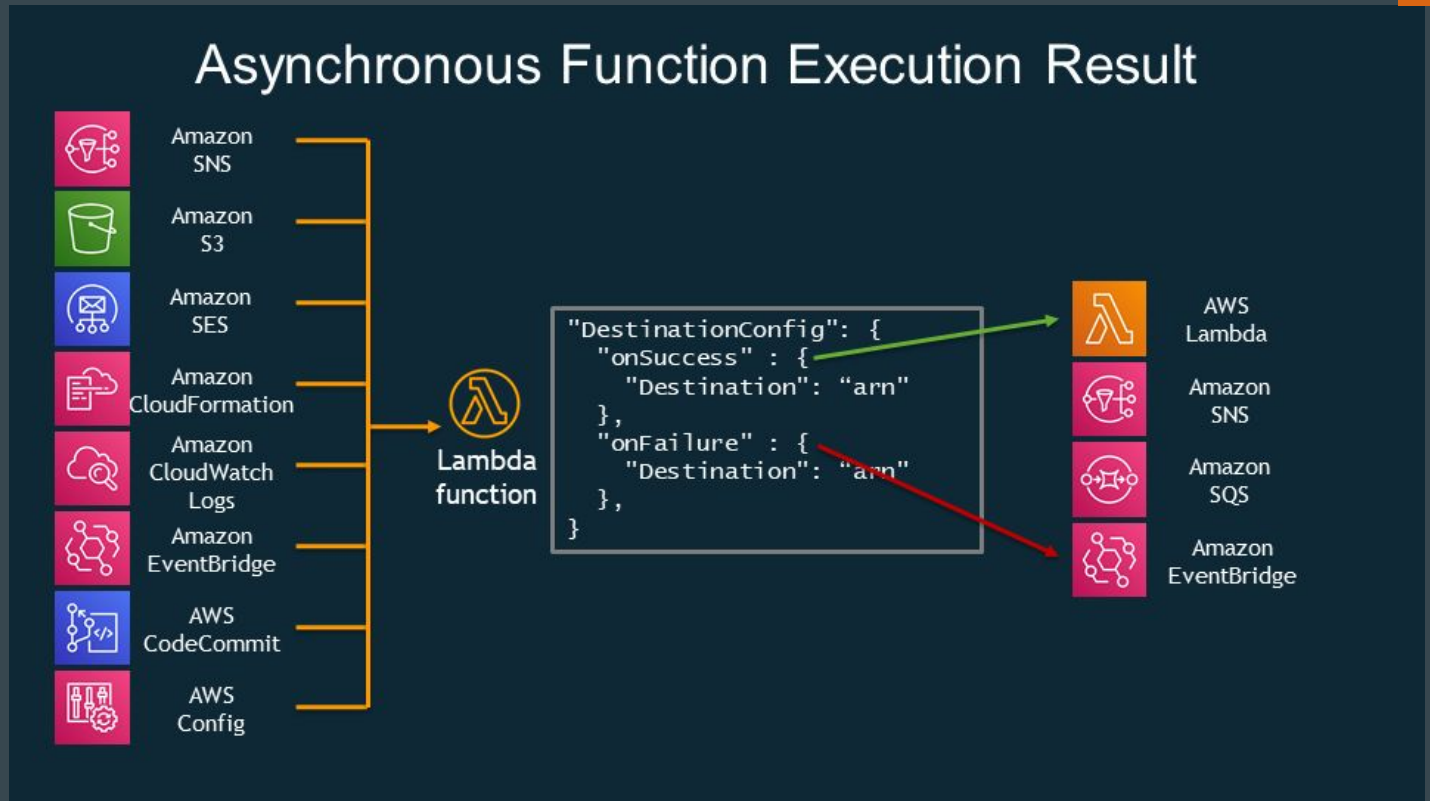
AWS Lambda Cold Starts



AWS Lambda - Sync vs. Async

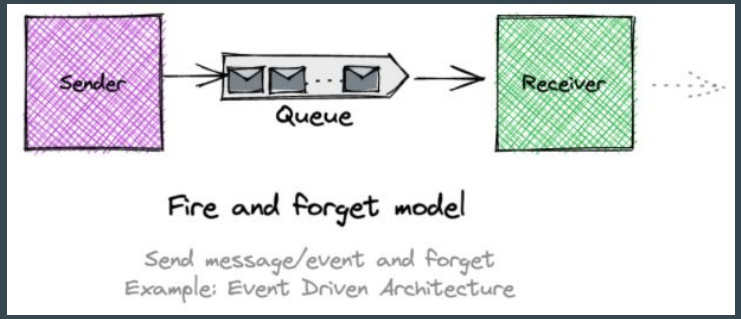


AWS Lambda Async Invocation



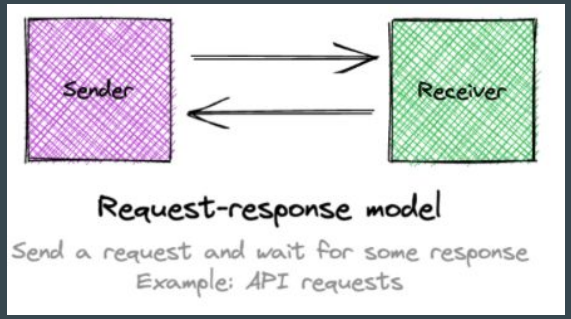
Asynchronous 👍

- Less coupling between services
- Scales better
- You don’t wait for a response
- More flexibility
- Difficult to grasp & debug



Synchronous 👎

- Highly coupled interactions
- Scaling is limited due to bottlenecks
- You have to wait for the response
- Less flexible
- Easier to reason about & debug



API Gateway & AppSync

API Gateway provides authentication, caching, monitoring, security.

With API Gateway, you can create a REST API

AppSync is for GraphQL, which is a popular interface between mobile devices & servers

With AppSync, you can push data to the client from the server directly (websockets)

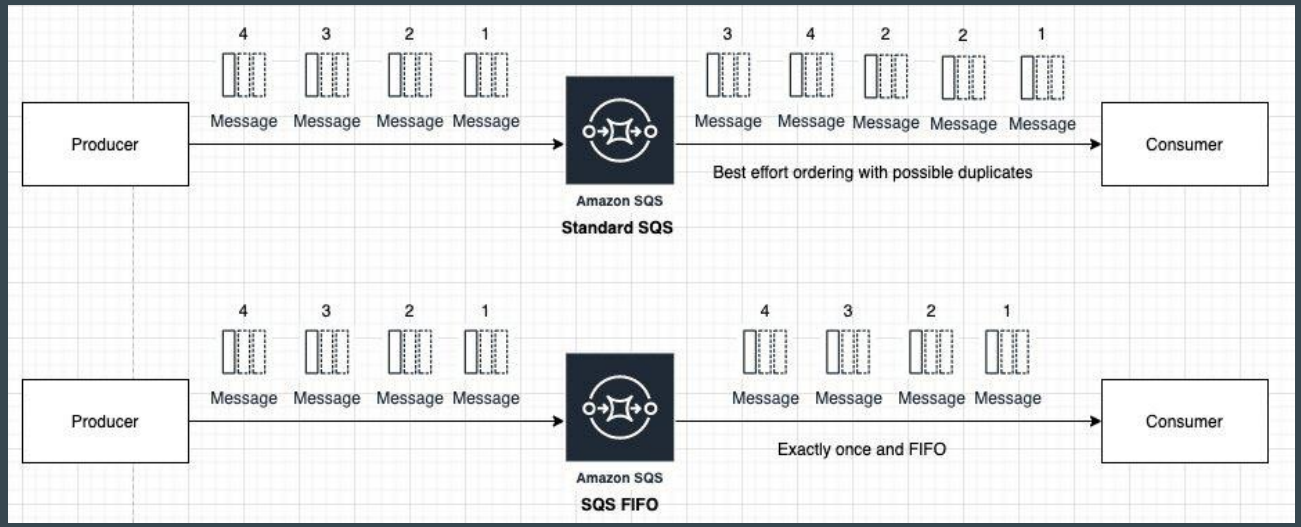
Key Components on AWS: Storage

- S3 (Object Storage)
  - Think as a key - value store
  - You can store large media files or a JSON document
  - S3 Notifications => AWS Lambda
- DynamoDB (NoSQL)
  - Blazing fast (single digit millisecond latency!)
  - Scales to zero (no cost if no requests)
  - DynamoDB streams let you create serverless event-driven apps
- Aurora Serverless (SQL)
  - SQL database that scales up and down in less than a second

Key Components on AWS: Integration

SQS (Simple Queueing Service)

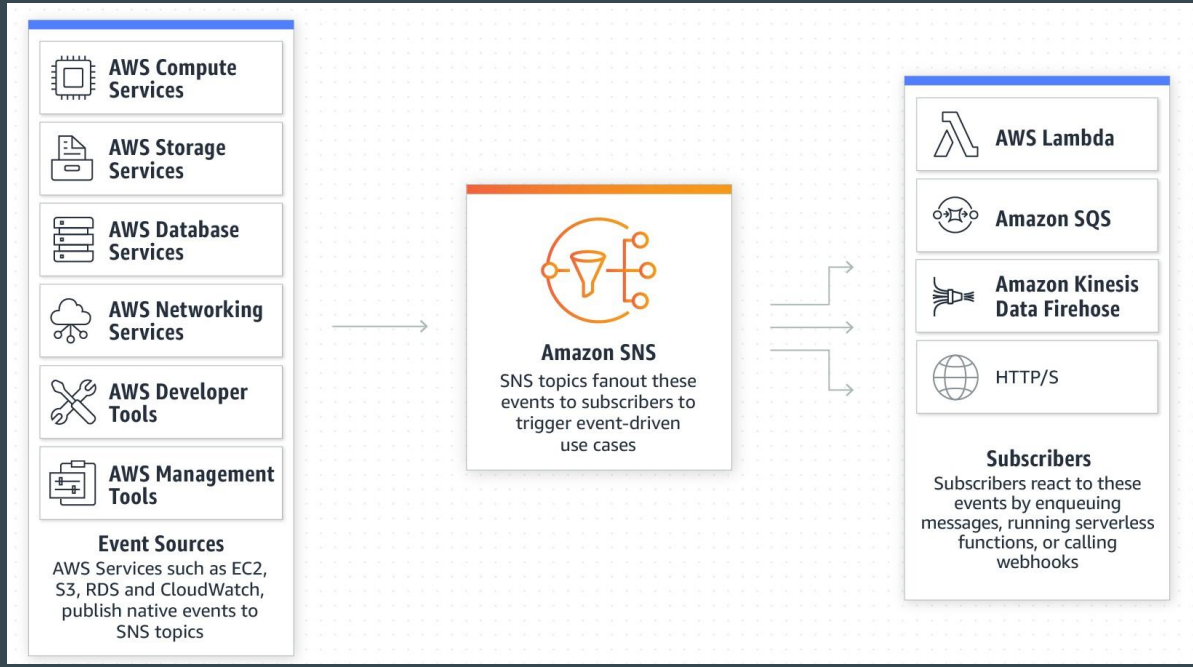
- Enqueue / Dequeue / Poll
- Oldest service on AWS!
- Durable Queue (retention of up to 14 days)
- Can act as a buffer between services

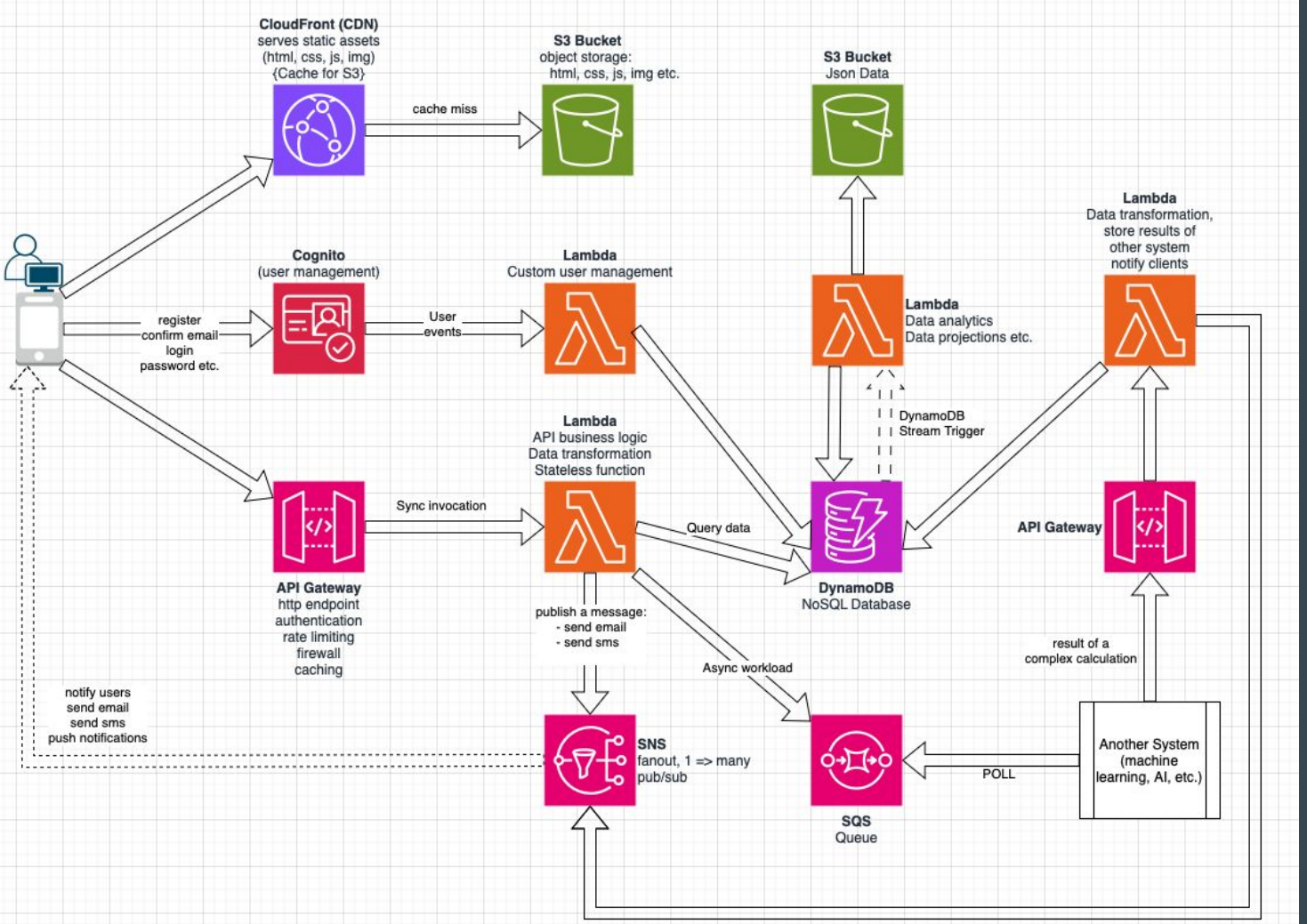


Key Components on AWS: Integration

SNS (Simple Notification Service)

- Publisher / Subscriber model
- Can send email, sms
- Push notifications to mobile




CloudFront (CDN)
serves static assets
(html, css, js, img)
{Cache for S3}
cache miss
S3 Bucket
object storage:
html, css, js, img etc.
S3 Bucket
Json Data
Lambda
Data transformation,
store results of
other system
notify clients
Cognito
(user management)
register
confirm email
login
password etc.
User
events
Lambda
Custom user management
Lambda
Data analytics
Data projections etc.
DynamoDB
Stream Trigger
Lambda
API business logic
Data transformation
Stateless function
Sync invocation
Query data
API Gateway
http endpoint
authentication
rate limiting
firewall
caching
DynamoDB
NoSQL Database
API Gateway
publish a message:
- send email
- send sms
Async workload
result of a
complex calculation
notify users
send email
send sms
push notifications
SNS
fanout, 1 => many
pub/sub
SQS
Queue
POLL
Another System
(machine
learning, AI, etc.)

Limitations

- Cold starts (provisioned concurrency, keep functions warm)
- Execution time: Lambda has max 15 minutes execution time
- Vendor lock-in ???
- Debugging is a bit difficult to get used to
- Expensive at scale
- 512mb ephemeral storage (/tmp)

Trade-offs

- Operational overhead vs. Ability to control
- Cost efficiency vs. Predictable costs (pay for idle vs expensive)
- Scaling
- Performance & responsiveness
- Flexibility vs lock-in
- Security & compliance (automatic patches, vs control over security policies and fine-grained network access)

Thank You

You can find me on

- email: selcukcihan@gmail.com
- selcukcihan.com
- twitter(x): @scihan

References and Links

- https://vercel.com/docs/functions
- https://aws.amazon.com/serverless
- https://cloud.google.com/serverless
- https://workers.cloudflare.com/
- https://auth0.com/
- https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/
- https://imgflip.com/i/7lmb41
- https://docs.aws.amazon.com/lambda/latest/dg/lambda-runtime-environment.html
- https://blog.symphonia.io/posts/2020-06-30_analyzing_cold_start_latency_of_aws_lambda
- https://docs.aws.amazon.com/lambda/latest/dg/lambda-services.html
- https://aws.amazon.com/blogs/architecture/understanding-the-different-ways-to-invoke-lambda-functions/
- https://aws.amazon.com/blogs/compute/introducing-aws-lambda-destinations/
- https://eda-visuals.boyney.io/visuals/sync-vs-async
- https://aws.amazon.com/sns/features/
- https://aws.amazon.com/blogs/architecture/application-integration-using-queues-and-messages/