AWS Certified Developer Associate DVA-C02

Plan de Estudio Personalizado

Preparado para:	Sumer Zambrano - Full Stack Developer
Experiencia:	3+ años en Node.js, TypeScript, AWS
Fecha:	23 de diciembre de 2025
Duración estimada:	3-4 meses (tiempo parcial)
Objetivo:	Certificación AWS DVA-C02 con 720+ puntos

1. Resumen Ejecutivo y Análisis Personalizado

🎯 TU SITUACIÓN ACTUAL

TUS FORTALEZAS (aprovéchalas):

- **Lambda & Serverless:** Experiencia directa con Lambda, SQS, SNS - esto es el 40% del examen
- **Arquitectura:** Hexagonal architecture y microservicios - ventaja enorme para entender patterns
- **DynamoDB:** Experiencia práctica - solo necesitas profundizar en GSI/LSI y optimización
- **Node.js/TypeScript:** El lenguaje del examen - puedes hacer todos los labs en tu stack
- **Proyectos reales:** Has migrado monolitos, manejado pagos, implementado pub/sub

GAPS A CUBRIR (priorízalos):

- **CI/CD:** CodePipeline, CodeBuild, CodeDeploy - nunca has usado estas herramientas AWS
- **Containers:** ECS/Fargate - experiencia limitada
- **Cognito:** Nunca implementado autenticación con Cognito User Pools
- **X-Ray:** Distributed tracing - solo conocimiento teórico
- **CloudFormation/CDK:** IaC con AWS - necesitas práctica intensiva

💡 RECOMENDACIÓN ESTRATÉGICA

Tu perfil es IDEAL para este examen. Con tu experiencia en backend AWS, puedes aprobar en 3-4 meses estudiando 2-3 horas diarias.

2. Información del Examen DVA-C02

Aspecto	Detalles
Nombre oficial	AWS Certified Developer - Associate
Código	DVA-C02
Preguntas	65 (50 scored + 15 unscored)
Duración	130 minutos (2 horas 10 min)
Formato	Multiple choice y multiple response
Puntaje mínimo	720/1000 (72%)
Validez	3 años
Costo	$150 USD

📊 DISTRIBUCIÓN DE DOMINIOS

Dominio	Peso	Tu Nivel	Prioridad
1. Development with AWS Services	32%	🟢 Alto	Media
2. Security	26%	🟡 Medio	Alta
3. Deployment	24%	🔴 Bajo	MUY ALTA
4. Troubleshooting & Optimization	18%	🟡 Medio	Alta

3. Plan de Estudio Optimizado (3-4 meses)

🗓️ TIMELINE REALISTA

Total: 14 semanas (3.5 meses) Dedicación: 2-3 horas/día durante semana (después del trabajo)Dedicación: 4-5 horas/día fines de semana**Total estimado: 250-300 horas**

📍 FASE 1: Quick Wins & Fundamentos (Semanas 1-3)

- **Objetivo:** Ganar confianza dominando servicios que ya conoces
- Lambda deep dive: Layers, versions, aliases, concurrency, cold starts
- DynamoDB avanzado: GSI/LSI, streams, DAX, capacity modes
- API Gateway: REST vs HTTP APIs, stages, throttling, caching
- SQS/SNS: FIFO queues, DLQ, fanout pattern
- **Resultado esperado:** 70% de Dominio 1 dominado

📍 FASE 2: Gaps Críticos - CI/CD (Semanas 4-6)

- **Objetivo:** Dominar Deployment (24% del examen) - TU MAYOR DEBILIDAD
- CodePipeline: Stages, actions, artifacts, approval gates
- CodeBuild: buildspec.yml, artifacts, environment variables
- CodeDeploy: Deployment strategies (rolling, blue/green, canary)
- CloudFormation: Templates, stacks, change sets, nested stacks
- CDK: Constructs, stacks, assets, synthesis
- **Resultado esperado:** 90% de Dominio 3 dominado

📍 FASE 3: Security & Cognito (Semanas 7-8)

- Cognito User Pools: Sign-up, sign-in, MFA, JWT tokens
- Lambda Authorizers: Token-based, request-based
- Secrets Manager: Rotation, retrieval, Lambda integration
- KMS: Customer managed keys, encryption contexts

📍 FASE 4: Observability & Containers (Semanas 9-10)

- CloudWatch: Logs, Metrics, Alarms, Dashboards
- X-Ray: Tracing, service maps, segments, subsegments
- ECS/Fargate: Task definitions, services, clusters

📍 FASE 5: Integration & Advanced Patterns (Semanas 11-12)

- Microservicios avanzados: Step Functions, Saga pattern, CQRS
- Event-driven architecture: EventBridge, pub/sub patterns

📍 FASE 6: Practice Exams & Final Review (Semanas 13-14)

- Practice Exam 1-3 (Tutorials Dojo + Whizlabs + AWS Official)
- Repaso de whitepapers clave
- **DÍA DEL EXAMEN** 🎯

4. Microservicios: Enfoque Práctico

🚀 PROCESO DE SETUP CORRECTO

Para cada microservicio, seguir este orden EXACTO:

Paso 1: Crear proyecto con CDK (PRIMERO)

```
mkdir microservice-name
cd microservice-name
cdk init app --language=typescript
```

– Borrar archivos npm:

```
rm -rf node_modules package-lock.json
```

– Y agregar pnpm

```
pnpm install
```

Instalar y configurar ESLint::

```
pnpm add -D eslint
```

```
npx eslint --init
```

Paso 2: Instalar dependencias adicionales

```
pnpm add zod uuid @aws-sdk/client-dynamodb @aws-sdk/lib-dynamodb
pnpm add -D jest-cucumber openapi3-ts ts-api-utils
```

Paso 3: Crear estructura Clean Architecture

```
mkdir -p src/{domain/{entities,repositories,errors},application/{use-cases,dtos}}
mkdir -p
src/infrastructure/{lambda/handlers,repositories,adapters,http,config,swagger}
mkdir -p tests/{fixtures/builders,unit,integration}
```

Paso 4: Crear .env.example

```
TABLE_NAME=microservice-dev-Table
AWS_REGION=us-east-1
NODE_ENV=development
```

Paso 5: Actualizar tsconfig.json

Agregar path aliases para @domain, @application, @infrastructure

Paso 6: Implementar capas

Domain → Application → Infrastructure → CDK Stack

📦 PACKAGE.JSON - SCRIPTS ESENCIALES

Scripts necesarios en cada microservicio:

EsLint

"lint": "eslint .",

"lint:fix": "eslint . --fix"

Build & Test

"build": "tsc"
"test": "jest"
"test:coverage": "jest --coverage"

CDK Commands

"cdk:synth": "cdk synth"
"cdk:deploy:dev": "cdk deploy --context environment=dev"
"cdk:deploy:prod": "cdk deploy --context environment=prod --require-approval always"

SAM Local (Testing)

"sam:local:api": "sam local start-api -t cdk.out/Stack.template.json --port 3000"

🛠️ ESTRUCTURA DE CARPETAS COMPLETA

Estructura base reutilizable para todos los microservicios:

```
microservice-name/
├── bin/
│   └── app.ts                    # CDK entry point
├── infrastructure/
│   └── lib/
│       └── stack.ts              # CDK Stack
├── src/
│   ├── domain/                   # Business logic
│   │   ├── entities/
│   │   ├── repositories/
│   │   └── errors/
│   ├── application/              # Use cases
│   │   ├── use-cases/
│   │   └── dtos/                 # Zod schemas
│   └── infrastructure/
│       ├── lambda/handlers/      # Lambda functions
│       ├── repositories/         # DynamoDB impl
│       ├── swagger/              # OpenAPI
│       └── config/
├── tests/
│   ├── fixtures/                 # Test data
│   ├── unit/
│   └── integration/              # BDD tests
└── docs/
    └── openapi.json              # Generated
```

✅ CHECKLIST POR MICROSERVICIO

Usa este checklist para cada uno de los 10 microservicios:

Setup Inicial

- ☐ cdk init app --language=typescript (PRIMERO)
- ☐ Instalar dependencias (zod, jest-cucumber, openapi3-ts)
- ☐ Crear estructura Clean Architecture
- ☐ Configurar .env.example
- ☐ Actualizar tsconfig.json con path aliases

Domain Layer

- ☐ Implementar entidades
- ☐ Implementar interfaces de repositorios
- ☐ Implementar errores de dominio
- ☐ Tests unitarios (>80% coverage)

Application Layer

- ☐ Implementar Use Cases (Create, Get, Update, Delete, List)
- ☐ Implementar DTOs con Zod schemas
- ☐ Tests BDD con jest-cucumber (Given-When-Then)

Infrastructure Layer

- ☐ Implementar repositorios DynamoDB
- ☐ Implementar Lambda handlers
- ☐ Configurar CloudWatch Logger
- ☐ HTTP utilities (apiResponse, errorHandler)

CDK Infrastructure

- ☐ Configurar DynamoDB Table
- ☐ Configurar Lambda Functions
- ☐ Configurar API Gateway
- ☐ Habilitar X-Ray tracing
- ☐ SI ES AUTH SERVICE: Exportar Lambda Authorizer ARN
- ☐ SI ES POST-AUTH: Importar Authorizer y configurar

Swagger/OpenAPI

- ☐ Implementar swaggerGenerator.ts
- ☐ Generar docs/openapi.json
- ☐ Probar /api-docs endpoint

Deployment

- ☐ cdk synth exitoso
- ☐ Deploy a DEV
- ☐ Probar todos los endpoints
- ☐ Verificar logs en CloudWatch

🏗️ ORDEN DE IMPLEMENTACIÓN

- 1️⃣ User Service - Repository Pattern (Días 1-2)

Objetivo: Estructura base reutilizable con Clean Architecture
Servicios AWS: Lambda + API Gateway + DynamoDB
Funcionalidades: CRUD completo, validación Zod, error handling, logging CloudWatch, Swagger
Conceptos clave: Clean Architecture, Dependency Injection, DTOs con Zod, BDD testing, Builder pattern

- 2️⃣ File Service - Valet Key Pattern (Días 3-4)

Objetivo: Manejo seguro de archivos
Servicios AWS: S3 + Lambda + API Gateway + DynamoDB
Funcionalidades: Upload/download con presigned URLs, metadata storage, file processing, S3 Events
Conceptos clave: Presigned URLs (GET/PUT), S3 security, multipart upload, event notifications

- 3️⃣ Notification Service - Pub/Sub (Días 5-7)

Objetivo: Mensajería asíncrona
Servicios AWS: SNS + SQS + Lambda
Funcionalidades: Publisher SNS, multiple subscribers SQS, Lambda consumers, DLQ, retry logic
Conceptos clave: Fanout pattern, message attributes, batch processing, idempotency

- 4️⃣ Product Service - CQRS (Días 8-10)

Objetivo: Separación de lecturas y escrituras
Servicios AWS: Lambda + DynamoDB + DynamoDB Streams + ElastiCache Redis
Funcionalidades: Command model (writes), Query model (reads con cache), streams para sync
Conceptos clave: Read/Write separation, eventual consistency, caching strategies, cache invalidation

- 5️⃣ Payment Service - Circuit Breaker (Días 11-13)

Objetivo: Resiliencia y manejo de fallos
Servicios AWS: Lambda + SQS + DLQ + CloudWatch Alarms
Funcionalidades: Circuit breaker, retry con exponential backoff, DLQ, health checks, alertas
Conceptos clave: Fault tolerance, graceful degradation, monitoring y alerting, error analysis

- 6️⃣ Container Service - ECS/Fargate (Días 15-18)

Objetivo: Containerización y orquestación
Servicios AWS: ECS/Fargate + ECR + Application Load Balancer + CloudWatch
Funcionalidades: Docker microservice, ECS task definitions, Fargate serverless, auto-scaling, health checks
Conceptos clave: Container best practices, Task vs Service, networking modes, resource allocation
🔒 **PROTEGIDO:** Importa Lambda Authorizer de Auth Service

- 7️⃣ Order Service - Saga Pattern (Días 19-22)

Objetivo: Orquestación de transacciones distribuidas
Servicios AWS: Step Functions + Lambda + DynamoDB + EventBridge
Funcionalidades: Workflow (validate → reserve → payment → fulfill), compensating transactions, state machine
Conceptos clave: Orchestration vs Choreography, distributed transactions, eventual consistency, error handling

- 8️⃣ Auth Service - Cognito + IAM (Días 23-26) ⚡ CRÍTICO

Objetivo: Autenticación y autorización
Servicios AWS: Cognito + Lambda Authorizers + API Gateway + Secrets Manager
Funcionalidades: User registration/login, JWT validation, Lambda Authorizer (EXPORTADO), IAM roles, MFA
Conceptos clave: OAuth 2.0/OpenID Connect, JWT structure, IAM best practices, CloudFormation Exports
⚡ **CRÍTICO:** Exporta Lambda Authorizer ARN con CfnOutput para microservicios 8-10

- 9️⃣ Analytics Service - X-Ray + Monitoring (Días 27-29)

Objetivo: Observabilidad y optimización
Servicios AWS: X-Ray + CloudWatch (Logs/Metrics/Alarms) + Lambda Insights
Funcionalidades: Distributed tracing, custom metrics, structured logging, dashboards, anomaly detection
Conceptos clave: Tracing vs Logging vs Metrics, service maps, performance profiling, Lambda power tuning
🔒 **PROTEGIDO:** Importa Lambda Authorizer, configura Security Schemes (Bearer JWT)

- 🔟 CI/CD Pipeline - Full Automation (Días 32-36)

Objetivo: Deployment automation completo
Servicios AWS: CodePipeline + CodeBuild + CodeDeploy + CloudFormation + S3
Funcionalidades: Pipeline completo, buildspec.yml, automated testing, blue/green, canary (10%→50%→100%), rollback
Conceptos clave: CI/CD best practices, deployment strategies, IaC, testing pyramid, pipeline stages

5. Quick Wins: Temas Fáciles Primero

Estos temas son FÁCILES de dominar y aparecen MUCHO en el examen. Dedica la primera semana:

1. Lambda Basics (2 días)

- Environment variables (NUNCA hardcodear secrets)
- Timeout (max 15 min)
- Memory (128MB - 10GB, más memory = más CPU)
- Concurrency limits (1000 default)

2. DynamoDB Basics (2 días)

- Partition key + Sort key
- Query (efficient) vs Scan (expensive)
- GSI/LSI (índices secundarios)

6. Recursos de Estudio Recomendados

📚 CURSOS

- **AWS Skill Builder:** ⭐⭐⭐ FREE

🎯 PRACTICE EXAMS (CRÍTICOS)

- **AWS Official:** 1 exam - $40 - ⭐⭐⭐⭐⭐

7. Estrategias para el Día del Examen

⏰ GESTIÓN DEL TIEMPO

- **130 minutos para 65 preguntas =** 2 minutos por pregunta
- Primera pasada (90 min): Responde con confianza
- Marca para revisar: Preguntas difíciles
- Segunda pasada (30 min): Revisar marcadas
- **NUNCA dejes preguntas sin responder**

📋 CHECKLIST LOGÍSTICO

- ☐ Confirmar fecha y hora del examen
- ☐ Elegir modalidad: Online o Test Center
- ☐ Probar webcam, micrófono, internet (si es online)
- ☐ Preparar documento ID: Pasaporte o DNI
- ☐ Descansar bien: 8 horas de sueño
- ☐ Llegar 30 min antes

8. Errores Comunes a Evitar

⚠️ TRAMPAS FRECUENTES

- **Lambda timeout de 30 min:** Max es 15 min → Usar Step Functions
- **DynamoDB Scan:** Lee TODA la tabla (caro) → Usar Query
- **Hardcodear credentials:** NUNCA → Usar IAM roles
- **Secrets sin encriptar:** Inseguro → Usar Secrets Manager

🚫 ERRORES DE ESTUDIO

- Solo ver videos sin práctica → Debes HACER labs
- No hacer practice exams → Fundamental para aprobar
- Memorizar en lugar de entender → El examen evalúa scenarios
- Posponer CI/CD → Es el 24% del examen y tu debilidad

9. Checklist Final Pre-Examen

✅ CONOCIMIENTO TÉCNICO

- ☐ Lambda: Environment vars, timeout, memory, concurrency
- ☐ API Gateway: REST vs HTTP, stages, throttling
- ☐ DynamoDB: PK/SK, Query vs Scan, GSI/LSI, Streams
- ☐ S3: Presigned URLs, versioning, lifecycle, events
- ☐ Cognito: User Pools, JWT validation, MFA
- ☐ CodePipeline: Stages, actions, approval gates
- ☐ CloudWatch: Logs, Metrics, Alarms, X-Ray
- ☐ ECS/Fargate: Task definitions, services

10. Anexos: Referencias Rápidas

⚙️ LÍMITES Y CUOTAS CLAVE

Servicio	Límite	Qué hacer
Lambda timeout	15 min max	Usar Step Functions para workflows largos
Lambda memory	128 MB - 10 GB	Más memory = más CPU
API Gateway timeout	29 sec	Para long-running, usar async (SQS + Lambda)
DynamoDB item size	400 KB	Usar S3 para large objects
SQS message size	256 KB	Usar S3 para large messages

¡VAS A APROBAR ESTE EXAMEN!

Tienes TODO lo necesario:

✅ 3+ años de experiencia REAL con AWS en producción
✅ Has migrado arquitecturas, optimizado sistemas
✅ Conoces Lambda, DynamoDB, SQS, SNS de memoria
✅ Entiendes arquitectura hexagonal y microservicios
✅ Has ganado hackathons (PRAXTHON, CAMUNDATHON)

Este examen NO es sobre memorizar teoría.
Es sobre demostrar que sabes USAR AWS para resolver problemas reales.

Nos vemos del otro lado, Developer! 🚀

Plan creado el 23 de diciembre de 2025
Basado en el exam guide oficial AWS DVA-C02