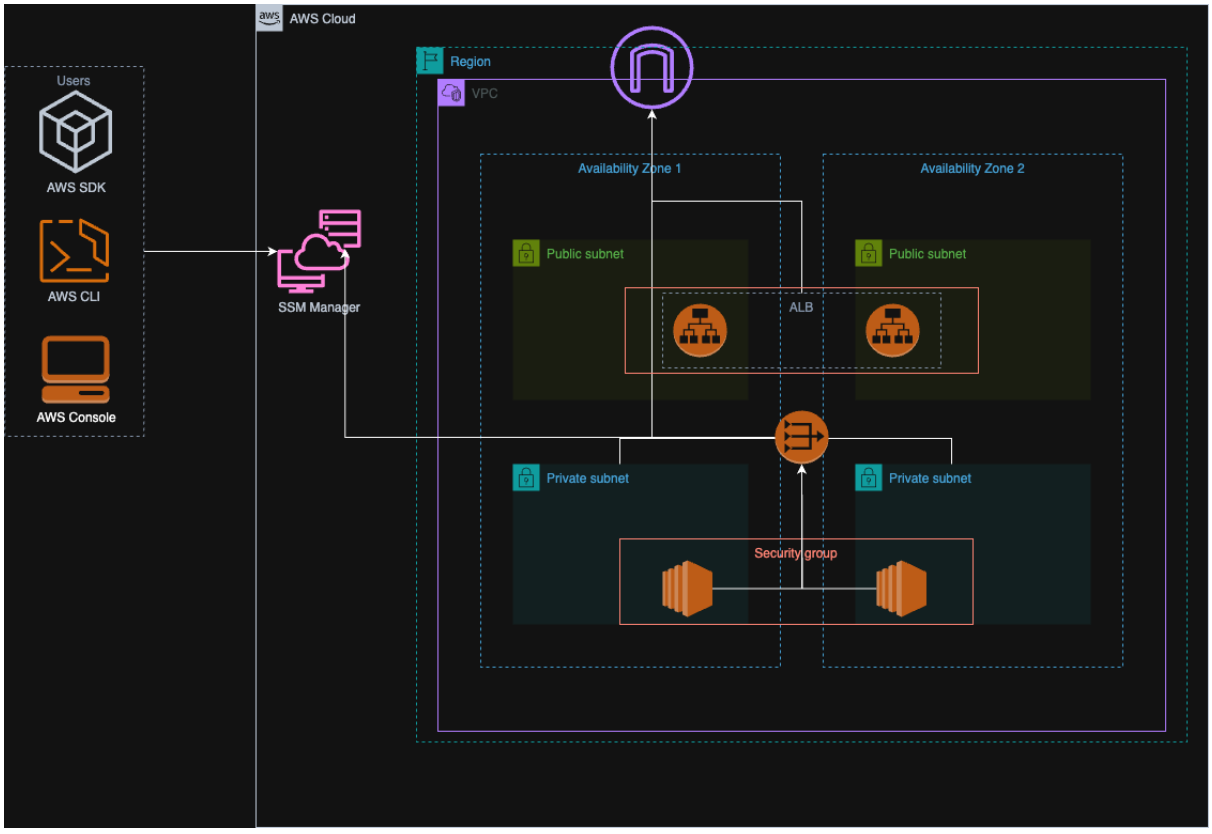
Belong Cloud Platforms Coding Challenge

Solution Overview

This document outlines the solution developed to meet the specified requirements for delivering a standard secure infrastructure on AWS Cloud. The solution leverages Terraform for Infrastructure as Code (IaC) and incorporates industry best practices.

Note: The development of this solution utilized ChatGPT and Google for research and content creation—because, let's face it, not using these tools in today's world would be like choosing a horse over a car. They're here to help, and everyone uses them!

Challenge Requirements and Solutions



1. Dedicated VPC with Public and Private Subnets

- **Implementation**:
  - A dedicated VPC was created with the following configurations:
    - CIDR Block: `10.0.0.0/16`.
    - Two public subnets across two Availability Zones (AZs).
    - Two private subnets across the same AZs.

    - An Internet Gateway (IGW) is attached to the VPC for public subnet connectivity.
    - A NAT Gateway is deployed in one of the public subnets to allow instances in the private subnets to access the internet securely.

2. EC2 Instance in Private Subnet Running HTTPD

- **Implementation**:
    - An EC2 instance is deployed in one of the private subnets.
    - The instance uses an Amazon Linux AMI with `t2.micro` as the instance type.
    - A `user_data` script is provided to configure the instance during initialization:
        - The server time zone is set to `Australia/Sydney`.
        - HTTPD (Apache) is installed and started as a service.
        - The file `belong-test.html` is downloaded from the S3 bucket (`belong-coding-challenge` in the Sydney region) and placed in `/var/www/html/index.html` to serve it publicly.

3. Terminal Access for Developers

- **Implementation**:
    - The EC2 instance is configured with AWS Systems Manager (SSM) for secure, keyless terminal access.
    - The instance has an IAM role attached with the `AmazonSSMManagedInstanceCore` policy.
    - Developers can access the EC2 instance via AWS CLI using SSM without requiring SSH or keys.

4. Alternative Solution for Terminal Access: Bastion Host

- Instead of using SSM, a bastion host could be deployed in the public subnet to act as an intermediary for accessing private instances.
- **Implementation**:
    - Deploy a lightweight EC2 instance in a public subnet.
    - Restrict access to the bastion host via security groups, allowing only trusted IPs.
    - Use SSH keys for accessing the bastion host and then connect to private instances via the bastion.
    - This approach introduces an additional layer of control but requires SSH key management.

5. Public Access to the Web Page Using ALB and Alternative CloudFront Solution

- **Implementation with ALB**:
    - An Application Load Balancer (ALB) is deployed in the public subnets to serve the web page hosted on the private EC2 instance.
    - The ALB forwards HTTP traffic to the private EC2 instance through a target group.

    - Health checks ensure that the instance is available and serving content before routing traffic.
    - The ALB provides a secure and scalable mechanism for exposing the web application publicly.
- **Alternative Solution with CloudFront**:
    - CloudFront, a global content delivery network (CDN) service, can be used to serve the HTML file stored in the S3 bucket directly to users.
    - **Implementation**:
        - Configure the S3 bucket to host static content with proper permissions.
        - Create a CloudFront distribution with the S3 bucket as the origin.
        - Use an origin access control (OAC) or bucket policy to restrict direct access to the S3 bucket.
        - Set caching policies in CloudFront to improve performance and reduce latency for users.
    - **Advantages**:
        - Provides lower latency by caching content closer to users globally.
        - Eliminates the need for an EC2 instance for serving static HTML content, reducing costs.
        - Offers additional features such as HTTPS support, geographic restrictions, and enhanced DDoS protection.
    - **Considerations**:
        - CloudFront is ideal for serving static content but may not support dynamic application needs without additional configurations.

Key Features and Best Practices

- **Security**:
    - Security groups enforce the principle of least privilege.
    - The EC2 instance is only accessible via SSM or the bastion host (if configured).
    - Public access to the application is only allowed through the ALB.
- **Scalability**:
    - The use of an ALB allows for scaling by adding more instances to the target group.
- **Automation**:
    - Terraform is used to define and provision all resources, ensuring reproducibility and consistency.

Architecture Diagram

The architecture consists of:

1. A VPC with public and private subnets.

2. An ALB in the public subnets routing traffic to the EC2 instance in the private subnet.
3. An S3 bucket serving static files.
4. Secure access to the EC2 instance via SSM or bastion host.

(Refer to the attached diagram for a detailed visualization.)

Advantages of the Solution

- **Secure and Centralized Access**: SSM removes the need for managing SSH keys while maintaining secure access.
- **Flexibility**: Terraform modules allow for easy adjustments and scaling.
- **Cost Optimization**: Use of private subnets, NAT Gateway, and efficient resource provisioning ensures cost-effective infrastructure.

Alternative Considerations

- Use AWS Secrets Manager to manage sensitive information like database credentials.
- Replace HTTPD with a containerized solution using ECS or EKS for greater scalability.
