Serverless Compute

Compute Options

Server-based

- EC2
- EC2 + Containers (ECS, EKS)

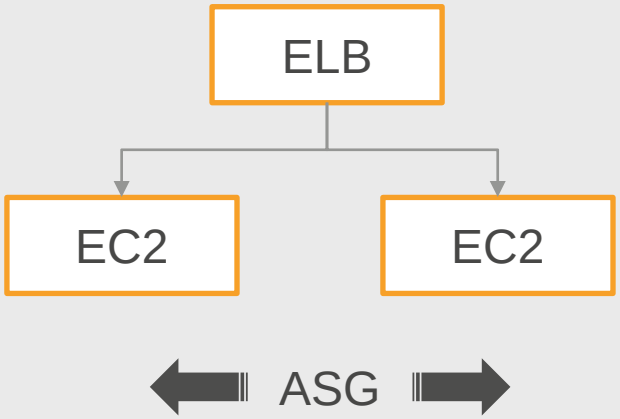
Serverless

- Fargate (ECS, EKS)
- Lambda

Shopping Cart

- Add Item
- Delete Item
- Update Item
- Apply Discount Coupon
- Checkout

EC2 - Host web application



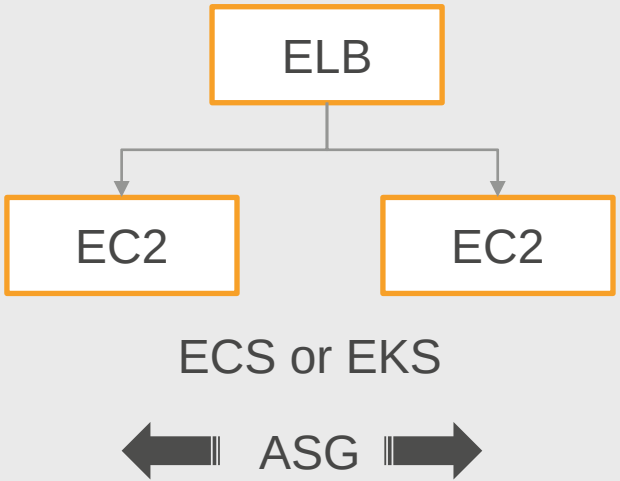
Infrastructure always running

Need to pay even when there is no traffic

- EC2
- ELB

This is an example of Server-based compute

Containers



Containers – Package application as self contained pods

Deploy reliably and consistently

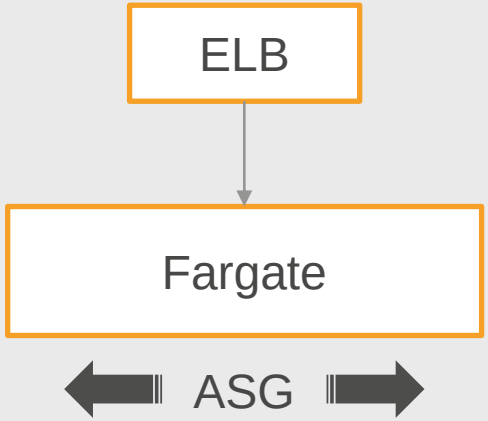
Elastic Container Service, Elastic Kubernetes Service

Infrastructure always running (EC2, ELB)

Another example of server-based compute

Need to pay even when there is no traffic

Fargate Container Service (ECS, EKS)



Fargate Serverless Compute for Containers

Specify CPU and Memory needed

Fargate provisions resources for containers - No Servers to manage

Example of Serverless-compute

Hourly charge for CPU and Memory

Scheduled Tasks – Pay only when the task is running

Web App - Need to pay even when there is no traffic

Lambda

Function as a Service

Lambda Functions – Shopping Cart

Add Item

Delete Item

Update Item

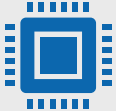
Apply Coupon

Checkout

Lambda

Create functions

Specify memory needed for function



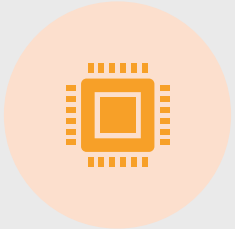
Lambda provisions resources with requested memory and proportional CPU

No servers to manage

Automatically scales and fault tolerant (multi-AZ)

Lambda Billing

Pay only when code is running – no need to pay for idle infrastructure



Pricing based on memory, and duration (GB-Seconds)

Data Transfer

Number of requests (20 cents per million requests)

Millisecond Metering (New for 2021)

Example - Lambda function took 130 milliseconds to complete

Billed Duration

200 ms (Before)

130 ms (Now)

Compute Savings Plan (up to 17% discount with 1 year or 3-year terms)

Lambda Usage

Perfect for interactive and scheduled tasks

Over 140 AWS services can invoke Lambda functions (API Gateway, S3, SNS, SQS, EC2,..)

Lambda Function Code

- Interactive Editor
- Zip file
- Containers (New for 2021)

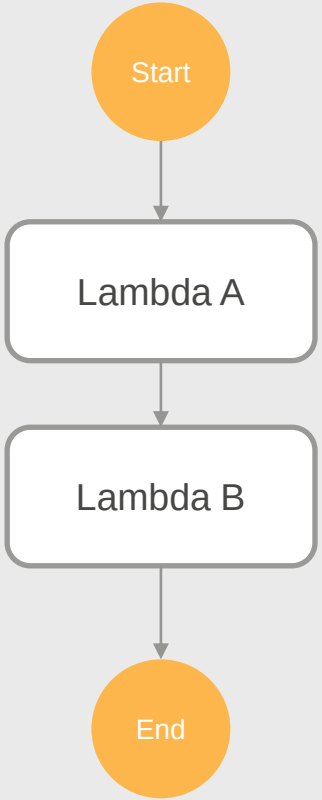
Run Time Limit

Timeout needs to be specified

Default is 3 seconds

Max duration of function execution is 15 minutes

Use Step Functions to Orchestrate



- Use Step Functions for long running workflows
- Build your app using smaller Lambda functions
- Orchestrate using Step Functions

Function Size Limit

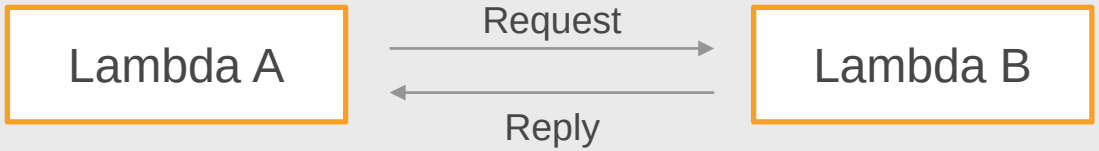
Max Size - Lambda Zip Archive, Interactive Coding

- 250 MB (uncompressed)
- 50 MB (compressed)

Max Size - Lambda Containers (New for 2021)

- 10 GB

Billing – Things to know



1. Lambda B Duration = 500 ms
2. Lambda A Duration = Waits for Lambda B + 100 ms = 600 ms
3. Total Elapsed Time = 600 ms
4. Billing duration = Lambda A + Lambda B = 600 + 500 = 1100 ms

Don't wait for another Lambda function to complete – you will end up paying for wait time

Use Step Functions when a workflow requires invoking multiple Lambda functions

Lambda Cannot Listen on Ports

Lambda function cannot listen on ports waiting for inbound connection

Lambda allows outbound connections for talking to other services

Run Time Support

Support for many languages

- Node.js, Python, Go, Java, .NET, Ruby, and more

Configure a custom runtime to use other languages

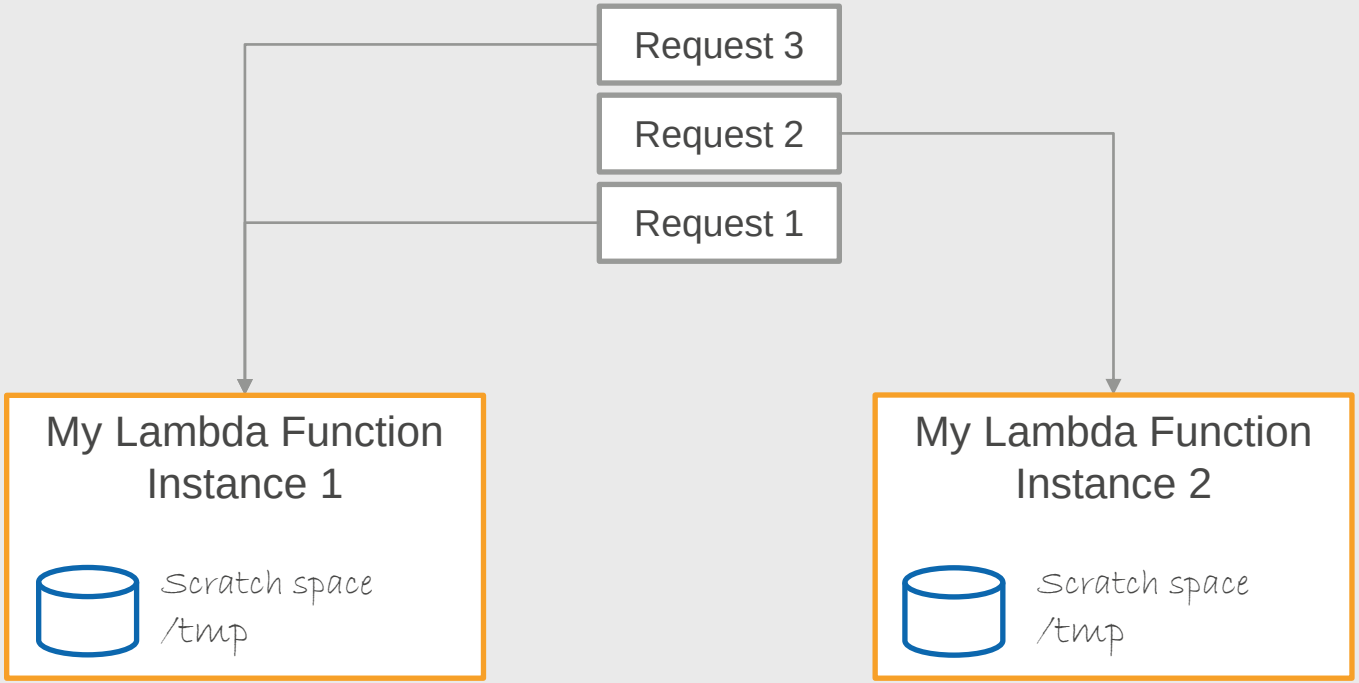
Lambda Scaling

Lambda automatically scales

Supports a variety of workloads

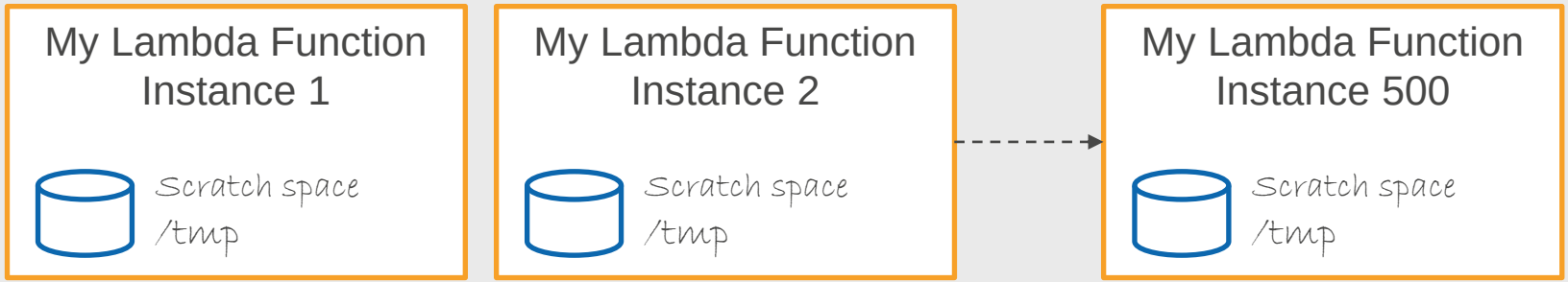
- Few requests per day
- 1000s of requests per second

Lambda Concurrency



A function instance processes one request at a time and can process many requests one after another

Lambda Scaling Characteristics



- Deploys 500 function instances for initial burst!
- Some regions deploy up to 3000 instances
- Additional 500 instances per minute until concurrency limit is reached

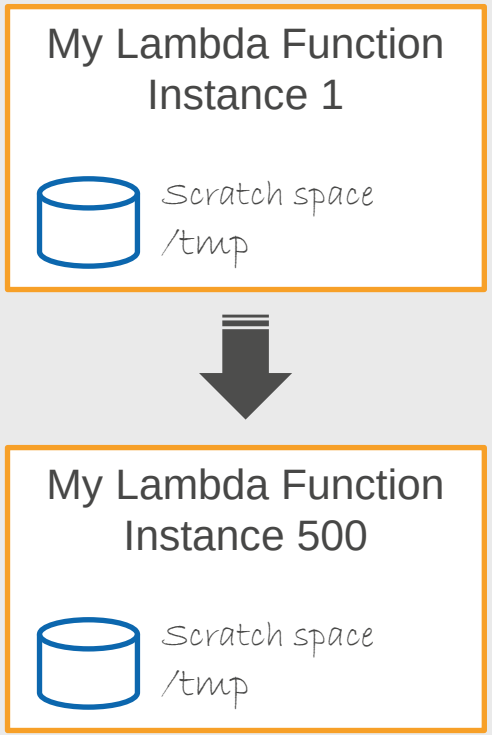
Max Concurrency

Limit number of concurrent instances

Configured using “Reserved concurrency” setting

Useful when backend system has scaling challenges

Application Requests Per Second



Example Calculation

1. Function takes 300 ms to process a request
2. Per second = 3.3 requests [1000/300]
3. With 500 Lambda instances, your app can support 3.3 X 500 = 1,600 requests per second
4. Throttling error when requests exceed concurrency limit
5. Soft limit – number of instances quota can be increased

Cold Start

Lambda scales the number of function instances based on traffic

Your function instances are removed when there is no traffic

To process new request, Lambda must initialize new instances – slowness in response (Cold Start)

Provisioned Concurrency

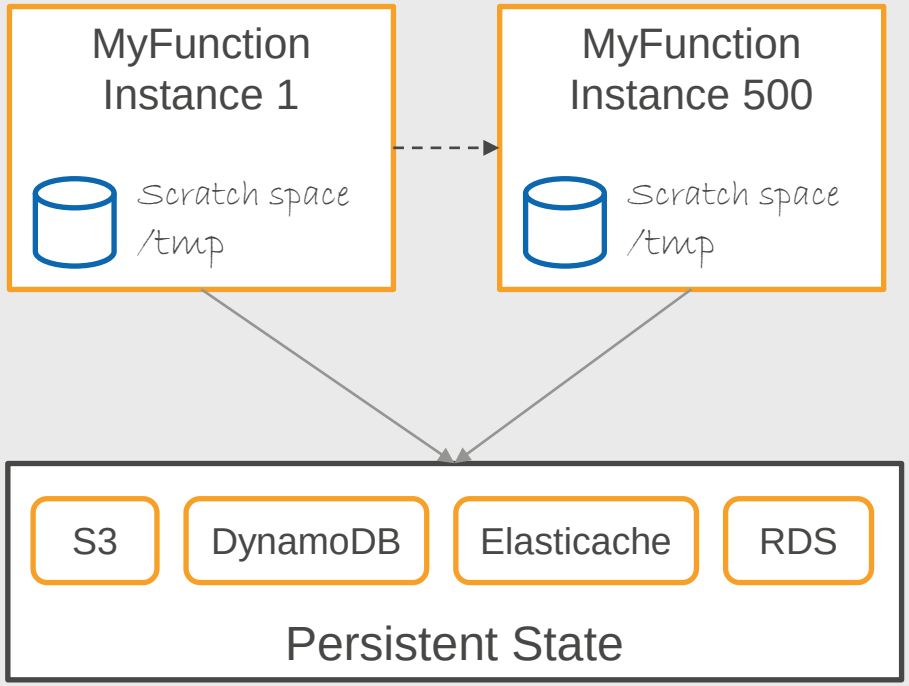
For Latency sensitive applications, keep lambda instances always ready!

Configured using “Provisioned concurrency” setting

Use AutoScaling to dynamically adjust provisioned concurrency

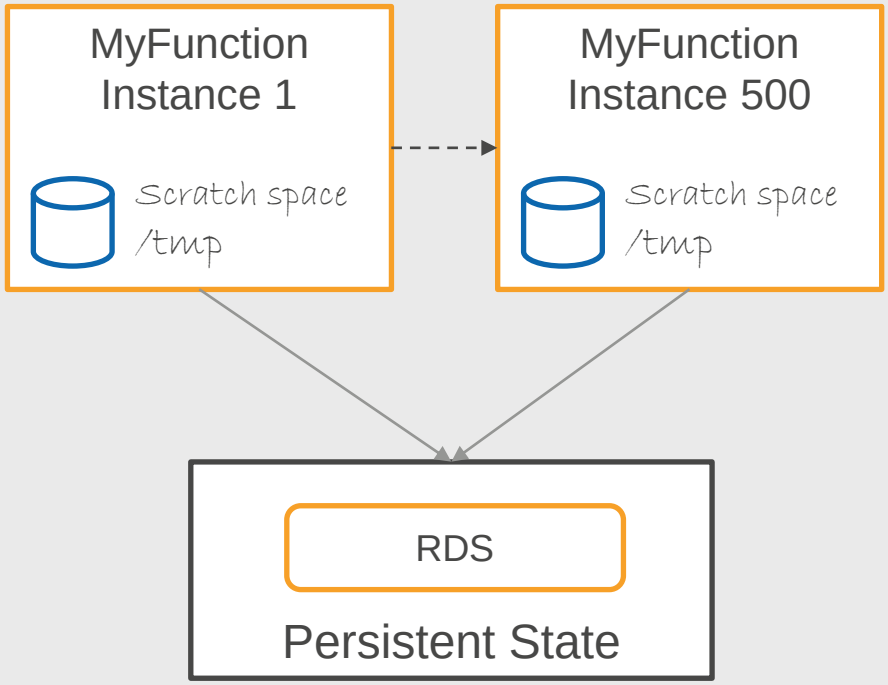
Additional charges apply

Stateless Design



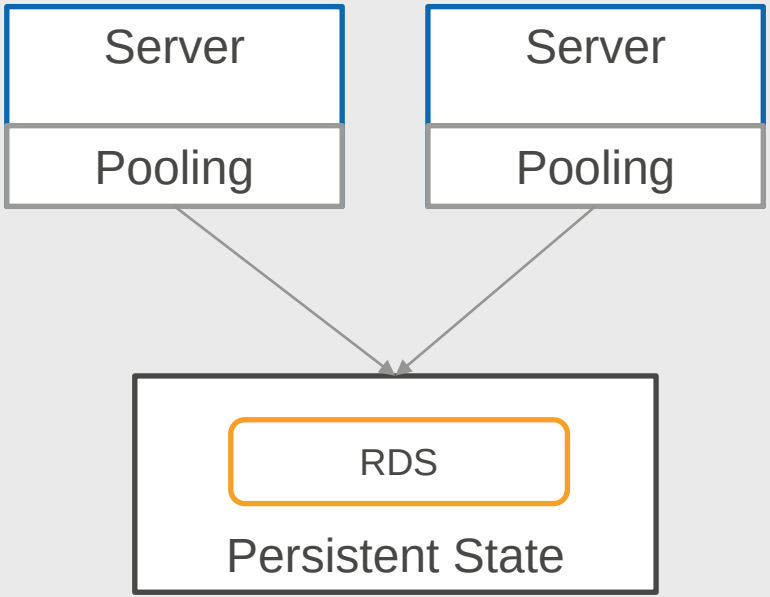
1. Request processed by any available instance
2. Function code should not assume any affinity to underlying compute infrastructure (stateless)
3. Don’t use /tmp to store user-state
4. For maintaining user-state information, use external store
5. Stateless design allow rapid scaling when needed

Lambda - RDS Connection



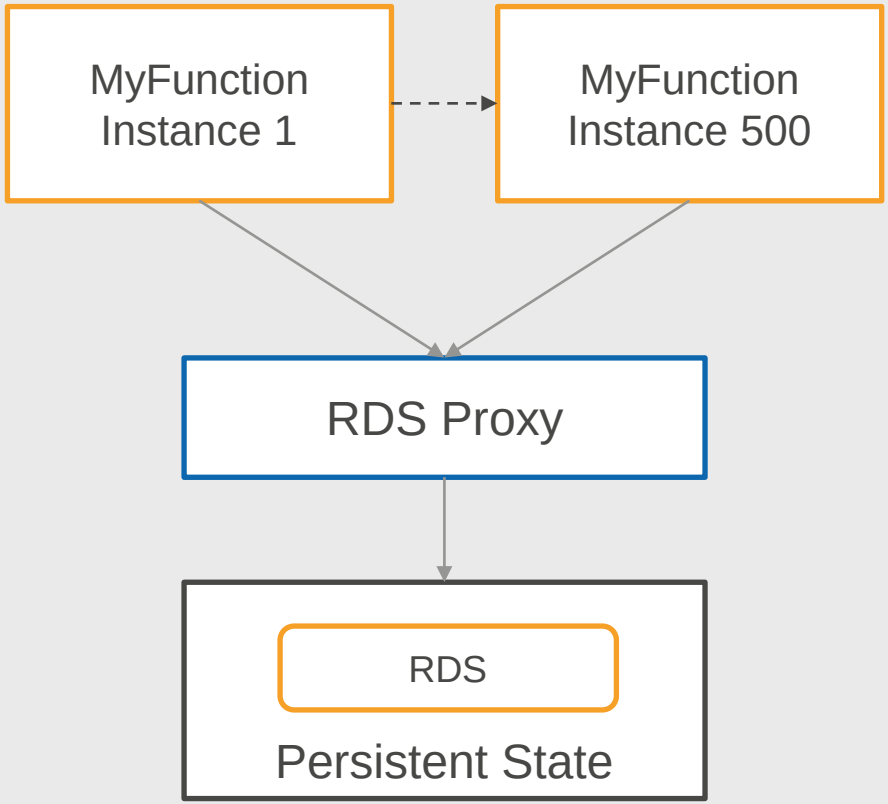
1. Large number of open connections to database
2. Open and Close at a high rate
3. Exhaust database resources

Server-based Connection Pooling



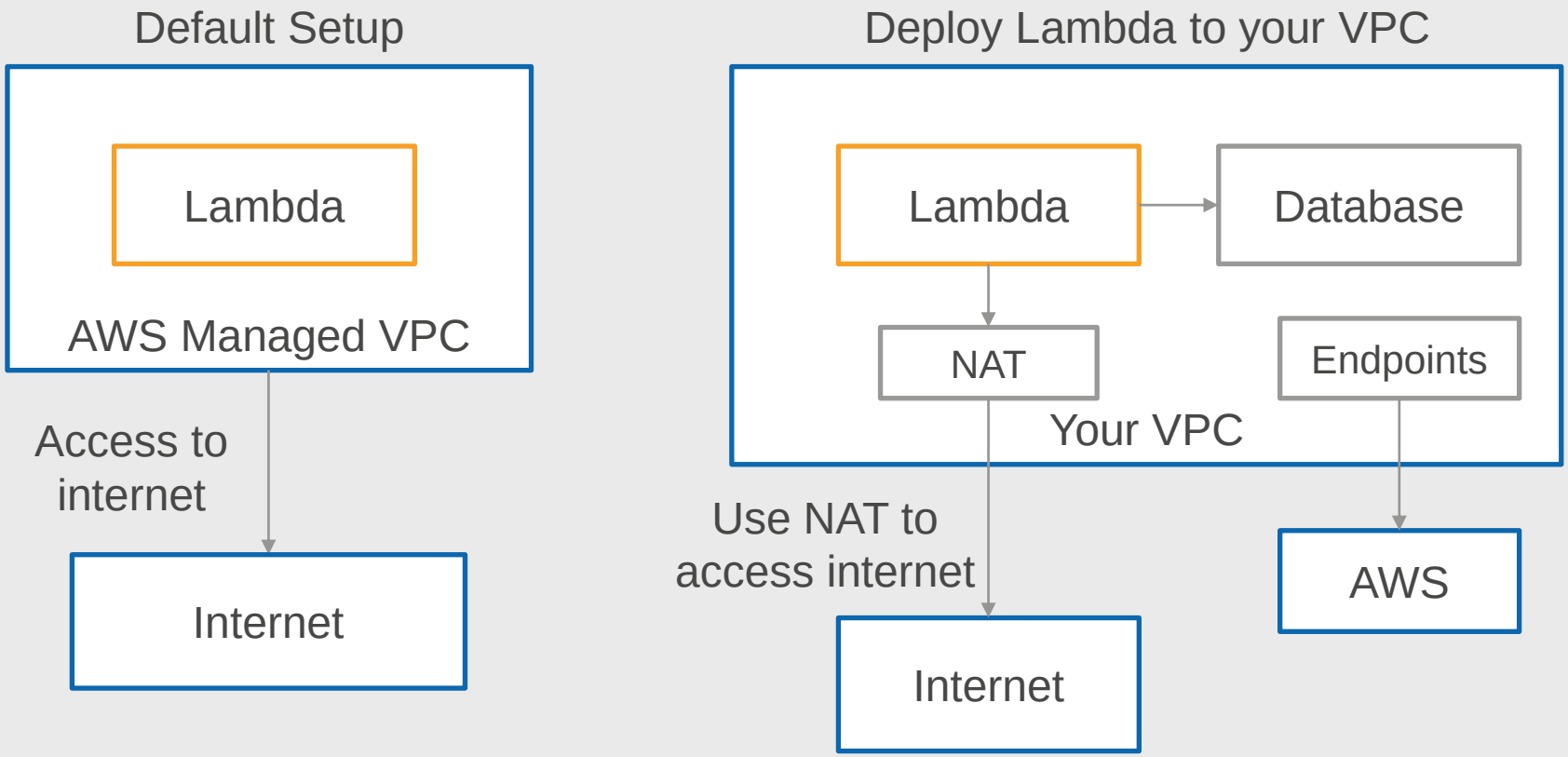
1. In a Server-based solution, connection is kept open and reused across different requests
2. Pool and reuse connections
3. Connection Pooling is tricky in Lambda functions

Lambda - Use RDS Proxy



1. RDS Proxy let's you share database connections across multiple Lambda instances
2. Improves performance!
3. Reduces database failover time
4. RDS Proxy maintain database credentials (using Secrets Manager or Paramter Store)
5. Lambda function can get a database connection from RDS Proxy using IAM Role

Network Isolation



Lambda Security

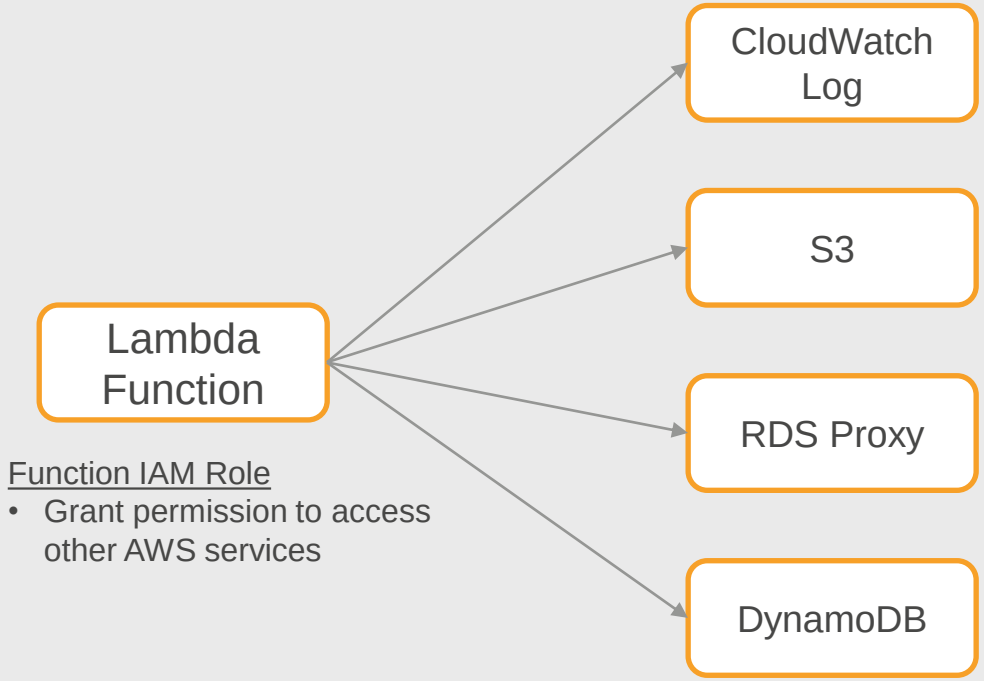
Who can call the lambda function
What resources can the function access

Lambda who can invoke

Identity-based Policy
- Grant permission to invoke lambda function
- IAM Role, Users or Groups

Resource-based Policy
- Grant permission to AWS service or identity to invoke function

Lambda resource access



Function IAM Role

- Grant permission to access other AWS services

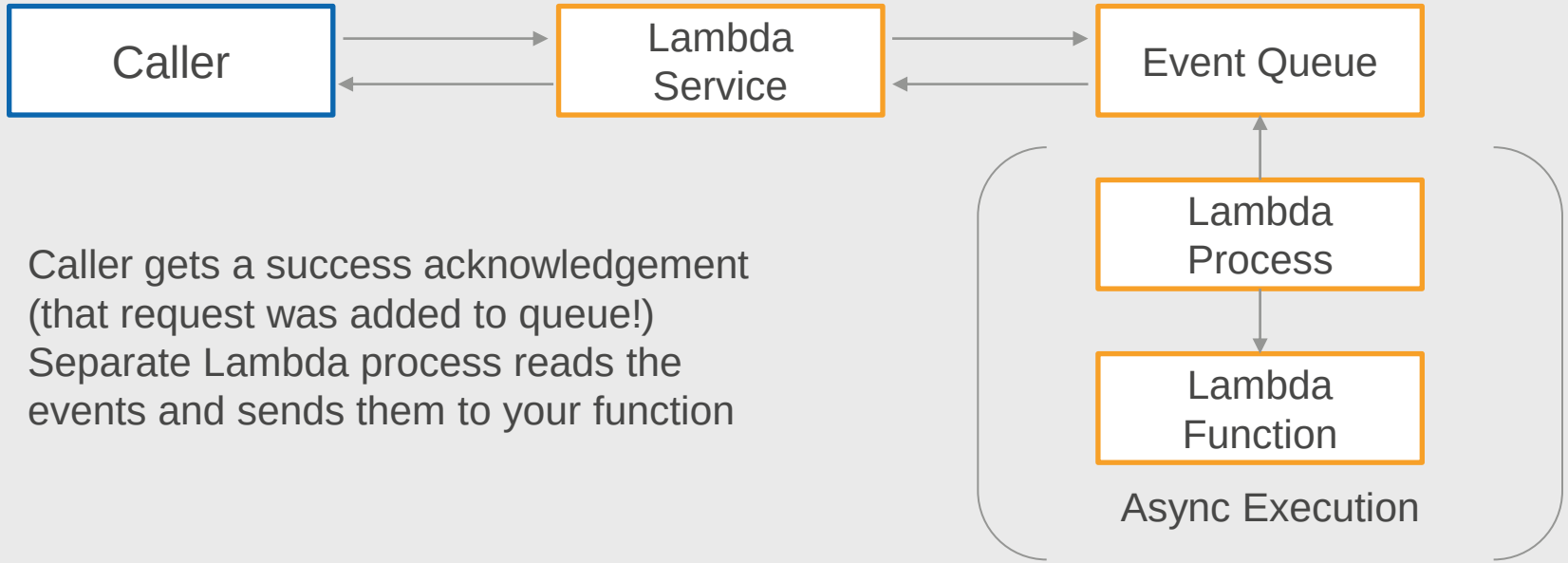
Lambda Invocation

Synchronous (Known as **RequestResponse** in Lambda API)
Asynchronous (Known as **Event** in Lambda API)

Synchronous Invocation (RequestResponse)

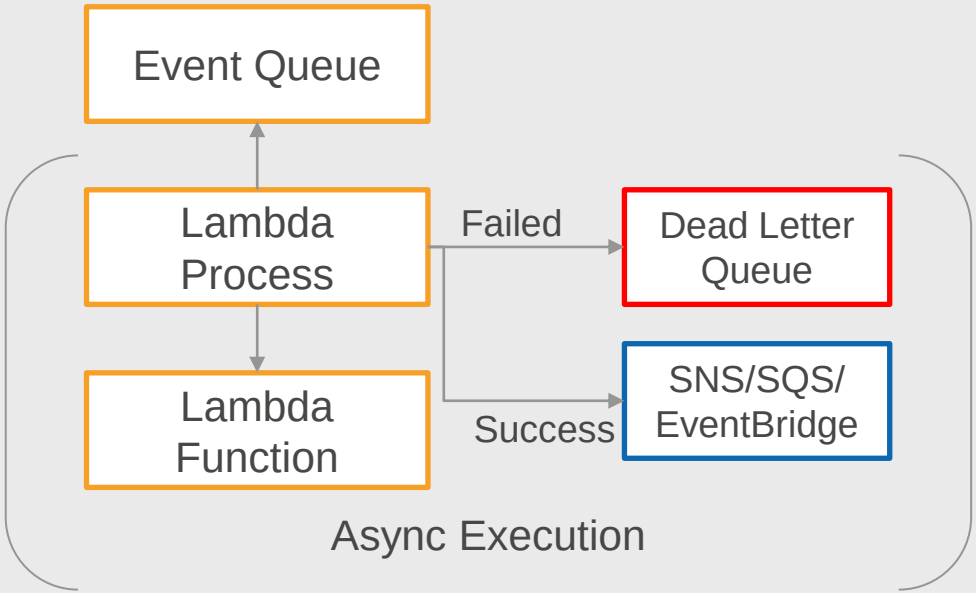
1. Caller waits for function execution to complete
2. Any errors are returned to the caller – invalid permission, invalid data, throttling error and more
3. Retries are handled by the caller
4. Throttling errors – caller must retry using exponential backoff strategy (increase wait time between successive retries)

Asynchronous Invocation (Event)



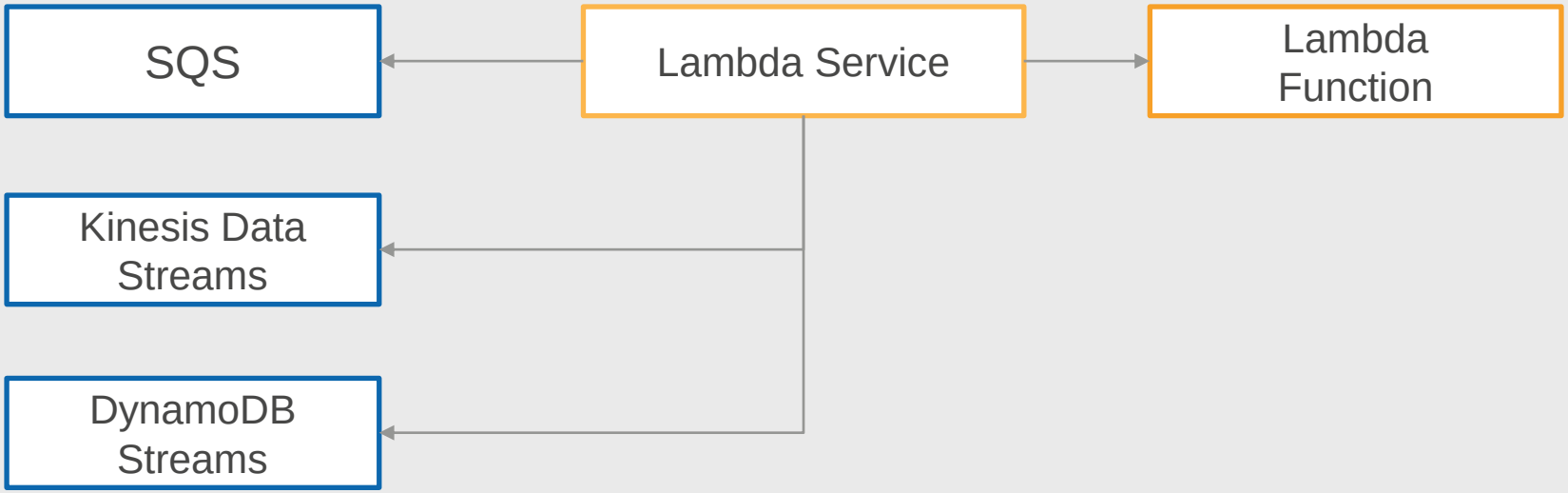
1. Caller gets a success acknowledgement (that request was added to queue!)
2. Separate Lambda process reads the events and sends them to your function

Error Handling – Asynchronous execution



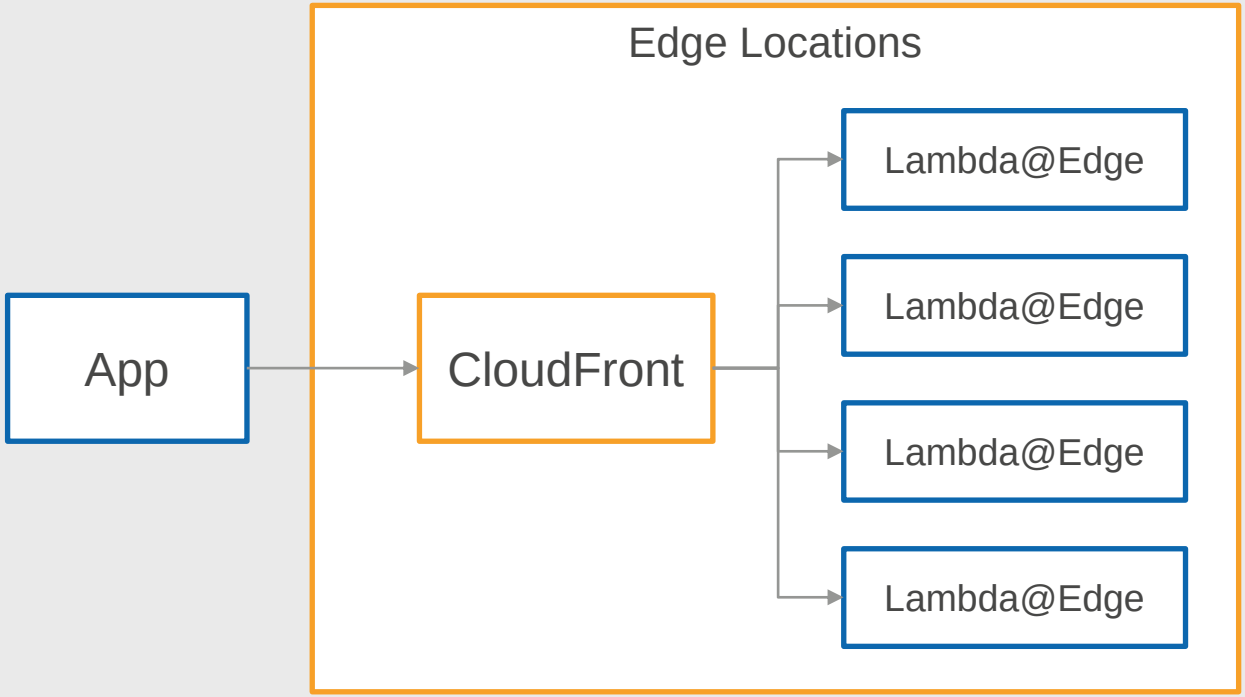
1. Function error – Lambda retries two more times
2. Throttling errors – Lambda will retry for up to six hours with exponential backoff
3. Original caller is not aware of errors!
4. Optional – Store failed events in a Dead-Letter Queue
5. Optional – Store send success events to SQS, SNS, EventBridge, …

Lambda Polling



Lambda service supports polling for messages and records in SQS, Kinesis Data Streams, DynamoDB Streams and invokes the configured Lambda Function!

Lambda@Edge



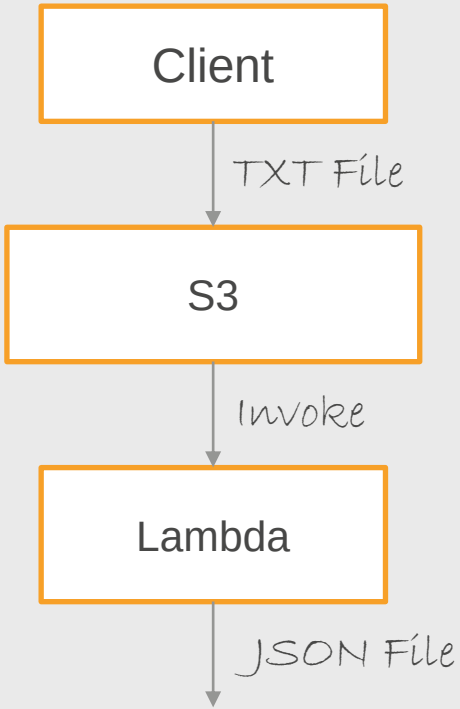
Lambda@Edge Use Cases

Customize cached content without routing request to origin

Improve security by checking for expired or invalid JWT token and redirect to login page

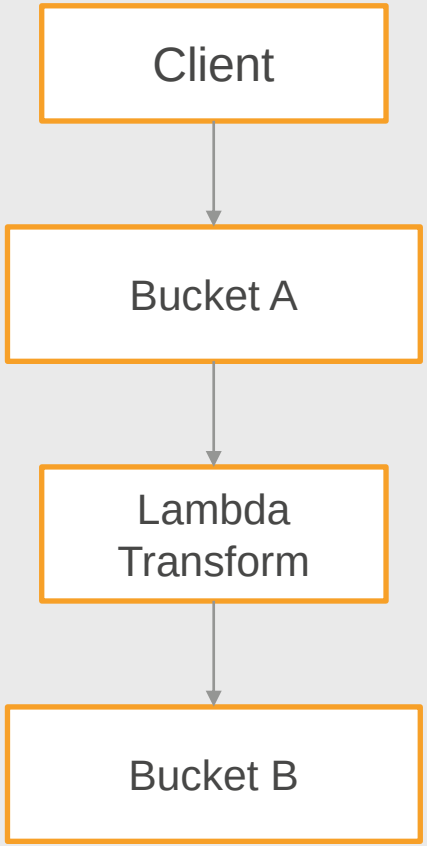
Add custom/dynamic security headers when calling the origin server

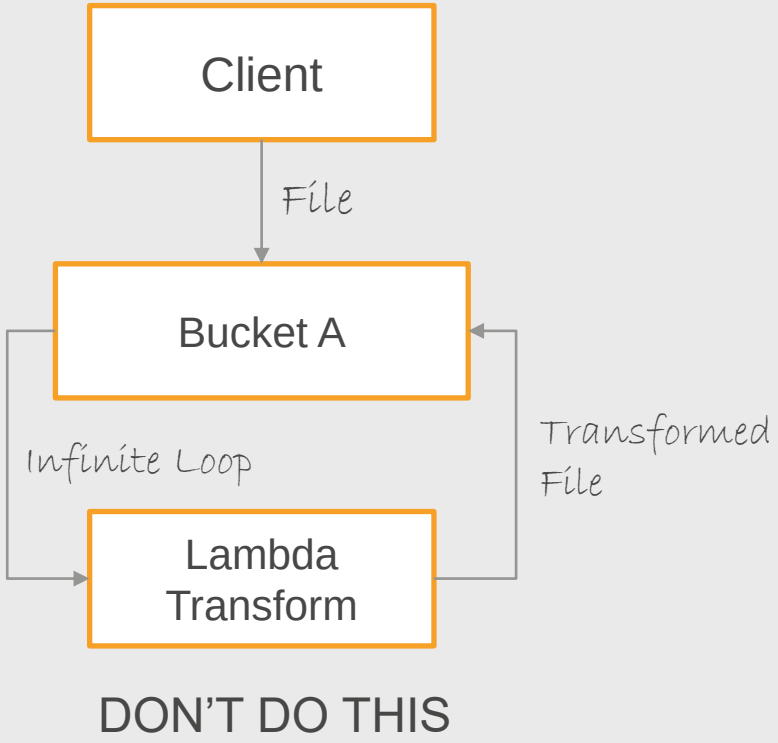
Lab – Transform S3 Object



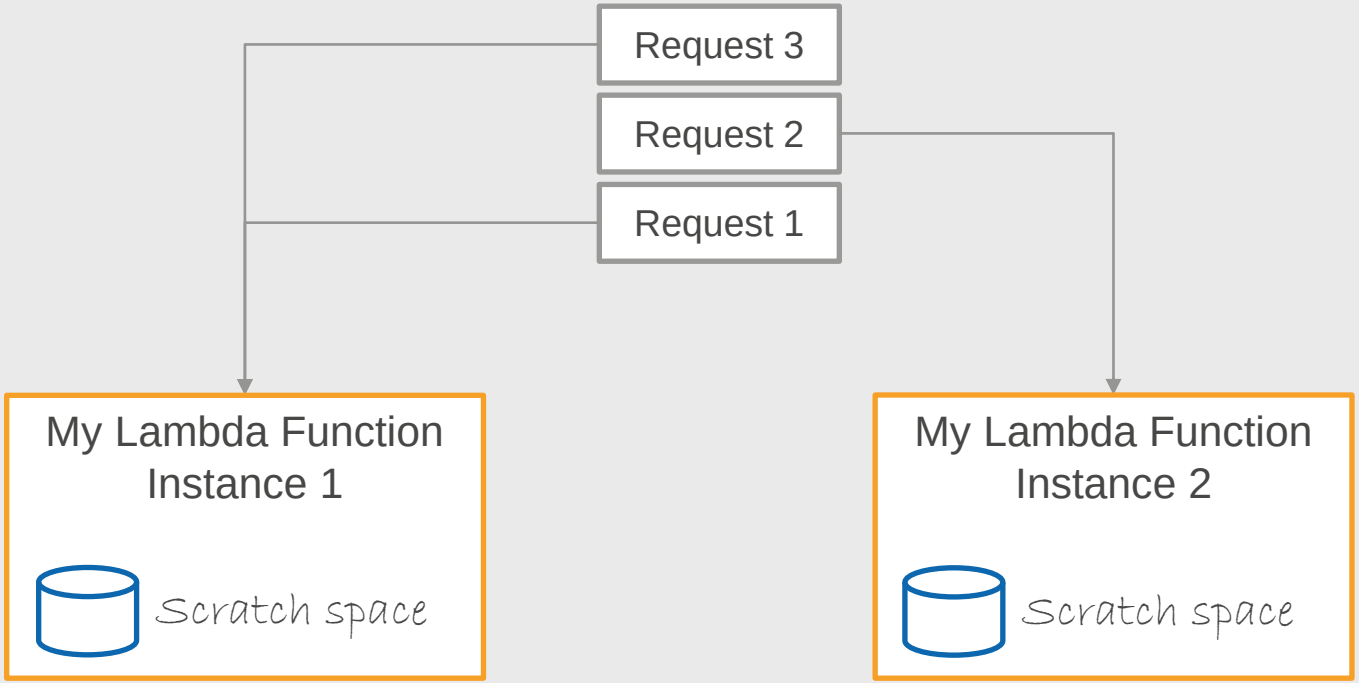
Python Regular Expressions
https://github.com/ChandraLingam

Use Different Buckets for Source, Destination





Lambda Concurrency



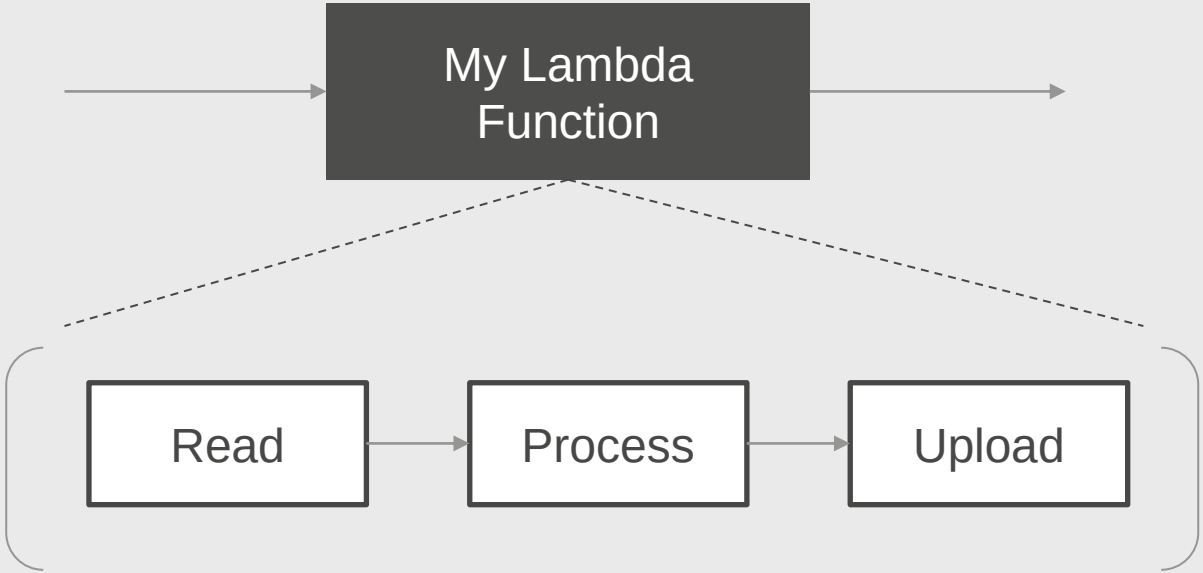
Lambda function instance processes one request at a time
Each instance can process many requests one-by-one

Lab – Xray Integration

Gain insight into distributed performance issues and errors

Enable Xray tracing for lambda function

Lab – Xray and Lambda Layers



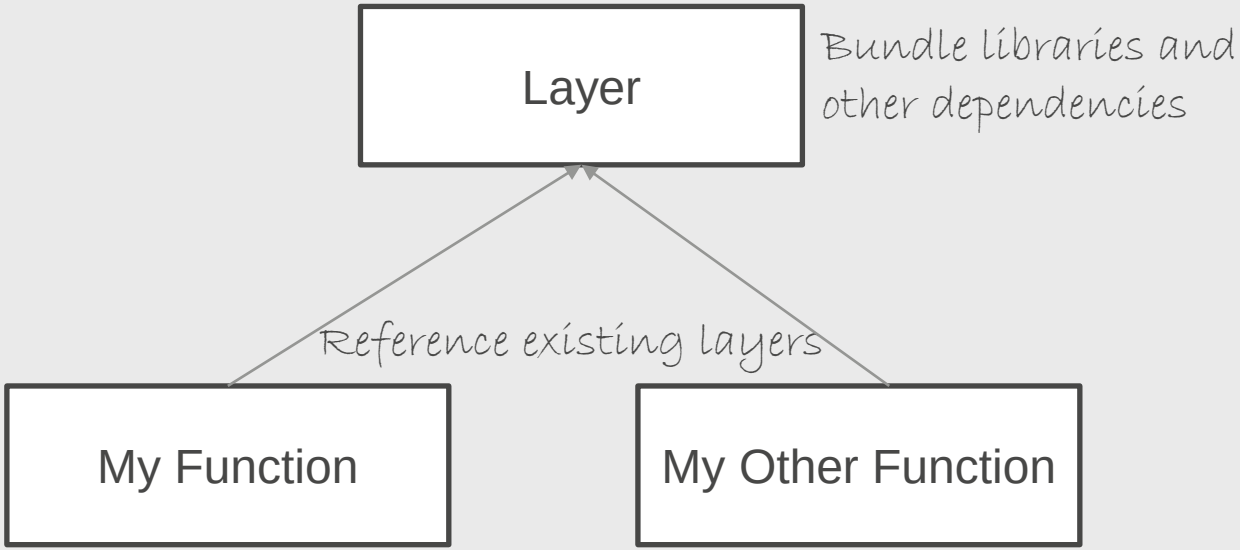
1. Import Xray library in the function
2. Instrument code

Issues with Xray Integration

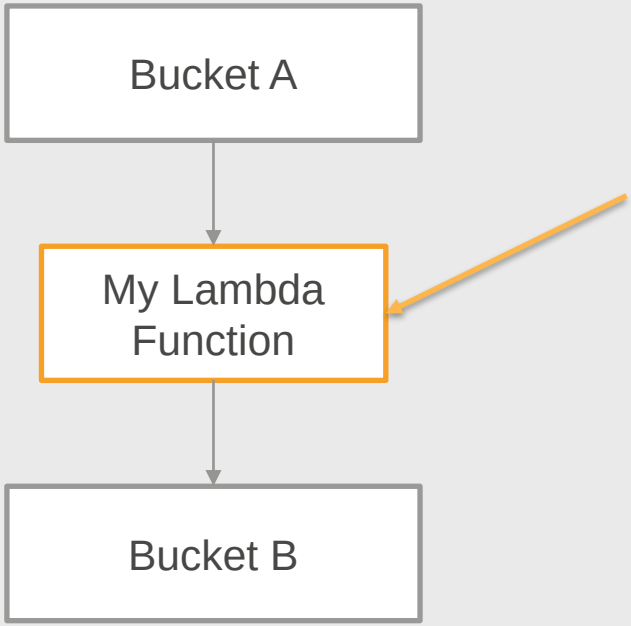
Challenge: Xray library is not part of Lambda Python environment (as of March-2021)

1. Package Xray library as a zip file and use with Lambda
2. Increases size of our function deployment
3. Solution: Use Lambda Layers!

Lambda Layers for code reuse



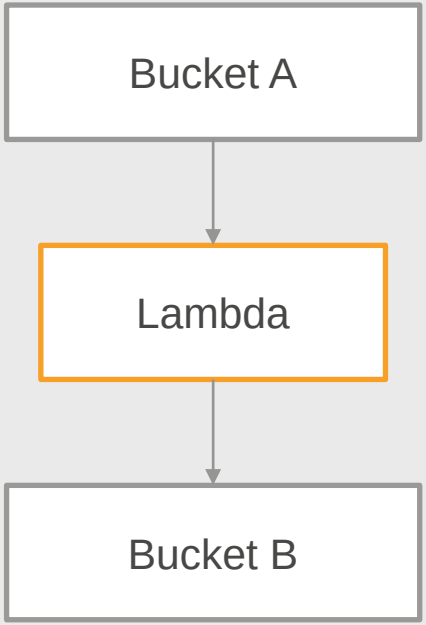
Challenges

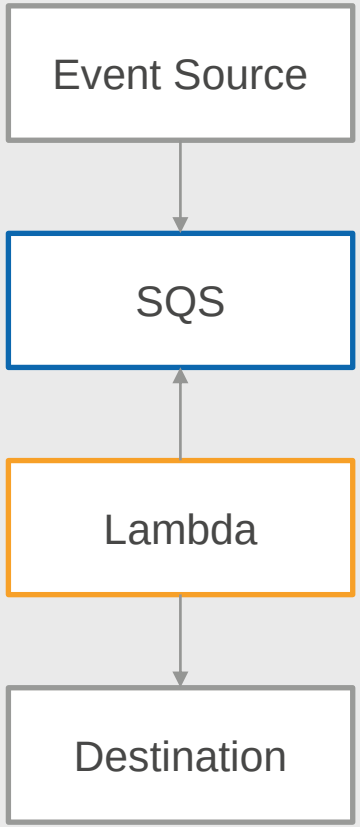


How to incorporate bug fixes and other code changes to the lambda function?

How to rollback changes in Lambda?

Decouple Processing





Use SQS to decouple processing

Easy to manage lambda changes

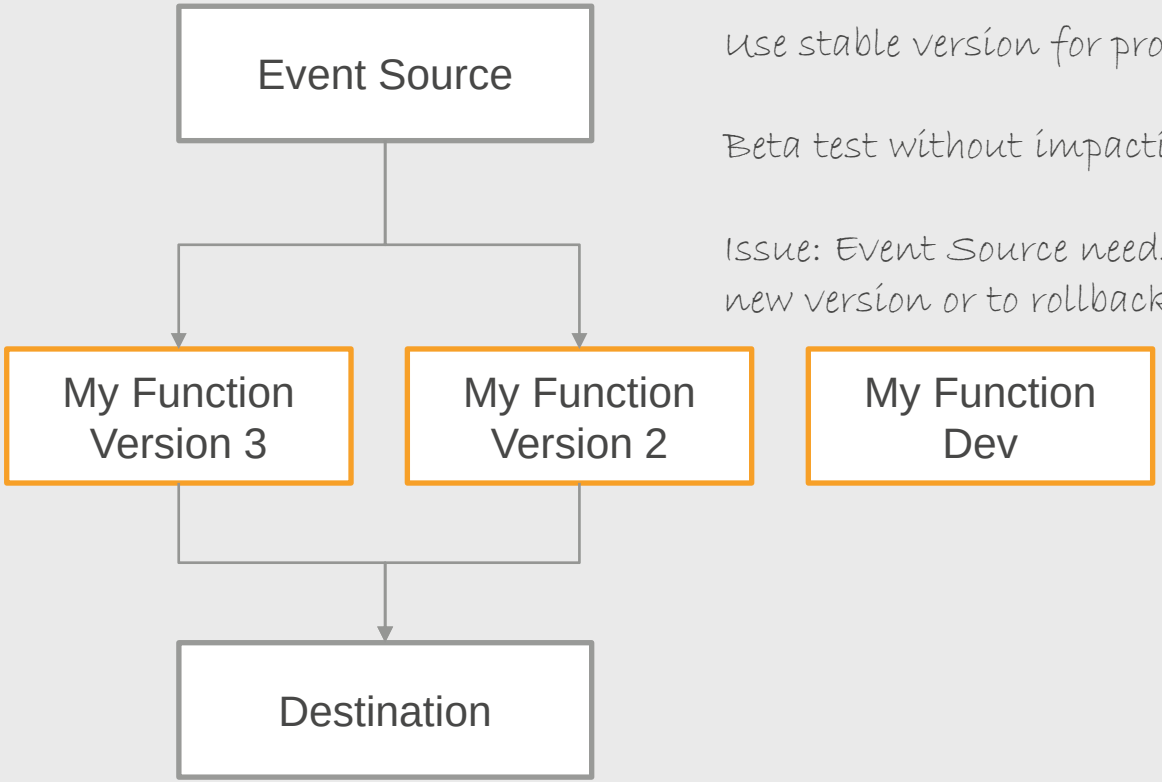
Lambda Versions

With versions, manage the deployment of your functions

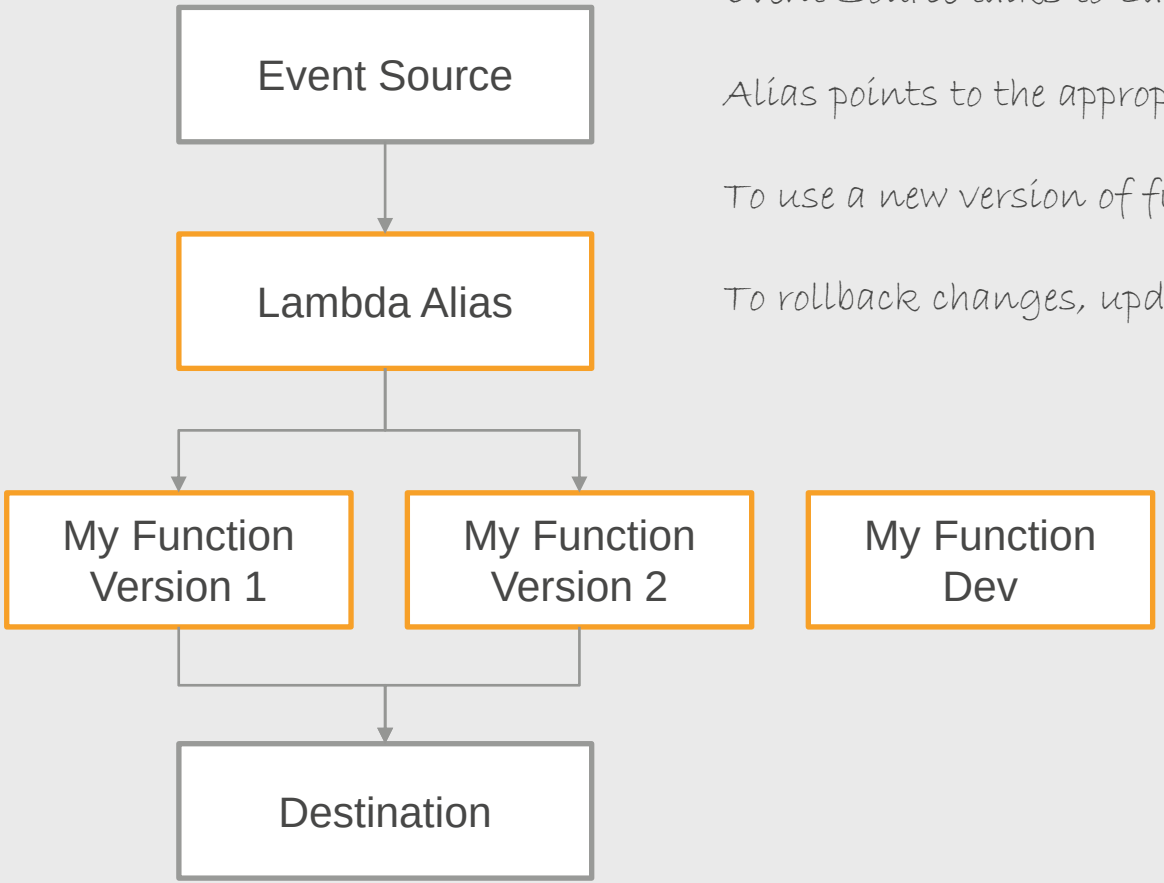
Use stable version for production

Beta test without impacting existing clients

Issue: Event Source needs to be updated to use new version or to rollback a version



Lambda Alias



Event Source talks to Lambda Alias

Alias points to the appropriate version

To use a new version of function, update alias

To rollback changes, update alias

Chandra Lingam

70,000+ Students

For AWS self-paced video courses, visit:

https://www.cloudwavetraining.com/

Join our discord channel

https://discord.gg/Uz88cE3wWp

Cloud Wave LLC