### **Deployment Report for VPC, Security Groups, and EKS using Terraform**

#### **Introduction**

This report documents the implementation process for deploying a modular infrastructure setup using Terraform. The key components deployed include a Virtual Private Cloud (VPC), Security Groups (SG), Elastic Kubernetes Service (EKS), Elastic Container Registry (ECR), and an NGINX Ingress Controller. This approach emphasizes scalability, reusability, and maintainability, ensuring the infrastructure aligns with best practices.

### **Approach**

1. **Modular Design:**
   * Each component (VPC, SG, EKS, etc.) is encapsulated within its own Terraform module. This allows for reusability, cleaner code organization, and easier updates or modifications.
2. **Dynamic Resource Creation:**
   * Resources are dynamically created using Terraform's for\_each feature, enabling the deployment of multiple instances of a component (e.g., multiple VPCs or EKS clusters) based on input variables.
3. **Consistent Naming:**
   * Local values are used to standardize resource names based on the Terraform workspace. For example, all resources are prefixed with rankbang-<environment> to identify and segregate resources based on the environment (e.g., dev, staging, or production).
4. **Dependency Management:**
   * Explicit dependencies are defined between modules to ensure correct resource creation order. For example:
     + Security Groups depend on the VPC.
     + EKS depends on both the VPC and Security Groups.
5. **Helm Integration:**
   * The NGINX Ingress Controller is deployed using the helm\_release resource. Configuration values are passed dynamically to the Helm chart for customization.

### 

### 

### 

### **Implementation Choices**

1. **VPC:**
   * A modular approach for creating VPCs ensures flexibility in defining custom networking setups.
   * Includes public and private subnets, NAT gateways, and route tables for efficient resource isolation.
2. **Security Groups:**
   * Security rules (ingress and egress) are defined dynamically, allowing granular control over resource access.
3. **EKS:**
   * The EKS module provisions a fully managed Kubernetes cluster integrated with the specified subnets and security groups.
   * This setup ensures high availability and scalability for containerized applications.
4. **ECR:**
   * Elastic Container Registries are provisioned for storing Docker images, adhering to consistent naming conventions.
5. **Helm Release:**
   * The NGINX Ingress Controller is deployed into a dedicated namespace using Helm charts, with dynamic configuration for scalability and resource optimization.

### **Challenges and Resolutions**

1. **Dynamic Resource Handling:**
   * Challenge: Managing multiple resources dynamically using for\_each while maintaining clarity in resource mappings.
   * Resolution: Used consistent keys in input variables to map resources accurately.
2. **Helm Chart Customization:**
   * Challenge: Passing dynamic values to the Helm chart for NGINX Ingress Controller.
   * Resolution: Utilized Terraform's dynamic blocks to iterate over configuration values.
3. **Scalability:**
   * Challenge: Designing the setup to support multiple environments (e.g., dev, staging, production).
   * Resolution: Leveraged Terraform workspaces and modular design for environment-specific deployments.