AWS Lambda & API Gateway: Serverless Architecture Guide

Interview Preparation - Lambda Functions, API Gateway, Integration & Best Practices

1. AWS Lambda - Serverless Compute Service

AWS Lambda is a serverless compute service that runs your code in response to events. You don't manage servers - AWS handles infrastructure, scaling, and availability automatically.

Key Lambda Concepts:

Concept	Description	Example
Function	Self-contained code that runs on events	process_order(event) returns response
Event Source	Service that triggers Lambda execution	API Gateway, S3, DynamoDB, SQS
Handler	Entry point function in your code	lambda_handler(event, context)
Runtime	Language environment for execution	Python 3.11, Node.js 18.x, Java 17
Memory	RAM allocated (128 MB - 10 GB)	512 MB = 0.33 vCPU 1024 MB = 0.58 vCPU
Timeout	Max execution time (1 sec - 15 min)	Default: 3 seconds Max: 900 seconds
Concurrency	Number of simultaneous executions	Default: 1000 concurrent executions per region

Lambda Benefits:

- ✓ No server management: AWS handles infrastructure
- ✓ Automatic scaling: From 0 to thousands of concurrent executions
- ✓ Pay per use: Only charged for compute time (100ms increments)
- ✓ High availability: Built-in fault tolerance across AZs
- ✓ Event-driven: Responds to triggers from 200+ AWS services

2. Lambda Supported Runtimes

Language	Runtime Versions	Use Case

Python	Python 3.11, 3.10, 3.9, 3.8	Data processing, ML, API backends, automation
Node.js	Node.js 18.x, 16.x, 14.x	Real-time apps, APIs, webhooks
Java	Java 17, 11, 8 (Corretto)	Enterprise apps, Android backends
Go	Go 1.x	High-performance APIs, system tools
.NET	.NET 6, .NET Core 3.1	Windows apps, enterprise systems
Ruby	Ruby 2.7	Web apps, scripting
Custom Runtime	Provided.al2 (Amazon Linux 2)	Rust, PHP, any language with runtime API
Container Image	Up to 10 GB image	Complex dependencies, custom environments

3. Lambda Function Creation - Step-by-Step

#	Action	Details / Options
1	Navigate to Lambda Console	AWS Console → Services → Lambda Click "Create function"
2	Choose Creation Method	Author from scratch: Write new code Use blueprint: Pre-built templates Container image: Use Docker image Browse serverless app repo
3	Basic Information	Function name: my-function Runtime: Python 3.11 Architecture: x86_64 or arm64
4	Permissions	Execution role: - Create new role (basic permissions) - Use existing role - Create from AWS policy templates
5	Advanced Settings (Optional)	Enable VPC: Select VPC, subnets Enable function URL: Public HTTPS endpoint Enable tags: For organization
6	Write Code	Inline editor: Write code in browser Upload .zip: Package dependencies Upload from S3: Large packages Container image: From ECR

7	Configure Function	Memory: 128 MB - 10,240 MB Timeout: 3 sec (default) - 900 sec Environment variables: Key-value pairs Ephemeral storage: 512 MB - 10 GB
8	Add Trigger (Optional)	API Gateway: HTTP/REST API S3: Object upload events DynamoDB: Stream events EventBridge: Scheduled/custom events
9	Add Destination (Optional)	On success: Send to SQS, SNS, Lambda On failure: Dead letter queue (DLQ)
10	Test Function	Create test event with JSON payload Click "Test" button View execution results and logs

Sample Lambda Function (Python):

```
import json

def lambda_handler(event, context):
    # Extract data from event
    name = event.get('name', 'World')

    # Business logic
    message = f'Hello {name} from Lambda!'

    # Return response
    return {
        'statusCode': 200,
        'body': json.dumps({'message': message})
    }
```

4. Amazon API Gateway - Building APIs

Amazon API Gateway is a fully managed service to create, publish, maintain, monitor, and secure APIs at any scale. It acts as a 'front door' for applications to access backend services.

API Gateway Types:

API Type	Protocol	Use Case	Features
REST API	HTTP/HTTPS	Full-featured APIs with caching, authorization	API keys, Usage plans, Request/response transformation, Caching
HTTP API	HTTP/HTTPS	Low-latency, cost-effective APIs	JWT authorization, CORS, Auto deploy, Cheaper than REST
WebSocket API	WebSocket	Real-time two-way communication	Chat apps, Live dashboards, Streaming data

API Gateway Benefits:

✓ Serverless: No infrastructure to manage

✓ Auto-scaling: Handles any volume of API calls

✓ Security: Built-in DDoS protection, AWS WAF integration

✓ Monitoring: CloudWatch metrics and logging

✓ Version management: Multiple API stages (dev, staging, prod)

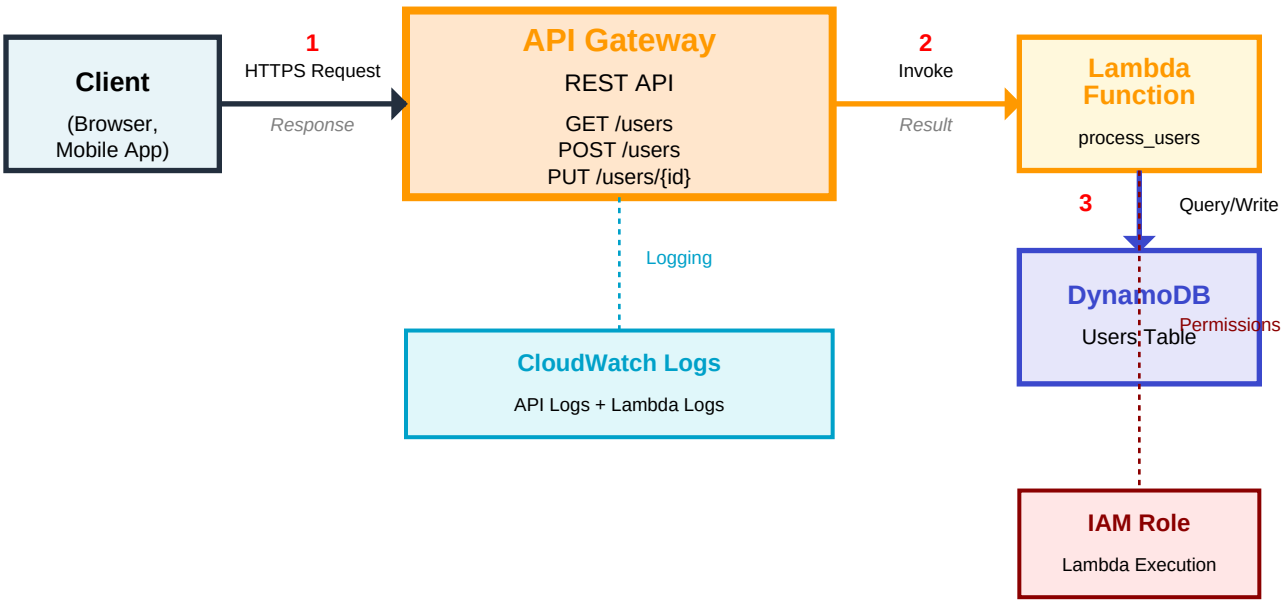
5. API Gateway Creation - Step-by-Step (REST API)

#	Action	Details / Options
1	Navigate to API Gateway Console	AWS Console → Services → API Gateway Click "Create API"
2	Choose API Type	REST API: Full features HTTP API: Lower cost, simpler WebSocket API: Real-time REST API (Private): VPC only
3	API Settings	Protocol: REST Create new API: Choose this API name: my-rest-api Description: Optional Endpoint Type: Regional, Edge, Private

4	Create Resources	Resource path: /users Resource name: users Enable CORS: Yes/No Create child resources: /users/{id}
5	Create Methods	Select resource: /users Create method: GET, POST, PUT, DELETE Integration type: Lambda, HTTP, Mock, AWS Service
6	Configure Integration	Integration type: Lambda Function Lambda Region: us-east-1 Lambda Function: my-function Use Lambda Proxy: Enable (passes entire request to Lambda)
7	Method Request Settings	Authorization: None, IAM, Cognito, Lambda authorizer API Key Required: Yes/No Request Validator: Validate body, parameters
8	Method Response	Status codes: 200, 400, 500 Response headers: Access-Control-Allow-Origin Response models: Define schema
9	Enable CORS (Optional)	Select resource → Actions → Enable CORS Allow origins: * or specific domains Allow methods: GET, POST, OPTIONS
10	Deploy API	Actions → Deploy API Deployment stage: dev, staging, prod Stage description: Optional Get invoke URL: https://abc123.execute-api.region.amazonaws.com/dev

6. Lambda + API Gateway Integration Architecture

The most common serverless pattern: API Gateway receives HTTP requests and triggers Lambda functions to process them. This is the foundation of serverless REST APIs.



Request Flow:

1. Client sends HTTPS request to API Gateway endpoint
2. API Gateway validates request and invokes Lambda function synchronously
3. Lambda processes request, queries DynamoDB, returns response
4. API Gateway formats response and returns to client (200 OK)

Typical Latency: API Gateway (50ms) + Lambda Cold Start (200-500ms) + Execution (varies)

Benefits: No servers, Auto-scaling, Pay per request, Built-in HA across AZs
Limitations: 29 sec API Gateway timeout, 15 min Lambda timeout, No persistent connections

7. API Gateway Integration Types

API Gateway supports multiple integration types to connect with backend services:

Integration Type	Description	Use Case	Configuration
Lambda Proxy	Passes entire request to Lambda, Lambda returns formatted response	Most common, simplest setup, Lambda controls everything	Enable "Lambda Proxy integration" checkbox
Lambda Custom	API Gateway transforms request/response	Complex transformations, legacy Lambda functions	Define mapping templates in VTL (Velocity Template Language)
HTTP Proxy	Forwards request to HTTP endpoint as-is	Proxy to existing REST API, microservices	Specify HTTP endpoint URL
HTTP Custom	Transform before forwarding to HTTP endpoint	Modify headers, request structure	Mapping templates + HTTP endpoint
AWS Service	Direct integration with AWS services	Expose DynamoDB, S3, SQS without Lambda	IAM role + service action (e.g., PutItem)
Mock	Returns response without backend	Testing, API prototyping	Define static response in API Gateway

8. Lambda Cold Start vs Warm Start

Understanding cold starts is critical for performance optimization and interviews:

Aspect	Cold Start	Warm Start
Definition	Lambda initializes new execution environment from scratch	Lambda reuses existing execution environment
When It Occurs	First invocation After idle period (15 min) Concurrency increase Code/config update	Subsequent invocations within 15 minutes
Latency	200ms - 2 seconds (depends on runtime, package size, VPC)	1-50ms (very fast)
Steps Involved	1. Download code 2. Start runtime 3. Initialize code 4. Execute handler	1. Execute handler only

Memory Impact	More memory = faster cold start (better CPU)	Memory doesn't affect warm start
VPC Impact	Adds 5-10 seconds (ENI creation)	No VPC overhead
Optimization	Reduce package size Use Provisioned Concurrency Keep functions warm	Already optimized

Cold Start Optimization Strategies:

1. Minimize package size: Remove unused dependencies
2. Use Provisioned Concurrency: Pre-warmed execution environments ($$$)
3. Increase memory: More memory = more CPU = faster init
4. Avoid VPC: Only use VPC when necessary (adds 5-10 sec)
5. Keep functions warm: Schedule EventBridge rule to ping every 5 min
6. Use arm64 architecture: 20% faster and 20% cheaper

9. Lambda + API Gateway Pricing

Lambda Pricing:

Component	Pricing	Free Tier	Example
Requests	$0.20 per 1M requests	1M requests/ month free	5M requests = $0.80/month
Duration (Compute)	$0.0000166667 per GB-second	400,000 GB-sec/ month free	512 MB, 1 sec, 1M requests = $8.33/month
Provisioned Concurrency	$0.0000041667 per GB-hour	None	5 instances @ 512 MB = $15/month

API Gateway Pricing:

API Type	Pricing per Million Requests	Caching	Free Tier
REST API	$3.50 (first 333M requests) $2.80 (next 667M requests) $2.38 (over 1B requests)	$0.02/hour per GB cache	1M API calls/ month for 12 months
HTTP API	$1.00 (first 300M requests) $0.90 (over 300M requests)	Not supported	1M API calls/ month for 12 months
WebSocket	$1.00 per million messages $0.25 per million connection minutes	Not applicable	1M messages/ month for 12 months

Sample Monthly Cost Calculation:

Component	Usage	Calculation	Cost
API Gateway (REST)	5M requests	5M × $3.50/1M	$17.50
Lambda Requests	5M invocations	(5M - 1M free) × $0.20/1M	$0.80
Lambda Duration	512 MB, 200ms avg, 5M requests	(512MB/1024) × 0.2s × 5M GB-seconds = 500K GB-sec (500K - 400K free) × $0.0000166667	$1.67
Data Transfer	10 GB OUT	10 GB × $0.09/GB	$0.90
			$20.87

10. Lambda + API Gateway Best Practices

#	Best Practice	Why It Matters	Implementation
1	Use Lambda Proxy Integration	Simplifies development, Lambda controls response	Enable "Use Lambda Proxy" in API Gateway
2	Enable CORS Correctly	Allows browser javascript to call API	Enable CORS in API Gateway + return headers in Lambda
3	Implement Error Handling	Return proper HTTP status codes (400, 500)	try/except in Lambda, return {statusCode: 500}
4	Use Environment Variables	Avoid hardcoding configs, secrets	Set in Lambda configuration, access via os.environ[]
5	Enable CloudWatch Logs	Debug issues, monitor performance	Auto-enabled, use print() or logger in code
6	Set Appropriate Timeouts	Prevent hanging functions, save costs	API Gateway: 29 sec max Lambda: Match workload (3-900s)
7	Use IAM/ Cognito for Auth	Secure APIs, control access	Add authorizer in API Gateway method
8	Optimize Cold Starts	Reduce latency for users	Minimize package size, increase memory, avoid VPC
9	Use API Stages	Separate dev, staging, prod environments	Deploy API to multiple stages with stage variables
10	Monitor with X-Ray	Trace requests through services	Enable X-Ray tracing in Lambda + API Gateway

11. Common Interview Questions & Answers

Question	Answer
What is serverless?	No server management, auto-scaling, pay-per-use. AWS manages infrastructure. You only write code.
Lambda vs EC2?	Lambda: Serverless, event-driven, max 15 min, auto-scale. EC2: Server-based, always running, full control, manual scaling.
What is cold start?	Initial delay when Lambda creates new execution environment. 200ms-2s. Optimize by reducing package size, increasing memory, avoiding VPC.
API Gateway vs ALB?	API Gateway: REST APIs, WebSocket, throttling, API keys. ALB: Load balancing, path-based routing, target groups. Use API GW for APIs, ALB for apps.
How to secure Lambda?	1. IAM roles for execution 2. Resource-based policies for invocation 3. VPC for private resources 4. Environment variables encryption (KMS)
Lambda limitations?	Max 15 min timeout, 10 GB memory, 6 MB sync payload, 250 MB async payload, 50 MB layer size, /tmp storage 512 MB - 10 GB
How does Lambda scale?	Automatically scales by creating concurrent execution environments. Default 1000 concurrent executions per region. Can request increase.
REST API vs HTTP API?	REST API: Full features, caching, usage plans, API keys. HTTP API: 70% cheaper, simpler, JWT auth, lower latency. Use HTTP for simple APIs.

12. Interview Key Points to Remember

When explaining Lambda:

- Serverless compute service - no server management
- Event-driven execution from 200+ AWS services
- Auto-scales from 0 to thousands of concurrent executions
- Pay only for compute time (100ms increments)

When explaining API Gateway:

- Fully managed API service - create, publish, secure APIs
- Three types: REST (full features), HTTP (cheaper), WebSocket (real-time)
- Built-in throttling, caching, authorization
- Common pattern: API Gateway → Lambda → DynamoDB/RDS

When discussing Integration:

- Lambda Proxy: Most common, simplest setup
- API Gateway invokes Lambda synchronously
- 29 second timeout at API Gateway level
- Enable CORS for browser-based apps

When discussing Cold Starts:

- Cold start = First invocation or after 15 min idle
- Adds 200ms - 2 seconds latency
- Optimize: Reduce package size, increase memory, avoid VPC
- Provisioned Concurrency: Pre-warmed environments (costs extra)