Infrastructures et Stockage et de Traitement de Données - IST

Leonard Cseres | May 24, 2025

Serverless

- Zero server ops (auto-scaling, no provisioning)
- No compute costs when idle
- Stateless
- Asynchronous, concurrent, easy to paralellize

Structure

- Initialization code: code outside the handler method
- Local storage: `/tmp`
- `event`: information about the event
- `context`: func. name, req. id, deployment params, remaining time, env

Deployment Parameters Memory, CPU, Timeout, Concurrency, Environment variables

Pricing

- Resources consumed (memory × CPU)
- Number of invocations
- Ingress/egress traffic

Lifecycle

- Outside of these phases the platform freezes the execution environment
- After invocation, the platform will keep the execution environment for some time for optimizing
- The function must be stateless, but the reuse of the exec. env. can be used for caching
  - Objects outside the handler function
  - `/tmp`
  - Background procs. or callbacks initiated by the function and did not complete when it ended resume if the platform reuses the execution environment

Permissions

(1) Policies allowing other AWS services to invoke the function

```
Version: "2012-10-17"
Statement:
  - Effect: "Allow"
    Principal:
      Service: "apigateway.amazonaws.com"
    Action: "lambda:InvokeFunction"
    Resource: "*"
```

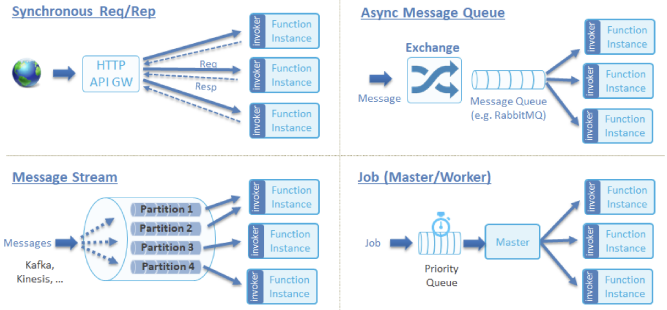
(2) Policies allowing the function to access other services

```
Version: "2012-10-17"
Statement:
  - Effect: "Allow"
    Action: ["s3:GetObject", "s3:ListBucket"]
    Resource:
      ["arn:aws:s3:::your-bucket-name/*",
           "arn:aws:s3:::your-bucket-name"]
```

S3 Object Lambda

- Use cases: redaction, conversion, augementation, compression, resizing, watermarking, custom row level authorization

S3 Access Point Alias for a bucket with custom permissions



S3 Object Lambda Access Point Enhanced S3 access point

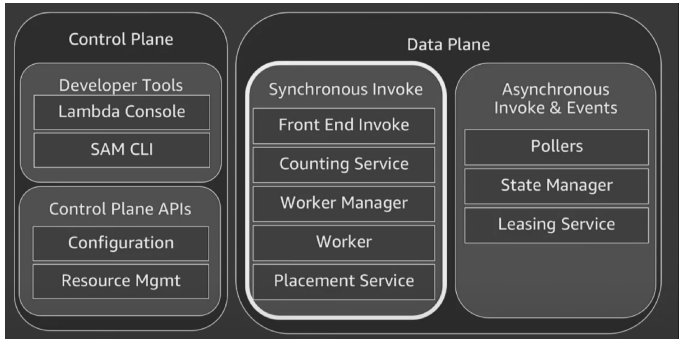
Configured with:

- Standard S3 access point
- AWS Lambda transform function
- (optional) IAM policy to restrict access to this accesss point

Cold Start Problem Mitigations

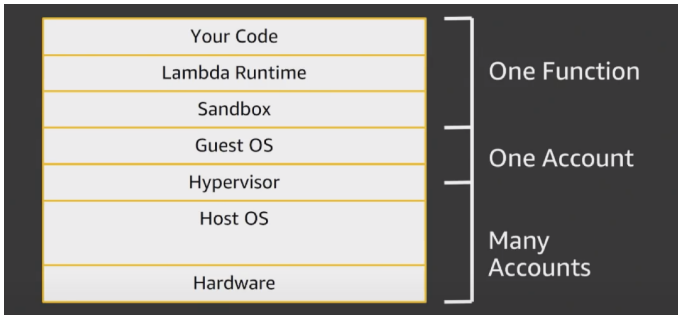
- Not only in FaaS, also in auto-scaling
- Reduce dependencies and optimize function initialization
- Fixed allocation of a set of VMs to run the function

FaaS platform



- Control plane: developer-facing API
- Data plane: underlying infrastructure

1. Call hits the LB
2. Frontend Invoke (entrypoint, retrieves metadata)
3. Counting Service (count invocations and concurrent functions)
4. Worker Manager (manages the instances, sandboxes, lifecyle)
5. Worker (compute instance, running the sandboxes)
6. Placement Service (optimizes the intelligent placement of the function, minimize the cold start, ensure optimal utilization, moniors health of workers)



Functions Security Functions are isolated from each other

- **Environment isolation**: customize the environment without affecting other
- **Secutity isolation**: data is not shared
- **Performance isolation**: "noisy neighbors" should not infer with other

Sandboxes are built from the same techo. as Docker

- One Sandbox executes several function invocations in serial
- A Sandbox is never reused across several functions

Hundreds or thousands of Guest OS's may run on a physical host

- Guest OS's are shared within an account
- A Guest OS is never reused across accounts.