IT資格道場

DVA-C02 学習用テキスト

予備知識ゼロから合格ラインへ ―― 読んで理解する1冊目

無料サンプル(第1章まで)

AWS Certified Developer - Associate (DVA-C02) Exam Guide Version 1.3 準拠
2026年7月版

本書のご利用にあたって

- 本書は「IT資格道場」(www.shikaku-doujou.com)の購入者限定テキストのうち、学習用テキストの第1章までを収録した**無料サンプル**です。どなたでもご覧いただけます。
- 無料サンプルであっても著作物です。本書の転載・改変しての再配布・二次販売を禁じます。
- 本書を AI モデルの学習データとして利用すること(学習目的の入力・データセット化を含む)を禁じます。
- 製品版(学習用テキスト・暗記用ノート・頻出度別ノートの3点)は、ご購入者を識別できる透かし入りのPDFとして提供されます。
- 本PDFはスマートフォン・タブレットでの閲覧に合わせたA5版面・文字サイズで組版しています。

不正な配布を見つけた場合は info@shikaku-doujou.com までご連絡ください。

目次

はじめに —— この教材の使い方

このテキストは、AWS の開発者向け資格 **AWS Certified Developer - Associate(試験コード DVA-C02)** に合格することを目的に作られています。入門資格の Cloud Practitioner(CLF-C02)、設計者向けの Solutions Architect - Associate(SAA-C03)に続く、**「自分でコードを書いて AWS サービスを動かす」開発者**のための資格です。

DVA-C02 は、**アプリケーション開発者**の視点で出題されます。公式が想定する受験者像は「AWS サービスを使ったアプリの開発・保守を 1 年以上経験した人」で、**少なくとも 1 つの高級プログラミング言語(Java・C#・Python・JavaScript・TypeScript・Go など)が書けること**が前提に挙げられています。とはいえ、身構える必要はありません。問われるのは高度なアルゴリズムではなく、**「AWS の各サービスを、SDK・CLI・API を通じてどう呼び出し、どう組み合わせ、どう安全に運ぶか」**という実装の勘所です。本書はそこに必要な内容だけを、出題範囲(4 ドメイン)に沿って説明します。

SAA-C03 との一番の違い —— 「どれを選ぶ?」ではなく「どう実装する?」

設計者資格(SAA)は「要件に合う構成はどれか」を選ぶ試験でした。開発者資格(DVA)は違います。**同じサービスでも、開発者としての「使い方」まで踏み込みます。**

たとえば DynamoDB。SAA では「大量・高速の単純な出し入れなら DynamoDB」と**選べれば**十分でした。DVA では、その先 ——**「パーティションキーは何を選ぶと負荷が偏らないか」「Query と Scan はどう違い、どちらが効率的か」「結果整合性と強い整合性の読み取りをコードでどう指定するか」「重複実行を防ぐ条件付き書き込みはどう書くか」**まで問われます。S3 も「保

存する」で終わらず、「**署名付き URL を SDK でどう発行するか**」「**大きなファイルをマルチパートでどう上げるか**」を問われます。

つまり DVA は、**サービスの「開発者インターフェース(SDK・API・設定パラメータ)」を知っているか**を測る試験です。本書は各サービスを「開発者として、どのパラメータをどう設定し、どのメソッドをどう呼ぶか」の粒度で説明します。

試験の基本情報

項目	内容
試験名	AWS Certified Developer - Associate(DVA-C02)
問題数	65 問(うち採点対象 50 問+採点されない評価用 15 問。どれが評価用かは受験者には分からない)
出題形式	択一(4 択から 1 つ)と複数選択(5 つ以上から 2 つ以上)
試験時間	130 分
合格ライン	100〜1,000 のスケールスコアで **720**
合否表示	合格/不合格(pass/fail)の判定
採点方式	補償型(全体で 720 に届けばよく、ドメインごとの足切りはない)
受験料	20,000 円 ※2026 年 7 月時点の AWS 公式の日本円価格(アソシエイトレベル共通の料金表による)。現地通貨建てのため、為替に応じて少なくとも年 1 回(毎年 5 月)見直されます

項目	内容
受験方法	テストセンター(ピアソン VUE)または自宅からのオンライン監督試験
言語	日本語で受験可(英語・韓国語・ポルトガル語・簡体字中国語・スペイン語も提供)
前提資格	なし(いきなり受験可。ただしプログラミング経験と入門資格相当の知識は前提)

未回答は不正解扱いになるだけで、**間違えても減点はありません**(推測による解答にペナルティなし)。分からない問題も必ずどれかを選んで埋めてください。試験時間は 130 分、65 問に対して 1 問あたり 2 分の計算です。コードやシナリオを読む時間が要るので、時間管理は意識してください。受験料や実施形式は変わることがあるので、申し込み時に公式ページの最新情報を確認してください(本書は AWS 公式 Exam Guide の現行版 **DVA-C02 Version 2.1(2024 年 12 月 12 日改訂)** に対応しています。ドメイン構成・配点・問題数は旧 Version 1.3 から変わっていません)。

試験の設計図 —— 4 ドメインと配点

DVA-C02 の出題範囲(Exam Guide)は、4 つのドメインの配点比率を明示しています。**この比率が、そのまま学習時間の配分の目安**になります。本書の頻出度(A/B/C)も、この公式配点を最重視して決めています。

ドメイン	配点	一言でいうと
① AWS サービスを使った開発	**32%**	書く —— Lambda・DynamoDB・S3・API・メッセージングをコードで動かす

ドメイン	配点	一言でいうと
② セキュリティ	**26%**	守る —— 認証認可(Cognito/IAM/STS)・暗号化(KMS)・機密情報の扱い
③ デプロイ	**24%**	届ける —— SAM/CloudFormation・CI/CD・デプロイ戦略
④ トラブルシューティングと最適化	**18%**	直す・速くする —— X-Ray・CloudWatch・キャッシュ・同時実行

最大ドメインが**開発(32%)**、次いで**セキュリティ(26%)**です。この 2 つで **58%** を占めます。**開発とセキュリティを厚く固めるのが合格の最短路**です。デプロイ(24%)も配点が大きく、SAM・パイプライン・デプロイ戦略は確実に取りにいきます。

本書の構成 —— 開発者のワークフロー順

本書は、開発者が実際にアプリを作る流れ ——**書く → 守る → 届ける → 直す・速くする**—— の順で 10 章に分けて説明します。ドメインの並びとほぼ一致します。

- 第 1〜4 章 … ドメイン①(開発 —— SDK/CLI・Lambda・データストア・API/統合)
- 第 5〜6 章 … ドメイン②(セキュリティ —— 認証認可・暗号化)
- 第 7〜8 章 … ドメイン③(デプロイ —— 準備・CI/CD)
- 第 9〜10 章 … ドメイン④(トラブルシューティングと最適化)

3 ステップ学習法

購入者限定テキストは 3 冊で 1 セットです。次の順番で使うと最短です。

- **STEP 1(理解する)**: 本書「学習用テキスト」を通読し、各サービスの開発者としての使い方を頭に入れる
- **STEP 2(覚える)**: 「暗記用ノート」の一問一答で、パラメータ・メソッド・使い分けを即答できるまで繰り返す
- **STEP 3(仕上げる)**: 「頻出度別ノート」で頻出度 A の論点から総点検し、本サイトの問題演習で出題形式に慣れる

フル通読が難しければ、先に「頻出度別ノート」で頻出度 A の論点を確認し、それを含む章から読み始めても構いません。コード例は数行の短いものだけです。言語は問いませんが、本書は例に **Python(AWS SDK for Python = Boto3)** を使います。読めれば十分で、書けなくても合格には支障ありません。

第1章 開発者のためのAWS操作術 —— SDK・CLI・API(ドメイン① その1)

最初に、すべての開発の土台になる「**コードから AWS を操作する仕組み**」を押さえます。この章は公式タスク 1.1「AWS 上でホストされるアプリのコードを開発する」の前提知識と、アーキテクチャパターンに対応します。

1-1 AWS を動かす 3 つの入口 —— API・SDK・CLI

AWS のすべてのサービスは、根っこでは **API(Web サービスの呼び出し口)** で動いています。開発者がそれを叩く方法は 3 つです。

- **AWS SDK**: 各プログラミング言語向けの開発キット。Python なら **Boto3**、JavaScript なら AWS SDK for JavaScript というように、言語ごとに用意されている。コードから AWS を呼ぶときの主役
- **AWS CLI(コマンドラインインターフェース)**: ターミナルからコマンドで AWS を操作。スクリプトや動作確認に使う
- **API を直接呼ぶ**: SDK/CLI を使わず、署名付きの HTTP リクエストを自分で組む(通常は SDK に任せる)

DVA では「アプリのコードから AWS サービスを使うには?」→ **その言語の AWS SDK を使う**、が基本の答えです。

1-2 SDK が裏でやってくれること

SDK は単なる薄いラッパーではありません。開発者が意識すべき「裏方の仕事」を肩代わりします。DVA はこの中身を問います。

- **認証情報の解決(クレデンシャルプロバイダーチェーン)**: SDK は決まった順序で認証情報を探します ——**①コードに直接渡した値 → ②環境変数 → ③共有認証情報ファイル(`~/.aws/credentials`)→ ④IAM ロール(EC2 のインスタンスプロファイルや Lambda の実行ロール)。本番では認証情報をコードに書かず、IAM ロールで自動解決させる**のが鉄則です(詳細は第 5 章)
- **リージョンとエンドポイントの決定**: どのリージョンの API を叩くかを設定から解決する
- **リトライ(再試行)**: 一時的なエラー(スロットリング=流量制限や、瞬断)に対し、SDK が**自動でリトライ**します。しかもただ即座に繰り返すのではなく、**指数バックオフ(リトライのたびに待ち時間を倍々に延ばす)** と **ジッター(待ち時間に少しランダムなばらつきを足す)** を効かせます。これは「みんなが同時にリトライして相手をさらに詰まらせる」事故を防ぐ定石で、DVA の頻出概念です
- **ページネーション**: 一覧取得の結果が多いとき、続きを取る「次ページトークン」の面倒を見る

Boto3 なら、リトライの挙動は設定で調整できます。

```
import boto3
from botocore.config import Config

# 最大5回まで、指数バックオフ+ジッターのadaptiveモードで再試行
cfg = Config(retries={"max_attempts": 5, "mode": "adaptive"})
ddb = boto3.client("dynamodb", config=cfg)
```

「一時的なエラーにどう備えるか」と問われたら、**指数バックオフ+ジッターでリトライ**が反射で出るようにしてください。

1-3 べき等性(idempotency)―― 同じ処理を二度実行しても壊れない

べき等性とは、**同じ操作を何回実行しても結果が変わらない**という性質です。分散システムでは、ネットワークの再送やリトライで、**同じリクエストが二重に届くこと**が普通に起こります。このとき「注文が二重に確定する」「請求が二回走る」といった事故を防ぐのがべき等な設計です。

実装の定石は 2 つ。

- **一意なキーで重複を弾く**: リクエストごとに一意の ID(冪等キー)を持たせ、「その ID の処理が既にあれば何もしない」ようにする
- **条件付き書き込み**: データストア側の「既に無ければ書く」機能を使う。DynamoDB なら条件式で実現できます

```
# orderId が「まだ無い」ときだけ書き込む=二重登録を防ぐ(べき等)
ddb.put_item(
    TableName="Orders",
    Item={"orderId": {"S": order_id}, "status": {"S": "NEW"}},
    ConditionExpression="attribute_not_exists(orderId)",
)
```

「メッセージが二重配信されても大丈夫にしたい」→ **処理をべき等にする**、がDVA の頻出パターンです。

1-4 同期と非同期 —— 待つか、待たないか

- **同期(synchronous)**: 呼び出して**結果を待つ**。API Gateway →Lambda で即座にレスポンスを返す、など。処理が長いと呼び出し側が待たされる
- **非同期(asynchronous)**: 呼び出して**結果を待たず先へ進む**。あとで通知やキューで結果を受ける。重い処理・時間のかかる処理に向く

設計の反射:「時間のかかる処理を Web リクエストの裏で走らせたい」→ **非同期にして、キュー(SQS)や Lambda の非同期呼び出しに逃がす**。API のタイムアウト(後述、API Gateway は既定 29 秒)を超える処理は、同期で待たせてはいけません。

1-5 疎結合とアーキテクチャパターン

DVA でも、SAA と同じく**疎結合(部品を直接つながず間に緩衝材を挟む)** が重視されます。開発者としては「コードの中で直接呼び出す(密結合)」のではなく、「キューやイベントを挟んで切り離す」実装を選べることが問われます。

公式タスクに列挙されたパターン用語も、言葉として押さえておきます。

- **モノリシック**: 全機能を 1 つの塊で作る。シンプルだが、一部の変更・障害が全体に及ぶ
- **マイクロサービス**: 機能ごとに小さく分けて独立にデプロイ。疎結合の代表
- **イベント駆動(event-driven)**:「何かが起きた(イベント)」をきっかけに処理が動く。Lambda の得意技
- **ファンアウト(fanout)**: 1 つのイベントを**複数の宛先へ同時に配る**(SNS →複数の SQS など)

- **オーケストレーション vs コレオグラフィ**: 中央の指揮者が全体を順に制御するのが**オーケストレーション**(Step Functions)、各サービスがイベントを見て自律的に動くのが**コレオグラフィ**(EventBridge 中心)
- **ステートフル vs ステートレス**: 状態を自分の中に持つのがステートフル、持たないのがステートレス。**スケールする設計はステートレス**(状態は外部の DynamoDB/ElastiCache に置く)
- **フォールトトレラント設計**: 失敗しても壊れない工夫。**リトライ(指数バックオフ+ジッター)** と、**デッドレターキュー(処理できなかったメッセージの退避先)** が二本柱
- **サーキットブレーカー(circuit breaker)**: 呼び出し先が**繰り返し失敗している間は、そのサービスを呼ぶこと自体をいったんやめて、すぐにエラー(または代替の応答)を返す**パターン。電気のブレーカーと同じ発想で、落ちている相手を叩き続けて自分の処理まで詰まる「連鎖障害」を防ぎます。一定時間が経ったら試しに少しだけ通し、回復していれば通常運転に戻します。**リトライは「もう一度やってみる」、サーキットブレーカーは「しばらくやめておく」** —— 対で覚えてください

1-6 Amazon Q Developer で開発を助けてもらう【2026-08-16 追記】

Exam Guide の **Version 2.1** で新設されたスキル **1.1.11「Amazon Q Developer を使って開発を支援する」** に対応する節です。**Amazon Q Developer は、V2.1 で in-scope サービス一覧に加わった唯一のサービス**なので、名前と役割を必ず押さえてください。

ひとことで言うと: AWS の生成 AI 開発アシスタントです。AWS の使い方を質問できるだけでなく、**IDE(Visual Studio Code・JetBrains・Visual Studio など)に拡張機能として入れて、コードを書く作業そのものを手伝わせます**。

中身は Amazon Bedrock の基盤モデルで、AWS のドキュメントを学習させた回答が返ります。

IDE でできること(この 6 つを役割で覚える)

機能	何をするか	旧称
インラインコード提案 (inline suggestions)	入力中のコードと開いているファイルを文脈に、**続きのコードをその場で提示**	——
エージェント型コーディング/機能開発	実現したいことを説明すると、**プロジェクトの文脈からコードを生成してファイルへ適用**(差分で確認・取り消し可)	`/dev`
ユニットテスト生成	対象コードの**ユニットテストを生成**し、開発サイクル全体でテストを自動化(第 8 章・スキル 3.3.6)	`/test`
ドキュメント生成	**README などの文書を作成・更新**	`/doc`
コードレビュー	コードベースを走査して**セキュリティ脆弱性とコード品質の問題を報告**	`/review`
コード変換	**Java の言語バージョンと依存関係を引き上げる** (Java 8/11 → 17・21 など)。.NET の移行にも対応	`/transform`

- **旧称の `/dev`・`/test`・`/doc`・`/review` はスラッシュコマンドとして案内されていた名残**です。**古い教材ではこの名前で出てくる**ので、「`/test` =ユニットテスト生成」のように**旧称と機能名を対で**覚えておくと、どちらの書

き方で問われても迷いません。現在チャットに残っているコマンドは `/clear`(会話を消す)・`/compact`(履歴を要約)・`/help` の 3 つです

- **コード変換の前提**: **変換元と同じバージョンの JDK がローカルに要ります**。Amazon Q はまず変換元の言語バージョンでビルドして情報が揃っているか確かめてから変換するため、ローカルにビルド環境が無いと始まりません(Maven でビルドできることも条件)。「JDK なしでクラウド側だけで変換できる」は誤りです
- **IDE 以外の入口**: AWS マネジメントコンソール・AWS ドキュメントサイト上のチャット、**コンソールの操作をコードに変換する Console-to-Code**、コマンドラインの Q CLI、Slack/Microsoft Teams などのチャットアプリからも使えます

設計の反射:「AI に開発を手伝わせるなら?」→ **Amazon Q Developer**。「テストコードを自動で作らせたい」→ **ユニットテスト生成(旧 `/test`)**。「脆弱性と品質をまとめて見てほしい」→ **コードレビュー(旧 `/review`)**。「Java 8 のアプリを新しい版へ」→ **コード変換(旧 `/transform`)**。

第1章の試験ポイント

- コードから AWS を使う主役は**言語ごとの SDK**(Python は Boto3)。認証情報は**コードに書かず IAM ロールで自動解決**(プロバイダーチェーン)
- 一時エラー・スロットリングへの備えは **SDK の自動リトライ+指数バックオフ+ジッター**
- **べき等性**=同じ処理を二度やっても壊れない。実装は**一意キー**や **DynamoDB の条件付き書き込み(attribute_not_exists)**
- 長い処理は**非同期**へ(API Gateway の既定 29 秒を超えるものを同期で待たせない)

- 疎結合・イベント駆動・ファンアウト・ステートレス・DLQ が開発ドメインの背骨
- **Amazon Q Developer(V2.1 で in-scope 追加)**: インライン提案/機能開発(旧 `/dev`)/**ユニットテスト生成(旧 `/test`)**/ドキュメント生成(旧 `/doc`)/**コードレビュー(旧 `/review`)**/コード変換(旧 `/transform`)。**コード変換は変換元と同じ JDK がローカルに要る**

サンプルはここまでです

続き(第2章〜巻末)は、DVA-C02 問題集のご購入で、暗記用ノート・頻出度別ノートと合わせて3点すべてダウンロードできます。

DVA-C02 学習用テキスト(無料サンプル)

版
2026年7月版(AWS Certified Developer - Associate (DVA-C02) Exam Guide Version 1.3 準拠)

発行
IT資格道場(https://www.shikaku-doujou.com)

お問い合わせ
info@shikaku-doujou.com