Network and Security Configuration Details

Overview

This document outlines the network and security configuration implemented for the Kubernetes infrastructure and supporting CI/CD automation. The configuration highlights the least privilege access and secure connectivity.

1. Network Architecture

1.1 Public and Private Subnets

- A Bastion Host is deployed in a public subnet and serves as the sole administrative access point to the private Kubernetes (EKS) cluster.
- Kubernetes nodes reside in private subnets and are not directly accessible from the public internet.
- Developers access the infrastructure via Bastion Host only, using secure SSH and VPN routes.

1.2 IP Whitelisting & VPN Enforcement

- Access to the Bastion Host is restricted to specific office IP ranges.
- All users must connect through a secure VPN endpoint that is positioned behind a firewall before accessing any resources.
- Direct access to private resources from the internet is disallowed.

2. Load Balancer Configuration

2.1 Customer-Facing Load Balancer

- An AWS-managed Load Balancer is configured as the public-facing endpoint for customer access
- The Load Balancer forwards traffic to application services without revealing internal ingress or service endpoints.

2.2 No Public Ingress Exposure

- Kubernetes ingress controllers are not exposed to the public network.
- Internal DNS and service discovery mechanisms are used for east-west traffic within the cluster.

3. Access Control and Security

3.1 Principle of Least Privilege

- IAM roles and Kubernetes RBAC policies can be implemented based on user roles:
  a. Reader: View-only access
  b. Writer: Limited access to modify resources
- No user has admin privileges by default.

3.2 VPN and MFA Enforcement (Can be Implemented)

- All infrastructure access should have VPN authentication.
- Multi-factor authentication (MFA) should be enforced for all VPN and IAM logins.

3.3 EKS Role-Based Access (Can be Implemented)

- EKS access policies should be tightly controlled using IAM roles mapped via aws-auth ConfigMap.
- Role mappings define whether a user can view or modify specific Kubernetes namespaces or resources.

4. Automation and CI/CD Integration (Can be Implemented)

4.1 Ansible Configuration

- All infrastructure configurations are defined as Ansible playbooks should be stored in a Git repository.
- The Git repository is secured with branch protections and review policies.

4.2 Jenkins Pipelines

- Jenkins pipelines should be used to validate and deploy Ansible changes to the environment.
- Pipelines are triggered automatically on code commits to the main branch or manually approved PR merges.
- All Jenkins jobs execute within a controlled and isolated environment using defined credentials and roles.

5. Monitoring and Logging (Can be Implemented)

- All VPN, SSH, and infrastructure access is logged and monitored.
- Alerts are configured for unauthorized access attempts and policy violations.

Note: There are a few topics that are not implemented in this project but can be used for better security and network configuration perspective.