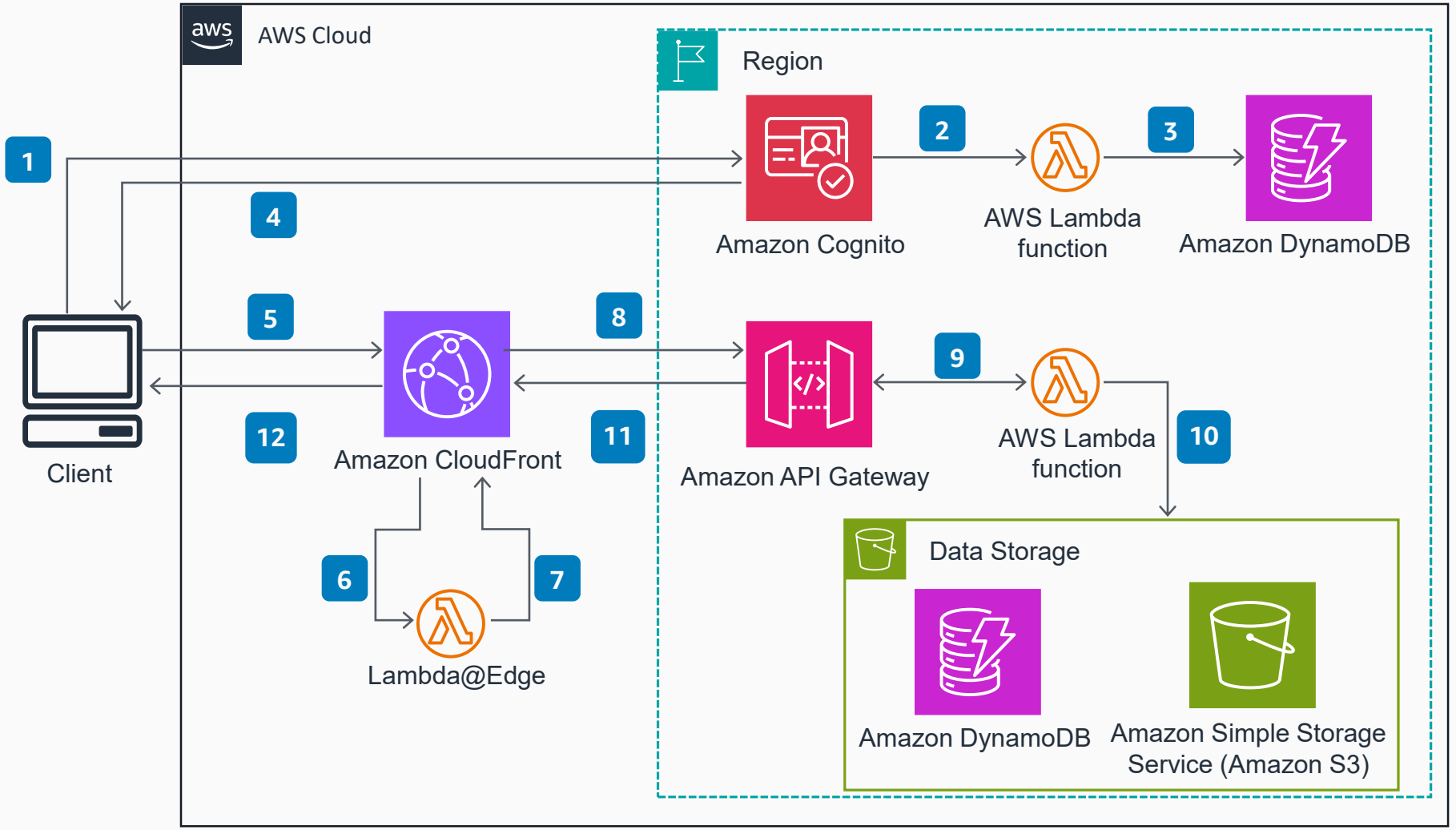
Guidance for Moving Your Paywall to the Edge on AWS

This architecture diagram demonstrates how to serve content and implement paywall logic at the edge with a Lambda@Edge feature of Amazon CloudFront.



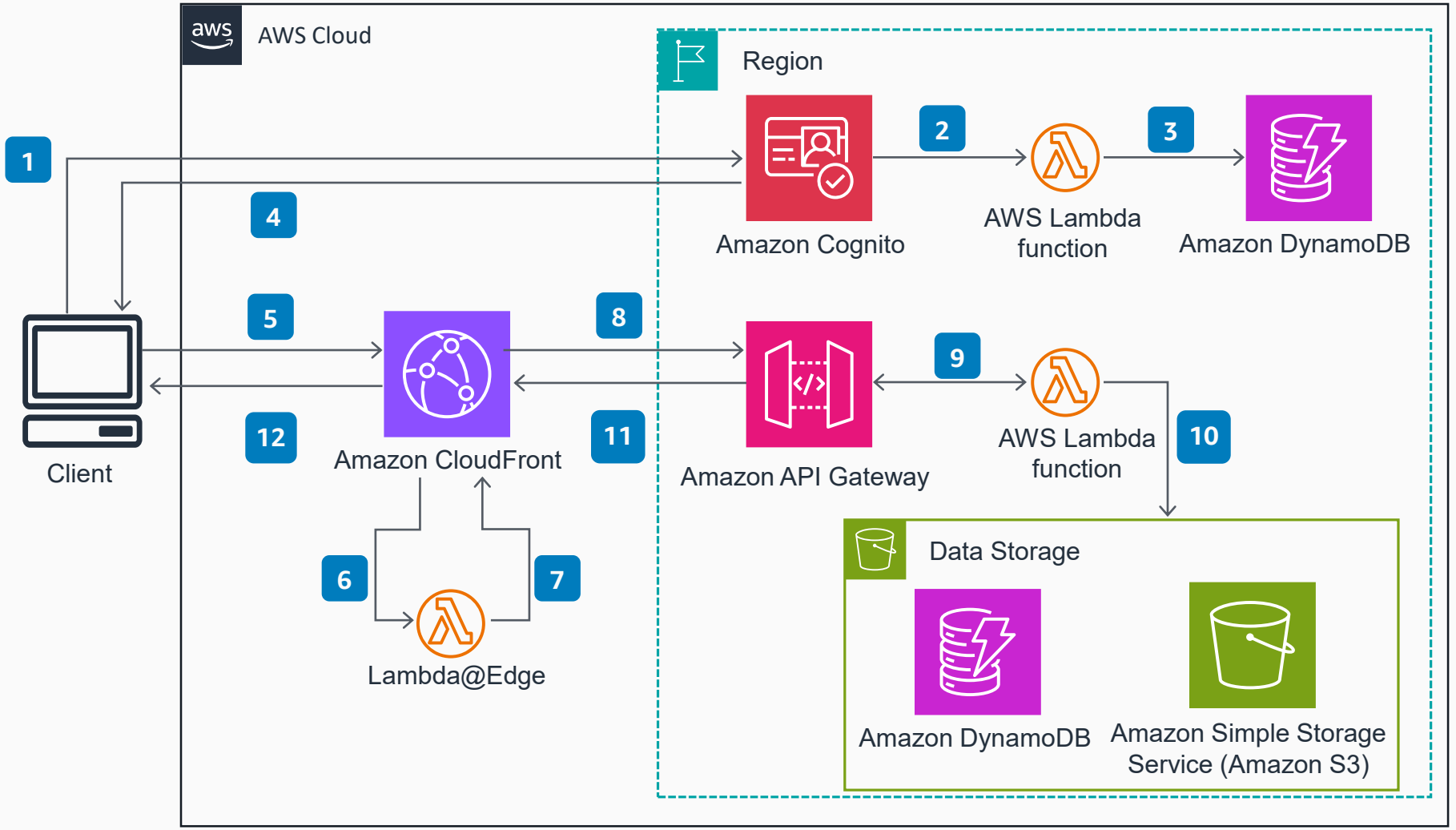
1. Client authenticates using an Amazon Cognito user pool (or another identity provider).
2. During authentication, the pre-token generation process for **Cognito** invokes an AWS Lambda function.
3. **Lambda** function looks up the user's subscription information in Amazon DynamoDB and adds it to the JSON Web Token (JWT) as a custom claim.
4. **Cognito** returns JWT to the Client, which stores it in a cookie to authorize content requests.
5. Client initiates a request for content, such as a news article, served by Amazon CloudFront. To authorize the request, the client includes the JWT in a cookie.
6. **CloudFront** invokes a Lambda@Edge function to update the viewer request headers based on whether the user is authorized to view the content.
7. Lambda@Edge validates the JWT and adds a custom header to the request indicating whether user has access to the content based on subscription's data in the JWT.
8. **CloudFront** creates a cache key based on the custom header added in Step 7. If content is not found in cache, it sends an origin request to Amazon API Gateway; otherwise, it skips to Step 12.

Reviewed for technical accuracy August 18, 2023

AWS Reference Architecture

Guidance for Moving Your Paywall to the Edge on AWS

This architecture diagram demonstrates how to serve content and implement paywall logic at the edge with a Lambda@Edge feature of Amazon CloudFront.



9 **API Gateway** invokes a **Lambda** function to retrieve content.

10 **Lambda** function examines the request header, added in Step 7, to determine if the user has a subscription. If the user has a subscription, it retrieves content from Amazon Simple Storage Service (Amazon S3) or **DynamoDB** based on an identity provided in the request URL. If the user does not have a subscription, the function returns a message that the user is not authorized to view the content.

11 **API Gateway** returns the response generated by the **Lambda** function, which will either be the full content or a message saying that the user is not authorized to access the item.

12 After serving content to the client, **CloudFront** caches it using a key that includes the custom header added in Step 7, thus enabling different versions of the content to be cached for subscribers and non-subscribers.

Reviewed for technical accuracy August 18, 2023

AWS Reference Architecture