Answer Sheet

Q1) A DevOps Engineer has recently imported a Linux VM hosted on-premises to AWS EC2. This instance is running a legacy application that is difficult to replicate or backup. However, you are still required to create backups for this instance since it holds important data. Your solution is to take an EBS snapshot of the volumes of the instance every day.
Which of the following is the EASIEST way to automate the backup process?

○ Create a CloudWatch Events rule that is scheduled every midnight. Set the target to a Systems Manager Run command to run the EC2 CreateSnapshot API call of the EBS volumes from inside the EC2 instance.
✓ Create a CloudWatch Events rule that is scheduled to run at midnight. Set the target to directly call the EC2 CreateSnapshot API to create a snapshot of the needed EBS volumes.
Explanation:-You can call the EC2 CreateSnapshot API directly as a target from CloudWatch Events. You can also run CloudWatch Events rules according to a schedule and create an automated snapshot of an existing Amazon Elastic Block Store (Amazon EBS) volume on a schedule. You can choose a fixed rate to create a snapshot at fixed intervals or use a cron expression to specify that the snapshot is made at a specific time of day. Creating rules with built-in targets is supported only in the AWS Management Console. From the CloudWatch Events console, create a new rule scheduled to run at midnight every day. Then on the Targets section, choose EC2 CreateSnapshot API call and input the EBS volume ID that you need to snapshot. You can specify multiple targets if you need to snapshot multiple EBS volumes.
Alternatively, you can also use the Amazon Data Lifecycle Manager (DLM) for EBS Snapshots. This service provides a simple, automated way to back up data stored on Amazon EBS volumes. You can define backup and retention schedules for EBS snapshots by creating lifecycle policies based on tags. With this feature, you no longer have to rely on custom scripts to create and manage your backups.
Hence, the correct answer is: Create a CloudWatch Events rule that is scheduled to run at midnight. Set the target to directly call the EC2 CreateSnapshot API to create a snapshot of the needed EBS volumes.
The option that says: Create a CloudWatch Events rule that is scheduled every midnight. Set the target to a Lambda function that will run the EC2 CreateSnapshot API to your defined EBS volumes is incorrect because although this solution is valid, you actually don't need to launch a custom Lambda function for this scenario. Remember that the scenario asks for the solution that is easiest to implement.
The option that says: Create a CloudWatch Events rule that is scheduled every midnight. Set the target to a Systems Manager Run command to run the EC2 CreateSnapshot API call of the EBS volumes from inside the EC2 instance is incorrect because although this solution meets the requirement, it entails an extra step of using Systems Manager Run command. This is not required at all since you can directly call the EC2 CreateSnapshot API from CloudWatch Events.
The option that says: Create a script from inside the instance that will run the aws ec2 create-snapshot command. Schedule it to run every midnight using crontab is incorrect because it entails a lot of manual scripting and configuration. A better solution is to use CloudWatch Events or use the Amazon Data Lifecycle Manager (DLM) for EBS Snapshots instead.
References:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/TakeScheduledSnapshot.html
https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/ScheduledEvents.html
https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/Create-CloudWatch-Events-Rule.html
Check out this Amazon CloudWatch Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudwatch/
○ Create a script from inside the instance that will run the aws ec2 create-snapshot command. Schedule it to run every midnight using crontab.
○ Create a CloudWatch Events rule that is scheduled every midnight. Set the target to a Lambda function that will run the EC2 CreateSnapshot API to your defined EBS volumes.

Q2) A financial company has a total of over a hundred Amazon EC2 instances running across their development, testing, and production environments in AWS. Based on a recent IT review, the company initiated a new compliance rule that mandates a monthly audit of every Linux and Windows EC2 instances check for system performance issues. Each instance must have a logging function that collects various system details and retrieve custom metrics from installed applications or services. The DevOps team will periodically review these logs and analyze their contents using AWS Analytics tools, and the result will be stored in an S3 bucket.
Which is the MOST recommended way to collect and analyze logs from the instances with MINIMAL effort?

○ Configure and install AWS SDK in each Amazon EC2 instance. Create a custom daemon script that would collect and push data to CloudWatch Logs periodically. Enable CloudWatch detailed monitoring and analyze the log data of all instances using CloudWatch Logs Insights.
✓ Configure and install the unified CloudWatch Logs agent in each Amazon EC2 instance that will automatically collect and push data to CloudWatch Logs. Analyze the log data with CloudWatch Logs Insights.
Explanation:-To collect logs from your Amazon EC2 instances and on-premises servers into CloudWatch Logs, AWS offers both a new unified CloudWatch agent, and an older CloudWatch Logs agent. It is recommended to use the unified CloudWatch agent which has the following advantages:
- You can collect both logs and advanced metrics with the installation and configuration of just one agent.
- The unified agent enables the collection of logs from servers running Windows Server.
- If you are using the agent to collect CloudWatch metrics, the unified agent also enables the collection of additional system metrics, for in-guest visibility.
- The unified agent provides better performance.
CloudWatch Logs Insights enables you to interactively search and analyze your log data in Amazon CloudWatch Logs. You can perform queries to help you quickly and effectively respond to operational issues. If an issue occurs, you can use CloudWatch Logs Insights to identify potential causes and validate deployed fixes.
CloudWatch Logs Insights includes a purpose-built query language with a few simple but powerful commands. CloudWatch Logs Insights provides sample queries, command descriptions, query autocompletion, and log field discovery to help you get started quickly. Sample queries are included for several types of AWS service logs.
Hence, the correct answer is: Configure and install the unified CloudWatch Logs agent in each Amazon EC2 instance that will automatically collect and push data to CloudWatch Logs. Analyze the log data with CloudWatch Logs Insights.
The option that says: Configure and install AWS SDK in each Amazon EC2 instance. Create a custom daemon script that would collect and push data to CloudWatch Logs periodically. Enable CloudWatch detailed monitoring and analyze the log data of all instances using CloudWatch Logs Insights is incorrect because although this is a valid solution, this entails a lot of effort to implement as you have to allocate time to install the AWS SDK to each instance and develop a custom monitoring solution. Remember that the question is specifically looking for a solution that can be implemented with minimal effort. In addition, it is unnecessary and not cost-efficient to enable detailed monitoring in CloudWatch in order to meet the requirements of this scenario since this can be done using CloudWatch Logs.
The option that says: Configure and install the AWS Systems Manager Agent (SSM Agent) in each EC2 instance that will automatically collect and push data to CloudWatch Logs. Analyze the log data with CloudWatch Logs Insights is incorrect because although this is also a valid solution, it is more efficient to use CloudWatch agent than an SSM agent. Manually connecting to an instance to view log files and troubleshoot an issue with

SSM Agent is time-consuming hence, for more efficient instance monitoring, you can use the CloudWatch Agent instead to send the log data to Amazon CloudWatch Logs.
The option that says: Configure and install AWS Inspector Agent in each Amazon EC2 instance that will collect and push data to CloudWatch Logs periodically. Set up a CloudWatch dashboard to properly analyze the log data of all EC2 instances is incorrect because AWS Inspector is simply a security assessment service that only helps you in checking for unintended network accessibility of your EC2 instances and for vulnerabilities on those EC2 instances. Furthermore, setting up an Amazon CloudWatch dashboard is not suitable since it's primarily used for scenarios where you have to monitor your resources in a single view, even those resources that are spread across different AWS Regions. It is better to use CloudWatch Logs Insights instead since it enables you to interactively search and analyze your log data.
References:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/WhatIsCloudWatchLogs.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/monitoring-ssm-agent.html
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/AnalyzingLogData.html
Check out this Amazon CloudWatch Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudwatch/
CloudWatch Agent vs SSM Agent vs Custom Daemon Scripts:
https://tutorialsdojo.com/aws-cheat-sheet-cloudwatch-agent-vs-ssm-agent-vs-custom-daemon-scripts/
- Configure and install the AWS Systems Manager Agent (SSM Agent) in each EC2 instance that will automatically collect and push data to CloudWatch Logs. Analyze the log data with CloudWatch Logs Insights.
- Configure and install AWS Inspector Agent in each Amazon EC2 instance that will collect and push data to CloudWatch Logs periodically. Set up a CloudWatch dashboard to properly analyze the log data of all EC2 instances.

Q3) You are hosting several repositories of your application code on AWS CodeCommit. You are the only developer at the moment and have full access to your AWS Account. However, a new team will be joining your group to help accelerate the development of a project hosted on one of your repositories. You want each member to have read/write access on that repository so that they can freely commit and push code using their IAM CodeCommit Git credentials. For security purposes, the members should not be allowed to delete CodeCommit repositories.
Which of the following steps will give them the proper access needed?

- Create a new IAM group and add the new team members in this group. Attach the AWSCodeCommitFullAccess policy to this group to allow access to all of the functionality of CodeCommit and repository-related resources.
- Create a new IAM group and add the new team members in this group. Attach the AWSCodeCommitReadOnly policy to this group to allow access to all of the functionality of CodeCommit and repository-related resources.
- Create a new IAM group and add the new team members in this group. Attach the AmazonCodeGuruReviewerPowerUser policy to this group to allow access to all of the functionality of CodeCommit and repository-related resources. This will also enable the Amazon CodeGuru Reviewer to automatically analyze the pull requests that developers make.
- ✅ Create a new IAM group and add the new team members on this group. Attach the AWSCodeCommitPowerUser policy to this group to allow user access to all of the functionality of CodeCommit and repository-related resources.

Explanation:-You can create a policy that denies users permissions to actions you specify on one or more branches. Alternatively, you can create a policy that allows actions on one or more branches that they might not otherwise have in other branches of a repository. You can use these policies with the appropriate managed (predefined) policies.
For example, you can create a Deny policy that denies users the ability to make changes to a branch named master, including deleting that branch, in a repository named TutorialsDojoManila. You can use this policy with the AWSCodeCommitPowerUser managed policy. Users with these two policies applied would be able to create and delete branches, create pull requests, and all other actions as allowed by AWSCodeCommitPowerUser, but they would not be able to push changes to the branch named master, add or edit a file in the master branch in the CodeCommit console, or merge branches or a pull request into the master branch. Because Deny is applied to GitPush, you must include a Null statement in the policy, to allow initial GitPush calls to be analyzed for validity when users make pushes from their local repos.
There are various AWS-managed policies that you can use for providing CodeCommit access. They are:
AWSCodeCommitFullAccess – Grants full access to CodeCommit. You should apply this policy only to administrative-level users to whom you want to grant full control over CodeCommit repositories and related resources in your AWS account, including the ability to delete repositories.
AWSCodeCommitPowerUser – Allows users access to all of the functionality of CodeCommit and repository-related resources, except it does not allow them to delete CodeCommit repositories or create or delete repository-related resources in other AWS services, such as Amazon CloudWatch Events. It is recommended to apply this policy to most users.
AWSCodeCommitReadOnly – Grants read-only access to CodeCommit and repository-related resources in other AWS services, as well as the ability to create and manage their own CodeCommit-related resources (such as Git credentials and SSH keys for their IAM user to use when accessing repositories). You should apply this policy to users to whom you want to grant the ability to read the contents of a repository, but not make any changes to its contents.
Remember that you can't modify these AWS-managed policies. In order to customize the permissions, you can add a Deny rule to the IAM Role in order to block certain capabilities included in these policies.
Hence, the correct answer is: Create a new IAM group and add the new team members in this group. Attach the AWSCodeCommitPowerUser policy to this group to allow access to all of the functionality of CodeCommit and repository-related resources.
The option that says: Create a new IAM group and add the new team members in this group. Attach the AWSCodeCommitReadOnly policy to this group to allow access to all of the functionality of CodeCommit and repository-related resources is incorrect because you have to use the AWSCodeCommitPowerUser managed policy instead. The AWSCodeCommitReadOnly policy is not enough since this is primarily used to provide read access only.
The option that says: Create a new IAM group and add the new team members in this group. Attach the AWSCodeCommitFullAccess policy to this group to allow access to all of the functionality of CodeCommit and repository-related resources is incorrect because the members will be able to delete CodeCommit repositories using this policy.
The option that says: Create a new IAM group and add the new team members in this group. Attach the AmazonCodeGuruReviewerPowerUser managed policy to this group to allow access to all of the functionality of CodeCommit and repository-related resources. This will also enable the Amazon CodeGuru Reviewer to automatically analyze the pull requests that developers make is incorrect because there is no such thing as AmazonCodeGuruReviewerPowerUser managed policy. You have to use the AWSCodeCommitPowerUser policy instead.
References:
https://docs.aws.amazon.com/codecommit/latest/userguide/auth-and-access-control-permissions-reference.html
https://docs.aws.amazon.com/codecommit/latest/userguide/auth-and-access-control-iam-identity-based-access-control.html
https://docs.aws.amazon.com/codecommit/latest/userguide/pull-requests.html

---

Q4) You are developing a mobile quiz app hosted on AWS and you're using AWS CodeDeploy to deploy the application on your cluster of EC2 instances. There is a hotfix that need to be applied after the scheduled deployment. These are files not included on the current application revision so you uploaded them manually to the target instances. On the next application release, the hotfix files were included on the new revision but your deployment fails so CodeDeploy auto rollbacks to the previous version.

However, you have noticed that the hotfix files that you've manually added were missing, and the application is not working properly.
Which of the following is the possible cause and how will you prevent it from happening in the future?

● By default, CodeDeploy retains all files on the deployment location but the auto rollback will deploy the old revision files cleanly. You should choose "Overwrite the content" option for future deployments so that only the files included in the old app revision will be overwritten and the existing contents will be retained.
● By default, CodeDeploy removes all files on the deployment location and the auto rollback will deploy the old revision files cleanly. You should choose "Overwrite the content" option for future deployments so that only the files included in the old app revision will be overwritten and the existing contents will be retained.
● By default, CodeDeploy retains all files on the deployment location but the auto rollback will deploy the old revision files cleanly. You should choose "Fail the deployment" option for future deployments so that only the files included in the old app revision will be deployed and the existing contents will be retained.
✅ By default, CodeDeploy removes all files on the deployment location and the auto rollback will deploy the old revision files cleanly. You should choose "Retain the content" option for future deployments so that only the files included in the old app revision will be deployed and the existing contents will be retained.
Explanation:-As part of the deployment process, the CodeDeploy agent removes from each instance all the files installed by the most recent deployment. If files that weren't part of a previous deployment appear in target deployment locations, you can choose what CodeDeploy does with them during the next deployment:
Fail the deployment — An error is reported and the deployment status is changed to Failed.
Overwrite the content — The version of the file from the application revision replaces the version already on the instance.
Retain the content — The file in the target location is kept and the version in the application revision is not copied to the instance.
Hence, the correct answer is: By default, CodeDeploy removes all files on the deployment location and the auto rollback will deploy the old revision files cleanly. You should choose "Retain the content" option for future deployments so that only the files included in the old app revision will be deployed and the existing contents will be retained.
The option that says: By default, CodeDeploy removes all files on the deployment location and the auto rollback will deploy the old revision files cleanly. You should choose "Overwrite the content" option for future deployments so that only the files included in the old app revision will be overwritten and the existing contents will be retained is incorrect. Since you want to retain the already existing hotfix files, you should use the "Retain the content" option instead.
The option that says: By default, CodeDeploy retains all files on the deployment location but the auto rollback will deploy the old revision files cleanly. You should choose "Overwrite the content" option for future deployments so that only the files included in the old app revision will be overwritten and the existing contents will be retained is incorrect. The "Overwrite content" option will actually removes the files and not retain them. As part of the deployment process, the CodeDeploy agent removes from each instance all the files installed by the most recent deployment and not retain them. Moreover, the "Retain the content" option is a more suitable option to choose in this scenario for future deployments.
The option that says: By default, CodeDeploy retains all files on the deployment location but the auto rollback will deploy the old revision files cleanly. You should choose "Fail the deployment" option for future deployments so that only the files included in the old app revision will be deployed and the existing contents will be retained is incorrect because as part of the deployment process, the CodeDeploy agent removes from each instance all the files installed by the most recent deployment and not retain them. You should also choose the "Retain the content" option for future deployments.
References:
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments-rollback-and-redeploy.html#deployments-rollback-and-redeploy-content-options
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments.html
Check out this AWS CodeDeploy Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-codedeploy/

Q5) A leading telecommunications company is migrating a multi-tier enterprise application to AWS which must be hosted on a single Amazon EC2 Dedicated Instance. The app cannot use Auto Scaling due to server licensing constraints. For its database tier, Amazon Aurora will be used to store the data and transactions of the application. Auto-healing must be configured to ensure high availability even in the event of Amazon EC2 or Aurora database outages.
Which of the following options provides the MOST cost-effective solution for this migration task?

● Set up the AWS Personal Health Dashboard (AWS Health) events to monitor your Amazon EC2 Dedicated Instance. Set up an Amazon EC2 instance and enable the built-in instance recovery feature. Create an Aurora database with a Read Replica on the other Availability Zone. Promote the replica as the primary in the event that the primary database instance fails.
● Set up the AWS Personal Health Dashboard (AWS Health) events to monitor your Amazon EC2 Dedicated Instance. Launch an Elastic IP address and attach it to the dedicated instance. Set up a second EC2 instance in the other Availability Zone. Set up an Amazon CloudWatch Events rule to trigger an AWS Lambda function to move the EIP to the second instance when the first instance fails. Set up a single-instance Amazon Aurora database.
● Set up the AWS Personal Health Dashboard (AWS Health) events to monitor your Amazon EC2 Dedicated Instance. Set up an Auto Scaling group with a minimum and maximum instance count of 1. Launch a single-instance Amazon Aurora database.
✅ Set up the AWS Personal Health Dashboard (AWS Health) events to monitor your Amazon EC2 Dedicated Instance. Set up a CloudWatch Events rule to trigger an AWS Lambda function to start a new EC2 instance in an available Availability Zone when the instance status reaches a failure state. Configure an Aurora database with a Read Replica in the other Availability Zone. In the event that the primary database instance fails, promote the read replica to a primary database instance.
Explanation:-You can create an Amazon CloudWatch alarm that monitors an Amazon EC2 instance and automatically recovers the instance if it becomes impaired due to an underlying hardware failure or a problem that requires AWS involvement to repair. Terminated instances cannot be recovered. A recovered instance is identical to the original instance, including the instance ID, private IP addresses, Elastic IP addresses, and all instance metadata. If the impaired instance is in a placement group, the recovered instance runs in the placement group.
When the StatusCheckFailed_System alarm is triggered, and the recover action is initiated, you will be notified by the Amazon SNS topic that you selected when you created the alarm and associated the recover action. During instance recovery, the instance is migrated during an instance reboot, and any data that is in-memory is lost. When the process is complete, information is published to the SNS topic you've configured for the alarm. Anyone who is subscribed to this SNS topic will receive an email notification that includes the status of the recovery attempt and any further instructions. You will notice an instance reboot on the recovered instance.
Examples of problems that cause system status checks to fail include:
- Loss of network connectivity
- Loss of system power
- Software issues on the physical host
- Hardware issues on the physical host that impact network reachability
If your instance has a public IPv4 address, it retains the public IPv4 address after recovery.

You can configure a CloudWatch alarm to automatically recover impaired EC2 instances and notify you through Amazon SNS. However, the SNS notification by itself doesn't include the results of the automatic recovery action.
You must also configure an Amazon CloudWatch Events rule to monitor AWS Personal Health Dashboard (AWS Health) events for your instance. Then, you are notified of the results of automatic recovery actions for an instance.
Hence, the correct answer is: Set up the AWS Personal Health Dashboard (AWS Health) events to monitor your Amazon EC2 Dedicated Instance. Set up a CloudWatch Events rule to trigger an AWS Lambda function to start a new EC2 instance in an available Availability Zone when the instance status reaches a failure state. Configure an Aurora database with a Read Replica in the other Availability Zone. In the event that the primary database instance fails, promote the read replica to a primary database instance.
The option that says: Set up the AWS Personal Health Dashboard (AWS Health) events to monitor your Amazon EC2 Dedicated Instance. Set up an Auto Scaling group with a minimum and maximum instance count of 1. Launch a single-instance Amazon Aurora database is incorrect because launching a single-instance Aurora database is not a highly available architecture. You have to set up at least a Read Replica that you can configure to be the new primary instance in the event of outages.
The option that says: Set up the AWS Personal Health Dashboard (AWS Health) events to monitor your Amazon EC2 Dedicated Instance. Set up an Amazon EC2 instance and enable the built-in instance recovery feature. Create an Aurora database with a Read Replica on the other Availability Zone. Promote the replica as the primary in the event that the primary database instance fails is incorrect because there is no built-in instance recovery failure feature for Amazon EC2. You have to use a combination of CloudWatch and a Lambda function to automatically recover the EC2 instance from failure.
The option that says: Set up the AWS Personal Health Dashboard (AWS Health) events to monitor your Amazon EC2 Dedicated Instance. Launch an Elastic IP address and attach it to the dedicated instance. Set up a second EC2 instance in the other Availability Zone. Set up an Amazon CloudWatch Events rule to trigger an AWS Lambda function to move the EIP to the second instance when the first instance fails. Set up a single-instance Amazon Aurora database is incorrect because setting up a second EC2 instance in other Availability Zone entails an additional cost. Launching a single-instance Aurora database is not a highly available architecture.
References:
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-recover.html
https://aws.amazon.com/premiumsupport/knowledge-center/cloudwatch-sns-ec2-automatic-recovery/
https://aws.amazon.com/premiumsupport/knowledge-center/automatic-recovery-ec2-cloudwatch/

Q6) A startup is developing an AI-powered traffic monitoring portal that will be hosted in AWS Cloud. The design of the cloud architecture should be highly available and fault-tolerant to avoid unnecessary outages that can affect the users. A DevOps Engineer was instructed to implement the architecture and also set up a system that automatically assesses applications for exposure, vulnerabilities, and deviations from the AWS best practices.
Among the options below, which is the MOST appropriate architecture that you should implement?

- Use Amazon GuardDuty for automated security assessment to help improve the security and compliance of your applications. Set up an Amazon ElastiCache cluster for the database caching of the portal. Launch an Auto Scaling group of Amazon EC2 instances on four Availability Zones then associate it to an Application Load Balancer. Set up a MySQL RDS database instance with Multi-AZ deployments configuration and Read Replicas. Using Amazon Route 53, create a CNAME record for the root domain to point to the load balancer.
- Use Amazon Macie for automated security assessment to help improve the security and compliance of your applications. Set up Amazon DynamoDB as the database of the portal. Launch an Auto Scaling group of EC2 instances on four Availability Zones with an Application Load Balancer in front to distribute the incoming traffic. Using Amazon Route 53, create a non-alias A record for the root domain to point to the load balancer.
- Use AWS Shield for automated security assessment to help improve the security and compliance of your applications. Launch an Auto Scaling group of Amazon EC2 instances on two Availability Zones with an Application Load Balancer in front. Set up a MySQL RDS database instance with Multi-AZ deployments configuration. Using Amazon Route 53, create a non-alias A record for the root domain to point to the load balancer.
- ✅ Use Amazon Inspector for automated security assessment to help improve the security and compliance of your applications. Launch an Auto Scaling group of Amazon EC2 instances on three Availability Zones. Set up an Application Load Balancer to distribute the incoming traffic. Set up an Amazon Aurora Multi-Master as the database tier. Using Amazon Route 53, create an alias record for the root domain to point to the load balancer.

Explanation:-Amazon Inspector is an automated security assessment service that helps improve the security and compliance of applications deployed on AWS. Amazon Inspector automatically assesses applications for exposure, vulnerabilities, and deviations from best practices. After performing an assessment, Amazon Inspector produces a detailed list of security findings prioritized by level of severity. These findings can be reviewed directly or as part of detailed assessment reports which are available via the Amazon Inspector console or API.
Amazon Inspector security assessments help you check for unintended network accessibility of your Amazon EC2 instances and for vulnerabilities on those EC2 instances. Amazon Inspector assessments are offered to you as pre-defined rules packages mapped to common security best practices and vulnerability definitions. Examples of built-in rules include checking for access to your EC2 instances from the internet, remote root login being enabled, or vulnerable software versions installed. These rules are regularly updated by AWS security researchers.
If you host a website on multiple Amazon EC2 instances, you can distribute traffic to your website across the instances by using an Elastic Load Balancing (ELB) load balancer. The ELB service automatically scales the load balancer as traffic to your website changes over time. The load balancer also can monitor the health of its registered instances and route domain traffic only to healthy instances. To route domain traffic to an ELB load balancer, use Amazon Route 53 to create an alias record that points to your load balancer. An alias record is a Route 53 extension to DNS. It's similar to a CNAME record, but you can create an alias record both for the root domain, such as tutorialsdojo.com, and for subdomains, such as www.tutorialsdojo.com. (You can create CNAME records only for subdomains.)
You can use Amazon EC2 Auto Scaling to maintain a minimum number of running instances for your application at all times. Amazon EC2 Auto Scaling can detect when your instance or application is unhealthy and replace it automatically to maintain the availability of your application. You can also use Amazon EC2 Auto Scaling to scale your Amazon EC2 capacity up or down automatically based on demand, using criteria that you specify.
In this scenario, all of the options are highly available architectures. The main difference here is how they use Amazon Route 53. Keep in mind that you have to create an alias record in Amazon Route 53 in order to point to your load balancer.
Hence, the correct answer is: Use Amazon Inspector for automated security assessment to help improve the security and compliance of your applications. Launch an Auto Scaling group of Amazon EC2 instances on three Availability Zones. Set up an Application Load Balancer to distribute the incoming traffic. Set up an Amazon Aurora Multi-Master as the database tier. Using Amazon Route 53, create an alias record for the root domain to point to the load balancer.
The option that says: Use Amazon Shield for automated security assessment to help improve the security and compliance of your applications. Launch an Auto Scaling group of Amazon EC2 instances on two Availability Zones with an Application Load Balancer in front. Set up a MySQL RDS database instance with Multi-AZ deployments configuration. Using Amazon Route 53, create a non-alias A record for the root domain to point to the load balancer is incorrect because AWS Shield is a managed Distributed Denial of Service (DDoS) protection service and not an automated security assessment. Moreover, you need to create an Alias record with the root DNS name and not an A record.
The option that says: Use Amazon GuardDuty for automated security assessment to help improve the security and compliance of your applications. Set up an Amazon ElastiCache cluster for the database caching of the portal. Launch an Auto Scaling group of Amazon EC2 instances on four Availability Zones then associate it to an Application Load Balancer. Set up a MySQL RDS database instance with Multi-AZ deployments configuration and Read Replicas. Using Amazon Route 53, create a CNAME record for the root domain to point to the load balancer is incorrect

because Amazon GuardDuty is a threat detection service that continuously monitors for malicious activity and unauthorized behavior to protect your AWS accounts. The correct service that you should use is Amazon Inspector since this is an automated security assessment service that helps improve the security and compliance of applications deployed on AWS. In addition, you can create CNAME records only for subdomains and not for the zone apex or root domain.
The option that says: Use Amazon Macie for automated security assessment to help improve the security and compliance of your applications. Set up Amazon DynamoDB as the database of the portal. Launch an Auto Scaling group of EC2 instances on four Availability Zones with an Application Load Balancer in front to distribute the incoming traffic. Using Amazon Route 53, create a non-alias A record for the root domain to point to the load balancer is incorrect because you should use Amazon Inspector instead of Amazon Macie, since this is just a security service that uses machine learning to automatically discover, classify, and protect sensitive data in AWS. In addition, you should create an alias record with the root DNS name and not a non-alias A record.
References:
http://docs.aws.amazon.com/AmazonS3/latest/dev/WebsiteHosting.html
http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-to-elb-load-balancer.html
https://docs.aws.amazon.com/autoscaling/ec2/userguide/attach-load-balancer-asg.html
Check out this Amazon Route 53 Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-route-53/

Q7) A telecommunications company has a web portal that requires a cross-region failover. The portal stores its data in an Amazon Aurora database in the primary region (us-west-1) and the Parallel Query feature is also enabled in the database to optimize some of the I/O and computation involved in processing data-intensive queries. The portal uses Route 53 to direct customer traffic to the active region.
Which among the options below should be taken to MINIMIZE downtime of the portal in the event that the primary database fails?

○ Configure the Route 53 record to balance traffic between both regions equally using the Failover routing policy. Enable the Aurora multi-master option and set up a Route 53 health check to analyze the health of the databases. Set the Route 53 record to automatically direct all traffic to the secondary region when a primary database fails.
○ Set up CloudWatch to monitor the status of the Amazon Aurora database. Create a CloudWatch Events rule to send a Slack message to the SysOps Team using Amazon SNS in the event of a database outage. Instruct the SysOps team to redirect traffic to an S3 static website that displays a downtime message. Manually promote the read replica as the primary instance and verify the portal's status. Redirect traffic from the S3 website to the secondary region.
✓ Launch a read replica of the primary database to the second region. Set up Amazon RDS Event Notification to publish status updates to an SNS topic. Create a Lambda function subscribed to the topic to monitor database health. Configure the Lambda function to promote the read replica as the primary in the event of a failure. Update the Route 53 record to redirect traffic from the primary region to the secondary region.
Explanation:-Amazon RDS uses the Amazon Simple Notification Service (Amazon SNS) to provide notification when an Amazon RDS event occurs. These notifications can be in any notification form supported by Amazon SNS for an AWS Region, such as an email, a text message, or a call to an HTTP endpoint.
Amazon RDS groups these events into categories that you can subscribe to so that you can be notified when an event in that category occurs. You can subscribe to an event category for a DB instance, DB cluster, DB cluster snapshot, DB parameter group, or DB security group. For example, if you subscribe to the Backup category for a given DB instance, you are notified whenever a backup-related event occurs that affects the DB instance. If you subscribe to a configuration change category for a DB security group, you are notified when the DB security group is changed. You also receive notification when an event notification subscription changes.
For Amazon Aurora, events occur at both the DB cluster and the DB instance level, so you can receive events if you subscribe to an Aurora DB cluster or an Aurora DB instance.
Event notifications are sent to the addresses that you provide when you create the subscription. You might want to create several different subscriptions, such as one subscription receiving all event notifications and another subscription that includes only critical events for your production DB instances. You can easily turn off notification without deleting a subscription by choosing No for Enabled in the Amazon RDS console or by setting the Enabled parameter to false using the AWS CLI or Amazon RDS API.
When you copy a snapshot to an AWS Region that is different from the source snapshot's AWS Region, the first copy is a full snapshot copy, even if you copy an incremental snapshot. A full snapshot copy contains all of the data and metadata required to restore the DB instance. After the first snapshot copy, you can copy incremental snapshots of the same DB instance to the same destination region within the same AWS account.
Depending on the AWS Regions involved and the amount of data to be copied, a cross-region snapshot copy can take hours to complete. In some cases, there might be a large number of cross-region snapshot copy requests from a given source AWS Region. In these cases, Amazon RDS might put new cross-region copy requests from that source AWS Region into a queue until some in-progress copies complete. No progress information is displayed about copy requests while they are in the queue. Progress information is displayed when the copy starts.
This means that a cross-region snapshot doesn't provide a high RPO compared with a Read Replica since the snapshot takes significant time to complete. Although this is better than Multi-AZ deployments since you can replicate your database across AWS Regions, using a Read Replica is still the best choice for providing a high RTO and RPO for disaster recovery.
Hence, the correct answer is: Launch a read replica of the primary database to the second region. Set up Amazon RDS Event Notification to publish status updates to an SNS topic. Create a Lambda function subscribed to the topic to monitor database health. Configure the Lambda function to promote the read replica as the primary in the event of a failure. Update the Route 53 record to redirect traffic from the primary region to the secondary region.
The option that says: Set up CloudWatch to monitor the status of the Amazon Aurora database. Create a CloudWatch Events rule to send a Slack message to the SysOps Team using Amazon SNS in the event of a database outage. Instruct the SysOps team to redirect traffic to an S3 static website that displays a downtime message. Manually promote the read replica as the primary instance and verify the portal's status. Redirect traffic from the S3 website to the secondary region is incorrect because this solution entails a lot of manual steps. It is possible that the SysOps Team might not respond to the Slack message immediately.
The option that says: Create a CloudWatch Events rule to periodically invoke a Lambda function that checks the health of the primary Amazon Aurora database every hour. Configure the Lambda function to promote the read replica as the primary if a failure was detected. Update the Route 53 record to redirect traffic from the primary to the secondary region is incorrect because although this is a valid option, there will still be a delay on the database failover since the CloudWatch rule only runs every hour. A better solution is to use Amazon RDS Event Notification instead.
The option that says: Configure the Route 53 record to balance traffic between both regions equally using the Weighted routing policy. Enable the Aurora multi-master option and set up a Route 53 health check to analyze the health of the databases. Set the Route 53 record to automatically direct all traffic to the secondary region when a primary database fails is incorrect because although an Aurora multi-master improves the availability of the database, the application will still experience downtime in the event of an AWS Region outage. Much like Amazon RDS Multi-AZ, the Aurora multi-master has its data replicated across multiple Availability Zones only but not to another AWS Region. You should use Amazon Aurora Global Database instead.
References:
https://aws.amazon.com/blogs/database/implementing-a-disaster-recovery-strategy-with-amazon-rds/

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CopySnapshot.html#USER_CopySnapshot.AcrossRegions
https://aws.amazon.com/rds/details/read-replicas/
Check out this Amazon RDS Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-relational-database-service-amazon-rds/

● Create a CloudWatch Events rule to periodically invoke a Lambda function that checks the health of the primary Amazon Aurora database every hour. Configure the Lambda function to promote the read replica as the primary if a failure was detected. Update the Route 53 record to redirect traffic from the primary to the secondary region.

Q8) An international tours and travel company is planning to launch a multi-tier Node.js web portal with a MySQL database to AWS. The portal must be highly available during the deployment of new portal versions in the future and have the ability to roll back the changes if necessary, to improve user experience. There are third-party applications that will also use the same MySQL database that the portal is using. The architecture should allow the IT Operations team to centrally view all the server logs from various EC2 instances and store the data for three months. It should also have a feature that allows the team to search and filter server logs in near-real-time for monitoring purposes. The solution should be cost-effective and preferably has less operational overhead.
As a DevOps Engineer, which of the following is the BEST solution that you should implement to satisfy the above requirements?

● Host the multi-tier Node.js web portal in an Auto Scaling group of EC2 instances behind an Application Load Balancer. Set up an Amazon RDS MySQL database instance. Configure the CloudWatch Log agent to send the application logs to Amazon CloudWatch Logs with 90-day data retention.

● Host the multi-tier Node.js web portal in an Auto Scaling group of EC2 instances behind an Application Load Balancer. Set up an Amazon RDS MySQL database instance. For log monitoring and analysis, configure CloudWatch Logs to fetch the application logs from the instances and send them to an Amazon ES cluster. Purge the contents of the Amazon ES domain every 90 days and then recreate it again.

✅ Using AWS Elastic Beanstalk, host the multi-tier Node.js web portal in a load-balancing and autoscaling environment. Set up an Amazon RDS MySQL database with a Multi-AZ deployments configuration that is decoupled from the Elastic Beanstalk stack. Set the log options to stream the application logs to Amazon CloudWatch Logs with 90-day retention.

Explanation:-With Elastic Beanstalk, you can quickly deploy and manage applications in the AWS Cloud without having to learn about the infrastructure that runs those applications. Elastic Beanstalk reduces management complexity without restricting choice or control. You simply upload your application, and Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring.
Elastic Beanstalk supports applications developed in Go, Java, .NET, Node.js, PHP, Python, and Ruby. When you deploy your application, Elastic Beanstalk builds the selected supported platform version and provisions one or more AWS resources, such as Amazon EC2 instances, to run your application.
You can interact with Elastic Beanstalk by using the AWS Management Console, the AWS Command Line Interface (AWS CLI), or eb, a high-level CLI designed specifically for Elastic Beanstalk.
An Amazon RDS instance attached to an Elastic Beanstalk environment is ideal for development and testing environments. However, it's not ideal for production environments because the lifecycle of the database instance is tied to the lifecycle of your application environment. If you terminate the environment, then you will lose your data because the Amazon RDS instance is deleted by the environment.
Hence, the correct answer is: Using AWS Elastic Beanstalk, host the multi-tier Node.js web portal in a load-balancing and autoscaling environment. Set up an Amazon RDS MySQL database with a Multi-AZ deployments configuration that is decoupled from the Elastic Beanstalk stack. Set the log options to stream the application logs to Amazon CloudWatch Logs with a 90-day retention.
The option that says: Using AWS Elastic Beanstalk, host the multi-tier Node.js web portal in a load-balancing and autoscaling environment. Set up an Amazon RDS MySQL instance with a Multi-AZ deployments configuration within the Elastic Beanstalk stack. Set the log options to stream the application logs to Amazon CloudWatch Logs with 90-day data retention is incorrect because it's not ideal to place the database within the Elastic Beanstalk environment because the lifecycle of the database instance is tied to the lifecycle of your application environment in production.
The option that says: Host the multi-tier Node.js web portal in an Auto Scaling group of EC2 instances behind an Application Load Balancer. Set up an Amazon RDS MySQL database instance. Configure the CloudWatch Log agent to send the application logs to Amazon CloudWatch Logs with 90-day data retention is incorrect because it is easier to upload the web portal to Elastic Beanstalk instead. Moreover, you have to enable Multi-AZ deployments in RDS to improve the availability of your database tier.
The option that says: Host the multi-tier Node.js web portal in an Auto Scaling group of EC2 instances behind an Application Load Balancer. Set up an Amazon RDS MySQL database instance. For log monitoring and analysis, configure CloudWatch Logs to fetch the application logs from the instances and send them to an Amazon ES cluster. Purge the contents of the Amazon ES domain every 90 days and then recreate it again is incorrect because although this solution may work, it entails a lot of operational overhead to maintain the Amazon ES cluster.
References:
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/AWSHowTo.RDS.html
https://aws.amazon.com/premiumsupport/knowledge-center/decouple-rds-from-beanstalk/
Check out this AWS Elastic Beanstalk Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-elastic-beanstalk/

● Using AWS Elastic Beanstalk, host the multi-tier Node.js web portal in a load-balancing and autoscaling environment. Set up an Amazon RDS MySQL instance with a Multi-AZ deployments configuration within the Elastic Beanstalk stack. Set the log options to stream the application logs to Amazon CloudWatch Logs with 90-day data retention.

Q9) A DevOps Engineer recently completed the CI/CD automation process in her organization. She used AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, and AWS CodePipeline to build, test, and deploy applications automatically to several On-Demand EC2 Instances. Her manager assigned a new task of including a step that performs a security assessment scan of the operating system on every application deployment.
How should the Engineer automate the security check?

● Monitor the Auto Scaling event notifications for launching new EC2 instances using Amazon CloudWatch Events. Configure CloudWatch Events to trigger an AWS X-Ray scan to your instances.

● Monitor the AWS CodeDeploy notifications for launching new EC2 instances using Amazon CloudWatch Events. Configure CloudWatch Events to trigger an AWS X-Ray scan to your instances.

● Monitor the Auto Scaling event notifications for launching new EC2 instances using Amazon CloudWatch Events. Configure CloudWatch Events to trigger an Amazon Inspector scan to your instances.

✅ Monitor the AWS CodeDeploy notifications for launching new EC2 instances using Amazon CloudWatch Events. Configure CloudWatch Events to trigger an Amazon Inspector scan to your instances.

Explanation:-Through an integration with Amazon CloudWatch Events, customers can now create events that automatically trigger Amazon Inspector assessments to run against your environments. Within Amazon CloudWatch Events, you can now create event rules that target your

Amazon Inspector assessment templates. When that CloudWatch Event occurs, Amazon Inspector will automatically be notified to run the specified assessment.
Amazon Inspector assessments can be triggered by any CloudWatch Event. You can set up a recurring Schedule event with either a simple fixed recurring rate or a more detailed Cron expression, or create an event pattern which monitors other AWS services for actions to trigger an assessment.
For example, you can create an event which monitors AWS Auto Scaling for new EC2 Instances being launched, or monitors AWS CodeDeploy notifications for when a code deployment has been successfully completed. Once CloudWatch Events have been configured against Amazon Inspector templates, these assessment events will be displayed in the Inspector console as part of your assessment templates so you can see all of the automated triggers for that assessment.
Hence, the correct answer is: Monitor the AWS CodeDeploy notifications for launching new EC2 instances using Amazon CloudWatch Events. Configure CloudWatch Events to trigger an Amazon Inspector scan to your instances.
The option that says: Monitor the Auto Scaling event notifications for launching new EC2 instances using Amazon CloudWatch Events. Configure CloudWatch Events to trigger an Amazon Inspector scan to your instances is incorrect because you have to monitor the event notifications from CodeDeploy instead of your Auto Scaling group.
The option that says: Monitor the Auto Scaling event notifications for launching new EC2 instances using Amazon CloudWatch Events. Configure CloudWatch Events to trigger an AWS X-Ray scan to your instances is incorrect because you have to use Amazon Inspector instead of AWS X-Ray. Moreover, it is more suitable to monitor the event notifications from CodeDeploy instead. Keep in mind that the AWS X-Ray service is primarily used by developers to analyze and debug applications.
The option that says: Monitor the AWS CodeDeploy notifications for launching new EC2 instances using Amazon CloudWatch Events. Configure CloudWatch Events to trigger an AWS X-Ray scan to your instances is incorrect because using Amazon Inspector to run assessment scans of your EC2 instances is a more appropriate solution instead of using AWS X-Ray.
References:
https://aws.amazon.com/about-aws/whats-new/2017/07/amazon-inspector-adds-event-triggers-to-automatically-run-assessments/
https://docs.aws.amazon.com/inspector/latest/userguide/inspector_assessments.html
Check out this Amazon Inspector Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-inspector/

Q10) You are working as a DevOps Engineer for a leading pharmaceutical company. After the recent annual IT audit, several Amazon EC2 instances were discovered running an outdated operating system, and there were many security vulnerabilities left unpatched. To safeguard your systems from various cybersecurity attacks, mitigating these vulnerabilities as soon as possible is crucial. You are also required to record all of the changes to patch and association compliance statuses.
Which among the following options is the MOST efficient way to solve this issue?

- Set up Kibana and Amazon QuickSight to apply, monitor, and visualize the patch statuses of all Amazon EC2 instances in the VPC.
- Manage, record, and deploy the security patches for the OS for the Amazon EC2 instances using AWS OpsWorks and Amazon ES.
- ✅ Manage, record, and deploy the OS security patches of the Amazon EC2 instances using a combination of AWS Systems Manager and AWS Config.

Explanation:-AWS Systems Manager Patch Manager automates the process of patching managed instances with security-related updates. For Linux-based instances, you can also install patches for non-security updates. 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, Ubuntu Server, Red Hat Enterprise Linux (RHEL), SUSE Linux Enterprise Server (SLES), 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. (Tags are keys that help identify and sort your resources within your organization.) You can add tags to your patch baselines themselves when you create or update them.
Since you are also required to record all of the changes to patch and association compliance statuses, you can use AWS Config to meet this requirement. AWS Config is a service that enables you to assess, audit, and evaluate the configurations of your AWS resources. Config continuously monitors and records your AWS resource configurations and allows you to automate the evaluation of recorded configurations against desired configurations.
Hence, the correct answer is: Manage, record, and deploy the OS security patches of the Amazon EC2 instances using a combination of AWS Systems Manager and AWS Config.
The option that says: Configure each of the Amazon EC2 instances to automatically install the required security patches for its operating system every week on the provided maintenance window is incorrect because Amazon EC2 does not have a built-in function to automatically install the security OS patch on a regular basis. You should use AWS Systems Manager Patch Manager instead.
The option that says: Set up Kibana and Amazon QuickSight to apply, monitor, and visualize the patch statuses of all Amazon EC2 instances in the VPC is incorrect because QuickSight and Kibana are primarily used for data visualization and not for patch management. You can use Amazon Elasticsearch (ES) with Kibana but this service is not suitable for this scenario.
The option that says: Manage, record, and deploy the security patches for the OS for the Amazon EC2 instances using AWS OpsWorks and Amazon ES is incorrect because the Amazon Elasticsearch Service (Amazon ES) is just an AWS-managed service that makes it easy to deploy, operate, and scale Elasticsearch clusters in AWS. AWS OpsWorks is simply a configuration management service that provides managed instances of Chef and Puppet. It is not suitable to be used to install the OS patches.
References:
https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-patch.html
https://aws.amazon.com/config/
Check out this AWS Systems Manager Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-systems-manager/

- Configure each of the Amazon EC2 instances to automatically install the required security patches for its operating system every week on the provided maintenance window.

Q11) You are working on a web application that allows designers to create mobile themes for their android phones. The app is hosted on a cluster of Auto Scaling ECS instances and its deployments are handled by AWS CodeDeploy. ALB health checks are not sufficient to tell that new version deployments are successful, rather you have custom validation scripts that verify all APIs of the application. You want to make sure that there are no 5XX error replies on the new version before continuing production deployment and that you are notified via email if results failed. You also want to configure automatic rollback to the older version when the validation fails.
Which combination of options should you implement to meet this requirement? (Select THREE)

- Create your validation scripts on AWS Lambda and invoke them after deployment to validate your new app version.

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 AfterAllowTestTraffic lifecycle hook 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:
- Create your validation scripts on AWS Lambda and define the functions on the AppSpec lifecycle hook to validate the app using test traffic.
- Have AWS CloudWatch Alarms trigger an AWS SNS notification when the threshold for 5xx is reached on CloudWatch.
- Associate CloudWatch Alarms to your deployment group to have it trigger a rollback when the 5xx error alarm is active.
The option that says: Create your validation scripts on AWS Lambda and invoke them after deployment to validate your new app version is incorrect since you will have to trigger the Lambda validation functions before going to production traffic. This also gives you the opportunity to rollback the deployment in case it is not working as expected.
The option that says: Have AWS Lambda trigger an AWS SNS notification after performing the validation of the new app revision is incorrect because you will need to monitor the results of each API test call from Lambda if you are going to implement this. It is better to implement results monitoring as a threshold on CloudWatch.
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.
References:
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
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-steps.html#deployment-steps-what-happens
Check out this AWS CodeDeploy Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-codedeploy/

- Have AWS CloudWatch Alarms trigger an AWS SNS notification when the threshold for 5xx is reached on CloudWatch.
- Associate CloudWatch Alarms to your deployment group to have it trigger a rollback when the 5xx error alarm is active.

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 AfterAllowTestTraffic lifecycle hook 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:
- Create your validation scripts on AWS Lambda and define the functions on the AppSpec lifecycle hook to validate the app using test traffic.
- Have AWS CloudWatch Alarms trigger an AWS SNS notification when the threshold for 5xx is reached on CloudWatch.
- Associate CloudWatch Alarms to your deployment group to have it trigger a rollback when the 5xx error alarm is active.
The option that says: Create your validation scripts on AWS Lambda and invoke them after deployment to validate your new app version is incorrect since you will have to trigger the Lambda validation functions before going to production traffic. This also gives you the opportunity to rollback the deployment in case it is not working as expected.
The option that says: Have AWS Lambda trigger an AWS SNS notification after performing the validation of the new app revision is incorrect because you will need to monitor the results of each API test call from Lambda if you are going to implement this. It is better to implement results monitoring as a threshold on CloudWatch.
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.
References:
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
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-steps.html#deployment-steps-what-happens
Check out this AWS CodeDeploy Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-codedeploy/

- Have Lambda send results to AWS CloudWatch Alarms directly and trigger a rollback when 5xx reply errors are received during deployment.
- Create your validation scripts on AWS Lambda and define the functions on the AppSpec lifecycle hook to validate the app using test traffic.

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 AfterAllowTestTraffic lifecycle hook 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:
- Create your validation scripts on AWS Lambda and define the functions on the AppSpec lifecycle hook to validate the app using test traffic.
- Have AWS CloudWatch Alarms trigger an AWS SNS notification when the threshold for 5xx is reached on CloudWatch.
- Associate CloudWatch Alarms to your deployment group to have it trigger a rollback when the 5xx error alarm is active.
The option that says: Create your validation scripts on AWS Lambda and invoke them after deployment to validate your new app version is incorrect since you will have to trigger the Lambda validation functions before going to production traffic. This also gives you the opportunity to rollback the

deployment in case it is not working as expected.
The option that says: Have AWS Lambda trigger an AWS SNS notification after performing the validation of the new app revision is incorrect because you will need to monitor the results of each API test call from Lambda if you are going to implement this. It is better to implement results monitoring as a threshold on CloudWatch.
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.
References:
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
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-steps.html#deployment-steps-what-happens
Check out this AWS CodeDeploy Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-codedeploy/
● Have AWS Lambda trigger an AWS SNS notification after performing the validation of the new app revision.

Q12) A digital payment gateway system is running in AWS which serves thousands of businesses worldwide. It is hosted in an Auto Scaling Group of EC2 instances behind an Application Load Balancer with an Amazon RDS database in a Multi-AZ deployment configuration. The company is using several CloudFormation templates in deploying the new version of the system. The AutoScalingRollingUpdate policy is used to control how CloudFormation handles rolling updates for their Auto Scaling group which replaces the old instances based on the parameters they have set. Lately, there were a lot of failed deployments which has caused system unavailability issues and business disruptions. They want to find out what's preventing their Auto Scaling group from updating correctly during a stack update.
In this scenario, how should the DevOps engineer troubleshoot this issue? (Select THREE)

✅ During a rolling update, suspend the following Auto Scaling processes: HealthCheck, ReplaceUnhealthy, AZRebalance, AlarmNotification, and ScheduledActions.
Explanation:-The AWS::AutoScaling::AutoScalingGroup resource uses the UpdatePolicy attribute to define how an Auto Scaling group resource is updated when the AWS CloudFormation stack is updated. If you don't have the right settings configured for the UpdatePolicy attribute, your rolling update can produce unexpected results.
You can use the AutoScalingRollingUpdate policy to control how AWS CloudFormation handles rolling updates for an Auto Scaling group. This common approach keeps the same Auto Scaling group, and then replaces the old instances based on the parameters that you set.
The AutoScalingRollingUpdate policy supports the following configuration options:
"UpdatePolicy": {
"AutoScalingRollingUpdate": {
"MaxBatchSize": Integer,
"MinInstancesInService": Integer,
"MinSuccessfulInstancesPercent": Integer,
"PauseTime": String,
"SuspendProcesses": [List of processes],
"WaitOnResourceSignals": Boolean
}
}
Using a rolling update has a risk of system outages and performance degradation due to the decreased availability of your running EC2 instances. If you want to ensure high availability of your application, you can also use the AutoScalingReplacingUpdate policy to perform an immediate rollback of the stack without any possibility of failure.
To find out what's preventing your Auto Scaling group from updating correctly during a stack update, work through the following troubleshooting scenarios as needed:
Configure WaitOnResourceSignals and PauseTime to avoid problems with success signals
- In your AutoScalingRollingUpdate policy, set the WaitOnResourceSignals property to false. Take note that if WaitOnResourceSignals is set to true, PauseTime changes to a timeout value. AWS CloudFormation waits to receive a success signal until the maximum time specified by the PauseTime value. If a signal is not received, AWS CloudFormation cancels the update. Then, AWS CloudFormation rolls back the stack with the same settings, including the same PauseTime value.
Configure MinSuccessfulInstancesPercent to avoid stack rollback
- If you're replacing a large number of instances during a rolling update and waiting for a success signal for each instance, complete the following: In your AutoScalingRollingUpdate policy, set the value of the MinSuccessfulInstancesPercent property. Take note that setting the MinSuccessfulInstancesPercent property prevents AWS CloudFormation from rolling back the entire stack if only a single instance fails to launch.
Configure SuspendProcesses to avoid unexpected changes to the Auto Scaling group
- During a rolling update, suspend the following Auto Scaling processes: HealthCheck, ReplaceUnhealthy, AZRebalance, AlarmNotification, and ScheduledActions. It is quite important to know that if you're using your Auto Scaling group with Elastic Load Balancing (ELB), you should not suspend the following processes: Launch, Terminate, and AddToLoadBalancer. These processes are required to make rolling updates. Take note that if an unexpected scaling action changes the state of the Auto Scaling group during a rolling update, the update can fail. The failure can result from an inconsistent view of the group by AWS CloudFormation.
Based on the above information, the correct answers are:
- In your AutoScalingRollingUpdate policy, set the WaitOnResourceSignals property to false.
- In your AutoScalingRollingUpdate policy, set the value of the MinSuccessfulInstancesPercent property to prevent AWS CloudFormation from rolling back the entire stack if only a single instance fails to launch
- During a rolling update, suspend the following Auto Scaling processes: HealthCheck, ReplaceUnhealthy, AZRebalance, AlarmNotification, and ScheduledActions
The option that says: Switch from AutoScalingRollingUpdate to AutoScalingReplacingUpdate policy by modifying the UpdatePolicy of the AWS::AutoScaling::AutoscalingGroup resource in the CloudFormation template. Set the WillReplace property to true is incorrect because although the AutoScalingReplacingUpdate policy provides an immediate rollback of the stack without any possibility of failure, this solution is not warranted since the scenario asks for the options that will help troubleshoot the issue.
The option that says: Suspend the following Auto Scaling processes that are related with your ELB: Launch, Terminate, and AddToLoadBalancer is incorrect because these processes are required by the ELB to make rolling updates.
References:
https://aws.amazon.com/premiumsupport/knowledge-center/auto-scaling-group-rolling-updates/
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-as-group.html
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-updatepolicy.html#cfn-attributes-updatepolicy-replacingupdate
Check out this AWS CloudFormation Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-cloudformation/

Switch from AutoScalingRollingUpdate to AutoScalingReplacingUpdate policy by modifying the UpdatePolicy of the AWS::AutoScaling::AutoscalingGroup resource in the CloudFormation template. Set the WillReplace property to true.

Suspend the following Auto Scaling processes that are related with your ELB: Launch, Terminate, and AddToLoadBalancer.

In your AutoScalingRollingUpdate policy, set the WaitOnResourceSignals property to false.

Explanation:-The AWS::AutoScaling::AutoScalingGroup resource uses the UpdatePolicy attribute to define how an Auto Scaling group resource is updated when the AWS CloudFormation stack is updated. If you don't have the right settings configured for the UpdatePolicy attribute, your rolling update can produce unexpected results.

You can use the AutoScalingRollingUpdate policy to control how AWS CloudFormation handles rolling updates for an Auto Scaling group. This common approach keeps the same Auto Scaling group, and then replaces the old instances based on the parameters that you set.

The AutoScalingRollingUpdate policy supports the following configuration options:

```
"UpdatePolicy": {
"AutoScalingRollingUpdate": {
"MaxBatchSize": Integer,
"MinInstancesInService": Integer,
"MinSuccessfulInstancesPercent": Integer,
"PauseTime": String,
"SuspendProcesses": [ List of processes ],
"WaitOnResourceSignals": Boolean
}
}
```

Using a rolling update has a risk of system outages and performance degradation due to the decreased availability of your running EC2 instances. If you want to ensure high availability of your application, you can also use the AutoScalingReplacingUpdate policy to perform an immediate rollback of the stack without any possibility of failure.

To find out what's preventing your Auto Scaling group from updating correctly during a stack update, work through the following troubleshooting scenarios as needed:

Configure WaitOnResourceSignals and PauseTime to avoid problems with success signals

- In your AutoScalingRollingUpdate policy, set the WaitOnResourceSignals property to false. Take note that if WaitOnResourceSignals is set to true, PauseTime changes to a timeout value. AWS CloudFormation waits to receive a success signal until the maximum time specified by the PauseTime value. If a signal is not received, AWS CloudFormation cancels the update. Then, AWS CloudFormation rolls back the stack with the same settings, including the same PauseTime value.

Configure MinSuccessfulInstancesPercent to avoid stack rollback

- If you're replacing a large number of instances during a rolling update and waiting for a success signal for each instance, complete the following: In your AutoScalingRollingUpdate policy, set the value of the MinSuccessfulInstancesPercent property. Take note that setting the MinSuccessfulInstancesPercent property prevents AWS CloudFormation from rolling back the entire stack if only a single instance fails to launch.

Configure SuspendProcesses to avoid unexpected changes to the Auto Scaling group

- During a rolling update, suspend the following Auto Scaling processes: HealthCheck, ReplaceUnhealthy, AZRebalance, AlarmNotification, and ScheduledActions. It is quite important to know that if you're using your Auto Scaling group with Elastic Load Balancing (ELB), you should not suspend the following processes: Launch, Terminate, and AddToLoadBalancer. These processes are required to make rolling updates. Take note that if an unexpected scaling action changes the state of the Auto Scaling group during a rolling update, the update can fail. The failure can result from an inconsistent view of the group by AWS CloudFormation.

Based on the above information, the correct answers are:

- In your AutoScalingRollingUpdate policy, set the WaitOnResourceSignals property to false.
- In your AutoScalingRollingUpdate policy, set the value of the MinSuccessfulInstancesPercent property to prevent AWS CloudFormation from rolling back the entire stack if only a single instance fails to launch
- During a rolling update, suspend the following Auto Scaling processes: HealthCheck, ReplaceUnhealthy, AZRebalance, AlarmNotification, and ScheduledActions

The option that says: Switch from AutoScalingRollingUpdate to AutoScalingReplacingUpdate policy by modifying the UpdatePolicy of the AWS::AutoScaling::AutoscalingGroup resource in the CloudFormation template. Set the WillReplace property to true is incorrect because although the AutoScalingReplacingUpdate policy provides an immediate rollback of the stack without any possibility of failure, this solution is not warranted since the scenario asks for the options that will help troubleshoot the issue.

The option that says: Suspend the following Auto Scaling processes that are related with your ELB: Launch, Terminate, and AddToLoadBalancer is incorrect because these processes are required by the ELB to make rolling updates.

References:

https://aws.amazon.com/premiumsupport/knowledge-center/auto-scaling-group-rolling-updates/

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-as-group.html

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-updatepolicy.html#cfn-attributes-updatepolicy-replacingupdate

Check out this AWS CloudFormation Cheat Sheet:

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

In your AutoScalingRollingUpdate policy, set the value of the MinSuccessfulInstancesPercent property to prevent AWS CloudFormation from rolling back the entire stack if only a single instance fails to launch.

Explanation:-The AWS::AutoScaling::AutoScalingGroup resource uses the UpdatePolicy attribute to define how an Auto Scaling group resource is updated when the AWS CloudFormation stack is updated. If you don't have the right settings configured for the UpdatePolicy attribute, your rolling update can produce unexpected results.

You can use the AutoScalingRollingUpdate policy to control how AWS CloudFormation handles rolling updates for an Auto Scaling group. This common approach keeps the same Auto Scaling group, and then replaces the old instances based on the parameters that you set.

The AutoScalingRollingUpdate policy supports the following configuration options:

```
"UpdatePolicy": {
"AutoScalingRollingUpdate": {
"MaxBatchSize": Integer,
"MinInstancesInService": Integer,
"MinSuccessfulInstancesPercent": Integer,
"PauseTime": String,
"SuspendProcesses": [ List of processes ],
"WaitOnResourceSignals": Boolean
}
}
```

Using a rolling update has a risk of system outages and performance degradation due to the decreased availability of your running EC2 instances. If you want to ensure high availability of your application, you can also use the AutoScalingReplacingUpdate policy to perform an immediate rollback of the stack without any possibility of failure.

To find out what's preventing your Auto Scaling group from updating correctly during a stack update, work through the following troubleshooting

scenarios as needed:
Configure WaitOnResourceSignals and PauseTime to avoid problems with success signals
- In your AutoScalingRollingUpdate policy, set the WaitOnResourceSignals property to false. Take note that if WaitOnResourceSignals is set to true, PauseTime changes to a timeout value. AWS CloudFormation waits to receive a success signal until the maximum time specified by the PauseTime value. If a signal is not received, AWS CloudFormation cancels the update. Then, AWS CloudFormation rolls back the stack with the same settings, including the same PauseTime value.
Configure MinSuccessfulInstancesPercent to avoid stack rollback
- If you're replacing a large number of instances during a rolling update and waiting for a success signal for each instance, complete the following: In your AutoScalingRollingUpdate policy, set the value of the MinSuccessfulInstancesPercent property. Take note that setting the MinSuccessfulInstancesPercent property prevents AWS CloudFormation from rolling back the entire stack if only a single instance fails to launch.
Configure SuspendProcesses to avoid unexpected changes to the Auto Scaling group
- During a rolling update, suspend the following Auto Scaling processes: HealthCheck, ReplaceUnhealthy, AZRebalance, AlarmNotification, and ScheduledActions. It is quite important to know that if you're using your Auto Scaling group with Elastic Load Balancing (ELB), you should not suspend the following processes: Launch, Terminate, and AddToLoadBalancer. These processes are required to make rolling updates. Take note that if an unexpected scaling action changes the state of the Auto Scaling group during a rolling update, the update can fail. The failure can result from an inconsistent view of the group by AWS CloudFormation.
Based on the above information, the correct answers are:
- In your AutoScalingRollingUpdate policy, set the WaitOnResourceSignals property to false.
- In your AutoScalingRollingUpdate policy, set the value of the MinSuccessfulInstancesPercent property to prevent AWS CloudFormation from rolling back the entire stack if only a single instance fails to launch
- During a rolling update, suspend the following Auto Scaling processes: HealthCheck, ReplaceUnhealthy, AZRebalance, AlarmNotification, and ScheduledActions
The option that says: Switch from AutoScalingRollingUpdate to AutoScalingReplacingUpdate policy by modifying the UpdatePolicy of the AWS::AutoScaling::AutoscalingGroup resource in the CloudFormation template. Set the WillReplace property to true is incorrect because although the AutoScalingReplacingUpdate policy provides an immediate rollback of the stack without any possibility of failure, this solution is not warranted since the scenario asks for the options that will help troubleshoot the issue.
The option that says: Suspend the following Auto Scaling processes that are related with your ELB: Launch, Terminate, and AddToLoadBalancer is incorrect because these processes are required by the ELB to make rolling updates.
References:
https://aws.amazon.com/premiumsupport/knowledge-center/auto-scaling-group-rolling-updates/
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-as-group.html
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-updatepolicy.html#cfn-attributes-updatepolicy-replacingupdate
Check out this AWS CloudFormation Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-cloudformation/

Q13) A JavaScript-based online salary calculator hosted on-premises is slated to be migrated to AWS. The application has no server-side code and is just composed of a UI powered by Vue.js and Bootstrap. Since the online calculator may contain sensitive financial data, adding HTTP response headers such as X-Content-Type-Options, X-Frame-Options and X-XSS-Protection should be implemented to comply with the Open Web Application Security Project (OWASP) standards.
Which of the following is the MOST suitable solution that you should implement?

- Host the application on an S3 bucket configured for website hosting. Set up a CloudFront web distribution and set the S3 bucket as the origin. Set a custom Request and Response Behavior in CloudFront that automatically adds the required security headers in the HTTP response.
- Host the application on an S3 bucket configured for website hosting then set up server access logging on the Amazon S3 bucket to track user activity. Enable Amazon S3 client-side encryption and configure it to return the required security headers.
- Host the application on an S3 bucket configured for website hosting then set up server access logging on the S3 bucket to track user activity. Configure the bucket policy of the S3 bucket to return the required security headers.
- ✅ Host the application on an S3 bucket configured for website hosting. Set up a CloudFront web distribution and set the S3 bucket as the origin with the origin response event set to trigger a Lambda@Edge function. Add the required security headers in the HTTP response using the Lambda function.

Explanation:-Security headers are a group of headers in the HTTP response from a server that tell your browser how to behave when handling your site's content. For example, X-XSS-Protection is a header that Internet Explorer and Chrome respect to stop the pages from loading when they detect cross-site scripting (XSS) attacks.
The following are some examples of security headers:
Strict Transport Security
Content-Security-Policy
X-Content-Type-Options
X-Frame-Options
X-XSS-Protection
Referrer-Policy
Whenever you navigate to a website, your browser requests a web page, and the server responds with the content along with HTTP headers. Headers such as cache-control are used by the browser to determine how long to cache content for, others such as content-type are used to indicate the media type of a resource and therefore how to interpret such resource.
You can set up a solution that uses a simple single-page website, hosted in an Amazon S3 bucket and using Amazon CloudFront. You can create a new Lambda@Edge function and associate it with your CloudFront distribution. Then configure the origin response trigger to execute the Lambda@Edge function add the security headers in the HTTP response.
Hence, the correct answer is: Host the application on an S3 bucket configured for website hosting. Set up a CloudFront web distribution and set the S3 bucket as the origin with the origin response event set to trigger a Lambda@Edge function. Add the required security headers in the HTTP response using the Lambda function.
The option that says: Host the application on an S3 bucket configured for website hosting then set up server access logging on the Amazon S3 bucket to track user activity. Enable Amazon S3 client-side encryption and configure it to return the required security headers is incorrect because you have to integrate the S3 bucket with CloudFront and use Lambda@Edge to add the required headers. Take note that you can't return an HTTP response with the required security headers by just using client-side encryption alone.
The option that says: Host the application on an S3 bucket configured for website hosting then set up server access logging on the S3 bucket to track user activity. Configure the bucket policy of the S3 bucket to return the required security headers is incorrect because you can't return an HTTP response with the required security headers by simply configuring the bucket policy of the S3 bucket.
The option that says: Host the application on an S3 bucket configured for website hosting. Set up a CloudFront web distribution and set the S3 bucket as the origin. Set a custom Request and Response Behavior in CloudFront that automatically adds the required security headers in the HTTP response is incorrect because configuring a custom Request and Response Behavior in CloudFront is not enough to automatically add the required security headers to the HTTP response. You have to use Lambda@Edge to add the headers to satisfy the requirement of this scenario.

References:
https://aws.amazon.com/blogs/networking-and-content-delivery/adding-http-security-headers-using-lambdaedge-and-amazon-cloudfront/
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/RequestAndResponseBehaviorCustomOrigin.html#request-custom-headers-behavior
Check out this Amazon CloudFront Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudfront/

Q14) A company is receiving a high number of bad reviews from their customers lately because their website takes a lot of time to load. Upon investigation, there is a significant latency experienced whenever a user logs into their site. The DevOps Engineer configured CloudFront to serve the static content to its users around the globe, yet the problem still persists. There are times when their users get HTTP 504 errors in the login process. The engineers were tasked to fix this problem immediately to prevent users from unsubscribing on their website which will result in financial loss to the company.
Which combination of options below should the DevOps Engineer use together to set up a solution for this scenario with MINIMAL costs? (Select TWO)

● Deploy your application stack to multiple and geographically disperse VPCs on various AWS regions. Set up a Transit VPC to easily connect all your VPCs and resources. Using the AWS Serverless Application Model (SAM) service, deploy several AWS Lambda functions in each region to improve the overall application performance.
● Replicate your application stack to multiple AWS regions to serve your users around the world. Set up a Route 53 record with Latency routing policy that will automatically route traffic to the region with the best latency to the user.
● In CloudFront, add a Cache-Control max-age directive to your objects. Improve the cache hit ratio by setting the longest practical value for max-age of your CloudFront distribution.
✅ Create an origin group with two origins to set up an origin failover in Amazon CloudFront. Specify one as the primary origin and the other as the second origin. This configuration will cause the CloudFront service to automatically switch to the second origin in the event that the primary origin returns specific HTTP status code failure responses.
Explanation:-Amazon CloudFront is a web service that speeds up distribution of your static and dynamic web content, such as .html, .css, .js, and image files, to your users. CloudFront delivers your content through a worldwide network of data centers called edge locations. When a user requests content that you're serving with CloudFront, the user is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance.
Lambda@Edge lets you run Lambda functions to customize the content that CloudFront delivers, executing the functions in AWS locations closer to the viewer. The functions run in response to CloudFront events, without provisioning or managing servers. You can use Lambda functions to change CloudFront requests and responses at the following points:
- After CloudFront receives a request from a viewer (viewer request)
- Before CloudFront forwards the request to the origin (origin request)
- After CloudFront receives the response from the origin (origin response)
- Before CloudFront forwards the response to the viewer (viewer response)
In the given scenario, you can use Lambda@Edge to allow your Lambda functions to customize the content that CloudFront delivers and to execute the authentication process in AWS locations closer to the users. In addition, you can set up an origin failover by creating an origin group with two origins with one as the primary origin and the other as the second origin, which CloudFront automatically switches to when the primary origin fails. This will alleviate the occasional HTTP 504 errors that users are experiencing.
Hence, the correct answers in this scenario are:
- Use Lambda@Edge to customize the content that the CloudFront distribution delivers to your users across the globe. This will allow the Lambda functions to execute the authentication process in a specific AWS location that is proximate to the user.
- Create an origin group with two origins to set up an origin failover in Amazon CloudFront. Specify one as the primary origin and the other as the second origin. This configuration will cause the CloudFront service to automatically switch to the second origin in the event that the primary origin returns specific HTTP status code failure responses.
The option that says: Replicate your application stack to multiple AWS regions to serve your users around the world. Set up a Route 53 record with Latency routing policy that will automatically route traffic to the region with the best latency to the user is incorrect because although this may resolve the performance issue, this solution entails a significant implementation cost since you have to deploy your application to multiple AWS regions. Remember that the scenario asks for a solution that will improve the performance of the application with minimal cost.
The option that says: Deploy your application stack to multiple and geographically disperse VPCs on various AWS regions. Set up a Transit VPC to easily connect all your VPCs and resources. Using the AWS Serverless Application Model (SAM) service, deploy several AWS Lambda functions in each region to improve the overall application performance is incorrect because although setting up multiple VPCs across various regions which are connected with a transit VPC is valid, this solution still entails higher setup and maintenance costs. A more cost-effective option would be to use Lambda@Edge instead.
The option that says: In CloudFront, add a Cache-Control max-age directive to your objects. Improve the cache hit ratio by setting the longest practical value for max-age of your CloudFront distribution is incorrect because improving the cache hit ratio for the CloudFront distribution is irrelevant in this scenario. You can improve your cache performance by increasing the proportion of your viewer requests that are served from CloudFront edge caches instead of going to your origin servers for content. However, take note that the problem in the scenario is the slow authentication process of your global users and not just the caching of the static objects.
References:
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/high_availability_origin_failover.html
https://docs.aws.amazon.com/lambda/latest/dg/lambda-edge.html
Check out these Amazon CloudFront and AWS Lambda Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudfront/
https://tutorialsdojo.com/aws-cheat-sheet-aws-lambda/
Tutorials Dojo's AWS Certified DevOps Engineer Professional Exam Study Guide:
https://tutorialsdojo.com/aws-cheat-sheet-aws-certified-devops-engineer-professional/
✅ Use Lambda@Edge to customize the content that the CloudFront distribution delivers to your users across the globe. This will allow the Lambda functions to execute the authentication process in a specific AWS location that is proximate to the user.
Explanation:-Amazon CloudFront is a web service that speeds up distribution of your static and dynamic web content, such as .html, .css, .js, and image files, to your users. CloudFront delivers your content through a worldwide network of data centers called edge locations. When a user requests content that you're serving with CloudFront, the user is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance.
Lambda@Edge lets you run Lambda functions to customize the content that CloudFront delivers, executing the functions in AWS locations closer to the viewer. The functions run in response to CloudFront events, without provisioning or managing servers. You can use Lambda functions to change CloudFront requests and responses at the following points:
- After CloudFront receives a request from a viewer (viewer request)
- Before CloudFront forwards the request to the origin (origin request)
- After CloudFront receives the response from the origin (origin response)

- Before CloudFront forwards the response to the viewer (viewer response)
In the given scenario, you can use Lambda@Edge to allow your Lambda functions to customize the content that CloudFront delivers and to execute the authentication process in AWS locations closer to the users. In addition, you can set up an origin failover by creating an origin group with two origins with one as the primary origin and the other as the second origin, which CloudFront automatically switches to when the primary origin fails. This will alleviate the occasional HTTP 504 errors that users are experiencing.
Hence, the correct answers in this scenario are:
- Use Lambda@Edge to customize the content that the CloudFront distribution delivers to your users across the globe. This will allow the Lambda functions to execute the authentication process in a specific AWS location that is proximate to the user.
- Create an origin group with two origins to set up an origin failover in Amazon CloudFront. Specify one as the primary origin and the other as the second origin. This configuration will cause the CloudFront service to automatically switch to the second origin in the event that the primary origin returns specific HTTP status code failure responses.
The option that says: Replicate your application stack to multiple AWS regions to serve your users around the world. Set up a Route 53 record with Latency routing policy that will automatically route traffic to the region with the best latency to the user is incorrect because although this may resolve the performance issue, this solution entails a significant implementation cost since you have to deploy your application to multiple AWS regions. Remember that the scenario asks for a solution that will improve the performance of the application with minimal cost.
The option that says: Deploy your application stack to multiple and geographically disperse VPCs on various AWS regions. Set up a Transit VPC to easily connect all your VPCs and resources. Using the AWS Serverless Application Model (SAM) service, deploy several AWS Lambda functions in each region to improve the overall application performance is incorrect because although setting up multiple VPCs across various regions which are connected with a transit VPC is valid, this solution still entails higher setup and maintenance costs. A more cost-effective option would be to use Lambda@Edge instead.
The option that says: In CloudFront, add a Cache-Control max-age directive to your objects. Improve the cache hit ratio by setting the longest practical value for max-age of your CloudFront distribution is incorrect because improving the cache hit ratio for the CloudFront distribution is irrelevant in this scenario. You can improve your cache performance by increasing the proportion of your viewer requests that are served from CloudFront edge caches instead of going to your origin servers for content. However, take note that the problem in the scenario is the slow authentication process of your global users and not just the caching of the static objects.
References:
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/high_availability_origin_failover.html
https://docs.aws.amazon.com/lambda/latest/dg/lambda-edge.html
Check out these Amazon CloudFront and AWS Lambda Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudfront/
https://tutorialsdojo.com/aws-cheat-sheet-aws-lambda/
Tutorials Dojo's AWS Certified DevOps Engineer Professional Exam Study Guide:
https://tutorialsdojo.com/aws-cheat-sheet-aws-certified-devops-engineer-professional/

Q15) You are developing a store module that lets users choose which plugins they want to activate on your mobile app that is deployed on ECS Fargate. When the cluster was created, the service was launched with the LATEST platform version which is 1.2.0. However, there is a new update on the platform version (1.3.0) that supports a Splunk log driver which you are planning to use.
What is the recommended way of updating the platform version of the service on the AWS Console?

- Enable the automatic platform version upgrade feature in ECS.
- Update the service on ECS and select “Redeploy” on the deployment strategy so that the cluster will be re-deployed with the new platform version.
- Update the task definition with the new platform version ARN so that ECS re-deploys the cluster with the new platform version.
- ✅ Update the service on ECS and select “Force new deployment” so that the cluster will be re-deployed with the new platform version.

Explanation:-AWS Fargate platform versions are used to refer to a specific runtime environment for Fargate task infrastructure. It is a combination of the kernel and container runtime versions. New platform versions are released as the runtime environment evolves, for example, if there are kernel or operating system updates, new features, bug fixes, or security updates. Security updates and patches are deployed automatically for your Fargate tasks. If a security issue is found that affects a platform version, AWS patches the platform version. In some cases, you may be notified that your Fargate tasks have been scheduled for retirement
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 the unused capacity to scale down, you can reduce the number of desired tasks in your service and free up resources.
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 your service with your custom configuration, 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 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.
By default, deployments are not forced but you can use the forceNewDeployment request parameter (or the --force-new-deployment parameter if you are using the AWS CLI) to force a new deployment of the service. You can use this option to trigger a new deployment with no service definition changes. For example, you can update a service's tasks to use a newer Docker image with the same image/tag combination (my_image:latest) or to roll Fargate tasks onto a newer platform version.
Hence, the correct answer is: Update the service on ECS and select “Force new deployment” so that the cluster will be re-deployed with the new platform version.
The option that says: Update the service on ECS and select “Redeploy” on the deployment strategy so that the cluster will be re-deployed with the new platform version is incorrect because there is no "Redeploy" deployment strategy option in ECS.
The option that says: Update the task definition with the new platform version ARN so that ECS re-deploys the cluster with the new platform version is incorrect because modifying the task definition will cause a new version to be deployed but since the ARN or the image revision did not change, no further deployments will be made by ECS.
The option that says: Enable the automatic platform version upgrade feature in ECS is incorrect because there is no such feature in Amazon ECS.
References:
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/platform_versions.html
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/update-service.html
https://docs.aws.amazon.com/cli/latest/reference/ecs/update-service.html
Check out this Amazon ECS Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-elastic-container-service-amazon-ecs/

Q16) An insurance firm has recently undergone digital transformation and AWS cloud adoption. Its app development team has four environments, namely DEV, TEST, PRE-PROD, and PROD, for its flagship application that is configured with AWS CodePipeline. After several weeks, they noticed that there were several outages caused by misconfigured files or faulty code blocks that were deployed into the PROD environment. A DevOps Engineer has been assigned to add the required steps to identify issues in the application before it is released.
Which of the following is the MOST appropriate combination of steps that the Engineer should implement to identify functional issues during the deployment process? (Select TWO)

✅ In the pipeline, add an AWS CodeDeploy action to deploy the latest version of the application to the PRE-PROD environment. Set up a manual approval action in the pipeline so that the QA team can perform the required tests. Add another CodeDeploy action that deploys the verified code to the PROD environment after the manual approval action.

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.

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.

Hence, the correct answers are:

- Add a test action to the pipeline to run both the unit and functional tests using AWS CodeBuild. Verify that the test results passed before deploying the new application revision to the PROD environment.
- In the pipeline, add an AWS CodeDeploy action to deploy the latest version of the application to the PRE-PROD environment. Set up a manual approval action in the pipeline so that the QA team can perform the required tests. Add another CodeDeploy action that deploys the verified code to the PROD environment after the manual approval action.

The option that says: Add a test action that uses Amazon Inspector to the pipeline. Run an assessment using the Runtime Behavior Analysis rules package to verify that the deployed code complies with the strict security standards of the company before deploying it to the PROD environment is incorrect because Amazon inspector just checks the security vulnerabilities of the EC2 instances and not the application functionality itself.

The option that says: Add a test action that uses Amazon GuardDuty to the pipeline. Run an assessment using the Runtime Behavior Analysis rules package to verify that the deployed code complies with the strict security standards of the company before deploying it to the PROD environment is incorrect because there is no Runtime Behavior Analysis rules package in Amazon GuardDuty.

The option that says: Migrate the pipeline from CodePipeline to AWS Data Pipeline to enable more CI/CD features. Add a test action that uses Amazon Macie to the pipeline. Run an assessment using the Runtime Behavior Analysis rules package to verify that the deployed code complies with the strict security standards of the company before deploying it to the PROD environment is incorrect because AWS Data Pipeline is a web service that simply helps you to reliably process and move data between different AWS compute and storage services, as well as on-premises data sources, at specified intervals. This service can't be used in your CI/CD process. Moreover, there is no Runtime Behavior Analysis rules package in Amazon Macie.

References:

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

https://docs.aws.amazon.com/codepipeline/latest/userguide/approvals-action-add.html

https://docs.aws.amazon.com/codepipeline/latest/userguide/best-practices.html

Check out these AWS CloudFormation and CodePipeline Cheat Sheets:

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

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

⚫ Migrate the pipeline from CodePipeline to AWS Data Pipeline to enable more CI/CD features. Add a test action that uses Amazon Macie to the pipeline. Run an assessment using the Runtime Behavior Analysis rules package to verify that the deployed code complies with the strict security standards of the company before deploying it to the PROD environment.

⚫ Add a test action that uses Amazon Inspector to the pipeline. Run an assessment using the Runtime Behavior Analysis rules package to verify that the deployed code complies with the strict security standards of the company before deploying it to the PROD environment.

✅ Add a test action to the pipeline to run both the unit and functional tests using AWS CodeBuild. Verify that the test results passed before deploying the new application revision to the PROD environment.

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.

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.

Hence, the correct answers are:

- Add a test action to the pipeline to run both the unit and functional tests using AWS CodeBuild. Verify that the test results passed before deploying the new application revision to the PROD environment.
- In the pipeline, add an AWS CodeDeploy action to deploy the latest version of the application to the PRE-PROD environment. Set up a manual approval action in the pipeline so that the QA team can perform the required tests. Add another CodeDeploy action that deploys the verified code to the PROD environment after the manual approval action.

The option that says: Add a test action that uses Amazon Inspector to the pipeline. Run an assessment using the Runtime Behavior Analysis rules package to verify that the deployed code complies with the strict security standards of the company before deploying it to the PROD environment is incorrect because Amazon inspector just checks the security vulnerabilities of the EC2 instances and not the application functionality itself.

The option that says: Add a test action that uses Amazon GuardDuty to the pipeline. Run an assessment using the Runtime Behavior Analysis rules package to verify that the deployed code complies with the strict security standards of the company before deploying it to the PROD environment is incorrect because there is no Runtime Behavior Analysis rules package in Amazon GuardDuty.

The option that says: Migrate the pipeline from CodePipeline to AWS Data Pipeline to enable more CI/CD features. Add a test action that uses Amazon Macie to the pipeline. Run an assessment using the Runtime Behavior Analysis rules package to verify that the deployed code complies with the strict security standards of the company before deploying it to the PROD environment is incorrect because AWS Data Pipeline is a web service that simply helps you to reliably process and move data between different AWS compute and storage services, as well as on-premises data sources, at specified intervals. This service can't be used in your CI/CD process. Moreover, there is no Runtime Behavior Analysis rules package in Amazon Macie.

References:

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

https://docs.aws.amazon.com/codepipeline/latest/userguide/approvals-action-add.html

https://docs.aws.amazon.com/codepipeline/latest/userguide/best-practices.html

Check out these AWS CloudFormation and CodePipeline Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-aws-cloudformation/
https://tutorialsdojo.com/aws-cheat-sheet-aws-codepipeline/

Add a test action that uses Amazon GuardDuty to the pipeline. Run an assessment using the Runtime Behavior Analysis rules package to verify that the deployed code complies with the strict security standards of the company before deploying it to the PROD environment.

Q17) A multinational corporation has multiple AWS accounts that are consolidated using AWS Organizations. For security purposes, a new system should be configured that automatically detects suspicious activities in any of its accounts, such as SSH brute force attacks or compromised EC2 instances that serve malware. All of the gathered information must be centrally stored in its dedicated security account for audit purposes, and the events should be stored in an S3 bucket.
As a DevOps Engineer, which solution should you implement in order to meet this requirement?

Automatically detect SSH brute force or malware attacks by enabling Amazon Macie in the security account only. Configure the security account as the Macie Administrator for every member account. Set up a new CloudWatch Events rule in the security account. Configure the rule to send all findings to Amazon Kinesis Data Streams. Launch a custom shell script in Lambda function to read data from the Kinesis Data Streams and push them to the S3 bucket.

Automatically detect SSH brute force or malware attacks by enabling Amazon Macie in every account. Set up the security account as the Macie Administrator for every member account of the organization. Create an Amazon CloudWatch Events rule in the security account. Configure the rule to send all findings to Amazon Kinesis Data Firehose, which should push the findings to an Amazon S3 bucket.

Automatically detect SSH brute force or malware attacks by enabling Amazon GuardDuty in the security account only. Set up the security account as the GuardDuty Administrator for every member account. Create a new CloudWatch rule in the security account. Configure the rule to send all findings to Amazon Kinesis Data Streams. Launch a custom shell script in Lambda function to read data from the Kinesis Data Streams and push them to the S3 bucket.

✅ Automatically detect SSH brute force or malware attacks by enabling Amazon GuardDuty in every account. Configure the security account as the GuardDuty Administrator for every member of the organization. Set up a new CloudWatch rule in the security account. Configure the rule to send all findings to Amazon Kinesis Data Firehose, which will push the findings to the S3 bucket.

Explanation:-Amazon GuardDuty is a threat detection service that continuously monitors for malicious activity and unauthorized behavior to protect your AWS accounts and workloads. With the cloud, the collection and aggregation of account and network activities is simplified, but it can be time consuming for security teams to continuously analyze event log data for potential threats. With GuardDuty, you now have an intelligent and cost-effective option for continuous threat detection in the AWS Cloud. The service uses machine learning, anomaly detection, and integrated threat intelligence to identify and prioritize potential threats. GuardDuty analyzes tens of billions of events across multiple AWS data sources, such as AWS CloudTrail, Amazon VPC Flow Logs, and DNS logs. With a few clicks in the AWS Management Console, GuardDuty can be enabled with no software or hardware to deploy or maintain. By integrating with AWS CloudWatch Events, GuardDuty alerts are actionable, easy to aggregate across multiple accounts, and straightforward to push into existing event management and workflow systems.
GuardDuty makes enablement and management across multiple accounts easy. Through the multi-account feature, all member accounts findings can be aggregated with a GuardDuty administrator account. This enables security team to manage all GuardDuty findings from across the organization in one single account. The aggregated findings are also available through CloudWatch Events, making it easy to integrate with an existing enterprise event management system.
Hence, the correct answer is: Automatically detect SSH brute force or malware attacks by enabling Amazon GuardDuty in every account. Configure the security account as the GuardDuty Administrator for every member of the organization. Set up a new CloudWatch rule in the security account. Configure the rule to send all findings to Amazon Kinesis Data Firehose, which will push the findings to the S3 bucket.
The option that says: Automatically detect SSH brute force or malware attacks by enabling Amazon Macie in every account. Set up the security account as the Macie Administrator for every member account of the organization. Create an Amazon CloudWatch Events rule in the security account. Configure the rule to send all findings to Amazon Kinesis Data Firehose, which should push the findings to an Amazon S3 bucket is incorrect because you have to use Amazon GuardDuty instead of Amazon Macie. Take note that Amazon Macie cannot detect SSH brute force or malware attacks.
The option that says: Automatically detect SSH brute force or malware attacks by enabling Amazon Macie in the security account only. Configure the security account as the Macie Administrator for every member account. Set up a new CloudWatch Events rule in the security account. Configure the rule to send all findings to Amazon Kinesis Data Streams. Launch a custom shell script in Lambda function to read data from the Kinesis Data Streams and push them to the S3 bucket is incorrect because you don't need to create a custom shell script in Lambda or use Kinesis Data Streams. You can simply configure the CloudWatch Event rule to send all findings to Amazon Kinesis Data Firehose, which will push the findings to the S3 bucket.
The option that says: Automatically detect SSH brute force or malware attacks by enabling Amazon GuardDuty in the security account only. Set up the security account as the GuardDuty Administrator for every member account. Create a new CloudWatch rule in the security account. Configure the rule to send all findings to Amazon Kinesis Data Streams. Launch a custom shell script in Lambda function to read data from the Kinesis Data Streams and push them to the S3 bucket is incorrect because although it is valid to use Amazon GuardDuty in this scenario, the implementation for storing the findings is incorrect. You can simply configure the CloudWatch Event rule to send all findings to Amazon Kinesis Data Firehose, which will push the findings to the S3 bucket.
References:
https://aws.amazon.com/guardduty/
https://aws.amazon.com/blogs/security/how-to-manage-amazon-guardduty-security-findings-across-multiple-accounts/
Check out this Amazon GuardDuty Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-guardduty/

Q18) You are the DevOps Engineer for an IT consulting firm that has various teams and departments. Using AWS Organizations, these entities have been grouped into several organizational units (OUs). The IT Security team reported that there was a suspected breach in your environment where a third-party AWS account was suddenly added to your organization without any prior approval. The external account has high-level access privileges to the accounts that you own but fortunately, no detrimental action was performed.
Which of the following is the MOST appropriate monitoring set up that notifies you for any changes to your AWS accounts? (Select TWO)

✅ Launch a new trail in Amazon CloudTrail to capture all API calls to your AWS Organizations, including calls from the AWS Organizations console. Also, track all code calls to the AWS Organizations APIs. Integrate CloudWatch Events and Amazon SNS to raise events when administrator-specified actions occur in an organization and configure it to send a notification.

✅ Monitor the compliance of your AWS Organizations using AWS Config. Launch a new SNS Topic or Amazon CloudWatch Events that will send alerts to you for any changes.

Use the AWS Systems Manager and CloudWatch Events to monitor all changes to your organization and also to notify you of any new activities or configurations made to your account.

Create an Amazon CloudWatch Dashboard in order to monitor any changes to your organization. Launch a new Amazon SNS topic that would

send you and your team a notification.

Launch an AWS-approved third-party monitoring tool from the AWS Marketplace that would send alerts if a breach was detected. Analyze any possible breach using AWS GuardDuty. Use Amazon SNS to notify the administrators.

Q19) A DevOps Engineer is designing a service that aggregates clickstream data in real-time. The service should also deliver a report to its subscribers once a week via email only. The data being handled are geographically distributed, high volume, and unpredictable. The service should identify and create sessions from real-time clickstream events with a feature to do an ad hoc analysis.
Which among the options below is the MOST suitable solution that the Engineer should implement with the LEAST amount of cost?

Collect the real-time clickstream data using a custom Amazon EMR application hosted on extra-large EC2 instances then build and analyze the sessions using Kinesis Data Analytics. The aggregated analytics will trigger the real-time events on Lambda and then send them to SQS which in turn, sends data to an S3 bucket. The clickstream data is ingested to a table by an AWS Glue crawler that will be used by Amazon Athena for running queries and ad hoc analysis.

Collect the real-time clickstream data using Amazon CloudWatch Events then build and analyze the sessions using Kinesis Data Analytics. The aggregated analytics will trigger the real-time events on Lambda and then send them to Kinesis Data Firehose which in turn sends data to an S3 bucket. The clickstream data is ingested to a table by an AWS Glue crawler that will be used by Amazon Athena for running queries and ad hoc analysis.

✅ Collect the real-time clickstream data using Amazon Kinesis Data Stream then build and analyze the sessions using Kinesis Data Analytics. The aggregated analytics will trigger the real-time events on Lambda and then send them to Kinesis Data Firehose which in turn sends data to an S3 bucket. The clickstream data is ingested to a table by an AWS Glue crawler that will be used by Amazon Athena for running queries and ad hoc analysis.

Explanation:-Amazon Kinesis makes it easy to collect, process, and analyze real-time, streaming data so you can get timely insights and react quickly to new information. Amazon Kinesis offers key capabilities to cost-effectively process streaming data at any scale, along with the flexibility to choose the tools that best suit the requirements of your application. With Amazon Kinesis, you can ingest real-time data such as video, audio, application logs, website clickstreams, and IoT telemetry data for machine learning, analytics, and other applications. Amazon Kinesis enables you to process and analyze data as it arrives and respond instantly instead of having to wait until all your data is collected before the processing can begin.
Hence, the correct answer is: Collect the real-time clickstream data using Amazon Kinesis Data Stream then build and analyze the sessions using Kinesis Data Analytics. The aggregated analytics will trigger the real-time events on Lambda and then send them to Kinesis Data Firehose which in turn, sends data to an S3 bucket. The clickstream data is ingested to a table by an AWS Glue crawler that will be used by Amazon Athena for running queries and ad hoc analysis.
The option that says: Collect the real-time clickstream data using Amazon CloudWatch Events then build and analyze the sessions using Kinesis Data Analytics. The aggregated analytics will trigger the real-time events on Lambda and then send them to Kinesis Data Firehose which in turn, sends data to an S3 bucket. The clickstream data is ingested to a table by an AWS Glue crawler that will be used by Amazon Athena for running queries and ad hoc analysis is incorrect because you can't collect real-time clickstream data using Amazon CloudWatch Events. You have to use Amazon Kinesis Data Streams instead.
The option that says: Collect the real-time clickstream data using a custom Amazon EMR application hosted on extra-large EC2 instances then build and analyze the sessions using Kinesis Data Analytics. The aggregated analytics will trigger the real-time events on Lambda and then send them to SQS which in turn, sends data to an S3 bucket. The clickstream data is ingested to a table by an AWS Glue crawler that will be used by Amazon Athena for running queries and ad hoc analysis is incorrect because although it is possible to collect the clickstream using Amazon EMR, it entails a significant amount of cost due to the use of extra-large EC2 instances. Using Amazon Kinesis is a more cost-effective option.
The option that says: Collect the real-time clickstream data using Amazon SQS then build and analyze the sessions using Kinesis Data Analytics. The aggregated analytics will trigger the real-time events on Lambda and then send them to Kinesis Data Firehose which in turn, sends data to an Instance Store-backed EC2 instance. The clickstream data is ingested to a table by an AWS Glue crawler that will be used by Amazon Athena for running queries and ad hoc analysis is incorrect because using Amazon SQS is not the appropriate service to use for collecting clickstream data in real time.
References:
https://aws.amazon.com/blogs/big-data/create-real-time-clickstream-sessions-and-run-analytics-with-amazon-kinesis-data-analytics-aws-glue-and-amazon-athena/
https://docs.aws.amazon.com/solutions/latest/real-time-web-analytics-with-kinesis/architecture.html
Check out these Amazon Kinesis and Athena Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-kinesis/
https://tutorialsdojo.com/aws-cheat-sheet-amazon-athena/

Collect the real-time clickstream data using Amazon SQS then build and analyze the sessions using Kinesis Data Analytics. The aggregated analytics will trigger the real-time events on Lambda and then send them to Kinesis Data Firehose which in turn, sends data to an Instance Store-backed EC2 instance. The clickstream data is ingested to a table by an AWS Glue crawler that will be used by Amazon Athena for running queries and ad hoc analysis.

Q20) Your development team uploads a new version of the company's web store app on AWS ECR. To simplify tracking of versions, they tag the latest image as "store-prd:latest". You are automating the deployment of the new version using AWS CLI command so that the new image will be pulled from ECR and deployed on your ECS Fargate cluster. However, the latest image is not being deployed when you run the following aws cli command:
aws ecs update-service --service store-prd --task-definition task-store-prd
Which of the following options will solve this problem?

Update the task definition with the new image ARN from ECR so that ECS re-deploys the cluster pulling the new image.

Add --platform-version=latest option to your AWS CLI command so that ECS re-deploys the cluster pulling the new image.

Add --re-deploy=yes option to the AWS CLI command so that ECS re-deploys the cluster pulling the new image.

✅ Add --force-new-deployment option to the AWS CLI command so that ECS re-deploys your cluster pulling the new image.

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 the unused capacity to scale down, you can reduce the number of desired tasks in your service and free up resources.
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 your service with your custom configuration, 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 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.

By default, deployments are not forced but you can use the forceNewDeployment request parameter (or the --force-new-deployment parameter if you are using the AWS CLI) to force a new deployment of the service. You can use this option to trigger a new deployment with no service definition changes. For example, you can update a service's tasks to use a newer Docker image with the same image/tag combination (my_image:latest) or to roll Fargate tasks onto a newer platform version.
Hence, the correct answer is: Add --force-new-deployment option to the AWS CLI command so that ECS re-deploys your cluster pulling the new image.
The option that says: Add --re-deploy=yes option to the AWS CLI command so that ECS re-deploys the cluster pulling the new image is incorrect because there is no such option in the ecs update-service command. You have to use the --force-new-deployment option instead.
The option the says: Update the task definition with the new image ARN from ECR so that ECS re-deploys the cluster pulling the new image is incorrect because although modifying the task definition will cause a new version to be deployed, no further deployments will be made by ECS since the ARN or the image revision remain unchanged.
The option that says: Add --platform-version=latest option to your AWS CLI command so that ECS re-deploys the cluster pulling the new image is incorrect because this option is primarily used to specify the Fargate platform that will be used by ECS and not the image version you want to deploy.
References:
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/platform_versions.html
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/update-service.html
https://docs.aws.amazon.com/cli/latest/reference/ecs/update-service.html
Check out this Amazon ECS Cheat Sheet:
https://tutorialsdojo.com/amazon-elastic-container-service-amazon-ecs/

Q21) A leading software development company is using AWS CodeCommit as the source control repository of one of its cloud-based applications. There is a new requirement to automate its continuous integration and continuous deployment pipeline on its development, testing, and production environments. A security code review must be done to ensure that there is no leaked Personally Identifiable Information (PII) or other sensitive data. Each change should go through both unit testing as well as functional testing. Any code push to CodeCommit should automatically trigger the CI/CD pipeline, and an email notification to devops-administrators@tutorialsdojo.com should be sent in the event of build or deployment failures. In addition, an approval to stage the assets to Amazon S3 after tests should also be performed. The solution must strictly follow the CI/CD best practices.
Which among the following options should a DevOps Engineer implement to satisfy all of the requirements above?

● Configure AWS CodePipeline to have a trigger to start off the pipeline when a new code change is committed on the master branch. Add the required stages in the pipeline for security review, unit tests, functional tests, and manual approval. Use CloudWatch Logs to detect changes in pipeline stages. Send an email notification to devops-administrators@tutorialsdojo.com using Amazon SES.
● Configure AWS CodePipeline to have a trigger to start off the pipeline when a new code change is committed on a certain branch. Add the required stages in the pipeline for security review, unit tests, functional tests, and manual approval. Use CloudWatch Events to detect changes in pipeline stages. Send an email notification to devops-administrators@tutorialsdojo.com using Amazon SES.
● Configure the AWS CodePipeline to have a trigger to start off the pipeline when a new code change is committed on the master branch. In the pipeline, set up the required stages for security review, unit tests, functional tests, and manual approval. Use CloudTrail to detect changes in pipeline stages. Send an email notification to devops-administrators@tutorialsdojo.com using Amazon SNS.
✅ Configure the AWS CodePipeline to have a trigger to start off the pipeline when a new code change is committed on a certain code branch. In the pipeline, set up the required stages for security review, unit tests, functional tests, and manual approval. Use Amazon CloudWatch Events to detect changes in pipeline stages. Send an email notification to devops-administrators@tutorialsdojo.com using Amazon SNS.
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.
Hence, the correct answer is: Configure the AWS CodePipeline to have a trigger to start off the pipeline when a new code change is committed on a certain code branch. In the pipeline, set up the required stages for security review, unit tests, functional tests, and manual approval. Use Amazon CloudWatch Events to detect changes in pipeline stages. Send an email notification to devops-administrators@tutorialsdojo.com using Amazon SNS.
The option that says: Configure AWS CodePipeline to have a trigger to start off the pipeline when a new code change is committed on a certain branch. Add the required stages in the pipeline for security review, unit tests, functional tests, and manual approval. Use CloudWatch Events to detect changes in pipeline stages. Send an email notification to devops-administrators@tutorialsdojo.com using Amazon SES is incorrect because you have to use Amazon SNS in order to send an email notification and not Amazon SES.
The option that says: Configure the AWS CodePipeline to have a trigger to start off the pipeline when a new code change is committed on the master branch. In the pipeline, set up the required stages for security review, unit tests, functional tests, and manual approval. Use CloudTrail to detect changes in pipeline stages. Send an email notification to devops-administrators@tutorialsdojo.com using Amazon SNS is incorrect because you have to use CloudWatch Events to detect the changes in the pipeline stages and not CloudTrail.
The option that says: Configure AWS CodePipeline to have a trigger to start off the pipeline when a new code change is committed on the master branch. Add the required stages in the pipeline for security review, unit tests, functional tests, and manual approval. Use CloudWatch Logs to detect changes in pipeline stages. Send an email notification to devops-administrators@tutorialsdojo.com using Amazon SES is incorrect because using CloudWatch Logs is not the appropriate service to use in this scenario. In order to automatically detect the changes in the pipeline stages, the most suitable service that you have to use is Amazon CloudWatch Events.
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
Check out this AWS CodePipeline Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-codepipeline/

Q22) A leading IT consulting firm is building a GraphQL API service and a mobile application that lets people post photos and videos of the traffic situations and other issues in the city's public roads. Users can include a text report and constructive feedback to the authorities. The department of public works shall rectify the problems based on the data gathered by the

system. In order for the mobile app to run on various mobile and tablet devices, the firm decided to develop it using the React Native mobile framework, which will consume and send data to the GraphQL API. The backend service will be responsible for storing the photos and videos in an Amazon S3 bucket. The API will also need access to the Amazon DynamoDB database to store the text reports. The firm has recently deployed the mobile app prototype, however, during testing, the GraphQL API showed a lot of issues. The team decided to remove the API to proceed with the project and refactor the mobile application instead so that it will directly connect to both DynamoDB and S3 as well as handle user authentication.
Which of the following options provides a cost-effective and scalable architecture for this project? (Select TWO)

Create an identity pool in AWS Identity and Access Management (IAM) that will be used to store the end-user identities organized for your mobile app. IAM will automatically create the required IAM roles for authenticated identities as well as for unauthenticated "guest" identities that define permissions for the users. Download and integrate the AWS SDK for React Native with your app, and import the files required to use IAM. Configure your app to pass the credentials provider instance to the client object, which passes the temporary security credentials to the client.

Using the STS AssumeRoleWithWebIdentity API, set up a web identity federation and register with social identity providers like Facebook, Google or any other OpenID Connect (OIDC)-compatible IdP. Create a new IAM Role and grant permissions to allow access to Amazon S3 and DynamoDB. Configure the mobile application to use the AWS temporary security credentials to store the photos and videos to an S3 bucket and persist the text-based reports to the DynamoDB table.

Explanation:-With web identity federation, you don't need to create custom sign-in code or manage your own user identities. Instead, users of your app can sign in using a well-known external identity provider (IdP), such as Login with Amazon, Facebook, Google, or any of your OpenID Connect (OIDC)-compatible IdP. They can receive an authentication token, and then exchange that token for temporary security credentials in AWS that map to an IAM role with permissions to use the resources in your AWS account. Using an IdP helps you keep your AWS account secure because you don't have to embed and distribute long-term security credentials with your application.

The preferred way to use web identity federation is to use Amazon Cognito. For example, You are a developer that builds a game for a mobile device where each user data such as scores and profiles are stored in Amazon S3 and Amazon DynamoDB. You could also store this data locally on the device and use Amazon Cognito to keep it synchronized across devices. You know that for security and maintenance reasons, long-term AWS security credentials should not be distributed with the game. You might also know that the game might have a large number of users. For all of these reasons, you don't want to create new user identities in IAM for each player. Instead, you build the game so that users can sign in using an identity that they've already established with a well-known external identity provider (IdP), such as Login with Amazon, Facebook, Google, or any OpenID Connect (OIDC)-compatible IdP. Your game can take advantage of the authentication mechanism from one of these providers to validate the user's identity.

To enable the mobile app to access your AWS resources, you should first register for a developer ID with your chosen IdPs. You can also configure the application with each of these providers. In your AWS account that contains the Amazon S3 bucket and DynamoDB table for the game, you should use Amazon Cognito to create IAM roles that precisely define permissions that the game needs. If you are using an OIDC IdP, you can also create an IAM OIDC identity provider entity to establish trust between your AWS account and the IdP.

In the app's code, you can call the sign-in interface for the IdP that you configured previously. The IdP handles all the details of letting the user sign in, and the app gets an OAuth access token or OIDC ID token from the provider. Your mobile app can trade this authentication information for a set of temporary security credentials that consist of an AWS access key ID, a secret access key, and a session token. The app can then use these credentials to access web services offered by AWS. The app is limited to the permissions that are defined in the role that it assumes.

In this scenario, you have a mobile app that needs to have access to the DynamoDB and S3 bucket. You can achieve this by using Web Identity Federation with AssumeRoleWithWebIdentity API which provides temporary security credentials and an IAM role. You can also use Amazon Cognito to simplify the process.

Hence, the correct answers are:

- Using the STS AssumeRoleWithWebIdentity API, set up a web identity federation and register with social identity providers like Facebook, Google or any other OpenID Connect (OIDC)-compatible IdP. Create a new IAM Role and grant permissions to allow access to Amazon S3 and DynamoDB. Configure the mobile application to use the AWS temporary security credentials to store the photos and videos to an S3 bucket and persist the text-based reports to the DynamoDB table.
- Create an identity pool in Amazon Cognito that will be used to store the end-user identities organized for your mobile app. Amazon Cognito will automatically create the required IAM roles for authenticated identities as well as for unauthenticated "guest" identities that define permissions for Amazon Cognito users. Download and integrate the AWS SDK for React Native with your app, and import the files required to use Amazon Cognito. Configure your app to pass the credentials provider instance to the client object, which passes the temporary security credentials to the client.

The option that says: Create an identity pool in AWS Identity and Access Management (IAM) that will be used to store the end-user identities organized for your mobile app. IAM will automatically create the required IAM roles for authenticated identities as well as for unauthenticated "guest" identities that define permissions for the users. Download and integrate the AWS SDK for React Native with your app, and import the files required to use IAM. Configure your app to pass the credentials provider instance to the client object, which passes the temporary security credentials to the client is incorrect because you cannot create identity pools with guest identities using the AWS Identity and Access Management (IAM) service. You can only implement this using Amazon Cognito.

The option that says: Using the STS AssumeRoleWithSAML API, set up a web identity federation and register with various social identity providers like Facebook, Google, or any other OpenID Connect (OIDC)-compatible IdP. Set up an IAM role for that provider and grant permissions for the IAM role to allow access to Amazon S3 and DynamoDB. Configure the mobile app to use the AWS temporary security credentials to store the photos and videos to an S3 bucket and persist the text-based reports to the DynamoDB table is incorrect because you should have used the AssumRoleWithWebIdentity API instead of AssumeRoleWithSAML, as this is used in SAML authentication response and not for web identity authentication.

The option that says: Using the STS AssumeRoleWithWebIdentity API, set up a web identity federation and register with various social identity providers like Facebook, Google, or any other OpenID Connect (OIDC)-compatible IdP. Set up an IAM role for that provider and grant permissions for the IAM role to allow access to Amazon S3 and DynamoDB. Configure the mobile application to use the AWS access and secret keys to store the photos and videos to an S3 bucket and persist the text-based report to a DynamoDB table is incorrect because even though the use of Amazon Cognito is valid, it is wrong to store and use the AWS access and secret keys from the mobile app itself. This is a security risk and you should use the temporary security credentials instead.

References:
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_oidc_cognito.html
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_oidc.html
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp.html

Check out this AWS IAM Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-identity-and-access-management-iam/

Using the STS AssumeRoleWithSAML API, set up a web identity federation and register with various social identity providers like Facebook, Google, or any other OpenID Connect (OIDC)-compatible IdP. Set up an IAM role for that provider and grant permissions for the IAM role to allow access to Amazon S3 and DynamoDB. Configure the mobile app to use the AWS temporary security credentials to store the photos and videos to an S3 bucket and persist the text-based reports to the DynamoDB table.

Create an identity pool in Amazon Cognito that will be used to store the end-user identities organized for your mobile app. Amazon Cognito will automatically create the required IAM roles for authenticated identities as well as for unauthenticated "guest" identities that define permissions for Amazon Cognito users. Download and integrate the AWS SDK for React Native with your app, and import the files required to use Amazon Cognito. Configure your app to pass the credentials provider instance to the client object, which passes the temporary security credentials to the client.

Explanation:-With web identity federation, you don't need to create custom sign-in code or manage your own user identities. Instead, users of your app can sign in using a well-known external identity provider (IdP), such as Login with Amazon, Facebook, Google, or any of your OpenID Connect (OIDC)-compatible IdP. They can receive an authentication token, and then exchange that token for temporary security credentials in AWS that map to an IAM role with permissions to use the resources in your AWS account. Using an IdP helps you keep your AWS account secure because you don't have to embed and distribute long-term security credentials with your application.
The preferred way to use web identity federation is to use Amazon Cognito. For example, You are a developer that builds a game for a mobile device where each user data such as scores and profiles are stored in Amazon S3 and Amazon DynamoDB. You could also store this data locally on the device and use Amazon Cognito to keep it synchronized across devices. You know that for security and maintenance reasons, long-term AWS security credentials should not be distributed with the game. You might also know that the game might have a large number of users. For all of these reasons, you don't want to create new user identities in IAM for each player. Instead, you build the game so that users can sign in using an identity that they've already established with a well-known external identity provider (IdP), such as Login with Amazon, Facebook, Google, or any OpenID Connect (OIDC)-compatible IdP. Your game can take advantage of the authentication mechanism from one of these providers to validate the user's identity.
To enable the mobile app to access your AWS resources, you should first register for a developer ID with your chosen IdPs. You can also configure the application with each of these providers. In your AWS account that contains the Amazon S3 bucket and DynamoDB table for the game, you should use Amazon Cognito to create IAM roles that precisely define permissions that the game needs. If you are using an OIDC IdP, you can also create an IAM OIDC identity provider entity to establish trust between your AWS account and the IdP.
In the app's code, you can call the sign-in interface for the IdP that you configured previously. The IdP handles all the details of letting the user sign in, and the app gets an OAuth access token or OIDC ID token from the provider. Your mobile app can trade this authentication information for a set of temporary security credentials that consist of an AWS access key ID, a secret access key, and a session token. The app can then use these credentials to access web services offered by AWS. The app is limited to the permissions that are defined in the role that it assumes.
In this scenario, you have a mobile app that needs to have access to the DynamoDB and S3 bucket. You can achieve this by using Web Identity Federation with AssumeRoleWithWebIdentity API which provides temporary security credentials and an IAM role. You can also use Amazon Cognito to simplify the process.
Hence, the correct answers are:
- Using the STS AssumeRoleWithWebIdentity API, set up a web identity federation and register with social identity providers like Facebook, Google or any other OpenID Connect (OIDC)-compatible IdP. Create a new IAM Role and grant permissions to allow access to Amazon S3 and DynamoDB. Configure the mobile application to use the AWS temporary security credentials to store the photos and videos to an S3 bucket and persist the text-based reports to the DynamoDB table.
- Create an identity pool in Amazon Cognito that will be used to store the end-user identities organized for your mobile app. Amazon Cognito will automatically create the required IAM roles for authenticated identities as well as for unauthenticated "guest" identities that define permissions for Amazon Cognito users. Download and integrate the AWS SDK for React Native with your app, and import the files required to use Amazon Cognito. Configure your app to pass the credentials provider instance to the client object, which passes the temporary security credentials to the client.
The option that says: Create an identity pool in AWS Identity and Access Management (IAM) that will be used to store the end-user identities organized for your mobile app. IAM will automatically create the required IAM roles for authenticated identities as well as for unauthenticated "guest" identities that define permissions for the users. Download and integrate the AWS SDK for React Native with your app, and import the files required to use IAM. Configure your app to pass the credentials provider instance to the client object, which passes the temporary security credentials to the client is incorrect because you cannot create identity pools with guest identities using the AWS Identity and Access Management (IAM) service. You can only implement this using Amazon Cognito.
The option that says: Using the STS AssumeRoleWithSAML API, set up a web identity federation and register with various social identity providers like Facebook, Google, or any other OpenID Connect (OIDC)-compatible IdP. Set up an IAM role for that provider and grant permissions for the IAM role to allow access to Amazon S3 and DynamoDB. Configure the mobile app to use the AWS temporary security credentials to store the photos and videos to an S3 bucket and persist the text-based reports to the DynamoDB table is incorrect because you should have used the AssumRoleWithWebIdentity API instead of AssumeRoleWithSAML, as this is used in SAML authentication response and not for web identity authentication.
The option that says: Using the STS AssumeRoleWithWebIdentity API, set up a web identity federation and register with various social identity providers like Facebook, Google, or any other OpenID Connect (OIDC)-compatible IdP. Set up an IAM role for that provider and grant permissions for the IAM role to allow access to Amazon S3 and DynamoDB. Configure the mobile application to use the AWS access and secret keys to store the photos and videos to an S3 bucket and persist the text-based report to a DynamoDB table is incorrect because even though the use of Amazon Cognito is valid, it is wrong to store and use the AWS access and secret keys from the mobile app itself. This is a security risk and you should use the temporary security credentials instead.
References:
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_oidc_cognito.html
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_oidc.html
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp.html
Check out this AWS IAM Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-identity-and-access-management-iam/

Q23) A DevOps Engineer has been assigned to develop an automated workflow to ensure that the required patches of all of their Windows EC2 instances are properly applied. It is of utmost importance that the EC2 instance reboots do not occur at the same time on all of their Windows instances in order to maintain their system uptime requirements. Any unavailability issues of their systems would likely cause a loss of revenue in the company since the customer transactions will not be processed in a timely manner.
How can the engineer meet the above requirements?

● Set up a Patch Group with unique tags that you will assign to all of your Amazon EC2 Windows Instances and then associate the predefined AWS-DefaultPatchBaseline baseline on your patch group. Create a new maintenance window and associate it with your patch group. Assign the AWS-RunPatchBaseline document as a task within your maintenance window.
● Set up two Patch Groups with unique tags that you will assign to all of your Amazon EC2 Windows Instances. Associate the predefined AWS-DefaultPatchBaseline baseline on both patch groups. Integrate AWS Systems Manager Run Command and CloudWatch Events to set up a cron expression that will automate the patch execution for the two Patch Groups. Use an AWS Systems Manager State Manager document to define custom commands which will be executed during patch execution.
✅ Set up two Patch Groups with unique tags that you will assign to all of your Amazon EC2 Windows Instances. Associate the predefined AWS-DefaultPatchBaseline baseline on both patch groups. Set up two non-overlapping maintenance windows and associate each with a different patch group. Register targets with specific maintenance windows using Patch Group tags. Assign the AWS-RunPatchBaseline document as a task within each maintenance window which has a different processing start time.
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 Patch Manager to apply patches for both operating systems and 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.
You can use a patch group to associate instances with a specific patch baseline. Patch groups help ensure that you are deploying the appropriate patches, based on the associated patch baseline rules, to the correct set of instances. Patch groups can also help you avoid deploying patches before they have been adequately tested. For example, you can create patch groups for different environments (such as Development, Test, and Production) and register each patch group to an appropriate patch baseline.
When you run AWS-RunPatchBaseline, you can target managed instances using their instance ID or tags. SSM Agent and Patch Manager will then evaluate which patch baseline to use based on the patch group value that you added to the instance.
You create a patch group by using Amazon EC2 tags. Unlike other tagging scenarios across Systems Manager, a patch group must be defined with the tag key: Patch Group. Note that the key is case-sensitive. You can specify any value, for example, "web servers," but the key must be Patch Group.
The AWS-DefaultPatchBaseline baseline is primarily used to approve all Windows Server operating system patches that are classified as "CriticalUpdates" or "SecurityUpdates" and that have an MSRC severity of "Critical" or "Important". Patches are auto-approved seven days after release.
Hence, the option that says: Set up two Patch Groups with unique tags that you will assign to all of your Amazon EC2 Windows Instances. Associate the predefined AWS-DefaultPatchBaseline baseline on both patch groups. Set up two non-overlapping maintenance windows and associate each with a different patch group. Register targets with specific maintenance windows using Patch Group tags. Assign the AWS-RunPatchBaseline document as a task within each maintenance window which has a different processing start time is the correct answer as it properly uses two Patch Groups, non-overlapping maintenance windows and the AWS-DefaultPatchBaseline baseline to ensure that the EC2 instance reboots do not occur at the same time.
The option that says: Set up a Patch Group with unique tags that you will assign to all of your Amazon EC2 Windows Instances and then associate the predefined AWS-DefaultPatchBaseline baseline on your patch group. Create a new maintenance window and associate it with your patch group. Assign the AWS-RunPatchBaseline document as a task within your maintenance window is incorrect because although it is correct to use a Patch Group, you must create another Patch Group to avoid any unavailability issues. Having two non-overlapping maintenance windows will ensure that there will be another set of running Windows EC2 instances while the other set is being patched.
The option that says: Set up a Patch Group with unique tags that you will assign to all of your Amazon EC2 Windows Instances. Associate the predefined AWS-DefaultPatchBaseline baseline on both patch groups. Set up a CloudWatch Events rule configured to use a cron expression to automate the execution of patching in a given schedule using the AWS Systems Manager Run command. Use an AWS Systems Manager State Manager document to define custom commands which will be executed during patch execution is incorrect because the AWS Systems Manager Run Command is primarily used to remotely manage the configuration of your managed instances while AWS Systems Manager State Manager is just a configuration management service that automates the process of keeping your Amazon EC2 and hybrid infrastructure in a state that you define. These two services, including CloudWatch Events, are not suitable for this scenario. The better solution would be to use AWS Systems Manager Maintenance Windows which lets you define a schedule for when to perform potentially disruptive actions on your instances such as patching an operating system, updating drivers, or installing software or patches.
The option that says: Set up a Patch Group with unique tags that you will assign to all of your Amazon EC2 Windows Instances. Associate the predefined AWS-DefaultPatchBaseline baseline on both patch groups. Set up a CloudWatch Events rule configured to use a cron expression to automate the execution of patching in a given schedule using the AWS Systems Manager Run command. Use an AWS Systems Manager State Manager document to define custom commands which will be executed during patch execution is incorrect because, just as what is mentioned in the previous option, you have to use Maintenance Windows for scheduling the patches and you also need to set up two Patch Groups in this scenario instead of one.
References:
https://aws.amazon.com/blogs/mt/patching-your-windows-ec2-instances-using-aws-systems-manager-patch-manager/
https://docs.aws.amazon.com/systems-manager/latest/userguide/patch-manager-ssm-documents.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-scheduletasks.html
Check out this AWS Systems Manager Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-systems-manager/

- Set up a Patch Group with unique tags that you will assign to all of your Amazon EC2 Windows Instances. Associate the predefined AWS-DefaultPatchBaseline baseline on both patch groups. Set up a CloudWatch Events rule configured to use a cron expression to automate the execution of patching in a given schedule using the AWS Systems Manager Run command. Use an AWS Systems Manager State Manager document to define custom commands which will be executed during patch execution.

Q24) You're running a cluster of EC2 instances on AWS that serves authentication processes and session handling for your new application. After three weeks of operation, your monitoring team flagged your instances with constantly high CPU usage, and the login module response is very slow. Upon checking, it was discovered that your Amazon EC2 instances were hacked and exploited for mining Bitcoins. You have immediately taken down the cluster and created a new one with your custom AMI. You want to have a solution to detect compromised EC2 instances as well as detect malicious activity within your AWS environment.
Which of the following will help you with this?

- Use AWS Inspector
- Use Amazon Macie
- Enable VPC Flow Logs
- ✓ Enable Amazon GuardDuty

Explanation:-Amazon GuardDuty offers threat detection that enables you to continuously monitor and protect your AWS accounts and workloads. GuardDuty analyzes continuous streams of meta-data generated from your account and network activity found in AWS CloudTrail Events, Amazon VPC Flow Logs, and DNS Logs. It also uses integrated threat intelligence such as known malicious IP addresses, anomaly detection, and machine learning to identify threats more accurately. This can include issues like escalations of privileges, uses of exposed credentials, or communication with malicious IPs, URLs, or domains.
For example, GuardDuty can detect compromised EC2 instances serving malware or mining bitcoin. It also monitors AWS account access behavior for signs of compromise, such as unauthorized infrastructure deployments, like instances deployed in a region that has never been used, or unusual API calls, like a password policy change to reduce password strength.
Hence, the correct answer is: Enable Amazon GuardDuty.
The option that says: Use Amazon Macie is incorrect because this service is available to protect data stored in Amazon S3 only. It uses machine learning to automatically discover, classify, and protect sensitive data in AWS.
The option that says: Use AWS Inspector is incorrect because this service is just an automated security assessment service that helps improve the security and compliance of applications deployed on AWS. It will not detect possible malicious activity on your instances.

The option that says: Enable VPC Flow Logs is incorrect because this feature only collects traffic logs flowing to your VPC. It does not provide analysis of the traffic logs.
References:
https://aws.amazon.com/guardduty/faqs/
https://docs.aws.amazon.com/guardduty/latest/ug/what-is-guardduty.html
https://aws.amazon.com/guardduty/
Check out this Amazon GuardDuty Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-guardduty/

Q25) There are several teams sharing your single AWS account that hosts your production infrastructure. Your teams are primarily storing media and images on AWS S3 buckets; some are used for public internet use, while other buckets are used for internal applications only. Since the permissions on these public access buckets are listed on AWS Trusted Advisor, you want to take advantage of it to make sure that all public buckets only allow List operations for intended users. You want to be notified when any public S3 bucket have the wrong permissions and have it automatically changed if needed.
Which of the following should you implement to meet the requirements in this scenario? (Select THREE)

Set up a custom AWS Inspector rule that checks public S3 buckets permissions. Send an action to AWS Systems Manager to correct the S3 bucket policy.
Utilize CloudWatch Events to monitor Trusted Advisor security recommendation results and then set a trigger to send an email using SNS to notify you about the results of the check.
Explanation:-AWS Config enables you to assess, audit, and evaluate the configurations of your AWS resources. Config continuously monitors and records your AWS resource configurations and allows you to automate the evaluation of recorded configurations against desired configurations. You can configure AWS Config to stream configuration changes and notifications to an Amazon SNS topic. This way, you can be notified when AWS Config evaluates your custom or managed rules against your resources.
AWS Config can monitor your Amazon Simple Storage Service (S3) bucket ACLs and policies for violations that allow public read or public write access. If AWS Config finds a policy violation, it can trigger an Amazon CloudWatch Event rule to trigger an AWS Lambda function which either corrects the S3 bucket ACL, or notifies you via Amazon Simple Notification Service (Amazon SNS) that the policy is in violation and allows public read or public write access.
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.
Hence, the correct answers are:
Set up a custom AWS Config rule that checks public S3 buckets permissions. Then, send a non-compliance notification to your subscribed SNS topic.
Set up a custom AWS Config rule to check public S3 buckets permissions and have it send an event on CloudWatch Events about policy violations. Have CloudWatch Events trigger a Lambda Function to update the S3 bucket permission.
Utilize CloudWatch Events to monitor Trusted Advisor security recommendation results and then set a trigger to send an email using SNS to notify you about the results of the check.
The option that says: Set up a custom AWS Inspector rule that checks public S3 buckets permissions. Send an action to AWS Systems Manager to correct the S3 bucket policy is incorrect because the AWS Inspector service is just an automated security assessment service that is primarily used for EC2 instances. You have to use AWS Config instead.
The option that says: Set up a custom AWS Config rule to execute a remediation action to update the permissions on the public S3 bucket is incorrect because AWS Config alone cannot run the remediation. This should be integrated with the AWS Systems Manager Automation service where you can configure the actions for your remediation.
The option that says: Create a Lambda function that executes every hour to refresh AWS Trusted Advisor scan results via API. Subscribe to AWS Trusted Advisor notification messages to receive the results is incorrect because it is better to use AWS Config instead of AWS Trusted Advisor in this scenario. Moreover, the Trusted Advisor only sends the the summary notification every week so this won't notify you immediately about your non-compliant resources.
References:
https://aws.amazon.com/blogs/security/how-to-use-aws-config-to-monitor-for-and-respond-to-amazon-s3-buckets-allowing-public-access/
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 Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-aws-config/
https://tutorialsdojo.com/aws-cheat-sheet-aws-trusted-advisor/
https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudwatch/
Set up a custom AWS Config rule to execute a remediation action to update the permissions on the public S3 bucket.
Set up a custom AWS Config rule to check public S3 buckets permissions and have it send an event on CloudWatch Events about policy violations. Have CloudWatch Events trigger a Lambda Function to update the S3 bucket permission.
Explanation:-AWS Config enables you to assess, audit, and evaluate the configurations of your AWS resources. Config continuously monitors and records your AWS resource configurations and allows you to automate the evaluation of recorded configurations against desired configurations. You can configure AWS Config to stream configuration changes and notifications to an Amazon SNS topic. This way, you can be notified when AWS Config evaluates your custom or managed rules against your resources.
AWS Config can monitor your Amazon Simple Storage Service (S3) bucket ACLs and policies for violations that allow public read or public write access. If AWS Config finds a policy violation, it can trigger an Amazon CloudWatch Event rule to trigger an AWS Lambda function which either corrects the S3 bucket ACL, or notifies you via Amazon Simple Notification Service (Amazon SNS) that the policy is in violation and allows public read or public write access.
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.
Hence, the correct answers are:
Set up a custom AWS Config rule that checks public S3 buckets permissions. Then, send a non-compliance notification to your subscribed SNS topic.
Set up a custom AWS Config rule to check public S3 buckets permissions and have it send an event on CloudWatch Events about policy violations. Have CloudWatch Events trigger a Lambda Function to update the S3 bucket permission.
Utilize CloudWatch Events to monitor Trusted Advisor security recommendation results and then set a trigger to send an email using SNS to notify you about the results of the check.
The option that says: Set up a custom AWS Inspector rule that checks public S3 buckets permissions. Send an action to AWS Systems Manager to correct the S3 bucket policy is incorrect because the AWS Inspector service is just an automated security assessment service that is primarily used for EC2 instances. You have to use AWS Config instead.

The option that says: Set up a custom AWS Config rule to execute a remediation action to update the permissions on the public S3 bucket is incorrect because AWS Config alone cannot run the remediation. This should be integrated with the AWS Systems Manager Automation service where you can configure the actions for your remediation.
The option that says: Create a Lambda function that executes every hour to refresh AWS Trusted Advisor scan results via API. Subscribe to AWS Trusted Advisor notification messages to receive the results is incorrect because it is better to use AWS Config instead of AWS Trusted Advisor in this scenario. Moreover, the Trusted Advisor only sends the the summary notification every week so this won't notify you immediately about your non-compliant resources.
References:
https://aws.amazon.com/blogs/security/how-to-use-aws-config-to-monitor-for-and-respond-to-amazon-s3-buckets-allowing-public-access/
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 Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-aws-config/
https://tutorialsdojo.com/aws-cheat-sheet-aws-trusted-advisor/
https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudwatch/

○ Create a Lambda function that executes every hour to refresh AWS Trusted Advisor scan results via API. Subscribe to AWS Trusted advisor notification messages to receive the results.

✓ Set up a custom AWS Config rule that checks public S3 buckets permissions. Then, send a non-compliance notification to your subscribed SNS topic.

Explanation:-AWS Config enables you to assess, audit, and evaluate the configurations of your AWS resources. Config continuously monitors and records your AWS resource configurations and allows you to automate the evaluation of recorded configurations against desired configurations. You can configure AWS Config to stream configuration changes and notifications to an Amazon SNS topic. This way, you can be notified when AWS Config evaluates your custom or managed rules against your resources.
AWS Config can monitor your Amazon Simple Storage Service (S3) bucket ACLs and policies for violations that allow public read or public write access. If AWS Config finds a policy violation, it can trigger an Amazon CloudWatch Event rule to trigger an AWS Lambda function which either corrects the S3 bucket ACL, or notifies you via Amazon Simple Notification Service (Amazon SNS) that the policy is in violation and allows public read or public write access.
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.
Hence, the correct answers are:
Set up a custom AWS Config rule that checks public S3 buckets permissions. Then, send a non-compliance notification to your subscribed SNS topic.
Set up a custom AWS Config rule to check public S3 buckets permissions and have it send an event on CloudWatch Events about policy violations. Have CloudWatch Events trigger a Lambda Function to update the S3 bucket permission.
Utilize CloudWatch Events to monitor Trusted Advisor security recommendation results and then set a trigger to send an email using SNS to notify you about the results of the check.
The option that says: Set up a custom AWS Inspector rule that checks public S3 buckets permissions. Send an action to AWS Systems Manager to correct the S3 bucket policy is incorrect because the AWS Inspector service is just an automated security assessment service that is primarily used for EC2 instances. You have to use AWS Config instead.
The option that says: Set up a custom AWS Config rule to execute a remediation action to update the permissions on the public S3 bucket is incorrect because AWS Config alone cannot run the remediation. This should be integrated with the AWS Systems Manager Automation service where you can configure the actions for your remediation.
The option that says: Create a Lambda function that executes every hour to refresh AWS Trusted Advisor scan results via API. Subscribe to AWS Trusted Advisor notification messages to receive the results is incorrect because it is better to use AWS Config instead of AWS Trusted Advisor in this scenario. Moreover, the Trusted Advisor only sends the the summary notification every week so this won't notify you immediately about your non-compliant resources.
References:
https://aws.amazon.com/blogs/security/how-to-use-aws-config-to-monitor-for-and-respond-to-amazon-s3-buckets-allowing-public-access/
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 Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-aws-config/
https://tutorialsdojo.com/aws-cheat-sheet-aws-trusted-advisor/
https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudwatch/

Q26) You have an Amazon S3 bucket named tutorialsdojo-devops that is shared by all the DevOps admin from your team. It is used to store artifact files for your several CI/CD pipelines. One of your junior developers accidentally changed the S3 bucket policy that denied all pipelines from downloading the needed artifact files, causing all deployments to halt. In the future, you want to be notified of any S3 bucket policy changes so that you can quickly identify and take action if any problem occurs. Which of the following options will help you achieve this?

○ Enable S3 server access logging on your bucket. Create a CloudWatch Metric Filter for bucket policy events. Create an Alarm for this metric to notify you whenever an event is matched.

○ Create a CloudTrail trail that sends logs to a CloudWatch Log group. Create a CloudWatch Metric Filter for S3 bucket policy events on the log group. Create an Alarm that will send you a notification whenever this metric threshold is reached.

○ Enable S3 server access logging on your bucket. Send the access logs to a CloudWatch log group. Create a CloudWatch Metric Filter for bucket policy events on the log group. Create an Alarm for this metric to notify you whenever the threshold is reached.

✓ Create a CloudTrail trail that sends logs to a CloudWatch Log group. Create a CloudWatch Event rule for S3 bucket policy events on the log group. Create an Alarm based on the event that will send you a notification whenever an event is matched.

Explanation:-You can configure alarms for several scenarios in CloudTrail events. In this case, you can create an Amazon CloudWatch alarm that is triggered when an Amazon S3 API call is made to PUT or DELETE bucket policy, bucket lifecycle, bucket replication, or to PUT a bucket ACL. A CloudTrail trail is required since it will send its logs to a CloudWatch Log group. To create an alarm, you must first create a metric filter and then configure an alarm based on the filter.
Since all CloudTrail events will be sent to the Log group, you will need to create a Metric to filter specific S3 events that change the bucket policy. For notification, you will then create a CloudWatch Alarm for this Metric with a threshold of >=1 and set your email as a notification recipient. Even a single S3 bucket event on the log group will trigger this alarm and should send you a notification.
Hence, the correct answer is: Create a CloudTrail trail that sends logs to a CloudWatch Log group. Create a CloudWatch Metric Filter for S3 bucket policy events on the log group. Create an Alarm that will send you a notification whenever this metric threshold is reached.
The option that says: Enable S3 server access logging on your bucket. Create a CloudWatch Metric Filter for bucket policy events. Create an Alarm

for this metric to notify you whenever an event is matched is incorrect because S3 Server access logging is primarily used to provide detailed records for the requests that are made to a bucket. Each access log record provides details about a single access request, such as the requester, bucket name, request time, request action, response status, and an error code, if relevant. It is more appropriate to use CloudWatch or CloudTrail to track the S3 bucket policy changes.
The option that says: Enable S3 server access logging on your bucket. Send the access logs to a CloudWatch log group. Create a CloudWatch Metric Filter for bucket policy events on the log group. Create an Alarm for this metric to notify you whenever the threshold is reached is incorrect because you can't directly send the S3 server access logs to CloudWatch logs. You need to use CloudTrail to send the events to a log group before you can create a metric and alarm for those events.
The option that says: Create a CloudTrail trail that sends logs to a CloudWatch Log group. Create a CloudWatch Event rule for S3 bucket policy events on the log group. Create an Alarm based on the event that will send you a notification whenever an event is matched is incorrect because you can't use CloudWatch Events to filter your log groups directly.
References:
https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html
https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html#cloudwatch-alarms-for-cloudtrail-s3-bucket-activity
https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/ConsoleAlarms.html
Check out this AWS CloudTrail Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-cloudtrail/

Q27) You have an ECS Fargate cluster that runs a batch job when data files are loaded on an Amazon S3 bucket. To save costs, you want to set the minimum number of ECS Tasks to 1, and only increase the task count when objects are uploaded on the S3 bucket once again. Once processing is done, the S3 bucket is now empty and you want to set the ECS Task count back to 1. The object-level logging has already been enabled in the bucket.
Which of the following options is the EASIEST way to implement this?

- Use CloudWatch Alarms on CloudTrail since the S3 object-level operations are recorded on CloudTrail. Create two Lambda functions for increasing/decreasing the ECS task count. Set these as respective targets for the CloudWatch Alarm depending on the S3 event.
- Create a CloudWatch Event rule to detect S3 object PUT operations and set the target to a Lambda function that will run Amazon ECS API command to increase the number of tasks on ECS. Create another rule to detect S3 DELETE operations and run the Lambda function to reduce the number of ECS tasks.
- Use CloudWatch Alarms on CloudTrail since this S3 object-level operations are recorded on CloudTrail. Set two alarm actions to update ECS task count to scale-out/scale-in depending on the S3 event.
- ✅ Create a CloudWatch Event rule to detect S3 object PUT operations and set the target to the ECS cluster with the increased number of tasks. Create another rule to detect S3 DELETE operations and set the target to the ECS Cluster with 1 as the Task count.

Explanation:-You can use CloudWatch Events to run Amazon ECS tasks when certain AWS events occur. You can set up a CloudWatch Events rule that runs an Amazon ECS task whenever a file is uploaded to a certain Amazon S3 bucket using the Amazon S3 PUT operation. You can also declare a reduced number of ECS tasks whenever a file is deleted on the S3 bucket using the DELETE operation.
First, you must create a CloudWatch Events rule for the S3 service that will watch for object-level operations – PUT and DELETE objects. For object-level operations, it is required to create a CloudTrail trail first. On the Targets section, select the "ECS task" and input the needed values such as the cluster name, task definition and the task count. You need two rules – one for the scale-up and another for the scale-down of the ECS task count.
Hence, the correct answer is: Create a CloudWatch Event rule to detect S3 object PUT operations and set the target to the ECS cluster with the increased number of tasks. Create another rule to detect S3 DELETE operations and set the target to the ECS Cluster with 1 as the Task count.
The option that says: Create a CloudWatch Event rule to detect S3 object PUT operations and set the target to a Lambda function that will run Amazon ECS API command to increase the number of tasks on ECS. Create another rule to detect S3 DELETE operations and run the Lambda function to reduce the number of ECS tasks is incorrect because although this solution meets the requirement, creating your own Lambda function for this scenario is not really necessary. It is much simpler to control ECS task directly as target for the CloudWatch Event rule. Take note that the scenario asks for a solution that is the easiest to implement.
The option that says: Use CloudWatch Alarms on CloudTrail since the S3 object-level operations are recorded on CloudTrail. Create two Lambda functions for increasing/decreasing the ECS task count. Set these as respective targets for the CloudWatch Alarm depending on the S3 event is incorrect because using CloudTrail, CloudWatch Alarm, and two Lambda functions creates an unnecessary complexity to what you want to achieve. CloudWatch Events can directly target an ECS task on the Targets section when you create a new rule.
The option that says: Use CloudWatch Alarms on CloudTrail since this S3 object-level operations are recorded on CloudTrail. Set two alarm actions to update ECS task count to scale-out/scale-in depending on the S3 event is incorrect because you can't directly set CloudWatch Alarms to update the ECS task count. You have to use CloudWatch Events instead.
References:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/CloudWatch-Events-tutorial-ECS.html
https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/Create-CloudWatch-Events-Rule.html
Check out this Amazon CloudWatch Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudwatch/

Q28) A technology consulting company has an Oracle Real Application Clusters (RAC) database on their on-premises network which they want to migrate to AWS Cloud. Their Chief Technology Officer instructed the DevOps team to automate the patch management process of the operating system in which their database will run. They are also mandated as well to set up scheduled backups to comply with the disaster recovery plan of the company.
What should the DevOps team do to satisfy the requirement for this scenario with the LEAST amount of effort?

- Migrate the database that is hosted on-premises to Amazon RDS which provides a multi-AZ failover feature for your Oracle RAC cluster. The RPO and RTO will be reduced in the event of system failure since Amazon RDS offers features such as patch management and maintenance of the underlying host.
- Migrate the on-premises database to Amazon Aurora. Enable automated backups for your Aurora RAC cluster. With Amazon Aurora, patching is automatically handled during the system maintenance window.
- Migrate the Oracle RAC database to a large EBS-backed Amazon EC2 instance. Launch an AWS Lambda function that will call the CreateSnapshot EC2 API to automate the creation of Amazon EBS snapshots of the database. Integrate CloudWatch Events and Lambda in order to run the function on a regular basis. Set up the CodeDeploy and CodePipeline services to automate the patch management process of the database.
- ✅ Migrate the Oracle RAC database to a large EBS-backed Amazon EC2 instance then install the SSM agent. Use the AWS Systems Manager Patch Manager to automate the patch management process. Set up the Amazon Data Lifecycle Manager service to automate the creation of Amazon EBS snapshots from the EBS volumes of the EC2 instance.

Explanation:-AWS Systems Manager Patch Manager automates the process of patching managed instances with security-related updates. For Linux-based instances, you can also install patches for non-security updates. 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.
Amazon Data Lifecycle Manager (DLM) for EBS Snapshots provides a simple, automated way to back up data stored on Amazon EBS volumes. You can define backup and retention schedules for EBS snapshots by creating lifecycle policies based on tags. With this feature, you no longer have to rely on custom scripts to create and manage your backups.
Oracle RAC is supported via the deployment using Amazon EC2 only since Amazon RDS and Aurora do not support it. Amazon RDS does not support certain features in Oracle such as Multitenant Database, Real Application Clusters (RAC), Unified Auditing, Database Vault and many more.
You can use AWS Systems Manager Patch Manager to automate the process of patching managed instances with security-related updates.
Hence, the correct answer is: Migrate the Oracle RAC database to a large EBS-backed Amazon EC2 instance then install the SSM agent. Use the AWS Systems Manager Patch Manager to automate the patch management process. Set up the Amazon Data Lifecycle Manager service to automate the creation of Amazon EBS snapshots from the EBS volumes of the EC2 instance.
The option that says: Migrate the database that is hosted on-premises to Amazon RDS which provides a multi-AZ failover feature for your Oracle RAC cluster. The RPO and RTO will be reduced in the event of system failure since Amazon RDS offers features such as patch management and maintenance of the underlying host is incorrect because Amazon RDS doesn't support Oracle RAC.
The option that says: Migrate the on-premises database to Amazon Aurora. Enable automated backups for your Aurora RAC cluster. With Amazon Aurora, patching is automatically handled during the system maintenance window is incorrect because, just like Amazon RDS, the Amazon Aurora doesn't support Oracle RAC as well.
The option that says: Migrate the Oracle RAC database to a large EBS-backed Amazon EC2 instance. Launch an AWS Lambda function that will call the CreateSnapshot EC2 API to automate the creation of Amazon EBS snapshots of the database. Integrate CloudWatch Events and Lambda in order to run the function on a regular basis. Set up the CodeDeploy and CodePipeline services to automate the patch management process of the database is incorrect because CodeDeploy and CodePipeline are CI/CD services and are not suitable for patch management. You should use AWS Systems Manager Patch Manager instead. In addition, the Amazon Data Lifecycle Manager service is the recommended way to automate the creation of Amazon EBS snapshots and not a combination of Lambda and CloudWatch Events.
References:
https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-patch.html
https://aws.amazon.com/rds/oracle/faqs/
Check out this Amazon RDS Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-relational-database-service-amazon-rds/

Q29) You have created a new Elastic Beanstalk environment to be used as a pre-production stage for load testing new code version. Since code changes are committed on a regular basis, you sometimes need to deploy new versions 2 to 3 times each day. You need to deploy a new version as quickly as possible in a cost-effective way to give ample time for the QA team to test it.
Which of the following implementations is suited for this scenario?

- Implement a blue/green deployment strategy to have the new version ready for quick switching.
- Use Rolling as the deployment policy to deploy new versions.
- ✅ Use All at once as deployment policy to deploy new versions.

Explanation:-In ElasticBeanstalk, you can choose from a variety of deployment methods:
All at once – Deploy the new version to all instances simultaneously. All instances in your environment are out of service for a short time while the deployment occurs. This is the method that provides the least amount of time for deployment.
Rolling – Deploy the new version in batches. Each batch is taken out of service during the deployment phase, reducing your environment's capacity by the number of instances in a batch.
Rolling with additional batch – Deploy the new version in batches, but first launch a new batch of instances to ensure full capacity during the deployment process.
Immutable – Deploy the new version to a fresh group of instances by performing an immutable update.
Blue/Green - Deploy the new version to a separate environment, and then swap CNAMEs of the two environments to redirect traffic to the new version instantly.
Refer to the table below for the characteristics of each deployment method as well as the amount of time it takes to do the deployment, as seen in the Deploy Time column:
Hence, the correct answer is: Use All at once as the deployment policy to deploy new versions.
The option that says: Use Rolling as the deployment policy to deploy new versions is incorrect because this deployment type is not as fast as “All at once” deployment since the code is deployed in batches.
The option that says: Use Immutable as the deployment policy to deploy code on new instances is incorrect because this deployment type takes time as new instances are being provisioned during the deployment.
The option that says: Implement a blue/green deployment strategy to have the new version ready for quick switching is incorrect because just like the Immutable deployment, this type also takes time since the new instances are provisioned first before the actual deployment starts. Plus, it incurs additional cost as 2 sets of instances are running at the same time.
References:
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.rolling-version-deploy.html
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.CNAMESwap.html
Check out this AWS Elastic Beanstalk Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-elastic-beanstalk/

- Use Immutable as the deployment policy to deploy code on new instances.

Q30) An international IT consulting firm has multiple on-premises data centers across the globe. Their technical team regularly uploads financial and regulatory files from each of their respective data centers to a centralized web portal hosted in AWS. It uses an Amazon S3 bucket named financial-tdojo-reports to store the data. Another team downloads various reports from a CloudFront web distribution that uses the same Amazon S3 bucket as the origin. A DevOps Engineer noticed that the staff are using both the CloudFront link and the direct Amazon S3 URLs to download the reports. The IT Security team of the company considered this as a security risk, and they recommended to re-design the architecture. A new system must be implemented that prevents anyone from bypassing the CloudFront distribution and disable direct access from Amazon S3 URLs.
What should the Engineer do to meet the above requirement?

- In the CloudFront web distribution, set up a field-level encryption configuration and for each user, revoke the existing permission to access Amazon S3 URLs to download the objects.
- Configure the distribution to use Signed URLs and create a special CloudFront user called an origin access identity (OAI). Grant permission to the OAI to read the objects in the S3 bucket.
- ✅ Set up a special CloudFront user called an origin access identity (OAI). Grant the origin access identity permission to read the objects in your

bucket. For each user, revoke the existing permission to access Amazon S3 URLs to download the objects.

Explanation:-To restrict access to content that you serve from Amazon S3 buckets, you create CloudFront signed URLs or signed cookies to limit access to files in your Amazon S3 bucket, and then you create a special CloudFront user called an origin access identity (OAI) and associate it with your distribution. Then you configure permissions so that CloudFront can use the OAI to access and serve files to your users, but users can't use a direct URL to the S3 bucket to access a file there.

In this scenario, the main objective is to prevent the staff from using the direct Amazon S3 URLs to download the reports. The best solution that you can choose here is to use an Origin Access Identity (OAI) and remove anyone else's permission to use the S3 URLs to read the objects.

Hence, the correct answer is: Set up a special CloudFront user called an origin access identity (OAI). Grant the origin access identity permission to read the objects in your bucket. For each user, revoke the existing permission to access Amazon S3 URLs to download the objects.

The option that says: Set up a custom SSL in your CloudFront web distribution instead of the default SSL. For each user, revoke the existing permission to access Amazon S3 URLs to download the objects incorrect because SSL is not needed in this particular scenario. What you need to implement is an OAI.

The option that says: In the CloudFront web distribution, set up a field-level encryption configuration and for each user, revoke the existing permission to access Amazon S3 URLs to download the objects is incorrect because the field-level encryption configuration is primarily used for safeguarding sensitive fields in your CloudFront. Therefore, it is not suitable for this scenario.

The option that says: Configure the distribution to use Signed URLs and create a special CloudFront user called an origin access identity (OAI). Grant permission to the OAI to read the objects in the S3 bucket is incorrect because although it is recommended to use Signed URLs and OAI in CloudFront, this option is still missing the crucial step of removing the user permissions to directly use the S3 URLs in order to download the files.

References:

https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/PrivateContent.html

https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-overview.html

https://aws.amazon.com/blogs/networking-and-content-delivery/amazon-s3-amazon-cloudfront-a-match-made-in-the-cloud/

Check out this Amazon CloudFront Cheat Sheet:

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

● Set up a custom SSL in your CloudFront web distribution instead of the default SSL. For each user, revoke the existing permission to access Amazon S3 URLs to download the objects.

Q31) A company is planning to launch a mobile marketplace using AWS Amplify and AWS Mobile Hub which will serve millions of users worldwide. The backend APIs will be launched to multiple AWS regions to process the sales and financial transactions on the region closest to the users to lower the latency. A DevOps Engineer was instructed to design the system architecture to ensure that the transactions made in one region are automatically replicated to other regions. In the coming months ahead, it is expected that the marketplace will have millions of users across North America, South America, Europe, and Asia.
Which of the following is the MOST scalable, cost-effective and highly available architecture that the Engineer should implement?

● Set up an Aurora Multi-Master database on all of the required AWS regions. Store the individual transactions to the Amazon Aurora instance in the local region and replicate the transactions across regions using Aurora replication. Any changes made in one of the tables will be automatically replicated across all other tables.

● Store the individual transactions in each local region to a DynamoDB table. Use a Lambda function to read recent writes from the primary DynamoDB table and replay the data to tables in all other regions.

● Set up a Global DynamoDB table in your preferred region which will automatically create new replica tables on all AWS regions. Store the individual transactions to a DynamoDB replica table in each local region. Any changes made in one of the replica tables will be automatically replicated across all other tables worldwide.

✅ In your preferred AWS region, set up a Global DynamoDB table and then enable the DynamoDB Streams option. Set up replica tables in the other AWS regions where you want to replicate your data. In each local region, store the individual transactions to a DynamoDB replica table in the same region.

Explanation:-Amazon DynamoDB is a key-value and document database that delivers single-digit millisecond performance at any scale. It's a fully managed, multiregion, multimaster, durable database with built-in security, backup and restore, and in-memory caching for internet-scale applications. DynamoDB can handle more than 10 trillion requests per day and can support peaks of more than 20 million requests per second.

Global Tables builds upon DynamoDB's global footprint to provide you with a fully managed, multi-region, and multi-master database that provides fast, local, read and write performance for massively scaled, global applications. Global Tables replicates your Amazon DynamoDB tables automatically across your choice of AWS regions.

Global Tables eliminates the difficult work of replicating data between regions and resolving update conflicts, enabling you to focus on your application's business logic. In addition, Global Tables enables your applications to stay highly available even in the unlikely event of isolation or degradation of an entire region.

For example, suppose that you have a large customer base spread across three geographic areas—the US East Coast, the US West Coast, and Western Europe. Customers can update their profile information using your application. To satisfy this use case, you need to create three identical DynamoDB tables named CustomerProfiles, in three different AWS Regions where the customers are located. These three tables would be entirely separate from each other. Changes to the data in one table would not be reflected in the other tables. Without a managed replication solution, you could write code to replicate data changes among these tables. However, doing this would be a time-consuming and labor-intensive effort.

Instead of writing your own code, you could create a global table consisting of your three Region-specific CustomerProfiles tables. DynamoDB would then automatically replicate data changes among those tables so that changes to CustomerProfiles data in one Region would seamlessly propagate to the other Regions. In addition, if one of the AWS Regions were to become temporarily unavailable, your customers could still access the same CustomerProfiles data in the other Regions.

DynamoDB global tables are ideal for massively scaled applications with globally dispersed users. In such an environment, users expect very fast application performance. Global tables provide automatic multi-master replication to AWS Regions worldwide. They enable you to deliver low-latency data access to your users no matter where they are located.

Hence, the correct answer is: In your preferred AWS region, set up a Global DynamoDB table and then enable the DynamoDB Streams option. Set up replica tables in the other AWS regions where you want to replicate your data. In each local region, store the individual transactions to a DynamoDB replica table in the same region.

The option that says: Store the individual transactions in each local region to a DynamoDB table. Use a Lambda function to read recent writes from the primary DynamoDB table and replay the data to tables in all other regions is incorrect because using an AWS Lambda function to replicate all data across regions is not a scalable solution. Remember that there will be millions of people who will use the mobile app around the world, and this entails a lot of replication and compute capacity for a single Lambda function. In this scenario, the best way is to use Global DynamoDB tables with DynamoDB Stream option enabled, to automatically handle the replication process.

The option that says: Set up a Global DynamoDB table in your preferred region which will automatically create new replica tables on all AWS regions. Store the individual transactions to a DynamoDB replica table in each local region. Any changes made in one of the replica tables will be automatically replicated across all other tables worldwide is incorrect because even though the option correctly uses Global DynamoDB Tables, it will not automatically create new replica tables on all AWS regions. You have to manually specify and create the replica tables in the specific AWS regions where you want to replicate your data. Take note as well that the DynamoDB Stream option must be enabled in order for the Global

DynamoDB Table to work.
The option that says: Set up an Aurora Multi-Master database on all of the required AWS regions. Store the individual transactions to the Amazon Aurora instance in the local region and replicate the transactions across regions using Aurora replication. Any changes made in one of the tables will be automatically replicated across all other tables is incorrect because although using Multi-Master Amazon Aurora across multiple regions may work, this architecture is not the most cost-effective. In addition, DynamoDB provides better global scalability for mobile applications compared with Amazon Aurora.
References:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/globaltables_HowItWorks.html
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GlobalTables.html
https://aws.amazon.com/dynamodb/global-tables/
Check out this Amazon DynamoDB Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-dynamodb/

Q32) A multinational company is using multiple AWS accounts for its global cloud architecture. The AWS resources in their production account are shared among various business units of the company. A single business unit may have one or more AWS accounts that have resources in the production account. Recently, there were a lot of incidents in which the developers from a specific business unit accidentally terminated the Amazon EC2 instances owned by another business unit. A DevOps Engineer was tasked to come up with a solution to only allow a specific business unit who owns the EC2 instances and other AWS resources to terminate their own resources.
How should the Engineer implement a multi-account strategy to satisfy this requirement?

- Centrally manage all of your accounts using AWS Service Catalog. Group your accounts, which belong to a specific business unit, to the individual Organization Unit (OU). Set up a Service Control Policy in the production account which has a policy that allows access to the EC2 instances including resource-level permission to terminate the instances owned by a particular business unit. Associate the cross-account access and the SCP to the OUs, which will then be automatically inherited by its member accounts.
- Centrally manage all of your accounts using a multi-account aggregator in AWS Config. Configure AWS Config to allow access to certain Amazon EC2 instances in production per business unit.
- Centrally manage all of your accounts using AWS Organizations. Group your accounts, which belong to a specific business unit, to the individual Organization Unit (OU). Set up an IAM Role in the production account for each business unit which has a policy that allows access to the Amazon EC2 instances including resource-level permission to terminate the instances that it owns. Use an AWSServiceRoleForOrganizations service-linked role to the individual member accounts of the OU to enable trusted access.
- ✅ Centrally manage all of your accounts using AWS Organizations. Group your accounts, which belong to a specific business unit, to individual Organization Units (OU). Set up an IAM Role in the production account which has a policy that allows access to the EC2 instances including resource-level permission to terminate the instances owned by a particular business unit. Associate the cross-account access and the IAM policy to every member accounts of the OU.

Explanation:-AWS Organizations is an account management service that enables you to consolidate multiple AWS accounts into an organization that you create and centrally manage. AWS Organizations includes account management and consolidated billing capabilities that enable you to better meet the budgetary, security, and compliance needs of your business. As an administrator of an organization, you can create accounts in your organization and invite existing accounts to join the organization.
You can use organizational units (OUs) to group accounts together to administer as a single unit. This greatly simplifies the management of your accounts. For example, you can attach a policy-based control to an OU, and all accounts within the OU automatically inherit the policy. You can create multiple OUs within a single organization, and you can create OUs within other OUs. Each OU can contain multiple accounts, and you can move accounts from one OU to another. However, OU names must be unique within a parent OU or root.
Resource-level permissions refer to the ability to specify which resources users are allowed to perform actions on. Amazon EC2 has partial support for resource-level permissions. This means that for certain Amazon EC2 actions, you can control when users are allowed to use those actions based on conditions that have to be fulfilled, or specific resources that users are allowed to use. For example, you can grant users permissions to launch instances, but only of a specific type, and only using a specific AMI.
Hence, the correct answer is: Centrally manage all of your accounts using AWS Organizations. Group your accounts, which belong to a specific business unit, to individual Organization Units (OU). Set up an IAM Role in the production account which has a policy that allows access to the EC2 instances including resource-level permission to terminate the instances owned by a particular business unit. Associate the cross-account access and the IAM policy to every member accounts of the OU.
The option that says: Centrally manage all of your accounts using AWS Organizations. Group your accounts, which belong to a specific business unit, to the individual Organization Unit (OU). Set up an IAM Role in the production account for each business unit which has a policy that allows access to the Amazon EC2 instances including resource-level permission to terminate the instances that it owns. Use an AWSServiceRoleForOrganizations service-linked role to the individual member accounts of the OU to enable trusted access is incorrect because the AWSServiceRoleForOrganizations service-linked role is primarily used to only allow AWS Organizations to create service-linked roles for other AWS services. This service-linked role is present in all organizations and not just in a specific OU.
The option that says: Centrally manage all of your accounts using a multi-account aggregator in AWS Config. Configure AWS Config to allow access to certain Amazon EC2 instances in production per business unit is incorrect because an aggregator is simply an AWS Config resource type that collects AWS Config configuration and compliance data from the following various AWS accounts. You have to use AWS Organizations instead to consolidate all of the AWS accounts that the company owns.
The option that says: Centrally manage all of your accounts using AWS Service Catalog. Group your accounts, which belong to a specific business unit, to the individual Organization Unit (OU). Set up a Service Control Policy in the production account which has a policy that allows access to the EC2 instances including resource-level permission to terminate the instances owned by a particular business unit. Associate the cross-account access and the SCP to the OUs, which will then be automatically inherited by its member accounts is incorrect because AWS Service Catalog simply allows organizations to create and manage catalogs of IT services that are approved for use on AWS. A more suitable service to use here is AWS Organizations.
References:
https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_ous.html
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-supported-iam-actions-resources.html
https://docs.aws.amazon.com/IAM/latest/UserGuide/tutorial_cross-account-with-roles.html
Check out this AWS Organizations Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-organizations/
Service Control Policies (SCP) vs IAM Policies:
https://tutorialsdojo.com/aws-cheat-sheet-service-control-policies-scp-vs-iam-policies/
Comparison of AWS Services Cheat Sheets:
https://tutorialsdojo.com/comparison-of-aws-services-for-udemy-students/
