1）あなたは Amazon EBS ボリュームを使用する Amazon EC2 上で実行されているアプリケーションサーバー向けに、自動データバックアップソリューションを導入する業務を担当しています。単一障害点を回避し、データの耐久性を高めるために、分散データストアを使用してバックアップを取りたいと考えています。また、データを 1 時間以内に復元できるように、毎日のバックアップを 30 日間保存する必要があります。

アプリケーションサーバーでスケジューリングデーモンにより毎日スクリプトを実行し、この仕組みを実装するには、どうすればよいですか?

A) ec2-create-volume API を呼び出すためのスクリプトを作成し、現在の日時を使用して Amazon EBS ボリュームにタグを付けて、バックアップデータを 2 つ目の Amazon EBS ボリュームにコピーする。ec2-describe-volumes API を使用して既存のバックアップボリュームを列挙する。ec2-delete-volume API を呼び出して、30 日より前の日時のタグが付けられているバックアップボリュームを削除する。

B) Amazon Glacier UploadArchive API を呼び出すためのスクリプトを作成し、現在の日時を使用してバックアップアーカイブにタグを付ける。ListVaults API を使用して既存のバックアップアーカイブを列挙する。DeleteVault API を呼び出して、30 日より前の日時のタグが付けられているバックアップアーカイブを削除する。

C) ec2-create-snapshot API を呼び出すためのスクリプトを作成し、現在の日時を使用して Amazon EBS スナップショットにタグを付ける。ec2-describe-snapshots API を使用して既存の Amazon EBS スナップショットを列挙する。ec2-delete-snapshot API を呼び出して、30 日より前の日時のタグが付けられている Amazon EBS スナップショットを削除する。

D) ec2-create-volume API を呼び出すためのスクリプトを作成し、現在の日時を使用して Amazon EBS ボリュームにタグを付けて、ec2-copy-snapshot API を使用して新しい Amazon EBS ボリュームにデータをバックアップする。ec2-describe-snapshot API を使用して既存のバックアップボリュームを列挙する。ec2-delete-snaphot API を呼び出して、30 日より前の日時のタグが付けられている Amazon EBS バックアップボリュームを削除する。

2）あなたはモバイルデバイス向けの新しい写真共有アプリケーションを開発したベンチャー企業に勤めています。ここ数か月にわたって、このアプリケーションの人気が高まってきています。このため、負荷の増大が原因となって、アプリケーションのパフォーマンスが低下してしまいしました。

このアプリケーションでは、Auto Scaling が設定された PHP アプリケーションの層と、当初 AWS CloudFormation を使用してデプロイされた MySQL RDS インスタンスから構成される、2 層アーキテクチャを採用しています。また、Auto Scaling group には、最小値 4 と最大値 8 が設定されています。インスタンスの CPU 使用率が高いため、現在の希望する容量は 8 です。

分析の結果、パフォーマンス上の問題の原因は CPU 能力の制約であると確信しました。一方で、メモリの使用率は低いままです。そのため、M3 汎用インスタンスから C3 コンピューティング最適化インスタンスに移行することにします。

エンドユーザーに対するサービス中断を最小限に抑えながらこの変更をデプロイするには、どうしますか?

A) AWS マネジメントコンソールにサインインし、既存の起動設定をコピーして、C3 インスタンスを指定した新しい起動設定を作成する。また、新しい起動設定を使用して Auto Scaling group を更新する。その後、Auto Scaling によって、実行されているすべてのインスタンスのインスタンスタイプが更新される。
B) AWS マネジメントコンソールにサインインし、新しい C3 インスタンスタイプを使用して既存の起動設定を更新する。Auto Scaling group に、AutoScaling RollingUpdate を指定した UpdatePolicy 属性を追加する。
C) AWS CloudFormation テンプレートで指定されている起動設定を、新しい C3 インスタンスタイプを使用して更新する。そして、新しいテンプレートを使用して、スタックの更新を実行する。その後、Auto Scaling によって、新しいインスタンスタイプに指定され、インスタンスが更新される。
D) AWS CloudFormation テンプレートで指定されている起動設定を、新しい C3 インスタンスタイプを使用して更新する。また、Auto Scaling group に、AutoScalingRollingUpdate を指定した UpdatePolicy 属性を追加し、そして、新しいテンプレートを使用して、スタックの更新を実行する。

3）ネットワーキング、IAM ポリシー、および複数の 3 層アプリケーションが関係する、複雑なシステムがあります。この新システムの要件は現在も更新されているため、最終的な設計に含まれる AWS コンポーネントの数はまだわかりません。

インフラストラクチャの自動化とバージョン管理を行えるように、AWS CloudFormation を利用してこれらの AWS リソースの定義を始めたいとします。

俊敏性を備えた新環境を費用対効果と信頼性の高い方法で顧客に提供するために、AWS CloudFormation をどのように利用しますか?

A) システムで必要なすべてのリソースが含まれる単一のテンプレートを手動で作成し、バージョン管理を適用するテンプレートが 1 つだけになるようにする。
B) システムの論理パートごとに別個のテンプレートを複数作成し、AWS CloudFormation でネストされたスタックを作成して、複数のテンプレートにバージョン管理を適用する。
C) システムの論理パートごとに別個のテンプレートを複数作成し、SDK がインストールされている Amazon Elastic Compute Cloud (EC2) インスタンスを使用して出力を次々に提供して、より詳細に管理する。
D) ネットワーキングレイヤーは頻繁には変更されないため、これを Amazon Virtual Private Cloud (VPC) を使用して手動で構成してから、その他すべての一時リソースを AWS CloudFormation を利用して定義する。

4）あなたは、Auto Scaling group に含まれる Amazon EC2 インスタンスが起動された後、設定とアプリケーションの動的なデプロイを自動化するためのシステムを導入しました。このシステムでは、マスターノードのないスタンドアロン構成で機能する設定管理ツールを使用しています。アプリケーションの負荷は随時変化するため、新しいインスタンスは、インスタンスのオペレーティングシステムの起動後 3 分以内に稼働状態にする必要があります。各デプロイステージが完了するまでかかる時間は以下のとおりです。

- **設定管理エージェントのインストール: 2 分**
- **アーティファクトを使用したインスタンスの設定: 4 分**
- **アプリケーションフレームワークのインストール: 15 分**

- **アプリケーションコードのデプロイ: 1 分**

このようなスタンドアロンのエージェント構成でのデプロイを自動化するには、どの手順を採用する必要がありますか?

A) エージェントをインストールするための Amazon EC2 UserData スクリプトを使用して Auto Scaling 起動設定を構成し、Amazon S3 バケットから設定アーティファクトとアプリケーションコードを取得してから、エージェントを実行してインフラストラクチャとアプリケーションを設定する。

B) エージェント、設定アーティファクト、アプリケーションフレームワーク、コードといったすべてのコンポーネントが事前にインストールされた、カスタム Amazon マシンイメージを作成する。エージェントを実行して起動時にシステムを設定するための起動スクリプトを作成する。

C) 設定管理エージェントとアプリケーションフレームワークが事前にインストールされた、カスタム Amazon マシンイメージを作成する。Amazon S3 バケットから設定アーティファクトとアプリケーションコードを取得するための Amazon EC2 UserData スクリプトを使用して Auto Scaling 起動設定を構成してから、エージェントを実行してシステムを設定する。

D) Amazon EC2 API をポーリングして、Auto Scaling group 内で起動される新しいインスタンスの有無を確認するためのウェブサービスを作成する。新しいインスタンスが認識されたら、エージェントをインストールするためのリモートスクリプトを SSH 経由で実行し、設定アーティファクトとアプリケーションコードを SCP でコピーして、最後にエージェントを実行してシステムを設定する。

5）継続的デプロイプロセスの一環として、アプリケーションには、新しい AMI を使用して本番環境にデプロイされる前に、I/O 負荷パフォーマンステストを実施します。このアプリケーションでは、インスタンスごとに 1 つの Amazon EBS PIOPS ボリュームを使用しており、一貫した I/O パフォーマンスが求められます。

I/O 負荷パフォーマンステストにおいて、適切な結果を繰り返し可能な方法で得るには、以下のうちどれを行う必要がありますか?

A) テスト時の I/O ブロックサイズがランダムに選択されるようにする。

B) テスト前にすべてのブロックに対して読み取りを実行することで、Amazon EBS ボリュームの事前ウォーミングが行われるようにする。

C) バックアップとして、Amazon EBS ボリュームのスナップショットが作成されるようにする。

D) Amazon EBS ボリュームが暗号化されるようにする。

E) テスト前にボリュームのスナップショットを作成することで、Amazon EBS ボリュームの事前ウォーミングが行われるようにする。

6）あなたのソーシャルメディアマーケティングアプリケーションに、AWS Elastic Beanstalk で実行される Ruby で記述されたコンポーネントが含まれています。このアプリケーションコンポーネントは、さまざまなマーケティングキャンペーンを支援するために、ソーシャルメディアサイトにメッセージを投稿します。経営陣は、マーケティングキャンペーンの有効性を過去および将来の取り組みと比較して分析するため、こうしたソー

シャルメディアのメッセージへの返信を記録するよう求めています。返信を読み取るためにソーシャルメディアサイトの API と連携する新しいアプリケーションコンポーネントは、既に開発済みです。

履歴データを分析する際にいつでもアクセスできる堅牢なデータストアにソーシャルメディアの返信を記録するには、どの手順を採用すべきですか?

A) Amazon EC2 インスタンスの Auto Scaling group に新しいアプリケーションコンポーネントをデプロイし、ソーシャルメディアサイトからデータを読み取って、そのデータを Amazon Elastic Block Store を使用して保存し、AWS Data Pipeline によって Amazon Kinesis に公開して分析する。

B) 新しいアプリケーションコンポーネントを Elastic Beanstalk アプリケーションとしてデプロイし、ソーシャルメディアサイトからデータを読み取って、そのデータを DynamoDB に保存し、Apache Hive と Amazon Elastic MapReduce を併用して分析する。

C) Amazon EC2 インスタンスの Auto Scaling group に新しいアプリケーションコンポーネントをデプロイし、ソーシャルメディアサイトからデータを読み取って、そのデータを Amazon Glacier に保存し、AWS Data Pipeline によって Amazon RedShift に公開して分析する。

D) 新しいアプリケーションコンポーネントを Amazon Elastic Beanstalk アプリケーションとしてデプロイし、ソーシャルメディアサイトからデータを読み取って、そのデータを Amazon Elastic Block Store を使用して保存し、Amazon Kinesis によって Amazon CloudWatch にデータをストリーミングして分析する。

7）前四半期の月々の請求額を確認したところ、経営陣は Amazon からの請求額全体が増加していることに気付きました。あなたがこのコスト増加について調査した結果、新しいサービスの 1 つが、アプリケーションのバケット内にあるすべてのオブジェクトのメタデータのキャッシュを構築するために、Amazon S3 に対して多数の GET Bucket API 呼び出しを行っていることを発見しました。上司は、こうした GET Bucket API の新たな呼び出しの数を減らすための、費用対効果に優れた新しい方法を考案するよう求めています。

このコストを軽減するには、どの手順を採用すべきですか?

A) オブジェクトの一覧を新しいバケットに自動的にプッシュするように Amazon S3 バケットのライフサイクルポリシーを更新し、この一覧を使用して、アプリケーションのバケットに関連付けられたオブジェクトを確認する。

B) 新しい DynamoDB テーブルを作成する。新しい DynamoDB テーブルを使用して、Amazon S3 にアップロードされたすべてのオブジェクトに関するすべてのメタデータを保存する。新しいオブジェクトがアップロードされるときは常に、DynamoDB にあるアプリケーションの内部的な Amazon S3 オブジェクトメタデータのキャッシュを更新する。

C) Amazon SNS を使用して、新しいオブジェクトに関するすべてのメタデータを保存するように新しい DynamoDB テーブルを自動的に更新する、新しい Amazon S3 オブジェクトに関する通知を作成する。アプリケーションを Amazon SNS トピックにサブスクライブし、DynamoDB テーブルにある内部的な Amazon S3 オブジェクトメタデータのキャッシュを更新する。

D) すべてのイメージを Amazon SQS にアップロードし、すべてのイメージを Amazon S3 に移動するように SQS ライフサイクルをセットアップして、アプリケーションに対する Amazon SNS 通知を開始してアプリケーションの内部的な Amazon S3 オブジェクトメタデータのキャッシュを更新する。

E) すべてのイメージを ElastiCache ファイルキャッシュサーバーにアップロードする。すべてのファイルメタデータを ElastiCache ファイルキャッシュサーバーから読み取るようにアプリケーションを更新し、すべてのファイルを長期保存用の Amazon S3 にプッシュするように ElastiCache ポリシーを設定する。

8）チームで、ある AWS Elastic Beanstalk アプリケーションを担当しています。ビジネス上の要件として、継続的なデプロイモデルに移行し、ダウンタイムなしで 1 日に何回もアプリケーションの更新をリリースする必要があります。

これを実現しながら、緊急時に以前のバージョンにほぼ即座にロールバックできるようにするには、どうすべきですか?

A) Elastic Beanstalk 環境でローリング更新を有効にし、アプリケーションの起動を考慮した適切な停止時間を設定する。

B) 新しいアプリケーションバージョンを実行する 2 つ目の Elastic Beanstalk 環境を作成し、これらの環境の CNAME を交換する。

C) コードリポジトリ内の新しいアプリケーションバージョンをポーリングし、実行中の各 Elastic Beanstalk インスタンスにダウンロードしてインストールするためのアプリケーションを開発する。

D) 新しいアプリケーションバージョンを使用する 2 つ目の Elastic Beanstalk 環境を作成し、HTTP 301 レスポンスコードで新しい環境にクライアントをリダイレクトするように既存の環境を設定する。

9）あなたの現在のログ分析用アプリケーションでは、ウェブアプリケーションの上位 10 名のユーザーに関するレポートを生成するまでに 4 時間以上かかります。あなたは、この情報をリアルタイムで報告できるシステムを導入し、レポートを常に最新の状態に維持し、ウェブアプリケーションのリクエスト数の増加に対応するよう求められています。

この要件を満たすことができる、費用対効果に優れた方法を選択してください。

A) データを CloudWatch ログにパブリッシュし、自動スケーリングを行って負荷にオンデマンドで対応するようにアプリケーションを設定する。

B) ログデータを Amazon S3 バケットにパブリッシュする。AWS CloudFormation を使用して Auto Scaling group を作成する。このグループによって、Amazon S3 に保存されているログファイルを取得するように設定されたポストプロセッシングアプリケーションをスケールする。

C) ログデータを Amazon Kinesis データストリームに公開し、ログデータを処理するように設定されたログ処理用アプリケーションをサブスクライブする。

D) Amazon EMR クラスターのサイズを増やすように Auto Scaling group を設定する。

E) マルチ AZ Amazon RDS MySQL クラスターを作成し、ログデータを MySQL に書き込んで、ユーザー数に関する必要な情報を取得するためのマップリデュースジョブを実行する。

１０）サードパーティのサプライヤへの注文を処理するあなたのアプリケーションは、単一の Amazon EC2 インスタンス上で実行されています。注文は Amazon SQS キューから取得され、5 分ごとに一括処理されます。ビジネス上の要件として、処理における遅延が 1 時間を超えてはなりません。週に約 3 回、アプリケーションで障害が発生して注文の処理が停止し、手動での再起動が必要になります。

このアプリケーションの障害耐性を、費用対効果に優れた方法で高めるには、どの手順を採用すべきですか?（2 つ選択してください）

A) 処理用インスタンスを監視し、障害が検出された場合にインスタンスを再起動するように設定された、2 つ目の「ウォッチドッグ」インスタンスを作成する。

B) 処理を実行するように設定されたインスタンスを起動するための Auto Scaling 起動設定を作成する。最小数および最大数として 1 が指定された起動設定を使用する、Auto Scaling group を作成する。

C) 処理を実行するように設定されたインスタンスを起動するための Auto Scaling 起動設定を作成する。最小数として 2 が、最大数として 10 が指定された起動設定を使用し、Amazon SQS キューのサイズに基づいてスケールする、Auto Scaling group を作成する。

D) ロードバランサーを作成し、インスタンスを Elastic Load Balancing に登録する。処理を実行するアプリケーションの HTTP エンドポイントを呼び出すように Elastic Load Balancing ヘルスチェックを設定する。

E) カスタム CloudWatch メトリックスと InstanceId ディメンションを送信するように処理用アプリケーションを変更する。メトリックスで Insufficient 状態が 10 分間続いた場合に、インスタンスを終了するための Amazon EC2 アクションを実行するように設定された CloudWatch アラームを作成する。