1. **6. QUESTION**

A government agency plans to store confidential tax documents on AWS. Due to the sensitive information in the files, the Solutions Architect must restrict the data access requests made to the storage solution to a specific Amazon VPC only. The solution should also prevent the files from being deleted or overwritten to meet the regulatory requirement of having a write-once-read-many (WORM) storage model.

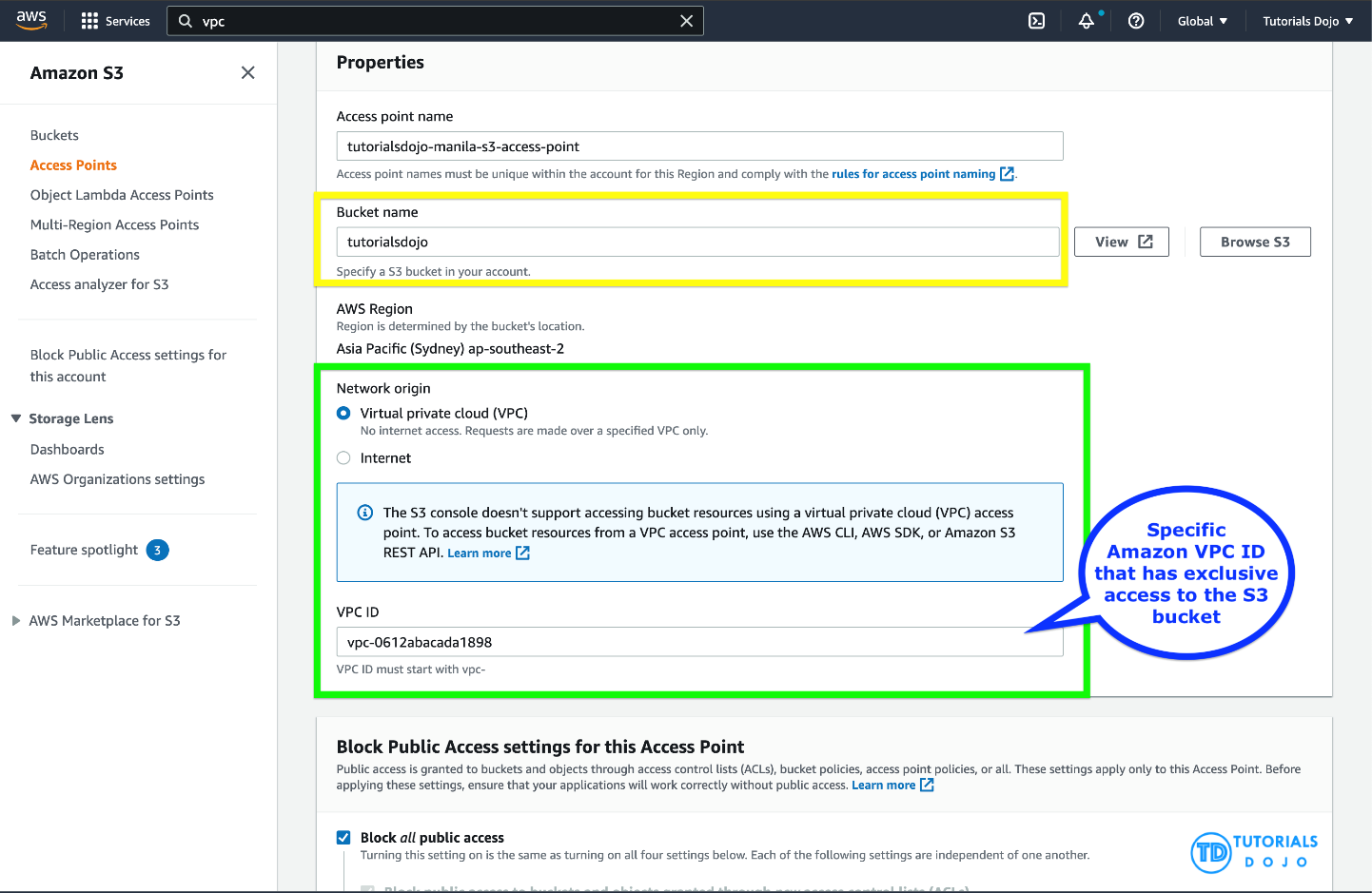
Which combination of the following options should the Architect implement? (Select TWO.)

* + **Configure an Amazon S3 Access Point for the S3 bucket to restrict data access to a particular Amazon VPC only.**
  + Store the tax documents in the Amazon S3 Glacier Instant Retrieval storage class to restrict fast data retrieval to a particular Amazon VPC of your choice.
  + **Enable Object Lock but disable Object Versioning on the new Amazon S3 bucket to comply with the write-once-read-many (WORM) storage model requirement.**
  + **Create a new Amazon S3 bucket with the S3 Object Lock feature enabled. Store the documents in the bucket and set the Legal Hold option for object retention.**
  + **Set up a new Amazon S3 bucket to store the tax documents and integrate it with AWS Network Firewall. Configure the Network Firewall to only accept data access requests from a specific Amazon VPC.**

**Incorrect**

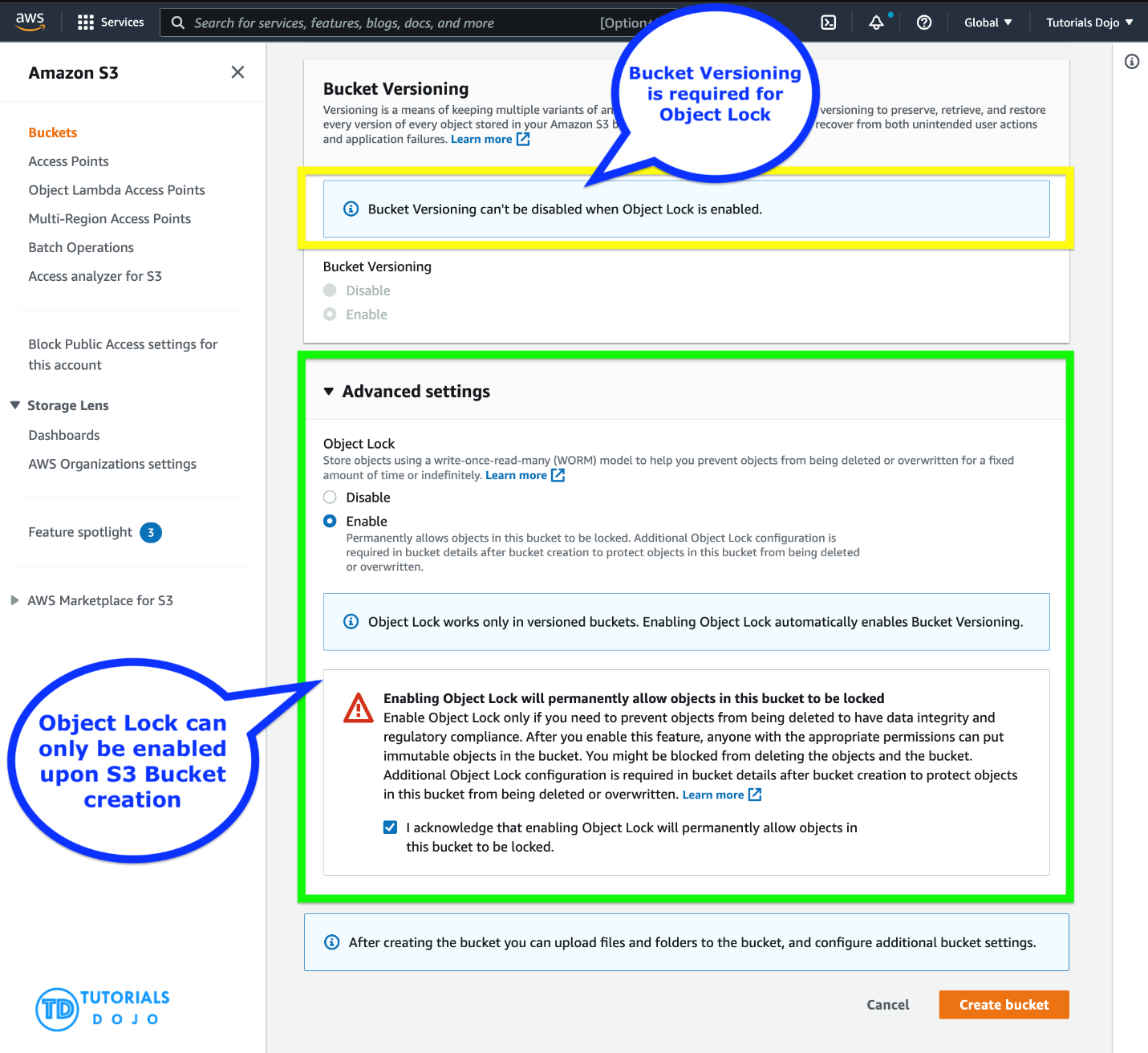
Amazon S3 access points simplify data access for any AWS service or customer application that stores data in S3. Access points are named network endpoints that are attached to buckets that you can use to perform S3 object operations, such as GetObject and PutObject.

Each access point has distinct permissions and network controls that S3 applies for any request that is made through that access point. Each access point enforces a customized access point policy that works in conjunction with the bucket policy that is attached to the underlying bucket. You can configure any access point to accept requests only from a virtual private cloud (VPC) to restrict Amazon S3 data access to a private network. You can also configure custom block public access settings for each access point.



You can also use Amazon S3 Multi-Region Access Points to provide a global endpoint that applications can use to fulfill requests from S3 buckets located in multiple AWS Regions. You can use Multi-Region Access Points to build multi-Region applications with the same simple architecture used in a single Region, and then run those applications anywhere in the world. Instead of sending requests over the congested public internet, Multi-Region Access Points provide built-in network resilience with acceleration of internet-based requests to Amazon S3. Application requests made to a Multi-Region Access Point global endpoint use AWS Global Accelerator to automatically route over the AWS global network to the S3 bucket with the lowest network latency.

With S3 Object Lock, you can store objects using a write-once-read-many (WORM) model. Object Lock can help prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely. You can use Object Lock to help meet regulatory requirements that require WORM storage, or to simply add another layer of protection against object changes and deletion.



Before you lock any objects, you have to enable a bucket to use S3 Object Lock. You enable Object Lock when you create a bucket. After you enable Object Lock on a bucket, you can lock objects in that bucket. When you create a bucket with Object Lock enabled, you can’t disable Object Lock or suspend versioning for that bucket.

Hence, the correct answers are:

**– Configure an Amazon S3 Access Point for the S3 bucket to restrict data access to a particular Amazon VPC only.**

**– Create a new Amazon S3 bucket with the S3 Object Lock feature enabled. Store the documents in the bucket and set the Legal Hold option for object retention.**

The option that says: **Set up a new Amazon S3 bucket to store the tax documents and integrate it with AWS Network Firewall. Configure the Network Firewall to only accept data access requests from a specific Amazon VPC** is incorrect because you cannot directly use an AWS Network Firewall to restrict S3 bucket data access requests to a specific Amazon VPC only. You have to use an Amazon S3 Access Point instead for this particular use case. An AWS Network Firewall is commonly integrated to your Amazon VPC and not to an S3 bucket.

The option that says: **Store the tax documents in the Amazon S3 Glacier Instant Retrieval storage class to restrict fast data retrieval to a particular Amazon VPC of your choice** is incorrect because Amazon S3 Glacier Instant Retrieval is just an archive storage class that delivers the lowest-cost storage for long-lived data that is rarely accessed and requires retrieval in milliseconds. It neither provides write-once-read-many (WORM) storage nor a fine-grained network control that restricts S3 bucket access to a specific Amazon VPC.

The option that says: **Enable Object Lock but disable Object Versioning on the new Amazon S3 bucket to comply with the write-once-read-many (WORM) storage model requirement** is incorrect. Although the Object Lock feature does provide write-once-read-many (WORM) storage, the Object Versioning feature must also be enabled too in order for this to work. In fact, you cannot manually disable the Object Versioning feature if you have already selected the Object Lock option.

**8. QUESTION**

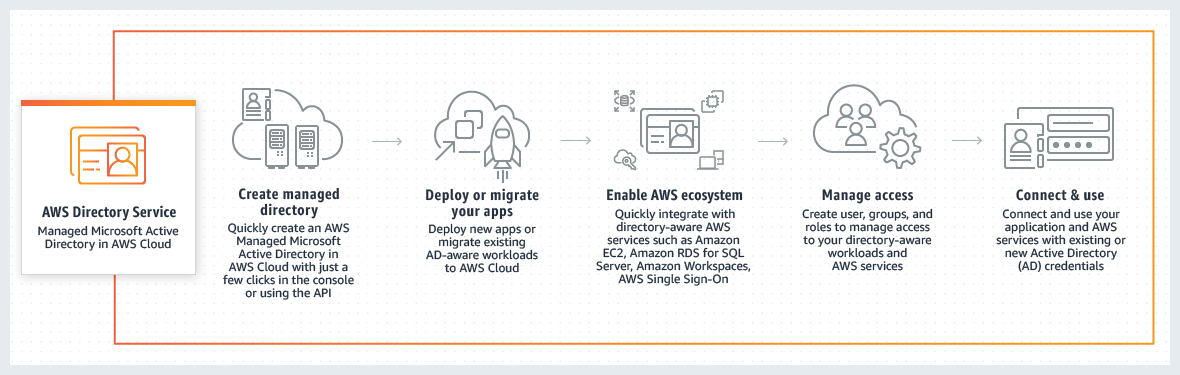
A telecommunications company is planning to give AWS Console access to developers. Company policy mandates the use of identity federation and role-based access control. Currently, the roles are already assigned using groups in the corporate Active Directory.

In this scenario, what combination of the following services can provide developers access to the AWS console? (Select TWO.)

* **AWS Directory Service AD Connector**
* **IAM Roles**
* AWS Directory Service Simple AD
* **IAM Groups**
* Lambda

**Incorrect**

Considering that the company is using a corporate Active Directory, it is best to use **AWS Directory Service AD Connector** for easier integration. In addition, since the roles are already assigned using groups in the corporate Active Directory, it would be better to also use **IAM Roles**. Take note that you can assign an IAM Role to the users or groups from your Active Directory once it is integrated with your VPC via the AWS Directory Service AD Connector.



**AWS Directory Service** provides multiple ways to use Amazon Cloud Directory and Microsoft Active Directory (AD) with other AWS services. Directories store information about users, groups, and devices, and administrators use them to manage access to information and resources. AWS Directory Service provides multiple directory choices for customers who want to use existing Microsoft AD or Lightweight Directory Access Protocol (LDAP)–aware applications in the cloud. It also offers those same choices to developers who need a directory to manage users, groups, devices, and access.

**AWS Directory Service Simple AD** is incorrect because this just provides a **subset** of the features offered by AWS Managed Microsoft AD, including the ability to manage user accounts and group memberships, create and apply group policies, securely connect to Amazon EC2 instances, and provide Kerberos-based single sign-on (SSO). In this scenario, the more suitable component to use is the AD Connector since it is a directory gateway with which you can redirect directory requests to your on-premises Microsoft Active Directory.

**IAM Groups** is incorrect because this is just a collection of *IAM* users. *Groups* let you specify permissions for multiple users, which can make it easier to manage the permissions for those users. In this scenario, the more suitable one to use is IAM Roles in order for permissions to create AWS Directory Service resources.

**Lambda** is incorrect because this is primarily used for serverless computing.

**9. QUESTION**

A company has a web application that uses Amazon CloudFront to distribute its images, videos, and other static contents stored in its S3 bucket to its users around the world. The company has recently introduced a new member-only access feature to some of its high-quality media files. There is a requirement to provide access to multiple private media files only to their paying subscribers without having to change their current URLs.

Which of the following is the most suitable solution that you should implement to satisfy this requirement?

* **Configure your CloudFront distribution to use Match Viewer as its Origin Protocol Policy which will automatically match the user request. This will allow access to the private content if the request is a paying member and deny it if it is not a member.**
* **Use Signed Cookies to control who can access the private files in your CloudFront distribution by modifying your application to determine whether a user should have access to your content. For members, send the required Set-Cookie headers to the viewer which will unlock the content only to them.**
* Create a Signed URL with a custom policy which only allows the members to see the private files.
* Configure your CloudFront distribution to use Field-Level Encryption to protect your private data and only allow access to members.

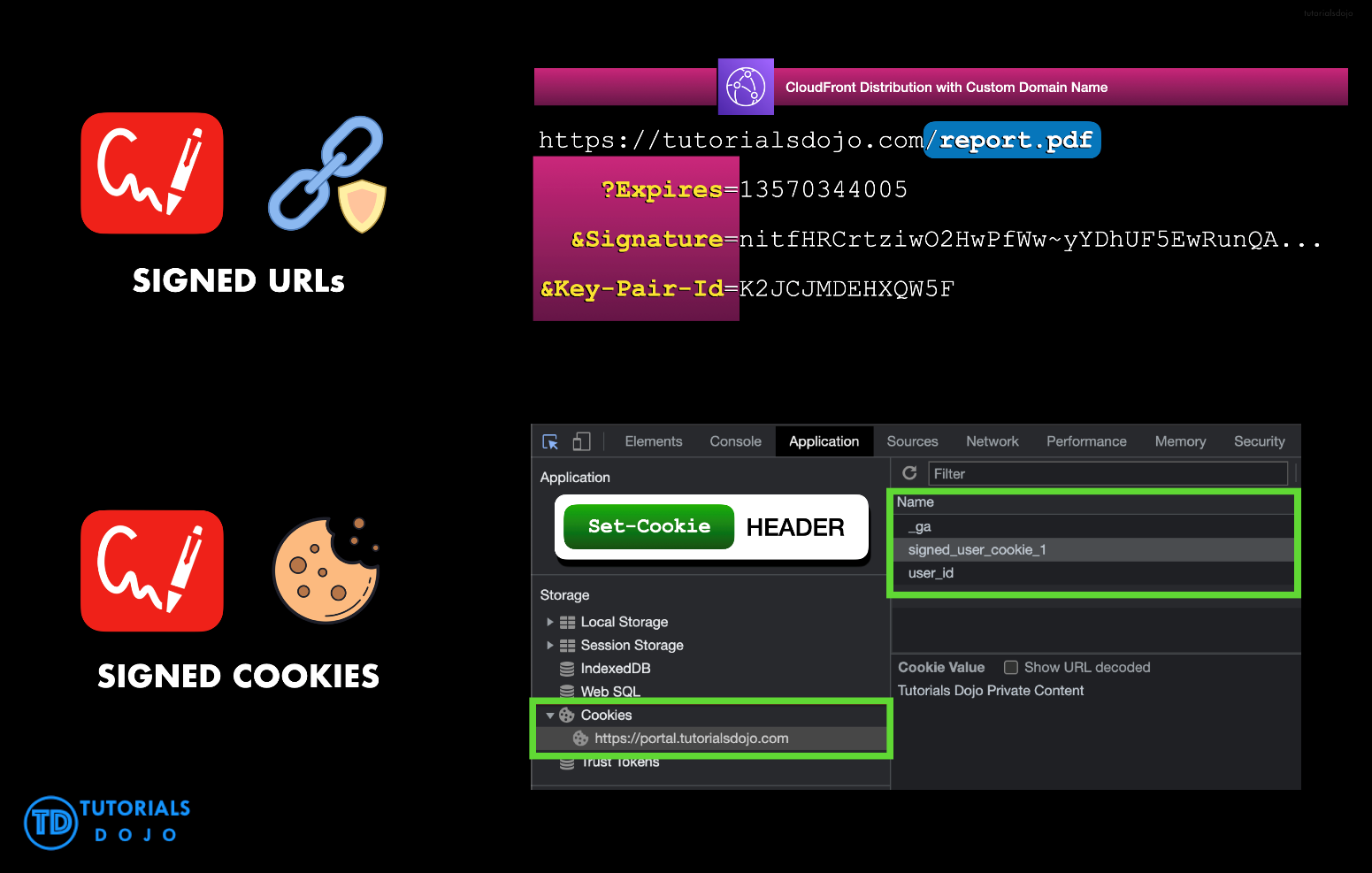
**Incorrect**

Many companies that distribute content over the internet want to restrict access to documents, business data, media streams, or content that is intended for selected users, for example, users who have paid a fee. To securely serve this private content by using CloudFront, you can do the following:

– Require that your users access your private content by using special CloudFront signed URLs or signed cookies.

– Require that your users access your content by using CloudFront URLs, not URLs that access content directly on the origin server (for example, Amazon S3 or a private HTTP server). Requiring CloudFront URLs isn’t necessary, but we recommend it to prevent users from bypassing the restrictions that you specify in signed URLs or signed cookies.

CloudFront signed URLs and signed cookies provide the same basic functionality: they allow you to control who can access your content.



If you want to serve private content through CloudFront and you’re trying to decide whether to use signed URLs or signed cookies, consider the following:

Use **signed URLs** for the following cases:

– You want to use an RTMP distribution. Signed cookies aren’t supported for RTMP distributions.

– You want to restrict access to individual files, for example, an installation download for your application.

– Your users are using a client (for example, a custom HTTP client) that doesn’t support cookies.

Use **signed cookies** for the following cases:

– You want to provide access to multiple restricted files, for example, all of the files for a video in HLS format or all of the files in the subscribers’ area of a website.

– You don’t want to change your current URLs.

Hence, the correct answer for this scenario is the option that says: **Use Signed Cookies to control who can access the private files in your CloudFront distribution by modifying your application to determine whether a user should have access to your content. For members, send the required Set-Cookie headers to the viewer which will unlock the content only to them.**

The option that says: **Configure your CloudFront distribution to use Match Viewer as its Origin Protocol Policy which will automatically match the user request. This will allow access to the private content if the request is a paying member and deny it if it is not a member** is incorrect because a Match Viewer is an Origin Protocol Policy that configures CloudFront to communicate with your origin using HTTP or HTTPS, depending on the protocol of the viewer request. CloudFront caches the object only once even if viewers make requests using both HTTP and HTTPS protocols.

The option that says: **Create a Signed URL with a custom policy which only allows the members to see the private files** is incorrect because Signed URLs are primarily used for providing access to individual files, as shown in the above explanation. In addition, the scenario explicitly says that they don’t want to change their current URLs which is why implementing Signed Cookies is more suitable than Signed URLs.

The option that says: **Configure your CloudFront distribution to use Field-Level Encryption to protect your private data and only allow access to members** is incorrect because Field-Level Encryption only allows you to securely upload user-submitted sensitive information to your web servers. It does not provide access to download multiple private files.

**10. QUESTION**

A popular social network is hosted in AWS and is using a DynamoDB table as its database. There is a requirement to implement a ‘follow’ feature where users can subscribe to certain updates made by a particular user and be notified via email.

Which of the following is the most suitable solution that you should implement to meet the requirement?

* **Enable DynamoDB Stream and create an AWS Lambda trigger, as well as the IAM role which contains all of the permissions that the Lambda function will need at runtime. The data from the stream record will be processed by the Lambda function which will then publish a message to SNS Topic that will notify the subscribers via email.**
* Using the Kinesis Client Library (KCL), write an application that leverages on DynamoDB Streams Kinesis Adapter that will fetch data from the DynamoDB Streams endpoint. When there are updates made by a particular user, notify the subscribers via email using SNS.
* Create a Lambda function that uses DynamoDB Streams Kinesis Adapter which will fetch data from the DynamoDB Streams endpoint. Set up an SNS Topic that will notify the subscribers via email when there is an update made by a particular user.
* **Set up a DAX cluster to access the source DynamoDB table. Create a new DynamoDB trigger and a Lambda function. For every update made in the user data, the trigger will send data to the Lambda function which will then notify the subscribers via email using SNS.**

**Incorrect**

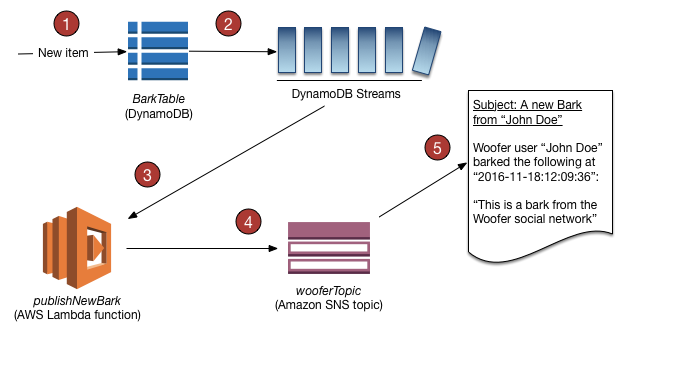
A **DynamoDB stream** is an ordered flow of information about changes to items in an Amazon DynamoDB table. When you enable a stream on a table, DynamoDB captures information about every modification to data items in the table.

Whenever an application creates, updates, or deletes items in the table, DynamoDB Streams writes a stream record with the primary key attribute(s) of the items that were modified. A *stream record*contains information about a data modification to a single item in a DynamoDB table. You can configure the stream so that the stream records capture additional information, such as the “before” and “after” images of modified items.

Amazon DynamoDB is integrated with AWS Lambda so that you can create *triggers*—pieces of code that automatically respond to events in DynamoDB Streams. With triggers, you can build applications that react to data modifications in DynamoDB tables.

If you enable DynamoDB Streams on a table, you can associate the stream ARN with a Lambda function that you write. Immediately after an item in the table is modified, a new record appears in the table’s stream. AWS Lambda polls the stream and invokes your Lambda function synchronously when it detects new stream records. The Lambda function can perform any actions you specify, such as sending a notification or initiating a workflow.

Hence, the correct answer in this scenario is the option that says: **Enable DynamoDB Stream and create an AWS Lambda trigger, as well as the IAM role which contains all of the permissions that the Lambda function will need at runtime. The data from the stream record will be processed by the Lambda function which will then publish a message to SNS Topic that will notify the subscribers via email**.



The option that says: **Using the Kinesis Client Library (KCL), write an application that leverages on DynamoDB Streams Kinesis Adapter that will fetch data from the DynamoDB Streams endpoint. When there are updates made by a particular user, notify the subscribers via email using SNS** is incorrect. Although this is a valid solution, it is missing a vital step which is to enable DynamoDB Streams. With the DynamoDB Streams Kinesis Adapter in place, you can begin developing applications via the KCL interface, with the API calls seamlessly directed at the DynamoDB Streams endpoint. Remember that the DynamoDB Stream feature is not enabled by default.

The option that says: **Create a Lambda function that uses DynamoDB Streams Kinesis Adapter which will fetch data from the DynamoDB Streams endpoint. Set up an SNS Topic that will notify the subscribers via email when there is an update made by a particular user** is incorrect because just like in the above, you have to manually enable DynamoDB Streams first before you can use its endpoint.

The option that says: **Set up a DAX cluster to access the source DynamoDB table. Create a new DynamoDB trigger and a Lambda function. For every update made in the user data, the trigger will send data to the Lambda function which will then notify the subscribers via email using SNS** is incorrect because the DynamoDB Accelerator (DAX) feature is primarily used to significantly improve the in-memory read performance of your database, and not to capture the time-ordered sequence of item-level modifications. You should use DynamoDB Streams in this scenario instead.

**11. QUESTION**

A company uses Amazon EC2 instances, Amazon RDS and, Amazon S3 to run its application. During a recent review of its infrastructure costs, the company noticed unusual spending patterns.

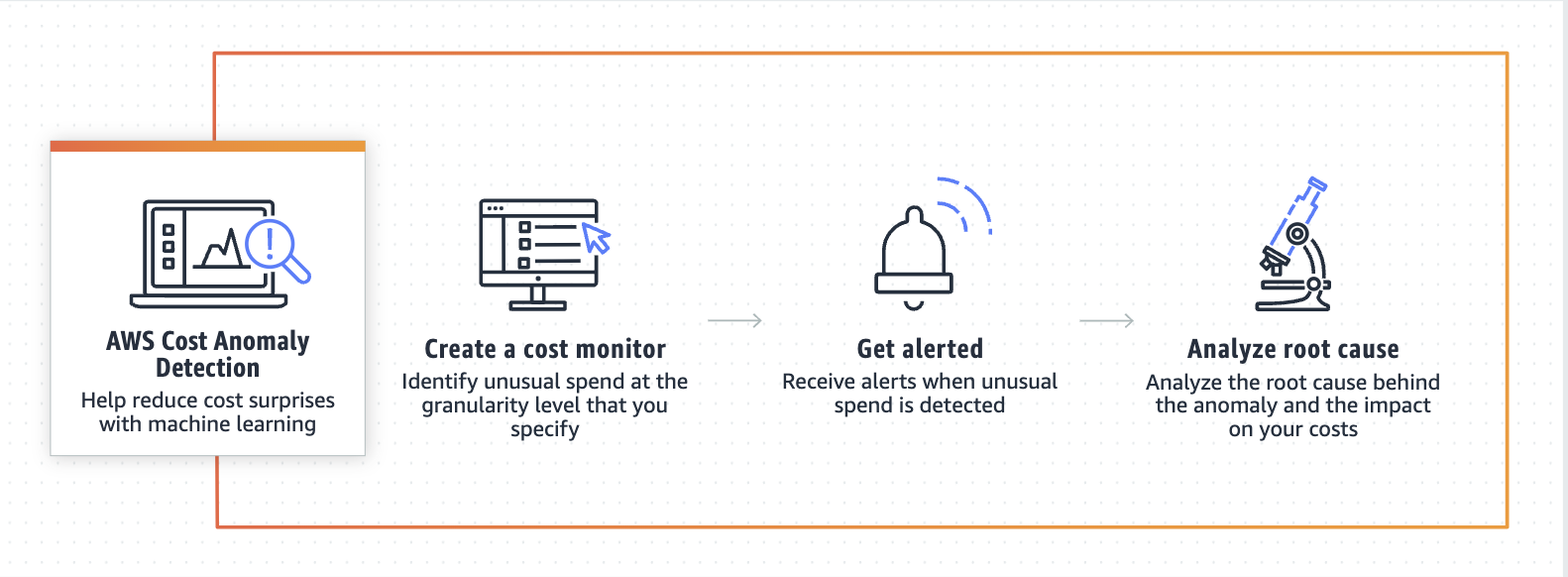
The company wants to monitor usage costs and send alerts to the appropriate departments when there is unusual spending from their workload.

Which option will meet these requirements?

* Use AWS Cost Explorer and enable multi-year data at monthly granularity.
* **In the AWS Billing and Cost Management console, create a cost monitor using AWS Cost Anomaly Detection.**
* Create a zero spend budget template in AWS Budgets.
* **Enable Amazon CloudWatch to monitor costs and detect unusual spending.**

**Incorrect**

**AWS Cost Anomaly Detection** is an AWS Cost Management feature. This feature uses machine learning models to detect and alert on anomalous spend patterns in your deployed AWS services.



Using AWS Cost Anomaly Detection includes the following benefits:

– receive alerts individually in aggregated reports either in an email message or an Amazon SNS topic.  
– evaluate spending patterns using machine learning methods to minimize false positive alerts. For example, you can evaluate weekly or monthly seasonality and natural growth.  
– investigate the root cause of the anomaly, such as the AWS account, service, Region, or usage type that’s driving the cost increase.  
– configure how to evaluate your costs. Choose whether you want to analyze all of your AWS services independently or analyze specific member accounts, cost allocation tags, or cost categories.

Hence, the correct answer is: **In the AWS Billing and Cost Management console, create a cost monitor using AWS Cost Anomaly Detection.**

The option that says: **Create a zero spend budget template in AWS Budgets** is incorrect because the zero spend budget template merely gives you an alert when your spending exceeds AWS Free Tier limits.

The option that says: **Use AWS Cost Explorer and enable multi-year data at monthly granularity** is incorrect. This solution would help in analyzing spending over time, but it won’t actively alert you to anomalies or unusual spending patterns.

The option that says: **Enable Amazon CloudWatch to monitor costs and detect unusual spending**is incorrect. Although the Amazon Cloudwatch can be configured to alert based on the EstimateCharges metric, it has no capability of determining whether breaches of a set limit are due to unusual spending or normal/expected increases.

##### **7. QUESTION**

A company seeks to rearchitect its subsystem to an event-driven design using AWS Lambda. However, the company has some hesitations, being that their workloads are all container-based Java services. The company wants to minimize cold starts and outlier latencies when serving requests in the most cost-effective manner.

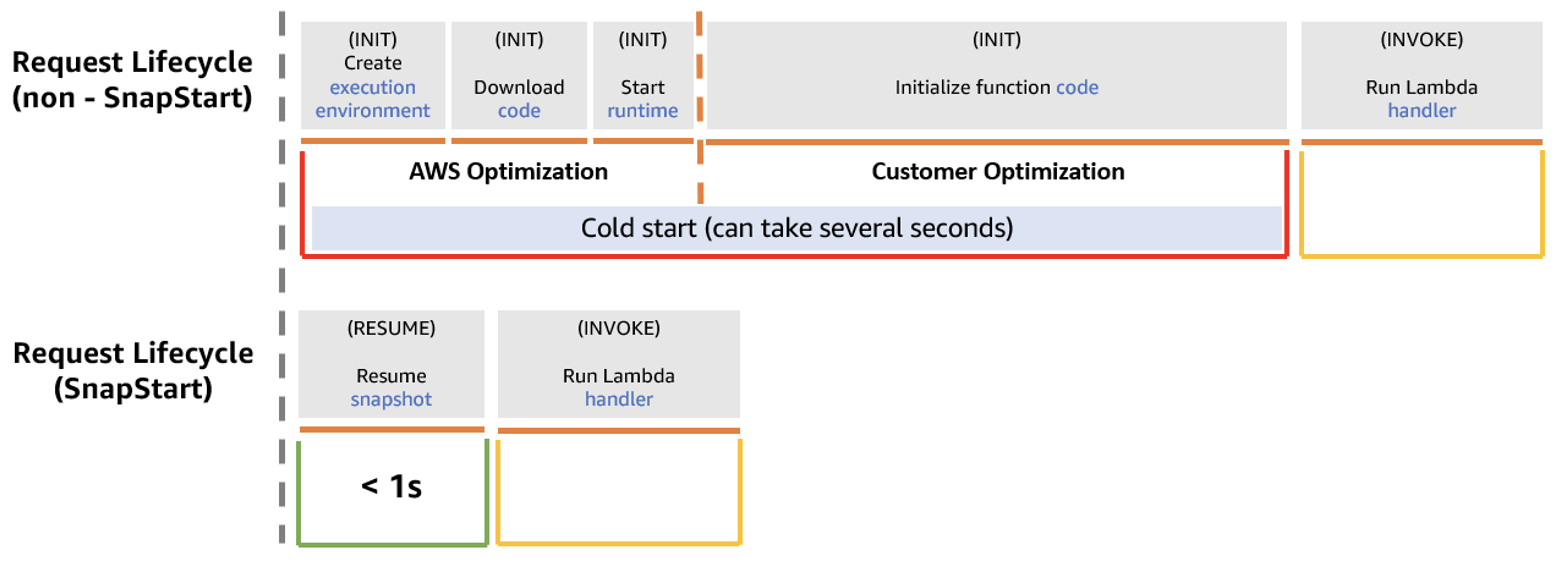
Which would meet the requirements?

* **Enable Lambda SnapStart.**
* Set up Lambda layers for dependencies.
* Configure response streaming for Lambda functions.
* **Enable Lambda provisioned concurrency.**

**Incorrect**

**AWS Lambda** is a compute service that lets you run code without provisioning or managing servers.

Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling and logging. With Lambda, all you need to do is supply your code in one of the language runtimes that Lambda supports.

  
With **Lambda SnapStart** for Java, Lambda initializes functions as new versions are published. Lambda then takes a Firecracker microVM snapshot of the memory and disk state of the initialized execution environment, encrypts the snapshot, and caches it for low-latency access.

In this scenario, leveraging the cached initialized environment in Lambda SnapStart is, in a way, cheating cold starts.

Hence, the correct answer is: **Enable Lambda SnapStart**.

The option that says: **Enable Lambda provisioned concurrency** is incorrect. On the aspect of cold start times alone, this may be a viable option. However, running with provisioned concurrency incurs overhead costs that may not be warranted during periods when the function experiences minimal to zero invocations.

The option that says: **Configure response streaming for Lambda functions** is incorrect. Streaming responses should improve overall time-to-first-byte latencies. However, in cases where startup time drowns all other latencies, as in the case of Java runtimes, this solution is insufficient.

The option that says: **Set up Lambda layers for dependencies** is incorrect. Using Lambda layers may yield some startup time deduction, but it is more commonly utilized for build/space optimization or dependency reuse.