DESIGN.md - AWS Application Load Balancer (ALB) Module Design

This document provides an overview of the design principles, usage tips, and best practices adhered to by the AWS Application Load Balancer (ALB) Terraform module.

1. Module Overview

This Terraform module is designed to provision and configure an AWS Application Load Balancer (ALB) as a reusable component. It encapsulates the creation of the ALB itself, its associated security group, HTTP and HTTPS listeners, and optional access logging to an S3 bucket. Furthermore, it integrates with AWS Systems Manager Parameter Store to publish key ALB attributes, facilitating discovery and consumption by other infrastructure components or applications. The module aims to provide a flexible, secure, and production-ready ALB deployment.

2. Usage Tips

- **Environment-Specific Naming:** Leverage the `target_environment` variable to ensure unique and descriptive naming for ALBs across different development, staging, and production environments.
- **Subnet Selection:** Always provide at least two subnets in different Availability Zones for high availability, as enforced by the `subnet_ids` variable validation.
- **Certificate Management:** For HTTPS listeners, ensure the `https_tls_cert_arn` variable points to a valid ACM certificate in the same region as the ALB. Consider using a separate module or data source to manage/retrieve ACM certificates.
- **Security Group Ingress/Egress:**
  - **Ingress:** Be specific with `http_ingress_cidrs`, `http_ingress_sg_ids`, `https_ingress_cidrs`, and `https_ingress_sg_ids`. For internal ALBs, restrict access to internal networks or specific security groups. For internet-facing ALBs, consider limiting to known IP ranges if possible, or `0.0.0.0/0` with caution.
  - **Egress:** The module defaults to allowing all outbound traffic if no specific egress rules are provided. For stricter security, define `http_egress_cidrs`, `http_egress_sg_ids`, `https_egress_cidrs`, or `https_egress_sg_ids` to restrict outbound connections to only necessary backend services.
- **Access Logging:** It is highly recommended to enable `enable_access_logs` in production environments for auditing, troubleshooting, and security analysis. Configure `access_logs_s3_bucket_lifecycle_enabled` and related variables (`transition_days`, `expiration_days`) to manage log retention and cost effectively.
- **SSM Parameter Store:** Utilize the SSM parameters exported by this module in other Terraform modules or applications to dynamically retrieve ALB details (e.g., DNS name for Route 53 records, ARN for target group attachments).
- **Tagging:** Implement a consistent tagging strategy using the `tags` variable to categorize resources for cost allocation, operational management, and compliance.

3. Alignment with Best Practices

3.1 General Best Practices

- **Modularity:** The module is self-contained and focuses on a single core responsibility (ALB deployment), making it reusable across various projects and environments.
- **Clarity and Readability:** Resources, variables, and outputs are clearly named and include descriptions, enhancing understanding for anyone reviewing the code.
- **Reusability:** Designed with variables to allow customization for different use cases without modifying the core module code.

3.2 Security Best Practices

- **Principle of Least Privilege:**
  - **Security Group Ingress:** Allows granular control over incoming traffic via CIDR blocks and security group IDs, enabling users to restrict access to only necessary sources.
  - **Security Group Egress:** Provides options to restrict outbound traffic from the ALB, preventing unauthorized connections from the load balancer itself.

  - **S3 Bucket Policy:** The S3 bucket policy for access logs is strictly scoped to allow only the AWS ELB service account to `PutObject` on the designated log bucket and prefix, adhering to least privilege. The `s3:GetBucketAcl` permission is included as it is sometimes required by AWS services to verify bucket permissions before writing, though `s3:PutObject` is the primary action for log delivery. Its presence does not contradict modern practices, especially when coupled with `BucketOwnerPreferred` ownership controls, as it ensures proper object ownership.

- **Access Logging:** Supports enabling detailed ALB access logs, which are crucial for security auditing, incident response, and performance monitoring.
- **Encryption at Rest (S3):** Automatically enables server-side encryption (SSE-256) for the S3 access logs bucket, ensuring logs are encrypted at rest.
- **Secure Defaults:** The `internal_alb` defaults to `true`, promoting internal-facing ALBs by default for internal applications, which is generally more secure. HTTPS is also enabled by default.

3.3 Terraform Best Practices

- **Explicit Provider Versioning:** The `versions.tf` file explicitly defines required Terraform and AWS provider versions, ensuring consistent deployments and preventing unexpected breaking changes from new provider versions.
- **Clear Variable Definitions:** All input variables are well-documented with descriptions, types, and default values, making the module easy to understand and use.
- **Local Values:** Uses `locals` to define derived values (e.g., `shared_alb_name`), promoting DRY (Don't Repeat Yourself) principles and improving readability.
- **Outputs:** Clearly defined outputs expose essential ALB attributes, allowing other modules or root configurations to easily consume information about the deployed ALB.
- **Conditional Resource Creation:** Employs `count` and `dynamic` blocks (e.g., for HTTP/HTTPS listeners, access logs, S3 lifecycle rules, security group rules) to conditionally create or configure resources based on input variables, reducing complexity and improving flexibility.
- **Validation Rules:** Includes validation for critical inputs like `subnet_ids` and `https_tls_cert_arn` to ensure valid configurations are provided, preventing common deployment errors.
- **Explicit Dependencies:** Uses `depends_on` for resources like S3 bucket policy and ownership controls to ensure correct creation order and prevent race conditions.

3.4 AWS Best Practices

- **High Availability:** Requires at least two subnets for ALB deployment, aligning with AWS recommendations for high availability across multiple Availability Zones.
- **Service-Specific S3 Permissions:** Correctly utilizes the `aws_elb_service_account` data source to grant the necessary permissions for the ALB service to write logs to the S3 bucket, following AWS's prescribed method.
- **SSM Parameter Store for Discovery:** Leverages AWS Systems Manager Parameter Store as a centralized, secure location to store and retrieve ALB metadata, facilitating integration with other AWS services or applications.
- **S3 Lifecycle Management:** Supports S3 lifecycle rules for access logs, enabling automated cost optimization by transitioning older logs to cheaper storage classes (e.g., Standard-IA) and eventually expiring them.
- **S3 Object Ownership:** Configures S3 bucket ownership controls (`BucketOwnerPreferred`) to ensure that objects written by the ALB log delivery service are owned by the bucket owner, which is a best practice for cross-account log delivery.