Answer Sheet

Q1) You are working as a DevOps engineer for a company that has hundreds of Amazon EC2 instances in their AWS account which runs their enterprise web applications. The company is using Slack, which is a cloud-based proprietary instant messaging platform, for updates and system alerts. There is a new requirement to create a system that should automatically send all AWS-scheduled maintenance notifications to the Slack channel of the company. This will easily notify their IT Operations team if there are any AWS-initiated changes to their EC2 instances and other resources.
Which of the following is the EASIEST method that you should implement in order to meet the above requirement?

- Use a combination of Amazon EC2 Events and Amazon CloudWatch to track and monitor the AWS-initiated activities to your resources. Create an alarm using CloudWatch Alarms which can invoke an AWS Lambda function to send notifications to the company's Slack channel.
- Use a combination of AWS Config and AWS Trusted Advisor to track any AWS-initiated changes. Create rules in AWS Config which can trigger an event that invokes an AWS Lambda function to send notifications to the company's Slack channel.
- Use a combination of AWS Support APIs and AWS CloudTrail to track and monitor the AWS-initiated activities to your resources. Create a new trail using AWS CloudTrail which can invoke an AWS Lambda function to send notifications to the company's Slack channel.
- ✅ Use a combination of AWS Personal Health Dashboard and Amazon CloudWatch Events to track the AWS-initiated activities to your resources. Create an event using CloudWatch Events which can invoke an AWS Lambda function to send notifications to the company's Slack channel.

Explanation:-AWS Health provides ongoing visibility into the state of your AWS resources, services, and accounts. The service gives you awareness and remediation guidance for resource performance or availability issues that affect your applications running on AWS. AWS Health provides relevant and timely information to help you manage events in progress. AWS Health also helps to be aware of and to prepare for planned activities. The service delivers alerts and notifications triggered by changes in the health of AWS resources, so that you get near-instant event visibility and guidance to help accelerate troubleshooting.
All customers can use the Personal Health Dashboard (PHD), powered by the AWS Health API. The dashboard requires no setup, and it's ready to use for authenticated AWS users. Additionally, AWS Support customers who have a Business or Enterprise support plan can use the AWS Health API to integrate with in-house and third-party systems.
You can use Amazon CloudWatch Events to detect and react to changes in the status of AWS Personal Health Dashboard (AWS Health) events. Then, based on the rules that you create, CloudWatch Events invokes one or more target actions when an event matches the values that you specify in a rule. Depending on the type of event, you can send notifications, capture event information, take corrective action, initiate events, or take other actions.
Only those AWS Health events that are specific to your AWS account and resources are published to CloudWatch Events. This includes events such as EBS volume lost, EC2 instance store drive performance degraded, and all the scheduled change events. In contrast, Service Health Dashboard events provide information about the regional availability of a service and are not specific to AWS accounts, so they are not published to CloudWatch Events. These event types have the word "operational" in the title in the Personal Health Dashboard; for example, "SWF operational issue".
Hence, the correct answer is: Use a combination of AWS Personal Health Dashboard and Amazon CloudWatch Events to track the AWS-initiated activities to your resources. Create an event using CloudWatch Events which can invoke an AWS Lambda function to send notifications to the company's Slack channel.
The option that says: Use a combination of AWS Config and AWS Trusted Advisor to track any AWS-initiated changes. Create rules in AWS Config which can trigger an event that invokes an AWS Lambda function to send notifications to the company's Slack channel is incorrect because AWS Config is not capable of tracking any AWS-initiated changes or maintenance in your AWS resources.
The option that says: Use a combination of Amazon EC2 Events and Amazon CloudWatch to track and monitor the AWS-initiated activities to your resources. Create an alarm using CloudWatch Alarms which can invoke an AWS Lambda function to send notifications to the company's Slack channel is incorrect because although the use of CloudWatch is acceptable, especially CloudWatch Events, it is still invalid to use Amazon EC2 Events to track AWS-initiated activities. A better combination should be the AWS Personal Health Dashboard and Amazon CloudWatch Events.
The option that says: Use a combination of AWS Support APIs and AWS CloudTrail to track and monitor the AWS-initiated activities to your resources. Create a new trail using AWS CloudTrail which can invoke an AWS Lambda function to send notifications to the company's Slack channel is incorrect because an integration of AWS Support API and AWS CloudTrail will not be able to properly monitor the AWS-initiated activities in your resources since CloudTrail simply tracks all of the API events of your account only but not the AWS-initiated events or maintenance.
References:
https://docs.aws.amazon.com/health/latest/ug/cloudwatch-events-health.html#automating-instance-actions
https://docs.aws.amazon.com/health/latest/ug/cloudwatch-events-health.html
https://docs.aws.amazon.com/health/latest/ug/what-is-aws-health.html

---

Q2) A company has a fleet of Linux and Windows servers which they use for their enterprise application. They need to implement an automated daily check of each golden AMI they own to monitor the latest Common Vulnerabilities and Exposures (CVE) using Amazon Inspector.
Which among the options below is the MOST suitable solution that they should implement?

- Use an AWS Lambda function to launch an Amazon EC2 instance for each operating system from the golden AMI and add a custom tag for tracking. Create another Lambda function that will call the Amazon Inspector API action StartAssessmentRun after the EC2 instances have booted up, which will run the assessment against all instances with the custom tag you added. Trigger the function every day using an Amazon CloudWatch Alarms.
- Use an AWS Lambda function to launch an Amazon EC2 instance for each operating system from the golden AMI and add a custom tag for tracking. Create another Lambda function that will trigger the Amazon Inspector assessment for all instances with the custom tag you added right after the EC2 instances have booted up. Trigger the function every day using an Amazon CloudWatch Events rule.
- Use AWS Step Functions to launch an Amazon EC2 instance for each operating system from the golden AMI and install the Amazon Inspector agent. Configure the Step Functions to trigger the Amazon Inspector assessment for all instances right after the EC2 instances have booted up. Configure the Step Functions to run every day using the CloudWatch Event Bus.
- ✅ Use AWS Step Functions to launch an Amazon EC2 instance for each operating system from the golden AMI, install the Amazon Inspector agent, and add a custom tag for tracking. Configure the Step Functions to trigger the Amazon Inspector assessment for all instances with the custom tag you added right after the EC2 instances have booted up. Trigger the Step Functions every day using an Amazon CloudWatch Events rule.

Explanation:-Amazon Inspector tests the network accessibility of your Amazon EC2 instances and the security state of your applications that run on those instances. Amazon Inspector assesses applications for exposure, vulnerabilities, and deviations from best practices. After performing an assessment, Amazon Inspector produces a detailed list of security findings that is organized by level of severity.
With Amazon Inspector, you can automate security vulnerability assessments throughout your development and deployment pipelines or for static production systems. This allows you to make security testing a regular part of development and IT operations.
Amazon Inspector also offers predefined software called an agent that you can optionally install in the operating system of the EC2 instances that you want to assess. The agent monitors the behavior of the EC2 instances, including network, file system, and process activity. It also collects a wide set of behavior and configuration data (telemetry).
If you want to set up a recurring schedule for your assessment, you can configure your assessment template to run automatically by creating a

Lambda function using the AWS Lambda console. Alternatively, you can select the "Set up recurring assessment runs once every , starting now" checkbox and specify the recurrence pattern (number of days) using the up and down arrows.
Hence, the correct answer is to use AWS Step Functions to launch an Amazon EC2 instance for each operating system from the golden AMI, install the Amazon Inspector agent, and add a custom tag for tracking. Configure the Step Functions to trigger the Amazon Inspector assessment for all instances with the custom tag you added right after the EC2 instances have booted up. Trigger the Step Functions every day using an Amazon CloudWatch Events rule.
The option that says: Use an AWS Lambda function to launch an Amazon EC2 instance for each operating system from the golden AMI and add a custom tag for tracking. Create another Lambda function that will call the Amazon Inspector API action StartAssessmentRun after the EC2 instances have booted up, which will run the assessment against all instances with the custom tag you added. Trigger the function every day using an Amazon CloudWatch Alarms is incorrect because you can't trigger a Lambda function on a regular basis using CloudWatch Alarms. You have to use CloudWatch Events instead. Moreover, you have to install the Amazon Inspector agent to the EC2 instance in order to run the security assessments.
The option that says: Use an AWS Lambda function to launch an Amazon EC2 instance for each operating system from the golden AMI and add a custom tag for tracking. Create another Lambda function that will trigger the Amazon Inspector assessment for all instances with the custom tag you added right after the EC2 instances have booted up. Trigger the function every day using an Amazon CloudWatch Events rule is incorrect because, in order for this solution to work, you have to install the Amazon Inspector agent first to the EC2 instance before you can run the security assessments.
The option that says: Use AWS Step Functions to launch an Amazon EC2 instance for each operating system from the golden AMI and install the Amazon Inspector agent. Configure the Step Functions to trigger the Amazon Inspector assessment for all instances right after the EC2 instances have booted up. Configure the Step Functions to run every day using the CloudWatch Event Bus is incorrect because the CloudWatch Event bus is primarily used to accept events from AWS services, other AWS accounts, and PutEvents API calls. You should also add a custom tag to the EC2 instance in order to run the Amazon Inspector assessments.
References:
https://aws.amazon.com/premiumsupport/knowledge-center/set-up-amazon-inspector/
https://docs.aws.amazon.com/inspector/latest/userguide/inspector_assessments.html#assessment_runs-schedule
https://aws.amazon.com/blogs/security/how-to-set-up-continuous-golden-ami-vulnerability-assessments-with-amazon-inspector/
Check out this Amazon Inspector Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-inspector/

Q3) A company has an Amazon ECS cluster which consists of multiple Amazon EC2 instances that runs a Docker-based application. The development team always pushes a new image to a private Docker container registry whenever they publish a new version of their application. They stop and start all of the tasks to ensure that the containers have the latest version of the application. However, the team noticed that some tasks are still running the old image of the application.
As the DevOps engineer, what should you do to fix this issue?

- Ensure that the latest tag is being used in the Docker image of the task definition.
- Migrate your repository from your private Docker container registry to Amazon ECR.
- Configure the task definition to use the repository-url/image@digest format then manually update the SHA-256 digest of the image.
- ✅ Restart the ECS agent.

Explanation:-You can update a running service to change the number of tasks that are maintained by a service, which task definition is used by the tasks, or if your tasks are using the Fargate launch type, you can change the platform version your service uses. If you have an application that needs more capacity, you can scale up your service. If you have unused capacity to scale down, you can reduce the number of desired tasks in your service and free up resources. If you have updated the Docker image of your application, you can create a new task definition with that image and deploy it to your service.
If your updated Docker image uses the same tag as what is in the existing task definition for your service (for example, my_image:latest), you do not need to create a new revision of your task definition. You can update the service using the procedure below, keep the current settings for your service, and select Force new deployment. The new tasks launched by the deployment pull the current image/tag combination from your repository when they start.
The service scheduler uses the minimum healthy percent and maximum percent parameters (in the deployment configuration for the service) to determine the deployment strategy. When a new task starts, the Amazon ECS container agent pulls the latest version of the specified image and tag for the container to use. However, subsequent updates to a repository image are not propagated to already running tasks.
To have your service use a newly updated Docker image with the same tag as in the existing task definition (for example, my_image:latest) or keep the current settings for your service, select Force new deployment. The new tasks launched by the deployment pull the current image/tag combination from your repository when they start. The Force new deployment option is also used when updating a Fargate task to use a more current platform version when you specify LATEST. For example, if you specified LATEST and your running tasks are using the 1.0.0 platform version and you want them to relaunch using a newer platform version.
If the ECS task is still running an old image then it is possible that the ECS agent is not running properly. Hence, the correct answer in this scenario is to restart the ECS agent so it can pull the latest Docker image.
Ensuring that the latest tag is being used in the Docker image of the task definition is incorrect because although this will release you from the burden of constantly updating your task definition, this is still not the solution for this issue. Remember that there are other tasks that were successfully updated which means that the image tag is not the root cause.
Configuring the task definition to use the repository-url/image@digest format then manually updating the SHA-256 digest of the image is incorrect because this will just explicitly fetch the exact Docker image from the registry based on the provided SHA-256 digest and not based on its tag (e.g. latest, 1.0.0).
Migrating your repository from your private Docker container registry to Amazon ECR is incorrect because the issue will be the same even if you used Amazon ECR if the ECS agent is not running properly in one of the instances.
References:
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/update-service.html
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task_definition_parameters.html
Check out this Amazon ECS Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-elastic-container-service-amazon-ecs/
Tutorials Dojo's AWS Certified DevOps Engineer Professional Exam Study Guide:
https://tutorialsdojo.com/aws-cheat-sheet-aws-certified-devops-engineer-professional/

Q4) A company has several legacy systems which use both on-premises servers as well as EC2 instances in AWS. The cluster nodes in AWS are configured to have a local IP address and a fixed hostname in order for the on-premises servers to communicate with them. There is a requirement to automate the configuration of the cluster which consists of 10 nodes to ensure high availability across three Availability Zones. There should also be a corresponding elastic network interface in a specific subnet for each Availability Zone. Each of the cluster node's local IP address and hostname must be static and should not change even if the instance reboots or gets terminated.
Which of the following solutions below provides the LEAST amount of effort to automate this architecture?

Launch an Elastic Beanstalk application with 10 EC2 instances, each has a corresponding ENI, hostname, and AZ as input parameters. Use the Elastic Beanstalk health agent daemon process to configure the hostname of the instance and attach a specific ENI based on the current environment.

Develop a custom AWS CLI script to launch the EC2 instances, each with an attached ENI, a unique name and placed in a specific AZ. Replace the missing EC2 instance by running the script again in the event that one of the instances in the cluster got rebooted or terminated.

Set up a CloudFormation child stack template which launches an Auto Scaling group consisting of just one EC2 instance then provide a list of ENIs, hostnames, and the specific AZs as stack parameters. Set both the MinSize and MaxSize parameters of the Auto Scaling group to 1. Add a user data script that will attach an ENI to the instance once launched. Use CloudFormation nested stacks to provision a total of 10 nodes needed for the cluster, and deploy the stack using a master template.

Explanation:-A stack is a collection of AWS resources that you can manage as a single unit. In other words, you can create, update, or delete a collection of resources by creating, updating, or deleting stacks. All the resources in a stack are defined by the stack's AWS CloudFormation template. A stack, for instance, can include all the resources required to run a web application, such as a web server, a database, and networking rules. If you no longer require that web application, you can simply delete the stack, and all of its related resources are deleted.

AWS CloudFormation ensures all stack resources are created or deleted as appropriate. Because AWS CloudFormation treats the stack resources as a single unit, they must all be created or deleted successfully for the stack to be created or deleted. If a resource cannot be created, AWS CloudFormation rolls the stack back and automatically deletes any resources that were created. If a resource cannot be deleted, any remaining resources are retained until the stack can be successfully deleted.

Nested stacks are stacks created as part of other stacks. You create a nested stack within another stack by using the AWS::CloudFormation::Stack resource.

As your infrastructure grows, common patterns can emerge in which you declare the same components in multiple templates. You can separate out these common components and create dedicated templates for them. Then use the resource in your template to reference other templates, creating nested stacks.

For example, assume that you have a load balancer configuration that you use for most of your stacks. Instead of copying and pasting the same configurations into your templates, you can create a dedicated template for the load balancer. Then, you just use the resource to reference that template from within other templates.

By setting up both the MinSize and MaxSize parameters of the Auto Scaling group to 1, you can ensure that your EC2 instance can recover again in the event of systems failure with exactly the same parameters defined in the CloudFormation template. This is one of the Auto Scaling strategies which provides high availability with the least possible cost. In this scenario, there is no mention about the scalability of the solution but only its availability.

Hence, the correct answer is to: Set up a CloudFormation child stack template which launches an Auto Scaling group consisting of just one EC2 instance then provide a list of ENIs, hostnames and the specific AZs as stack parameters. Set both the MinSize and MaxSize parameters of the Auto Scaling group to 1. Add a user data script that will attach an ENI to the instance once launched. Use CloudFormation nested stacks to provision a total of 10 nodes needed for the cluster, and deploy the stack using a master template.

The option that launches an Elastic Beanstalk application with 10 EC2 instances is incorrect because this involves a lot of manual configuration which will make it hard for you to replicate the same stack to another AWS region. Moreover, the Elastic Beanstalk health agent is primarily used to monitor the health of your instances and can't be used to configure the hostname of the instance nor attach a specific ENI.

The option that uses a DynamoDB table to store the list of ENIs and hostnames subnets is incorrect because you cannot maintain the assignment of the individual local IP address and hostname for each instances using Systems Manager.

The option that develops a custom AWS CLI script to launch the EC2 instances is incorrect because this is not an automated solution. A better way to meet this requirement is to use CloudFormation.

References:

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-nested-stacks.html

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stacks.html

Check out this AWS CloudFormation Cheat Sheet:

https://tutorialsdojo.com/aws-cheat-sheet-aws-cloudformation/

Use a DynamoDB table to store the list of ENIs and hostnames subnets which will be used by the cluster. Set up a single AWS CloudFormation template to manage an Auto Scaling group with a minimum and maximum size of 10. Maintain the assignment of each local IP address and hostname of the instances by using Systems Manager.

Q5) A leading IT consultancy firm has several Python-based Flask and Django web applications hosted in AWS. Some of their developers are freelance contractors located overseas. The firm wants to automate remediation actions for issues relating to the health of its AWS resources by using the AWS Health Dashboard and the AWS Health API. They need to automatically detect any of their own IAM access key that is accidentally or deliberately listed on a public Github repository. Once detected, the IAM access key must be immediately deleted and a notification should be sent to the DevOps team.
As a DevOps Engineer, how can you meet this requirement?

Set up three Lambda functions in AWS Step Functions that deletes the exposed IAM access key, summarizes the recent API activity for the exposed key using CloudTrail and sends a notification to the IT Security team using Amazon SNS. Create an AWS Personal Health Dashboard rule for the AWS_RISK_CREDENTIALS_EXPOSED event to monitor any exposed IAM keys from the Internet. Set the Step Functions as the target of the CloudWatch Events rule.

Set up three Lambda functions in AWS Step Functions that deletes the exposed IAM access key, summarizes the recent API activity for the exposed key using CloudTrail and sends a notification to the IT Security team using Amazon SNS. Use a combination of Amazon GuardDuty and Amazon Macie to monitor any exposed IAM keys from the Internet. Set the Step Functions as the target of the CloudWatch Events rule.

Set up three Lambda functions in AWS Step Functions that deletes the exposed IAM access key, summarizes the recent API activity for the exposed key using CloudTrail and sends a notification to the IT Security team using Amazon SNS. Create a CloudWatch Events rule with an aws.health event source and the AWS_RISK_CREDENTIALS_EXPOSED event to monitor any exposed IAM keys from the Internet. Set the Step Functions as the target of the CloudWatch Events rule.

Explanation:-AWS Step Functions lets you coordinate multiple AWS services into serverless workflows so you can build and update apps quickly. Using Step Functions, you can design and run workflows that stitch together services, such as AWS Lambda, AWS Fargate, and Amazon SageMaker, into feature-rich applications. Workflows are made up of a series of steps, with the output of one step acting as input into the next. Application development is simpler and more intuitive using Step Functions, because it translates your workflow into a state machine diagram that is easy to understand, easy to explain to others, and easy to change.

AWS proactively monitors popular code repository sites for exposed AWS Identity and Access Management (IAM) access keys. On detection of an exposed IAM access key, AWS Health generates an AWS_RISK_CREDENTIALS_EXPOSED CloudWatch Event. In response to this event, you can set up an automated workflow deletes the exposed IAM Access Key, summarizes the recent API activity for the exposed key, and sends the summary message to an Amazon Simple Notification Service (SNS) Topic to notify the subscribers which are all orchestrated by an AWS Step Functions state machine.

You can use Amazon CloudWatch Events to detect and react to changes in the status of AWS Personal Health Dashboard (AWS Health) events. Then, based on the rules that you create, CloudWatch Events invokes one or more target actions when an event matches the values that you

specify in a rule. Depending on the type of event, you can send notifications, capture event information, take corrective action, initiate events, or take other actions.
You can automate actions in response to new scheduled events for your EC2 instances. For example, you can create CloudWatch Events rules for EC2 scheduled events generated by the AWS Health service. These rules can then trigger targets, such as AWS Systems Manager Automation documents, to automate actions.
Hence, the correct answer is: Set up three Lambda functions in AWS Step Functions that deletes the exposed IAM access key, summarizes the recent API activity for the exposed key using CloudTrail and sends notification to the IT Security team using Amazon SNS. Create a CloudWatch Events rule with an aws.health event source and the AWS_RISK_CREDENTIALS_EXPOSED event to monitor any exposed IAM keys from the Internet. Set the Step Functions as the target of the CloudWatch Events rule.
The option that says: Set up three Lambda functions in AWS Step Functions that deletes the exposed IAM access key, summarizes the recent API activity for the exposed key using CloudTrail and sends a notification to the IT Security team using Amazon SNS. Use a combination of Amazon GuardDuty and Amazon Macie to monitor any exposed IAM keys from the Internet. Set the Step Functions as the target of the CloudWatch Events rule is incorrect because you can't monitor any exposed IAM keys from the Internet using Amazon GuardDuty and Amazon Macie. Amazon GuardDuty is a threat detection service that continuously monitors for malicious activity and unauthorized behavior to protect your AWS accounts and workloads while Amazon Macie is simply a security service that uses machine learning to automatically discover, classify, and protect sensitive data in AWS.
The option that says: Set up three Lambda functions in AWS Step Functions that deletes the exposed IAM access key, summarizes the recent API activity for the exposed key using CloudTrail and sends a notification to the IT Security team using Amazon SNS. Create an AWS Config rule for the AWS_RISK_CREDENTIALS_EXPOSED event with Multi-Account Multi-Region Data Aggregation to monitor any exposed IAM keys from the Internet. Set the Step Functions as the target of the CloudWatch Events rule is incorrect because the use of Multi-Account Multi-Region Data Aggregation in CloudTrail will not satisfy the requirement. An aggregator is simply an AWS Config resource type that collects AWS Config configuration and compliance data from multiple accounts across multiple regions.
The option that says: Set up three Lambda functions in AWS Step Functions that deletes the exposed IAM access key, summarizes the recent API activity for the exposed key using CloudTrail and sends a notification to the IT Security team using Amazon SNS. Create an AWS Personal Health Dashboard rule for the AWS_RISK_CREDENTIALS_EXPOSED event to monitor any exposed IAM keys from the Internet. Set the Step Functions as the target of the CloudWatch Events rule is incorrect because you have to use the AWS Health API instead of the AWS Personal Health Dashboard. The AWS_RISK_CREDENTIALS_EXPOSED event is only applicable from an aws.health event source and not from an AWS Personal Health Dashboard rule.
References:
https://aws.amazon.com/blogs/compute/automate-your-it-operations-using-aws-step-functions-and-amazon-cloudwatch-events/
https://github.com/aws/aws-health-tools/tree/master/automated-actions/AWS_RISK_CREDENTIALS_EXPOSED
https://docs.aws.amazon.com/health/latest/ug/cloudwatch-events-health.html
Check out these AWS Health and Amazon CloudWatch Cheat Sheets:
https://tutorialsdojo.com/aws-health/
https://tutorialsdojo.com/amazon-cloudwatch/
● Set up three Lambda functions in AWS Step Functions that deletes the exposed IAM access key, summarizes the recent API activity for the exposed key using CloudTrail and sends a notification to the IT Security team using Amazon SNS. Create an AWS Config rule for the AWS_RISK_CREDENTIALS_EXPOSED event with Multi-Account Multi-Region Data Aggregation to monitor any exposed IAM keys from the Internet. Set the Step Functions as the target of the CloudWatch Events rule.

Q6) An American tech company used an AWS CloudFormation template to deploy its static corporate website hosted on Amazon S3 in the US East (N. Virginia) region. The template defines an Amazon S3 bucket with a Lambda-backed custom resource that downloads the content from a file server into the bucket. There is a new task for the DevOps Engineer to move the website to the US West (Oregon) region to better serve its customers on the West Coast with lower latency. However, the application stack could not be deleted successfully in CloudFormation.
Which among the following options shows the root cause of this issue, and how can the DevOps Engineer mitigate this problem for current and future versions of the website?

● The CloudFormation stack deletion fails for an S3 bucket because the DeletionPolicy attribute is set to Snapshot. To fix the issue, set the DeletionPolicy to Delete instead.
● The CloudFormation stack deletion fails for an S3 bucket because it is not yet empty. To fix the issue, set the DeletionPolicy to ForceDelete instead.
✔ The CloudFormation stack deletion fails for an S3 bucket that still has contents. To fix the issue, modify the Lambda function code of the custom resource to recursively empty the bucket if the stack is selected for deletion.
Explanation:-When you associate a Lambda function with a custom resource, the function is invoked whenever the custom resource is created, updated, or deleted. AWS CloudFormation calls a Lambda API to invoke the function and to pass all the request data (such as the request type and resource properties) to the function. The power and customizability of Lambda functions in combination with AWS CloudFormation enable a wide range of scenarios, such as dynamically looking up AMI IDs during stack creation, or implementing and using utility functions, such as string reversal functions.
AWS CloudFormation templates that declare an Amazon Elastic Compute Cloud (Amazon EC2) instance must also specify an Amazon Machine Image (AMI) ID, which includes an operating system and other software and configuration information used to launch the instance. The correct AMI ID depends on the instance type and region in which you're launching your stack. And IDs can change regularly, such as when an AMI is updated with software updates.
Normally, you might map AMI IDs to specific instance types and regions. To update the IDs, you manually change them in each of your templates. By using custom resources and AWS Lambda (Lambda), you can create a function that gets the IDs of the latest AMIs for the region and instance type that you're using so that you don't have to maintain mappings.
You can also run the custom resource to recursively empty the bucket when the CloudFormation stack is triggered for deletion. In CloudFormation, you can only delete empty buckets. Any request for deletion will fail for buckets that still have contents. To control how AWS CloudFormation handles the bucket when the stack is deleted, you can set a deletion policy for your bucket. You can choose to retain the bucket or to delete the bucket.
Hence, the correct answer is: The CloudFormation stack deletion fails for an S3 bucket that still has contents. To fix the issue, modify the Lambda function code of the custom resource to recursively empty the bucket if the stack is selected for deletion.
The option that says: The CloudFormation stack deletion fails for an S3 bucket that is used as a static web hosting. To fix the issue, modify the CloudFormation template to remove the website configuration for the S3 bucket is incorrect because the CloudFormation deletion process will not be hindered simply because your S3 bucket is configured for static web hosting. The primary root cause of this issue is that the CloudFormation stack deletion fails for an S3 bucket that still has contents.
The option that says: The CloudFormation stack deletion fails for an S3 bucket because the DeletionPolicy attribute is set to Snapshot. To fix the issue, set the DeletionPolicy to Delete instead is incorrect because you can only set the DeletionPolicy to either Retain or Delete for an Amazon S3 resource. In addition, the CloudFormation deletion will still fail as long as the S3 bucket is not empty, even if the DeletionPolicy attribute is already set to Delete.
The option that says: The CloudFormation stack deletion fails for an S3 bucket is not yet empty. To fix the issue, set the DeletionPolicy to

ForceDelete instead is incorrect because although the provided root cause is accurate, the configuration for DeletionPolicy remains invalid. ForceDelete is not a valid value for the deletion policy attribute.
References:
https://aws.amazon.com/blogs/devops/faster-auto-scaling-in-aws-cloudformation-stacks-with-lambda-backed-custom-resources/
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/walkthrough-custom-resources-lambda-lookup-amiids.html
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-custom-resources-lambda.html
Check out this AWS CloudFormation Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-cloudformation/
- The CloudFormation stack deletion fails for an S3 bucket that is used as a static web hosting. To fix the issue, modify the CloudFormation template to remove the website configuration for the S3 bucket.

Q7) A fast-growing company has multiple AWS accounts which are consolidated using AWS Organizations and they expect to add new accounts soon. As the DevOps engineer, you were instructed to design a centralized logging solution to deliver all of their VPC Flow Logs and CloudWatch Logs across all of their sub-accounts to their dedicated Audit account for compliance purposes. The logs should also be properly indexed in order to perform search, retrieval, and analysis.
Which of the following is the MOST suitable solution that you should implement to meet the above requirements?

- In the Audit account, launch a new Lambda function which will push all of the required logs to a self-hosted ElasticSearch cluster in a large EC2 instance. Integrate the Lambda function to a CloudWatch subscription filter to collect all of the logs from the sub-accounts and stream them to the Lambda function deployed in the Audit account.
- In the Audit account, launch a new Lambda function which will send all VPC Flow Logs and CloudWatch Logs to an Amazon ES cluster. Use a CloudWatch subscription filter in the sub-accounts to stream all of the logs to the Lambda function in the Audit account.
- In the Audit account, create an Amazon SQS queue that will push all logs to an Amazon ES cluster. Use a CloudWatch subscription filter to stream both VPC Flow Logs and CloudWatch Logs from their sub-accounts to the SQS queue in the Audit account.
- ✅ In the Audit account, create a new stream in Kinesis Data Streams and a Lambda function that acts as an event handler to send all of the logs to the Amazon ES cluster. Create a CloudWatch subscription filter and use Kinesis Data Streams to stream all of the VPC Flow Logs and CloudWatch Logs from the sub-accounts to the Kinesis data stream in the Audit account.

Explanation:-You can load streaming data into your Amazon Elasticsearch Service domain from many different sourcesin AWS. Some sources, like Amazon Kinesis Data Firehose and Amazon CloudWatch Logs, have built-in support for Amazon ES. Others, like Amazon S3, Amazon Kinesis Data Streams, and Amazon DynamoDB, use AWS Lambda functions as event handlers. The Lambda functions respond to new data by processing it and streaming it to your domain.
You can use subscriptions to get access to a real-time feed of log events from CloudWatch Logs and have it delivered to other services such as a Amazon Kinesis stream, Amazon Kinesis Data Firehose stream, or AWS Lambda for custom processing, analysis, or loading to other systems. To begin subscribing to log events, create the receiving source, such as a Kinesis stream, where the events will be delivered. A subscription filter defines the filter pattern to use for filtering which log events get delivered to your AWS resource, as well as information about where to send matching log events to.
You can collaborate with an owner of a different AWS account and receive their log events on your AWS resources, such as a Amazon Kinesis stream (this is known as cross-account data sharing). For example, this log event data can be read from a centralized Amazon Kinesis stream to perform custom processing and analysis. Custom processing is especially useful when you collaborate and analyze data across many accounts. For example, a company's information security group might want to analyze data for real-time intrusion detection or anomalous behaviors so it could conduct an audit of accounts in all divisions in the company by collecting their federated production logs for central processing.
A real-time stream of event data across those accounts can be assembled and delivered to the information security groups who can use Kinesis to attach the data to their existing security analytic systems. Kinesis streams are currently the only resource supported as a destination for cross-account subscriptions.
Hence, the correct solution is: In the Audit account, create a new stream in Kinesis Data Streams and a Lambda function that acts as an event handler to send all of the logs to the Amazon ES cluster. Create a CloudWatch subscription filter and use Kinesis Data Streams to stream all of the VPC Flow Logs and CloudWatch Logs from the sub-accounts to the Kinesis data stream in the Audit account.
The option that that says: In the Audit account, launch a new Lambda function which will send all VPC Flow Logs and CloudWatch Logs to an Amazon ES cluster. Use a CloudWatch subscription filter in the sub-accounts to stream all of the logs to the Lambda function in the Audit account is incorrect because launching a Kinesis data stream is a more suitable option than just a Lambda function that will accept the logs from other accounts and send them to Amazon ES.
The option that that says: In the Audit account, create an Amazon SQS queue that will push all logs to an Amazon ES cluster. Use a CloudWatch subscription filter to stream both VPC Flow Logs and CloudWatch Logs from their sub-accounts to the SQS queue in the Audit account is incorrect because the CloudWatch subscription filter doesn't directly support SQS. You should use a Kinesis Data Stream, Kinesis Firehose or Lambda function.
The option that that says: In the Audit account, launch a new Lambda function which will push all of the required logs to a self-hosted ElasticSearch cluster in a large EC2 instance. Integrate the Lambda function to a CloudWatch subscription filter to collect all of the logs from the sub-accounts and stream them to the Lambda function deployed in the Audit account is incorrect because it is better to use Amazon ES instead of launching a self-hosted ElasticSearch cluster. Launching a Kinesis data stream is a more suitable option than just a Lambda function that will accept the logs from other accounts and send them to Amazon ES.
References:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs//CrossAccountSubscriptions.html
https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/es-aws-integrations.html
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs//SubscriptionFilters.html#FirehoseExample
Check out this Amazon CloudWatch Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudwatch/

Q8) An insurance firm has its on-premises data center integrated with its VPC in AWS Cloud via a Direct Connect connection. All of their financial records are stored on-premises and then uploaded to an Amazon S3 bucket for backup purposes. There is a new mandate by the IT Security team that only a trusted group of administrators can change the objects in the bucket including the object permissions. The team instructed a DevOps Engineer to develop a solution that provides near real-time, automated checks to monitor their data.
How should the Engineer satisfy this requirement?

- Use Amazon CloudWatch to monitor any changes to the Amazon S3 object-level and bucket-level permissions. Create a custom metric that checks if the user who initiated the change is an administrator.
- Create a new AWS Config rule with a Periodic trigger type that queries the Amazon S3 Server Access Logs for changes on a regular basis. Configure the rule to check if the user who initiated the change is an administrator.
- Create a new AWS Config rule with a Configuration changes trigger type that tracks any changes to the Amazon S3 bucket-level permissions. Modify the rule to use Step Functions to check if the user who initiated the change is an administrator.

✅ Set up an Amazon S3 data events in CloudTrail to track any object changes. Create a rule to run your Lambda function in response to an Amazon S3 data event that checks if the user who initiated the change is an administrator.

Explanation:-When an event occurs in your account, CloudTrail evaluates whether the event matches the settings for your trails. Only events that match your trail settings are delivered to your Amazon S3 bucket and Amazon CloudWatch Logs log group.

You can configure your trails to log the following:

Data events: These events provide insight into the resource operations performed on or within a resource. These are also known as data plane operations.

Management events: Management events provide insight into management operations that are performed on resources in your AWS account. These are also known as control plane operations. Management events can also include non-API events that occur in your account. For example, when a user logs in to your account, CloudTrail logs the ConsoleLogin event.

You can configure multiple trails differently so that the trails process and log only the events that you specify. For example, one trail can log read-only data and management events, so that all read-only events are delivered to one S3 bucket. Another trail can log only write-only data and management events, so that all write-only events are delivered to a separate S3 bucket.

You can also configure your trails to have one trail log and deliver all management events to one S3 bucket, and configure another trail to log and deliver all data events to another S3 bucket.

Data events provide insight into the resource operations performed on or within a resource. These are also known as data plane operations. Data events are often high-volume activities.

Example data events include:

Amazon S3 object-level API activity (for example, GetObject, DeleteObject, and PutObject API operations)

AWS Lambda function execution activity (the Invoke API)

Data events are disabled by default when you create a trail. To record CloudTrail data events, you must explicitly add the supported resources or resource types for which you want to collect activity to a trail.

Hence, the correct answer is: Set up an Amazon S3 data events in CloudTrail to track any object changes. Create a rule to run your Lambda function in response to an Amazon S3 data event that checks if the user who initiated the change is an administrator.

The option that says: Create a new AWS Config rule with a Periodic trigger type that queries the Amazon S3 Server Access Logs for changes on a regular basis. Configure the rule to check if the user who initiated the change is an administrator is incorrect because the Amazon Server access logging only provides detailed records for the requests that are made to the S3 bucket only and not the IAM Policy changes. Moreover, the AWS Config rule will only run on a specific schedule which means that it won't provide near real-time monitoring in comparison to Amazon S3 data events.

The option that says: Use Amazon CloudWatch to monitor any changes to the Amazon S3 object-level and bucket-level permissions. Create a custom metric that checks if the user who initiated the change is an administrator is incorrect because CloudWatch is not a suitable service to use in monitoring your configuration changes. Moreover, you can't directly create a custom metric that automatically checks if the user, who initiated the change, is indeed an administrator.

The option that says: Create a new AWS Config rule with a Configuration changes trigger type that tracks any changes to the Amazon S3 bucket-level permissions. Modify the rule to use Step Functions check if the user who initiated the change is an administrator is incorrect because you can't directly use Step Functions in conjunction with AWS Config. Although this type of trigger will cause the AWS Config to run evaluations when there is any change in Amazon S3, the use of Amazon S3 Data Event is still a more preferred way to track the changes in near real-time.

References:

https://aws.amazon.com/blogs/aws/cloudtrail-update-capture-and-process-amazon-s3-object-level-api-activity/

https://docs.aws.amazon.com/awscloudtrail/latest/userguide/logging-management-and-data-events-with-cloudtrail.html#example-logging-all-S3-objects

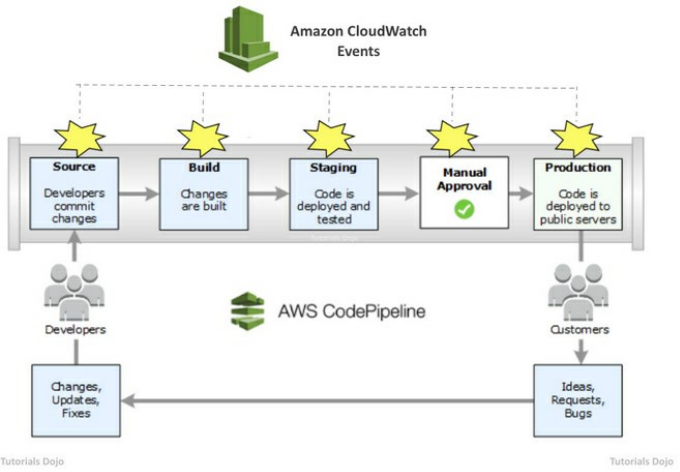
https://docs.aws.amazon.com/awscloudtrail/latest/userguide/logging-data-events-with-cloudtrail.html

Check out this AWS CloudTrail Cheat Sheet:

https://tutorialsdojo.com/aws-cheat-sheet-aws-cloudtrail/

Q9)

A startup recently hired you as a replacement for their DevOps engineer who abruptly resigned from his position. Since there was no time for him to hand over his tasks to anyone, you have to start from scratch and understand the current configurations he made in the client's AWS account. You saw that there is an existing CloudWatch Events rule with a custom event pattern as shown below:



What does this CloudWatch Events rule do?

- It will capture all rejected or failed build actions across all the pipelines in AWS CodePipeline and send a notification.
- It will capture all manual approval actions across all the pipelines in AWS CodePipeline and send a notification.
- ✅ It will capture all rejected or failed approval actions across all the pipelines in AWS CodePipeline and send a notification.

Explanation:-Monitoring is an important part of maintaining the reliability, availability, and performance of AWS CodePipeline. You should collect monitoring data from all parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs.

You can use the following tools to monitor your CodePipeline pipelines and their resources:

Amazon CloudWatch Events — Use Amazon CloudWatch Events to detect and react to pipeline execution state changes (for example, send an Amazon SNS notification or invoke a Lambda function).

AWS CloudTrail — Use CloudTrail to capture API calls made by or on behalf of CodePipeline in your AWS account and deliver the log files to an Amazon S3 bucket. You can choose to have CloudWatch publish Amazon SNS notifications when new log files are delivered so you can take quick action.
Console and CLI — You can use the CodePipeline console and CLI to view details about the status of a pipeline or a particular pipeline execution.
Amazon CloudWatch Events is a web service that monitors your AWS resources and the applications you run on AWS. You can use Amazon CloudWatch Events to detect and react to changes in the state of a pipeline, stage, or action. Then, based on rules you create, CloudWatch Events invokes one or more target actions when a pipeline, stage, or action enters the state you specify in a rule. Depending on the type of state change, you might want to send notifications, capture state information, take corrective action, initiate events, or take other actions.
You can configure notifications to be sent when the state changes for:
- Specified pipelines or all your pipelines. You control this by using "detail-type": "CodePipeline Pipeline Execution State Change".
- Specified stages or all your stages, within a specified pipeline or all your pipelines. You control this by using "detail-type": "CodePipeline Stage Execution State Change".
- Specified actions or all actions, within a specified stage or all stages, within a specified pipeline or all your pipelines. You control this by using "detail-type": "CodePipeline Action Execution State Change".
Hence, the correct answer is: It will capture all rejected or failed approval actions across all the pipelines in AWS CodePipeline and send a notification.
The option that says: It will capture all manual approval actions across all the pipelines in AWS CodePipeline and send a notification is incorrect because this will just capture all rejected or failed approval actions excluding the successful ones.
The option that says: It will capture all pipelines with a FAILED state in AWS CodePipeline and send a notification is incorrect because the custom event pattern is tracking the changes of the specific "Actions" and not the entire "State" of the pipeline.
The option that says: It will capture all rejected or failed build actions across all the pipelines in AWS CodePipeline and send a notification is incorrect because the indicated source is aws.codepipeline and not aws.codebuild which means that this rule tracks the failed or rejected approval actions across all the pipelines, and not the status of the build actions.
References:
https://docs.aws.amazon.com/codepipeline/latest/userguide/detect-state-changes-cloudwatch-events.html
https://docs.aws.amazon.com/codepipeline/latest/userguide/monitoring.html
○ It will capture all pipelines with a FAILED state in AWS CodePipeline and send a notification.

Q10) A company is developing a serverless application that uses AWS Lambda, AWS SAM, and Amazon API Gateway. There is a requirement to fully automate the backend Lambda deployment in such a way that the deployment will automatically run whenever a new commit is pushed to an AWS CodeCommit repository. There should also be a separate environment pipeline for TEST and PROD environments. In addition, the TEST environment should be the only one that allows automatic deployment. As a DevOps Engineer, who can you satisfy these requirements?

○ Set up two pipelines using AWS CodePipeline for TEST and PROD environments. On both pipelines, set up a manual approval step for application deployment. Set up separate CodeCommit repositories for each environment and configure each pipeline to retrieve the source code from CodeCommit. Deploy the Lambda functions with AWS CloudFormation by setting up a deployment step in the pipeline.
○ Create a new pipeline using AWS CodePipeline and a new CodeCommit repository for each environment. Configure CodePipeline to retrieve the application source code from the appropriate repository. Deploy the Lambda functions with AWS CloudFormation by creating a deployment step in the pipeline.
✓ Set up two pipelines using AWS CodePipeline for TEST and PROD environments. Configure the PROD pipeline to have a manual approval step. Create a CodeCommit repository with a branch for each environment and configure the pipeline to retrieve the source code from CodeCommit according to its branch. Deploy the Lambda functions with AWS CloudFormation by setting up a deployment step in the pipeline.
Explanation:-In AWS CodePipeline, you can add an approval action to a stage in a pipeline at the point where you want the pipeline execution to stop so that someone with the required AWS Identity and Access Management permissions can approve or reject the action.
If the action is approved, the pipeline execution resumes. If the action is rejected—or if no one approves or rejects the action within seven days of the pipeline reaching the action and stopping—the result is the same as an action failing, and the pipeline execution does not continue.
You might use manual approvals for these reasons:
- You want someone to perform a code review or change management review before a revision is allowed into the next stage of a pipeline.
- You want someone to perform manual quality assurance testing on the latest version of an application, or to confirm the integrity of a build artifact, before it is released.
- You want someone to review new or updated text before it is published to a company website.
Hence, the correct answer is: Set up two pipelines using AWS CodePipeline for TEST and PROD environments. Configure the PROD pipeline to have a manual approval step. Create a CodeCommit repository with a branch for each environment and configure the pipeline to retrieve the source code from CodeCommit according to its branch. Deploy the Lambda functions with AWS CloudFormation by setting up a deployment step in the pipeline.
The option that says: Create a new pipeline using AWS CodePipeline and a new CodeCommit repository for each environment. Configure CodePipeline to retrieve the application source code from the appropriate repository. Deploy the Lambda functions with AWS CloudFormation by creating a deployment step in the pipeline is incorrect because you should add a manual approval step on the PROD pipeline as mentioned in the requirements of the scenario.
The option that says: Set up two pipelines using AWS CodePipeline for TEST and PROD environments. In the PROD pipeline, set up a manual approval step for application deployment. Set up separate CodeCommit repositories for each environment and configure each pipeline to retrieve the source code from CodeCommit. Deploy the Lambda functions with AWS CloudFormation by setting up a deployment step in the pipeline is incorrect because you don't need to create separate CodeCommit repositories for the two environments. You just need to create two different branches from a single repository.
The option that says: Set up two pipelines using AWS CodePipeline for TEST and PROD environments. On both pipelines, set up a manual approval step for application deployment. Set up separate CodeCommit repositories for each environment and configure each pipeline to retrieve the source code from CodeCommit. Deploy the Lambda functions with AWS CloudFormation by setting up a deployment step in the pipeline is incorrect because you should add the manual approval step on the PROD pipeline only, excluding the TEST pipeline. Moreover, you don't need to create separate CodeCommit repositories for the two environments. You just need to create two different branches from a single repository.
References:
https://docs.aws.amazon.com/codepipeline/latest/userguide/approvals.html
https://docs.aws.amazon.com/codepipeline/latest/userguide/approvals-action-add.html
https://docs.aws.amazon.com/codepipeline/latest/userguide/welcome.html
Check out this AWS CodePipeline Cheat Sheet:
https://tutorialsdojo.com/aws-codepipeline/
Tutorials Dojo's AWS Certified DevOps Engineer Professional Exam Study Guide:
https://tutorialsdojo.com/aws-certified-devops-engineer-professional/
○ Set up two pipelines using AWS CodePipeline for TEST and PROD environments. In the PROD pipeline, set up a manual approval step for application deployment. Set up separate CodeCommit repositories for each environment and configure each pipeline to retrieve the source code

from CodeCommit. Deploy the Lambda functions with AWS CloudFormation by setting up a deployment step in the pipeline.

Q11) A local e-commerce website is gaining an unprecedented number of users in the company's home country to the point that people from other countries are requesting access to their site. In order to support the international growth of their flagship application, the company needs to add new features in their website to support shipping, value-added tax (VAT) calculations and other specific requirements for each new country. At the same time, they also have a set of new features that need to be developed for their e-commerce site, which is only specific to the existing local users. Each feature may take about 3 months to complete all the required planning, development, and testing stages.
As the DevOps Engineer, how should you properly manage the application feature deployments in the MOST efficient manner for this scenario?

○ Create a new repository for each new application feature in AWS CodeCommit and then commit all of the code changes to the respective repositories.
○ In AWS CodeCommit, instruct the developers to commit the new code for the new features in the master branch. Delay all other application deployment related to international expansion until all features are ready for their local users. Implement feature flags in your application to enable or disable specific features.
✓ In the application code repository in AWS CodeCommit, create a feature branch for each application feature that will be added. Once the feature is tested, merge the commits to the master or release branch.
Explanation:-In Git, branches are simply pointers or references to a commit. In development, they're a convenient way to organize your work. You can use branches to separate work on a new or different version of files without impacting work in other branches. You can use branches to develop new features, store a specific version of your project from a particular commit, and more.
In CodeCommit, you can change the default branch for your repository. This default branch is the one used as the base or default branch in local repos when users clone the repository. You can also create and delete branches and view details about a branch. You can quickly compare differences between a branch and the default branch (or any two branches). To view the history of branches and merges in your repository, you can use the Commit Visualizer.
In Git, one of its advantages is its branching capabilities which provide an isolated environment for every change to your codebase that doesn't affect your master or release branch. Git branches are cheap and easy to merge which is quite opposite with centralized version control systems. When you want to start working on a new feature or application functionality, you can easily create a new branch to ensure that the master branch always contains a production-quality code. This also lets you represent development work at the same granularity as your agile backlog.
Hence, the correct answer is: In the application code repository in AWS CodeCommit, create a feature branch for each application feature that will be added. Once the feature is tested, merge the commits to the master or release branch.
The option that says: In AWS CodeCommit, instruct the developers to commit the new code for the new features in the master branch. Delay all other application deployment related to international expansion until all features are ready for their local users is incorrect because delaying the application deployments is an inefficient process that could even stagnate the company's expansion. Although you can commit the new code for the features in the master branch and enable/disable them using feature flags, this development process is still inefficient and actually riskier since all of the changes are in a single branch. If there is a development issue in one of the features, your production code will be directly affected. A better solution for this is to create a separate feature branch for each new feature.
The option that says: Create a Git tag in CodeCommit to mark each commit with a label for each corresponding application feature is incorrect because these are just mere labels for your references. This process may affect the master branch which always contains the production-quality code. If there is a bug in one of the features, you will have to laboriously debug and trace the code changes on a single branch instead of a specific feature branch.
The option that says: Create a new repository for each new application feature in AWS CodeCommit and then commit all of the code changes to the respective repositories is incorrect because this is not appropriate at all since you can simply create new branches for your features, and not new repositories. A single application should be hosted in one repository which can be branched off for your feature development, testing, and application release.
References:
https://docs.aws.amazon.com/codecommit/latest/userguide/branches.html
https://docs.aws.amazon.com/codecommit/latest/userguide/welcome.html
Check out this AWS CodeCommit Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-codecommit/
Tutorials Dojo's AWS Certified DevOps Engineer Professional Exam Study Guide:
https://tutorialsdojo.com/aws-cheat-sheet-aws-certified-devops-engineer-professional/
○ Create a Git tag in CodeCommit to mark each commit with a label for each corresponding application feature.

Q12) A software development company has a microservices architecture that consists of several AWS Lambda functions with a DynamoDB table as its data store. The current workflow of the development team is to manually deploy the new version of the Lambda function right after the QA team completed the testing. There is a new requirement to improve the workflow by automating the tests as well as the code deployments. Whenever there is a new release, the application traffic to the new versions of each microservice should be incrementally shifted over time after deployment. This will provide them the option to verify the changes to a subset of users in production and easily rollback the changes if needed.
Which of the following solutions will improve the velocity of the company's development workflow?

○ Develop a custom shell script that utilizes a post-commit hook to upload the latest version of the Lambda function in an S3 bucket. Set up the S3 event trigger which will invoke a Lambda function that deploys the new version. Specify the percentage of traffic that will initially be routed to your updated Lambda as well as the interval to deploy the code over time.
○ Set up a new pipeline in AWS CodePipeline and configure a post-commit hook to start the pipeline after all the automated tests have passed. Configure AWS CodeDeploy to use an All-at-once deployment configuration for deployments.
✓ Set up a new pipeline in AWS CodePipeline and configure a new source code step that will automatically trigger whenever a new code is pushed in AWS CodeCommit. Use AWS CodeBuild for the build step to run the tests automatically then set up an AWS CodeDeploy configuration to deploy the updated Lambda function. Select the predefined CodeDeployDefault.LambdaLinear10PercentEvery3Minutes configuration option for deployment.
Explanation:-AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy. With CodeBuild, you don’t need to provision, manage, and scale your own build servers. CodeBuild scales continuously and processes multiple builds concurrently, so your builds are not left waiting in a queue.
AWS CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software. You can quickly model and configure the different stages of a software release process. CodePipeline automates the steps required to release your software changes continuously.
When you deploy to an AWS Lambda compute platform, the deployment configuration specifies the way traffic is shifted to the new Lambda function versions in your application.
There are three ways traffic can shift during a deployment:

Canary: Traffic is shifted in two increments. You can choose from predefined canary options that specify the percentage of traffic shifted to your updated Lambda function version in the first increment and the interval, in minutes, before the remaining traffic is shifted in the second increment.
Linear: Traffic is shifted in equal increments with an equal number of minutes between each increment. You can choose from predefined linear options that specify the percentage of traffic shifted in each increment and the number of minutes between each increment.
All-at-once: All traffic is shifted from the original Lambda function to the updated Lambda function version all at once.
Hence, the correct answer is Set up a new pipeline in AWS CodePipeline and configure a new source code step that will automatically trigger whenever a new code is pushed in AWS CodeCommit. Use AWS CodeBuild for the build step to run the tests automatically then set up an AWS CodeDeploy configuration to deploy the updated Lambda function. Select the predefined CodeDeployDefault.LambdaLinear10PercentEvery3Minutes configuration option for deployment.
The option that says: Set up a new pipeline in AWS CodePipeline and configure a post-commit hook to start the pipeline after all the automated tests have passed. Configure AWS CodeDeploy to use an All-at-once deployment configuration for deployments is incorrect because the All-at-once configuration will cause all traffic to be shifted from the original Lambda function to the updated Lambda function version all at once, just as what its name implies.
The option that says: Set up an AWS CodeBuild configuration which automatically starts whenever a new code is pushed. Configure CloudFormation to trigger a pipeline in AWS CodePipeline that deploys the new Lambda version. Specify the percentage of traffic that will initially be routed to your updated Lambda function as well as the interval to deploy the code over time in the CloudFormation template is incorrect because you can just use CodeDeploy for deployments to streamline the workflow instead of using a combination of CodePipeline and CloudFormation.
The option that says: Develop a custom shell script that utilizes a post-commit hook to upload the latest version of the Lambda function in an S3 bucket. Set up the S3 event trigger which will invoke a Lambda function that deploys the new version. Specify the percentage of traffic that will initially be routed to your updated Lambda as well as the interval to deploy the code over time is incorrect because developing a custom shell script as well as using S3 event triggers take a lot of time and administrative effort to implement. You should use a combination of CodeCommit, CodeBuild, CodeDeploy, and CodePipeline instead.
References:
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-configurations.html
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-steps-lambda.html
Check out this AWS CodeDeploy Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-codedeploy/
Tutorials Dojo's AWS Certified DevOps Engineer Professional Exam Study Guide:
https://tutorialsdojo.com/aws-cheat-sheet-aws-certified-devops-engineer-professional/
● Set up an AWS CodeBuild configuration which automatically starts whenever a new code is pushed. Configure CloudFormation to trigger a pipeline in AWS CodePipeline that deploys the new Lambda version. Specify the percentage of traffic that will initially be routed to your updated Lambda function as well as the interval to deploy the code over time in the CloudFormation template.

Q13) An application is hosted in an Auto Scaling group of Amazon EC2 instances with public IP addresses in a public subnet. The instances are configured with a user data script which fetch and install the required system dependencies of the application from the Internet upon launch. A change was recently introduced to prohibit any Internet access from these instances to improve the security but after its implementation, the instances could not get the external dependencies anymore. Upon investigation, all instances are properly running but the hosted application is not starting up completely due to the incomplete installation.
Which of the following is the MOST secure solution to solve this issue and also ensure that the instances do not have public Internet access?

● Set up a brand new security group for the Amazon EC2 instances. Use a whitelist configuration to only allow outbound traffic to the site where all of the application dependencies are hosted. Delete the security group rule once the installation is complete. Use AWS Config to monitor the compliance.
● Deploy the Amazon EC2 instances in a private subnet and associate Elastic IP addresses on each of them. Run a custom shell script to disassociate the Elastic IP addresses after the application has been successfully installed and is running properly.
● Use a NAT gateway to disallow any traffic to the VPC which originated from the public Internet. Deploy the Amazon EC2 instances to a private subnet then set the subnet's route table to use the NAT gateway as its default route.
✅ Download all of the external application dependencies from the public Internet and then store them to an S3 bucket. Set up a VPC endpoint for the S3 bucket and then assign an IAM instance profile to the instances in order to allow them to fetch the required dependencies from the bucket.
Explanation:-A VPC endpoint enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by PrivateLink without requiring an Internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network.
Endpoints are virtual devices. They are horizontally scaled, redundant, and highly available VPC components that allow communication between instances in your VPC and services without imposing availability risks or bandwidth constraints on your network traffic.
There are two types of VPC endpoints: interface endpoints and gateway endpoints. You can create the type of VPC endpoint required by the supported service. S3 and DynamoDB are using Gateway endpoints while most of the services are using Interface endpoints.
You can use an S3 bucket to store the required dependencies and then set up a VPC Endpoint to allow your EC2 instances to access the data without having to traverse the public Internet.
Hence, the correct answer is the option that says: Download all of the external application dependencies from the public Internet and then store them to an S3 bucket. Set up a VPC endpoint for the S3 bucket and then assign an IAM instance profile to the instances in order to allow them to fetch the required dependencies from the bucket.
The option that says: Deploy the Amazon EC2 instances in a private subnet and associate Elastic IP addresses on each of them. Run a custom shell script to disassociate the Elastic IP addresses after the application has been successfully installed and is running properly is incorrect because it is possible that the custom shell script may fail and the disassociation of the Elastic IP addresses might not be fully implemented which will allow the EC2 instances to access the Internet.
The option that says: Use a NAT gateway to disallow any traffic to the VPC which originated from the public Internet. Deploy the Amazon EC2 instances to a private subnet then set the subnet's route table to use the NAT gateway as its default route is incorrect because although a NAT Gateway can safeguard the instances from any incoming traffic that were initiated from the Internet, it still permits them to send outgoing requests externally.
The option that says: Set up a brand new security group for the Amazon EC2 instances. Use a whitelist configuration to only allow outbound traffic to the site where all of the application dependencies are hosted. Delete the security group rule once the installation is complete. Use AWS Config to monitor the compliance is incorrect because this solution has a high operational overhead since the actions are done manually. This is susceptible to human error such as in the event that the DevOps team forgets to delete the security group. The use of AWS Config will just monitor and inform you about the security violation but it won't do anything to remediate the issue.
References:
https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html
https://docs.aws.amazon.com/vpc/latest/userguide/vpce-gateway.html

Check out this Amazon VPC Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-vpc/
Tutorials Dojo's AWS Certified DevOps Engineer Professional Exam Study Guide:
https://tutorialsdojo.com/aws-cheat-sheet-aws-certified-devops-engineer-professional/

Q14) A technology company is planning to develop its custom online forum that covers various AWS-related technologies. They are planning to use AWS Fargate to host the containerized application and Amazon DynamoDB as its data store. The DevOps team is instructed to define the schema of the DynamoDB table with the required indexes, partition key, sort key, projected attributes, and others. To minimize cost, the schema must support certain search operations using the least provisioned read capacity units. A Thread attribute contains the user comments in JSON format. The sample data set is shown in the diagram below:
The online forum should support searches within ForumName attribute for items where the Subject begins with a particular letter, such as 'a' or 'b'. It should allow fetches of items within the given LastPostDateTime time frame as well as the capability to return the threads that have been posted within the last quarter.
Which of the following schema configuration meets the above requirements?

- Set the Subject attribute as the primary key and ForumName as the sort key. Create a Global Secondary Index with Thread as the sort key and LastPostDateTime as a projected attribute.
- Set the ForumName attribute as the primary key and Subject as the sort key. Create a Global Secondary Index with Thread as the sort key and fetch operations for LastPostDateTime.
- Set the Subject attribute as the primary key and ForumName as the sort key. Create a Local Secondary Index with LastPostDateTime as the sort key and the Thread as a projected attribute.
- ✅ Set the ForumName attribute as the primary key and Subject as the sort key. Create a Local Secondary Index with LastPostDateTime as the sort key and the Thread as a projected attribute.

Explanation:-Amazon DynamoDB provides fast access to items in a table by specifying primary key values. However, many applications might benefit from having one or more secondary (or alternate) keys available, to allow efficient access to data with attributes other than the primary key. To address this, you can create one or more secondary indexes on a table and issue Query or Scan requests against these indexes.
A secondary index is a data structure that contains a subset of attributes from a table, along with an alternate key to support Query operations. You can retrieve data from the index using a Query, in much the same way as you use Query with a table. A table can have multiple secondary indexes, which gives your applications access to many different query patterns.
DynamoDB supports two types of secondary indexes:
- Global secondary index — an index with a partition key and a sort key that can be different from those on the base table. A global secondary index is considered "global" because queries on the index can span all of the data in the base table, across all partitions.
- Local secondary index — an index that has the same partition key as the base table, but a different sort key. A local secondary index is "local" in the sense that every partition of a local secondary index is scoped to a base table partition that has the same partition key value.
A local secondary index maintains an alternate sort key for a given partition key value. A local secondary index also contains a copy of some or all of the attributes from its base table; you specify which attributes are projected into the local secondary index when you create the table. The data in a local secondary index is organized by the same partition key as the base table, but with a different sort key. This lets you access data items efficiently across this different dimension. For greater query or scan flexibility, you can create up to five local secondary indexes per table.
Suppose that an application needs to find all of the threads that have been posted within the last three months. Without a local secondary index, the application would have to Scan the entire Thread table and discard any posts that were not within the specified time frame. With a local secondary index, a Query operation could use LastPostDateTime as a sort key and find the data quickly.
In the provided scenario, you can create a local secondary index named LastPostIndex to meet the requirements. Note that the partition key is the same as that of the Thread table, but the sort key is LastPostDateTime as shown in the diagram below:
With LastPostIndex, an application could use ForumName and LastPostDateTime as query criteria. However, to retrieve any additional attributes, DynamoDB must perform additional read operations against the Thread table. These extra reads are known as fetches, and they can increase the total amount of provisioned throughput required for a query.
Suppose that you wanted to populate a webpage with a list of all the threads in "S3" and the number of replies for each thread, sorted by the last reply date/time beginning with the most recent reply. To populate this list, you would need the following attributes:
Subject
Replies
LastPostDateTime
The most efficient way to query this data and to avoid fetch operations would be to project the Replies attribute from the table into the local secondary index, as shown in this diagram.
DynamoDB stores all of the items with the same partition key value contiguously. In this example, given a particular ForumName, a Query operation could immediately locate all of the threads for that forum. Within a group of items with the same partition key value, the items are sorted by sort key value. If the sort key (Subject) is also provided in the query, DynamoDB can narrow down the results that are returned — for example, returning all of the threads in the "S3" forum that have a Subject beginning with the letter "a".
A projection is the set of attributes that is copied from a table into a secondary index. The partition key and sort key of the table are always projected into the index; you can project other attributes to support your application's query requirements. When you query an index, Amazon DynamoDB can access any attribute in the projection as if those attributes were in a table of their own.
Hence, the correct answer is: Set the ForumName attribute as the primary key and Subject as the sort key. Create a Local Secondary Index with LastPostDateTime as the sort key and the Thread as a projected attribute.
The option that says: Set the Subject attribute as the primary key and ForumName as the sort key. Create a Local Secondary Index with LastPostDateTime as the sort key and the Thread as a projected attribute is incorrect because the scenario says that the online forum should support searches within ForumName attribute for items where the Subject begins with a particular letter. DynamoDB stores all of the items with the same partition key value contiguously. In this example, given a particular ForumName, a Query operation could immediately locate all of the threads for that forum. Within a group of items with the same partition key value, the items are sorted by sort key value. If the sort key (Subject) is also provided in the query, DynamoDB can narrow down the results that are returned—for example, returning all of the threads in the "S3" forum that have a Subject beginning with the letter "a". Hence, you should set the ForumName attribute as the primary key and Subject as the sort key instead.
The option that says: Set the ForumName attribute as the primary key and Subject as the sort key. Create a Global Secondary Index with Thread as the sort key and fetch operations for LastPostDateTime is incorrect because using a fetches operation can increase the total amount of provisioned throughput required for a query. Remember that the scenario mentioned that the schema must support certain search operations using the least provisioned read capacity units to minimize cost. In addition, you should create an LSI instead of GSI.
The option that says: Set the Subject attribute as the primary key and ForumName as the sort key. Create a Global Secondary Index with Thread as the sort key and LastPostDateTime as a projected attribute is incorrect because you should use a Local Secondary Index instead. You should also set the ForumName attribute as the primary key and Subject as the sort key instead.
References:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SecondaryIndexes.html
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/LSI.html
Check out this Amazon DynamoDB Cheat Sheet:

https://tutorialsdojo.com/aws-cheat-sheet-amazon-dynamodb/

Q15) A company is planning to host their enterprise web application in an Amazon ECS Cluster which uses the Fargate launch type. The database credentials should be provided to the application image by using environment variables. A DevOps engineer was instructed to ensure that the credentials are highly secured when passed to the image and must be kept in a dedicated storage with lifecycle management as well as key rotation.
Which of the following is the MOST suitable solution that the DevOps engineer should implement?

Store the database credentials using Docker Secrets in the ECS task definition file of the ECS Cluster in order to centrally manage these sensitive data and securely transmit these only to those containers that need access to them. Ensure that the secrets are encrypted and can only be accessed to those services which have been granted explicit access to it via IAM Role, and only while those service tasks are running.

Store the database credentials and encrypt with KMS In the ECS task definition file of the ECS Cluster. Store the task definition JSON file in a private S3 bucket and ensure that HTTPS is enabled on the bucket to encrypt the data in-flight. Create an IAM role to the ECS task definition script that allows access to the specific S3 bucket and then pass the --cli-input-json parameter when calling the ECS register-task-definition. Reference the task definition JSON file in the S3 bucket which contains the database credentials.

Keep the database credentials using the AWS Secrets Manager and then encrypt them using AWS KMS. Set up an IAM Role for your Amazon ECS task execution role and reference it with your task definition which allows access to both KMS and AWS Secrets Manager. Within your container definition, specify secrets with the name of the environment variable to set in the container and the full ARN of the Secrets Manager secret which contains the sensitive data, to present to the container.

Explanation:-Amazon ECS enables you to inject sensitive data into your containers by storing your sensitive data in either AWS Secrets Manager secrets or AWS Systems Manager Parameter Store parameters and then referencing them in your container definition. This feature is supported by tasks using both the EC2 and Fargate launch types.

Within your container definition, specify secrets with the name of the environment variable to set in the container and the full ARN of either the Secrets Manager secret or Systems Manager Parameter Store parameter containing the sensitive data to present to the container. The parameter that you reference can be from a different Region than the container using it but must be from within the same account.

AWS Secrets Manager is a secrets management service that helps you protect access to your applications, services, and IT resources. This service enables you to easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle. Using Secrets Manager, you can secure and manage secrets used to access resources in the AWS Cloud, on third-party services, and on-premises.

If you want a single store for configuration and secrets, you can use Parameter Store. If you want a dedicated secrets store with lifecycle management, use Secrets Manager. Hence, the correct answer is to: Keep the database credentials using the AWS Secrets Manager and then encrypt them using AWS KMS. Set up an IAM Role for your Amazon ECS task execution role and reference it with your task definition which allows access to both KMS and AWS Secrets Manager. Within your container definition, specify secrets with the name of the environment variable to set in the container and the full ARN of the Secrets Manager secret which contains the sensitive data, to present to the container.

The option that says: Keep the database credentials using the AWS Systems Manager Parameter Store and then encrypt them using AWS KMS. Set up an IAM Role for your Amazon ECS task execution role and reference it with your task definition, which allows access to both KMS and the Parameter Store. Within your container definition, specify secrets with the name of the environment variable to set in the container and the full ARN of the Systems Manager Parameter Store parameter containing the sensitive data to present to the container is incorrect because although the use of Systems Manager Parameter Store in securing sensitive data in ECS is valid, this service doesn't provide dedicated storage with lifecycle management and key rotation, unlike Secrets Manager.

The option that says: Store the database credentials and encrypt with KMS In the ECS task definition file of the ECS Cluster. Store the task definition JSON file in a private S3 bucket and ensure that HTTPS is enabled on the bucket to encrypt the data in-flight. Create an IAM role to the ECS task definiton script that allows access to the specific S3 bucket and then pass the --cli-input-json parameter when calling the ECS register-task-definition. Reference the task definition JSON file in the S3 bucket which contains the database credentials is incorrect because although the solution may work, it is not recommended to store sensitive credentials in S3. This entails a lot of overhead and manual configuration steps which can be simplified by simply using the Secrets Manager or Systems Manager Parameter Store.

The option that says: Store the database credentials using Docker Secrets in the ECS task definition file of the ECS Cluster in order to centrally manage these sensitive data and securely transmit these only to those containers that need access to them. Ensure that the secrets are encrypted and can only be accessed to those services which have been granted explicit access to it via IAM Role, and only while those service tasks are running is incorrect because although you can use Docker Secrets to secure the sensitive database credentials, this feature is only applicable in Docker Swarm. In AWS, the recommended way to secure sensitive data is either through the use of Secrets Manager or Systems Manager Parameter Store.

References:
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/specifying-sensitive-data.html
https://aws.amazon.com/blogs/mt/the-right-way-to-store-secrets-using-parameter-store/

Check out this Amazon ECS Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-elastic-container-service-amazon-ecs/

Keep the database credentials using the AWS Systems Manager Parameter Store and then encrypt them using AWS KMS. Set up an IAM Role for your Amazon ECS task execution role and reference it with your task definition, which allows access to both KMS and the Parameter Store. Within your container definition, specify secrets with the name of the environment variable to set in the container and the full ARN of the Systems Manager Parameter Store parameter containing the sensitive data to present to the container.

Q16) You are working as a DevOps engineer for a leading telecommunications company which is planning to host a distributed system in AWS. Their system must be hosted on multiple Linux-based application servers which must use the same configuration file that tracks any changes in the cluster such as adding or removing a server. The configuration file is named as tdojo-nodes.config which contains the list of private IP addresses of the servers in the cluster and other metadata.
Which of the following is the MOST automated way to meet the above requirements?

Use AWS OpsWorks Stacks which layers the application server nodes of the cluster using a Chef recipe associated with the Deploy Lifecycle Event. Set up a configuration which populates the tdojo-nodes.config file and runs each layer's Deploy recipes that updates the configuration file when a cluster change is detected.

Store the tdojo-nodes.config configuration file in Amazon S3 and develop a crontab script that will periodically poll any changes to the file and download it if there is any. Use a Node.js based process manager such as PM2 to restart the application servers in the cluster in the event that the configuration file is modified. Use CloudWatch Events to monitor the changes in your cluster and update the configuration file in S3 for any changes.

Layer the application server nodes of the cluster using AWS OpsWorks Stacks and add a Chef recipe associated with the Configure Lifecycle Event which populates the tdojo-nodes.config file. Set up a configuration which runs each layer's Configure recipes that updates the configuration file when a cluster change is detected.

Explanation:-In AWS OpsWorks Stacks Lifecycle Events, each layer has a set of five lifecycle events, each of which has an associated set of recipes that are specific to the layer. When an event occurs on a layer's instance, AWS OpsWorks Stacks automatically runs the appropriate set of recipes. To provide a custom response to these events, implement custom recipes and assign them to the appropriate events for each layer. AWS OpsWorks Stacks runs those recipes after the event's built-in recipes.

When AWS OpsWorks Stacks runs a command on an instance—for example, a deploy command in response to a Deploy lifecycle event—it adds a set of attributes to the instance's node object that describes the stack's current configuration. For Deploy events and Execute Recipes stack commands, AWS OpsWorks Stacks installs deploy attributes, which provide some additional deployment information.
There are five lifecycle events namely: Setup, Configure, Deploy, UnDeploy and Shutdown. The Configure event occurs on all of the stack's instances when one of the following occurs:
- An instance enters or leaves the online state.
- You associate an Elastic IP address with an instance or disassociate one from an instance.
- You attach an Elastic Load Balancing load balancer to a layer, or detach one from a layer.
For example, suppose that your stack has instances A, B, and C, and you start a new instance, D. After D has finished running its setup recipes, AWS OpsWorks Stacks triggers the Configure event on A, B, C, and D. If you subsequently stop A, AWS OpsWorks Stacks triggers the Configure event on B, C, and D. AWS OpsWorks Stacks responds to the Configure event by running each layer's Configure recipes, which update the instances' configuration to reflect the current set of online instances. The Configure event is therefore a good time to regenerate configuration files. For example, the HAProxy Configure recipes reconfigure the load balancer to accommodate any changes in the set of online application server instances.
Hence, the correct solution for this scenario is: Layer the application server nodes of the cluster using AWS OpsWorks Stacks and add a Chef recipe associated with the Configure Lifecycle Event which populates the tdojo-nodes.config file. Set up a configuration which runs each layer's Configure recipes that updates the configuration file when a cluster change is detected."
The option that says: Use AWS OpsWorks Stacks which layers the application server nodes of the cluster using a Chef recipe associated with the Deploy Lifecycle Event. Set up a configuration which populates the tdojo-nodes.config file and runs each layer's Deploy recipes that updates the configuration file when a cluster change is detected is incorrect because although this is properly using the AWS OpsWorks Stacks Lifecycle Events to track the configuration file, the type of Lifecycle event being used is wrong. You should use the Configure Lifecycle Event instead.
The option that says: Store the tdojo-nodes.config configuration file in CodeCommit and set up a CodeDeploy deployment group based on the tags of each application server nodes of the cluster. Integrate CodeDeploy with Amazon CloudWatch Events to automatically update the configuration file of each server if a new node is added or removed in the cluster then persist the changes in CodeCommit is incorrect because CodeCommit is not a suitable service to use to store your dynamic configuration files. Moreover, the integration of CodeDeploy and CloudWatch Events is only applicable when the actual deployment is being executed and not suitable for monitoring your cluster.
The option that says: Store the tdojo-nodes.config configuration file in Amazon S3 and develop a crontab script that will periodically poll any changes to the file and download it if there is any. Use a Node.js based process manager such as PM2 to restart the application servers in the cluster in the event that the configuration file is modified. Use CloudWatch Events to monitor the changes in your cluster and update the configuration file in S3 for any changes is incorrect because using Amazon S3 to store the configuration file entails a lot of management overhead. A better solution is to use AWS OpsWorks Stacks instead of CloudWatch Events and S3.
References:
https://docs.aws.amazon.com/opsworks/latest/userguide/welcome_classic.html#welcome-classic-lifecycle
https://docs.aws.amazon.com/opsworks/latest/userguide/workingcookbook-events.html
Check out this AWS OpsWorks Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-opsworks/
● Store the tdojo-nodes.config configuration file in CodeCommit and set up a CodeDeploy deployment group based on the tags of each application server nodes of the cluster. Integrate CodeDeploy with Amazon CloudWatch Events to automatically update the configuration file of each server if a new node is added or removed in the cluster then persist the changes in CodeCommit.

Q17) A retail company is planning to migrate its on-premises data center to AWS to scale its infrastructure and reach more customers. Their multitier web applications will be moved to the cloud and will use a variety of AWS services, IAM policies, and custom network configuration. The requirements can be changed anytime by their Solutions Architect, which means there will be a lot of modifications to the AWS components being deployed. CloudFormation will be used to automate, launch, and version-control the new cloud environment in AWS in various regions.
Which of the following is the MOST recommended way to set up CloudFormation in this scenario?

● Prepare multiple separate CloudFormation templates for each logical part of the architecture. Store the CloudFormation resource outputs to AWS Systems Manager Parameter Store. Upload and manage the templates in AWS CodeCommit.
● Prepare a single master CloudFormation template containing all logical parts of the architecture. Upload and maintain the template in AWS CodeCommit.
● Prepare a single master CloudFormation template containing all logical parts of the architecture. Store the CloudFormation resource outputs to a DynamoDB table that will be used by the template. Upload and manage the template in AWS CodeCommit.
✅ Prepare multiple separate CloudFormation templates for each logical part of the architecture. Use cross-stack references to export resources from one AWS CloudFormation stack to another and maintain the templates in AWS CodeCommit.
Explanation:-When you organize your AWS resources based on lifecycle and ownership, you might want to build a stack that uses resources that are in another stack. You can hard-code values or use input parameters to pass resource names and IDs. However, these methods can make templates difficult to reuse or can increase the overhead to get a stack running. Instead, use cross-stack references to export resources from a stack so that other stacks can use them. Stacks can use the exported resources by calling them using the Fn::ImportValue function.
For example, you might have a network stack that includes a VPC, a security group, and a subnet. You want all public web applications to use these resources. By exporting the resources, you allow all stacks with public web applications to use them.
To export resources from one AWS CloudFormation stack to another, create a cross-stack reference. Cross-stack references let you use a layered or service-oriented architecture. Instead of including all resources in a single stack, you create related AWS resources in separate stacks; then you can refer to required resource outputs from other stacks. By restricting cross-stack references to outputs, you control the parts of a stack that are referenced by other stacks.
For example, you might have a network stack with a VPC, a security group, and a subnet for public web applications, and a separate public web application stack. To ensure that the web applications use the security group and subnet from the network stack, you create a cross-stack reference that allows the web application stack to reference resource outputs from the network stack. With a cross-stack reference, owners of the web application stacks don't need to create or maintain networking rules or assets.
To create a cross-stack reference, use the Export output field to flag the value of a resource output for export. Then, use the Fn::ImportValue intrinsic function to import the value.
Hence, the correct answer is: Prepare multiple separate CloudFormation templates for each logical part of the architecture. Use cross-stack references to export resources from one AWS CloudFormation stack to another and maintain the templates in AWS CodeCommit.
The option that says: Prepare a single master CloudFormation template containing all logical parts of the architecture. Store the CloudFormation resource outputs to a DynamoDB table that will be used by the template. Upload and manage the template in AWS CodeCommit is incorrect because it is better to use multiple separate CloudFormation templates to handle each logical part of the architecture, considering that you are deploying multitier web applications that use a variety of AWS services, IAM policies, and custom network configuration. This will provide a better management of each part of your architecture. In addition, you can simply use cross-stack references in CloudFormation instead of storing the resource outputs in a DynamoDB table.
The option that says: Prepare a single master CloudFormation template containing all logical parts of the architecture. Upload and maintain the

template in AWS CodeCommit is incorrect because just as mentioned above, it is better to use multiple separate CloudFormation templates to handle each logical part of the architecture.
The option that says: Prepare multiple separate CloudFormation templates for each logical part of the architecture. Store the CloudFormation resource outputs to AWS Systems Manager Parameter Store. Upload and manage the templates in AWS CodeCommit is incorrect because it is better to handle each logical part of the architecture on a separate CloudFormation template for easier management. Although you can integrate AWS Systems Manager Parameter Store with CloudFormation, this service is more suitable to store data such as passwords, database strings, and license codes as parameter values but not resource outputs. You should create a cross-stack reference to export resources from one AWS CloudFormation stack to another.
References:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/best-practices.html#cross-stack
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/walkthrough-crossstackref.html
https://aws.amazon.com/premiumsupport/knowledge-center/cloudformation-reference-resource/
Check out this AWS CloudFormation Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-cloudformation/
Check out this AWS CodeCommit Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-codecommit/

Q18) On a recent AWS resources audit on your AWS Test environment account, you have found several load balancers with minimal usage in which some of them do not have back-end EC2 instances at all. Your developers are using these load balancers to test their applications on the public Internet, but they forgot to delete them after testing. These load balancers are also showing up on AWS Trusted Advisor as unused/underutilized which incur unnecessary costs. You want to be notified whenever there are changes in the Trusted Advisor checks so that you can quickly take action based on the results.
Which of the following options will help you achieve this? (Select THREE)

Launch a small EC2 instance that runs a custom script that programmatically checks the utilization of your load balancers. Send out email notification via Amazon SES.
✅ Create a Lambda function and integrate it with CloudWatch Events. Configure the function to run on a regular basis and to check AWS Trusted Advisor via API. Based on the results, publish a message to an Amazon SNS Topic to notify the subscribers.
Explanation:-AWS Trusted Advisor is integrated with the Amazon CloudWatch Events and Amazon CloudWatch services. You can use Amazon CloudWatch Events to detect and react to changes in the status of Trusted Advisor checks. And you can use Amazon CloudWatch to create alarms on Trusted Advisor metrics for check status changes, resource status changes, and service limit utilization
Based on the rules that you create, CloudWatch Events invokes one or more target actions when a check status changes to the value you specify in a rule. Depending on the type of status change, you might want to send notifications, capture status information, take corrective action, initiate events, or take other actions.
Hence, the correct answers are:
- Create a Lambda function and integrate it with CloudWatch Events. Configure the function to run on a regular basis and to check AWS Trusted Advisor via API. Based on the results, publish a message to an Amazon SNS Topic to notify the subscribers.
- Utilize CloudWatch Events to monitor Trusted Advisor recommendation results. Set up a trigger to send an email using SNS to notify you about the results of the check.
- Use Amazon CloudWatch to create alarms on Trusted Advisor metrics in order to detect the load balancers with low utilization. Specify an SNS topic for notification.
The option that says: Create a CloudWatch alarm to monitor the utilization of your load balancers using ELB metrics. Use Amazon SES to send an email notification is incorrect because you have to use the AWS Trusted Advisor metrics and not ELB metrics. Moreover, you have to use SNS and not SES for sending an email notification.
The option that says: Use AWS Config in order to automatically detect idle load balancers. Use Amazon SNS to deliver the notifications is incorrect because AWS Config cannot detect the utilization of AWS resources. You have to use AWS Trusted Advisor instead.
The option that says: Launch a small EC2 instance that runs a custom script that programmatically checks the utilization of your load balancers. Send out email notification via Amazon SES is incorrect because this entails a lot of manual action to develop a custom script. You can simply just use Trusted Advisor to monitor the idle load balancers. Moreover, you have to use SNS for notifications and not SES.
References:
https://docs.aws.amazon.com/awssupport/latest/user/cloudwatch-ta.html
https://docs.aws.amazon.com/awssupport/latest/user/trustedadvisor.html
https://docs.aws.amazon.com/awssupport/latest/user/cloudwatch-events-ta.html
Check out these AWS Trusted Advisor and CloudWatch Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-aws-trusted-advisor/
https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudwatch/
Create a CloudWatch alarm to monitor the utilization of your load balancers using ELB metrics. Use Amazon SES to send an email notification.
✅ Use Amazon CloudWatch to create alarms on Trusted Advisor metrics in order to detect the load balancers with low utilization. Specify an SNS topic for notification.
Explanation:-AWS Trusted Advisor is integrated with the Amazon CloudWatch Events and Amazon CloudWatch services. You can use Amazon CloudWatch Events to detect and react to changes in the status of Trusted Advisor checks. And you can use Amazon CloudWatch to create alarms on Trusted Advisor metrics for check status changes, resource status changes, and service limit utilization
Based on the rules that you create, CloudWatch Events invokes one or more target actions when a check status changes to the value you specify in a rule. Depending on the type of status change, you might want to send notifications, capture status information, take corrective action, initiate events, or take other actions.
Hence, the correct answers are:
- Create a Lambda function and integrate it with CloudWatch Events. Configure the function to run on a regular basis and to check AWS Trusted Advisor via API. Based on the results, publish a message to an Amazon SNS Topic to notify the subscribers.
- Utilize CloudWatch Events to monitor Trusted Advisor recommendation results. Set up a trigger to send an email using SNS to notify you about the results of the check.
- Use Amazon CloudWatch to create alarms on Trusted Advisor metrics in order to detect the load balancers with low utilization. Specify an SNS topic for notification.
The option that says: Create a CloudWatch alarm to monitor the utilization of your load balancers using ELB metrics. Use Amazon SES to send an email notification is incorrect because you have to use the AWS Trusted Advisor metrics and not ELB metrics. Moreover, you have to use SNS and not SES for sending an email notification.
The option that says: Use AWS Config in order to automatically detect idle load balancers. Use Amazon SNS to deliver the notifications is incorrect because AWS Config cannot detect the utilization of AWS resources. You have to use AWS Trusted Advisor instead.
The option that says: Launch a small EC2 instance that runs a custom script that programmatically checks the utilization of your load balancers. Send out email notification via Amazon SES is incorrect because this entails a lot of manual action to develop a custom script. You can simply just use Trusted Advisor to monitor the idle load balancers. Moreover, you have to use SNS for notifications and not SES.

References:
https://docs.aws.amazon.com/awssupport/latest/user/cloudwatch-ta.html
https://docs.aws.amazon.com/awssupport/latest/user/trustedadvisor.html
https://docs.aws.amazon.com/awssupport/latest/user/cloudwatch-events-ta.html
Check out these AWS Trusted Advisor and CloudWatch Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-aws-trusted-advisor/
https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudwatch/
✅ Utilize CloudWatch Events to monitor Trusted Advisor recommendation results. Set up a trigger to send an email using SNS to notify you about the results of the check.
Explanation:-AWS Trusted Advisor is integrated with the Amazon CloudWatch Events and Amazon CloudWatch services. You can use Amazon CloudWatch Events to detect and react to changes in the status of Trusted Advisor checks. And you can use Amazon CloudWatch to create alarms on Trusted Advisor metrics for check status changes, resource status changes, and service limit utilization
Based on the rules that you create, CloudWatch Events invokes one or more target actions when a check status changes to the value you specify in a rule. Depending on the type of status change, you might want to send notifications, capture status information, take corrective action, initiate events, or take other actions.
Hence, the correct answers are:
- Create a Lambda function and integrate it with CloudWatch Events. Configure the function to run on a regular basis and to check AWS Trusted Advisor via API. Based on the results, publish a message to an Amazon SNS Topic to notify the subscribers.
- Utilize CloudWatch Events to monitor Trusted Advisor recommendation results. Set up a trigger to send an email using SNS to notify you about the results of the check.
- Use Amazon CloudWatch to create alarms on Trusted Advisor metrics in order to detect the load balancers with low utilization. Specify an SNS topic for notification.
The option that says: Create a CloudWatch alarm to monitor the utilization of your load balancers using ELB metrics. Use Amazon SES to send an email notification is incorrect because you have to use the AWS Trusted Advisor metrics and not ELB metrics. Moreover, you have to use SNS and not SES for sending an email notification.
The option that says: Use AWS Config in order to automatically detect idle load balancers. Use Amazon SNS to deliver the notifications is incorrect because AWS Config cannot detect the utilization of AWS resources. You have to use AWS Trusted Advisor instead.
The option that says: Launch a small EC2 instance that runs a custom script that programmatically checks the utilization of your load balancers. Send out email notification via Amazon SES is incorrect because this entails a lot of manual action to develop a custom script. You can simply just use Trusted Advisor to monitor the idle load balancers. Moreover, you have to use SNS for notifications and not SES.
References:
https://docs.aws.amazon.com/awssupport/latest/user/cloudwatch-ta.html
https://docs.aws.amazon.com/awssupport/latest/user/trustedadvisor.html
https://docs.aws.amazon.com/awssupport/latest/user/cloudwatch-events-ta.html
Check out these AWS Trusted Advisor and CloudWatch Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-aws-trusted-advisor/
https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudwatch/
⚪ Use AWS Config in order to automatically detect the load balancers with low utilization. Use Amazon SNS to deliver the notifications.

Q19) A company would like to set up an audit process to ensure that their enterprise application is running exclusively on Amazon EC2 Dedicated Hosts. They are also concerned about the increasing costs of their application software licensing from their third-party vendor. To meet the compliance requirement, a DevOps Engineer must create a workflow to audit the enterprise applications hosted in their VPC.
Which of the following options should the Engineer implement to satisfy the requirement with the LEAST administrative overhead?

⚪ Record configuration changes for Dedicated Hosts using the AWS Systems Manager Configuration Compliance. Utilize the PutComplianceItems API action to scan and populate a collection of noncompliant Amazon EC2 instances based on their host placement configuration. Store these instance IDs to Systems Manager Parameter Store and generate a report by calling the ListComplianceSummaries API action.
⚪ Install the Amazon Inspector agent to all EC2 instances. Record configuration changes for Dedicated Hosts by using Amazon Inspector. Set up an automatic assessment runs through a Lambda Function by using the inspector-scheduled-run blueprint.
⚪ Record configuration changes for Dedicated Hosts by AWS CloudTrail. Filter all EC2 RunCommand API actions in the logs to detect any changes to the instances. Analyze the host placement of the instance using a Lambda function and store the instance IDs of noncompliant resources in an Amazon S3 bucket. Generate a report by using Amazon Athena to query the S3 data.
✅ Record configuration changes for EC2 Instances and Dedicated Hosts by turning on the Config Recording option in AWS Config. Set up a custom AWS Config rule that triggers a Lambda function by using the config-rule-change-triggered blueprint. Customize the predefined evaluation logic to verify host placement to return a NON_COMPLIANT result whenever the EC2 instance is not running on a Dedicated Host. Use the AWS Config report to address noncompliant Amazon EC2 instances.
Explanation:-You can use AWS Config to record configuration changes for Dedicated Hosts, and instances that are launched, stopped, or terminated on them. You can then use the information captured by AWS Config as a data source for license reporting.
AWS Config records configuration information for Dedicated Hosts and instances individually and pairs this information through relationships. There are three reporting conditions:
AWS Config recording status — When On, AWS Config is recording one or more AWS resource types, which can include Dedicated Hosts and Dedicated Instances. To capture the information required for license reporting, verify that hosts and instances are being recorded with the following fields.
-Host recording status — When Enabled, the configuration information for Dedicated Hosts is recorded.
-Instance recording status — When Enabled, the configuration information for Dedicated Instances is recorded.
If any of these three conditions are disabled, the icon in the Edit Config Recording button is red. To derive the full benefit of this tool, ensure that all three recording methods are enabled. When all three are enabled, the icon is green. To edit the settings, choose Edit Config Recording. You are directed to the Set up AWS Config page in the AWS Config console, where you can set up AWS Config and start recording for your hosts, instances, and other supported resource types. AWS Config records your resources after it discovers them, which might take several minutes.
After AWS Config starts recording configuration changes to your hosts and instances, you can get the configuration history of any host that you have allocated or released and any instance that you have launched, stopped, or terminated. For example, at any point in the configuration history of a Dedicated Host, you can look up how many instances are launched on that host, along with the number of sockets and cores on the host. For any of those instances, you can also look up the ID of its Amazon Machine Image (AMI). You can use this information to report on licensing for your own server-bound software that is licensed per-socket or per-core.
You can view configuration histories in any of the following ways.
-By using the AWS Config console. For each recorded resource, you can view a timeline page, which provides a history of configuration details. To view this page, choose the gray icon in the Config Timeline column of the Dedicated Hosts page.
-By running AWS CLI commands. First, you can use the list-discovered-resources command to get a list of all hosts and instances. Then, you can

use the get-resource-config-history command to get the configuration details of a host or instance for a specific time interval.
-By using the AWS Config API in your applications. First, you can use the ListDiscoveredResources action to get a list of all hosts and instances. Then, you can use the GetResourceConfigHistory action to get the configuration details of a host or instance for a specific time interval.
Hence, the correct answer is: Record configuration changes for EC2 Instances and Dedicated Hosts by turning on the Config Recording option in AWS Config. Set up a custom AWS Config rule that triggers a Lambda function by using the config-rule-change-triggered blueprint. Customize the predefined evaluation logic to verify host placement to return a NON_COMPLIANT result whenever the EC2 instance is not running on a Dedicated Host. Use the AWS Config report to address noncompliant Amazon EC2 instances.
The option that says: Record configuration changes for Dedicated Hosts using the AWS Systems Manager Configuration Compliance. Utilize the PutComplianceItems API action to scan and populate a collection of noncompliant Amazon EC2 instances based on their host placement configuration. Store these instance IDs to Systems Manager Parameter Store and generate a report by calling the ListComplianceSummaries API action is incorrect because the AWS Systems Manager Configuration Compliance service is primarily used to scan your fleet of managed instances for patch compliance and configuration inconsistencies. A better solution is to use AWS Config to record the status of your Dedicated Hosts.
The option that says: Install the Amazon Inspector agent to all EC2 instances. Record configuration changes for Dedicated Hosts by using Amazon Inspector. Set up an automatic assessment runs through a Lambda Function by using the inspector-scheduled-run blueprint is incorrect because Amazon Inspector is just an automated security assessment service that helps improve the security and compliance of applications deployed on AWS. It is not capable of recording the status of your EC2 instances nor detect if they are configured as a Dedicated Host.
The option that says: Record configuration changes for Dedicated Hosts by AWS CloudTrail. Filter all EC2 RunCommand API actions in the logs to detect any changes to the instances. Analyze the host placement of the instance using a Lambda function and store the instance IDs of noncompliant resources in an Amazon S3 bucket. Generate a report by using Amazon Athena to query the S3 data is incorrect because although this may be a possible solution, it entails a lot of administrative effort in comparison to just using AWS Config.
References:
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/dedicated-hosts-aws-config.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-compliance-about.html#sysman-compliance-custom
https://aws.amazon.com/blogs/aws/now-available-ec2-dedicated-hosts/
Check out these AWS Systems Manager and AWS Config Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-aws-systems-manager/
https://tutorialsdojo.com/aws-cheat-sheet-aws-config/

Q20) A company has a suite of applications that uses the MERN stack for the presentation tier and NGINX for the web tier. AWS CodeDeploy will be used to automate its application deployments. The DevOps team created a deployment group for their TEST environment and performed functional tests within the application. The team will set up additional deployment groups for STAGING and PROD environments later on. The current log level is configured within the NGINX settings, but the team wants to change this configuration dynamically when the deployment occurs. This will enable them to set different log level configurations depending on the deployment group without having a different application revision for each group.
Which among the options below provides the LEAST management overhead and does not require different script versions for each deployment group?

● Develop a custom shell script that uses the DEPLOYMENT_GROUP_ID environment variable in CodeDeploy to identify which deployment group the Amazon EC2 instance is associated with. In the appspec.yml config file, add a reference to this script as part of the ApplicationStart lifecycle hook. Configure the log level settings of the instance based on the result of the script.
● Use the AWS Resource Groups Tag Editor to add a tag on each EC2 instance based on its deployment group. Attach a shell script in the application revision that will fetch the instance tag using the aws ec2 describe-tags CLI command to determine which deployment group the Amazon EC2 instance is associated with. In the appspec.yml config file, add a reference to this script as part of the ValidateService lifecycle hook. Configure the log level settings of the instance based on the result of the script.
✅ Develop a custom shell script that uses the DEPLOYMENT_GROUP_NAME environment variable in CodeDeploy to identify which deployment group the Amazon EC2 instance is associated with. In the appspec.yml config file, add a reference to this script as part of the BeforeInstall lifecycle hook. Configure the log level settings of the instance based on the result of the script.
Explanation:-The content in the 'hooks' section of the AppSpec file varies, depending on the compute platform for your deployment. The 'hooks' section for an EC2/On-Premises deployment contains mappings that link deployment lifecycle event hooks to one or more scripts. The 'hooks' section for a Lambda or an Amazon ECS deployment specifies Lambda validation functions to run during a deployment lifecycle event. If an event hook is not present, no operation is executed for that event.
There is also a set of available environment variables for the hooks. During each deployment lifecycle event, hook scripts can access the following environment variables:
During each deployment lifecycle event, hook scripts can access the following environment variables:
APPLICATION_NAME - The name of the application in CodeDeploy that is part of the current deployment (for example, WordPress_App).
DEPLOYMENT_ID - The ID CodeDeploy has assigned to the current deployment (for example, d-AB1CDEF23).
DEPLOYMENT_GROUP_NAME - The name of the deployment group in CodeDeploy that is part of the current deployment (for example, WordPress_DepGroup).
DEPLOYMENT_GROUP_ID - The ID of the deployment group in CodeDeploy that is part of the current deployment (for example, b1a2189b-dd90-4ef5-8f40-4c1c5EXAMPLE).
LIFECYCLE_EVENT - The name of the current deployment lifecycle event (for example, AfterInstall).
These environment variables are local to each deployment lifecycle event.
The following script changes the listening port on an Apache HTTP server to 9090 instead of 80 if the value of DEPLOYMENT_GROUP_NAME is equal to Staging. This script must be invoked during the BeforeInstall deployment lifecycle event:

```
if [ "$DEPLOYMENT_GROUP_NAME" == "Staging" ]
then
sed -i -e 's/Listen 80/Listen 9090/g' /etc/httpd/conf/httpd.conf
fi
```

The following script example changes the verbosity level of messages recorded in its error log from warning to debug if the value of the DEPLOYMENT_GROUP_NAME environment variable is equal to Staging. This script must be invoked during the BeforeInstall deployment lifecycle event:

```
if [ "$DEPLOYMENT_GROUP_NAME" == "Staging" ]
then
sed -i -e 's/LogLevel warn/LogLevel debug/g' /etc/httpd/conf/httpd.conf
fi
```

Hence, the correct answer is: Develop a custom shell script that uses the DEPLOYMENT_GROUP_NAME environment variable in CodeDeploy to identify which deployment group the Amazon EC2 instance is associated with. In the appspec.yml config file, add a reference to this script as part of the BeforeInstall lifecycle hook. Configure the log level settings of the instance based on the result of the script.
The option that says: Use the AWS Resource Groups Tag Editor to add a tag on each EC2 instance based on its deployment group. Attach a shell script in the application revision that will fetch the instance tag using the aws ec2 describe-tags CLI command to determine which deployment group

the Amazon EC2 instance is associated with. In the appspec.yml config file, add a reference to this script as part of the ValidateService lifecycle hook. Configure the log level settings of the instance based on the result of the script is incorrect because it is better to use the DEPLOYMENT_GROUP_NAME environment variable in CodeDeploy instead of adding tags and using AWS CLI for deployment. Take note that you also have to provide your access keys for the AWS CLI in order to run the aws ec2 describe-tags CLI command. This creates an unnecessary management overhead. Moreover, you have to add a reference to the script as part of the BeforeInstall lifecycle hook and not to the ValidateService.
The option that says: Set up a custom environment variable in CodeDeploy for each environment with a value of TEST, STAGING or PROD. Attach a shell script in the application revision that will read the custom variable and determine which deployment group the Amazon EC2 instance is associated with. In the appspec.yml config file, add a reference to this script as part of the ValidateService lifecycle hook. Configure the log level settings of the instance based on the result of the script is incorrect because there is no such thing as custom environment variable in CodeDeploy. During each deployment lifecycle event, hook scripts can only access the following predefined environment variables: APPLICATION_NAME, DEPLOYMENT_ID, DEPLOYMENT_GROUP_NAME, DEPLOYMENT_GROUP_ID and LIFECYCLE_EVENT. In addition, you have to add a reference to the script as part of the BeforeInstall lifecycle hook and not to the ValidateService.
The option that says: Develop a custom shell script that uses the DEPLOYMENT_GROUP_ID environment variable in CodeDeploy to identify which deployment group the Amazon EC2 instance is associated with. In the appspec.yml config file, add a reference to this script as part of the ApplicationStart lifecycle hook. Configure the log level settings of the instance based on the result of the script is incorrect because you have to use the DEPLOYMENT_GROUP_NAME environment variable in CodeDeploy to identify which deployment group the Amazon EC2 instance is associated with and not DEPLOYMENT_GROUP_ID. Moreover, you have to add a reference to the script as part of the BeforeInstall lifecycle hook and not to the ApplicationStart.
References:
https://aws.amazon.com/blogs/devops/using-codedeploy-environment-variables/
https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file-structure-hooks.html#reference-appspec-file-structure-environment-variable-availability
Check out this AWS CodeDeploy Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-codedeploy/

- Set up a custom environment variable in CodeDeploy for each environment with a value of TEST, STAGING or PROD. Attach a shell script in the application revision that will read the custom variable and determine which deployment group the Amazon EC2 instance is associated with. In the appspec.yml config file, add a reference to this script as part of the ValidateService lifecycle hook. Configure the log level settings of the instance based on the result of the script.

Q21) A DevOps engineer has been tasked to implement a reliable solution to maintain all of their Windows and Linux servers both in AWS and in on-premises data center. There should be a system that allows them to easily update the operating systems of their servers and apply the core application patches with minimum management overhead. The patches must be consistent across all levels in order to meet the company's security compliance.
Which of the following is the MOST suitable solution that you should implement?

- Configure and install the AWS OpsWorks agent on all of your EC2 instances in your VPC and your on-premises servers. Set up an OpsWorks stack with separate layers for each OS then fetch a recipe from the Chef supermarket site (supermarket.chef.io) to automate the execution of the patch commands for each layer during maintenance windows.
- Develop a custom python script to install the latest OS patches on the Linux servers. Set up a scheduled job to automatically run this script using the cron scheduler on Linux servers. Enable Windows Update in order to automatically patch Windows servers or set up a scheduled task using Windows Task Scheduler to periodically run the python script.
- ✅ Configure and install AWS Systems Manager agent on all of the EC2 instances in your VPC as well as your physical servers on-premises. Use the Systems Manager Patch Manager service and specify the required Systems Manager Resource Groups for your hybrid architecture. Utilize a preconfigured patch baseline and then run scheduled patch updates during maintenance windows.

Explanation:-AWS Systems Manager Patch Manager automates the process of patching managed instances with both security-related and other types of updates. You can use the Patch Manager to apply patches for both operating systems and applications. (On Windows Server, application support is limited to updates for Microsoft applications.) You can patch fleets of Amazon EC2 instances or your on-premises servers and virtual machines (VMs) by operating system type. This includes supported versions of Windows Server, Ubuntu Server, Red Hat Enterprise Linux (RHEL), SUSE Linux Enterprise Server (SLES), CentOS, Amazon Linux, and Amazon Linux 2. You can scan instances to see only a report of missing patches, or you can scan and automatically install all missing patches.
Patch Manager uses patch baselines, which include rules for auto-approving patches within days of their release, as well as a list of approved and rejected patches. You can install patches on a regular basis by scheduling patching to run as a Systems Manager maintenance window task. You can also install patches individually or to large groups of instances by using Amazon EC2 tags. You can add tags to your patch baselines themselves when you create or update them.
A resource group is a collection of AWS resources that are all in the same AWS Region, and that match criteria provided in a query. You build queries in the AWS Resource Groups (Resource Groups) console, or pass them as arguments to Resource Groups commands in the AWS CLI.
With AWS Resource Groups, you can create a custom console that organizes and consolidates information based on criteria that you specify in tags. After you add resources to a group you created in Resource Groups, use AWS Systems Manager tools such as Automation to simplify management tasks on your resource group. You can also use the resource groups you create as the basis for viewing monitoring and configuration insights in Systems Manager.
Hence, the correct answer is: Configure and install AWS Systems Manager agent on all of the EC2 instances in your VPC as well as your physical servers on-premises. Use the Systems Manager Patch Manager service and specify the required Systems Manager Resource Groups for your hybrid architecture. Utilize a preconfigured patch baseline and then run scheduled patch updates during maintenance windows.
The option which uses an AWS OpsWorks agent is incorrect because the OpsWorks service is primarily used for application deployment and not for applying application patches or upgrading the operating systems of your servers.
The option which uses a custom python script is incorrect because this solution entails a high management overhead since you need to develop a new script and maintain a number of cron schedulers in your Linux servers and Windows Task Scheduler jobs on your Windows servers.
The option which uses the AWS Systems Manager Parameter Store is incorrect because this is not a suitable service to use to handle the patching activities of your servers. You have to use AWS Systems Manager Patch Manager instead.
References:
https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-patch.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-resource-groups.html
Check out this AWS Systems Manager Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-systems-manager/

- Store the login credentials of each Linux and Windows servers on the AWS Systems Manager Parameter Store. Use Systems Manager Resource Groups to set up one group for your Linux servers and another one for your Windows servers. Remotely login, run, and deploy the patch updates to all of your servers using the credentials stored in the Systems Manager Parameter Store and through the use of the Systems Manager Run Command.

Q22) A multi-tier enterprise web application is hosted in an Auto Scaling group of On-Demand Amazon EC2 instances across multiple Availability Zones behind an Application Load Balancer. For its database tier, Amazon Aurora is used in storing thousands of transactions and user data. A DevOps Engineer was instructed to implement a secure and manageable method in obtaining the database password credentials as well as providing access to AWS resources during deployment.
Which among the following options is the MOST suitable setup that the Engineer should use?

- Using the AWS Systems Manager Parameter Store service, store the sensitive database credentials as SecureString parameters and the access keys as plaintext parameters. Configure your instances to retrieve the credentials and access keys from the Parameter Store.
- Create an IAM role for your EC2 instances that allows access to other AWS services. Configure your instances to use the IAM Role and store the database credentials in an encrypted configuration file in an Amazon S3 bucket with SSE.
- Store the sensitive database credentials and access keys from AWS Systems Manager Parameter Store as SecureString parameters. Configure your instances to retrieve the credentials and access keys from the Parameter Store.
- ✅ Store the sensitive database credentials from AWS Secrets Manager. Create an IAM role for your EC2 instances that allows access to other AWS services. Configure your instances to use the IAM Role and retrieve the credentials from the AWS Secrets Manager.

Explanation:-AWS Secrets Manager is an AWS service that makes it easier for you to manage secrets. Secrets can be database credentials, passwords, third-party API keys, and even arbitrary text. You can store and control access to these secrets centrally by using the Secrets Manager console, the Secrets Manager command line interface (CLI), or the Secrets Manager API and SDKs.
In the past, when you created a custom application that retrieves information from a database, you typically had to embed the credentials (the secret) for accessing the database directly in the application. When the time came to rotate the credentials, you had to do much more than just create new credentials. You had to invest time to update the application to use the new credentials. Then you had to distribute the updated application. If you had multiple applications with shared credentials and you missed updating one of them, the application would break. Because of this risk, many customers have chosen not to regularly rotate their credentials, which effectively substitutes one risk for another.
Secrets Manager enables you to replace hardcoded credentials in your code (including passwords), with an API call to Secrets Manager to retrieve the secret programmatically. This helps ensure that the secret can't be compromised by someone examining your code, because the secret simply isn't there. Also, you can configure Secrets Manager to automatically rotate the secret for you according to a specified schedule. This enables you to replace long-term secrets with short-term ones, which significantly reduces the risk of compromise.
An IAM role is an IAM identity that you can create in your account that has specific permissions. An IAM role is similar to an IAM user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session.
You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources. For example, you might want to grant users in your AWS account access to resources they don't usually have, or grant users in one AWS account access to resources in another account. Or you might want to allow a mobile app to use AWS resources, but not want to embed AWS keys within the app (where they can be difficult to rotate and where users can potentially extract them). Sometimes you want to give AWS access to users who already have identities defined outside of AWS, such as in your corporate directory. Or, you might want to grant access to your account to third parties so that they can perform an audit on your resources.
Hence, the correct answer is: Store the sensitive database credentials from AWS Secrets Manager. Create an IAM role for your EC2 instances that allows access to other AWS services. Configure your instances to use the IAM Role and retrieve the credentials from the AWS Secrets Manager.
The option that says: Store the sensitive database credentials and access keys from AWS Systems Manager Parameter Store as SecureString parameters. Configure your instances to retrieve the credentials and access keys from the Parameter Store is incorrect because you should use an IAM role instead of directly using access keys. AWS Systems Manager Parameter Store is primarily used to store passwords, database strings, and license codes as parameter values.
The option that says: Using the AWS Systems Manager Parameter Store service, store the sensitive database credentials as SecureString parameters and the access keys as plaintext parameters. Configure your instances to retrieve the credentials and access keys from the Parameter Store is incorrect because although it is valid to store the database credentials to AWS Systems Manager Parameter Store, you still need to use an IAM Role to allow EC2 instances to access your AWS resources and not by using access keys.
The option that says: Create an IAM role for your EC2 instances that allows access to other AWS services. Configure your instances to use the IAM Role and store the database credentials in an encrypted configuration file in an Amazon S3 bucket with SSE is incorrect because it is not appropriate to store sensitive database credentials to S3 even if it is using Server Side Encryption. You should use either AWS Systems Manager Parameter Store or AWS Secrets Manager to store the credentials.
References:
https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#use-roles-with-ec2
https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html
Check out this AWS Identity & Access Management (IAM) Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-identity-and-access-management-iam/
Check out this AWS Secrets Manager Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-secrets-manager/

Q23) A commercial bank has a hybrid cloud architecture in AWS where its online banking platform is hosted. The CTO instructed its Lead DevOps Engineer to implement a suitable deployment set up that minimizes the impact on their production environment. The CI/CD process should be configured as follows:
- A new fleet of Amazon EC2 instances should be automatically launched first before the actual production deployment. The additional instances will serve traffic during the deployment.
- All available EC2 instances across various Availability Zones must be load-balanced and must automatically heal if it becomes impaired due to an underlying hardware failure in Amazon EC2.
- At least half of the incoming traffic must be rerouted to the new application version that is hosted to the new instances.
- The deployment should be considered successful if traffic is rerouted to at least half of the available EC2 instances.
- All temporary files must be deleted before routing traffic to the new fleet of instances. Ensure that any other files that were automatically generated during the deployment process are removed.
- To reduce costs, the EC2 instances that host the old version in the deployment group must be terminated immediately.
What should the Engineer do to satisfy these requirements?

- Launch an Application Load Balancer and use a blue/green deployment for releasing new application versions. Associate the Auto Scaling group and the Application Load Balancer target group with the deployment group. In CodeDeploy, use the CodeDeployDefault.HalfAtAtime as the deployment configuration and configure it to terminate the original instances in the Auto Scaling group after deployment. Use the BlockTraffic hook within appspec.yml to purge the temporary files.
- Launch an Application Load Balancer and use in-place deployment for releasing new application versions. Use the CodeDeployDefault.OneAtAtime as the deployment configuration and associate the Auto Scaling group with the deployment group. Configure AWS CodeDeploy to terminate all EC2 instances in the original Auto Scaling group and use the AllowTraffic hook within the appspec.yml configuration file

to purge the temporary files.

Launch an Application Load Balancer and use an in-place deployment for releasing new application versions. Associate the Auto Scaling group and Application Load Balancer target group with the deployment group. In CodeDeploy, use the CodeDeployDefault AllatOnce as a deployment configuration and add a configuration to terminate the original instances in the Auto Scaling group after the deployment. Use the BlockTraffic hook within appsec.yml to purge the temporary files.

Launch an Application Load Balancer and use a blue/green deployment for releasing new application versions. Associate the Auto Scaling group and the Application Load Balancer target group with the deployment group. Create a custom deployment configuration for the deployment group in CodeDeploy with minimum healthy hosts defined as 50% and configure it to also terminate the original instances in the Auto Scaling group after deployment. Use the BeforeAllowTraffic Traffic hook within appsec.yml to purge the temporary files.

Explanation:-The content in the 'hooks' section of the AppSpec file varies, depending on the compute platform for your deployment. The 'hooks' section for an EC2/On-Premises deployment contains mappings that link deployment lifecycle event hooks to one or more scripts. The 'hooks' section for a Lambda or an Amazon ECS deployment specifies Lambda validation functions to run during a deployment lifecycle event. If an event hook is not present, no operation is executed for that event. This section is required only if you are running scripts or Lambda validation functions as part of the deployment.

An EC2/On-Premises deployment hook is executed once per deployment to an instance. You can specify one or more scripts to run in a hook. Each hook for a lifecycle event is specified with a string on a separate line. Here are descriptions of the hooks available for use in your AppSpec file.

ApplicationStop – This deployment lifecycle event occurs even before the application revision is downloaded. You can specify scripts for this event to gracefully stop the application or remove currently installed packages in preparation for a deployment. The AppSpec file and scripts used for this deployment lifecycle event are from the previous successfully deployed application revision.

DownloadBundle – During this deployment lifecycle event, the CodeDeploy agent copies the application revision files to a temporary location: /opt/codedeploy-agent/deployment-root/deployment-group-id/deployment-id/deployment-archive folder on Amazon Linux, Ubuntu Server, and RHEL Amazon EC2 instances.

C:\ProgramData\Amazon\CodeDeploy\deployment-group-id\deployment-id\deployment-archive folder on Windows Server Amazon EC2 instances. This event is reserved for the CodeDeploy agent and cannot be used to run scripts.

BeforeInstall – You can use this deployment lifecycle event for preinstall tasks, such as decrypting files and creating a backup of the current version.

Install – During this deployment lifecycle event, the CodeDeploy agent copies the revision files from the temporary location to the final destination folder. This event is reserved for the CodeDeploy agent and cannot be used to run scripts.

AfterInstall – You can use this deployment lifecycle event for tasks such as configuring your application or changing file permissions.

ApplicationStart – You typically use this deployment lifecycle event to restart services that were stopped during ApplicationStop.

ValidateService – This is the last deployment lifecycle event. It is used to verify the deployment was completed successfully.

BeforeBlockTraffic – You can use this deployment lifecycle event to run tasks on instances before they are deregistered from a load balancer.

BlockTraffic – During this deployment lifecycle event, internet traffic is blocked from accessing instances that are currently serving traffic. This event is reserved for the CodeDeploy agent and cannot be used to run scripts.

AfterBlockTraffic – You can use this deployment lifecycle event to run tasks on instances after they are deregistered from a load balancer.

BeforeAllowTraffic – You can use this deployment lifecycle event to run tasks on instances before they are registered with a load balancer.

AllowTraffic – During this deployment lifecycle event, internet traffic is allowed to access instances after a deployment. This event is reserved for the CodeDeploy agent and cannot be used to run scripts.

AfterAllowTraffic – You can use this deployment lifecycle event to run tasks on instances after they are registered with a load balancer.

Hence, the correct answer is: Launch an Application Load Balancer and use a blue/green deployment for releasing new application versions. Associate the Auto Scaling group and the Application Load Balancer target group with the deployment group. Create a custom deployment configuration for the deployment group in CodeDeploy with minimum healthy hosts defined as 50% and configure it to also terminate the original instances in the Auto Scaling group after deployment. Use the BeforeAllowTraffic Traffic hook within appsec.yml to purge the temporary files.

The option that says: Launch an Application Load Balancer and use in-place deployment for releasing new application versions. Use the CodeDeployDefault.OneAtAtime as the deployment configuration and associate the Auto Scaling group with the deployment group. Configure AWS CodeDeploy to terminate all EC2 instances in the original Auto Scaling group and use the AllowTraffic hook within the appspec.yml configuration file to purge the temporary files is incorrect because you should use blue/green deployment instead of in-place. In addition, the AllowTraffic event just allows the incoming traffic to the instances after a deployment. This event is reserved for the CodeDeploy agent and cannot be used to run scripts.

The option that says: Launch an Application Load Balancer and use a blue/green deployment for releasing new application versions. Associate the Auto Scaling group and the Application Load Balancer target group with the deployment group. In CodeDeploy, use the CodeDeployDefault.HalfAtAtime as the deployment configuration and configure it to terminate the original instances in the Auto Scaling group after deployment. Use the BlockTraffic hook within appspec.yml to purge the temporary files is incorrect because the BlockTraffic event is reserved for the CodeDeploy agent and cannot be used to run custom scripts such as deleting the temporary files.

The option that says: Launch an Application Load Balancer and use an in-place deployment for releasing new application versions. Associate the Auto Scaling group and Application Load Balancer target group with the deployment group. In CodeDeploy, use the CodeDeployDefault.AllatOnce as a deployment configuration and add a configuration to terminate the original instances in the Auto Scaling group after the deployment. Use the BlockTraffic hook within appsec.yml to purge the temporary files is incorrect because you should use a blue/green deployment instead of in-place. It is also incorrect to use the CodeDeployDefault AllatOnce deployment configuration as this attempts to deploy the application revision to as many instances as possible at once.

References:

https://docs.aws.amazon.com/codedeploy/latest/userguide/troubleshooting-deployments.html#troubleshooting-deployments-lifecycle-event-failures

https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file-structure-hooks.html#appspec-hooks-server

Check out this AWS CodeDeploy Cheat Sheet:

https://tutorialsdojo.com/aws-cheat-sheet-aws-codedeploy/

Q24) A software development company is using AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, and AWS CodePipeline for its CI/CD process. To further improve their systems, they need to implement a solution that automatically detects and reacts to changes in the state of their deployments in AWS CodeDeploy. Any changes must be rolled back automatically if the deployment process fails, and a notification must be sent to the DevOps Team's Slack channel for easy monitoring.
Which of the following is the MOST suitable configuration that you should implement to satisfy this requirement?

Monitor the API calls in the CodeDeploy project using AWS CloudTrail. Send a message to the DevOps Team's Slack Channel when the PutLifecycleEventHookExecutionStatus API call has been detected. Rollback the changes by using the AWS CLI.

Set up a CloudWatch Alarm that tracks the CloudWatch metrics of the CodeDeploy project. Configure the CloudWatch Alarm to automatically send out a message to the DevOps Team's Slack Channel when the deployment fails. Configure AWS CodeDeploy to use the Roll back when alarm thresholds are met setting.

Set up a CloudWatch Events rule to monitor AWS CodeDeploy operations with a Lambda function as a target. Configure the rule to send out a message to the DevOps Team's Slack Channel in the event that the deployment fails. Configure AWS CodeDeploy to use the Roll back when a deployment fails setting.

Explanation:-You can monitor CodeDeploy deployments using the following CloudWatch tools: Amazon CloudWatch Events, CloudWatch alarms,

and Amazon CloudWatch Logs.
Reviewing the logs created by the CodeDeploy agent and deployments can help you troubleshoot the causes of deployment failures. As an alternative to reviewing CodeDeploy logs on one instance at a time, you can use CloudWatch Logs to monitor all logs in a central location.
You can use Amazon CloudWatch Events to detect and react to changes in the state of an instance or a deployment (an "event") in your CodeDeploy operations. Then, based on rules you create, CloudWatch Events will invoke one or more target actions when a deployment or instance enters the state you specify in a rule. Depending on the type of state change, you might want to send notifications, capture state information, take corrective action, initiate events, or take other actions.
You can select the following types of targets when using CloudWatch Events as part of your CodeDeploy operations:
- AWS Lambda functions
- Kinesis streams
- Amazon SQS queues
- Built-in targets (CloudWatch alarm actions)
- Amazon SNS topics
The following are some use cases:
- Use a Lambda function to pass a notification to a Slack channel whenever deployments fail.
- Push data about deployments or instances to a Kinesis stream to support comprehensive, real-time status monitoring.
- Use CloudWatch alarm actions to automatically stop, terminate, reboot, or recover Amazon EC2 instances when a deployment or instance event you specify occurs.
Hence, the correct answer is: Set up a CloudWatch Events rule to monitor AWS CodeDeploy operations with a Lambda function as a target. Configure the rule to send out a message to the DevOps Team's Slack Channel in the event that the deployment fails. Configure AWS CodeDeploy to use the Roll back when a deployment fails setting.
The option that says: Set up a CloudWatch Alarm that tracks the CloudWatch metrics of the CodeDeploy project. Configure the CloudWatch Alarm to automatically send out a message to the DevOps Team's Slack Channel when the deployment fails. Configure AWS CodeDeploy to use the Roll back when alarm thresholds are met setting is incorrect because CloudWatch Alarm can't directly send a message to a Slack Channel. You have to use a CloudWatch Events with an associated Lambda function to notify the DevOps Team via Slack.
The option that says: Configure a CodeDeploy agent to send a notification to the DevOps Team's Slack Channel when the deployment fails. Configure AWS CodeDeploy to automatically roll back whenever the deployment is not successful is incorrect because a CodeDeploy agent is primarily used for deployment and not for sending custom messages to non-AWS resources such as a Slack Channel.
The option that says: Monitor the API calls in the CodeDeploy project using AWS CloudTrail. Send a message to the DevOps Team's Slack Channel when the PutLifecycleEventHookExecutionStatus API call has been detected. Rollback the changes by using the AWS CLI is incorrect because this API simply sets the result of a Lambda validation function. This is not a suitable solution since invoking various API calls is not necessary at all. You simply have to integrate a CloudWatch Events rule with an associated Lambda function to your CodeDeploy project in order to meet the specified requirement.
References:
https://docs.aws.amazon.com/codedeploy/latest/userguide/monitoring-cloudwatch-events.html
https://docs.aws.amazon.com/codedeploy/latest/userguide/monitoring-cloudwatch.html
Check out this AWS CodeDeploy Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-codedeploy/
● Configure a CodeDeploy agent to send notification to the DevOps Team's Slack Channel when the deployment fails. Configure AWS CodeDeploy to automatically roll back whenever the deployment is not successful.

Q25) A leading digital payments company is using AWS to host its suite of web applications which uses external APIs for credit and debit transactions. The current architecture is using CloudTrail with several trails to log all API actions. Each trail is protected with an IAM policy to restrict access from unauthorized users. In order to maintain the system's PCI DSS compliance, a solution must be implemented that allows them to trace the integrity of each file and prevent the files from being tampered. Which of the following is the MOST suitable solution with the LEAST amount of effort to implement?

● In the Amazon S3 bucket of the trail, enable the log file integrity feature that will automatically generate a digest file for every log file that CloudTrail delivers. Grant the IT Security team full access to download the file integrity logs stored in the S3 bucket via an IAM policy.
● Use AWS Systems Manager State Manager to directly enable the log file integrity feature in CloudTrail. This will automatically generate a digest file for every log file that CloudTrail delivers. Verify the integrity of the delivered CloudTrail files using the generated digest files.
● Use AWS Config to directly enable the log file integrity feature in CloudTrail. This will automatically generate a digest file for every log file that CloudTrail delivers. Verify the integrity of the delivered CloudTrail files using the generated digest files.
✅ In AWS CloudTrail, enable the log file integrity feature on the trail that will automatically generate a digest file for every log file that CloudTrail delivers. Verify the integrity of the delivered CloudTrail files using the generated digest files.
Explanation:-To determine whether a log file was modified, deleted, or unchanged after CloudTrail delivered it, you can use CloudTrail log file integrity validation. This feature is built using industry-standard algorithms: SHA-256 for hashing and SHA-256 with RSA for digital signing. This makes it computationally infeasible to modify, delete or forge CloudTrail log files without detection. You can use the AWS CLI to validate the files in the location where CloudTrail delivered them.
Validated log files are invaluable in security and forensic investigations. For example, a validated log file enables you to assert positively that the log file itself has not changed, or that particular user credentials performed specific API activity. The CloudTrail log file integrity validation process also lets you know if a log file has been deleted or changed, or assert positively that no log files were delivered to your account during a given period of time.
When you enable log file integrity validation, CloudTrail creates a hash for every log file that it delivers. Every hour, CloudTrail also creates and delivers a file that references the log files for the last hour and contains a hash of each. This file is called a digest file. CloudTrail signs each digest file using the private key of a public and private key pair. After delivery, you can use the public key to validate the digest file. CloudTrail uses different key pairs for each AWS region.
The digest files are delivered to the same Amazon S3 bucket associated with your trail as your CloudTrail log files. If your log files are delivered from all regions or from multiple accounts into a single Amazon S3 bucket, CloudTrail will deliver the digest files from those regions and accounts into the same bucket.
The digest files are put into a folder separate from the log files. This separation of digest files and log files enables you to enforce granular security policies and permits existing log processing solutions to continue to operate without modification. Each digest file also contains the digital signature of the previous digest file if one exists. The signature for the current digest file is in the metadata properties of the digest file Amazon S3 object.
Hence, the correct answer is: In AWS CloudTrail, enable the log file integrity feature on the trail that will automatically generate a digest file for every log file that CloudTrail delivers. Verify the integrity of the delivered CloudTrail files using the generated digest files.
The option that says: Use AWS Systems Manager State Manager to directly enable the log file integrity feature in CloudTrail. This will automatically generate a digest file for every log file that CloudTrail delivers. Verify the integrity of the delivered CloudTrail files using the generated digest files is incorrect because there is no direct way that you can enable the log file integrity feature in CloudTrail using AWS Systems Manager State Manager. This must be manually enabled using the Console or the AWS CLI.
The option that says: In the Amazon S3 bucket of the trail, enable the log file integrity feature that will automatically generate a digest file for every

log file that CloudTrail delivers. Grant the IT Security team full access to download the file integrity logs stored in the S3 bucket via an IAM policy is incorrect because the log file integrity feature must be configured in the trail itself and not on the S3 bucket.
The option that says: Use AWS Config to directly enable the log file integrity feature in CloudTrail. This will automatically generate a digest file for every log file that CloudTrail delivers. Verify the integrity of the delivered CloudTrail files using the generated digest files is incorrect because there is no direct way that you can enable the log file integrity feature in CloudTrail using AWS Config. You have to manually enable it.
References:
https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-log-file-validation-intro.html
https://aws.amazon.com/blogs/aws/aws-cloudtrail-update-sse-kms-encryption-log-file-integrity-verification/
Check out this AWS CloudTrail Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-cloudtrail/

Q26) You have developed a web portal for your analytics team that they can use to query and visualize patterns of the collected data from the customer's click behavior. The portal relies on data processed by an Amazon EMR cluster which consists of an Auto Scaling group of EC2 instances that you've provisioned on AWS. After 3 months of operation, you checked AWS Trusted Advisor, which recommends that you terminate some EMR instances that were underutilized. Upon checking the EMR cluster, you forgot to define scale-down parameters which contributed to high billing costs. To prevent this from happening in the future, you want to be notified for AWS Trusted Advisor recommendation as frequently as possible.
Which of the following solutions can help you achieve this requirement? (Select THREE)

Enable the built-in Trusted Advisor notification feature to automatically receive notification emails which includes the summary of savings estimates along with Trusted Advisor check results.
Write a Lambda function that runs daily to refresh AWS Trusted Advisor via API and then publish a message to an SNS Topic to notify the subscribers based on the results.
Explanation:-You can use Amazon CloudWatch Events to detect and react to changes in the status of Trusted Advisor checks. Then, based on the rules that you create, CloudWatch Events invokes one or more target actions when a check status changes to the value you specify in a rule. Depending on the type of status change, you might want to send notifications, capture status information, take corrective action, initiate events, or take other actions. You can select the following types of targets when using CloudWatch Events as a part of your Trusted Advisor workflow:
AWS Lambda functions
Amazon Kinesis streams
Amazon Simple Queue Service queues
Built-in targets (CloudWatch alarm actions)
Amazon Simple Notification Service topics
The following are some use cases:
Use a Lambda function to pass a notification to a Slack channel when check status changes.
Push data about checks to a Kinesis stream to support comprehensive, real-time status monitoring.
You can also configure CloudWatch Logs to send a notification whenever an alarm is triggered. Doing so enables you to respond quickly based on the logs collected by CloudWatch Logs. CloudWatch uses Amazon Simple Notification Service (SNS) to send an email.
Hence, the correct answers are:
- Write a Lambda function that runs daily to refresh AWS Trusted Advisor via API and then publish a message to an SNS Topic to notify the subscribers based on the results.
- Write a Lambda function that runs daily to refresh AWS Trusted Advisor changes via API and send results to CloudWatch Logs. Create a CloudWatch Log metric and have it send an alarm notification when it is triggered.
- Use CloudWatch Events to monitor Trusted Advisor checks and set a trigger to send an email using SNS to notify you about the results of the check.
The option that says: Use CloudWatch Events to monitor Trusted Advisor checks and set a trigger to send an email using SES to notify you about the results of the check is just partially correct since the integration for sending email notification should use SNS, not SES.
The option that says: Enable the built-in CloudWatch Events notification feature that scans changes in the Trusted Advisor check results. Send notifications automatically to the account owner is incorrect because there is no automatic checks on CloudWatch Events for AWS Trusted Advisor in the first place. You need to manually configure those rules and have them trigger notifications to you.
The option that says: Enable the built-in Trusted Advisor notification feature to automatically receive notification emails which include the summary of savings estimates along with Trusted Advisor check results is incorrect because although this is a viable solution, the notification will be sent on a weekly basis only, which can already incur a lot of cost by the time it notifies.
References:
https://docs.aws.amazon.com/awssupport/latest/user/cloudwatch-events-ta.html
https://docs.aws.amazon.com/lambda/latest/dg/with-scheduled-events.html
https://docs.aws.amazon.com/lambda/latest/dg/tutorial-scheduled-events-schedule-expressions.html
Enable the built-in CloudWatch Events notification feature that scans changes in the Trusted Advisor check result. Send notifications automatically to the account owner.
Use CloudWatch Events to monitor Trusted Advisor checks and set a trigger to send an email using SNS to notify you about the results of the check.
Explanation:-You can use Amazon CloudWatch Events to detect and react to changes in the status of Trusted Advisor checks. Then, based on the rules that you create, CloudWatch Events invokes one or more target actions when a check status changes to the value you specify in a rule. Depending on the type of status change, you might want to send notifications, capture status information, take corrective action, initiate events, or take other actions. You can select the following types of targets when using CloudWatch Events as a part of your Trusted Advisor workflow:
AWS Lambda functions
Amazon Kinesis streams
Amazon Simple Queue Service queues
Built-in targets (CloudWatch alarm actions)
Amazon Simple Notification Service topics
The following are some use cases:
Use a Lambda function to pass a notification to a Slack channel when check status changes.
Push data about checks to a Kinesis stream to support comprehensive, real-time status monitoring.
You can also configure CloudWatch Logs to send a notification whenever an alarm is triggered. Doing so enables you to respond quickly based on the logs collected by CloudWatch Logs. CloudWatch uses Amazon Simple Notification Service (SNS) to send an email.
Hence, the correct answers are:
- Write a Lambda function that runs daily to refresh AWS Trusted Advisor via API and then publish a message to an SNS Topic to notify the subscribers based on the results.
- Write a Lambda function that runs daily to refresh AWS Trusted Advisor changes via API and send results to CloudWatch Logs. Create a CloudWatch Log metric and have it send an alarm notification when it is triggered.
- Use CloudWatch Events to monitor Trusted Advisor checks and set a trigger to send an email using SNS to notify you about the results of the

check.
The option that says: Use CloudWatch Events to monitor Trusted Advisor checks and set a trigger to send an email using SES to notify you about the results of the check is just partially correct since the integration for sending email notification should use SNS, not SES.
The option that says: Enable the built-in CloudWatch Events notification feature that scans changes in the Trusted Advisor check results. Send notifications automatically to the account owner is incorrect because there is no automatic checks on CloudWatch Events for AWS Trusted Advisor in the first place. You need to manually configure those rules and have them trigger notifications to you.
The option that says: Enable the built-in Trusted Advisor notification feature to automatically receive notification emails which include the summary of savings estimates along with Trusted Advisor check results is incorrect because although this is a viable solution, the notification will be sent on a weekly basis only, which can already incur a lot of cost by the time it notifies.
References:
https://docs.aws.amazon.com/awssupport/latest/user/cloudwatch-events-ta.html
https://docs.aws.amazon.com/lambda/latest/dg/with-scheduled-events.html
https://docs.aws.amazon.com/lambda/latest/dg/tutorial-scheduled-events-schedule-expressions.html
● Use CloudWatch Events to monitor Trusted Advisor checks and set a trigger to send an email using SES to notify you about the results of the check.
✅ Write a Lambda function that runs daily to refresh AWS Trusted Advisor changes via API and send results to CloudWatch Logs. Create a CloudWatch Log metric and have it send an alarm notification when it is triggered.
Explanation:-You can use Amazon CloudWatch Events to detect and react to changes in the status of Trusted Advisor checks. Then, based on the rules that you create, CloudWatch Events invokes one or more target actions when a check status changes to the value you specify in a rule. Depending on the type of status change, you might want to send notifications, capture status information, take corrective action, initiate events, or take other actions. You can select the following types of targets when using CloudWatch Events as a part of your Trusted Advisor workflow:
AWS Lambda functions
Amazon Kinesis streams
Amazon Simple Queue Service queues
Built-in targets (CloudWatch alarm actions)
Amazon Simple Notification Service topics
The following are some use cases:
Use a Lambda function to pass a notification to a Slack channel when check status changes.
Push data about checks to a Kinesis stream to support comprehensive, real-time status monitoring.
You can also configure CloudWatch Logs to send a notification whenever an alarm is triggered. Doing so enables you to respond quickly based on the logs collected by CloudWatch Logs. CloudWatch uses Amazon Simple Notification Service (SNS) to send an email.
Hence, the correct answers are:
- Write a Lambda function that runs daily to refresh AWS Trusted Advisor via API and then publish a message to an SNS Topic to notify the subscribers based on the results.
- Write a Lambda function that runs daily to refresh AWS Trusted Advisor changes via API and send results to CloudWatch Logs. Create a CloudWatch Log metric and have it send an alarm notification when it is triggered.
- Use CloudWatch Events to monitor Trusted Advisor checks and set a trigger to send an email using SNS to notify you about the results of the check.
The option that says: Use CloudWatch Events to monitor Trusted Advisor checks and set a trigger to send an email using SES to notify you about the results of the check is just partially correct since the integration for sending email notification should use SNS, not SES.
The option that says: Enable the built-in CloudWatch Events notification feature that scans changes in the Trusted Advisor check results. Send notifications automatically to the account owner is incorrect because there is no automatic checks on CloudWatch Events for AWS Trusted Advisor in the first place. You need to manually configure those rules and have them trigger notifications to you.
The option that says: Enable the built-in Trusted Advisor notification feature to automatically receive notification emails which include the summary of savings estimates along with Trusted Advisor check results is incorrect because although this is a viable solution, the notification will be sent on a weekly basis only, which can already incur a lot of cost by the time it notifies.
References:
https://docs.aws.amazon.com/awssupport/latest/user/cloudwatch-events-ta.html
https://docs.aws.amazon.com/lambda/latest/dg/with-scheduled-events.html
https://docs.aws.amazon.com/lambda/latest/dg/tutorial-scheduled-events-schedule-expressions.html

---

Q27) The company is re-architecting its monolithic system to a serverless application in AWS to save on cost. The deployment of the succeeding new version of the application must be initially rolled out to a small number of users first for testing before the full release. If the post-hook tests fail, the system should automatically roll back the deployment. The DevOps Engineer was assigned to design an efficient deployment setup that mitigates any unnecessary outage that impacts their production environment.
As a DevOps Engineer, how should you satisfy this requirement? (Select TWO)

● Launch an Application Load Balancer with an Amazon API Gateway private integration. Attach two target groups to the load balancer wherein the first target group points to the current version and the second target group to the new version. Route 20% of the incoming traffic to the new version. Once the new version becomes stable, detach the first target group from the load balancer.
● Create a new record in Route 53 with a Failover routing policy. Configure the primary record to route 20% of incoming traffic to the new version and set the secondary record to route the rest of the traffic to the current version. Once the new version stabilizes, update the primary record to route all traffic to the new version.
● Launch a Network Load Balancer with an Amazon API Gateway private integration. Attach two target groups to the load balancer. Configure the first target group with the current version and the second target group with the new version. Configure the load balancer to route 20% of the incoming traffic to the new version and once it becomes stable, detach the first target group from the load balancer.
✅ Set up one AWS Lambda Function Alias that points to both the current and new versions. Route 20% of incoming traffic to the new version and once it is considered stable, update the alias to route all traffic to the new version.
Explanation:-You can create one or more aliases for your AWS Lambda function. A Lambda alias is like a pointer to a specific Lambda function version. Each alias has a unique ARN. An alias can only point to a function version, not to another alias. You can update an alias to point to a new version of the function. Event sources such as Amazon S3 invoke your Lambda function. These event sources maintain a mapping that identifies the function to invoke when events occur. If you specify a Lambda function alias in the mapping configuration, you don't need to update the mapping when the function version changes. In a resource policy, you can grant permissions for event sources to use your Lambda function. If you specify an alias ARN in the policy, you don't need to update the policy when the function version changes.
Use routing configuration on an alias to send a portion of traffic to a second function version. For example, you can reduce the risk of deploying a new version by configuring the alias to send most of the traffic to the existing version, and only a small percentage of traffic to the new version. You can point an alias to a maximum of two Lambda function versions.
In API Gateway, you create a canary release deployment when deploying the API with canary settings as an additional input to the deployment creation operation.

You can also create a canary release deployment from an existing non-canary deployment by making a stage:update request to add the canary settings on the stage.
When creating a non-canary release deployment, you can specify a non-existing stage name. API Gateway creates one if the specified stage does not exist. However, you cannot specify any non-existing stage name when creating a canary release deployment. You will get an error and API Gateway will not create any canary release deployment.
Hence, the correct answers are:
- Set up one AWS Lambda Function Alias that points to both the current and new versions. Route 20% of incoming traffic to the new version and once it is considered stable, update the alias to route all traffic to the new version.
- Set up a canary deployment in Amazon API Gateway that routes 20% of the incoming traffic to the canary release. Promote the canary release to production once the initial tests have passed.
The option that says: Launch an Application Load Balancer with an Amazon API Gateway private integration. Attach two target groups to the load balancer wherein the first target group points to the current version and the second target group to the new version. Route 20% of the incoming traffic to the new version. Once the new version becomes stable, detach the first target group from the load balancer is incorrect because you can only set up an Amazon API Gateway private integration with a Network Load Balancer and not an Application Load Balancer.
The option that says: Launch a Network Load Balancer with an Amazon API Gateway private integration. Attach two target groups to the load balancer. Configure the first target group with the current version and the second target group with the new version. Configure the load balancer to route 20% of the incoming traffic to the new version and once it becomes stable, detach the first target group from the load balancer is incorrect because the Network Load Balancer does not support weighted target groups, unlike the Application Load Balancer.
The option that says: Create a new record in Route 53 with a Failover routing policy. Configure the primary record to route 20% of incoming traffic to the new version and set the secondary record to route the rest of the traffic to the current version. Once the new version stabilizes, update the primary record to route all traffic to the new version is incorrect because the failover routing policy simply lets you route traffic to a resource when the resource is healthy, or to a different resource when the first resource is unhealthy. This type of routing is not an appropriate setup. A better solution is to use Canary deployment release in API Gateway to deploy the serverless application.
References:
https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-private-integration.html
https://docs.aws.amazon.com/apigateway/latest/developerguide/canary-release.html
Check out this Amazon API Gateway Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-api-gateway/
✅ Set up a canary deployment in Amazon API Gateway that routes 20% of the incoming traffic to the canary release. Promote the canary release to production once the initial tests have passed.
Explanation:-You can create one or more aliases for your AWS Lambda function. A Lambda alias is like a pointer to a specific Lambda function version. Each alias has a unique ARN. An alias can only point to a function version, not to another alias. You can update an alias to point to a new version of the function. Event sources such as Amazon S3 invoke your Lambda function. These event sources maintain a mapping that identifies the function to invoke when events occur. If you specify a Lambda function alias in the mapping configuration, you don't need to update the mapping when the function version changes. In a resource policy, you can grant permissions for event sources to use your Lambda function. If you specify an alias ARN in the policy, you don't need to update the policy when the function version changes.
Use routing configuration on an alias to send a portion of traffic to a second function version. For example, you can reduce the risk of deploying a new version by configuring the alias to send most of the traffic to the existing version, and only a small percentage of traffic to the new version. You can point an alias to a maximum of two Lambda function versions.
In API Gateway, you create a canary release deployment when deploying the API with canary settings as an additional input to the deployment creation operation.
You can also create a canary release deployment from an existing non-canary deployment by making a stage:update request to add the canary settings on the stage.
When creating a non-canary release deployment, you can specify a non-existing stage name. API Gateway creates one if the specified stage does not exist. However, you cannot specify any non-existing stage name when creating a canary release deployment. You will get an error and API Gateway will not create any canary release deployment.
Hence, the correct answers are:
- Set up one AWS Lambda Function Alias that points to both the current and new versions. Route 20% of incoming traffic to the new version and once it is considered stable, update the alias to route all traffic to the new version.
- Set up a canary deployment in Amazon API Gateway that routes 20% of the incoming traffic to the canary release. Promote the canary release to production once the initial tests have passed.
The option that says: Launch an Application Load Balancer with an Amazon API Gateway private integration. Attach two target groups to the load balancer wherein the first target group points to the current version and the second target group to the new version. Route 20% of the incoming traffic to the new version. Once the new version becomes stable, detach the first target group from the load balancer is incorrect because you can only set up an Amazon API Gateway private integration with a Network Load Balancer and not an Application Load Balancer.
The option that says: Launch a Network Load Balancer with an Amazon API Gateway private integration. Attach two target groups to the load balancer. Configure the first target group with the current version and the second target group with the new version. Configure the load balancer to route 20% of the incoming traffic to the new version and once it becomes stable, detach the first target group from the load balancer is incorrect because the Network Load Balancer does not support weighted target groups, unlike the Application Load Balancer.
The option that says: Create a new record in Route 53 with a Failover routing policy. Configure the primary record to route 20% of incoming traffic to the new version and set the secondary record to route the rest of the traffic to the current version. Once the new version stabilizes, update the primary record to route all traffic to the new version is incorrect because the failover routing policy simply lets you route traffic to a resource when the resource is healthy, or to a different resource when the first resource is unhealthy. This type of routing is not an appropriate setup. A better solution is to use Canary deployment release in API Gateway to deploy the serverless application.
References:
https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-private-integration.html
https://docs.aws.amazon.com/apigateway/latest/developerguide/canary-release.html
Check out this Amazon API Gateway Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-api-gateway/

Q28) You are instructed to set up a configuration management for all of your infrastructure in AWS. To comply with the company's strict security policies, the solution should provide a near real-time dashboard of the compliance posture of your systems with a feature to detect violations.
Which solution would be able to meet the above requirements?

- Tag all of your resources and use Trusted Advisor to monitor both the compliant and non-compliant resources. Use the AWS Management Console to monitor the status of your compliance posture.
- Use Amazon inspector to monitor the compliance posture of your systems and store the reports to Amazon CloudWatch Logs. Use a CloudWatch dashboard with a custom metric filter to monitor and view all of the specific compliance requirements.
- Use AWS Service Catalog to create the required resource configurations for your compliance posture. Monitor the compliance and violations of

all of your cloud resources using a custom CloudWatch dashboard with an integrated Amazon SNS to send the notifications.

✅ Use AWS Config to record all configuration changes and store the data reports to Amazon S3. Use Amazon QuickSight to analyze the dataset.

Explanation:-AWS Config provides you a visual dashboard to help you quickly spot non-compliant resources and take appropriate action. IT Administrators, Security Experts, and Compliance Officers can see a shared view of your AWS resources compliance posture.

When you run your applications on AWS, you usually use AWS resources, which you must create and manage collectively. As the demand for your application keeps growing, so does your need to keep track of your AWS resources.

To exercise better governance over your resource configurations and to detect resource misconfigurations, you need fine-grained visibility into what resources exist and how these resources are configured at any time. You can use AWS Config to notify you whenever resources are created, modified, or deleted without having to monitor these changes by polling the calls made to each resource.

You can use AWS Config rules to evaluate the configuration settings of your AWS resources. When AWS Config detects that a resource violates the conditions in one of your rules, AWS Config flags the resource as noncompliant and sends a notification. AWS Config continuously evaluates your resources as they are created, changed, or deleted.

Hence, the correct answer is: Use AWS Config to record all configuration changes and store the data reports to Amazon S3. Use Amazon QuickSight to analyze the dataset.

The option that says: Use AWS Service Catalog to create the required resource configurations for your compliance posture. Monitor the compliance and violations of all of your cloud resources using a custom CloudWatch dashboard with an integrated Amazon SNS to send the notifications is incorrect because although AWS Service Catalog can be used for resource configuration, it is not fully capable of detecting violations of your AWS configuration rules.

The option that says: Tag all of your resources and use Trusted Advisor to monitor both the compliant and non-compliant resources. Use the AWS Management Console to monitor the status of your compliance posture is incorrect because the Trusted Advisor service is not suitable for configuration management and automatic violation detection. You should use AWS Config instead.

The option that says: Use Amazon inspector to monitor the compliance posture of your systems and store the reports to Amazon CloudWatch Logs. Use a CloudWatch dashboard with a custom metric filter to monitor and view all of the specific compliance requirements is incorrect because the Amazon Inspector service is primarily used to help you check for unintended network accessibility of your Amazon EC2 instances and for vulnerabilities on those EC2 instances.

References:

https://docs.aws.amazon.com/config/latest/developerguide/WhatIsConfig.html

https://docs.aws.amazon.com/quicksight/latest/user/QS-compliance.html

https://aws.amazon.com/config/features/

Check out this AWS Config Cheat Sheet:

https://tutorialsdojo.com/aws-cheat-sheet-aws-config/

Q29) A leading telecommunications company is using CloudFormation templates to deploy enterprise applications to their production, staging, and development environments in AWS. Their current process involves manual changes to their CloudFormation templates in order to specify the configuration variables and static attributes for each environment. The DevOps Engineer was tasked to set up automated deployments using AWS CodePipeline and ensure that the CloudFormation template is reusable across multiple pipelines.
How should the DevOps Engineer satisfy this requirement?

⚫ Launch a new pipeline using AWS CodePipeline for each environment with multiple stages for each application. Trigger the CloudFormation deployments using a Lambda function to dynamically modify the UserData of the EC2 instances that were launched in each environment.

⚫ Set up a Lambda-backed custom resource in the CloudFormation templates. Configure the custom resource to monitor the status of the pipeline in AWS CodePipeline in order to detect which environment was launched. Use the cfn-init helper script to modify the launch configuration of each application stack based on its environment.

✅ Launch a new pipeline using AWS CodePipeline that has multiple stages for each environment and configure it to use input parameters. Switch the associated UserData of the EC2 instances to match the environment where the application stack is being launched using CloudFormation mappings. Specify parameter overrides for AWS CloudFormation actions.

Explanation:-Continuous delivery is a release practice in which code changes are automatically built, tested, and prepared for release to production. With AWS CloudFormation and CodePipeline, you can use continuous delivery to automatically build and test changes to your AWS CloudFormation templates before promoting them to production stacks. This release process lets you rapidly and reliably make changes to your AWS infrastructure.

For example, you can create a workflow that automatically builds a test stack when you submit an updated template to a code repository. After AWS CloudFormation builds the test stack, you can test it and then decide whether to push the changes to a production stack.

You can use CodePipeline to build a continuous delivery workflow by building a pipeline for AWS CloudFormation stacks. CodePipeline has built-in integration with AWS CloudFormation, so you can specify AWS CloudFormation-specific actions, such as creating, updating, or deleting a stack, within a pipeline.In a CodePipeline stage, you can specify parameter overrides for AWS CloudFormation actions. Parameter overrides let you specify template parameter values that override values in a template configuration file. AWS CloudFormation provides functions to help you specify dynamic values (values that are unknown until the pipeline runs).

You can set the Fn::GetArtifactAtt function which retrieves the value of an attribute from an input artifact, such as the S3 bucket name where the artifact is stored. You can use this function to specify attributes of an artifact, such as its filename or S3 bucket name, that can be used in the pipeline.

Hence, the correct answer is: Launch a new pipeline using AWS CodePipeline that has multiple stages for each environment and configure it to use input parameters. Switch the associated UserData of the EC2 instances to match the environment where the application stack is being launched using CloudFormation mappings. Specify parameter overrides for AWS CloudFormation actions.

The option that says: Set up a Lambda-backed custom resource in the CloudFormation templates. Configure the custom resource to monitor the status of the pipeline in AWS CodePipeline in order to detect which environment was launched. Use the cfn-init helper script to modify the launch configuration of each application stack based on its environment is incorrect because monitoring the pipeline using a custom resource in CloudFormation entails a lot of administrative overhead. A better solution would be to use input parameters or parameter overrides for AWS CloudFormation actions.

The option that says: Launch a new pipeline using AWS CodePipeline for each environment with multiple stages for each application. Trigger the CloudFormation deployments using a Lambda function to dynamically modify the UserData of the EC2 instances that were launched in each environment is incorrect because using a Lambda function to modify the UserData of the already running EC2 instances is not a suitable solution. The parameters should have been dynamically populated and set before the resources were launched by using parameter overrides.

The option that says: Manually configure the CloudFormation templates to use input parameters. Add a configuration that whenever the CloudFormation stack is updated, it will dynamically modify the LaunchConfiguration and UserData sections of the EC2 instances is incorrect because although using input parameters is helpful in this scenario, you should still integrate CloudFormation and CodePipeline in order to properly map the configuration files for each environment.

References:

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/mappings-section-structure.html

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/continuous-delivery-codepipeline.html

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/continuous-delivery-codepipeline-parameter-override-functions.html
Check out this AWS CloudFormation Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-cloudformation/
Elastic Beanstalk vs CloudFormation vs OpsWorks vs CodeDeploy:
https://tutorialsdojo.com/aws-cheat-sheet-elastic-beanstalk-vs-cloudformation-vs-opsworks-vs-codedeploy/

- Manually configure the CloudFormation templates to use input parameters. Add a configuration that whenever the CloudFormation stack is updated, it will dynamically modify the LaunchConfiguration and UserData sections of the EC2 instances.

Q30) An online data analytics application is launched to 12 On-Demand EC2 instances across three Availability Zones using a golden AMI in AWS. Each instance has only 10% utilization after business hours but increases to 30% utilization during peak hours. There are also some third-party applications that uses the application from all over the globe with no specific schedule. In the morning, there is always a sudden CPU utilization increase on the EC2 instances due to the number of users logging in to use the application. However, its CPU utilization usually stabilizes after a few hours. A DevOps Engineer has been instructed to reduce costs and improve the overall reliability of the system.
Which among the following options provides the MOST suitable solution in this scenario?

- Set up two AWS Config rules and two Lambda functions. Configure each rule to invoke a Lambda function and regularly run before and after the peak hours. The first function should stop nine instances after the peak hours end while the second function should restart the nine instances before the business day begins.
- Launch a group of Scheduled Reserved Instances that regularly run before and after the peak hours. Integrate CloudWatch Events and AWS Lambda to regularly stop nine instances after the peak hours every day and restart the nine instances before the business day begins.
- Set up two CloudWatch Events rules and two Lambda functions. Configure each CloudWatch Event rule to invoke a Lambda function and regularly run before and after the peak hours. The first function should stop nine instances after the peak hours end while the second function should restart the nine instances before the business day begins.
- ✅ Set up an Auto Scaling group using the golden AMI with a scaling action based on the CPU Utilization average. Configure a scheduled action for the group to adjust the minimum number of Amazon EC2 instances to three after business hours end, and reset to six before business hours begin.

Explanation:-Scaling based on a schedule allows you to set your own scaling schedule for predictable load changes. For example, every week the traffic to your web application starts to increase on Wednesday, remains high on Thursday, and starts to decrease on Friday. You can plan your scaling actions based on the predictable traffic patterns of your web application. Scaling actions are performed automatically as a function of time and date.
To configure your Auto Scaling group to scale based on a schedule, you create a scheduled action. The scheduled action tells Amazon EC2 Auto Scaling to perform a scaling action at specified times. To create a scheduled scaling action, you specify the start time when the scaling action should take effect, and the new minimum, maximum, and desired sizes for the scaling action. At the specified time, Amazon EC2 Auto Scaling updates the group with the values for minimum, maximum, and desired size specified by the scaling action.
You can create scheduled actions for scaling one time only or for scaling on a recurring schedule.
Hence, the correct answer is: Set up an Auto Scaling group using the golden AMI with a scaling action based on the CPU Utilization average. Configure a scheduled action for the group to adjust the minimum number of Amazon EC2 instances to three after business hours end, and reset to six before business hours begin.
The option that says: Set up two CloudWatch Events rules and two Lambda functions. Configure each CloudWatch Event rule to invoke a Lambda function and regularly run before and after the peak hours. The first function should stop nine instances after the peak hours end while the second function should restart the nine instances before the business day begins is incorrect because you can simply configure a scheduled action for the Auto Scaling group to adjust the minimum number of the available EC2 instances without using CloudWatch Events or a Lambda function.
The option that says: Set up two AWS Config rules and two Lambda functions. Configure each rule to invoke a Lambda function and regularly run before and after the peak hours. The first function should stop nine instances after the peak hours end while the second function should restart the nine instances before the business day begins is incorrect because using AWS Config is not an appropriate service to use in this scenario. A better solution is to configure a scheduled action in the Auto Scaling group.
The option that says: Launch a group of Scheduled Reserved Instances that regularly run before and after the peak hours. Integrate CloudWatch Events and AWS Lambda to regularly stop nine instances after the peak hours every day and restart the nine instances before the business day begins is incorrect because although your operating costs will be decreased by using Scheduled Reserved Instances, this set up is still not appropriate since the traffic to the portal is not entirely predictable. Take note that there are third-party applications that use the application from all over the globe with no specific schedule. Hence, the use of Scheduled Reserved Instances is not recommended.
References:
https://docs.aws.amazon.com/autoscaling/ec2/userguide/schedule_time.html
https://docs.aws.amazon.com/autoscaling/ec2/userguide/scaling_plan.html
Check out this AWS Auto Scaling Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-auto-scaling/

Q31) A company has a suite of applications which is hosted in AWS and each app has its own AMI. Currently, a new AMI must be manually created and deployed to the server if there is a new application version. A DevOps engineer was instructed to automate the process of generating the AMIs to streamline the company's CI/CD workflow. The ID of the newly created AMI must be stored in a centralized location where other build pipelines can programmatically access.
Which of the following is the MOST cost-effective way to accomplish this requirement with the LEAST amount of overhead?

- Use an open source machine image creation tool such as Packer then configure it with values defining how the AMI should be created. Set up a Jenkins pipeline hosted in a large EC2 instance to start the Packer when triggered to build an AMI. Store the AMI IDs to an Amazon DynamoDB table.
- Set up a brand new pipeline in AWS CodePipeline with several EC2 instances for processing. Configure it to download and save the latest operating system of the application in an Open Virtualization Format (OVF) image format and then store image to an S3 bucket. Customize the image using guestfish interactive shell or the virt-rescue shell. Convert the OVF to an AMI using the virtual machine (VM) import command and then store the AMI IDs to AWS Systems Manager Parameter Store.
- ✅ Using AWS Systems Manager, create an Automation document with values that configure how the machine image should be created. Launch a new pipeline with a custom action in AWS CodePipeline and integrate it with CloudWatch Events to execute the automation document. Build the AMI when the process is triggered. Store the AMI IDs in Systems Manager Parameter Store.

Explanation:-Systems Manager Automation simplifies common maintenance and deployment tasks of Amazon EC2 instances and other AWS resources. Automation enables you to do the following.
- Build Automation workflows to configure and manage instances and AWS resources.
- Create custom workflows or use pre-defined workflows maintained by AWS.
- Receive notifications about Automation tasks and workflows by using Amazon CloudWatch Events.
- Monitor Automation progress and execution details by using the Amazon EC2 or the AWS Systems Manager console.

Automation offers one-click automations for simplifying complex tasks such as creating golden Amazon Machines Images (AMIs), and recovering

unreachable EC2 instances. Here are some examples:
- Use the AWS-UpdateLinuxAmi and AWS-UpdateWindowsAmi documents to create golden AMIs from a source AMI. You can run custom scripts before and after updates are applied. You can also include or exclude specific packages from being installed.
- Use the AWSSupport-ExecuteEC2Rescue document to recover impaired instances. An instance can become unreachable for a variety of reasons, including network misconfigurations, RDP issues, or firewall settings. Troubleshooting and regaining access to the instance previously required dozens of manual steps before you could regain access. The AWSSupport-ExecuteEC2Rescue document lets you regain access by specifying an instance ID and clicking a button.
A Systems Manager Automation document defines the Automation workflow (the actions that Systems Manager performs on your managed instances and AWS resources). Automation includes several pre-defined Automation documents that you can use to perform common tasks like restarting one or more Amazon EC2 instances or creating an Amazon Machine Image (AMI). Documents use JavaScript Object Notation (JSON) or YAML, and they include steps and parameters that you specify. Steps run in sequential order.
Hence, the correct answer is: Using AWS Systems Manager, create an Automation document with values that configure how the machine image should be created. Launch a new pipeline with a custom action in AWS CodePipeline and integrate it with CloudWatch Events to execute the automation document. Build the AMI when the process is triggered. Store the AMI IDs in Systems Manager Parameter Store.
The option that says: Set up a brand new pipeline in AWS CodePipeline with several EC2 instances for processing. Configure it to download and save the latest operating system of the application in an Open Virtualization Format (OVF) image format and then store image to an S3 bucket. Customize the image using guestfish interactive shell or the virt-rescue shell. Convert the OVF to an AMI using the virtual machine (VM) import command and then store the AMI IDs to AWS Systems Manager Parameter Store is incorrect because manually customizing the image using an interactive shell and downloading each application image in an OVF file entails a lot of effort. It is also better to use the AWS Systems Manager Automation instead of creating a new pipeline in AWS CodePipeline.
The option that says: Using AWS CodePipeline, set up a new pipeline that will take an EBS snapshot of the EBS-backed EC2 instance which runs the latest version of each application. Launch a new EC2 instance from the generated snapshot and update the running instance using a Lambda function. Take a snapshot of the updated EC2 instance and then afterwards, convert it to an Amazon Machine Image(AMI). Store all of the AMI ID in an S3 bucket is incorrect because although you can technically generate an AMI using an EBS volume snapshot, this process is still tedious and entails a lot of configuration. Using the AWS Systems Manager Automation to generate the AMIs is a more suitable solution.
The option that says: Use an open-source machine image creation tool such as Packer then configure it with values defining how the AMI should be created. Set up a Jenkins pipeline hosted in a large EC2 instance to start the Packer when triggered to build an AMI. Store the AMI IDs to an Amazon DynamoDB table is incorrect because although this may work, this solution costs more to maintain than other options since it uses an EC2 instance and an Amazon DynamoDB table. There is also an associated overhead in configuring and using Packer for generating the AMIs as well as in preparing the Jenkins pipeline.
References:
https://aws.amazon.com/blogs/big-data/create-custom-amis-and-push-updates-to-a-running-amazon-emr-cluster-using-amazon-ec2-systems-manager/
https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-automation.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/automation-documents.html
Check out this AWS Systems Manager Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-systems-manager/
● Using AWS CodePipeline, set up a new pipeline that will take an EBS snapshot of the EBS-backed EC2 instance which runs the latest version of each application. Launch a new EC2 instance from the generated snapshot and update the running instance using a Lambda function. Take a snapshot of the updated EC2 instance and then afterwards, convert it to an Amazon Machine Image (AMI). Store all of the AMI ID in an S3 bucket.

Q32) A company has recently developed a serverless application that is composed of several Lambda functions and a DynamoDB database. For the CI/CD process, you have built a continuous deployment pipeline using AWS CodeCommit, AWS CodeBuild, and AWS CodePipeline. You have also configured the source, build, test, and deployment stages of the pipeline of the application. However, upon review, the Lead DevOps engineer asked you to improve the current pipeline configuration that you've made to mitigate the risk of failed deployments. The deployment stage should release the new application version to a small subset of users only for verification before fully releasing the change to all users.
The pipeline's deployment stage must be modified to meet this requirement. Which of the following is the MOST suitable setup that you should implement?

● Publish a new version on the serverless application using CloudFormation. Set up a manual approval action in CodePipeline in order to verify and approve the version that will be deployed. Once the change has been verified, invoke the Lambda functions to use the production alias using CodePipeline.
● Develop a custom script that uses AWS CLI to update the Lambda functions. Integrate the script in CodeBuild that will automatically publish the new version of the application by switching to the production alias.
● Use AWS CloudFormation to define and publish the new version of the serverless application. Deploy the new version of the AWS Lambda functions with AWS CodeDeploy using the CodeDeployDefault.LambdaAllAtOnce predefined deployment configuration.
✅ Define and publish the new version on the serverless application using CloudFormation. Deploy the new version of the AWS Lambda functions with AWS CodeDeploy using the CodeDeployDefault.LambdaCanary10Percent5Minutes predefined deployment configuration.
Explanation:-CodeDeploy is a deployment service that automates application deployments to Amazon EC2 instances, on-premises instances, serverless Lambda functions, or Amazon ECS services. When you deploy to an AWS Lambda compute platform, the deployment configuration specifies the way traffic is shifted to the new Lambda function versions in your application.
There are three ways traffic can shift during a deployment:
- Canary: Traffic is shifted in two increments. You can choose from predefined canary options that specify the percentage of traffic shifted to your updated Lambda function version in the first increment and the interval, in minutes, before the remaining traffic is shifted in the second increment.
- Linear: Traffic is shifted in equal increments with an equal number of minutes between each increment. You can choose from predefined linear options that specify the percentage of traffic shifted in each increment and the number of minutes between each increment.
- All-at-once: All traffic is shifted from the original Lambda function to the updated Lambda function version all at once.
The following table lists the predefined configurations available for AWS Lambda deployments.
Hence, the correct answer is: Define and publish the new version on the serverless application using CloudFormation. Deploy the new version of the AWS Lambda functions with AWS CodeDeploy using the CodeDeployDefault.LambdaCanary10Percent5Minutes predefined deployment configuration.
The option that says: Publish a new version on the serverless application using CloudFormation. Set up a manual approval action in CodePipeline in order to verify and approve the version that will be deployed. Once the change has been verified, invoke the Lambda functions to use the production alias using CodePipeline is incorrect because although this set up allows you to verify the new version of the serverless application before the production deployment, it still fails to meet the requirement of deploying the change to a small subset of users only. This deployment configuration will release the new version to all users.
The option that says: Develop a custom script that uses AWS CLI to update the Lambda functions. Integrate the script in CodeBuild that will automatically publish the new version of the application by switching to the production alias is incorrect because developing a custom script to update the Lambda functions is not necessary. Moreover, AWS CodeBuild is not capable of publishing new versions of your Lambda functions. You

should use AWS CodeDeploy instead.
The option that says: Use AWS CloudFormation to define and publish the new version on the serverless application. Deploy the new version of the AWS Lambda functions with AWS CodeDeploy using the CodeDeployDefault.LambdaAllAtOnce predefined deployment configuration is incorrect because this configuration will deploy the changes to all users immediately. Keep in mind that the scenario requires that the change should only be deployed to a small subset of users only. You have to use a Canary deployment instead or the CodeDeployDefault.LambdaCanary10Percent5Minutes configuration.
References:
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-configurations.html
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-steps-lambda.html
Check out this AWS CodeDeploy Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-codedeploy/

Q33) You are developing a mobile news homepage that curates several news sources to a single page. The app is mainly composed of several Lambda functions configured as a deployment group on AWS CodeDeploy. For each new app version, you need to test all APIs before fully deploying it to production. The APIs are using a set of AWS Lambda validation scripts. You want the ability to check the APIs during deployments and be notified for any API errors as well as automatic rollback if the validation fails.
Which combination of the options below can help you implement a solution for this scenario? (Select THREE)

✅ Define your Lambda validation scripts on the AppSpec lifecycle hook during deployment to run the validation using test traffic and trigger a rollback if checks fail.
Explanation:-You can use CloudWatch Alarms to track metrics on your new deployment and you can set thresholds for those metrics in your Auto Scaling groups being managed by CodeDeploy. This can invoke an action if the metric you are tracking crosses the threshold for a defined period of time. You can also monitor metrics such as instance CPU utilization, Memory utilization or custom metrics you have configured. If the alarm is activated, CloudWatch initiates actions such as sending a notification to Amazon Simple Notification Service, stopping a CodeDeploy deployment, or changing the state of an instance. You will also have the option to automatically roll back a deployment when a deployment fails or when a CloudWatch alarm is activated. CodeDeploy will redeploy the last known working version of the application when it rolls back.
With Amazon SNS, you can create triggers that send notifications to subscribers of an Amazon SNS topic when specified events, such as success or failure events, occur in deployments and instances. CloudWatch Alarms can trigger sending out notification to your configured SNS topic.
The BeforeAllowTraffic and AfterAllowTraffic lifecycle hooks of the AppSpec.yaml file allows you to use Lambda functions to validate the new version task set using the test traffic during the deployment. For example, a Lambda function can serve traffic to the test listener and track metrics from the replacement task set. If rollbacks are configured, you can configure a CloudWatch alarm that triggers a rollback when the validation test in your Lambda function fails.
Hence, the correct answers are:
Define your Lambda validation scripts on the AppSpec lifecycle hook during deployment to run the validation using test traffic and trigger a rollback if checks fail.
Associate an AWS CloudWatch Alarm to your deployment group that can send a notification to an AWS SNS topic when threshold for 5xx is reached on CloudWatch.
Configure your Lambda validation scripts to run during deployment and configure a CloudWatch Alarm that will trigger a rollback when the function validation fails.
The option that says: Add a step on AWS CodeDeploy to trigger your Lambda validation scripts after deployment and invoke them after deployment to validate your new app version is incorrect because you will want the validation script to run before production traffic is flowing on the new app version. You can use AppSpec hooks to do this, which also includes an option to rollback when validation fails.
The option that says: Have CodeDeploy run the AWS Lambda validations after the deployment so you can test with production traffic. When errors are found, have another trigger to rollback the deployment is incorrect because when the new app version is deployed to production, there's a possibility that clients will notice these errors. Rollback will take some time as the old version will need to be re-deployed. It is better to run the validation scripts during the deployment using the test traffic.
The option that says: Have Lambda send results to AWS CloudWatch Alarms directly and trigger a rollback when 5xx reply errors are received during deployment is incorrect because CloudWatch Alarms can't receive direct test results from AWS Lambda. If you want to do this, store test logs on CloudWatch logs and have CloudWatch Events monitor those logs.
References:
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-steps.html#deployment-steps-what-happens
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments-rollback-and-redeploy.html#deployments-rollback-and-redeploy-automatic-rollbacks
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-groups-configure-advanced-options.html
Check out these AWS Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudwatch/
https://tutorialsdojo.com/aws-cheat-sheet-amazon-sns/
⚪ Have CodeDeploy run the AWS Lambda validations after the deployment so you can test with production traffic. When errors are found, have another trigger to rollback the deployment.
✅ Associate an AWS CloudWatch Alarm to your deployment group that can send a notification to an AWS SNS topic when threshold for 5xx is reached on CloudWatch.
Explanation:-You can use CloudWatch Alarms to track metrics on your new deployment and you can set thresholds for those metrics in your Auto Scaling groups being managed by CodeDeploy. This can invoke an action if the metric you are tracking crosses the threshold for a defined period of time. You can also monitor metrics such as instance CPU utilization, Memory utilization or custom metrics you have configured. If the alarm is activated, CloudWatch initiates actions such as sending a notification to Amazon Simple Notification Service, stopping a CodeDeploy deployment, or changing the state of an instance. You will also have the option to automatically roll back a deployment when a deployment fails or when a CloudWatch alarm is activated. CodeDeploy will redeploy the last known working version of the application when it rolls back.
With Amazon SNS, you can create triggers that send notifications to subscribers of an Amazon SNS topic when specified events, such as success or failure events, occur in deployments and instances. CloudWatch Alarms can trigger sending out notification to your configured SNS topic.
The BeforeAllowTraffic and AfterAllowTraffic lifecycle hooks of the AppSpec.yaml file allows you to use Lambda functions to validate the new version task set using the test traffic during the deployment. For example, a Lambda function can serve traffic to the test listener and track metrics from the replacement task set. If rollbacks are configured, you can configure a CloudWatch alarm that triggers a rollback when the validation test in your Lambda function fails.
Hence, the correct answers are:
Define your Lambda validation scripts on the AppSpec lifecycle hook during deployment to run the validation using test traffic and trigger a rollback if checks fail.
Associate an AWS CloudWatch Alarm to your deployment group that can send a notification to an AWS SNS topic when threshold for 5xx is reached on CloudWatch.
Configure your Lambda validation scripts to run during deployment and configure a CloudWatch Alarm that will trigger a rollback when the function

validation fails.
The option that says: Add a step on AWS CodeDeploy to trigger your Lambda validation scripts after deployment and invoke them after deployment to validate your new app version is incorrect because you will want the validation script to run before production traffic is flowing on the new app version. You can use AppSpec hooks to do this, which also includes an option to rollback when validation fails.
The option that says: Have CodeDeploy run the AWS Lambda validations after the deployment so you can test with production traffic. When errors are found, have another trigger to rollback the deployment is incorrect because when the new app version is deployed to production, there's a possibility that clients will notice these errors. Rollback will take some time as the old version will need to be re-deployed. It is better to run the validation scripts during the deployment using the test traffic.
The option that says: Have Lambda send results to AWS CloudWatch Alarms directly and trigger a rollback when 5xx reply errors are received during deployment is incorrect because CloudWatch Alarms can't receive direct test results from AWS Lambda. If you want to do this, store test logs on CloudWatch logs and have CloudWatch Events monitor those logs.
References:
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-steps.html#deployment-steps-what-happens
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments-rollback-and-redeploy.html#deployments-rollback-and-redeploy-automatic-rollbacks
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-groups-configure-advanced-options.html
Check out these AWS Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudwatch/
https://tutorialsdojo.com/aws-cheat-sheet-amazon-sns/
✅ Configure your Lambda validation scripts to run during deployment and configure a CloudWatch Alarm that will trigger a rollback when the function validation fails.
Explanation:-You can use CloudWatch Alarms to track metrics on your new deployment and you can set thresholds for those metrics in your Auto Scaling groups being managed by CodeDeploy. This can invoke an action if the metric you are tracking crosses the threshold for a defined period of time. You can also monitor metrics such as instance CPU utilization, Memory utilization or custom metrics you have configured. If the alarm is activated, CloudWatch initiates actions such as sending a notification to Amazon Simple Notification Service, stopping a CodeDeploy deployment, or changing the state of an instance. You will also have the option to automatically roll back a deployment when a deployment fails or when a CloudWatch alarm is activated. CodeDeploy will redeploy the last known working version of the application when it rolls back.
With Amazon SNS, you can create triggers that send notifications to subscribers of an Amazon SNS topic when specified events, such as success or failure events, occur in deployments and instances. CloudWatch Alarms can trigger sending out notification to your configured SNS topic.
The BeforeAllowTraffic and AfterAllowTraffic lifecycle hooks of the AppSpec.yaml file allows you to use Lambda functions to validate the new version task set using the test traffic during the deployment. For example, a Lambda function can serve traffic to the test listener and track metrics from the replacement task set. If rollbacks are configured, you can configure a CloudWatch alarm that triggers a rollback when the validation test in your Lambda function fails.
Hence, the correct answers are:
Define your Lambda validation scripts on the AppSpec lifecycle hook during deployment to run the validation using test traffic and trigger a rollback if checks fail.
Associate an AWS CloudWatch Alarm to your deployment group that can send a notification to an AWS SNS topic when threshold for 5xx is reached on CloudWatch.
Configure your Lambda validation scripts to run during deployment and configure a CloudWatch Alarm that will trigger a rollback when the function validation fails.
The option that says: Add a step on AWS CodeDeploy to trigger your Lambda validation scripts after deployment and invoke them after deployment to validate your new app version is incorrect because you will want the validation script to run before production traffic is flowing on the new app version. You can use AppSpec hooks to do this, which also includes an option to rollback when validation fails.
The option that says: Have CodeDeploy run the AWS Lambda validations after the deployment so you can test with production traffic. When errors are found, have another trigger to rollback the deployment is incorrect because when the new app version is deployed to production, there's a possibility that clients will notice these errors. Rollback will take some time as the old version will need to be re-deployed. It is better to run the validation scripts during the deployment using the test traffic.
The option that says: Have Lambda send results to AWS CloudWatch Alarms directly and trigger a rollback when 5xx reply errors are received during deployment is incorrect because CloudWatch Alarms can't receive direct test results from AWS Lambda. If you want to do this, store test logs on CloudWatch logs and have CloudWatch Events monitor those logs.
References:
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-steps.html#deployment-steps-what-happens
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments-rollback-and-redeploy.html#deployments-rollback-and-redeploy-automatic-rollbacks
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-groups-configure-advanced-options.html
Check out these AWS Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudwatch/
https://tutorialsdojo.com/aws-cheat-sheet-amazon-sns/
● Add a step on AWS CodeDeploy to trigger your Lambda validation scripts after deployment and invoke them after deployment to validate your new app version.
● Have Lambda send results to AWS CloudWatch Alarms directly and trigger a rollback when 5xx reply errors are received during deployment.

Q34) A government agency has a VMware-based automated server build system on their on-premises network which uses a virtualization software that allows them to create server images of their application. They instructed their DevOps Engineer to set up a system that will allow them to test their server images using their on-premises server pipeline to resemble the build and behavior on Amazon EC2. In this way, the agency can verify the functionality of their application, detect incompatibility issues, and determine any prerequisites on the new Amazon Linux 2 operating system that will be used in AWS.
Which of the following solutions should the DevOps Engineer implement to accomplish this task?

● Use an AWS OpsWorks deployment agent that can reformat the target server on-premises to use Amazon Linux and then install the application afterwards. Start the testing once the installation is complete.
● Download the latest AmazonLinux2.iso of the Amazon Linux 2 operating system and import it to your on-premises network. Directly launch a new on-premises server based on the imported ISO, without any virtual platform. Deploy the application, and commence testing.
● Launch a new on-premises server with any distribution of Linux operating system such as CentOS, Ubuntu or Fedora since these are technically the same. Deploy the application to the server for testing.
✅ Launch an Amazon EC2 instance with the latest Amazon Linux OS in AWS. Use the AWS Import/Export service to export the EC2 image to a VMware ISO in an S3 bucket and then import the ISO to an on-premises server. Once done, commence the testing activity to verify the application's functionalities.
Explanation:-The VM Import/Export enables you to easily import virtual machine images from your existing environment to Amazon EC2 instances and export them back to your on-premises environment. This offering allows you to leverage your existing investments in the virtual machines that

you have built to meet your IT security, configuration management, and compliance requirements by bringing those virtual machines into Amazon EC2 as ready-to-use instances. You can also export imported instances back to your on-premises virtualization infrastructure, allowing you to deploy workloads across your IT infrastructure.
To import your images, use the AWS CLI or other developer tools to import a virtual machine (VM) image from your VMware environment. If you use the VMware vSphere virtualization platform, you can also use the AWS Management Portal for vCenter to import your VM. As part of the import process, VM Import will convert your VM into an Amazon EC2 AMI, which you can use to run Amazon EC2 instances. Once your VM has been imported, you can take advantage of Amazon's elasticity, scalability and monitoring via offerings like Auto Scaling, Elastic Load Balancing and CloudWatch to support your imported images.
You can export previously imported EC2 instances using the Amazon EC2 API tools. You simply specify the target instance, virtual machine file format and a destination S3 bucket, and VM Import/Export will automatically export the instance to the S3 bucket. You can then download and launch the exported VM within your on-premises virtualization infrastructure.
You can import Windows and Linux VMs that use VMware ESX or Workstation, Microsoft Hyper-V, and Citrix Xen virtualization formats. And you can export previously imported EC2 instances to VMware ESX, Microsoft Hyper-V or Citrix Xen formats.
Hence, the correct answer is: Launch an Amazon EC2 instance with the latest Amazon Linux OS in AWS. Use the AWS Import/Export service to export the EC2 image to a VMware ISO in an S3 bucket and then import the ISO to an on-premises server. Once done, commence the testing activity to verify the application's functionalities.
The option that says: Download the latest ISO of the Amazon Linux 2 operating system and import it to your on-premises network. Launch a new on-premises server based on the imported ISO, deploy the application, and commence testing is incorrect because there is no way to directly download the AmazonLinux2.iso for Amazon Linux 2. You have to use VM Import/Export service instead or alternatively, run the Amazon Linux 2 as a virtual machine in your on-premises data center. Again, you won't be able to directly download the ISO image but you can get the Amazon Linux 2 image for the specific virtualization platform of your choice. If you are using VMware, you can download the ESX image *.ova and for VirtualBox, you'll get the *.vdi image file. What you should do first is to prepare the seed.iso boot image then connect it to the VM of your choice on first boot.
The option that says: Use an AWS OpsWorks deployment agent that can reformat the target server on-premises to use Amazon Linux and then install the application afterwards. Start the testing once the installation is complete is incorrect because OpsWorks is primarily used to deploy applications to both your on-premises servers as well as your EC2 instances which reside in your VPC. It doesn't have the capability to reformat the target server to use an Amazon Linux OS or to switch to another type of OS at all.
The option that says: Launch a new on-premises server with any distribution of Linux operating system such as CentOS, Ubuntu or Fedora since these are technically the same. Deploy the application to the server for testing is incorrect because these Linux distributions are actually different from one another. There could be some incompatibility issues between the different Linux operating systems which is why you need to test your application on a specific Amazon Linux 2 type only.
References:
https://docs.aws.amazon.com/vm-import/latest/userguide/vmexport_image.html
https://aws.amazon.com/ec2/vm-import/
https://docs.aws.amazon.com/vm-import/latest/userguide/vmimport-image-import.html
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/amazon-linux-2-virtual-machine.html

---

Q35) The Development team of a leading IT consultancy company would like to add a manual approval action before their new application versions are deployed to their production environment. The approval action must be strictly enforced even if the unit and integration tests are all successful. They have set up a pipeline using CodePipeline to orchestrate the workflow of their continuous integration and continuous delivery processes. The new versions of the application are built using CodeBuild and are deployed to a fleet of Amazon EC2 instances using CodeDeploy.
Which of the following provides the SIMPLEST and the MOST cost-effective solution?

● After the last deploy action of the pipeline, set up a test action to verify the application's functionality. Add the required action steps to automatically do the unit and integration tests using AWS Step Functions. Mark the action as successful if all of the tests have been successfully passed. Create a manual approval action and inform the team of the stage being triggered using SNS. Add a deploy action to deploy the app to the next stage at the end of the pipeline.
● After the last deploy action of the pipeline, set up a test action to verify the application's functionality. Add the required action steps to automatically do the unit and integration tests using a third-party CI/CD Tool such as GitLab or Jenkins hosted in Amazon EC2. Mark the action as successful if all of the tests have been successfully passed. Create a manual approval action and inform the team of the stage being triggered using SNS. Add a deploy action to deploy the app to the next stage at the end of the pipeline.
✅ After the last deploy action of the pipeline, set up a manual approval action and inform the team of the stage being triggered using SNS. In CodeBuild, add the required actions to automatically do the unit and integration tests. Add a deploy action to deploy the app to the next stage at the end of the pipeline.
Explanation:-You can automate your release process by using AWS CodePipeline to test your code and run your builds with CodeBuild. You can create reports in CodeBuild that contain details about tests that are run during builds.
You can create tests such as unit tests, configuration tests, and functional tests. The test file format can be JUnit XML or Cucumber JSON. Create your test cases with any test framework that can create files in one of those formats (for example, Surefire JUnit plugin, TestNG, and Cucumber). To create a test report, you add a report group name to the buildspec file of a build project with information about your test cases. When you run the build project, the test cases are run and a test report is created. You do not need to create a report group before you run your tests. If you specify a report group name, CodeBuild creates a report group for you when you run your reports. If you want to use a report group that already exists, you specify its ARN in the buildspec file.
In AWS CodePipeline, you can add an approval action to a stage in a pipeline at the point where you want the pipeline execution to stop so that someone with the required AWS Identity and Access Management permissions can approve or reject the action.
If the action is approved, the pipeline execution resumes. If the action is rejected—or if no one approves or rejects the action within seven days of the pipeline reaching the action and stopping—the result is the same as an action failing, and the pipeline execution does not continue.
You might use manual approvals for these reasons:
- You want someone to perform a code review or change management review before a revision is allowed into the next stage of a pipeline.
- You want someone to perform manual quality assurance testing on the latest version of an application, or to confirm the integrity of a build artifact, before it is released.
- You want someone to review new or updated text before it is published to a company website.
Hence, the correct answer is: After the last deploy action of the pipeline, set up a manual approval action and inform the team of the stage being triggered using SNS. In CodeBuild, add the required actions to automatically do the unit and integration tests. Add a deploy action to deploy the app to the next stage at the end of the pipeline.
The option that says: After the last deploy action of the pipeline, set up a test action to verify the application's functionality. In CodeBuild, add the required actions to automatically do the unit and integration tests. Mark the action as successful if all of the tests have been successfully passed. Create a custom action with a corresponding custom job worker that performs the approval action. Inform the team of the stage being triggered using SNS. Add a deploy action to deploy the app to the next stage at the end of the pipeline is incorrect because you can just simply set up a manual approval action instead of creating a custom action. That takes a lot of effort to configure including the development of a custom job worker.
The option that says: After the last deploy action of the pipeline, set up a test action to verify the application's functionality. Add the required action

steps to automatically do the unit and integration tests using AWS Step Functions. Mark the action as successful if all of the tests have been successfully passed. Create a manual approval action and inform the team of the stage being triggered using SNS. Add a deploy action to deploy the app to the next stage at the end of the pipeline is incorrect because it is tedious to automatically perform the unit and integration tests using AWS Step Functions. You can just use CodeBuild to handle all of the tests.
The option that says: After the last deploy action of the pipeline, set up a test action to verify the application's functionality. Add the required action steps to automatically do the unit and integration tests using a third-party CI/CD Tool such as GitLab or Jenkins hosted in Amazon EC2. Mark the action as successful if all of the tests have been successfully passed. Create a manual approval action and inform the team of the stage being triggered using SNS. Add a deploy action to deploy the app to the next stage at the end of the pipeline is incorrect because this solution entails an additional burden to install, configure and launch a third-party CI/CD tool in Amazon EC2. A more simple solution is to just use CodeBuild for tests.
References:
https://docs.aws.amazon.com/codepipeline/latest/userguide/approvals.html
https://docs.aws.amazon.com/codepipeline/latest/userguide/approvals-action-add.html
https://docs.aws.amazon.com/codebuild/latest/userguide/how-to-create-pipeline.html#how-to-create-pipeline-add-test
Check out this AWS CodePipeline Cheat Sheet:
https://tutorialsdojo.com/aws-codepipeline/
● After the last deploy action of the pipeline, set up a test action to verify the application's functionality. In CodeBuild, add the required actions to automatically do the unit and integration tests. Mark the action as successful if all of the tests have been successfully passed. Create a custom action with a corresponding custom job worker that performs the approval action. Inform the team of the stage being triggered using SNS. Add a deploy action to deploy the app to the next stage at the end of the pipeline.

Q36) A leading food and beverage company is currently migrating its Docker-based application that is hosted on-premies to AWS. The application will be hosted in an Amazon ECS cluster with multiple ECS services to run its various workloads. The cluster is configured to use an Application Load Balancer to distribute traffic evenly across the tasks in your service. A DevOps Engineer was instructed to configure the cluster to automatically collect logs from all of the services and upload them to an S3 bucket for near-real-time analysis.
How should the Engineer configure the ECS set up to satisfy these requirements? (Select THREE)

✅ Capture detailed information about requests sent to your load balancer by enabling access logging on the Application Load Balancer. Configure it to store the logs to the S3 bucket.
Explanation:-You can configure the containers in your tasks to send log information to CloudWatch Logs. If you are using the Fargate launch type for your tasks, this allows you to view the logs from your containers. If you are using the EC2 launch type, this enables you to view different logs from your containers in one convenient location, and it prevents your container logs from taking up disk space on your container instances.
The type of information that is logged by the containers in your task depends mostly on their ENTRYPOINT command. By default, the logs that are captured show the command output that you would normally see in an interactive terminal if you ran the container locally, which are the STDOUT and STDERR I/O streams. The awslogs log driver simply passes these logs from Docker to CloudWatch Logs.
You can use subscriptions to get access to a real-time feed of log events from CloudWatch Logs and have it delivered to other services such as a Amazon Kinesis stream, Amazon Kinesis Data Firehose stream, or AWS Lambda for custom processing, analysis, or loading to other systems. To begin subscribing to log events, create the receiving source, such as a Kinesis stream, where the events will be delivered. A subscription filter defines the filter pattern to use for filtering which log events get delivered to your AWS resource, as well as information about where to send matching log events to.
Elastic Load Balancing provides access logs that capture detailed information about requests sent to your load balancer. Each log contains information such as the time the request was received, the client's IP address, latencies, request paths, and server responses. You can use these access logs to analyze traffic patterns and troubleshoot issues.
Access logging is an optional feature of Elastic Load Balancing that is disabled by default. After you enable access logging for your load balancer, Elastic Load Balancing captures the logs and stores them in the Amazon S3 bucket that you specify as compressed files. You can disable access logging at any time.
Each access log file is automatically encrypted before it is stored in your S3 bucket and decrypted when you access it. You do not need to take any action; the encryption and decryption is performed transparently. Each log file is encrypted with a unique key, which is itself encrypted with a master key that is regularly rotated.
Hence, the correct answers are:
- Create the required IAM Policy and attach it to the ecsInstanceRole. Install the Amazon CloudWatch Logs agent on the Amazon ECS instances. Use the awslogs Log Driver in the Amazon ECS task definition.
- Capture detailed information about requests sent to your load balancer by enabling access logging on the Application Load Balancer. Configure it to store the logs to the S3 bucket.
- Create a CloudWatch Logs subscription filter integrated with Amazon Kinesis to analyze the logs. Configure the CloudWatch Logs to export the logs to an S3 bucket.
The option that says: Set up Amazon Macie to analyze the access logs in the S3 bucket is incorrect because Amazon Macie is just a security service that uses machine learning to automatically discover, classify, and protect sensitive data in AWS. It can't analyze logs in near-real-time unlike Kinesis.
The option that says: Integrate a Lambda function with CloudWatch Events to run a process every minute that invokes the CreateLogGroup and CreateExportTask CloudWatch Logs API to push the logs to the S3 bucket is incorrect because although this step may work, it is still better to use the awslogs Log Driver instead of developing a custom scheduled job. It is unnecessary since you only need to change the log driver in your task definition.
The option that says: Capture detailed information about requests sent to your load balancer by using Detailed Monitoring in CloudWatch. Configure it to store the logs to the S3 bucket is incorrect because the Detailed Monitoring feature simply sends the metric data for your instance to CloudWatch in 1-minute periods.
References:
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/using_awslogs.html
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-access-logs.html
Check out these Amazon ECS and AWS Elastic Load Balancing Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-elastic-container-service-amazon-ecs/
https://tutorialsdojo.com/aws-cheat-sheet-aws-elastic-load-balancing-elb/
● Integrate a Lambda function with CloudWatch Events to run a process every minute that invokes the CreateLogGroup and CreateExportTask CloudWatch Logs API to push the logs to the S3 bucket.
● Set up Amazon Macie to analyze the access logs in the S3 bucket.
✅ Create the required IAM Policy and attach it to the ecsInstanceRole. Install the Amazon CloudWatch Logs agent on the Amazon ECS instances. Use the awslogs Log Driver in the Amazon ECS task definition.
Explanation:-You can configure the containers in your tasks to send log information to CloudWatch Logs. If you are using the Fargate launch type for your tasks, this allows you to view the logs from your containers. If you are using the EC2 launch type, this enables you to view different logs from your containers in one convenient location, and it prevents your container logs from taking up disk space on your container instances.
The type of information that is logged by the containers in your task depends mostly on their ENTRYPOINT command. By default, the logs that are

captured show the command output that you would normally see in an interactive terminal if you ran the container locally, which are the STDOUT and STDERR I/O streams. The awslogs log driver simply passes these logs from Docker to CloudWatch Logs.
You can use subscriptions to get access to a real-time feed of log events from CloudWatch Logs and have it delivered to other services such as a Amazon Kinesis stream, Amazon Kinesis Data Firehose stream, or AWS Lambda for custom processing, analysis, or loading to other systems. To begin subscribing to log events, create the receiving source, such as a Kinesis stream, where the events will be delivered. A subscription filter defines the filter pattern to use for filtering which log events get delivered to your AWS resource, as well as information about where to send matching log events to.
Elastic Load Balancing provides access logs that capture detailed information about requests sent to your load balancer. Each log contains information such as the time the request was received, the client's IP address, latencies, request paths, and server responses. You can use these access logs to analyze traffic patterns and troubleshoot issues.
Access logging is an optional feature of Elastic Load Balancing that is disabled by default. After you enable access logging for your load balancer, Elastic Load Balancing captures the logs and stores them in the Amazon S3 bucket that you specify as compressed files. You can disable access logging at any time.
Each access log file is automatically encrypted before it is stored in your S3 bucket and decrypted when you access it. You do not need to take any action; the encryption and decryption is performed transparently. Each log file is encrypted with a unique key, which is itself encrypted with a master key that is regularly rotated.
Hence, the correct answers are:
- Create the required IAM Policy and attach it to the ecsInstanceRole. Install the Amazon CloudWatch Logs agent on the Amazon ECS instances. Use the awslogs Log Driver in the Amazon ECS task definition.
- Capture detailed information about requests sent to your load balancer by enabling access logging on the Application Load Balancer. Configure it to store the logs to the S3 bucket.
- Create a CloudWatch Logs subscription filter integrated with Amazon Kinesis to analyze the logs. Configure the CloudWatch Logs to export the logs to an S3 bucket.
The option that says: Set up Amazon Macie to analyze the access logs in the S3 bucket is incorrect because Amazon Macie is just a security service that uses machine learning to automatically discover, classify, and protect sensitive data in AWS. It can't analyze logs in near-real-time unlike Kinesis.
The option that says: Integrate a Lambda function with CloudWatch Events to run a process every minute that invokes the CreateLogGroup and CreateExportTask CloudWatch Logs API to push the logs to the S3 bucket is incorrect because although this step may work, it is still better to use the awslogs Log Driver instead of developing a custom scheduled job. It is unnecessary since you only need to change the log driver in your task definition.
The option that says: Capture detailed information about requests sent to your load balancer by using Detailed Monitoring in CloudWatch. Configure it to store the logs to the S3 bucket is incorrect because the Detailed Monitoring feature simply sends the metric data for your instance to CloudWatch in 1-minute periods.
References:
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/using_awslogs.html
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-access-logs.html
Check out these Amazon ECS and AWS Elastic Load Balancing Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-elastic-container-service-amazon-ecs/
https://tutorialsdojo.com/aws-cheat-sheet-aws-elastic-load-balancing-elb/
● Capture detailed information about requests sent to your load balancer by using Detailed Monitoring in CloudWatch. Configure it to store the logs to the S3 bucket.
✅ Create a CloudWatch Logs subscription filter integrated with Amazon Kinesis to analyze the logs. Configure the CloudWatch Logs to export the logs to an S3 bucket.
Explanation:-You can configure the containers in your tasks to send log information to CloudWatch Logs. If you are using the Fargate launch type for your tasks, this allows you to view the logs from your containers. If you are using the EC2 launch type, this enables you to view different logs from your containers in one convenient location, and it prevents your container logs from taking up disk space on your container instances.
The type of information that is logged by the containers in your task depends mostly on their ENTRYPOINT command. By default, the logs that are captured show the command output that you would normally see in an interactive terminal if you ran the container locally, which are the STDOUT and STDERR I/O streams. The awslogs log driver simply passes these logs from Docker to CloudWatch Logs.
You can use subscriptions to get access to a real-time feed of log events from CloudWatch Logs and have it delivered to other services such as a Amazon Kinesis stream, Amazon Kinesis Data Firehose stream, or AWS Lambda for custom processing, analysis, or loading to other systems. To begin subscribing to log events, create the receiving source, such as a Kinesis stream, where the events will be delivered. A subscription filter defines the filter pattern to use for filtering which log events get delivered to your AWS resource, as well as information about where to send matching log events to.
Elastic Load Balancing provides access logs that capture detailed information about requests sent to your load balancer. Each log contains information such as the time the request was received, the client's IP address, latencies, request paths, and server responses. You can use these access logs to analyze traffic patterns and troubleshoot issues.
Access logging is an optional feature of Elastic Load Balancing that is disabled by default. After you enable access logging for your load balancer, Elastic Load Balancing captures the logs and stores them in the Amazon S3 bucket that you specify as compressed files. You can disable access logging at any time.
Each access log file is automatically encrypted before it is stored in your S3 bucket and decrypted when you access it. You do not need to take any action; the encryption and decryption is performed transparently. Each log file is encrypted with a unique key, which is itself encrypted with a master key that is regularly rotated.
Hence, the correct answers are:
- Create the required IAM Policy and attach it to the ecsInstanceRole. Install the Amazon CloudWatch Logs agent on the Amazon ECS instances. Use the awslogs Log Driver in the Amazon ECS task definition.
- Capture detailed information about requests sent to your load balancer by enabling access logging on the Application Load Balancer. Configure it to store the logs to the S3 bucket.
- Create a CloudWatch Logs subscription filter integrated with Amazon Kinesis to analyze the logs. Configure the CloudWatch Logs to export the logs to an S3 bucket.
The option that says: Set up Amazon Macie to analyze the access logs in the S3 bucket is incorrect because Amazon Macie is just a security service that uses machine learning to automatically discover, classify, and protect sensitive data in AWS. It can't analyze logs in near-real-time unlike Kinesis.
The option that says: Integrate a Lambda function with CloudWatch Events to run a process every minute that invokes the CreateLogGroup and CreateExportTask CloudWatch Logs API to push the logs to the S3 bucket is incorrect because although this step may work, it is still better to use the awslogs Log Driver instead of developing a custom scheduled job. It is unnecessary since you only need to change the log driver in your task definition.
The option that says: Capture detailed information about requests sent to your load balancer by using Detailed Monitoring in CloudWatch. Configure it to store the logs to the S3 bucket is incorrect because the Detailed Monitoring feature simply sends the metric data for your instance to CloudWatch in 1-minute periods.
References:
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/using_awslogs.html

https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-access-logs.html
Check out these Amazon ECS and AWS Elastic Load Balancing Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-elastic-container-service-amazon-ecs/
https://tutorialsdojo.com/aws-cheat-sheet-aws-elastic-load-balancing-elb/

Q37) You are working as a DevOps Engineer in a leading aerospace engineering company that has a hybrid cloud architecture that connects its on-premises data center with AWS via Direct Connect Gateway. There is a new requirement in which you have to implement an automated OS patching solution for all of the Windows servers hosted on-premises as well as in AWS Cloud. The AWS Systems Manager service should be utilized to automate the patching of your servers.
Which combination of steps should you set up to satisfy this requirement? (Select TWO)

Download and install the SSM Agent on the hybrid servers by using the activation codes and activation IDs that you obtained. Register the servers or virtual machines on-premises to the AWS Systems Manager service. In the SSM console, the hybrid instances will show with an i- prefix. Apply the patches using the Systems Manager State Manager.

✅ Download and install the SSM Agent on the hybrid servers by using the activation codes and activation IDs that you obtained. Register the servers or virtual machines on-premises to the AWS Systems Manager service. In the SSM console, the hybrid instances will show with an mi- prefix. Apply the patches using the Systems Manager Patch Manager.

Explanation:-A hybrid environment includes on-premises servers and virtual machines (VMs) that have been configured for use with Systems Manager, including VMs in other cloud environments. After following the steps below, the users who have been granted permissions by the AWS account administrator can use AWS Systems Manager to configure and manage their organization's on-premises servers and virtual machines (VMs).
To configure your hybrid servers and VMs for AWS Systems Manager, just follow these provided steps:
1. Complete General Systems Manager Setup Steps
2. Create an IAM Service Role for a Hybrid Environment
3. Install a TLS certificate on On-Premises Servers and VMs
4. Create a Managed-Instance Activation for a Hybrid Environment
5. Install SSM Agent for a Hybrid Environment (Windows)
6. Install SSM Agent for a Hybrid Environment (Linux)
7. (Optional) Enable the Advanced-Instances Tier
Configuring your hybrid environment for Systems Manager enables you to do the following:
- Create a consistent and secure way to remotely manage your hybrid workloads from one location using the same tools or scripts.
- Centralize access control for actions that can be performed on your servers and VMs by using AWS Identity and Access Management (IAM).
- Centralize auditing and your view into the actions performed on your servers and VMs by recording all actions in AWS CloudTrail.
- Centralize monitoring by configuring CloudWatch Events and Amazon SNS to send notifications about service execution success.
After you finish configuring your servers and VMs for Systems Manager, your hybrid machines are listed in the AWS Management Console and described as managed instances. Amazon EC2 instances configured for Systems Manager are also described as managed instances. In the console, however, the IDs of your hybrid instances are distinguished from Amazon EC2 instances with the prefix "mi-". Amazon EC2 instance IDs use the prefix "i-".
Servers and virtual machines (VMs) in a hybrid environment require an IAM role to communicate with the Systems Manager service. The role grants AssumeRole trust to the Systems Manager service. You only need to create the service role for a hybrid environment once for each AWS account.
Hence, the correct answers are:
- Set up a single IAM service role for AWS Systems Manager to enable the service to execute the STS AssumeRole operation. Allow the generation of service tokens by registering the IAM role. Use the service role to perform the managed-instance activation.
- Download and install the SSM Agent on the hybrid servers by using the activation codes and activation IDs that you obtained. Register the servers or virtual machines on-premises to the AWS Systems Manager service. In the SSM console, the hybrid instances will show with an mi- prefix. Apply the patches using the Systems Manager Patch Manager.
The option that says: Set up several IAM service roles for AWS Systems Manager to enable the service to execute the STS AssumeRoleWithSAML operation. Allow the generation of service tokens by registering the IAM role. Use the service role to perform the managed-instance activation is incorrect because you have to execute the AssumeRole operation instead and not the AssumeRoleWithSAML operation. Moreover, you only need to set up a single IAM service role.
The option that says: Download and install the SSM Agent on the hybrid servers by using the activation codes and activation IDs that you obtained. Register the servers or virtual machines on-premises to the AWS Systems Manager service. In the SSM console, the hybrid instances will show with an i- prefix. Apply the patches using the Systems Manager Patch Manager is incorrect because the hybrid instances will show with an mi- prefix in the SSM console and not with an i- prefix.
The option that says: Download and install the SSM Agent on the hybrid servers by using the activation codes and activation IDs that you obtained. Register the servers or virtual machines on-premises to the AWS Systems Manager service. In the SSM console, the hybrid instances will show with an i- prefix. Apply the patches using the Systems Manager State Manager is incorrect because the AWS Systems Manager State Manager is just a secure and scalable configuration management service that automates the process of keeping your Amazon EC2 and hybrid infrastructure in a state that you define. You have to apply the patches using the Systems Manager Patch Manager instead. In addition, the hybrid instances will show with an mi- prefix and not with an i- prefix.
References:
https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-managedinstances.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-service-role.html
Check out this AWS Systems Manager Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-systems-manager/

✅ Set up a single IAM service role for AWS Systems Manager to enable the service to execute the STS AssumeRole operation. Allow the generation of service tokens by registering the IAM role. Use the service role to perform the managed-instance activation.

Explanation:-A hybrid environment includes on-premises servers and virtual machines (VMs) that have been configured for use with Systems Manager, including VMs in other cloud environments. After following the steps below, the users who have been granted permissions by the AWS account administrator can use AWS Systems Manager to configure and manage their organization's on-premises servers and virtual machines (VMs).
To configure your hybrid servers and VMs for AWS Systems Manager, just follow these provided steps:
1. Complete General Systems Manager Setup Steps
2. Create an IAM Service Role for a Hybrid Environment
3. Install a TLS certificate on On-Premises Servers and VMs
4. Create a Managed-Instance Activation for a Hybrid Environment
5. Install SSM Agent for a Hybrid Environment (Windows)
6. Install SSM Agent for a Hybrid Environment (Linux)
7. (Optional) Enable the Advanced-Instances Tier
Configuring your hybrid environment for Systems Manager enables you to do the following:

- Create a consistent and secure way to remotely manage your hybrid workloads from one location using the same tools or scripts.
- Centralize access control for actions that can be performed on your servers and VMs by using AWS Identity and Access Management (IAM).
- Centralize auditing and your view into the actions performed on your servers and VMs by recording all actions in AWS CloudTrail.
- Centralize monitoring by configuring CloudWatch Events and Amazon SNS to send notifications about service execution success.
After you finish configuring your servers and VMs for Systems Manager, your hybrid machines are listed in the AWS Management Console and described as managed instances. Amazon EC2 instances configured for Systems Manager are also described as managed instances. In the console, however, the IDs of your hybrid instances are distinguished from Amazon EC2 instances with the prefix "mi-". Amazon EC2 instance IDs use the prefix "i-".
Servers and virtual machines (VMs) in a hybrid environment require an IAM role to communicate with the Systems Manager service. The role grants AssumeRole trust to the Systems Manager service. You only need to create the service role for a hybrid environment once for each AWS account.
Hence, the correct answers are:
- Set up a single IAM service role for AWS Systems Manager to enable the service to execute the STS AssumeRole operation. Allow the generation of service tokens by registering the IAM role. Use the service role to perform the managed-instance activation.
- Download and install the SSM Agent on the hybrid servers by using the activation codes and activation IDs that you obtained. Register the servers or virtual machines on-premises to the AWS Systems Manager service. In the SSM console, the hybrid instances will show with an mi- prefix. Apply the patches using the Systems Manager Patch Manager.
The option that says: Set up several IAM service roles for AWS Systems Manager to enable the service to execute the STS AssumeRoleWithSAML operation. Allow the generation of service tokens by registering the IAM role. Use the service role to perform the managed-instance activation is incorrect because you have to execute the AssumeRole operation instead and not the AssumeRoleWithSAML operation. Moreover, you only need to set up a single IAM service role.
The option that says: Download and install the SSM Agent on the hybrid servers by using the activation codes and activation IDs that you obtained. Register the servers or virtual machines on-premises to the AWS Systems Manager service. In the SSM console, the hybrid instances will show with an i- prefix. Apply the patches using the Systems Manager Patch Manager is incorrect because the hybrid instances will show with an mi- prefix in the SSM console and not with an i- prefix.
The option that says: Download and install the SSM Agent on the hybrid servers by using the activation codes and activation IDs that you obtained. Register the servers or virtual machines on-premises to the AWS Systems Manager service. In the SSM console, the hybrid instances will show with an i- prefix. Apply the patches using the Systems Manager State Manager is incorrect because the AWS Systems Manager State Manager is just a secure and scalable configuration management service that automates the process of keeping your Amazon EC2 and hybrid infrastructure in a state that you define. You have to apply the patches using the Systems Manager Patch Manager instead. In addition, the hybrid instances will show with an mi- prefix and not with an i- prefix.
References:
https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-managedinstances.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-service-role.html
Check out this AWS Systems Manager Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-systems-manager/
● Set up several IAM service roles for AWS Systems Manager to enable the service to execute the STS AssumeRoleWithSAML operation. Allow the generation of service tokens by registering the IAM role. Use the service role to perform the managed-instance activation.
● Download and install the SSM Agent on the hybrid servers by using the activation codes and activation IDs that you obtained. Register the servers or virtual machines on-premises to the AWS Systems Manager service. In the SSM console, the hybrid instances will show with an i- prefix. Apply the patches using the Systems Manager Patch Manager.

Q38) A company has an application hosted in an Auto Scaling group of EC2 instances which calls an external API with a URL of http://api.tutorialsdojo.com as part of its processing. There was a recent deployment that changed the protocol of the URL from HTTP to HTTPS but after that, the application has stopped working properly. The DevOps engineer has verified using his POSTMAN tool that the external API works without any issues and the VPC being utilized is still using the default network ACL. Which of the following is the MOST appropriate course of action that the engineer should take to determine the root cause of this problem?

● Log in to the AWS Management Console and then in the VPC flow logs, look for ACCEPT records which were originated from the Auto Scaling group. Verify that the ingress security group rules of the Auto Scaling Group allow the incoming traffic from the external API.
● Log in to the AWS Management Console and view the application logs in Amazon CloudWatch Logs to troubleshoot the issue. Verify that the ingress security group rules of the Auto Scaling Group, as well as the network ACL, allow the incoming traffic from the external API.
✅ Log in to the AWS Management Console and look for REJECT records in the VPC flow logs which originated from the Auto Scaling group. Verify that the egress security group rules of the Auto Scaling Group allow the outgoing traffic to the external API.
Explanation:-Amazon Virtual Private Cloud provides features that you can use to increase and monitor the security for your virtual private cloud (VPC):
Security groups: Security groups act as a firewall for associated Amazon EC2 instances, controlling both inbound and outbound traffic at the instance level. When you launch an instance, you can associate it with one or more security groups that you've created. Each instance in your VPC could belong to a different set of security groups. If you don't specify a security group when you launch an instance, the instance is automatically associated with the default security group for the VPC.
Network access control lists (ACLs): Network ACLs act as a firewall for associated subnets, controlling both inbound and outbound traffic at the subnet level.
Flow logs: Flow logs capture information about the IP traffic going to and from network interfaces in your VPC. You can create a flow log for a VPC, subnet, or individual network interface. Flow log data is published to CloudWatch Logs or Amazon S3, and can help you diagnose overly restrictive or overly permissive security group and network ACL rules.
Traffic mirroring: You can copy network traffic from an elastic network interface of an Amazon EC2 instance. You can then send the traffic to out-of-band security and monitoring appliances.
You can use AWS Identity and Access Management to control who in your organization has permission to create and manage security groups, network ACLs, and flow logs. For example, you can give only your network administrators that permission, but not personnel who only need to launch instances.
For HTTP traffic, you must add an inbound rule on port 80 from the source address 0.0.0.0/0. For HTTPS traffic, add an inbound rule on port 443 from the source address 0.0.0.0/0. These inbound rules allow traffic from IPv4 addresses. To allow IPv6 traffic, add inbound rules on the same ports from the source address ::/0. Because security groups are stateful, the return traffic from the instance to users is allowed automatically, so you don't need to modify the security group's outbound rules.
The default network ACL is configured to allow all traffic to flow in and out of the subnets with which it is associated. Each network ACL also includes a rule whose rule number is an asterisk. This rule ensures that if a packet doesn't match any of the other numbered rules, it's denied.
In this scenario, the change of the URL from HTTP to HTTPS means that the application is using port 443 and not port 80 any more. Since the application is the one that initiates the call to the external API, it makes sense to check if the egress security group rules allow outgoing HTTPS (443) traffic.
Hence, the correct answer is: Log in to the AWS Management Console and look for REJECT records in the VPC flow logs which originated from the

Auto Scaling group. Verify that the egress security group rules of the Auto Scaling Group allow the outgoing traffic to the external API.
The option that says: Log in to the AWS Management Console and then in the VPC flow logs, look for ACCEPT records which were originated from the Auto Scaling group. Verify that the ingress security group rules of the Auto Scaling Group allow the incoming traffic from the external API is incorrect because you should first check the egress rules (instead of ingress) of your security group first. Remember that it is the Auto Scaling group of EC2 instances that initiates the call to the external API and not the other way around. In addition, it is more effective if you look for REJECT records instead of ACCEPT records in VPC Flow Logs to view the details of the failed connection to your external API.
The option that says: Log in to the AWS Management Console and view the application logs in Amazon CloudWatch Logs to troubleshoot the issue. Verify that the existing egress security group rules of the Auto Scaling Group, as well as the network ACL, allow the outgoing traffic to the external API is incorrect because, in the first place, the scenario didn't mention that the CloudWatch Logs agent is installed in the EC2 instances. Although it is right to check the existing egress security group rules, you don't need to check the network ACL since the architecture is already using a default one which is configured to allow all traffic.
The option that says: Log in to the AWS Management Console and view the application logs in Amazon CloudWatch Logs to troubleshoot the issue. Verify that the ingress security group rules of the Auto Scaling Group, as well as the network ACL, allow the incoming traffic from the external API is incorrect because you have to check the egress security group rules first instead. The scenario also didn't mention that the CloudWatch Logs agent is installed in the EC2 instances which means that you might not be able to view the application logs in CloudWatch.
References:
https://aws.amazon.com/premiumsupport/knowledge-center/connect-http-https-ec2/
https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html
https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html
Check out this Amazon VPC Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-vpc/
● Log in to the AWS Management Console and view the application logs in Amazon CloudWatch Logs to troubleshoot the issue. Verify that the existing egress security group rules of the Auto Scaling Group, as well as the network ACL, allow the outgoing traffic to the external API.

Q39) Due to the growth of its regional e-commerce website, the company has decided to expand its operations globally in the coming months ahead. The REST API web services of the app is currently running in an Auto Scaling group of EC2 instances across multiple Availability Zones behind an Application Load Balancer. For its database tier, the website is using a single Amazon Aurora MySQL database instance in the AWS Region where the company is based. The company wants to consolidate and store the data of their offerings into a single data source for their product catalog across all regions. For data privacy compliance, they need to ensure that the personal information of their users as well as their purchases and financial data are kept in their respective region.
Which of the following options can meet the above requirements and entails the LEAST amount of change to the application?

● Set up a new Amazon Redshift database to store the product catalog. Launch a new set of Amazon DynamoDB tables to store the personal information and financial data of their customers.
● Set up a DynamoDB global table to store the product catalog data of the e-commerce website. Use regional DynamoDB tables for storing the personal information and financial data of their customers.
● Set up multiple read replicas in your Amazon Aurora cluster to store the product catalog data. Launch a new DynamoDB global table for storing the personal information and financial data of their customers.
✅ Set up multiple read replicas in your Amazon Aurora cluster to store the product catalog data. Launch an additional local Amazon Aurora instances in each AWS Region for storing the personal information and financial data of their customers.
Explanation:-An Aurora global database consists of one primary AWS Region where your data is mastered, and one read-only, secondary AWS Region. Aurora replicates data to the secondary AWS Region with typical latency of under a second. You issue write operations directly to the primary DB instance in the primary AWS Region. An Aurora global database uses dedicated infrastructure to replicate your data, leaving database resources available entirely to serve application workloads. Applications with a worldwide footprint can use reader instances in the secondary AWS Region for low latency reads. In the unlikely event your database becomes degraded or isolated in an AWS region, you can promote the secondary AWS Region to take full read-write workloads in under a minute.
The Aurora cluster in the primary AWS Region where your data is mastered performs both read and write operations. The cluster in the secondary region enables low-latency reads. You can scale up the secondary cluster independently by adding one or more DB instances (Aurora Replicas) to serve read-only workloads. For disaster recovery, you can remove and promote the secondary cluster to allow full read and write operations.
Only the primary cluster performs write operations. Clients that perform write operations connect to the DB cluster endpoint of the primary cluster.
Hence, the correct answer is: Set up multiple read replicas in your Amazon Aurora cluster to store the product catalog data. Launch an additional local Amazon Aurora instances in each AWS Region for storing the personal information and financial data of their customers.
The option that says: Set up a new Amazon Redshift database to store the product catalog. Launch a new set of Amazon DynamoDB tables to store the personal information and financial data of their customers is incorrect because this solution entails a significant overhead of refactoring your application to use Redshift instead of Aurora. Moreover, Redshift is primarily used as a data warehouse solution and not suitable for OLTP or e-commerce websites.
The option that says: Set up a DynamoDB global table to store the product catalog data of the e-commerce website. Use regional DynamoDB tables for storing the personal information and financial data of their customers is incorrect because although the use of Global and Regional DynamoDB is acceptable, this solution still entails a lot of changes to the application. There is no assurance that the application can work with a NoSQL database and even so, you have to implement a series of code changes in order for this solution to work.
The option that says: Set up multiple read replicas in your Amazon Aurora cluster to store the product catalog data. Launch a new DynamoDB global table for storing the personal information and financial data of their customers is incorrect because although the use of Read Replicas is appropriate, this solution still requires you to do a lot of code changes since you will use a different database to store your regional data.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-global-database.html#aurora-global-database.advantages
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Replication.CrossRegion.html
Check out this Amazon Aurora Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-aurora/

Q40) An insurance firm is using CloudFormation for deploying their applications in AWS. They have a multi-tier web application which stores financial data in an Amazon RDS MySQL database in a Multi-AZ deployments configuration. They instructed their DevOps Engineer to upgrade the RDS instance to the latest major version of MySQL database. It is of utmost importance to ensure minimal downtime when doing the upgrade to avoid any business disruption.
Which of the following should the engineer implement to properly upgrade the database while minimizing downtime?

● In the AWS::RDS::DBInstance resource type in the CloudFormation template, update the DBEngineVersion property to the latest MySQL database version. Launch a new RDS Read Replica with the same properties as the primary database instance that will be upgraded. Finally, trigger an Update Stack operation in CloudFormation.
● In the AWS::RDS::DBInstance resource type in the CloudFormation template, update the EngineVersion property to the latest MySQL database

version. Afterwards, directly trigger an Update Stack operation in CloudFormation.

In the AWS::RDS::DBInstance resource type in the CloudFormation template, update the DBEngineVersion property to the latest MySQL database version. Trigger an Update Stack operation in CloudFormation. Launch a new RDS Read Replica with the same properties as the primary database instance that will be upgraded. Finally, perform a second Update Stack operation.

In the AWS::RDS::DBInstance resource type in the CloudFormation template, update the EngineVersion property to the latest MySQL database version. Create a second application stack and launch a new Read Replica with the same properties as the primary database instance that will be upgraded. Finally, perform an Update Stack operation in CloudFormation.

Explanation:-If your MySQL DB instance is currently in use with a production application, you can follow a procedure to upgrade the database version for your DB instance that can reduce the amount of downtime for your application.

Periodically, Amazon RDS performs maintenance on Amazon RDS resources. Maintenance most often involves updates to the DB instance's underlying hardware, underlying operating system (OS), or database engine version. Updates to the operating system most often occur for security issues and should be done as soon as possible.

Some maintenance items require that Amazon RDS take your DB instance offline for a short time. Maintenance items that require a resource to be offline include required operating system or database patching. Required patching is automatically scheduled only for patches that are related to security and instance reliability. Such patching occurs infrequently (typically once every few months) and seldom requires more than a fraction of your maintenance window.

When you modify the database engine for your DB instance in a Multi-AZ deployment, Amazon RDS upgrades both the primary and secondary DB instances at the same time. In this case, the database engine for the entire Multi-AZ deployment is shut down during the upgrade.

Hence, the correct answer is: In the AWS::RDS::DBInstance resource type in the CloudFormation template, update the EngineVersion property to the latest MySQL database version. Create a second application stack and launch a new Read Replica with the same properties as the primary database instance that will be upgraded. Finally, perform an Update Stack operation in CloudFormation.

The option that says: In the AWS::RDS::DBInstance resource type in the CloudFormation template, update the DBEngineVersion property to the latest MySQL database version. Trigger an Update Stack operation in CloudFormation. Launch a new RDS Read Replica with the same properties as the primary database instance that will be upgraded. Finally, perform a second Update Stack operation is incorrect because this solution may possibly experience downtime since you trigger the Update Stack operation first before creating a Read Replica, which you could have used as a backup instance in the event of update failures. Remember that when you modify the database engine for your RDS Multi-AZ instance, the database engine for the entire Multi-AZ deployment is shut down during the upgrade. In addition, there is no such DBEngineVersion property.

The option that says: In the AWS::RDS::DBInstance resource type in the CloudFormation template, update the DBEngineVersion property to the latest MySQL database version. Launch a new RDS Read Replica with the same properties as the primary database instance that will be upgraded. Finally, trigger an Update Stack operation in CloudFormation is incorrect because there is no such DBEngineVersion property in CloudFormation. You have to use the EngineVersion property instead.

The option that says: In the AWS::RDS::DBInstance resource type in the CloudFormation template, update the EngineVersion property to the latest MySQL database version. Afterwards, directly trigger an Update Stack operation in CloudFormation is incorrect because if the database upgrade fails, your entire system will be unavailable since you have no Read Replicas that you can use as a failover. Remember that when you modify the database engine for your DB instance in a Multi-AZ deployment configuration, Amazon RDS upgrades both the primary and secondary DB instances at the same time which means that the RDS shuts down the whole database.

References:

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-rds-database-instance.html

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_UpgradeDBInstance.Maintenance.html

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_UpgradeDBInstance.MySQL.html#USER_UpgradeDBInstance.MySQL.ReducedDowntime

Check out these AWS CloudFormation and Amazon RDS Cheat Sheets:

https://tutorialsdojo.com/aws-cheat-sheet-aws-cloudformation/

https://tutorialsdojo.com/aws-cheat-sheet-amazon-relational-database-service-amazon-rds/

Q41) A government-sponsored health service is running their web application containing information about the clinics, hospitals, medical specialists and other medical services in the country. They also have a set of public web services which enable third-party companies to search medical data for their respective applications and clients. Lambda functions are used for the public APIs. For its database-tier, an Amazon DynamoDB table stores all of the data with an Amazon ES domain which supports the search feature and stores the indexes. A DevOps engineer has been instructed to ensure that in the event of a failed deployment, there should be no downtime and a system should be in place to prevent subsequent deployments. The service must strictly maintain full capacity during API deployment without any reduced capacity to avoid degradation of service.
How can the engineer meet the above requirements in the MOST efficient way?

Deploy the DynamoDB tables, Lambda functions, and Amazon ES domain using AWS CloudFormation. Deploy changes with an AWS CodeDeploy blue/green deployment. Host the web application in Amazon S3 and enable cross-region replication.

Deploy the DynamoDB tables, Lambda functions, and Amazon ES domain using AWS CloudFormation. Deploy changes with an AWS CodeDeploy in-place deployment. Host the web application in AWS Elastic Beanstalk and set the deployment policy to Rolling.

Deploy the DynamoDB tables, Lambda functions, and Amazon ES domain using AWS CloudFormation. Deploy changes with an AWS CodeDeploy blue/green deployment. Host the web application in AWS Elastic Beanstalk and set the deployment policy to Immutable.

Explanation:-AWS Elastic Beanstalk provides several options for how deployments are processed, including deployment policies (All at once, Rolling, Rolling with additional batch, and Immutable) and options that let you configure batch size and health check behavior during deployments. By default, your environment uses all-at-once deployments. If you created the environment with the EB CLI and it's an automatically scaling environment (you didn't specify the --single option), it uses rolling deployments.

With rolling deployments, Elastic Beanstalk splits the environment's EC2 instances into batches and deploys the new version of the application to one batch at a time, leaving the rest of the instances in the environment running the old version of the application. During a rolling deployment, some instances serve requests with the old version of the application, while instances in completed batches serve other requests with the new version.

To maintain full capacity during deployments, you can configure your environment to launch a new batch of instances before taking any instances out of service. This option is known as a rolling deployment with an additional batch. When the deployment completes, Elastic Beanstalk terminates the additional batch of instances."

Immutable deployments perform an immutable update to launch a full set of new instances running the new version of the application in a separate Auto Scaling group, alongside the instances running the old version. Immutable deployments can prevent issues caused by partially completed rolling deployments. If the new instances don't pass health checks, Elastic Beanstalk terminates them, leaving the original instances untouched.

Hence, the correct answer is: Deploy the DynamoDB tables, Lambda functions, and Amazon ES domain using AWS CloudFormation. Deploy changes with an AWS CodeDeploy blue/green deployment. Host the web application in AWS Elastic Beanstalk and set the deployment policy to Immutable.

The option that says: Deploy the DynamoDB tables, Lambda functions, and Amazon ES domain using AWS CloudFormation. Deploy changes with an AWS CodeDeploy blue/green deployment. Host the web application in AWS Elastic Beanstalk and set the deployment policy to All at Once is incorrect because this policy deploys the new version to all instances simultaneously which means that the instances in your environment are out of service for a short time while the deployment occurs.

The option that says: Deploy the DynamoDB tables, Lambda functions, and Amazon ES domain using AWS CloudFormation. Deploy changes with an AWS CodeDeploy in-place deployment. Host the web application in AWS Elastic Beanstalk and set the deployment policy to Rolling is incorrect because this policy will deploy the new version in batches where each batch is taken out of service during the deployment phase, reducing your environment's capacity by the number of instances in a batch.
The option that says: Deploy the DynamoDB tables, Lambda functions, and Amazon ES domain using AWS CloudFormation. Deploy changes with an AWS CodeDeploy blue/green deployment. Host the web application in Amazon S3 and enable cross-region replication is incorrect because you can't host a dynamic web application in Amazon S3.
References:
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.rolling-version-deploy.html
https://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html#welcome-deployment-overview
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/environmentmgmt-updates-immutable.html
Check out this AWS Elastic Beanstalk Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-elastic-beanstalk/
● Deploy the DynamoDB tables, Lambda functions, and Amazon ES domain using AWS CloudFormation. Deploy changes with an AWS CodeDeploy blue/green deployment. Host the web application in AWS Elastic Beanstalk and set the deployment policy to All at Once.

Q42) A company is planning to deploy a new version of their legacy application in AWS which is deployed to an Auto Scaling group of EC2 instances with a Classic Load Balancer in front. To avoid any disruption of their services, they need to implement canary testing first before all of the traffic is shifted to the new application version.
Which of the following solutions can meet the this requirement?

● Set up an Amazon API Gateway private integration with the Classic Load Balancer and prepare a separate stage for the new application version. Configure the API Gateway to do a canary release deployment.
● Do a Canary deployment using CodeDeploy with a CodeDeployDefault.LambdaCanary10Percent30Minutes deployment configuration.
● Prepare another stack that consists of a Classic Load Balancer and Auto Scaling group which contains the new application version for blue/green environments. Use an Amazon CloudFront web distribution to adjust the weight of the incoming traffic to the two Classic Load Balancers.
✅ Prepare another stack that consists of a Classic Load Balancer and Auto Scaling group which contains the new application version for blue/green environments. Create weighted Alias A records in Route 53 for the two Classic Load Balancers to adjust the traffic.
Explanation:-The purpose of a canary deployment is to reduce the risk of deploying a new version that impacts the workload. The method will incrementally deploy the new version, making it visible to new users in a slow fashion. As you gain confidence in the deployment, you will deploy it to replace the current version in its entirety.
To properly implement the canary deployment, you should do the following steps:
- Use a router or load balancer that allows you to send a small percentage of users to the new version.
- Use a dimension on your KPIs to indicate which version is reporting the metrics.
- Use the metric to measure the success of the deployment; this indicates whether the deployment should continue or rollback.
- Increase the load on the new version until either all users are on the new version or you have fully rolled back.
Hence, the correct answer is: Prepare another stack that consists of a Classic Load Balancer and Auto Scaling group which contains the new application version for blue/green environments. Create weighted Alias A records in Route 53 for the two Classic Load Balancers to adjust the traffic.
The option that says: Do a Canary deployment using CodeDeploy with a CodeDeployDefault.LambdaCanary10Percent30Minutes deployment configuration is incorrect because this specific configuration type is only applicable for Lambda functions and for the applications hosted in an Auto Scaling group.
The option that says: Prepare another stack that consists of a Classic Load Balancer and Auto Scaling group which contains the new application version for blue/green environments. Use an Amazon CloudFront web distribution to adjust the weight of the incoming traffic to the two Classic Load Balancers is incorrect because you can't use CloudFront to adjust the weight of the incoming traffic to your application. You should use Route 53 instead.
The option that says: Set up an Amazon API Gateway private integration with the Classic Load Balancer and prepare a separate stage for the new application version. Configure the API Gateway to do a canary release deployment is incorrect because you can only integrate a Network Load Balancer to your Amazon API Gateway. Moreover, this service is only applicable for APIs and not for full-fledged web applications.
References:
https://wa.aws.amazon.com/wat.concept.canary-deployment.en.html
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-to-elb-load-balancer.html
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resource-record-sets-values-weighted.html

---

Q43) A software development company is doing an all-in migration of their on-premises resources to AWS. The company has a hybrid architecture which comprises of over a thousand on-premises VMware servers and a few EC2 instances in their VPC. The company is using a VMWare vCenter Server for data center management of their vSphere environments and virtual servers. A DevOps engineer is tasked to implement a solution that will collect various information from their on-premises and EC2 instances such as operating system details, MAC address, IP address and many others. The Operations team should also be able to analyze the collected data in a visual format.
Which of the following is the MOST appropriate solution that the engineer should implement with the LEAST amount of effort?

● Develop a custom python script and install them on both VMware servers and EC2 instances to collect all of the required information. Push the data to a centralized S3 bucket. Use VMware vSphere to collect the data from your on-premises resources and push the results into a file gateway in order to store the data in Amazon S3. Use Amazon Athena on the S3 bucket to analyze the data.
● Register all of the on-premises virtual machines as well as the EC2 instances to AWS Service Catalog where all the required information such as the operating system details and many others will be automatically populated. Export the consolidated data from AWS Service Catalog to an Amazon S3 bucket and then use Amazon QuickSight for analytics.
● Install the AWS Systems Manager Agent (SSM Agent) on all on-premises virtual machines and the EC2 instances. Utilize the AWS Systems Manager Inventory service to provide visibility into your Amazon EC2 and on-premises computing environment. Set up a resource data sync to an S3 bucket in order to analyze the data with Amazon QuickSight.
✅ Using the AWS Application Discovery Service, deploy the Agentless Discovery Connector in an OVA file format to your VMware vCenter and then install the AWS Discovery Agents on the EC2 instances to collect the required data. Use the AWS Migration Hub Dashboard to analyze your hybrid infrastructure.
Explanation:-AWS Application Discovery Service helps you plan your migration to the AWS cloud by collecting usage and configuration data about your on-premises servers. Application Discovery Service is integrated with AWS Migration Hub, which simplifies your migration tracking. After performing discovery, you can view the discovered servers, group them into applications, and then track the migration status of each application from the Migration Hub console. The discovered data can be exported for analysis in Microsoft Excel or AWS analysis tools such as Amazon Athena and Amazon QuickSight.
Using Application Discovery Service APIs, you can export the system performance and utilization data for your discovered servers. You can input this data into your cost model to compute the cost of running those servers in AWS. Additionally, you can export the network connections and process

data to understand the network connections that exist between servers. This will help you determine the network dependencies between servers and group them into applications for migration planning.
Application Discovery Service offers two ways of performing discovery and collecting data about your on-premises servers:
- Agentless discovery can be performed by deploying the AWS Agentless Discovery Connector (OVA file) through your VMware vCenter. After the Discovery Connector is configured, it identifies virtual machines (VMs) and hosts associated with vCenter. The Discovery Connector collects the following static configuration data: Server hostnames, IP addresses, MAC addresses, disk resource allocations. Additionally, it collects the utilization data for each VM and computes average and peak utilization for metrics such as CPU, RAM, and Disk I/O. You can export a summary of the system performance information for all the VMs associated with a given VM host and perform a cost analysis of running them in AWS.
- Agent-based discovery can be performed by deploying the AWS Application Discovery Agent on each of your VMs and physical servers. The agent installer is available for both Windows and Linux operating systems. It collects static configuration data, detailed time-series system-performance information, inbound and outbound network connections, and processes that are running. You can export this data to perform a detailed cost analysis and to identify network connections between servers for grouping servers as applications.
The Agentless discovery uses the AWS Discovery Connector which is a VMware appliance that can collect information only about VMware virtual machines (VMs). This mode doesn't require you to install a connector on each host. You install the Discovery Connector as a VM in your VMware vCenter Server environment using an Open Virtualization Archive (OVA) file. Because the Discovery Connector relies on VMware metadata to gather server information regardless of operating system, it minimizes the time required for initial on-premises infrastructure assessment.
Hence, the correct answer is: Using the AWS Application Discovery Service, deploy the Agentless Discovery Connector in an OVA file format to your VMware vCenter and then install the AWS Discovery Agents on the EC2 instances to collect the required data. Use the AWS Migration Hub Dashboard to analyze your hybrid infrastructure.
The option that says: Develop a custom python script and install them on both VMware servers and EC2 instances to collect all of the required information. Push the data to a centralized S3 bucket. Use VMware vSphere to collect the data from your on-premises resources and push the results into a file gateway in order to store the data in Amazon S3. Use Amazon Athena on the S3 bucket to analyze the data is incorrect because although this solution may work, it takes a lot of effort to develop a custom python script as well as to manually install it to over a thousand VMWare servers on the company's on-premises data center.
The option that says: Register all of the on-premises virtual machines as well as the EC2 instances to AWS Service Catalog where all the required information such as the operating system details, and many others will be automatically populated. Export the consolidated data from AWS Service Catalog to an Amazon S3, bucket and then use Amazon QuickSight for analytics is incorrect because the AWS Service Catalog service doesn't have the capability to integrate with the on-premises VMWare servers. This service only allows organizations to create and manage catalogs of IT services that are approved for use on AWS.
The option that says: Install the AWS Systems Manager Agent (SSM Agent) on all on-premises virtual machines and the EC2 instances. Utilize the AWS Systems Manager Inventory service to provide visibility into your Amazon EC2 and on-premises computing environment. Set up a resource data sync to an S3 bucket in order to analyze the data with Amazon QuickSight is incorrect because although this solution is valid, this is definitely not the one that can be implemented with the least amount of effort. Although you can use the SSM Agent to fetch all of the required information about your servers, the task of installing it to each and every on-premises VMWare server is a herculean task which entails a lot of execution time. Moreover, the scenario mentioned that the company is doing an all-in migration of their on-premises resources to AWS which means that installing the SSM agent is not appropriate. A better solution would be to use Agentless Discovery Connector of the AWS Application Discovery Service to your on-premises VMware vCenter, which can easily fetch the required information from hundreds of VMware servers.
References:
https://docs.aws.amazon.com/application-discovery/latest/userguide/what-is-appdiscovery.html
https://docs.aws.amazon.com/application-discovery/latest/userguide/discovery-connector.html
https://docs.aws.amazon.com/application-discovery/latest/userguide/dashboard.html
Tutorials Dojo's AWS Certified DevOps Engineer Professional Exam Study Guide:
https://tutorialsdojo.com/aws-cheat-sheet-aws-certified-devops-engineer-professional/

Q44) A company that develops smart home devices has a set of serverless APIs that uses several independent AWS Lambda functions that handles web, mobile, Internet of Things (IoT), and 3rd party API requests. The current CI/CD pipeline builds, tests, packages, and deploys each Lambda function in sequence using AWS CodePipeline and AWS CodeBuild. A CloudWatch Events rule is used to monitor and ensure that the pipeline execution starts promptly after a code change has been made. The SysOps team monitors all of the AWS resources as well as the performance of each pipeline. They noticed that for the past few days, the pipeline takes too long to finish the build and deployment process. The team instructed a DevOps Engineer to implement a solution that will expedite the deployment process of the independent Lambda functions.
Which of the following is the MOST suitable solution that the DevOps Engineer should implement?

- Set up a custom CodeBuild execution environment with multiprocessing option that runs builds in parallel.
- Enable local caching in AWS CodeBuild using a Docker layer cache mode.
- Upgrade the compute type of the build environment in CodeBuild pipeline with a higher memory, vCPUs, and disk space.
- ✅ Execute actions for each Lambda function in parallel by setting up a configuration that specifies the same runOrder value in CodePipeline.

Explanation:-In AWS CodePipeline, an action is part of the sequence in a stage of a pipeline. It is a task performed on the artifact in that stage. Pipeline actions occur in a specified order, in sequence or in parallel, as determined in the configuration of the stage.
CodePipeline provides support for six types of actions:
- Source
- Build
- Test
- Deploy
- Approval
- Invoke
By default, any pipeline you successfully create in AWS CodePipeline has a valid structure. However, if you manually create or edit a JSON file to create a pipeline or update a pipeline from the AWS CLI, you might inadvertently create a structure that is not valid. The following reference can help you better understand the requirements for your pipeline structure and how to troubleshoot issues.
The default runOrder value for an action is 1. The value must be a positive integer (natural number). You cannot use fractions, decimals, negative numbers, or zero. To specify a serial sequence of actions, use the smallest number for the first action and larger numbers for each of the rest of the actions in sequence. To specify parallel actions, use the same integer for each action you want to run in parallel.
For example, if you want three actions to run in sequence in a stage, you would give the first action the runOrder value of 1, the second action the runOrder value of 2, and the third the runOrder value of 3. However, if you want the second and third actions to run in parallel, you would give the first action the runOrder value of 1 and both the second and third actions the runOrder value of 2.
The numbering of serial actions do not have to be in strict sequence. For example, if you have three actions in a sequence and decide to remove the second action, you do not need to renumber the runOrder value of the third action. Because the runOrder value of that action (3) is higher than the runOrder value of the first action (1), it runs serially after the first action in the stage.
Hence, the correct answer is: Execute actions for each Lambda function in parallel by setting up a configuration that specifies the same runOrder

value in CodePipeline.
The option that says: Upgrade the compute type of the build environment in CodeBuild pipeline with a higher memory, vCPUs, and disk space is incorrect because although it may help speed up the build time, it will only improve the performance of CodeBuild and not the entire pipeline. A better solution is to run the tasks in parallel in CodePipeline.
The option that says: Set up a custom CodeBuild execution environment with multiprocessing option that runs builds in parallel is incorrect because there is no multiprocessing option in CodeBuild. You can upgrade the compute type of the build environment or use local cache, but there is no such thing as multiprocessing in CodeBuild.
The option that says: Enable local caching in AWS CodeBuild using a Docker layer cache mode is incorrect because although using a local cache can help expedite the build process, the use of Docker layer cache mode is only applicable for containerized applications and not for Lambda functions.
References:
https://docs.aws.amazon.com/codepipeline/latest/userguide/reference-pipeline-structure.html
https://docs.aws.amazon.com/codepipeline/latest/userguide/actions.html
Check out this AWS CodePipeline Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-codepipeline/

Q45) A leading software development company has various web applications hosted in an Auto Scaling group of EC2 instances which are designed for high availability and fault tolerance. They are using AWS CloudFormation to easily manage their cloud infrastructure as code as well as for deployment. Currently, they have to manually update their CloudFormation templates for every new available AMI of their application. This procedure is prone to human errors and entails a high management overhead on their deployment process.
Which of the following is the MOST suitable and cost-effective solution that the DevOps engineer should implement to automate this process?

- Launch an EC2 instance to run a custom shell script every hour to check for new AMIs. The script should update the launch configuration resource block of the CloudFormation template with the new AMI ID if there are new ones available.
- Configure the AWS CloudFormation template to use conditional statements to check if new AMIs are available. Fetch the new AMI ID using the cfn-init helper script and reference it in the launch configuration resource block.
- ✅ Pull the new AMI IDs using an AWS Lambda-backed custom resource in the CloudFormation template. Reference the AMI ID that the custom resource fetched in the launch configuration resource block.

Explanation:-Custom resources enable you to write custom provisioning logic in templates that AWS CloudFormation runs anytime you create, update (if you changed the custom resource), or delete stacks. For example, you might want to include resources that aren't available as AWS CloudFormation resource types. You can include those resources by using custom resources. That way you can still manage all your related resources in a single stack.
Use the AWS::CloudFormation::CustomResource or alternatively, the Custom:: resource type to define custom resources in your templates. Custom resources require one property: the service token, which specifies where AWS CloudFormation sends requests to, such as an Amazon SNS topic. When you associate a Lambda function with a custom resource, the function is invoked whenever the custom resource is created, updated, or deleted. AWS CloudFormation calls a Lambda API to invoke the function and to pass all the request data (such as the request type and resource properties) to the function. The power and customizability of Lambda functions in combination with AWS CloudFormation enable a wide range of scenarios, such as dynamically looking up AMI IDs during stack creation, or implementing and using utility functions, such as string reversal functions.
AWS CloudFormation templates that declare an Amazon Elastic Compute Cloud (Amazon EC2) instance must also specify an Amazon Machine Image (AMI) ID, which includes an operating system and other software and configuration information used to launch the instance. The correct AMI ID depends on the instance type and region in which you're launching your stack. And IDs can change regularly, such as when an AMI is updated with software updates.
Normally, you might map AMI IDs to specific instance types and regions. To update the IDs, you manually change them in each of your templates. By using custom resources and AWS Lambda (Lambda), you can create a function that gets the IDs of the latest AMIs for the region and instance type that you're using so that you don't have to maintain mappings.
Hence, the correct answer is: Use an AWS Lambda-backed custom resource in the CloudFormation template to pull the new AMI IDs. Reference the AMI ID that the custom resource fetched in the launch configuration resource block.
The option that says: Configure the CloudFormation template to use AMI mappings. Integrate AWS Lambda and Amazon CloudWatch Events to create a function that regularly runs every hour to detect new AMIs as well as update the mapping in the template. Reference the AMI mappings in the launch configuration resource block is incorrect because although this solution may work, it is not economical to set up a scheduled job that runs every 1 hour just to detect new AMIs and update your CloudFormation templates. This is an inefficient solution since the AMIs are not updated that often to begin with which means that most of the hourly processing done by the Lambda function will yield no result. A better design would be to use AWS Lambda-backed custom resource instead in CloudFormation which will fetch the new AMI IDs upon deployment.
The option that says: Configure the AWS CloudFormation template to use conditional statements to check if new AMIs are available. Fetch the new AMI ID using the cfn-init helper script and reference it in the launch configuration resource block is incorrect because a cfn-init helper script is primarily used to fetch metadata, install packages and start/stop services to your EC2 instances that are already running. A better solution to implement here is to use AWS Lambda-backed custom resource in the CloudFormation template to pull the new AMI IDs.
The option that says: Launch an EC2 instance to run a custom shell script every hour to check for new AMIs. The script should update the launch configuration resource block of the CloudFormation template with the new AMI ID if there are new ones available is incorrect because although this solution may work, it includes the unnecessary cost of running an EC2 instance which is charged 24/7 but only does the actual processing every hour. This can simply be replaced by using an AWS Lambda-backed custom resource in CloudFormation.
References:
https://aws.amazon.com/blogs/devops/faster-auto-scaling-in-aws-cloudformation-stacks-with-lambda-backed-custom-resources/
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/walkthrough-custom-resources-lambda-lookup-amiids.html
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-custom-resources-lambda.html
Check out this AWS CloudFormation Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-cloudformation/

- Configure the CloudFormation template to use AMI mappings. Integrate AWS Lambda and Amazon CloudWatch Events to create a function that regularly runs every hour to detect new AMIs as well as update the mapping in the template. Reference the AMI mappings in the launch configuration resource block.

Q46) A company wants to implement a continuous deployment workflow that will facilitate the process for source code promotion in their development, staging, and production environments in AWS. In the event of system degradation or failure, they should also have the ability to roll back the recent deployment of their application.
Which of the following CI/CD designs is the MOST suitable one to implement and will incur the LEAST amount of downtime?

- Create a repository in AWS CodeCommit for the development environment and another one for the production environment. Set up AWS

CodeBuild to build and merge the two repositories. Do a blue/green deployment using AWS CodeDeploy to deploy the latest code in production.

Create an Amazon ECR repository and then create a development branch to hold merged changes made by the developers. Set up AWS CodeBuild to build and test the code stored in the development branch which is triggered to run on every new commit. Merge to the master branch using pull requests that will be approved by senior developers. To deploy the latest code to the production environment, set up a blue/green deployment using AWS CodeDeploy.

Create several repositories in AWS CodeCommit for each of their developers and then create a centralized development branch to hold merged changes from each of the developer's repository. Set up AWS CodeBuild to build and test the code stored in the development branch. Merge to the master branch using pull requests that will be approved by senior developers. To deploy the latest code to the production environment, set up a blue/green deployment using AWS CodeDeploy.

Create a single repository in AWS CodeCommit and create a development branch to hold merged changes. Set up AWS CodeBuild to build and test the code stored in the development branch which is triggered to run on every new commit. Merge to the master branch using pull requests that will be approved by senior developers. To deploy the latest code to the production environment, set up a blue/green deployment using AWS CodeDeploy.

Explanation:-A repository is the fundamental version control object in CodeCommit. It's where you securely store code and files for your project. It also stores your project history, from the first commit through the latest changes. You can share your repository with other users so you can work together on a project. If you add AWS tags to repositories, you can set up notifications so that repository users receive email about events (for example, another user commenting on code). You can also change the default settings for your repository, browse its contents, and more. You can create triggers for your repository so that code pushes or other events trigger actions, such as emails or code functions. You can even configure a repository on your local computer (a local repo) to push your changes to more than one repository.

In designing your CI/CD process in AWS, you can use a single repository in AWS CodeCommit and create different branches for development, master, and release. You can use CodeBuild to build your application and run tests to verify that all of the core features of your application are working. For deployment, you can either select an in-place or blue/green deployment using CodeDeploy.

Hence, the correct answer is: Create a single repository in AWS CodeCommit and create a development branch to hold merged changes. Set up AWS CodeBuild to build and test the code stored in the development branch which is triggered to run on every new commit. Merge to the master branch using pull requests that will be approved by senior developers. To deploy the latest code to the production environment, set up a blue/green deployment using AWS CodeDeploy.

The option that says: Create several repositories in AWS CodeCommit for each of their developers and then create a centralized development branch to hold merged changes from each of the developer's repository. Set up AWS CodeBuild to build and test the code stored in the development branch. Merge to the master branch using pull requests that will be approved by senior developers. To deploy the latest code to the production environment, set up a blue/green deployment using AWS CodeDeploy is incorrect because creating a separate repository for each developer is absurd since they can simply clone the code instead. A single repository will suffice in this scenario which can have several branches for development and production deployment purposes.

The option that says: Create a repository in AWS CodeCommit for the development environment and another one for the production environment. Set up AWS CodeBuild to build and merge the two repositories. Do a blue/green deployment using AWS CodeDeploy to deploy the latest code in production is incorrect because you don't need to create two CodeCommit repositories for one application. Instead, you can just create at least two different branches to separate your development and production code.

The option that says: Create an Amazon ECR repository and then create a development branch to hold merged changes made by the developers. Set up AWS CodeBuild to build and test the code stored in the development branch which is triggered to run on every new commit. Merge to the master branch using pull requests that will be approved by senior developers. To deploy the latest code to the production environment, set up a blue/green deployment using AWS CodeDeploy is incorrect because Amazon ECR is a fully-managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images. This is not a suitable service to be used to store your application code.

References:

https://aws.amazon.com/devops/continuous-integration/

https://aws.amazon.com/devops/continuous-delivery/

https://docs.aws.amazon.com/codecommit/latest/userguide/repositories.html

Check out this AWS CodeCommit Cheat Sheet:

https://tutorialsdojo.com/aws-cheat-sheet-aws-codecommit/

Tutorials Dojo's AWS Certified DevOps Engineer Professional Exam Study Guide:

https://tutorialsdojo.com/aws-cheat-sheet-aws-certified-devops-engineer-professional/

Q47) A global cryptocurrency trading company has a suite of web applications hosted in an Auto Scaling group of Amazon EC2 instances across multiple Available Zones behind an Application Load Balancer to distribute the incoming traffic. The Auto Scaling group is configured to use Elastic Load Balancing health checks for scaling instead of the default EC2 status checks. However, there are several occasions when some instances are automatically terminated after failing the HTTPS health checks in the ALB that purges all the logs stored in the instance. To improve system monitoring, a DevOps Engineer must implement a solution that collects all of the application and server logs effectively. The Operations team should be able to perform a root cause analysis based on the logs, even if the Auto Scaling group immediately terminated the instance.
How can the DevOps Engineer automate the log collection from the Amazon EC2 instances with the LEAST amount of effort?

Delay the termination of unhealthy Amazon EC2 instances by adding a lifecycle hook to your Auto Scaling group to move instances in the Terminating state to the Terminating:Wait state. Use AWS Step Functions to collect the application logs and send them to a CloudWatch Log group. Resume the instance termination once all the logs are sent to CloudWatch Logs.

Delay the termination of unhealthy Amazon EC2 instances by adding a lifecycle hook to your Auto Scaling group to move instances in the Terminating state to the Terminating:Wait state. Set up a CloudWatch Events rule for the EC2 Instance Terminate Successful Auto Scaling Event with an associated Lambda function. Use the AWS Systems Manager Run Command to run a script that collects and uploads the application logs from the instance to a CloudWatch Logs group. Resume the instance termination once all the logs are sent.

Delay the termination of unhealthy Amazon EC2 instances by adding a lifecycle hook to your Auto Scaling group to move instances in the Terminating state to the Terminating:Wait state. Set up a CloudWatch Events rule for the EC2 Instance-terminate Lifecycle Action Auto Scaling Event with an associated Lambda function. Trigger the CloudWatch agent to push the application logs and then resume the instance termination once all the logs are sent to CloudWatch Logs.

Explanation:-The EC2 instances in an Auto Scaling group have a path, or lifecycle, that differs from that of other EC2 instances. The lifecycle starts when the Auto Scaling group launches an instance and puts it into service. The lifecycle ends when you terminate the instance, or the Auto Scaling group takes the instance out of service and terminates it.

You can add a lifecycle hook to your Auto Scaling group so that you can perform custom actions when instances launch or terminate.

When Amazon EC2 Auto Scaling responds to a scale out event, it launches one or more instances. These instances start in the Pending state. If you added an autoscaling:EC2_INSTANCE_LAUNCHING lifecycle hook to your Auto Scaling group, the instances move from the Pending state to the Pending:Wait state. After you complete the lifecycle action, the instances enter the Pending:Proceed state. When the instances are fully configured, they are attached to the Auto Scaling group and they enter the InService state.

When Amazon EC2 Auto Scaling responds to a scale in event, it terminates one or more instances. These instances are detached from the Auto Scaling group and enter the Terminating state. If you added an autoscaling:EC2_INSTANCE_TERMINATING lifecycle hook to your Auto Scaling

group, the instances move from the Terminating state to the Terminating:Wait state. After you complete the lifecycle action, the instances enter the Terminating:Proceed state. When the instances are fully terminated, they enter the Terminated state.
Using CloudWatch agent is the most suitable tool to use to collect the logs. The unified CloudWatch agent enables you to do the following:
- Collect more system-level metrics from Amazon EC2 instances across operating systems. The metrics can include in-guest metrics, in addition to the metrics for EC2 instances. The additional metrics that can be collected are listed in Metrics Collected by the CloudWatch Agent.
- Collect system-level metrics from on-premises servers. These can include servers in a hybrid environment as well as servers not managed by AWS.
- Retrieve custom metrics from your applications or services using the StatsD and collectd protocols. StatsD is supported on both Linux servers and servers running Windows Server. collectd is supported only on Linux servers.
- Collect logs from Amazon EC2 instances and on-premises servers, running either Linux or Windows Server.
You can store and view the metrics that you collect with the CloudWatch agent in CloudWatch just as you can with any other CloudWatch metrics. The default namespace for metrics collected by the CloudWatch agent is CWAgent, although you can specify a different namespace when you configure the agent.
Hence, the correct answer is: Delay the termination of unhealthy Amazon EC2 instances by adding a lifecycle hook to your Auto Scaling group to move instances in the Terminating state to the Terminating:Wait state. Set up a CloudWatch Events rule for the EC2 Instance-terminate Lifecycle Action Auto Scaling Event with an associated Lambda function. Trigger the CloudWatch agent to push the application logs and then resume the instance termination once all the logs are sent to CloudWatch Logs.
The option that says: Delay the termination of unhealthy Amazon EC2 instances by adding a lifecycle hook to your Auto Scaling group to move instances in the Terminating state to the Pending:Wait state. Set up a CloudWatch Events rule for the EC2 Instance-terminate Lifecycle Action Auto Scaling Event with an associated Lambda function. Use the AWS Systems Manager Automation to run a script that collects and uploads the application logs from the instance to a CloudWatch Logs group. Resume the instance termination once all the logs are sent is incorrect because the Pending:Wait state refers to the scale-out action in Amazon EC2 Auto Scaling and not for scale-in or for terminating the instances.
The option that says: Delay the termination of unhealthy Amazon EC2 instances by adding a lifecycle hook to your Auto Scaling group to move instances in the Terminating state to the Terminating:Wait state. Use AWS Step Functions to collect the application logs and send them to a CloudWatch Log group. Resume the instance termination once all the logs are sent to CloudWatch Logs is incorrect because using AWS Step Functions is inappropriate in collecting the logs from your EC2 instances. You should use a CloudWatch agent instead.
The option that says: Delay the termination of unhealthy Amazon EC2 instances by adding a lifecycle hook to your Auto Scaling group to move instances in the Terminating state to the Terminating:Wait state. Set up a CloudWatch Events rule for the EC2 Instance Terminate Successful Auto Scaling Event with an associated Lambda function. Use the AWS Systems Manager Run Command to run a script that collects and uploads the application logs from the instance to a CloudWatch Logs group. Resume the instance termination once all the logs are sent is incorrect because although this solution could work, it entails a lot of effort to write a custom script that the AWS Systems Manager Run Command will run. Remember that the scenario asks for a solution that you can implement with the least amount of effort. This solution can be simplified by automatically uploading the logs using a CloudWatch Agent. You have to use the EC2 Instance-terminate Lifecycle Action event instead.
References:
https://docs.aws.amazon.com/autoscaling/ec2/userguide/AutoScalingGroupLifecycle.html
https://docs.aws.amazon.com/autoscaling/ec2/userguide/cloud-watch-events.html#terminate-successful
https://aws.amazon.com/premiumsupport/knowledge-center/auto-scaling-delay-termination/
Check out this AWS Auto Scaling Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-auto-scaling/
● Delay the termination of unhealthy Amazon EC2 instances by adding a lifecycle hook to your Auto Scaling group to move instances in the Terminating state to the Pending:Wait state. Set up a CloudWatch Events rule for the EC2 Instance-terminate Lifecycle Action Auto Scaling Event with an associated Lambda function. Use the AWS Systems Manager Automation to run a script that collects and uploads the application logs from the instance to a CloudWatch Logs group. Resume the instance termination once all the logs are sent.

Q48) A leading pharmaceutical company has several web applications and GraphQL APIs hosted in a fleet of Amazon EC2 instances. There is a private EC2 instance with no Internet access that needs to upload a new object to a private Amazon S3 bucket. However, you are getting an HTTP 403: Access Denied error whenever you try to fetch the objects from the instance. What are the possible causes for this issue? (Select TWO)

● The Cross-Region replication (CRR) feature, which is used to copy objects across Amazon S3 buckets in different AWS Regions, is enabled in the bucket.
● The bucket has a lifecycle policy that automatically moves data to the S3 Intelligent-Tiering storage class.
✅ The Amazon Virtual Private Cloud (Amazon VPC) endpoint policy is not properly configured.
Explanation:-A VPC endpoint enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by PrivateLink without requiring an Internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network.
Endpoints are virtual devices. They are horizontally scaled, redundant, and highly available VPC components that allow communication between instances in your VPC and services without imposing availability risks or bandwidth constraints on your network traffic.
There are two types of VPC endpoints: interface endpoints and gateway endpoints. You should create the type of VPC endpoint required by the supported service.
If an IAM user has permission with s3:PutObject action on an Amazon Simple Storage Service (Amazon S3) bucket and got an HTTP 403: Access Denied error when uploading an object, then there might be an issue with the bucket or VPC endpoint policy.
If the IAM user has the correct user permissions to upload to the bucket, then check the following policies for any settings that might be preventing the uploads:
- IAM user permission to s3:PutObjectAcl
- Conditions in the bucket policy
- Access allowed by an Amazon Virtual Private Cloud (Amazon VPC) endpoint policy
Hence, the correct answers are:
- The Amazon S3 bucket policy is not properly configured
- The Amazon Virtual Private Cloud (Amazon VPC) endpoint policy is not properly configured
The following options are incorrect since these will not cause an HTTP 403: Access Denied error:
- The Cross-Region replication (CRR) feature, which is used to copy objects across Amazon S3 buckets in different AWS Regions, is enabled in the bucket.
- The bucket has a lifecycle policy that automatically moves data to the S3 Intelligent-Tiering storage class
- Versioning is enabled in the Amazon S3 bucket.
References:
https://aws.amazon.com/premiumsupport/knowledge-center/s3-403-upload-bucket/
https://aws.amazon.com/premiumsupport/knowledge-center/s3-troubleshoot-403/
Tutorials Dojo's AWS Certified DevOps Engineer Professional Exam Study Guide:

https://tutorialsdojo.com/aws-cheat-sheet-aws-certified-devops-engineer-professional/

Versioning is enabled in the Amazon S3 bucket.

The Amazon S3 bucket policy is not properly configured.

Explanation:-A VPC endpoint enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by PrivateLink without requiring an Internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network.

Endpoints are virtual devices. They are horizontally scaled, redundant, and highly available VPC components that allow communication between instances in your VPC and services without imposing availability risks or bandwidth constraints on your network traffic.

There are two types of VPC endpoints: interface endpoints and gateway endpoints. You should create the type of VPC endpoint required by the supported service.

If an IAM user has permission with s3:PutObject action on an Amazon Simple Storage Service (Amazon S3) bucket and got an HTTP 403: Access Denied error when uploading an object, then there might be an issue with the bucket or VPC endpoint policy.

If the IAM user has the correct user permissions to upload to the bucket, then check the following policies for any settings that might be preventing the uploads:

- IAM user permission to s3:PutObjectAcl
- Conditions in the bucket policy
- Access allowed by an Amazon Virtual Private Cloud (Amazon VPC) endpoint policy

Hence, the correct answers are:

- The Amazon S3 bucket policy is not properly configured
- The Amazon Virtual Private Cloud (Amazon VPC) endpoint policy is not properly configured

The following options are incorrect since these will not cause an HTTP 403: Access Denied error:

- The Cross-Region replication (CRR) feature, which is used to copy objects across Amazon S3 buckets in different AWS Regions, is enabled in the bucket.
- The bucket has a lifecycle policy that automatically moves data to the S3 Intelligent-Tiering storage class
- Versioning is enabled in the Amazon S3 bucket.

References:

https://aws.amazon.com/premiumsupport/knowledge-center/s3-403-upload-bucket/

https://aws.amazon.com/premiumsupport/knowledge-center/s3-troubleshoot-403/

Tutorials Dojo's AWS Certified DevOps Engineer Professional Exam Study Guide:

https://tutorialsdojo.com/aws-cheat-sheet-aws-certified-devops-engineer-professional/

Q49) A company has a hybrid cloud architecture where its on-premises data center is connected to its multiple virtual private clouds in AWS using a Transit Gateway. They have a service-oriented architecture with several application services distributed across their VPCs as well on its local data center. Gathering logs from each service takes a significant amount of time, especially if one module experiences a system outage. To rectify this issue, aggregating the system logs from the on-premises and AWS servers should be implemented. There should also be a way to analyze the logs for audit and review purposes.
As a DevOps Engineer, which among the following options is the MOST cost-effective solution that entails the LEAST amount of effort to implement?

Install the Unified CloudWatch Logs agent to all AWS resources to collect the system and application logs. Develop a shell script that utilizes AWS CLI to upload the on-premises logs to an S3 bucket in a central account. Analyze the log data using a custom-made Amazon EMR cluster.

Install the Unified CloudWatch Logs agent to all on-premises and AWS resources to collect the system and application logs. Consolidate all of the collected logs to your on-premises file server. Develop a custom-built solution that uses an open-source ELK stack running Elasticsearch, Logstash, and Kibana to analyze the logs.

Install the Unified CloudWatch Logs agent to all AWS resources to collect the system and application logs. Develop a shell script that utilizes AWS CLI to upload the on-premises logs to an S3 bucket for individual accounts. Analyze the log data using Amazon Macie.

Install the Unified CloudWatch Logs agent to all on-premises and AWS resources to collect the system and application logs. Store the collected data to an Amazon S3 bucket in a central account. Set up an Amazon S3 trigger that invokes a Lambda function to analyze the logs as well as detect any irregularities. Analyze the log data using Amazon Athena.

Explanation:-Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real-time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications. The CloudWatch home page automatically displays metrics about every AWS service you use. You can additionally create custom dashboards to display metrics about your custom applications and display custom collections of metrics that you choose.

The unified CloudWatch agent enables you to do the following:

- Collect more system-level metrics from Amazon EC2 instances across operating systems. The metrics can include in-guest metrics, in addition to the metrics for EC2 instances. The additional metrics that can be collected are listed in Metrics Collected by the CloudWatch Agent.
- Collect system-level metrics from on-premises servers. These can include servers in a hybrid environment as well as servers not managed by AWS.
- Retrieve custom metrics from your applications or services using the StatsD and collectd protocols. StatsD is supported on both Linux servers and servers running Windows Server. collectd is supported only on Linux servers.
- Collect logs from Amazon EC2 instances and on-premises servers, running either Linux or Windows Server.

You can store and view the metrics that you collect with the CloudWatch agent in CloudWatch just as you can with any other CloudWatch metrics. The default namespace for metrics collected by the CloudWatch agent is CWAgent, although you can specify a different namespace when you configure the agent.

The logs collected by the unified CloudWatch agent are processed and stored in Amazon CloudWatch Logs, just like logs collected by the older CloudWatch Logs agent.

Amazon Athena is an interactive query service that makes it easy to analyze data directly in Amazon Simple Storage Service (Amazon S3) using standard SQL. With a few actions in the AWS Management Console, you can point Athena at your data stored in Amazon S3 and begin using standard SQL to run ad-hoc queries and get results in seconds.

Athena is serverless, so there is no infrastructure to set up or manage, and you pay only for the queries you run. Athena scales automatically — executing queries in parallel — so results are fast, even with large datasets and complex queries.

Hence, the correct answer is: Install the Unified CloudWatch Logs agent to all on-premises and AWS resources to collect the system and application logs. Store the collected data to an Amazon S3 bucket in a central account. Set up an Amazon S3 trigger that invokes a Lambda function to analyze the logs as well as to detect any irregularities. Analyze the log data using Amazon Athena.

The option that says: Install the Unified CloudWatch Logs agent to all AWS resources to collect the system and application logs. Develop a shell script that utilizes AWS CLI to upload the on-premises logs to an S3 bucket in a central account. Analyze the log data using a custom-made Amazon EMR cluster is incorrect because you can simply install the CloudWatch Logs Agent directly on your on-premises servers. You don't need to develop a shell script to upload it on an S3 bucket. Moreover, it takes a significant amount of effort in building a custom-made Amazon EMR cluster. You can

just use Amazon Athena instead to simplify the process.
The option that says: Install the Unified CloudWatch Logs agent to all AWS resources to collect the system and application logs. Develop a shell script that utilizes AWS CLI to upload the on-premises logs to an S3 bucket for individual accounts. Analyze the log data using Amazon Macie is incorrect because Amazon Macie is just a security service that uses machine learning to automatically discover, classify, and protect sensitive data in AWS. Moreover, you can simply install the Unified CloudWatch Logs Agent to the on-premises data center instead of developing a shell script.
The option that says: Install the Unified CloudWatch Logs agent to all on-premises and AWS resources to collect the system and application logs. Consolidate all of the collected logs to your on-premises file server. Develop a custom-built solution that uses as an open-source ELK stack running Elasticsearch, Logstash, and Kibana to analyze the logs is incorrect because although the use of CloudWatch Logs is valid, developing a custom-build ELK stack solution takes a significant amount of time to implement.
References:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/Install-CloudWatch-Agent.html
https://docs.aws.amazon.com/athena/latest/ug/what-is.html
Check out these Amazon CloudWatch and Athena Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudwatch/
https://tutorialsdojo.com/aws-cheat-sheet-amazon-athena/

Q50) A development company is currently using AWS CodeBuild for automated building and testing of their application. They recently hired a DevOps engineer to review their current process as well as to provide recommendations for optimization and security. It is of utmost importance that the engineer identifies security issues and ensure that the company complies with AWS security best practices. One of their buildpsec.yaml files is shown below:
Which of the following changes should the DevOps engineer recommend? (Select TWO)

Hash the environment variables and passwords using a Base64 encoder to prevent other developers from seeing the credentials in plaintext.
✅ Using the AWS Systems Manager Parameter Store, create a DATABASE_PASSWORD secure string parameter then remove the DATABASE_PASSWORD from the environment variables.
Explanation:-AWS Systems Manager Run Command lets you remotely and securely manage the configuration of your managed instances. A managed instance is any Amazon EC2 instance or on-premises machine in your hybrid environment that has been configured for Systems Manager. Run Command enables you to automate common administrative tasks and perform ad hoc configuration changes at scale. You can use Run Command from the AWS console, the AWS Command Line Interface, AWS Tools for Windows PowerShell, or the AWS SDKs. Run Command is offered at no additional cost.
Administrators use Run Command to perform the following types of tasks on their managed instances: install or bootstrap applications, build a deployment pipeline, capture log files when an instance is terminated from an Auto Scaling group, and join instances to a Windows domain, to name a few.
AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management and secrets management. You can store data such as passwords, database strings, and license codes as parameter values. You can store values as plain text or encrypted data. You can then reference values by using the unique name that you specified when you created the parameter.
Using an IAM Role is better than storing and using your AWS access keys since this is a security risk. The same goes for your database passwords and any other credentials.
Hence, the correct answers are:
- Configure the CodeBuild project to use an IAM Role with the required permissions and remove the AWS credentials from the buildpsec.yaml file. Run scp and ssh commands using the AWS Systems Manager Run Command.
- Using the AWS Systems Manager Parameter Store, create a DATABASE_PASSWORD secure string parameter then remove the DATABASE_PASSWORD from the environment variables.
The option that says: In the post-build phase of the buildpsec.yaml file, add a configuration that will remove all temporary files which contain the environment variables and passwords from the container is incorrect because this solution still exposes the sensitive credentials in the buildspec.yaml file. You should use an IAM Role and store the database password in Systems Manager Parameter Store instead.
The option that says: Store the environment variables to the tutorialsdojo-db S3 bucket and then enable Server Side Encryption. In the pre_build phase of the buildspec.yaml file, add the configuration that will download and export the environment variables is incorrect because storing sensitive passwords in Amazon S3 is a security risk, especially if the bucket was accidentally set to public.
The option that says: Hash the environment variables and passwords using a Base64 encoder to prevent other developers to see the credentials in plaintext is incorrect because hashed credentials could be reversed to its original plaintext form. Using the Systems Manager Parameter Store feature is still the best way to store your database credentials.
References:
https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-parameter-store.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/execute-remote-commands.html
https://aws.amazon.com/systems-manager/
Check out this AWS Systems Manager Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-systems-manager/
Store the environment variables to the tutorialsdojo-db S3 bucket and then enable Server Side Encryption. In the pre_build phase of the buildspec.yaml file, add the configuration that will download and export the environment variables.
✅ Configure the CodeBuild project to use an IAM Role with the required permissions and remove the AWS credentials from the buildpsec.yaml file. Run scp and ssh commands using the AWS Systems Manager Run Command.
Explanation:-AWS Systems Manager Run Command lets you remotely and securely manage the configuration of your managed instances. A managed instance is any Amazon EC2 instance or on-premises machine in your hybrid environment that has been configured for Systems Manager. Run Command enables you to automate common administrative tasks and perform ad hoc configuration changes at scale. You can use Run Command from the AWS console, the AWS Command Line Interface, AWS Tools for Windows PowerShell, or the AWS SDKs. Run Command is offered at no additional cost.
Administrators use Run Command to perform the following types of tasks on their managed instances: install or bootstrap applications, build a deployment pipeline, capture log files when an instance is terminated from an Auto Scaling group, and join instances to a Windows domain, to name a few.
AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management and secrets management. You can store data such as passwords, database strings, and license codes as parameter values. You can store values as plain text or encrypted data. You can then reference values by using the unique name that you specified when you created the parameter.
Using an IAM Role is better than storing and using your AWS access keys since this is a security risk. The same goes for your database passwords and any other credentials.
Hence, the correct answers are:
- Configure the CodeBuild project to use an IAM Role with the required permissions and remove the AWS credentials from the buildpsec.yaml file. Run scp and ssh commands using the AWS Systems Manager Run Command.
- Using the AWS Systems Manager Parameter Store, create a DATABASE_PASSWORD secure string parameter then remove the DATABASE_PASSWORD from the environment variables.

The option that says: In the post-build phase of the buildpsec.yaml file, add a configuration that will remove all temporary files which contain the environment variables and passwords from the container is incorrect because this solution still exposes the sensitive credentials in the buildspec.yaml file. You should use an IAM Role and store the database password in Systems Manager Parameter Store instead.
The option that says: Store the environment variables to the tutorialsdojo-db S3 bucket and then enable Server Side Encryption. In the pre_build phase of the buildspec.yaml file, add the configuration that will download and export the environment variables is incorrect because storing sensitive passwords in Amazon S3 is a security risk, especially if the bucket was accidentally set to public.
The option that says: Hash the environment variables and passwords using a Base64 encoder to prevent other developers to see the credentials in plaintext is incorrect because hashed credentials could be reversed to its original plaintext form. Using the Systems Manager Parameter Store feature is still the best way to store your database credentials.
References:
https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-parameter-store.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/execute-remote-commands.html
https://aws.amazon.com/systems-manager/
Check out this AWS Systems Manager Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-systems-manager/

- In the post-build phase of the buildpsec.yaml file, add a configuration that will remove all temporary files which contain the environment variables and passwords from the container.