白皮书：

基于Terraform与EKS的企业级云原生基础设施实践

发布方：成都易定云科技有限公司
日　期：2026年3月
版　本：1.0

摘要

随着企业应用全面转向云原生架构，如何在AWS上构建高可用、可扩展、安全合规的Kubernetes基础设施，已成为技术团队面临的核心挑战。成都易定云科技有限公司（以下简称“易定云”）作为AWS高级合作伙伴，结合大量企业级落地经验，沉淀出一套基于Terraform、Amazon EKS、Karpenter的端到端基础设施即代码（IaC）解决方案。

本白皮书深入剖析易定云在VPC网络设计、EKS集群构建、节点弹性伸缩、安全组精细化管理、监控解决方案，日记ELK方案、灾备处理等方面的最佳实践，并分享我们在多云环境下的自动化运维与成本优化见解，为企业云原生转型提供可复用的架构参考。

一、背景与挑战

在云原生技术广泛普及的当下，越来越多的企业选择在AWS上运行Kubernetes集群。然而，在实际落地过程中，企业普遍面临以下痛点：

基础设施碎片化

手动创建VPC、子网、安全组、EKS集群等资源，操作繁琐且容易出错，缺乏版本控制与变更追溯。

环境一致性难保障

开发、测试、生产环境配置差异大，传统手工运维难以保证环境一致性，导致“开发环境跑得通，生产环境出问题”的困境。

弹性伸缩能力不足

传统的节点组（Node Group）基于固定实例规格和数量，难以应对突发流量，且容易造成资源浪费。

安全合规压力大

网络隔离、最小权限原则、日志审计等要求严格，手工配置容易遗漏或配置不当。

缺乏成熟的监控系统

没有统一的日记采集平台，以及良好的可视化监控页面。

易定云见解：
我们始终坚信，基础设施即代码（IaC）是企业云原生化的基石。只有将网络、集群、安全策略全部代码化，才能实现“声明式运维”，确保环境的可复现性、可审计性和可治理性。

二、解决方案架构与易定云实践

易定云基于Terraform构建了一套完整的EKS基础设施即代码方案，实现了从网络到应用运行环境的全栈自动化部署。

2.1整体架构

本方案采用分层设计，将网络层、集群层、节点层、应用层、运维层进行解耦，便于独立演进与复用。

网络层（VPC模块）

创建跨多个可用区的VPC，划分公网子网、应用私有子网、数据库私有子网，实现精细化的网络隔离。NAT网关支持高可用模式（多可用区部署）或单点模式，兼顾成本与容灾。

```
└── vpc
    ├── main.tf
    ├── nat.tf
    ├── outputs.tf
    ├── route_tables.tf
    ├── security_groups.tf
    ├── subnet.tf
    ├── subnet_groups.tf
    ├── variables.tf
    └── vpc.tf
```

安全组层（Security Group模块）

为ALB、应用服务、RDS、Redis分别创建独立安全组，严格控制访问来源与端口。

集群层（EKS模块）

创建EKS集群，配置控制平面访问策略（公网+私网），启用API审计、控制器管理、调度器等日志，并将日志集中至CloudWatch Logs。

节点层

支持两种模式——传统的托管节点组（Managed Node Group）与Karpenter动态弹性伸缩模式，满足不同场景的容量管理需求。

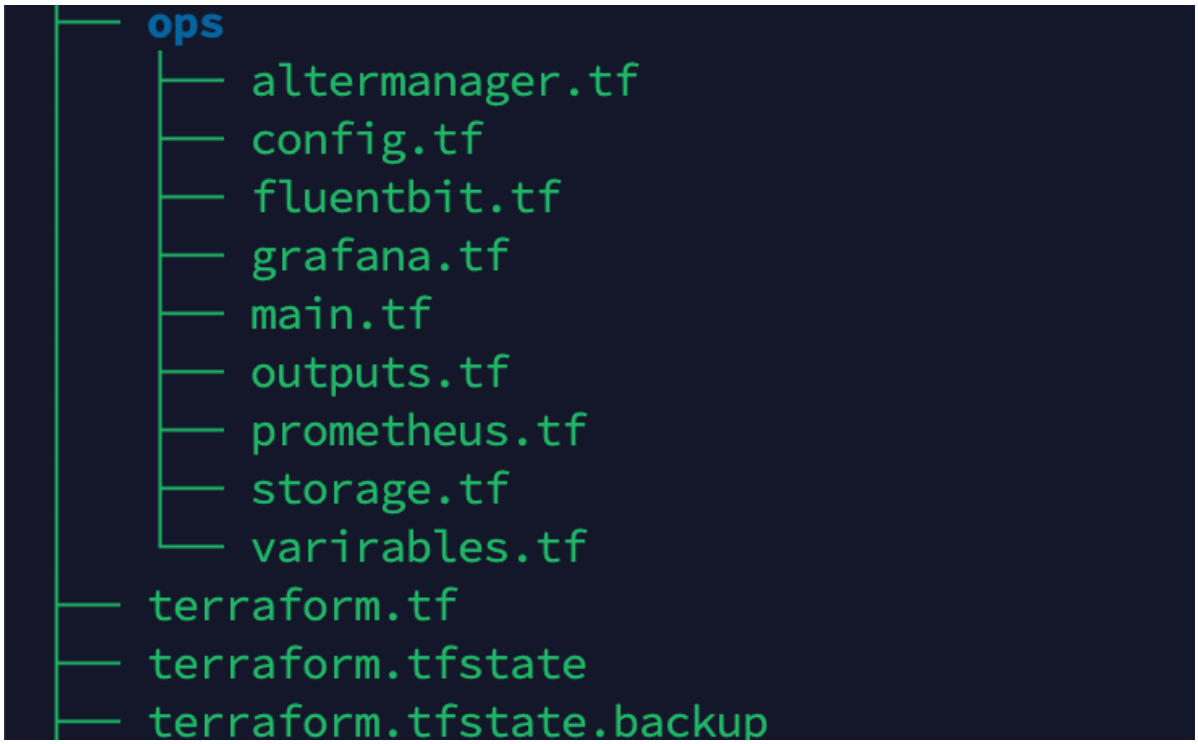
扩展组件层

通过Helm一键部署AWS Load Balancer Controller、EBS CSI Driver、Karpenter等核心插件，实现与AWS服务深度集成。

```
terraform-set-env/
├── eks
│   ├── addons.tf
│   ├── eks.tf
│   ├── iam_roles.tf
│   ├── karpenter.tf
│   ├── main.tf
│   ├── node_groups.tf
│   ├── oidc.tf
│   ├── outputs.tf
│   ├── policies
│   │   └── alb-controller-policy.json
│   └── variables.tf
```

运维层

通过helm 一件部署prometheus 以及fluentbit 采集日记到opensearch，完成客户日记方面的统一治理需求。



2.2网络设计实践

在VPC设计中，易定云遵循以下原则：

多可用区高可用

所有子网均部署在两个以上可用区，确保单可用区故障时业务不受影响。

子网分层隔离

公网子网

部署ALB、NAT网关，通过Internet Gateway访问公网。

应用私有子网

部署EKS工作节点，通过NAT网关访问公网进行镜像拉取、系统更新。

数据库私有子网

部署RDS、ElastiCache等数据类服务，仅允许应用私有子网访问，且默认无公网出口，最大程度保障数据安全。

```
resource "aws_subnet" "public" {
  count = 2

  vpc_id                  = aws_vpc.main.id
  cidr_block              = cidrsubnet(var.vpc_cidr, 8, count.index) # 192.168.0.0/24 192.168.1.0/24
  availability_zone       = data.aws_availability_zones.available.names[count.index]
  map_public_ip_on_launch = true

  tags = merge(
    var.tags,
    {
      Name        = "${var.project_name}-${var.environment}-public-subnet-${local.subnet_letters[count.index]}"
      Environment = var.environment
      Type        = "Public"
      Tier        = "Public"
    }
  )
}

# ============================================================================
# Application Private Subnets (2个) - /20
# ============================================================================

resource "aws_subnet" "app_private" {
  count = 2

  vpc_id            = aws_vpc.main.id
  cidr_block        = cidrsubnet(var.vpc_cidr, 4, count.index + 2) # 192.168.32.0/20, 192.168.48.0/20
  availability_zone = data.aws_availability_zones.available.names[count.index]

  tags = merge(
    var.tags,
    {
      Name        = "${var.project_name}-${var.environment}-app-private-subnet-${local.subnet_letters[count.index]}"
      Environment = var.environment
      Type        = "Private"
      Tier        = "Application"
    }
  )
}
```

公网子网关联公共路由表，默认路由指向Internet Gateway。

应用私有子网与数据库私有子网分别关联独立路由表，默认路由指向NAT网关，实现出站互联网访问与入站访问隔离。

```
# Application route to NAT Gateway
resource "aws_route" "app_private_nat" {
  count = var.enable_nat_gateway ? 2 : 0

  route_table_id         = aws_route_table.app_private[count.index].id
  destination_cidr_block = "0.0.0.0/0"
  nat_gateway_id         = var.single_nat_gateway ? aws_nat_gateway.main[0].id : aws_nat_gateway.main[count.index].id
}

# Associate app private subnets with app route tables
resource "aws_route_table_association" "app_private" {
  count = 2

  subnet_id      = aws_subnet.app_private[count.index].id
  route_table_id = aws_route_table.app_private[count.index].id
}

# ============================================================================
# 3. Database Private Route Tables (2个, 每个数据库子网一个)
# ============================================================================

resource "aws_route_table" "db_private" {
  count = 2

  vpc_id = aws_vpc.main.id

  tags = merge(
    var.tags,
    {
      Name        = "${var.project_name}-${var.environment}-db-private-rt-${local.subnet_letters[count.index]}"
      Environment = var.environment
      Type        = "Private"
      Tier        = "Database"
    }
  )
```

易定云见解：

我们推荐在生产环境中启用 single_nat_gateway = false，为每个可用区部署独立的NAT网关，虽然成本略有增加，但避免了单点故障风险。同时，将数据库子网与业务子网严格隔离，即使应用层被入侵，攻击者也无法直接访问数据库层，显著提升纵深防御能力。

2.3EKS集群构建与安全加固

易定云在EKS集群配置中融入了以下安全实践：

控制平面访问控制

同时启用私网访问与公网访问，但对公网访问进行CIDR白名单限制，仅允许企业办公网IP或堡垒机IP访问。

集群日志全面开启

启用API、审计、控制器管理、调度器、认证器等所有日志类型，并配置CloudWatch Logs保留7天以上，满足安全合规要求。

```
resource "aws_eks_cluster" "main" {
  name     = var.cluster_name
  version  = var.cluster_version
  role_arn = aws_iam_role.cluster.arn

  vpc_config {
    subnet_ids              = length(var.control_plane_subnet_ids) > 0 ? var.control_plane_subnet_ids : var.private_subnet_id
    endpoint_private_access = var.cluster_endpoint_private_access
    endpoint_public_access  = var.cluster_endpoint_public_access
    public_access_cidrs     = var.cluster_endpoint_public_access_cidrs
    security_group_ids      = [aws_security_group.cluster.id]
  }

  enabled_cluster_log_types = var.enabled_cluster_log_types

  depends_on = [
    aws_iam_role_policy_attachment.cluster_policy,
    aws_cloudwatch_log_group.cluster
  ]

  tags = merge(
    var.tags,
    {
      Name        = var.cluster_name
      Environment = var.environment
    }
```

节点安全组精细化配置

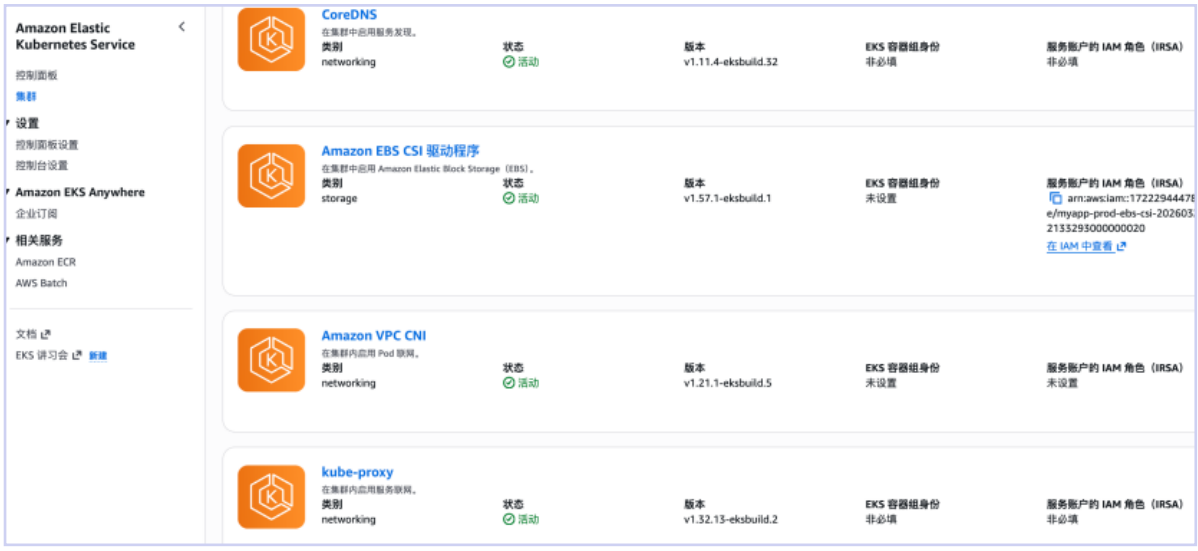
控制平面与节点间仅开放必要端口（TCP 443、1025–65535）。
节点间允许所有通信，满足Kubernetes内部通信需求。
节点出方向全部放行，但通过NAT网关的源IP限制，防止滥用。

```
# Control plane to node communication
resource "aws_security_group_rule" "node_ingress_cluster" {
  type                     = "ingress"
  from_port                = 1025
  to_port                  = 65535
  protocol                 = "tcp"
  source_security_group_id = aws_security_group.cluster.id
  security_group_id        = aws_security_group.node.id
  description              = "Allow control plane to communicate with nodes"
}

# Node to control plane communication
resource "aws_security_group_rule" "cluster_ingress_node" {
  type                     = "ingress"
  from_port                = 443
  to_port                  = 443
  protocol                 = "tcp"
  source_security_group_id = aws_security_group.node.id
  security_group_id        = aws_security_group.cluster.id
  description              = "Allow nodes to communicate with control plane"
}
```

OIDC与IAM集成

通过为EKS集群配置OpenID Connect（OIDC）提供商，实现Kubernetes ServiceAccount与IAM角色的精确绑定。例如，ALB Controller、EBS CSI Driver、Karpenter均通过ServiceAccount与IAM角色关联，遵循最小权限原则。

```
resource "aws_iam_role" "karpenter_controller" {
  count = var.enable_karpenter ? 1 : 0

  name_prefix = "${var.cluster_name}-karpenter-controller-"

  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [{
      Action = "sts:AssumeRoleWithWebIdentity"
      Effect = "Allow"
      Principal = {
        Federated = aws_iam_openid_connect_provider.cluster.arn
      }
      Condition = {
        StringEquals = {
          "${replace(aws_iam_openid_connect_provider.cluster.url, "https://", "")}:sub" = "system:serviceaccount:karpent
          "${replace(aws_iam_openid_connect_provider.cluster.url, "https://", "")}:aud" = "sts.amazonaws.com"
        }
      }
    }]
  })

  tags = merge(
    var.tags,
    {
      Name        = "${var.cluster_name}-karpenter-controller-role"
      Environment = var.environment
    }
  )
}

resource "aws_iam_policy" "karpenter_controller" {
  count = var.enable_karpenter ? 1 : 0

  name_prefix = "${var.cluster_name}-karpenter-controller-"
  description = "IAM policy for Karpenter controller"
```



2.4弹性伸缩与成本优化

在节点弹性伸缩方面，易定云提供两套方案，客户可根据业务场景灵活选择：

方案	适用场景	核心优势
托管节点组	流量平稳、规格固定的生产环境	管理简单，支持滚动更新、自动修复
Karpenter	流量波动大、混合实例类型需求	动态选择最优实例规格，支持Spot实例，可降低30%–50%计算成本

Karpenter核心能力：

智能实例选择

根据Pod的资源请求（CPU、内存），自动选择最合适的实例类型。

混合容量类型

同时支持Spot与On-Demand实例，通过配置比例平衡成本与稳定性。

中断处理

通过EventBridge监听Spot实例中断事件，自动将Pod调度至其他节点，实现优雅驱逐。

易定云见解：

我们强烈推荐生产环境采用“托管节点组 + Karpenter”混合模式。用托管节点组维持少量稳定节点运行核心服务，用Karpenter应对突发流量和批处理任务，既保障了核心业务的稳定性，又实现了计算成本的极致优化。

```
aws-load-balancer-controller-545cf64c68-5xqm8   1/1     Running   0          34m
aws-load-balancer-controller-545cf64c68-7q9rj   1/1     Running   0          34m
aws-node-rpkwq                                  2/2     Running   0          35m
aws-node-wp7gt                                  2/2     Running   0          35m
coredns-5457f6cbdf-6rrw7                        1/1     Running   0          35m
coredns-5457f6cbdf-mh56j                        1/1     Running   0          35m
ebs-csi-controller-68d99c8f8c-7scr4             6/6     Running   0          35m
ebs-csi-controller-68d99c8f8c-szljm             6/6     Running   0          35m
ebs-csi-node-2ztm5                              3/3     Running   0          35m
ebs-csi-node-z2tzl                              3/3     Running   0          35m
kube-proxy-psg8t                                1/1     Running   0          35m
kube-proxy-q5hc2                                1/1     Running   0          34m
```

2.5关键扩展组件集成

易定云通过Helm与Terraform的结合，实现了扩展组件的声明式部署：

AWS Load Balancer Controller

支持通过Kubernetes Ingress或Service类型为LoadBalancer自动创建AWS ALB/NLB，并提供基于注解的丰富配置（SSL证书、安全策略、访问控制等）。

EBS CSI Driver

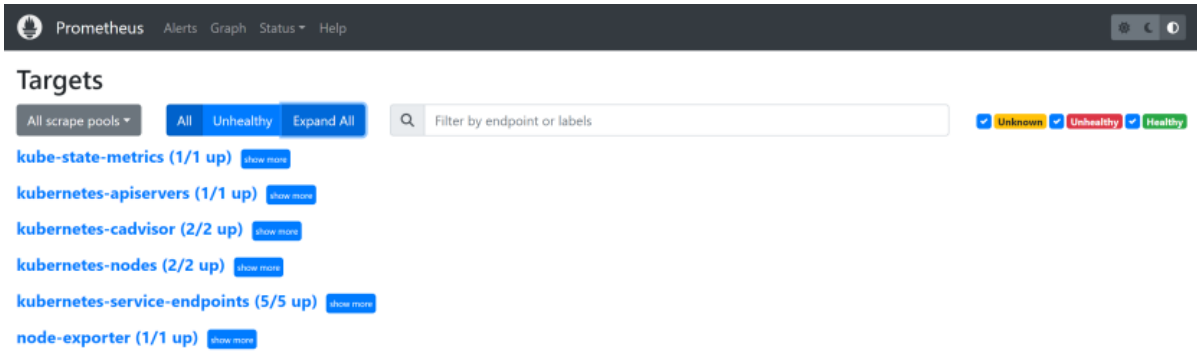
为Pod提供持久化存储能力，支持动态创建EBS卷，并通过StorageClass实现不同性能类型（gp3、io2）的配置。

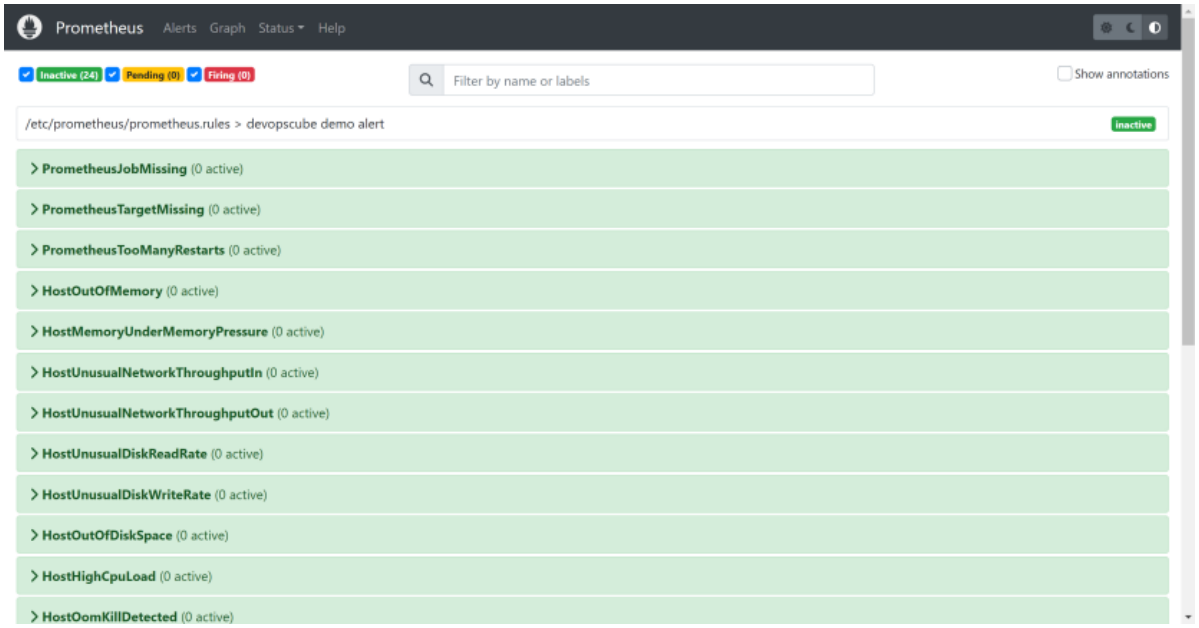
Karpenter

除节点弹性外，还通过SQS队列与EventBridge集成，实时响应Spot实例中断、EC2健康事件，实现节点自动替换。

Prometheus

自动采集集群以及节点状态，通过PQL语句进行查询监控告警、确保及时识别响应资源故障。





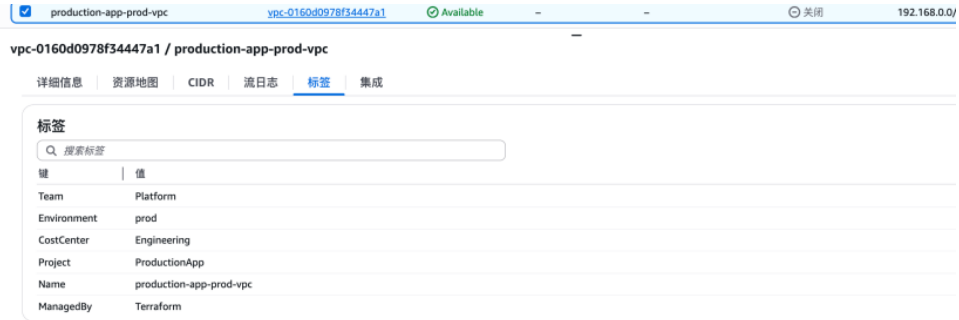
三、易定云核心实践与差异化

基于大量企业级项目经验，易定云在此方案中沉淀了以下独特见解与实践：

3.1资源命名规范与标签治理

我们严格定义并执行资源命名规范：
{project_name}-{environment}-{resource_type}-{identifier}
例如：production-app-prod-app-private-subnet-a
同时，为每个资源打上标准标签：
Environment：区分prod/staging/dev
Project：项目归属
ManagedBy：标识由Terraform管理
CostCenter：便于成本分摊

良好的命名与标签体系，是实现多环境隔离、成本可视化、故障定位的基础。易定云可为企业定制标签策略，并与AWS Cost Explorer、Budgets集成，实现精细化的成本管理与告警。



3.2多环境配置复用

通过Terraform变量与模块化设计，我们实现了基础设施配置在不同环境间的复用与差异化。例如：
dev环境: single_nat_gateway=true,instance_types = ["t3.medium”],以降低成本。
prod环境: single_nat_gateway = false,instance_types = ["c7i.xlarge”,”m7i.xlarge”],以保障高可用与性能。

```
variable "enable_nat_gateway" {
  description = "Enable NAT Gateway"
  type        = bool
  default     = true
}

variable "single_nat_gateway" {
  description = "Use single NAT Gateway for all private subnets"
  type        = bool
  default     = false
}

variable "tags" {
  description = "Additional tags for all resources"
  type        = map(string)
  default     = {}
}

# ALB Security Group允许的CIDR
```

3.3基础设施变更的安全管控

我们将Terraform代码托管于Git，并与CI/CD流水线（如GitLab CI、GitHub Actions）集成：

变更前

执行 terraform plan 生成变更预览，需经人工审批。

变更后

变更后：执行 terraform apply 并自动记录状态文件至S3后端，实现状态锁定与并发控制。

这种做法有效避免了“运维人员直接控制台操作”带来的配置漂移和变更不可追溯问题。

3.4安全与合规内建

我们将Terraform代码托管于Git，并与CI/CD流水线（如GitLab CI、GitHub Actions）集成：

最小权限IAM

为集群、节点、控制器分别创建独立IAM角色，仅授予必要权限。

Secrets安全

敏感信息（如数据库密码）使用AWS Secrets Manager或SSM Parameter Store存储，通过Kubernetes CSI Driver挂载至Pod，避免明文泄露。

网络审计

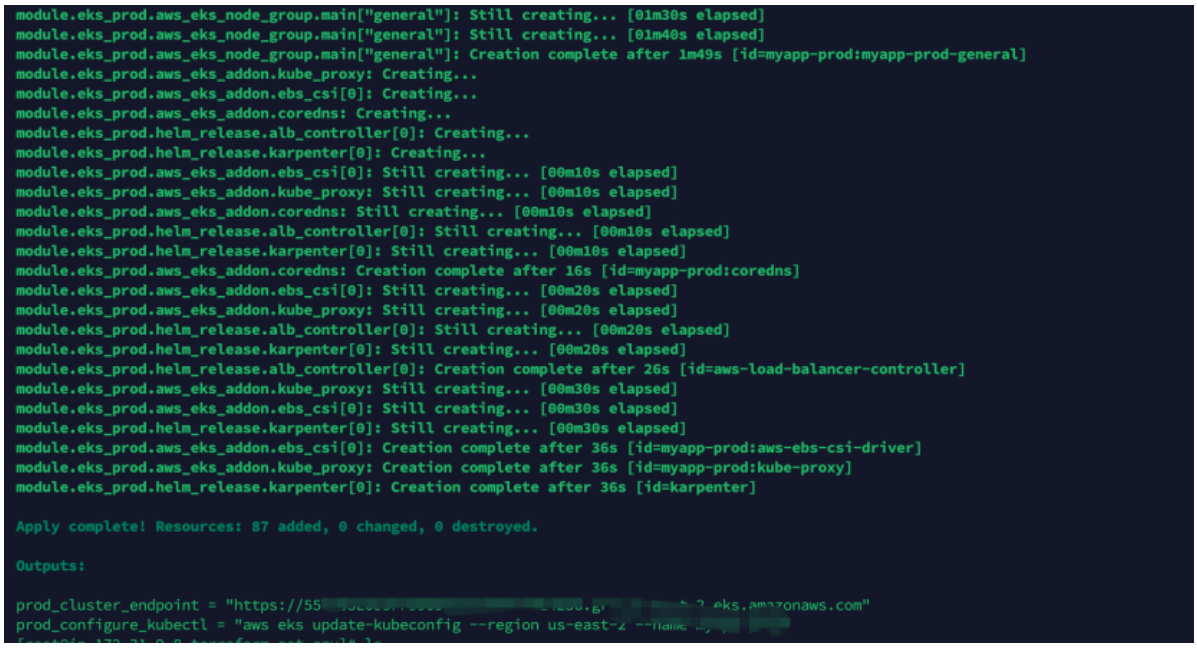
通过VPC Flow Logs记录所有网络流量，结合AWS Security Hub进行持续合规检查。

四、关键成果与客户收益

采用易定云的EKS基础设施方案，客户可获得以下显著收益：

部署效率提升80%

从手工配置数小时缩短至Terraform一键部署，新环境交付时间从天级降至分钟级。



成本显著优化

通过Karpenter引入Spot实例，计算成本平均降低30%–40%；同时，多环境共用一个模块，避免了重复建设。

```yaml
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: spot-pool
spec:
  # 1. 权重：设置较高的权重，让 Karpenter 优先使用此 Spot 节点池
  weight: 100

  # 2. 限制：限制最大节点数量，防止 Spot 价格波动导致疯狂扩容产生高额费用
  limits:
    cpu: 1000
    memory: 1000Gi

  # 3. 模板：定义节点所需的硬件和标签
  template:
    metadata:
      labels:
        workload-type: spot-workloads
    spec:
      # 核心配置：指定容量类型为 Spot
      requirements:
        - key: karpenter.sh/capacity-type
          operator: In
          values: ["spot"]
```

安全合规达标

满足企业级网络隔离、最小权限、日志审计等要求，通过金融、政府等监管机构的合规审查。

运维风险降低

所有基础设施变更可追溯、可回滚，杜绝了“登录控制台随意改”带来的配置漂移和故障隐患。

弹性能力增强

Karpenter的智能弹性伸缩，使集群可平滑应对突发流量峰值，且无需预留大量Buffer资源。

五、总结与展望

成都易定云科技有限公司基于Terraform与Amazon EKS构建的企业级云原生基础设施方案，将网络、集群、节点、扩展组件等全部代码化，实现了高可用、安全合规、弹性伸缩、成本可控的目标。

未来

我们坚信，未来的基础设施管理将完全由代码驱动
企业需要的不再是“运维工程师”，而是具备编程思维的“平台工程师”
易定云将持续深化在IaC、GitOps、FinOps领域的探索
结合AWS不断推出的新服务（如EKS Auto Mode、Amazon Q Developer）
为客户提供更智能、更高效的云原生基础设施解决方案

FUTURE