(1) ある企業が、レガシーアプリケーションを Amazon EC2 に移行しようとしています。MySQL データベースに接続するためのユーザー名とパスワードが、アプリケーションのソースコードに直接コーディングされています。このデータベースは、Amazon RDS for MySQL データベースインスタンスに移行される予定です。この企業は、移行プロセスの一環として、データベース認証情報を格納および自動ローテートするためのセキュアな方法を実装したい、と考えています。

これらの要件を満たすには、どうすればよいですか。

A) データベース認証情報を Amazon Machine Image (AMI) の環境変数内に格納する。AMI を置換することによって認証情報をローテートする。
B) データベース認証情報を AWS Systems Manager の Parameter Store に格納する。認証情報を自動ローテートするよう、Parameter Store を構成する。
C) データベース認証情報を EC2 インスタンス上の環境変数内に格納する。EC2 インスタンスを再起動することによって認証情報をローテートする。
D) データベース認証情報を AWS Secrets Manager に格納する。認証情報を自動ローテートするよう、Secrets Manager を構成する。

(2) コメントを投稿したユーザーがほぼリアルタイムでフィードバックを受け取ることができる機能を備えた Web アプリケーションを、開発者が設計しています。

これらの要件を満たすには、どうすればよいですか (2 つ選択してください)。

A) AWS AppSync スキーマ、および対応する API を作成する。Amazon DynamoDB テーブルをデータストアとして使用する。
B) Amazon API Gateway 内に WebSocket API を作成する。AWS Lambda 関数をバックエンドとして使用する。Amazon DynamoDB テーブルをデータストアとして使用する。
C) Amazon RDS データベースをバックエンドとして使用する AWS Elastic Beanstalk アプリケーションを作成する。長寿命 TCP/IP ソケットを許可するよう、アプリケーションを構成する。
D) Amazon API Gateway 内に GraphQL エンドポイントを作成する。Amazon DynamoDB テーブルをデータストアとして使用する。
E) Amazon CloudFront で WebSocket を有効化する。AWS Lambda 関数をオリジンとして使用する。Amazon Aurora データベースクラスターをデータストアとして使用する。

(3) 開発者が、サインアップ機能とサインイン機能をアプリケーションに追加しようとしています。このアプリケーションでは、カスタム分析ソリューションに対する API 呼び出しを実行し、ユーザーのサインインイベントをロギングする必要があります。

これらの要件を満たすには、どうすればよいですか (2 つ選択してください)。

A) Amazon Cognito を使用して、サインアップ機能とサインイン機能を追加する。
B) AWS IAM を使用して、サインアップ機能とサインイン機能を追加する。
C) 認証後イベントによってトリガされる API 呼び出しを実行するよう、AWS Config ルールを構成する。
D) 認証後イベントによってトリガされる API 呼び出しを実行する Amazon API Gateway メソッドを呼び出す。
E) 認証後イベントによってトリガされる API 呼び出しを実行する AWS Lambda 関数を実行する。

(4) ある企業が、ある AWS アカウント内の REST API に対して Amazon API Gateway を使用しています。セキュリティチームは、別の AWS アカウント内の IAM ユーザーだけがこれらの API にアクセスできるようにしたいと考えています。

これらの要件を満たすには、どうすればよいですか (2 つ選択してください)。

A) IAM 権限ポリシーを作成し、各 IAM ユーザーにアタッチする。API に対するメソッド承認タイプを AWS_IAM に設定する。署名バージョン 4 を使用して API リクエストに署名する。
B) Amazon Cognito ユーザープールを作成し、各 IAM ユーザーをこのプールに追加する。API に対するメソッド承認タイプを COGNITO_USER_POOLS に設定する。Amazon Cognito の IAM 認証情報を使用して認証し、ID トークンをリクエストヘッダーに追加する。
C) Amazon Cognito アイデンティティプールを作成し、各 IAM ユーザーをこのプールに追加する。API に対するメソッド承認タイプを COGNITO_USER_POOLS に設定する。Amazon Cognito の IAM 認証情報を使用して認証し、アクセストークンをリクエストヘッダーに追加する。
D) 各 IAM ユーザーにのみアクセスを許可する、API に対するリソースポリシーを作成する。
E) 各 IAM ユーザーにのみアクセスを許可する、API に対する Amazon Cognito オーソライザーを作成する。API に対するメソッド承認タイプを COGNITO_USER_POOLS に設定する。

(5) 開発者が、テキストファイルを .pdf ファイルに変換するアプリケーションを作成しています。このテキストファイルは、別のアプリケーションによって変換元 Amazon S3 バケットに格納されています。開発者は、テキストファイルが Amazon S3 に格納されたときにそのテキストファイルを読み取り、AWS Lambda を使用して .pdf ファイルに変換したいと考えています。開発者は、Amazon S3 および Amazon CloudWatch Logs に対するアクセスを許可する IAM ポリシーをすでに作成しています。

この Lambda 関数に適切な権限を付与するには、どうすればよいですか。

A) AWS IAM を使用して、Lambda 実行ロールを作成する。IAM ポリシーをこのロールにアタッチする。この Lambda 実行ロールを Lambda 関数に割り当てる。
B) AWS IAM を使用して、Lambda 実行ユーザーを作成する。IAM ポリシーをこのユーザーにアタッチする。この Lambda 実行ユーザーを Lambda 関数に割り当てる。
C) AWS IAM を使用して、Lambda 実行ロールを作成する。IAM ポリシーをこのロールにアタッチする。IAM ロールを Lambda 関数内に環境変数として格納する。
D) AWS IAM を使用して、Lambda 実行ユーザーを作成する。IAM ポリシーをこの ユーザーにアタッチする。IAM ユーザー認証情報を Lambda 関数内に環境変数として格納する。

(6) ある企業の AWS ワークロードが、複数の地域に分散しています。開発者は、Amazon Aurora データベースを us-west-1 リージョン内に作成しました。このデータベースは、顧客管理型 AWS KMS キーを使用して暗号化されています。開発者は、同様の暗号化データベースを us-east-1 リージョン内に作成したいと考えています。

この要件を満たすには、どうすればよいですか。

A) データベースのスナップショットを us-west-1 リージョン内に作成する。このスナップショットを us-east-1 リージョンにコピーし、us-east-1 リージョン内の KMS キーを指定する。コピーしたスナップショットからデータベースを復元する。
B) データベースの非暗号化スナップショットを us-west-1 リージョン内に作成する。このスナップショットを us-east-1 リージョンにコピーする。コピーしたスナップショットからデータベースを復元する。us-east-1 リージョン内の KMS キーを使用して、暗号化を有効化する。
C) データベースの暗号化を無効化する。データベースのスナップショットを us-west-1 リージョン内に作成する。このスナップショットを us-east-1 リージョンにコピーする。コピーしたスナップショットからデータベースを復元する。
D) us-east-1 リージョンで、us-west-1 リージョン内のデータベースの前回の自動バックアップデータからデータベースを復元する。us-east-1 リージョン内の KMS キーを使用して、暗号化を有効化する。

(7) 開発者が、Amazon ElastiCache for Memcached を既存のレコード格納アプリケーションに追加しようとしています。その目的は、データベースの負荷を減らし、パフォーマンスを向上させることです。開発者は、典型的なレコード処理パターンの分析結果に基づき、遅延読み込み機能を使用することにしました。

遅延読み込み機能を適切に実装するための擬似コード例はどれですか。

A)
```
record_value = db.query("UPDATE Records SET Details = {1} WHERE ID == {0}", record_key, record_value)
cache.set (record_key, record_value)
```
B)
```
record_value = cache.get(record_key)
if (record_value == NULL)
    record_value = db.query("SELECT Details FROM Records WHERE ID == {0}", record_key)
    cache.set (record_key, record_value)
```
C)
```
record_value = cache.get (record_key)
db.query("UPDATE Records SET Details = {1} WHERE ID == {0}", record_key, record_value)
```
D)
```
record_value = db.query("SELECT Details FROM Records WHERE ID == {0}", record_key)
if (record_value != NULL)
    cache.set (record_key, record_value)
```

(8) 開発者が、Amazon EC2 インスタンスフリート上で動作するアプリケーションのパフォーマンスを追跡したいと考えています。また、インスタンスフリート全体に関する統計情報 (例: 平均リクエスト遅延、最大リクエスト遅延) を表示および追跡したいと考えています。さらに、平均応答時間がしきい値を上回ったとき、すぐに通知を受信したいとも考えています。

これらの要件を満たすには、どうすればよいですか。

A) 応答時間を測定し、Amazon S3 バケット内のログファイルを毎分更新する cron ジョブを各インスタンス上で構成する。ログファイルを読み取って新しいエントリを Amazon Elasticsearch Service (Amazon ES) クラスターに書き込む AWS Lambda 関数を、Amazon S3 イベント通知を使用してトリガする。結果を Kibana ダッシュボードに表示する。応答時間がしきい値を上回ったときに Amazon SNS トピックにアラートを送信するよう、Amazon ES を構成する。
B) 応答時間をシステムログに書き込むよう、アプリケーションを構成する。Amazon Inspector エージェントをインストールする。ログを継続的に読み取って応答時間を Amazon EventBridge に送信するよう、Inspector エージェントを構成する。メトリクスグラフを EventBridge コンソールに表示する。応答時間メトリクスの平均値がしきい値を上回ったときに Amazon SNS 通知を送信するよう、EventBridge カスタムルールを構成する。
C) 応答時間をログファイルに書き込むよう、アプリケーションを構成する。Amazon CloudWatch エージェントをインスタンスにインストールする。アプリケーションログを CloudWatch Logs にストリーミングするよう、CloudWatch エージェントを構成する。ログから応答時間のメトリクスフィルタを作成する。メトリクスグラフを CloudWatch コンソールに表示する。応答時間メトリクスの平均値がしきい値を上回ったときに Amazon SNS 通知を送信する CloudWatch アラームを作成する。
D) AWS Systems Manager エージェントをインスタンスにインストールする。応答時間を監視して Amazon CloudWatch にカスタムメトリクスとして送信するよう、Systems Manager エージェントを構成する。メトリクスグラフを Amazon QuickSight に表示する。応答時間メトリクスの平均値がしきい値を上回ったときに Amazon SNS 通知を送信する CloudWatch アラームを作成する。

(9) 開発者が、あるアプリケーションをローカル環境でテストしており、そのアプリケーションを AWS Lambda に展開しました。パッケージサイズを上限値以下に抑えるため、依存関係を展開ファイルに含めませんでした。アプリケーションをリモートでテストしたとき、依存関係が欠落しているため、関数が実行されませんでした。

この問題を解決するには、どうすればよいですか。

A) Lambda コンソールエディタを使用して、コードを更新し、欠落している依存関係を含めるようにする。
B) 欠落している依存関係を含めた .zip ファイルを別途作成し、このファイルを元の Lambda 展開パッケージに含める。
C) 欠落している依存関係への参照を、Lambda 関数の環境変数に追加する。
D) 欠落している依存関係を含めたレイヤーを Lambda 関数にアタッチする。

(10) 開発者が、Amazon API Gateway を使用する Web アプリケーションを作成しています。開発者は、開発ワークロード用環境と本番ワークロード用環境を別々に管理したいと考えています。この API では、AWS Lambda 関数を使用します。この関数には、開発用と本番用のエイリアスをそれぞれ割り当てます。

最小限の構成作業量でこの要件を満たすには、どうすればよいですか。

A) 各環境用の REST API を作成する。API をそれぞれ、Lambda 関数の開発用エイリアスおよび本番用エイリアスと結び付ける。この 2 つの API をそれぞれのステージに展開し、ステージ URL を使用して API にアクセスする。

B) REST API を 1 個作成する。エイリアスではなくステージ変数を使用して、API を Lambda 関数と結び付ける。その後、API を開発用ステージおよび本番用ステージに展開する。各ステージ内でステージ変数を作成する。その際、変数の値として別々のエイリアスを使用する。それぞれのステージ URL を使用して API にアクセスする。

C) REST API を 1 個作成する。この API を Lambda 関数の開発用エイリアスと結び付け、開発環境に展開する。カナリアを Lambda 関数の本番用エイリアスと結び付けるよう、本番用カナリアリリース展開を構成する。

D) REST API を 1 個作成する。この API を Lambda 関数の本番用エイリアスと結び付け、本番環境に展開する。カナリアを Lambda 関数の開発用エイリアスと結び付けるよう、開発用カナリアリリース展開を構成する。

解答

(1) D - AWS Secrets Manager は、データベース、アプリケーション、サービスなどの IT リソースにアクセスするために必要な認証情報を保護するのに役立ちます。ユーザーは Secrets Manager を使用することにより、データベース認証情報や API キーなどのシークレットを、そのシークレットのライフサイクルを通じて簡単にローテート、管理、および取得できます。ユーザーおよびアプリケーションがシークレットを取得するには、Secrets Manager API を呼び出します。機密情報をプレーンテキストでハードコーディングする必要はありません。また、組み込みの統合機能を使用して Amazon RDS、Amazon Redshift、および Amazon DocumentDB と統合することにより、シークレットをローテートできます。

(2) A、B - AWS AppSync を使用した場合、ユーザーは、柔軟性の高い API を作成して 1 個以上のデータソース内のデータにセキュアな方法でアクセスし、データを操作および結合することができます。これにより、アプリケーション開発が容易になります。AWS AppSync は、GraphQL を使用するマネージドサービスです。AWS AppSync を使用することにより、アプリケーションで必要なデータだけを簡単に取得できます。ユーザーは AWS AppSync を使用することにより、Amazon DynamoDB などのさまざまなデータソースを使用する拡張性の高いアプリケーション (例: データをリアルタイムで更新する必要があるアプリケーション) を作成できます。ユーザーは Amazon API Gateway を使用することにより、WebSocket API を AWS サービス (例: AWS Lambda、DynamoDB) 用または HTTP エンドポイント用のステートフルフロントエンドとして作成できます。WebSocket API は、クライアントアプリケーションから受信したメッセージの内容に基づいて、バックエンドを呼び出します。リクエストを受信してそれに応答する REST API と異なり、WebSocket API は、クライアントアプリケーションとバックエンドの間の双方向通信をサポートしています。

(3) A、E - Amazon Cognito を使用すれば、ユーザーサインアップ機能、ユーザーサインイン機能、およびアクセスコントロール機能を Web アプリケーションやモバイルアプリケーションに簡単に追加できます。また、ユーザーは、カスタム分析ソリューションに対する API 呼び出しを実行する AWS Lambda 関数を作成し、Amazon Cognito 認証後トリガを使用してその関数をトリガすることができます。

(4) A、D - リソースポリシーと署名バージョン 4 (SigV4) プロトコルを使用することにより、ある AWS アカウント内のユーザーに対して、別の AWS アカウントに対する API アクセスを許可することができます。

(5) A - AWS Lambda 関数の実行ロールは、AWS サービスおよび AWS リソースへのアクセスを Lambda 関数に許可するものです。関数の作成時にこの実行ロールを割り当てます。それにより、関数が呼び出されたとき、Lambda によってこの実行ロールが代行されます。

(6) A - ユーザーが暗号化スナップショットをコピーする場合、スナップショットのコピーも暗号化する必要があります。ユーザーが暗号化スナップショットをリージョン間でコピーする場合、コピー元スナップショットに対して使用されているのと同じ AWS KMS 暗号化キーを、スナップショットのコピーに対して使用することはできません。KMS キーはリージョン固有であるからです。代わりに、コピー先リージョン内で有効である KMS キーを指定する必要があります。

(7) B - 遅延読み込みとは、あるレコードが必要になるまでそのレコードの読み込みを遅らせることです。遅延読み込みでは、まずキャッシュが検査されます。当該レコードが存在していない場合、遅延読み込み機能によってデータベースからそのレコードが取得され、キャッシュに格納されます。

(8) C - ログとメトリクスを Amazon CloudWatch にストリーミングするよう、CloudWatch エージェントを構成できます。また、CloudWatch Logs に格納されているログからメトリクスフィルタを作成できます。

(9) D - ユーザーは、追加のコードとコンテンツをレイヤー形式で取り込むよう、AWS Lambda 関数を構成できます。レイヤーとは、ライブラリやカスタムランタイムなどの依存関係を含めた .zip アーカイブのことです。レイヤーを使用した場合、ユーザーは、ライブラリを展開パッケージに含めなくても、関数内でライブラリを使用できます。

(10) B - ユーザーは、Amazon API Gateway の展開ステージを使用することにより、各 API に対する複数のリリースステージ (例: アルファ、ベータ、本番) を管理できます。API 展開ステージは、構成可能なステージ変数に基づいて、別々のバックエンドエンドポイントとやりとりできます。ユーザーは、API Gateway のステージ変数を使用することにより、複数のバージョンとエイリアスを持つ 1 個の AWS Lambda 関数を参照できます。