Answer Sheet

Q1)

If you?re trying to configure an AWS Elastic Beanstalk worker tier for easy debugging if there are problems finishing queue jobs.

What should you configure?

- Configure Enhanced Health Reporting
- Configure Rolling Deployments.
- ✅ Configure a Dead Letter Queue
- Configure Blue-Green Deployments.

Q2)

You are building a mobile app for consumers to post cat pictures online. You will be storing the images in AWS S3. You want to run the system very cheaply and simply.

Which one of these options allows you to build a photo sharing application with the right authentication/authorization implementation?

- Use AWS API Gateway with a constantly rotating API Key to allow access from the clientside. Construct a custom build of the SDK and include S3 access in it.
- ✅ Build the application out using AWS Cognate and web Identity federation to allow users to log In using Face book or Google Accounts. Once they are logged in, the secret token passed to that user is used to directly access resources on AWS. like AWS 53
- Use JWT or SAML compliant systems to build authorization policies. Users log In with a username and password, and are given a token they can use indefinitely to make calls against the photo infrastructure.
- Create an AWS Auth Service Domain ad grant public signup and access to the domain. During setup, add at least one major social media site as a trusted Identity Provider for users.

Q3)

You have a development team that is continuously spending a lot of time rolling back updates for an application. They work on changes, and If the change fails, they spend more than 5-6h in rolling back the update.

Which of the below options can help reduce the time for rolling back application versions?

- Use Ops Works and redeploy using rollback feature.
- Use S3 to store each version and then re-deploy with Elastic Beanstalk
- ✅ Use Elastic Beanstalk and re-deploy using Application Versions
- Use Cloud Formation and update the stack with the previous template

---

Q4)

You run accounting software in the AWS cloud. This software needs to be online continuously during the day every day of the week, and has a very static requirement for compute resources.

You also have other. unrelated batch jobs that need to run once per day at any time of your choosing.

How should you minimize cost?

- Purchase a Medium Utilization Reserved Instance to run the accounting software. Turn it off after hours. Run the batch jobs with the same instance class, so the Reserved Instance credits are also applied to the batch
- ✅ Purchase a Heavy Utilization Reserved instance to run the accounting software. Turn it off after hours. Run the batch jobs with the same instance class, so the Reserved instance credits are also applied to the batch jobs.
- Purchase a Light Utilization Reserved Instance to run the accounting software. Turn it off after hours. Run the batch jobs with the same instance class, so the Reserved Instance credits are also applied to the batch jobs.
- Purchase a Full Utilization Reserved Instance to run the accounting software. Turn it off after hours. Run the batch jobs with the same instance class, so the Reserved Instance credits are also applied to the batch jobs.

Q5)

You want to pass queue messages that are 1 GB each.

How should you achieve this? Please select:

- Use AWS EFS as a shared pool storage medium. Store file system pointers to the files on disk in the SQS message bodies.
- Use Kinesis as a buffer stream for message bodies. Store the checkpoint id for the placement in the Kinesis Stream in SQS.
- Use SQS's support for message partitioning and multi-part uploads on Amazon S3.
- ✅ Use the Amazon SQS Extended Client Ubrary forJava and Amazon S3 as a storage mechanism for message bodies.

Q6)

Your server less architecture using AWS API Gateway, AWS Lambda, and AWS Dynamo DB experienced a large increase in traffic to a sustained 2000 requests per second, and dramatically increased in failure rates. Your requests, during normal operation, last 500 milliseconds on average.

Your Dynamo DB table did not exceed 50% of provisioned throughput, and Table primary keys are designed correctly.

What is the most likely issue?

- You used Consistent Read requests on Dynamo DB and are experiencing semaphore lock.
- Your API Gateway deployment Is throttling your requests.
- ✅ You did not request a limit increase on concurrent Lambda function executions.
- Your AWS API Gateway Deployment is bottlenecking on request .

Q7)

You are a Dev ops engineer for your company. You have been instructed to deploy ducker containers using the Ops work service.

How could you achieve this? Choose 2 answers from the options given below Please select:

- Use Cloud formation to deploy ducker containers since this is not possible in Opswork. Then attach the Cloud formation resources as a layer in Ops work.
- ✅ Use custom cookbooks for your Ops work stack and provide the Git repository which has the chef recipes for the Ducker containers.
- ✅ In the App for Opswork deployment. specify the git un for the recipes which will deploy the applications in the docker environment. ...-
- Use Elastic beanstalk to deploy ducker containers since this is not possible In Ops work. Then attach the elastic beanstalk environment as a layer in Ops work.

Q8) Which of the following are ways to ensure that data is secured while in transit when using the AWS Elastic load balancer. Choose 2 answers from the options given below Please select:

- Use an HTTP front end listener for your ELB
- Use a TCP front end listener for your ELS
- ✅ Use an HTTPS front end listener for your ELB
- ✅ Use an SSL front end listener for your ELB

Q9)

Your company is concerned with EBS volume backup on Amazon EC2 and wants to ensure they have proper backups and that the data Is durable.

What solution would you implement and why? Choose the correct answer from the options below?

- Write a crone job on the server that compresses the data that needs to be backed up using gzip compressi then use AWS CLI to copy the data into an S3 bucket for durability
- Configure Amazon Storage Gateway with EBS volumes as the data source and store the backups on premise through the storage gateway
- ✅ Write a crone job that uses the AWS CLI to take a snapshot of production EBS volumes. The data Is durable because EBS snapshots are stored on the Amazon 53 standard storage class
- Use a lifecycle policy to back up EBS volumes stored on Amazon 53 for durability

---

Q10)

One of your instances is reporting an unhealthy system status check. However, this is not something you should have to monitor and repair on your own.

How might you automate the repair of the system status check failure in an AWS environment? Choose the correct answer from the options given below ?

- ✅ Create Cloud Watch alarms for Status check Failed _System metrics and select EC2 action-Recover the instance
- Write a script that queries the EC2 API for each instance status check
- Implement a third party monitoring tool
- Write a script that periodically shuts down and starts instances based on certain stats.

Q11)

Your company has an application hosted on an Elastic beanstalk environment. You have been instructed that whenever application changes occur and new versions need to be deployed that the fastest deployment approach Is employed.

Which of the following deployment mechanisms will fulfill this requirement?

- Immutable
- ✅ All at once
- Rolling
- Rolling with batch

Q12)

You are using Elastic beanstalk to deploy an application that consists of a web and application server. There is a requirement to run some python scripts before the application version is deployed to the web server.

Which of the following can be used to achieve this?

- ✅ Make use of container commands
- Make use of multiple elastic beanstalk environments
- Make use of Ducker containers
- Make use of custom resources

Q13)

You have a setup in AWS which consists of EC2 Instances sitting behind and ELB. The launching and termination of the Instances are controlled via an Auto scaling Group. The architecture consists of a My SQL AWS RDS database.

Which of the following can be used to induce one more step towards a self-healing architecture for this design?

- Create one more Auto scaling Group in another region for fault tolerance
- ✓ Enable Multi-AZ feature for the AWS RDS database.
- Enable Read Replica's for the AWS RDS database.
- Create one more ELB in another region for fault tolerance

Q14)

which is done by specifying the Auto Scaling Rolling Update policy. This retains the same Auto Scaling group and replaces old InstYour application is currently running on Amazon EC2 instances behind a load balancer. Your management has decided to use a Blue/Green deployment strategy.

How should you implement this for each deployment?

- ✓ Create a new load balancer with new Amazon EC2 instances, carry out the deployment, and then switch DNS over to the new load balancer using Amazon Route 53 after testing
- Using AWS Cloud Formation. create a test stack for validating the code, and then deploy the code to each production Amazon EC2 instance.
- Set up Amazon Route 53 health checks to fail over from any Amazon EC2 instance that is currently being deployed to.
- Launch more Amazon EC2 instances to ensure high availability, de-register each Amazon EC2 instance from the load balancer, upgrade it, and test it. and then register it again with the load balancer.

Q15) What is the amount of time that Ops work stacks services waits for a response from an underlying instance before deeming it as a failed instance?

- ✓ 5 minutes
- 1 minute
- 20 minutes
- 60 minutes

Q16)

You have a set of EC2 Instances running behind an ELB. These EC2 Instances are launched via an Auto scaling Group. There is a requirement to ensure that the logs from the server are stored in a durable storage layer. This is so that log data can be analyzed by staff in the future.

Which of the following steps can be implemented to ensure this requirement is fulfilled. Choose 2 answers from the options given below ?

- ✓ Use AWS Data Pipeline to move log data from the Amazon 53 bucket to Amazon Redshift in order to proce and run reports
- Use AWS Data Pipeline to move log data from the Amazon S3 bucket to Amazon SQS in order to process and run reports
- On the web servers, create a scheduled task that executes a script to rotate and transmit the logs to Amazon Glacier.
- ✓ On the web servers, create a scheduled task that executes a script to rotate and transmit the logs to an Amazon S3 bucket.

Q17)

There is a company website that is going to be launched in the coming weeks. There is a probability that the traffic will be quite high in the ffrst couple of weeks.

In the event of a load failure, how can you set up DNS fall over to a static website? Choose the correct answer from the options given below?

- Duplicate the exact application architecture in another region and configure DNS weight-based routing
- ✓ Use Route 53 with the failover option to failover to a static S3 website bucket or Cloud Front distribution.
- Add more servers in case the application fails.
- Enable failover to an on-premise data center to the application hosted there.

Q18)

Your company uses an application hosted in AWS which consists of EC2 Instances. The logs of the EC2 instances need to be processed and analyzed in real time , since this is a requirement from the IT Security department.

Which of the following can be used to process the logs in real time?

- Use Amazon Glacier to store the logs and then use Amazon Kinesis to process and analyze the logs in real
- Use another EC2 Instance with a larger instance type to process the logs
- Use Cloud watch logs to process and analyze the logs in real time
- ✓ Use Amazon 53 to store the logs and then use Amazon Kinesis to process and analyze the logs in real time.

Q19)

Your company is hosting an application in AWS. The application consists of a set of web servers and AWS RDS. The application is a read intensive application. It has been noticed that the response time of the application decreases due to the load on the AWS RDS instance.

Which of the following measures can be taken to scale the data tier. Choose 2 answers from the options given below Please

select:

- [x] Create Amazon DB Read Replica's. Configure the application layer to query the read replica's for query needs.
- [ ] Use Auto scaling to scale out and scale in the database tier
- [ ] Use SQS to cache the database queries
- [x] Use Elastic Cache in front of your Amazon RDS DB to cache common queries.

Q20)

As part of your deployment pipeline, you want to enable automated testing of your AWS Cloud Formation template.

What testing should be performed to enable faster feedback while minimizing costs and risk? Select three answers from the options given below ?

- [x] Use the AWS Cloud Formation Validate Template to validate the syntax of the template
- [x] Update the stack with the template. If the template fails rollback will return the stack and its resources to exactly the same state.
- [ ] Validate the template's is syntax using a general JSON parser.
- [x] When creating the stack. specify an Amazon SNS topic to which your testing system Is subscribed. Your testing system runs tests when it receives notification that the stack is created or updated.

Q21) Which of the following resource is used in Cloud formation to create nested stacks Please select:

- [ ] AWS::Cloud Formation::Nested Stack
- [ ] AWS::Cloud Formation::Nested
- [x] AWS::Cloud Formation::Stack .
- [ ] AWS::Cloud Formation::Stack Net

Q22) Which of the following tools does not directly support AWS Ops Works, for monitoring your stacks?

- [ ] AWS CloudTrail
- [ ] Amazon Cloud Watch Metrics
- [x] AWS Config
- [ ] Amazon CloudWatch Logs

Q23)

You need to investigate one of the instances which is part of your Auto scaling Group.

How would you implement this.

- [ ] Put the instance in a In-service state
- [ ] Suspend the AZR balance process so that Auto scaling will not terminate the instance
- [x] Put the instance in a standby state
- [ ] Suspend the Add To Load Balancer process

Q24) Which of the following services can be used to instrument Dev ops in your company?

- [ ] AWS Cloud formation
- [ ] AWS Elastic Beanstalk
- [x] All of the above
- [ ] AWS Ops work

Q25)

You have a legacy application running that uses an m4.large instance size and cannot scale with Auto Scaling, but only has peak performance 5% of the time. This Is a huge waste of resources and money so your Senior Technical Manager has set you the task of trying to reduce costs while still keeping the legacy application running as it should.

Which of the following would best accomplish the task your manager has set you? Choose the correct answer from the options below ?

- [x] Use a T2 burs table performance instance.
- [ ] Use two t2.nano instances that have single Root I/O Virtualizations.
- [ ] Use t2.nano instance and add spot instances when they are required.
- [ ] Use a C4.large instance with enhanced networking.

Q26)

You have an application hosted in AWS , which sits on EC2 Instances behind an Elastic Load Balancer. You have added a new feature to your application and are now receiving complaints from users that the site has a slow response.

Which of the below actions can you carry out to help you pinpoint the issue ?

- [ ] Use Cloud trail to log all the API calls, and then traverse the log files to locate the issue
- [ ] Use Cloud watch, monitor the CPU utilization to see the times when the CPU peaked
- [x] Create some custom Cloud watch metrics which are pertinent to the key features of your application
- [ ] Review the Elastic Load Balancer logs

Q27) A leading game development company is planning to host its latest video game on AWS. It is expected that there will be millions of users around the globe that will play the game. The architecture should allow players to send or receive data on the backend in real-time for a better gaming experience. The application libraries and user data of the game must also comply with the data residency requirement wherein all files must remain in the same region. CodeCommit, CodeBuild, CodeDeploy, and CodePipeline should be utilized to build the CI/CD process.
As a DevOps Engineer, which of the following is the MOST suitable and efficient solution that you should implement to satisfy this requirement?

- Create a new pipeline using AWS CodePipeline and a CodeCommit repository as the source in your primary AWS region. Configure the repository to trigger the build and deployment actions whenever there is a new code update. Using the AWS Management Console, set up the pipeline to use cross-AZ actions that will run the build and deployment actions to other Availability Zones. The pipeline will automatically store output files on a default artifact bucket on each AZ.
- ✓ Create a new pipeline using AWS CodePipeline and a CodeCommit repository as the source in your primary AWS region. Configure the repository to trigger the build and deployment actions whenever there is a new code update. Using the AWS Management Console, set up the pipeline to use cross-region actions that will run the build and deployment actions to other regions. The pipeline will automatically store output files on a default artifact bucket on each region.

Explanation:-AWS CodePipeline includes a number of actions that help you configure build, test, and deploy resources for your automated release process. You can add actions to your pipeline that are in an AWS Region different from your pipeline. When an AWS service is the provider for an action, and this action type/provider type are in a different AWS Region from your pipeline, this is a cross-region action. Certain action types in CodePipeline may only be available in certain AWS Regions. Also note that there may AWS Regions where an action type is available, but a specific AWS provider for that action type is not available.
You can use the console, AWS CLI, or AWS CloudFormation to add cross-region actions in pipelines. If you use the console to create a pipeline or cross-region actions, default artifact buckets are configured by CodePipeline in the Regions where you have actions. When you use the AWS CLI, AWS CloudFormation, or an SDK to create a pipeline or cross-region actions, you provide the artifact bucket for each Region where you have actions. You must create the artifact bucket and encryption key in the same AWS Region as the cross-region action and in the same account as your pipeline.
You cannot create cross-region actions for the following action types: source actions, third-party actions, and custom actions. When a pipeline includes a cross-region action as part of a stage, CodePipeline replicates only the input artifacts of the cross-region action from the pipeline Region to the action's Region. The pipeline Region and the Region where your CloudWatch Events change detection resources are maintained remain the same. The Region where your pipeline is hosted does not change.
Hence, the correct answer is: Create a new pipeline using AWS CodePipeline and a CodeCommit repository as the source in your primary AWS region. Configure the repository to trigger the build and deployment actions whenever there is a new code update. Using the AWS Management Console, set up the pipeline to use cross-region actions that will run the build and deployment actions to other regions. The pipeline will automatically store output files on a default artifact bucket on each region.
The option that says: Create a new pipeline using AWS CodePipeline and a CodeCommit repository as the source on multiple AWS regions. Configure the repository of the pipeline for each region to trigger the build and deployment actions whenever there is a new code update. The pipeline will automatically store output files on a default artifact bucket on each region is incorrect because this deployment set up is not efficient since you have to maintain several pipelines in multiple AWS regions. A better solution would be to simply configure cross-region actions to build and deploy the application on multiple regions.
The option that says: Create a new pipeline using AWS CodePipeline and a CodeCommit repository as the source in your primary AWS region. Configure the repository to trigger the build and deployment actions whenever there is a new code update. Using the AWS Management Console, set up the pipeline to use cross-AZ actions that will run the build and deployment actions to other Availability Zones. The pipeline will automatically store output files on a default artifact bucket on each AZ is incorrect because there is no such thing as cross-AZ actions but only cross-region actions. You can build, test, and deploy resources to other AWS regions by adding cross-region actions in your pipeline.
The option that says: Create a new pipeline using AWS CodePipeline and a CodeCommit repository as the source in your primary AWS region. Set a CodeBuild test action to run the automated unit and integration tests. Configure the repository to trigger the build and deployment actions whenever there is a new code update. Using the AWS Management Console, set up the pipeline to use cross-region actions that will run the build and deployment actions to other regions. Manually configure the pipeline to automatically store output files on a single S3 bucket is incorrect because this will violate the data residency requirement. Take note that there is a requirement that the application libraries and user data of the game must remain in the same region.
References:
https://docs.aws.amazon.com/codepipeline/latest/userguide/actions-create-cross-region.html
https://aws.amazon.com/getting-started/projects/set-up-ci-cd-pipeline/
https://aws.amazon.com/about-aws/whats-new/2018/11/aws-codepipeline-now-supports-cross-region-actions/
Check out this AWS CodePipeline Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-codepipeline/

- Create a new pipeline using AWS CodePipeline and a CodeCommit repository as the source on multiple AWS regions. Configure the repository of the pipeline for each region to trigger the build and deployment actions whenever there is a new code update. The pipeline will automatically store output files on a default artifact bucket on each region.
- Create a new pipeline using AWS CodePipeline and a CodeCommit repository as the source in your primary AWS region. Set a CodeBuild test action to run the automated unit and integration tests. Configure the repository to trigger the build and deployment actions whenever there is a new code update. Using the AWS Management Console, set up the pipeline to use cross-region actions that will run the build and deployment actions to other regions. Manually configure the pipeline to automatically store output files on a single S3 bucket.

Q28)

You need to store a large volume of data. The data needs to be readily accessible for a short period, but then needs to be archived indefinitely after that.

What is a cost-effective solution that an help fulfil this requirement?

- Store your data in Amazon S3, and use lifecycle policies to archive to S3-Infrequently Access
- Keep all your data in 53 since this is a durable storage.
- ✓ Store your data in Amazon 53. and use lifecycle policies to archive to Amazon Glacier
- Store your data in an EBS volume, and use lifecycle policies to archive to Amazon Glacier.

Q29) Which of the following features of the Elastic Beanstalk service will allow you to perform a Blue Green Deployment ?

- Rebuild Environment
- Environment Configuration
- Swap Environment
- ✅ Swap URL's

Q30)

You have been requested to use Cloud Formation to maintain version control and achieve automation for the applications in your organization.

How can you best use Cloud Formation to keep everything agile and maintain multiple environments while keeping cost down?

- ✅ Create separate templates based on functionality, create nested stacks with Cloud Formation.
- Create multiple templates in one Cloud Formation stack.
- Use Cloud Formation custom resources to handle dependencies between stacks
- Combine all resources into one template for version control and automation.

Q31)

The operations team and the development team want a single place to view both operating system and application logs.

How should you implement this using AWS services? Choose two from the options below Please select:

- Using AWS Cloud Formation. merge the application logs with the operating system logs, and use IAM Roles to allow both teams to have access to view console output from Amazon EC2.
- ✅ Using AWS Cloud Formation. create a Cloud Watch Logs Log Group and send the operating system and application logs of interest using the Cloud Watch Logs Agent.
- Using AWS Cloud Formation and configuration management. set up remote logging to send events via UDP packets to Cloud Trail.
- ✅ Using configuration management, set up remote logging to send events to Amazon Kinesis and insert into Amazon Cloud Search or Amazon Red shift. depending on available analytic tools

---

Q32)

You have a set of EC2 Instances in an Auto scaling Group that processes messages from an SQS queue. The messages contain the location In S3 from where video?s need to be processed by the EC2 Instances.

When a scale in happens, it is noticed that an at times that the EC2 Instance Is still in a state of processing a video when the instance is terminated.

How can you implement a solution which will ensure this does not happen? Please select:

- Increase the minimum and maximum size for the Auto Scaling group. and change the scaling policies so they scale less dynamically.
- Suspend the AZRe balance termination policy
- Change the Cool Down property for the Auto scaling Group
- ✅ Use lifecycle hooks to ensure the processing is complete before the termination occurs

---

Q33)

You are designing a cloud formation template to install a set of web servers on EC2 Instances.

The following User data needs to be passed to the EC2 Instances # !/b in/bash sudo apt-get update sudo apt-get install -y nginx Where in the cloudformation template would you ideally pass this User Data Please select:

- ✅ In the properties section of the EC2 Instance in the resources section
- In the Metadata section othe EC2 Instance in the Output section
- In the Metadata section of the EC2 Instance in the resources section
- In the properties section of the EC2 Instance in the Output section

---

Q34) A CTO of a leading insurance company has recently decided to migrate its online customer portal to AWS. Their customers will be using the online portal to view their paid insurance premiums and manage their accounts. For improved scalability, the application should be hosted in an Auto Scaling group of On-Demand EC2 instances with a custom Amazon Machine Image (AMI). The same architecture will also be used for their non-production environments (DEV, TEST, and STAGING). You are instructed by the CTO to design a deployment strategy that securely stores the credentials of each environment, expedites the startup time for the EC2 instances, and allows the same AMI to work in all environments.
As a DevOps Engineer, how should you set up the deployment configuration to accomplish this task?

- Add a tag to each EC2 instance based on their environment. Use a preconfigured AMI from AWS Marketplace. Write a bootstrap script in the User Data to analyze the tag and set the environment configuration accordingly. Use the AWS Secrets Manager to store the credentials.
- Add a tag to each EC2 instance based on their environment. Preconfigure the AMI by installing all of the required applications and software dependencies using the AWS Systems Manager Session Manager. Set up a Lambda function that will be invoked by the User Data to analyze the associated tag and set the environment configuration accordingly. Use the AWS Secrets Manager to store the credentials.
- ✅ Add a tag to each EC2 instance based on their environment. Use AWS Systems Manager Automation to preconfigure the AMI by installing all of the required applications and software dependencies. Write a bootstrap script in the User Data to analyze the tag and set the environment configuration accordingly. Use the AWS Systems Manager Parameter Store to store the credentials as Secure String parameters.

Explanation:-AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management and secrets management. You can store data such as passwords, database strings, and license codes as parameter values. You can store values as plain text or encrypted data. You can then reference values by using the unique name that you specified when you created the parameter. Highly scalable, available, and durable, Parameter Store is backed by the AWS Cloud.
Systems Manager Automation simplifies common maintenance and deployment tasks of Amazon EC2 instances and other AWS resources. Automation enables you to do the following.
- Build Automation workflows to configure and manage instances and AWS resources.

- Create custom workflows or use pre-defined workflows maintained by AWS.
- Receive notifications about Automation tasks and workflows by using Amazon CloudWatch Events.
- Monitor Automation progress and execution details by using the Amazon EC2 or the AWS Systems Manager console.
Automation offers one-click automations for simplifying complex tasks such as creating golden Amazon Machines Images (AMIs), and recovering unreachable EC2 instances. Here are some examples:
- Use the AWS-UpdateLinuxAmi and AWS-UpdateWindowsAmi documents to create golden AMIs from a source AMI. You can run custom scripts before and after updates are applied. You can also include or exclude specific packages from being installed.
- Use the AWSSupport-ExecuteEC2Rescue document to recover impaired instances. An instance can become unreachable for a variety of reasons, including network misconfigurations, RDP issues, or firewall settings. Troubleshooting and regaining access to the instance previously required dozens of manual steps before you could regain access. The AWSSupport-ExecuteEC2Rescue document lets you regain access by specifying an instance ID and clicking a button.
A Systems Manager Automation document defines the Automation workflow (the actions that Systems Manager performs on your managed instances and AWS resources). Automation includes several pre-defined Automation documents that you can use to perform common tasks like restarting one or more Amazon EC2 instances or creating an Amazon Machine Image (AMI). Documents use JavaScript Object Notation (JSON) or YAML, and they include steps and parameters that you specify. Steps run in sequential order.
Hence, the correct answer is: Add a tag to each EC2 instance based on their environment. Use AWS Systems Manager Automation to preconfigure the AMI by installing all of the required applications and software dependencies. Write a bootstrap script in the User Data to analyze the tag and set the environment configuration accordingly. Use the AWS Systems Manager Parameter Store to store the credentials as Secure String parameters.
The option that says: Add a tag to each EC2 instance based on their environment. Preconfigure the AMI by installing all of the required applications and software dependencies using the AWS Systems Manager Session Manager. Set up a Lambda function that will be invoked by the User Data to analyze the associated tag and set the environment configuration accordingly. Use the AWS Secrets Manager to store the credentials is incorrect because the Session Manager service is just a fully managed AWS Systems Manager capability that lets you manage your Amazon EC2 instances through an interactive one-click browser-based shell or through the AWS CLI. It is not capable to build a custom AMI, unlike Systems Manager Automation.
The option that says: Add a tag to each EC2 instance based on their environment. Use a preconfigured AMI from AWS Marketplace. Write a bootstrap script in the User Data to analyze the tag and set the environment configuration accordingly. Use the AWS Secrets Manager to store the credentials is incorrect because the company is using a custom AMI and not a public AMI from AWS Marketplace. You have to preconfigure the AMI using the Systems Manager Automation instead.
The option that says: Add a tag to each EC2 instance based on their environment. Preconfigure the AMI by installing all of the required applications and software dependencies using the AWS Systems Manager State Manager. Set up a Lambda function that will be invoked by the User Data to analyze the associated tag and set the environment configuration accordingly. Use the AWS Systems Manager Parameter Store to store the credentials as Secure String parameters is incorrect because the AWS Systems Manager Patch Manager simply automates the process of patching managed instances with both security-related and other types of updates. A better solution is to preconfigure the AMI using the Systems Manager Automation instead.
References:
https://aws.amazon.com/blogs/big-data/create-custom-amis-and-push-updates-to-a-running-amazon-emr-cluster-using-amazon-ec2-systems-manager/
https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-automation.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/automation-documents.html
Check out this AWS Systems Manager Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-systems-manager/
○ Add a tag to each EC2 instance based on their environment. Preconfigure the AMI by installing all of the required applications and software dependencies using the AWS Systems Manager Patch Manager. Set up a Lambda function that will be invoked by the User Data to analyze the associated tag and set the environment configuration accordingly. Use the AWS Systems Manager Parameter Store to store the credentials as Secure String parameters.

Q35) A software development company has a hybrid cloud environment wherein its on-premises data center is connected to their VPC using an AWS Virtual Private Network (AWS VPN). Its development department has a proprietary source code analysis tool that follows the Open Web Application Security Project (OWASP) standard. The tool is hosted on a dedicated server in its data center that checks the code quality and security vulnerabilities of their projects. The company plans to use this tool to run checks against the source code as part of the pipeline before the code is compiled into a deployable package in CodeDeploy. The code checks take approximately an hour to complete.
As a DevOps Engineer, which among the options below is the MOST suitable solution that you should implement?

○ Create a pipeline in AWS CodePipeline. Create a shell script that clones the code repository from CodeCommit and run the source code analysis tool on-premises. Create an action in the pipeline that executes the shell script after the source stage and configure it to return the results to CodePipeline.
○ Create a pipeline in AWS CodePipeline. Expose the web services of the on-premises source-code analysis tool over the Internet. Set up an action that will run the tool using an API call. Set the pipeline to execute the script after the source stage. After the processing has been done, send the results to a public S3 bucket. Configure the pipeline to poll the contents of the bucket.
○ Create a pipeline in AWS CodePipeline. Set up an action that invokes a custom Lambda function after the source stage. Configure the function to execute the source code analysis tool, and return the results to CodePipeline. Ensure that the function waits for the execution to complete and store the output in a specified S3 bucket.
✓ Create a pipeline in AWS CodePipeline. Set up a custom action type and create an associated job worker that runs on-premises. Set the pipeline to invoke the custom action after the source stage. Configure the job worker to poll CodePipeline for job requests for the custom action then execute the source code analysis tool and return the status result to CodePipeline.
Explanation:-AWS CodePipeline includes a number of actions that help you configure build, test, and deploy resources for your automated release process. If your release process includes activities that are not included in the default actions, such as an internally developed build process or a test suite, you can create a custom action for that purpose and include it in your pipeline. You can use the AWS CLI to create custom actions in pipelines associated with your AWS account.
You can create custom actions for the following AWS CodePipeline action categories:
- A custom build action that builds or transforms the items
- A custom deploy action that deploys items to one or more servers, websites, or repositories
- A custom test action that configures and runs automated tests
- A custom invoke action that runs functions
When you create a custom action, you must also create a job worker that will poll CodePipeline for job requests for this custom action, execute the job, and return the status result to CodePipeline. This job worker can be located on any computer or resource as long as it has access to the public endpoint for CodePipeline. To easily manage access and security, consider hosting your job worker on an Amazon EC2 instance.
The following diagram shows a high-level view of a pipeline that includes a custom build action:
When a pipeline includes a custom action as part of a stage, the pipeline will create a job request. A custom job worker detects that request and

performs that job (in this example, a custom process using third-party build software). When the action is complete, the job worker returns either a success result or a failure result. If a success result is received, the pipeline will transition the revision and its artifacts to the next action. If a failure is returned, the pipeline will not transition the revision to the next action in the pipeline.
Hence, the correct answer is: Create a pipeline in AWS CodePipeline. Set up a custom action type and create an associated job worker that runs on-premises. Set the pipeline to invoke the custom action after the source stage. Configure the job worker to poll CodePipeline for job requests for the custom action then execute the source code analysis tool and return the status result to CodePipeline.
The option that says: Create a pipeline in AWS CodePipeline. Set up an action that invokes a custom Lambda function after the source stage. Configure the function to execute the source code analysis tool, and return the results to CodePipeline. Ensure that the function waits for the execution to complete and store the output in a specified S3 bucket is incorrect because using a custom Lambda function to execute the source code analysis tool is not an appropriate solution. Remember that Lambda functions can run up to 15 minutes only and the code checks could take approximately an hour to complete. It is likely that the Lambda function will timeout in the middle of the processing.
The option that says: Create a pipeline in AWS CodePipeline. Expose the web services of the on-premises source-code analysis tool over the Internet. Set up an action that will run the tool using an API call. Set the pipeline to execute the script after the source stage. After the processing has been done, send the results to a public S3 bucket. Configure the pipeline to poll the contents of the bucket is incorrect because this setup has a lot of security vulnerabilities. Exposing the web services of the tool can open up attacks to the on-premises data center. The use of a public S3 bucket is inappropriate as well. Moreover, a CodePipeline cannot directly poll the contents of an S3 bucket.
The option that says: Create a pipeline in AWS CodePipeline. Create a shell script that clones the code repository from CodeCommit and run the source code analysis tool on-premises. Create an action in the pipeline that executes the shell script after the source stage and configure it to return the results to CodePipeline is incorrect because writing a custom shell script is not needed. A better solution is to simply create a custom action type and create an associated job worker that runs on-premises.
References:
https://docs.aws.amazon.com/codepipeline/latest/userguide/actions-create-custom-action.html
https://docs.aws.amazon.com/codepipeline/latest/userguide/actions.html
Check out this AWS CodePipeline Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-codepipeline/

Q36)

A popular e-commerce website which has customers across the globe is hosted in the us-east-1 AWS region with a backup site in the us-west-1 region. Due to an unexpected regional outage in the us-east-1 region, the company initiated their disaster recovery plan and turned on the backup site. However, they discovered that the actual failover still entails several hours of manual effort to prepare and switch over the database. They also noticed that the database is missing up to three hours of data transactions when the regional outage happened.

Which of the following solutions should the DevOps engineer implement which will improve the RTO and RPO of the website for the cross-region failover?

● Configure an Amazon RDS Multi-AZ Deployment configuration and place the standby instance in the us-west-1 region. Set up the RDS option group to enable multi-region availability for native automation of cross-region recovery as well as for continuous data replication. Set up a notification system using Amazon SNS which is integrated with AWS Health API to monitor RDS-related systems events and notify the Operations team. In the actual failover where the primary database instance is down, RDS will automatically make the standby instance in the backup region as the primary instance.
● Use an ECS cluster to host a custom python script that calls the RDS API to create a snapshot of the database, create a cross-region snapshot copy, and restore the database instance from a snapshot in the backup region. Create a scheduled job using CloudWatch Events that triggers the Lambda function to snapshot a database instance every hour. Set up an SNS topic that will receive published messages about AWS-initiated RDS events from Trusted Advisor that will trigger the function to create a cross-region snapshot copy. During failover, restore the database from a snapshot in the backup region.
● Create a snapshot every hour using Amazon RDS scheduled instance lifecycle events which will also allow you to monitor specific RDS events. Perform a cross-region snapshot copy into the us-west-1 backup region once the SnapshotCreateComplete event occurred. Create an Amazon CloudWatch Alert which will trigger an action to restore the Amazon RDS database snapshot in the backup region when the CPU Utilization metric of the RDS instance in CloudWatch falls to 0% for more than 15 minutes.
✅ Use Step Functions with 2 Lambda functions that call the RDS API to create a snapshot of the database, create a cross-region snapshot copy, and restore the database instance from a snapshot in the backup region. Use CloudWatch Events to trigger the function to take a database snapshot every hour. Set up an SNS topic that will receive published messages from AWS Health API, RDS availability and other events that will trigger the Lambda function to create a cross-region snapshot copy. During failover, Configure the Lambda function to restore the database from a snapshot in the backup region.
Explanation:-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.
An incremental snapshot contains only the data that has changed after the most recent snapshot of the same DB instance. Incremental snapshot copying is faster and results in lower storage costs than full snapshot copying. Incremental snapshot copying across AWS Regions is supported for both unencrypted and encrypted snapshots. For shared snapshots, copying incremental snapshots is not supported. For shared snapshots, all of the copies are full snapshots, even within the same region.
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.
Take note that when you copy a source snapshot that is a snapshot copy, the copy isn't incremental because the snapshot copy doesn't include the required metadata for incremental copies.
Hence, the correct solution is: Use Step Functions with 2 Lambda functions that call the RDS API to create a snapshot of the database, create a cross-region snapshot copy, and restore the database instance from a snapshot in the backup region. Use CloudWatch Events to trigger the function to take a database snapshot every hour. Set up an SNS topic that will receive published messages from AWS Health API, RDS availability and other events that will trigger the Lambda function to create a cross-region snapshot copy. During failover, Configure the Lambda function to restore the database from a snapshot in the backup region.
The option that says: Configure an Amazon RDS Multi-AZ Deployment configuration and place the standby instance in the us-west-1 region. Set up the RDS option group to enable multi-region availability for native automation of cross-region recovery as well as for continuous data replication. Set up a notification system using Amazon SNS which is integrated with AWS Health API to monitor RDS-related systems events and notify the Operations team. In the actual failover where the primary database instance is down, RDS will automatically make the standby instance in the backup region as the primary instance is incorrect because the standby instance of an Amazon RDS Multi-AZ database can only be placed in the same AWS region where the primary instance is hosted. Thus, you cannot failover to the standby instance as your replacement to your primary

instance in another region. A better solution would be to set up a cross-region snapshot copy from the primary to the backup region. Another solution would be to use Read Replicas since these can be placed in another AWS region.
The option that says: Create a snapshot every hour using Amazon RDS scheduled instance lifecycle events which will also allow you to monitor specific RDS events. Perform a cross-region snapshot copy into the us-west-1 backup region once the SnapshotCreateComplete event occurred. Create an Amazon CloudWatch Alert which will trigger an action to restore the Amazon RDS database snapshot in the backup region when the CPU Utilization metric of the RDS instance in CloudWatch falls to 0% for more than 15 minutes is incorrect because you cannot create a snapshot using the Amazon RDS scheduled instance lifecycle events.
The option that says: Use an ECS cluster to host a custom python script that calls the RDS API to create a snapshot of the database, create a cross-region snapshot copy, and restore the database instance from a snapshot in the backup region. Create a scheduled job using CloudWatch Events that triggers the Lambda function to snapshot a database instance every hour. Set up an SNS topic that will receive published messages about AWS-initiated RDS events from Trusted Advisor that will trigger the function to create a cross-region snapshot copy. During failover, restore the database from a snapshot in the backup region is incorrect because the AWS Trusted Advisor doesn't provide any information regarding AWS-initiated RDS events. You should use the AWS Health API instead. Moreover, it is not necessary to provision an ECS cluster just to host a custom python program when you can simply use Lambda functions instead.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CopySnapshot.html#USER_CopySnapshot.AcrossRegions
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Replication.CrossRegion.html
Check out this Amazon RDS Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-relational-database-service-amazon-rds/

Q37) A company has a web application which runs on an Auto Scaling group of EC2 instances across multiple Availability Zones behind an Application Load Balancer. For its database tier, it is using an Amazon RDS MySQL and routes the incoming traffic to the load balancer using Amazon Route 53. The Application Load Balancer has a health check that monitors the status of the web servers and verifies that the servers can properly access the database. For compliance purposes, the management instructed their Operations team to implement a geographically isolated disaster recovery site to ensure business continuity. The required RPO is 5 minutes while the RTO should be 2 hours.
Which of the following options require the LEAST amount of changes to the application stack?

● Clone the application stack except for RDS in a different AWS Region. Enable Amazon RDS Multi-AZ deployments configuration and deploy the standby database instance in the new region. Configure the new application stack to point to the local RDS database instance. Set up a latency routing policy in Route 53 that will automatically route traffic to the new stack in the event of an outage.
✅ Clone the application stack except for RDS in a different AWS Region. Create Read Replicas in the new region and configure the new application stack to point to the local Amazon RDS database instance. Set up a failover routing policy in Route 53 that will automatically route traffic to the new application stack in the event of an outage.
Explanation:-When you have more than one resource performing the same function — for example, more than one HTTP server or mail server — you can configure Amazon Route 53 to check the health of your resources and respond to DNS queries using only the healthy resources. Suppose you have a website with a domain name of tutorialsdojo.com which is hosted on six servers, two each in three data centers around the world. You can configure Amazon Route 53 to check the health of those servers and to respond to DNS queries for tutorialsdojo.com using only the servers that are currently healthy.
Route 53 can check the health of your resources in both simple and complex configurations:
- In simple configurations, you create a group of records that all have the same name and type, such as a group of weighted records with a type of A for tutorialsdojo.com. You then configure Route 53 to check the health of the corresponding resources. Route 53 responds to DNS queries based on the health of your resources.
- In more complex configurations, you create a tree of records that route traffic based on multiple criteria. For example, if latency for your users is your most important criterion, then you might use latency alias records to route traffic to the region that provides the best latency. The latency alias records might have weighted records in each region as the alias target. The weighted records might route traffic to EC2 instances based on the instance type. As with a simple configuration, you can configure Route 53 to route traffic based on the health of your resources.
Hence, the correct answer is: Clone the application stack except for RDS in a different AWS Region. Create Read Replicas in the new region and configure the new application stack to point to the local Amazon RDS database instance. Set up a failover routing policy in Route 53 that will automatically route traffic to the new application stack in the event of an outage
The option that says: Clone the entire application stack except for its RDS database in a different Availability Zone. Create Read Replicas in another Availability Zone and configure the new stack to point to the local RDS instance. Set up a failover routing policy in Route 53 that will automatically route traffic to the new stack in the event of an outage is incorrect because this is only deployed in another Availability Zone which could also be affected by an AWS Region outage. The new stack should be deployed on a totally separate AWS Region instead.
The option that says: Clone the application stack except for RDS in a different AWS Region. Enable Amazon RDS Multi-AZ deployments configuration and deploy the standby database instance in the new region. Configure the new application stack to point to the local RDS database instance. Set up a latency routing policy in Route 53 that will automatically route traffic to the new stack in the event of an outage is incorrect because a Multi-AZ RDS database spans to several Availability Zones within a single Region only, and not to an entirely new region. You cannot deploy the standby database instance in the new AWS region.
The option that says: Configure the Amazon RDS to use Multi-AZ deployments configuration and create Read Replicas. Increase the number of application servers of the stack. Set up a latency routing policy in Route 53 that will automatically route traffic to the application servers is incorrect because although this architecture can cope with an individual AZ outage, the systems will still be unavailable in the event of an AWS Region-wide unavailability.
References:
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/AWSHowTo.RDS.html
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/dns-failover-configuring.html
Check out these AWS Elastic Beanstalk and Amazon Route 53 Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-aws-elastic-beanstalk/
https://tutorialsdojo.com/aws-cheat-sheet-amazon-route-53/
Tutorials Dojo's AWS Certified DevOps Engineer Professional Exam Study Guide:
https://tutorialsdojo.com/aws-cheat-sheet-aws-certified-devops-engineer-professional/
● Configure the Amazon RDS to use Multi-AZ deployments configuration and create Read Replicas. Increase the number of application servers of the stack. Set up a latency routing policy in Route 53 that will automatically route traffic to the application servers
● Clone the entire application stack except for its RDS database in a different Availability Zone. Create Read Replicas in another Availability Zone and configure the new stack to point to the local RDS instance. Set up a failover routing policy in Route 53 that will automatically route traffic to the new stack in the event of an outage.

Q38) A government agency recently decided to modernize its network infrastructure using AWS. They are developing a solution to store confidential files containing Personally Identifiable Information (PII) and other sensitive financial records of its citizens. All data in the storage solution must be encrypted both at rest and in transit. In addition, all of its data must also be replicated in

two locations that are at least 450 miles apart from each other.
As a DevOps Engineer, what solution should you implement to meet these requirements?

- Set up primary and secondary S3 buckets in two separate Availability Zones that are at least 450 miles apart. Create a bucket policy to enforce access to the buckets only through HTTPS and enforce S3 SSE-C encryption on all objects uploaded to the bucket. Enable cross-region replication (CRR) between the two buckets.
- Set up primary and secondary S3 buckets in two separate Availability Zones that are at least 450 miles apart. Create a bucket policy to enforce access to the buckets only through HTTPS and enforce AWS KMS encryption on all objects uploaded to the bucket. Enable Transfer Acceleration between the two buckets. Set up a KMS Customer Master Key (CMK) in the primary region for encrypting objects.
- Set up primary and secondary Amazon S3 buckets in two separate AWS Regions that are at least 450 miles apart. Create an IAM role to enforce access to the buckets only through HTTPS. Set up a bucket policy to enforce Amazon S3-Managed Keys (SSE-S3) encryption on all objects uploaded to the bucket. Enable cross-region replication (CRR) between the two buckets.
- ✅ Set up primary and secondary S3 buckets in two separate AWS Regions that are at least 450 miles apart. Create a bucket policy to enforce access to the buckets only through HTTPS and enforce S3-Managed Keys (SSE-S3) encryption on all objects uploaded to the bucket. Enable cross-region replication (CRR) between the two buckets.

Explanation:-Availability Zones give customers the ability to operate production applications and databases that are more highly available, fault-tolerant, and scalable than would be possible from a single data center. AWS maintains multiple AZs around the world and more zones are added at a fast pace. Each AZ can be multiple data centers (typically 3), and at full scale can be hundreds of thousands of servers. They are fully isolated partitions of the AWS Global Infrastructure. With their own power infrastructure, the AZs are physically separated by a meaningful distance, many kilometers, from any other AZ, although all are within 100 km (60 miles of each other).
All AZs are interconnected with high-bandwidth, low-latency networking, over fully redundant, dedicated metro fiber providing high-throughput, low-latency networking between AZs. The network performance is sufficient to accomplish synchronous replication between AZs. AWS Availability Zones are also powerful tools for helping build highly available applications. AZs make partitioning applications about as easy as it can be. If an application is partitioned across AZs, companies are better isolated and protected from issues such as lightning strikes, tornadoes, earthquakes and more.
By default, Amazon S3 allows both HTTP and HTTPS requests. To comply with the s3-bucket-ssl-requests-only rule, confirm that your bucket policies explicitly deny access to HTTP requests. Bucket policies that allow HTTPS requests without explicitly denying HTTP requests might not comply with the rule.
To determine HTTP or HTTPS requests in a bucket policy, use a condition that checks for the key "aws:SecureTransport". When this key is true, this means that the request is sent through HTTPS. To be sure to comply with the s3-bucket-ssl-requests-only rule, create a bucket policy that explicitly denies access when the request meets the condition "aws:SecureTransport": "false". This policy explicitly denies access to HTTP requests.
In this scenario, you should use AWS Regions since AZs are physically separated by only 100 km (60 miles) from each other. Within each AWS Region, S3 operates in a minimum of three AZs, each separated by miles to protect against local events like fires, floods et cetera. Take note that you can't launch an AZ-based S3 bucket.
Hence, the correct answer is: Set up primary and secondary S3 buckets in two separate AWS Regions that are at least 450 miles apart. Create a bucket policy to enforce access to the buckets only through HTTPS and enforce S3-Managed Keys (SSE-S3) encryption on all objects uploaded to the bucket. Enable cross-region replication (CRR) between the two buckets.
The option that says: Set up primary and secondary S3 buckets in two separate Availability Zones that are at least 450 miles apart. Create a bucket policy to enforce access to the buckets only through HTTPS and enforce S3 SSE-C encryption on all objects uploaded to the bucket. Enable cross-region replication (CRR) between the two buckets is incorrect because you can't create Amazon S3 buckets in two separate Availability Zones since this is a regional service.
The option that says: Set up primary and secondary Amazon S3 buckets in two separate AWS Regions that are at least 450 miles apart. Create an IAM role to enforce access to the buckets only through HTTPS. Set up a bucket policy to enforce Amazon S3-Managed Keys (SSE-S3) encryption on all objects uploaded to the bucket. Enable cross-region replication (CRR) between the two buckets is incorrect because you have to use the bucket policy to enforce access to the bucket using HTTPS only and not an IAM role.
The option that says: Set up primary and secondary S3 buckets in two separate Availability Zones that are at least 450 miles apart. Create a bucket policy to enforce access to the buckets only through HTTPS and enforce AWS KMS encryption on all objects uploaded to the bucket. Enable Transfer Acceleration between the two buckets. Set up a KMS Customer Master Key (CMK) in the primary region for encrypting objects is incorrect because you have to enable Cross-Region replication and not Transfer Acceleration. This feature simply enables fast, easy, and secure transfers of files over long distances between your client and an S3 bucket but not data replication.
References:
https://docs.aws.amazon.com/config/latest/developerguide/s3-bucket-ssl-requests-only.html
https://aws.amazon.com/premiumsupport/knowledge-center/s3-bucket-policy-for-config-rule/
https://aws.amazon.com/about-aws/global-infrastructure/regions_az/
Check out this Amazon S3 Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-s3/

Q39) A software development company is using AWS CodeCommit, CodeBuild, and CodePipeline for their CI/CD process. A newly hired DevOps engineer has recently applied the AWS-managed AWSCodeCommitPowerUser policy to the IAM Role that is being used by the development team. After several weeks, the team leader discovered that their developers can now directly push code to the master branch which was prohibited before. All other new capabilities brought by the AWS-managed policy is needed by the team instead of this particular action. This has caused an incident in which a junior developer pushed an untested code in the master branch and the changes directly went to their production environment.
What should the DevOps engineer do to restrict this specific action?

- Modify the AWSCodeCommitPowerUser AWS-managed policy to include a deny rule for the codecommit: GitPush action for the specific repositories in the resource statement with a condition for the master branch.
- Replace the AWSCodeCommitPowerUser AWS-managed policy with AWSCodeCommitReadOnly policy instead. In the IAM Role of the developer team, add an Allow rule for the codecommit:GitPush action for the specific repositories in the resource statement with a condition for the master branch.
- Replace the AWSCodeCommitPowerUser AWS-managed policy with AWSCodeCommitFullAccess policy instead. Include a deny rule for the codecommit: GitPush action for the specific repositories in the resource statement with a condition for the master branch.
- ✅ Maintain the recently added AWSCodeCommitPowerUser AWS-managed policy but create an additional policy to include a Deny rule for the codecommit:GitPush action. Add a restriction for the specific repositories in the resource statement with a condition for the master branch.

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 to maintain the recently added AWSCodeCommitPowerUser AWS-managed policy but create an additional policy to include a Deny rule for the codecommit:GitPush action. Add a restriction for the specific repositories in the resource statement with a condition for the master branch.
The option that says: Replace the AWSCodeCommitPowerUser AWS-managed policy with AWSCodeCommitReadOnly policy instead. In the IAM Role of the developer team, add an Allow rule for the codecommit:GitPush action for the specific repositories in the resource statement with a condition for the master branch is incorrect because the AWSCodeCommitReadOnly policy is quite restrictive and it will block some of the required actions by the development team such as the ability to create pull requests. Take note that the scenario says that all other new capabilities brought by the new AWS-managed policy (AWSCodeCommitPowerUser) are needed by the team except for the action that allows direct code push to the master branch.
The option that says: Modify the AWSCodeCommitPowerUser AWS-managed policy to include a deny rule for the codecommit: GitPush action for the specific repositories in the resource statement with a condition for the master branch is incorrect because you cannot modify an AWS-managed policy. You should just maintain the recently added policy and then add certain Deny rules to meet the requirement.
The option that says: Replace the AWSCodeCommitPowerUser AWS-managed policy with AWSCodeCommitFullAccess policy instead. Include a deny rule for the codecommit: GitPush action for the specific repositories in the resource statement with a condition for the master branch is incorrect because 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. This does not follow the AWS best practice of granting least privilege.
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

---

Q40) A financial company has several accounting applications that are hosted in AWS and used by thousands of small and medium businesses. As part of its Business Continuity Plan, the company is required to set up an automatic DNS failover for its applications to a disaster recovery (DR) environment. They instructed their DevOps team to configure Amazon Route 53 to automatically route to an alternate endpoint when their primary application stack in us-west-1 region experiences an outage or degradation of service.
What steps should the team take to satisfy this requirement? (Select TWO)

- Set up a record in Route 53 with a latency routing policy configuration. Associate the record with the primary and secondary record sets to distribute traffic to healthy service endpoints.
- Set up a CloudWatch Alarm to monitor the primary Route 53 DNS endpoint and create a custom Lambda function. Execute the ChangeResourceRecordSets API call using the function to initiate the failover to the secondary DNS record.
- Set up a record in Route 53 with a Weighted routing policy configuration. Associate the record with the primary and secondary record sets to distribute traffic to healthy service endpoints.
- ✅ Use a Failover routing policy configuration. Set up alias records in Route 53 that route traffic to AWS resources. Set the Evaluate Target Health option to Yes, then create all of the required non-alias records.

Explanation:-Use an active-passive failover configuration when you want a primary resource or group of resources to be available the majority of the time and you want a secondary resource or group of resources to be on standby in case all the primary resources become unavailable. When responding to queries, Route 53 includes only the healthy primary resources. If all the primary resources are unhealthy, Route 53 begins to include only the healthy secondary resources in response to DNS queries.
To create an active-passive failover configuration with one primary record and one secondary record, you just create the records and specify Failover for the routing policy. When the primary resource is healthy, Route 53 responds to DNS queries using the primary record. When the primary resource is unhealthy, Route 53 responds to DNS queries using the secondary record.
You can configure a health check that monitors an endpoint that you specify either by IP address or by domain name. At regular intervals that you specify, Route 53 submits automated requests over the Internet to your application, server, or other resource to verify that it's reachable, available, and functional. Optionally, you can configure the health check to make requests similar to those that your users make, such as requesting a web page from a specific URL.
When Route 53 checks the health of an endpoint, it sends an HTTP, HTTPS, or TCP request to the IP address and port that you specified when you created the health check. For a health check to succeed, your router and firewall rules must allow inbound traffic from the IP addresses that the Route 53 health checkers use.
Hence, the correct answers are:
- Set up health checks in Route 53 for non-alias records to each service endpoint. Configure the network access control list and the route table to allow Route 53 to send requests to the endpoints specified in the health checks.
- Use a Failover routing policy configuration. Set up alias records in Route 53 that route traffic to AWS resources. Set the Evaluate Target Health option to Yes, then create all of the required non-alias records.
The option that says: Set up a record in Route 53 with a Weighted routing policy configuration. Associate the record with the primary and secondary record sets to distribute traffic to healthy service endpoints is incorrect because Weighted routing simply lets you associate multiple resources with a single domain name (tutorialsdojo.com) or subdomain name (blog.tutorialsdojo.com) and choose how much traffic is routed to each resource. This can be useful for a variety of purposes, including load balancing and testing new versions of software.
The option that says: Set up a CloudWatch Alarm to monitor the primary Route 53 DNS endpoint and a create a custom Lambda function. Execute the ChangeResourceRecordSets API call using the function to initiate the failover to the secondary DNS record is incorrect because you have to use a Failover routing policy. Calling the Route 53 API is not applicable nor useful at all in this scenario.
The option that says: Set up a record in Route 53 with a latency routing policy configuration. Associate the record with the primary and secondary record sets to distribute traffic to healthy service endpoints is incorrect because the Latency routing policy simply improves the application performance for your users by serving their requests from the AWS Region that provides the lowest latency. You have to use a Failover routing policy instead.

References:
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/dns-failover-types.html
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/health-checks-types.html
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/dns-failover-router-firewall-rules.html
Check out this Amazon Route 53 Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-route-53/

Q41) A leading financial company provides GraphQL APIs to allow its various customers and partners to consume its global stock market data. The web service is hosted in an Auto Scaling group of EC2 instances behind an Application Load Balancer. All new versions of the service are deployed via a CI/CD pipeline. A DevOps Engineer has been instructed to track the health of the service in deployment to avoid any issues. If the latency of the service increases more than the defined threshold then the deployment should be halted until the service has been fully recovered.
Which solution should the Engineer implement to provide the FASTEST detection time for deployment issues?

- Define the thresholds to roll back the deployments based on the latency using the MinimumHealthyHosts deployment configuration in AWS CodeDeploy. Rollback the deployment if the threshold was breached.
- Collect the ELB access logs and use a Lambda function to calculate and detect the average latency of the service. When latency increases beyond the defined threshold, trigger the alarm and stop the current deployment.
- Enable Detailed Monitoring in CloudWatch to monitor and detect the latency in the Application Load Balancer. When latency increases beyond the defined threshold, trigger a CloudWatch alarm and stop the current deployment.
- ✅ Calculate the average latency using Amazon CloudWatch metrics that monitors the Application Load Balancer. Associate a CloudWatch alarm with the CodeDeploy deployment group. When latency increases beyond the defined threshold, it will automatically trigger an alarm that automatically stops the on-going deployment.

Explanation:-You can create a CloudWatch alarm for an instance or Amazon EC2 Auto Scaling group you are using in your CodeDeploy operations. An alarm watches a single metric over a time period you specify and performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods. CloudWatch alarms invoke actions when their state changes (for example, from OK to ALARM). Using native CloudWatch alarm functionality, you can specify any of the actions supported by CloudWatch when an instance you are using in a deployment fails, such as sending an Amazon SNS notification or stopping, terminating, rebooting, or recovering an instance. For your CodeDeploy operations, you can configure a deployment group to stop a deployment whenever any CloudWatch alarm you associate with the deployment group is activated.
You can associate up to ten CloudWatch alarms with a CodeDeploy deployment group. If any of the specified alarms are activated, the deployment stops, and the status is updated to Stopped. To use this option, you must grant CloudWatch permissions to your CodeDeploy service role.
Hence, the correct answer is: Calculate the average latency using Amazon CloudWatch metrics that monitors the Application Load Balancer. Associate a CloudWatch alarm with the CodeDeploy deployment group. When latency increases beyond the defined threshold, it will automatically trigger an alarm that automatically stops the on-going deployment.
The option that says: Collect the ELB access logs and use a Lambda function to calculate and detect the average latency of the service. When latency increases beyond the defined threshold, trigger the alarm and stop the current deployment is incorrect because although ELB access logs contain latency information, you still need to parse the data using Lambda. The calculation process entails a significant amount of time. A faster solution would be to use Amazon CloudWatch metrics.
The option that says: Define the thresholds to roll back the deployments based on the latency using the MinimumHealthyHosts deployment configuration in AWS CodeDeploy. Rollback the deployment if the threshold was breached is incorrect because the MinimumHealthyHosts is just a property of the DeploymentConfig resource that defines how many instances must remain healthy during an AWS CodeDeploy deployment. It doesn't calculate the latency of the service.
The option that says: Enable Detailed Monitoring in CloudWatch to monitor and detect the latency in the Application Load Balancer. When latency increases beyond the defined threshold, trigger a CloudWatch alarm and stop the current deployment is incorrect because you cannot enable detailed monitoring in your Application Load Balancer. The detailed monitoring feature in CloudWatch is primarily used to collect data from your EC2 and other resources in 1-minute frequency for an additional cost. This level of frequency is already available for your load balancers. If there are requests flowing through the load balancer, Elastic Load Balancing measures and sends its metrics in 60-second intervals.
References:
https://docs.aws.amazon.com/codedeploy/latest/userguide/monitoring-create-alarms.html
https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-cloudwatch-metrics.html#loadbalancing-metrics-clb
https://aws.amazon.com/premiumsupport/knowledge-center/elb-latency-troubleshooting/
Check out this AWS CodeDeploy and Elastic Load Balancing Cheat Sheets:
https://tutorialsdojo.com/aws-cheat-sheet-aws-codedeploy/
https://tutorialsdojo.com/aws-cheat-sheet-aws-elastic-load-balancing-elb/

Q42) A recent production incident has caused a data breach in one of the company's flagship application which is hosted in an Auto Scaling group of EC2 instances. In order to prevent this from happening again, a DevOps engineer was tasked to implement a solution that will automatically terminate any instance in production which was manually logged into via SSH. All of the EC2 instances that are being used by the application already have an Amazon CloudWatch Logs agent installed.
Which of the following is the MOST automated solution that the DevOps engineer should implement?

- Set up and integrate a CloudWatch Logs subscription with AWS Step Functions to add a special FOR_DELETION tag to the specific EC2 instance that had an SSH login event. Create a CloudWatch Events rule to trigger a second AWS Lambda function everyday at 12 PM that will terminate all of the EC2 instances with this tag.
- Set up a CloudWatch Alarm which will be triggered when an SSH login event occurred and configure it to also send a notification to an SNS topic once the alarm is triggered. Instruct the Support and Operations team to subscribe to the SNS topic and then manually terminate the detected EC2 instance as soon as possible.
- Set up a CloudWatch Alarm that will be triggered when there is an SSH login event and configure it to send a notification to an SQS queue. Launch a group of EC2 worker instances to consume the messages from the SQS queue and terminate the detected EC2 instances.
- ✅ Set up a CloudWatch Logs subscription with an AWS Lambda function which is configured to add a FOR_DELETION tag to the Amazon EC2 instance that produced the SSH login event. Run another Lambda function every day using the CloudWatch Events rule to terminate all EC2 instances with the custom tag for deletion.

Explanation:-You can use subscriptions to get access to a real-time feed of log events from CloudWatch Logs and have it delivered to other services such as a Amazon Kinesis stream, Amazon Kinesis Data Firehose stream, or AWS Lambda for custom processing, analysis, or loading to other systems. To begin subscribing to log events, create the receiving source, such as a Kinesis stream, where the events will be delivered. A subscription filter defines the filter pattern to use for filtering which log events get delivered to your AWS resource, as well as information about where to send matching log events to.
CloudWatch Logs also produces CloudWatch metrics about the forwarding of log events to subscriptions. You can use a subscription filter with Kinesis, Lambda, or Kinesis Data Firehose.

Hence, the correct answer is: Set up a CloudWatch Logs subscription with an AWS Lambda function which is configured to add a FOR_DELETION tag to the Amazon EC2 instance that produced the SSH login event. Run another Lambda function everyday using the CloudWatch Events rule to terminate all EC2 instances with the custom tag for deletion.
The option that says: Set up and integrate a CloudWatch Logs subscription with AWS Step Functions to add a special FOR_DELETION tag to the specific EC2 instance that had an SSH login event. Create a CloudWatch Events rule to trigger a second AWS Lambda function every day at 12 PM that will terminate all of the EC2 instances with this tag is incorrect because a CloudWatch Logs subscription cannot be directly integrated with an AWS Step Functions application.
The option that says: Set up a CloudWatch Alarm which will be triggered when an SSH login event occurred and configure it to also send a notification to an SNS topic once the alarm is triggered. Instruct the Support and Operations team to subscribe to the SNS topic and then manually terminate the detected EC2 instance as soon as possible is incorrect because although you can configure your Amazon CloudWatch Alarms to send a notification to SNS, this solution still involves a manual process. Remember that the scenario is asking for an automated system for this scenario.
The option that says: Set up a CloudWatch alarm that will be triggered when there is an SSH login event and configure it to send to a notification to an SQS queue. Launch a group of EC2 worker instances to consume the messages from the SQS queue and terminate the detected EC2 instances is incorrect because using SQS as well as worker instances is unnecessary since you can simply use Lambda functions for processing. In addition, Amazon CloudWatch Alarms can only send notifications to SNS and not SQS.
References:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/SubscriptionFilters.html#LambdaFunctionExample
https://aws.amazon.com/blogs/security/how-to-monitor-and-visualize-failed-ssh-access-attempts-to-amazon-ec2-linux-instances/
https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/Create-CloudWatch-Events-Scheduled-Rule.html
Check out this Amazon CloudWatch Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudwatch/

Q43) A company is planning to migrate its online customer portal to AWS. It should be hosted in AWS Elastic Beanstalk and use Amazon RDS MySQL in Multi-AZ configuration for its database. A DevOps Engineer was instructed to ensure that the application resources must be at full capacity during deployment by using a new group of instances. The solution should also include a way to roll back the change easily and prevent issues caused by partially completed rolling deployments. The application performance should not be affected while a new version of the app is being deployed.
Which is the MOST cost-effective deployment set up that the DevOps Engineer should implement to meet these requirements?

Host the online customer portal using AWS Elastic Beanstalk and integrate it to an external Amazon RDS MySQL database in Multi-AZ deployments configuration. Configure the Elastic Beanstalk to use blue/green deployment for releasing the new application version to a new environment. Swap the CNAME in the two environments to redirect traffic to the new version using the Swap Environment URLs feature. Once the deployment has been successfully implemented, keep the old environment running as a backup.

Host the online customer portal using AWS Elastic Beanstalk and integrate it to an external Amazon RDS MySQL database in Multi-AZ deployments configuration. Use immutable updates for application deployments.

Explanation:-AWS Elastic Beanstalk provides several options for how deployments are processed, including deployment policies (All at once, Rolling, Rolling with additional batch, and Immutable) and options that let you configure the batch size and health check behavior during deployments. By default, your environment uses all-at-once deployments. If you created the environment with the EB CLI and it's an automatically scaling environment (you didn't specify the --single option), it uses rolling deployments.
With rolling deployments, Elastic Beanstalk splits the environment's EC2 instances into batches and deploys the new version of the application to one batch at a time, leaving the rest of the instances in the environment running the old version of the application. During a rolling deployment, some instances serve requests with the old version of the application, while instances in completed batches serve other requests with the new version.
To maintain full capacity during deployments, you can configure your environment to launch a new batch of instances before taking any instances out of service. This option is known as a rolling deployment with an additional batch. When the deployment completes, Elastic Beanstalk terminates the additional batch of instances.
Immutable deployments perform an immutable update to launch a full set of new instances running the new version of the application in a separate Auto Scaling group, alongside the instances running the old version. Immutable deployments can prevent issues caused by partially completed rolling deployments. If the new instances don't pass health checks, Elastic Beanstalk terminates them, leaving the original instances untouched.
If your application doesn't pass all health checks, but still operates correctly at a lower health status, you can allow instances to pass health checks with a lower status, such as Warning, by modifying the Healthy threshold option. If your deployments fail because they don't pass health checks and you need to force an update regardless of health status, specify the Ignore health check option.
When you specify a batch size for rolling updates, Elastic Beanstalk also uses that value for rolling application restarts. Use rolling restarts when you need to restart the proxy and application servers running on your environment's instances without downtime.
AWS Elastic Beanstalk provides support for running Amazon Relational Database Service (Amazon RDS) instances in your Elastic Beanstalk environment. This works great for development and testing environments. However, it isn't ideal for a production environment because it ties the lifecycle of the database instance to the lifecycle of your application's environment.
Hence, the correct answer is: Host the online customer portal using AWS Elastic Beanstalk and integrate it to an external Amazon RDS MySQL database in Multi-AZ deployments configuration. Use immutable updates for application deployments.
The option that says: Host the online customer portal using AWS Elastic Beanstalk and integrate it to an external Amazon RDS MySQL database in Multi-AZ deployments configuration. Configure the Elastic Beanstalk to use blue/green deployment for releasing the new application version to a new environment. Swap the CNAME in the two environments to redirect traffic to the new version using the Swap Environment URLs feature. Once the deployment has been successfully implemented, keep the old environment running as a backup is incorrect because although using the blue/green deployment configuration is an ideal option, keeping the old environment running is not recommended since it entails a significant cost. Take note that the scenario asks for the most cost-effective solution, which is why the old environment should be deleted.
The option that says: Host the online customer portal using AWS Elastic Beanstalk coupled with an Amazon RDS MySQL database. In the Elastic Beanstalk database configuration, set the Availability option to High (Multi-AZ) to run a warm backup in a second Availability Zone. Use the All at once deployment policy to release the new application version is incorrect because this will deploy the new version to all existing instances and will not create new EC2 instances. Moreover, you should decouple your RDS database from Elastic Beanstalk as this is tied to the lifecycle of the database instance and to the lifecycle of your application's environment.
The option that says: Host the online customer portal using AWS Elastic Beanstalk coupled with Amazon RDS MySQL database as part of the environment. For high availability, set the Availability option to High (Multi-AZ) in the Elastic Beanstalk database configuration to run a warm backup in a second Availability Zone. Use the Rolling with additional batch policy for application deployments is incorrect because this type of configuration could potentially cause partially completed rolling deployments. The new batch of instances is within the same Auto Scaling group and not in a new one. The rollback process is also cumbersome to implement unlike the Immutable or Blue/Green deployment.
References:
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/AWSHowTo.RDS.html
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.rolling-version-deploy.html
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/environmentmgmt-updates-immutable.html
Check out this AWS Elastic Beanstalk Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-elastic-beanstalk/

Host the online customer portal using AWS Elastic Beanstalk coupled with Amazon RDS MySQL database as part of the environment. For high availability, set the Availability option to High (Multi-AZ) in the Elastic Beanstalk database configuration to run a warm backup in a second Availability Zone. Use the Rolling with additional batch policy for application deployments.

Host the online customer portal using AWS Elastic Beanstalk coupled with an Amazon RDS MySQL database. In the Elastic Beanstalk database configuration, set the Availability option to High (Multi-AZ) to run a warm backup in a second Availability Zone. Use the All at once deployment policy to release the new application version.

Q44) Your API servers are hosted on a cluster of EC2 instances deployed using Elastic Beanstalk. You need to deploy a new version and also change the instance type of the instances from m4.large to m4.xlarge. Since this is a production workload already operating at 80% capacity, you want to ensure that the workload on the current EC2 instances does not increase while doing the deployment.
Which of the following deployment policy will you implement which will incur the LEAST amount of cost?

Use Immutable as deployment policy to deploy the new version with complete capacity.

✅ Use Rolling with additional batch as deployment policy to maintain the capacity of the cluster during the deployment.

Explanation:-When a configuration change requires replacing instances, Elastic Beanstalk can perform the update in batches to avoid downtime while the change is propagated. During a rolling update, capacity is only reduced by the size of a single batch, which you can configure. With rolling deployments, Elastic Beanstalk splits the environment's EC2 instances into batches and deploys the new version of the application to one batch at a time, leaving the rest of the instances in the environment running the old version of the application. During a rolling deployment, some instances serve requests with the old version of the application, while instances in completed batches serve other requests with the new version.

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

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 in this scenario is: Use Rolling with additional batch as deployment policy to maintain the capacity of the cluster during the deployment.

The option that says: Use Rolling as the deployment policy to maintain the capacity of the cluster during the deployment is incorrect because this type will remove a batch of instances from the cluster while deploying the new instances. Hence, this will increase the load to the current EC2 instances. A better solution to implement in this scenario is to use Rolling with additional batch.

The option that says: Use All at once as deployment policy to deploy the new version with complete capacity is incorrect because this type will cause a brief downtime during deployment. Hence, this is not ideal for critical production applications.

The option that says: Use Immutable as deployment policy to deploy the new version with complete capacity is incorrect because although this type can maintain the compute capacity of the cluster, it entails a significant amount of cost since a new batch of instances will be launched.

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.rollingupdates.html
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/environmentmgmt-updates-immutable.html

Check out this AWS Elastic Beanstalk Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-elastic-beanstalk/

Use Rolling as the deployment policy to maintain the capacity of the cluster during the deployment.

Use All at once as deployment policy to deploy the new version with complete capacity.

Q45) A global IT consulting company has a multi-tier enterprise resource planning application which is hosted in AWS. It runs on an Auto Scaling group of EC2 instances across multiple Availability Zones behind an Application Load Balancer. For its database tier, all of its data is persisted in an Amazon RDS MySQL database running in a Multi-AZ deployments configuration. All of the static content of the application is durably stored in Amazon S3. The company is already using CloudFormation templates for managing and deploying your AWS resources. Few weeks ago, the company failed an IT audit due to their application's long recovery time and excessive data loss in a simulated disaster recovery scenario drill.
How should the DevOps Engineer implement a multi-region disaster recovery plan which has the LOWEST recovery time and the LEAST data loss?

Launch the application stack in another AWS region using the CloudFormation template. Create another Amazon RDS standby DB instance in the other region then enable cross-region replication between the original Amazon S3 bucket and a new S3 bucket. The Standby DB instance will automatically be the master DB in the event of an application fail over. Increase the capacity of the Auto Scaling group using the CloudFormation stack template to improve the scalability of the application.

Launch the application stack in another AWS region using the CloudFormation template. Take a daily Amazon RDS cross-region snapshot to the other region using a scheduled job running in Lambda and CloudWatch Events. Enable cross-region replication between the original S3 bucket and Amazon Glacier. In the event of application outages, launch a new application stack in the other AWS region and restore the database from the most recent snapshot.

Launch the application stack in another AWS region using the CloudFormation template. Enable cross-region replication between the original Amazon S3 bucket and a new S3 bucket. Set up an Application Load Balancer which will distribute the traffic to the other AWS region in the event of an outage. Maintain the Multi-AZ deployments configuration of the Amazon RDS database which can ensure the availability of your data even in the event of a regional AWS outage in the primary site.

✅ Launch the application stack in another AWS region using the CloudFormation template. Create an Amazon RDS Read Replica in the other region then enable cross-region replication between the original Amazon S3 bucket and a new S3 bucket. Promote the RDS Read Replica as the master in the event of application failover. Increase the capacity of the Auto Scaling group using the CloudFormation stack template to improve the scalability of the application.

Explanation:-Amazon RDS Read Replicas provide enhanced performance and durability for database (DB) instances. This feature makes it easy to elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads. You can create one or more replicas of a given source DB Instance and serve high-volume application read traffic from multiple copies of your data, thereby increasing aggregate read throughput. Read replicas can also be promoted when needed to become standalone DB instances. Read replicas are available in Amazon RDS for MySQL, MariaDB, PostgreSQL and Oracle as well as Amazon Aurora.

Read replicas in Amazon RDS for MySQL, MariaDB, PostgreSQL, and Oracle provide a complementary availability mechanism to Amazon RDS Multi-AZ Deployments. You can promote a read replica if the source DB instance fails. You can also replicate DB instances across AWS Regions as part of your disaster recovery strategy which is not available with Multi-AZ Deployments since this is only applicable in a single AWS Region. This functionality complements the synchronous replication, automatic failure detection, and failover provided with Multi-AZ deployments.

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 the application stack in another AWS region using the CloudFormation template. Create an Amazon RDS Read Replica in the other region then enable cross-region replication between the original Amazon S3 bucket and a new S3 bucket. Promote the RDS Read Replica as the master in the event of application failover. Increase the capacity of the Auto Scaling group using the CloudFormation stack template to improve the scalability of the application.
The option that says: Launch the application stack in another AWS region using the CloudFormation template. Take a daily Amazon RDS cross-region snapshot to the other region using a scheduled job running in Lambda and CloudWatch Events. Enable cross-region replication between the original S3 bucket and Amazon Glacier. In the event of application outages, launch a new application stack in the other AWS region and restore the database from the most recent snapshot is incorrect because although this solution may work, the use of cross-region snapshot doesn't provide the LOWEST recovery time and the LEAST data loss because a snapshot can take several hours to complete. The best solution to use here is to launch a Read Replica to another AWS Region, which asynchronously replicates the data from the source database to the other AWS Region.
The option that says: Launch the application stack in another AWS region using the CloudFormation template. Create another Amazon RDS Standby DB instance in the other region then enable cross-region replication between the original Amazon S3 bucket and a new S3 bucket. The Standby DB instance will automatically be the master DB in the event of application failover. Increase the capacity of the Auto Scaling group using the CloudFormation stack template to improve the scalability of the application is incorrect because the scope of the Multi-AZ deployments is bound to a single AWS Region only. You cannot host your standby DB instance to another AWS Region. You should either use Read Replicas or cross-region snapshots instead.
The option that says: Launch the application stack in another AWS region using the CloudFormation template. Enable cross-region replication between the original Amazon S3 bucket and a new S3 bucket. Set up an Application Load Balancer which will distribute the traffic to the other AWS region in the event of an outage. Maintain the Multi-AZ deployments configuration of the Amazon RDS database which can ensure the availability of your data even in the event of a regional AWS outage in the primary site is incorrect because an ELB can't distribute traffic to different AWS Regions, unlike Route 53. Moreover, a Multi-AZ deployments configuration can only handle an outage of one or more Availability Zones and not the entire AWS Region.
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/

Q46) You have created a mobile app that allows users to rate and review hotels they have stayed on. The app is hosted on Docker containers deployed on Amazon ECS. You need to run validation scripts for new deployments to determine if the application is working as expected before allowing production traffic to flow to the new app version. You also want to configure automatic rollback if the validation is not successful.
Which of the following options will you do to implement this validation?

○ Create your validation scripts in AWS Lambda and define them on the BeforeAllowTraffic lifecycle hook of the AppSpec.yaml file. The functions can validate the deployment before allowing production traffic and rollback if the tests fail.
○ Create your validation scripts in AWS Lambda and define them on the AfterAllowTraffic lifecycle hook of the AppSpec.yaml file. The functions can validate the deployment using production traffic and rollback if the tests fail.
✓ Create your validation scripts in AWS Lambda and define them on the AfterAllowTestTraffic lifecycle hook of the AppSpec.yaml file. The functions can validate the deployment using the test traffic and rollback if the tests fail.
Explanation:-You can use a Lambda function to validate part of the deployment of an updated Amazon ECS application. During an Amazon ECS deployment with validation tests, CodeDeploy can be configured to use a load balancer with two target groups: one production traffic listener and one test traffic listener. To add a validation test, you first implement the test in a Lambda function. Next, in your deployment AppSpec file, you specify the Lambda function for the lifecycle hook you want to test. If a validation test fails, the deployment stops, it is rolled back, and marked as failed. If the test succeeds, the deployment continues to the next deployment lifecycle event or hook.
The content in the 'hooks' section of the AppSpec file varies, depending on the compute platform for your deployment. The 'hooks' section for an EC2/On-Premises deployment contains mappings that link deployment lifecycle event hooks to one or more scripts. The 'hooks' section for a Lambda or an Amazon ECS deployment specifies Lambda validation functions to run during a deployment lifecycle event. If an event hook is not present, no operation is executed for that event.
When the deployment starts, the deployment lifecycle events start to execute one at a time. Some lifecycle events are hooks that only execute Lambda functions specified in the AppSpec file. An AWS Lambda hook is one Lambda function specified with a string on a new line after the name of the lifecycle event. On the AfterAllowTestTraffic hook, you can specify Lambda functions that can validate the deployment using the test traffic.
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. After the validation tests are complete, one of the following occurs:
-If validation fails and rollbacks are configured, the deployment status is marked Failed and components return to their state when the deployment started.
-If validation fails and rollbacks are not configured, the deployment status is marked Failed and components remain in their current state.
-If validation succeeds, the deployment continues to the BeforeAllowTraffic hook.
Hence, the correct answer is: Create your validation scripts in AWS Lambda and define them on the AfterAllowTestTraffic lifecycle hook of the AppSpec.yaml file. The functions can validate the deployment using the test traffic and rollback if the tests fail.
The option that says: Create your validation scripts in AWS Lambda and define them on the AfterInstall lifecycle hook of the AppSpec.yaml file. The functions can validate the deployment after installing the new version and rollback if the tests fail is incorrect because in the AfterInstall lifecycle hook, the new task version is not yet attached on the test listener and application load balancer, therefore, no traffic is flowing yet on this cluster.
The option that says: Create your validation scripts in AWS Lambda and define them on the BeforeAllowTraffic lifecycle hook of the AppSpec.yaml file. The functions can validate the deployment before allowing production traffic and rollback if the tests fail is incorrect because in the BeforeAllowTraffic lifecycle hook, validation has already succeeded which defeats its purpose. You have to use this hook to perform additional actions before allowing production traffic to flow to the new task version.
The option that says: Create your validation scripts in AWS Lambda and define them on the AfterAllowTraffic lifecycle hook of the AppSpec.yaml file. The functions can validate the deployment using production traffic and rollback if the tests fail is incorrect because in the AfterAllowTraffic lifecycle hook, production traffic is rerouted from the old task set to the new task set. You want to verify the application before opening it to production traffic, and not after.

References:
https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file-structure-hooks.html#appspec-hooks-ecs
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-steps.html#deployment-steps-what-happens
https://docs.aws.amazon.com/codedeploy/latest/userguide/tutorial-ecs-deployment-with-hooks.html
Check out this AWS CodeDeploy Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-codedeploy/
● Create your validation scripts in AWS Lambda and define them on the AfterInstall lifecycle hook of the AppSpec.yaml file. The functions can validate the deployment after installing the new version and rollback if the tests fail.

Q47) A leading media company has recently moved its .NET web application from its on-premises network to AWS Elastic Beanstalk for easier deployment and management. An Amazon S3 bucket is used to store static contents such as PDF files, images, videos, and the likes. An Amazon DynamoDB table stores all of the data of the application. The company has recently launched a series of global marketing campaigns that resulted in unpredictable spikes of incoming traffic. Upon checking, the Operations team discovered that over 80% of the traffic are just duplicate read requests.
As a DevOps Engineer, how can you improve the performance of the application for its users around the world?

● Cache the images stored in the S3 bucket using the AWS Elemental MediaStore. Set up a distributed cache layer using an ElastiCache for Redis Cluster to serve the repeated read requests on the web application.
● Cache the images stored in the S3 bucket using the AWS Elemental MediaPackage. Set up a distributed cache layer using an ElastiCache for Memcached Cluster to serve the repeated read requests on the web application.
● Use Lambda@Edge to cache the images stored in the S3 bucket. Use DynamoDB Accelerator (DAX) to cache repeated read requests on the web application.
✅ Create a CloudFront web distribution to cache images stored in the S3 bucket. Use DynamoDB Accelerator (DAX) to cache repeated read requests on the web application.
Explanation:-Amazon DynamoDB is designed for scale and performance. In most cases, the DynamoDB response times can be measured in single-digit milliseconds. However, there are certain use cases that require response times in microseconds. For these use cases, DynamoDB Accelerator (DAX) delivers fast response times for accessing eventually consistent data.
DAX is a DynamoDB-compatible caching service that enables you to benefit from fast in-memory performance for demanding applications. DAX addresses three core scenarios:
- As an in-memory cache, DAX reduces the response times of eventually consistent read workloads by an order of magnitude, from single-digit milliseconds to microseconds.
- DAX reduces operational and application complexity by providing a managed service that is API-compatible with DynamoDB. Therefore, it requires only minimal functional changes to use with an existing application.
- For read-heavy or bursty workloads, DAX provides increased throughput and potential operational cost savings by reducing the need to over-provision read capacity units. This is especially beneficial for applications that require repeated reads for individual keys.
Hence, the correct answer is: Create a CloudFront web distribution to cache images stored in the S3 bucket. Use DynamoDB Accelerator (DAX) to cache repeated read requests on the web application.
The option that says: Cache the images stored in the S3 bucket using the AWS Elemental MediaStore. Set up a distributed cache layer using an ElastiCache for Redis Cluster to serve the repeated read requests on the web application is incorrect because the AWS Elemental MediaStore is not a dedicated CDN service unlike CloudFront. AWS Elemental MediaStore is a video origination and storage service that offers the performance, consistency, and low latency required to deliver live video content combined with the security and durability Amazon offers across its services.
The option that says: Cache the images stored in the S3 bucket using the AWS Elemental MediaPackage. Set up a distributed cache layer using an ElastiCache for Memcached Cluster to serve the repeated read requests on the web application is incorrect because the AWS Elemental MediaPackage is primarily used for videos and not for photos. Moreover, CloudFront is a more suitable service to use as a Content Delivery Network. AWS Elemental MediaPackage is a video origination and just-in-time (JIT) packaging service that allows video providers to securely and reliably deliver live streaming content at scale.
The option that says: Use Lambda@Edge to cache the images stored in the S3 bucket. Use DynamoDB Accelerator (DAX) to cache repeated read requests on the web application is incorrect because Lambda@Edge is primarily used to run code closer to users of your application in order to improve application performance and reduce latency. Serving static content using Lambda@Edge is not a suitable use case.
References:
https://aws.amazon.com/dynamodb/dax/
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DAX.concepts.html
Check out this Amazon DynamoDB Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-amazon-dynamodb/

Q48) A dynamic Node.js-based photo sharing application hosted in four Amazon EC2 web servers is using a DynamoDB table for session management and an S3 bucket for storing media files. The users can upload, view, organize, and share their photos using the content management system of the application. When a user uploads an image, a Lambda function will be invoked to process the media file then store it in Amazon S3. Due to the recent growth of the application's user base in the country, they decided to manually add another six EC2 instances for the web tier to handle the peak load. However, each of the instances took more than half an hour to download the required application libraries and become fully configured.
Which of the following is the MOST resilient and highly available solution that will also lessen the deployment time of the new servers?

● Host the entire Node.js application to Amazon S3 as a static website. Create an Amazon CloudFront web distribution with the S3 bucket as its origin. Enable Auto Scaling in the Amazon DynamoDB table. In Route 53, point the application DNS record to the CloudFront URL.
● Deploy a Spot Fleet of EC2 instances with a target capacity of 20 then place them behind an Application Load Balancer. Configure Amazon Route 53 to point the application DNS record to the Application Load Balancer. Increase the RCU and WCU of the DynamoDB table.
✅ Host the entire application in Elastic Beanstalk. Create a custom AMI using AWS Systems Manager Automation which includes all of the required dependencies and web components. Configure the Elastic Beanstalk environment to have an Auto Scaling group of EC2 instances across multiple Availability Zones with a load balancer in front that balances the incoming traffic. Enable Amazon DynamoDB Auto Scaling and point the application DNS record to the Elastic Beanstalk load balancer using Amazon Route 53.
Explanation:-When you create an AWS Elastic Beanstalk environment, you can specify an Amazon Machine Image (AMI) to use instead of the standard Elastic Beanstalk AMI included in your platform version. A custom AMI can improve provisioning times when instances are launched in your environment if you need to install a lot of software that isn't included in the standard AMIs.
Using configuration files is great for configuring and customizing your environment quickly and consistently. Applying configurations, however, can start to take a long time during environment creation and updates. If you do a lot of server configuration in configuration files, you can reduce this time by making a custom AMI that already has the software and configuration that you need.
A custom AMI also allows you to make changes to low-level components, such as the Linux kernel, that are difficult to implement or take a long time to apply in configuration files. To create a custom AMI, launch an Elastic Beanstalk platform AMI in Amazon EC2, customize the software and

configuration to your needs, and then stop the instance and save an AMI from it.
Hence, the correct solution for this scenario is: Host the entire application in Elastic Beanstalk. Create a custom AMI using AWS Systems Manager Automation which includes all of the required dependencies and web components. Configure the Elastic Beanstalk environment to have an Auto Scaling group of EC2 instances across multiple Availability Zones with a load balancer in front that balances the incoming traffic. Enable Amazon DynamoDB Auto Scaling and point the application DNS record to the Elastic Beanstalk load balancer using Amazon Route 53.
The option that says: Set up your application to use AWS OpsWorks for deployment to automatically download the required libraries of each new EC2 instance once it is launched. Place the EC2 instances to an Auto Scaling group across multiple Availability Zones with an Application Load Balancer in front that balances the incoming traffic. Enable Amazon DynamoDB Auto Scaling and configure Amazon Route 53 to point the application DNS record to the Application Load Balancer is incorrect because this will still take time since you have to configure the instances one by one in OpsWorks instead of just using a custom AMI which already has the required dependencies.
The option that says: Deploy a Spot Fleet of EC2 instances with a target capacity of 20 then place them behind an Application Load Balancer. Configure Amazon Route 53 to point the application DNS record to the Application Load Balancer. Increase the RCU and WCU of the DynamoDB table is incorrect because using Spot Instances is susceptible to interruptions and could lead to outages of your application. Moreover, setting an exact number of target capacity is not recommended since your servers won't scale up or scale down based on the actual demand.
The option that says: Host the entire Node.js application to Amazon S3 as a static website. Create an Amazon CloudFront web distribution with the S3 bucket as its origin. Enable Auto Scaling in the Amazon DynamoDB table. In Route 53, point the application DNS record to the CloudFront URL is incorrect because the web application is a dynamic site and cannot be migrated to a static S3 website hosting.
References:
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.customenv.html
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create_deploy_nodejs.html
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/beanstalk-environment-configuration-advanced.html
Check out this AWS Elastic Beanstalk Cheat Sheet:
https://tutorialsdojo.com/aws-cheat-sheet-aws-elastic-beanstalk/
● Set up your application to use AWS OpsWorks for deployment to automatically download the required libraries of each new EC2 instance once it is launched. Place the EC2 instances to an Auto Scaling group across multiple Availability Zones with an Application Load Balancer in front that balances the incoming traffic. Enable Amazon DynamoDB Auto Scaling and configure Amazon Route 53 to point the application DNS record to the Application Load Balancer.

Q49) A government online portal that allows you to lodge your tax return is hosted in AWS. The portal uses a MEAN application stack with GraphQL API as its backend and a DynamoDB table as its data store. It is also utilizing several custom Chef recipes that are stored in a private Git repository. Since the portal has predictable peak traffic times, the company instructed its DevOps Engineer to configure their system to scale up the application instances only during the peak times. The deployment should perform rolling updates to the application environment with the least amount of management overhead for easy maintenance.
Which among the options below provides the MOST cost-effective solution?

● Migrate the application to Elastic Beanstalk. Configure the environment to use RollingWithAdditionalBatch as its deployment policy. This will launch an extra batch of instances first before starting the deployment in order to maintain full capacity. Attach the required IAM role that provides permission to the instances to access the DynamoDB table
● Create a new stack using AWS OpsWorks Stacks and push the custom recipes to an S3 bucket. Modify the configuration of the custom recipes to point to the Amazon S3 bucket. Add a new application layer for a standard Node.js application server. Configure the custom recipe to deploy the application using the S3 bucket. Set up load-based instances and attach an IAM role that provides permission to access the DynamoDB table.
● Migrate the application to Elastic Beanstalk. Configure the environment to use Rolling as its deployment policy. Attach the required IAM role that provides permission to the instances to access the DynamoDB table.
✅ Using AWS OpsWorks Stacks, create a new stack with a custom cookbook and upload the custom recipes to an S3 bucket. Modify the configuration of the custom recipes to point to the S3 bucket. Add a new application layer for a standard Node.js application server. Configure the custom recipe to deploy the application using the S3 bucket. Set up time-based instances and attach an IAM role that provides permission to access the DynamoDB table.
Explanation:-As your incoming traffic varies, your stack may have either too few instances to comfortably handle the load or more instances than necessary. You can save both time and money by using time-based or load-based instances to automatically increase or decrease a layer's instances so that you always have enough instances to adequately handle incoming traffic without paying for unneeded capacity. There's no need to monitor server loads or manually start or stop instances. In addition, time- and load-based instances automatically distribute, scale, and balance applications over multiple Availability Zones within a region, giving you geographic redundancy and scalability.
Automatic scaling is based on two instance types, which adjust a layer's online instances based on different criteria:
Time-based instances - They allow a stack to handle loads that follow a predictable pattern by including instances that run only at certain times or on certain days. For example, you could start some instances after 6PM to perform nightly backup tasks or stop some instances on weekends when traffic is lower.
Load-based instances - They allow a stack to handle variable loads by starting additional instances when traffic is high and stopping instances when traffic is low, based on any of several load metrics. For example, you can have AWS OpsWorks Stacks start instances when the average CPU utilization exceeds 80% and stop instances when the average CPU load falls below 60%.
Both time-based and load-based instances are supported for Linux stacks, while only time-based instances are supported for Windows stacks.
Unlike 24/7 instances, which you must start and stop manually, you do not start or stop time-based or load-based instances yourself. Instead, you configure the instances and AWS OpsWorks Stacks starts or stops them based on their configuration. For example, you configure time-based instances to start and stop on a specified schedule. AWS OpsWorks Stacks then starts and stops the instances according to that configuration.
Your custom cookbooks must be stored in an online repository, either an archive such as a .zip file or a source control manager such as Git. A stack can have only one custom cookbook repository, but the repository can contain any number of cookbooks. When you install or update the cookbooks, AWS OpsWorks Stacks installs the entire repository in a local cache on each of the stack's instances. When an instance needs, for example, to run one or more recipes, it uses the code from the local cache.
Hence, the correct answer is: Using AWS OpsWorks Stacks, create a new stack with a custom cookbook and upload the custom recipes to an S3 bucket. Modify the configuration of the custom recipes to point to the S3 bucket. Add a new application layer for a standard Node.js application server. Configure the custom recipe to deploy the application using the S3 bucket. Set up time-based instances and attach an IAM role that provides permission to access the DynamoDB table.
The option that says: Create a new stack using AWS OpsWorks Stacks and push the custom recipes to an S3 bucket. Modify the configuration of the custom recipes to point to the Amazon S3 bucket. Add a new application layer for a standard Node.js application server. Configure the custom recipe to deploy the application using the S3 bucket. Set up load-based instances and attach an IAM role that provides permission to access the DynamoDB table is incorrect because you have to use time-based instances instead. Remember that the scenario mentioned that the portal has predictable peak traffic times. Time-based instances allow a stack to handle loads that follow a predictable pattern by including instances that run only at certain times or on certain days.
The option that says: Migrate the application to Elastic Beanstalk. Configure the environment to use RollingWithAdditionalBatch as its deployment policy. This will launch an extra batch of instances first before starting the deployment in order to maintain full capacity. Attach the required IAM role

that provides permission to the instances to access the DynamoDB table is incorrect because although it is true that using RollingWithAdditionalBatch as your deployment policy will launch an extra batch of instances to maintain full capacity, the use of Elastic Beanstalk is not recommended since the architecture is already using several custom Chef recipes. A better solution would be to use AWS OpsWorks Stacks with time-based instances.

The option that says: Migrate the application to Elastic Beanstalk. Configure the environment to use Rolling as its deployment policy. Attach the required IAM role that provides permission to the instances to access the DynamoDB table is incorrect because, just as explained on the previous option, you should use AWS OpsWorks Stacks instead. With rolling deployments, Elastic Beanstalk splits the environment's instances into batches then deploys the new application version to one batch at a time, leaving the rest of the instances in the environment running the old application version.

References:

https://docs.aws.amazon.com/opsworks/latest/userguide/workingcookbook-installingcustom-enable.html#workingcookbook-installingcustom-enable-repo

https://docs.aws.amazon.com/opsworks/latest/userguide/workingcookbook.html

Check out this AWS OpsWorks Cheat Sheet:

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

Q50) A government agency is planning to launch a distributed system in AWS that processes thousands of transactions every day. The agency purchased a proprietary software with 100 licenses, which can be used by a maximum of 100 application servers. A DevOps Engineer needs to set up an automated solution that dynamically allocates the software licenses to the application servers. The Engineer also needs to provide a way to see the list of available licenses that are not in use.
Which of the following options below is the MOST suitable way to accomplish this task?

○ Prepare a CloudFormation template that uses an Auto Scaling group to launch the EC2 instances. Store the 100 license codes to AWS Systems Manager Parameter Store. Pull an available license from the Systems manager using the instance metadata script of the instance upon launch. Configure the metadata script to update the license mapping after the instance is terminated.

○ Prepare a CloudFormation template that uses an Auto Scaling group to launch the EC2 instances with an associated lifecycle hook for Instance Terminate. Store the 100 license codes to a public S3 bucket. Pull an available license from the bucket using the User Data script of the instance upon launch. Use the lifecycle hook to update the license mapping after the instance is terminated.

✓ Prepare a CloudFormation template that uses an Auto Scaling group to launch the EC2 instances with an associated lifecycle hook for Instance Terminate. Store the 100 license codes to a DynamoDB table. Pull an available license from the DynamoDB table using the User Data script of the instance upon launch. Use the lifecycle hook to update the license mapping after the instance is terminated.

Explanation:-You can use AWS CloudFormation to automatically install, configure, and start applications on Amazon EC2 instances. Doing so enables you to easily duplicate deployments and update existing installations without connecting directly to the instance, which can save you a lot of time and effort.

AWS CloudFormation includes a set of helper scripts (cfn-init, cfn-signal, cfn-get-metadata, and cfn-hup) that are based on cloud-init. You call these helper scripts from your AWS CloudFormation templates to install, configure, and update applications on Amazon EC2 instances that are in the same template.

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

Lifecycle hooks enable you to perform custom actions by pausing instances as an Auto Scaling group launches or terminates them. When an instance is paused, it remains in a wait state until either you complete the lifecycle action using the complete-lifecycle-action command or the CompleteLifecycleAction operation, or the timeout period ends (one hour by default).

For example, your newly launched instance completes its startup sequence and a lifecycle hook pauses the instance. While the instance is in a wait state, you can install or configure software on it, making sure that your instance is fully ready before it starts receiving traffic. For another example of the use of lifecycle hooks, when a scale-in event occurs, the terminating instance is first deregistered from the load balancer (if the Auto Scaling group is being used with Elastic Load Balancing). Then, a lifecycle hook pauses the instance before it is terminated. While the instance is in the wait state, you can, for example, connect to the instance and download logs or other data before the instance is fully terminated.

Hence, the correct answer is: Prepare a CloudFormation template that uses an Auto Scaling group to launch the EC2 instances with an associated lifecycle hook for Instance Terminate. Store the 100 license codes to a DynamoDB table. Pull an available license from the DynamoDB table using the User Data script of the instance upon launch. Use the lifecycle hook to update the license mapping after the instance is terminated.

The option that says: Prepare a CloudFormation template that uses an Auto Scaling group to launch the EC2 instances with an associated lifecycle hook for Instance Terminate. Store the 100 license codes to a public S3 bucket. Pull an available license from the bucket using the User Data script of the instance upon launch. Use the lifecycle hook to update the license mapping after the instance is terminated is incorrect because using a public S3 bucket to store the license codes is a security risk since it can be seen by anyone. You should use a DynamoDB table instead.

The option that says: Prepare a CloudFormation template that uses an Auto Scaling group to launch the EC2 instances with an associated lifecycle hook for Instance Terminate. Store the 100 license codes to AWS Certificate Manager (ACM). Pull an available license from ACM using the User Data script of the instance upon launch. Use the lifecycle hook to update the license mapping after the instance is terminated is incorrect because you cannot upload license codes to ACM. Take note that the AWS Certificate Manager is simply a service that lets you easily provision, manage, and deploy public and private Secure Sockets Layer/Transport Layer Security (SSL/TLS) certificates for use with AWS services and your internal connected resources.

The option that says: Prepare a CloudFormation template that uses an Auto Scaling group to launch the EC2 instances. Store the 100 license codes to AWS Systems Manager Parameter Store. Pull an available license from the Systems manager using the instance metadata script of the instance upon launch. Configure the metadata script to update the license mapping after the instance is terminated is incorrect because you should use a user data script instead of a metadata script to update the license mapping. In addition, you should set up a lifecycle hook for Instance Terminate in order to execute the mapping update before the instance is terminated.

References:

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-init.html

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

Check out this AWS CloudFormation Cheat Sheet:

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

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

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

○ Prepare a CloudFormation template that uses an Auto Scaling group to launch the EC2 instances with an associated lifecycle hook for Instance Terminate. Store the 100 license codes to AWS Certificate Manager (ACM). Pull an available license from ACM using the User Data script of the instance upon launch. Use the lifecycle hook to update the license mapping after the instance is terminated.
