You've been tasked with implementing an automated data backup solution for your application servers that run on Amazon EC2 with Amazon EBS volumes. You want to use a distributed data store for your backups to avoid single points of failure and to increase the durability of the data. Daily backups should be retained for 30 days so that you can restore data within an hour.

How can you implement this through a script that a scheduling daemon runs daily on the application servers?

A) Write the script to call the ec2-create-volume API, tag the Amazon EBS volume with the current date time group, and copy backup data to a second Amazon EBS volume. Use the ec2-describe-volumes API to enumerate existing backup volumes. Call the ec2-delete-volume API to prune backup volumes that are tagged with a date-time group older than 30 days.
B) Write the script to call the Amazon Glacier upload archive API, and tag the backup archive with the current date-time group. Use the list vaults API to enumerate existing backup archives. Call the delete vault API to prune backup archives that are tagged with a date-time group older than 30 days.
C) Write the script to call the ec2-create-snapshot API, and tag the Amazon EBS snapshot with the current date-time group. Use the ec2-describe-snapshots API to enumerate existing Amazon EBS snapshots. Call the ec2-delete-snapshot API to prune Amazon EBS snapshots that are tagged with a date-time group older than 30 days.
D) Write the script to call the ec2-create-volume API, tag the Amazon EBS volume with the current date-time group, and use the ec2-copy-snapshot API to back up data to the new Amazon EBS volume. Use the ec2-describe-snapshot API to enumerate existing backup volumes. Call the ec2-delete-snaphot API to prune backup Amazon EBS volumes that are tagged with a date-time group older than 30 days.

You work for a startup that has developed a new photo-sharing application for mobile devices. Over recent months your application has increased in popularity; this has resulted in a decrease in the performance of the application due to the increased load.

Your application has a two-tier architecture that is composed of an Auto Scaling PHP application tier and a MySQL RDS instance initially deployed with AWS CloudFormation. Your Auto Scaling group has a min value of 4 and a max value of 8. The desired capacity is now at 8 due to the high CPU utilization of the instances.

After some analysis, you are confident that the performance issues stem from a constraint in CPU capacity, while memory utilization remains low. You therefore decide to move from the general-purpose M3 instances to the compute-optimized C3 instances.

How would you deploy this change while minimizing any interruption to your end users?

A) Sign into the AWS Management Console, copy the old launch configuration, and create a new launch configuration that specifies the C3 instances. Update the Auto Scaling group with the new launch configuration. Auto Scaling will then update the instance type of all running instances
B) Sign into the AWS Management Console and update the existing launch configuration with the new C3 instance type. Add an UpdatePolicy attribute to your Auto Scaling group that specifies an AutoScaling RollingUpdate.
C) Update the launch configuration specified in the AWS CloudFormation template with the new C3 instance type. Run a stack update with the new template. Auto Scaling will then update the instances with the new instance type.

D) Update the launch configuration specified in the AWS CloudFormation template with the new C3 instance type. Also add an UpdatePolicy attribute to your Auto Scaling group that specifies an AutoScalingRollingUpdate. Run a stack update with the new template.

You have a complex system involving networking, IAM policies, and multiple, three-tier applications. You are still receiving requirements for the new system, so you don't yet know how many AWS components will be present in the final design.

You would like to start defining these AWS resources using AWS CloudFormation so that you can automate and version-control your infrastructure.

How would you use AWS CloudFormation to provide agile new environments for your customers in a cost-effective, reliable manner?

A) Create one single template by hand to encompass all resources that you need for the system, so you only have a single template to version-control.
B) Create multiple separate templates for each logical part of the system, create nested stacks in AWS CloudFormation and maintain several templates to version-control.
C) Create multiple separate templates for each logical part of the system, and provide the outputs from one to the next using an Amazon Elastic Compute Cloud (EC2) Instance running the SDK for finer granularity of control.
D) Manually construct the networking layer using Amazon Virtual Private Cloud (VPC) because this does not change often and then define all other ephemeral resources using AWS CloudFormation.

You have implemented a system to automate deployments of your configuration and application dynamically after an Amazon EC2 instance in an Auto Scaling group is launched. Your system uses a configuration management tool that works in a standalone configuration, where there is no master node. Due to the volatility of application load, new instances must be brought into service within three minutes of the launch of the instance operating system. The deployment stages take the following times to complete:

- **Installing configuration management agent: 2mins**
- **Configuring instance using artifacts: 4mins**
- **Installing application framework: 15mins**
- **Deploying application code: 1min**

What process should you use to automate the deployment using this type of standalone agent configuration?

A) Configure your Auto Scaling launch configuration with an Amazon EC2 UserData script to install the agent, pull configuration artifacts and application code from an Amazon S3 bucket, and then execute the agent to configure the infrastructure and application.
B) Build a custom Amazon Machine Image that includes all components pre-installed, including an agent, configuration artifacts, application frameworks, and code. Create a startup script that executes the agent to configure the system on startup.
C) Build a custom Amazon Machine Image that includes the configuration management agent and application framework pre-installed. Configure your Auto Scaling launch configuration with an Amazon EC2 UserData script to pull configuration artifacts and application code from an Amazon S3 bucket, and then execute the agent to configure the system.

D) Create a web service that polls the Amazon EC2 API to check for new instances that are launched in an Auto Scaling group. When it recognizes a new instance, execute a remote script via SSH to install the agent, SCP the configuration artifacts and application code, and finally execute the agent to configure the system.

As part of your continuous deployment process, your application undergoes an I/O load performance test before it is deployed to production using new AMIs. The application uses one Amazon EBS PIOPS volume per instance and requires consistent I/O performance.

Which of the following must be carried out to ensure that I/O load performance tests yield the correct results in a repeatable manner?

A) Ensure that the I/O block sizes for the test are randomly selected.
B) Ensure that the Amazon EBS volumes have been pre-warmed by reading all the blocks before the test.
C) Ensure that snapshots of the Amazon EBS volumes are created as a backup.
D) Ensure that the Amazon EBS volume is encrypted.
E) Ensure that the Amazon EBS volume has been pre-warmed by creating a snapshot of the volume before the test.

Your social media marketing application has a component written in Ruby running on AWS Elastic Beanstalk. This application component posts messages to social media sites in support of various marketing campaigns. Your management now requires you to record replies to these social media messages to analyze the effectiveness of the marketing campaign in comparison to past and future efforts. You've already developed a new application component to interface with the social media site APIs in order to read the replies.

Which process should you use to record the social media replies in a durable data store that can be accessed at any time for analytics of historical data?

A) Deploy the new application component in an Auto Scaling group of Amazon EC2 instances, read the data from the social media sites, store it with Amazon Elastic Block Store, and use AWS Data Pipeline to publish it to Amazon Kinesis for analytics.
B) Deploy the new application component as an Elastic Beanstalk application, read the data from the social media sites, store it in DynamoDB, and use Apache Hive with Amazon Elastic MapReduce for analytics.
C) Deploy the new application component in an Auto Scaling group of Amazon EC2 instances, read the data from the social media sites, store it in Amazon Glacier, and use AWS Data Pipeline to publish it to Amazon RedShift for analytics.
D) Deploy the new application component as an Amazon Elastic Beanstalk application, read the data from the social media site, store it with Amazon Elastic Block store, and use Amazon Kinesis to stream the data to Amazon CloudWatch for analytics.

After reviewing the last quarter's monthly bills, management has noticed an increase in the overall bill from Amazon. After researching this increase in cost, you discovered that one of your new services is doing a lot of GET Bucket API calls to Amazon S3 to build a metadata cache of all objects in the applications bucket. Your boss has asked you to come up with a new cost-effective way to help reduce the amount of these new GET Bucket API calls.

What process should you use to help mitigate the cost?

A) Update your Amazon S3 buckets' lifecycle policies to automatically push a list of objects to a new bucket, and use this list to view objects associated with the application's bucket.
B) Create a new DynamoDB table. Use the new DynamoDB table to store all metadata about all objects uploaded to Amazon S3. Any time a new object is uploaded, update the application's internal Amazon S3 object metadata cache from DynamoDB.
C) Using Amazon SNS, create a notification on any new Amazon S3 objects that automatically updates a new DynamoDB table to store all metadata about the new object. Subscribe the application to the Amazon SNS topic to update its internal Amazon S3 object metadata cache from the DynamoDB table.
D) Upload all images to Amazon SQS, set up SQS lifecycles to move all images to Amazon S3, and initiate an Amazon SNS notification to your application to update the application's internal Amazon S3 object metadata cache.
E) Upload all images to an ElastiCache filecache server. Update your application to now read all file metadata from the ElastiCache filecache server, and configure the ElastiCache policies to push all files to Amazon S3 for long-term storage.

Your team is responsible for an AWS Elastic Beanstalk application. The business requires that you move to a continuous deployment model, releasing updates to the application multiple times per day with zero downtime.

What should you do to enable this and still be able to roll back almost immediately in an emergency to the previous version?

A) Enable rolling updates in the Elastic Beanstalk environment, setting an appropriate pause time for application startup.
B) Create a second Elastic Beanstalk environment running the new application version, and swap the environment CNAMEs.
C) Develop the application to poll for a new application version in your code repository; download and install to each running Elastic Beanstalk instance.
D) Create a second Elastic Beanstalk environment with the new application version, and configure the old environment to redirect clients, using the HTTP 301 response code, to the new environment.

Your current log analysis application takes more than four hours to generate a report of the top 10 users of your web application. You have been asked to implement a system that can report this information in real time, ensure that the report is always up to date, and handle increases in the number of requests to your web application.

Choose the option that is cost-effective and can fulfill the requirements.

A) Publish your data to CloudWatch Logs, and configure your application to autoscale to handle the load on demand.
B) Publish your log data to an Amazon S3 bucket. Use AWS CloudFormation to create an Auto Scaling group to scale your post-processing application which is configured to pull down your log files stored in Amazon S3.
C) Post your log data to an Amazon Kinesis data stream, and subscribe your log-processing application so that is configured to process your logging data.
D) Configure an Auto Scaling group to increase the size of your Amazon EMR cluster.

E) Create a multi-AZ Amazon RDS MySQL cluster, post the logging data to MySQL, and run a map reduce job to retrieve the required information on user counts.

Your application has a single Amazon EC2 instance that processes orders with a third-party supplier. Orders are retrieved from an Amazon SQS queue and processed in batches every five minutes. There is a business requirement that delays in processing should be no more than one hour. Approximately three times a week, the application fails and orders stop being processed, requiring a manual restart.

Which steps should you take to make this more resilient in a cost-effective way? (Choose 2)

A) Create a second 'watchdog' instance configured to monitor the processing instance and restart it if a failure is detected.
B) Create an Auto Scaling launch configuration to launch instances configured to perform processing. Create an Auto Scaling group to use the launch configuration with a minimum and maximum of one.
C) Create an Auto Scaling launch configuration to launch instances configured to perform processing. Create an Auto Scaling group to use the launch configuration with a minimum of two and a maximum of ten, and to scale based on the size of the Amazon SQS queue.
D) Create a load balancer and register your instance with Elastic Load Balancing. Set the Elastic Load Balancing health check to call an HTTP endpoint in your application that executes the processing.
E) Modify the processing application to send a custom CloudWatch metric with a dimension of InstanceId. Create a CloudWatch alarm, configured when the metric is in an Insufficient state for 10 minutes, to take an Amazon EC2 action to terminate the instance.