Generated with sparks and insights from 3 sources

Based on my comprehensive research, I'll now provide you with an in-depth analysis of Terraform and Ansible automation for Cisco ISE deployment with pxGrid integration, Security Group Tags (SGT) implementation, and Catalyst Center integration.

Comprehensive Deep Research: Terraform and Ansible for Cisco ISE Deployment with pxGrid and SGT Integration

Executive Summary

This comprehensive research covers the complete automation of Cisco Identity Services Engine (ISE) 3.3+ deployment using Infrastructure as Code (IaC) tools - Terraform and Ansible - with deep integration into pxGrid services, Security Group Tags (SGT) management, and seamless Catalyst Center integration for Software-Defined Access (SDA) deployments.

1. Terraform Provider Deep Dive: Cisco ISE and Catalyst Center

1.1 Terraform Provider Architecture and Components

Current Provider Status (2024): - Primary Provider: CiscoISE/ciscoise[1] version 0.8.2-beta - ISE Version Support: 3.1 Patch 1, 3.2-Beta, 3.3-Patch-1, 3.4.0, 3.5.0 - Go Version Requirements: Go 1.20+ for provider development - Terraform Version: 1.0+

Catalyst Center Provider CiscoDevNet/catalystcenter[2]: - Version 0.3.x+ supporting Catalyst Center 2.3.7.9+ - Supports SDA fabric provisioning and ISE integration

1.2 Resource Management Matrix

Resource Category	Terraform Resources	Coverage Status	ISE Version
Security Groups	`ciscoise_sgt`, `ciscoise_sg_mapping`, `ciscoise_sg_mapping_deploy`	Complete	3.1+
Network Devices	`ciscoise_network_device`, `ciscoise_network_device_group`	Complete	3.1+
Policies	`ciscoise_authorization_profile`, `ciscoise_authorization_policy`	Complete	3.2+
Identity Stores	`ciscoise_active_directory`, `ciscoise_ldap`	Complete	3.1+
Sponsor Portals	`ciscoise_sponsor_portal`, `ciscoise_guest_portal`	Complete	3.3+
Certificates	`ciscoise_system_certificate`, `ciscoise_certificate_profile`	Complete	3.2+
pxGrid	`ciscoise_pxgrid_service_unregister`, `ciscoise_pxgrid_access_secret`	Partial	3.0+

1.3 Advanced Terraform Configuration

```
# terraform/providers.tf
terraform {
  required_version = ">= 1.0"
  required_providers {
    ciscoise = {
      source  = "CiscoISE/ciscoise"
      version = "0.8.2-beta"
    }
    catalystcenter = {
      source  = "CiscoDevNet/catalystcenter"
      version = "0.3.2"  # Latest verified version
    }
  }

  # Remote state management for production
  backend "s3" {
    bucket         = "ise-terraform-state-prod"
    key            = "infrastructure/terraform.tfstate"
    region         = "us-west-2"
    dynamodb_table = "terraform-state-lock"
    encrypt        = true
  }
}

provider "ciscoise" {
  username    = var.ise_username
  password    = var.ise_password
  base_url    = "https://${var.ise_primary_hostname}"
  ssl_verify  = var.ise_ssl_verify
  debug       = var.ise_debug_logs
  wait_on_rate_limit = true
  single_request_timeout = 120
}

provider "catalystcenter" {
  base_url = var.catalyst_center_url
  username = var.catalyst_center_username
  password = var.catalyst_center_password
  debug    = var.cc_debug_logs
  ssl_verify = false
}
```

1.4 State Management Best Practices

```
# terraform/environments/production/backend.tf
terraform {
  backend "remote" {
    hostname     = "app.terraform.io"
    organization = "your-organization"

    workspaces {
      name = "ise-prod"
    }
  }
}

# Module versioning strategy
module "ise_security_groups" {
  source  = "git::https://github.com/your-org/ise-modules.git//modules/

  version = "2.1.0"  # Pin exact version for production
  security_groups = var.security_groups
  deployment_environment = "production"
}
```

2. Ansible Collection Analysis: cisco.ise

2.1 Collection Architecture and Updates

Latest Collection Status (2024): - Collection: cisco.ise[3] version 2.10.0+ - Python SDK: ciscoisesdk 2.2.3+ (required dependency) - ISE Version Support: 3.1.0 through 3.5.0 - Python Requirements: Python >= 3.6

2.2 Comprehensive Module Coverage

Security Group Management:

```
# Complete SGT automation workflow
---
- name: Advanced SGT Management with pxGrid Integration
  hosts: ise_servers
  gather_facts: no

  tasks:
    - name: Create SGT with pxGrid propagation
      cisco.ise.sgt:
        ise_hostname: "{{ ise_hostname }}"
        ise_username: "{{ ise_username }}"
        ise_password: "{{ ise_password }}"
        state: present
        name: "Employees_HQ"
        description: "HQ Employee Security Group"
        value: 100
        propogateToApic: true
      register: sgt_result

    - name: Create IP-to-SGT mapping
      cisco.ise.sg_mapping:
        ise_hostname: "{{ ise_hostname }}"
        ise_username: "{{ ise_username }}"
        ise_password: "{{ ise_password }}"
        state: present
        name: "Corporate_Mapping_100"
        hostIp: "10.100.0.0"
        mask: 16
        sgt: "{{ sgt_result.item.value }}"
        deployTo: ALL
      register: mapping_result
```

pxGrid Configuration Automation:

```
# pxGrid service enabling and trust establishment
---
- name: pxGrid Service Configuration with Certificate Management
  hosts: ise_servers
  tasks:
    - name: Enable pxGrid persona on ISE nodes
      cisco.ise.node_deployment:
        ise_hostname: "{{ ise_hostname }}"
        ise_username: "{{ ise_username }}"
        ise_password: "{{ ise_password }}"
        hostname: "{{ item.hostname }}"
        persona: "{{ 'pxGrid' }}"
        state: present
      loop: "{{ ise_deployment.nodes | selectattr('enable_pxgrid', 'equ

    - name: Configure pxGrid auto-approval settings
      ansible.builtin.uri:
        url: "https://{{ ise_hostname }}/admin/API/pxGrid/Settings"
        method: PUT
        user: "{{ ise_username }}"
        password: "{{ ise_password }}"
        body:
          automaticallyApproveCertBasedAccounts: true
          allowPasswordBasedAccounts: true
        body_format: json
        validate_certs: false
```

2.3 Advanced Policy Configuration

Network Access Policy Automation:

```
# Complete policy automation with SGT integration
- name: Create Authorization Profile with SGT Enforcement
  cisco.ise.authorization_profile:
    ise_hostname: "{{ ise_hostname }}"
    state: present
    name: "Employee_Full_Access_With_SGT"
    description: "Full network access with Employees SGT"
    vlan:
      nameID: 10
      tagID: 10
    securityGroup:
      id: "{{ employees_sgt_id }}"
      name: "Employees"

- name: Create Dynamic Authorization Policy
  cisco.ise.authorization_policy:
    ise_hostname: "{{ ise_hostname }}"
    state: present
    name: "Employee_Wireless_802.1X"
    rule:
      condition:
        attribute: "Network_Access:UseCase"
        operator: "equals"
        value: "Wireless 802.1X"
      securityGroup: "Employees"
    profiles: ["Employee_Full_Access_With_SGT"]
```

3. pxGrid Integration Deep Dive

3.1 pxGrid Architecture and Communication Flows

pxGrid 2.0 Architecture:

```
┌─────────────────────────────────────────────────────────────┐
│                    pxGrid Controller (ISE)                   │
│                                                              │
│  ┌──────────────┐  ┌──────────────┐  ┌────────────────┐      │
│  │ Admin Node   │  │ MNT Node     │  │ pxGrid Service │      │
│  │ (+ pxGrid)   │  │ (+ pxGrid)   │  │ Publisher      │      │
│  └──────────────┘  └──────────────┘  └────────────────┘      │
└─────────────────────────┬─────────────────────────────────────┘
                          │ REST API
                          │ WebSockets
┌─────────────────────────┴───────────────────────────────────┐
│                 pxGrid Publishers/Subscribers                │
│                                                              │
│  ┌──────────────┐  ┌──────────────┐  ┌────────────────┐      │
│  │ Catalyst     │  │ Firepower    │  │ pxGrid Clients │      │
│  │ Center       │  │ FMC          │  │ (Custom)       │      │
│  └──────────────┘  └──────────────┘  └────────────────┘      │
└──────────────────────────────────────────────────────────────┘
```

3.2 Certificate Management and Trust Establishment

Automated Certificate Management:

```
 #!/bin/bash
# pxgrid-cert-setup.sh - Automated pxGrid certificate management

#!/bin/bash
set -euo pipefail

ISE_HOST="ise-pan.example.com"
```

```
ISE_USER="admin"
ISE_PASS="password"
PXGRID_SUBJECT="CN=pxgrid-client,C=US,ST=CA,L=SanJose,OU=IT,O=ExampleCo

echo "=== pxGrid Certificate Automation ==="

# Generate pxGrid certificate
openssl req -new -newkey rsa:2048 -nodes \
  -keyout pxgrid-client.key -out pxgrid-client.csr \
  -subj "$PXGRID_SUBJECT"

# Submit CSR to ISE via API
CSR_JSON=$(cat <<EOF
{
"pem": "$(cat pxgrid-client.csr | sed 's/$/\\n/' | tr -d '\n')",
  "subject": "$PXGRID_SUBJECT"
}
EOF)

# Upload to ISE
curl -k -u "$ISE_USER:$ISE_PASS" \
  -X POST \
  -H "Content-Type: application/json" \
  -d "$CSR_JSON" \
  "https://$ISE_HOST/admin/API/certificate/csr"
```

3.3 pxGrid Service Enablement Automation

Complete pxGrid Service Setup:

```
 # terraform/pxgrid-config.tf
resource "ciscoise_system_certificate" "pxgrid_certificate" {
  friendly_name = "pxGrid-Service-Certificate"
  certificate_usage = "PRIVKEY"
  subject = {
    common_name = "pxgrid.${var.ise_domain}"
    country = "US"
    state = "CA"
    locality = "San Jose"
    organization = var.company_name
    organizational_unit = "IT"
  }
  key_type = "RSA"
  key_length = 2048
  digest = "SHA256"
}

# Enable pxGrid service via remote execution
resource "null_resource" "enable_pxgrid_service" {
  depends_on = [ciscoise_system_certificate.pxgrid_certificate]

  provisioner "local-exec" {
    command = "python3 ${path.module}/scripts/enable_pxgrid.py"
    environment = {
      ISE_HOST = var.ise_primary_hostname
      ISE_USER = var.ise_username
      ISE_PASS = var.ise_password
    }
  }
}
```

Python Script for pxGrid Enablement:

```
 # scripts/enable_pxgrid.py
#!/usr/bin/env python3
import os
import requests
import json
from requests.auth import HTTPBasicAuth

def enable_pxgrid_service(host, username, password):
    """Enable pxGrid service and configure settings"""
    # Enable pxGrid persona
    url = f"https://{host}/admin/API/Infra/Node"
    headers = {"Content-Type": "application/json"}
    auth = HTTPBasicAuth(username, password)

    # Get current node configuration
    response = requests.get(url, headers=headers, auth=auth, verify=Fal
    nodes = response.json()

    # Enable pxGrid persona on appropriate nodes
    for node in nodes:
        if 'PSN' in node.get('roles', []):
            # Update node to add pxGrid persona
            update_url = f"{url}/{node['id']}"
            node['roles'].append('pxGrid')

            update_response = requests.put(
                update_url,
                headers=headers,
                auth=auth,
                json=node,
                verify=False
            )

            if update_response.status_code == 200:
                print(f"✓ pxGrid enabled on {node['hostname']}")

    # Configure pxGrid settings
    settings_url = f"https://{host}/admin/API/pxGrid/Settings"
    settings = {
        "automaticallyApproveCertBasedAccounts": True,
        "allowPasswordBasedAccounts": True,
        "autoEnableServices": True
    }

    settings_response = requests.put(
        settings_url,
        headers=headers,
        auth=auth,
        json=settings,
        verify=False
    )

    if settings_response.status_code == 200:
        print("✓ pxGrid settings configured")

    # Wait for service startup
    print("  Waiting for pxGrid service startup...")
    import time
    time.sleep(30)

    # Verify service status
    status_url = f"https://{host}/admin/API/pxGrid/ServiceStatus"
    status_response = requests.get(status_url, headers=headers, auth=au

    if status_response.status_code == 200:
        status = status_response.json()
        if status.get('serviceStatus') == 'running':
            print("✓ pxGrid service is running")
        else:
            print("⚠ pxGrid service status:", status.get('serviceStatus

if __name__ == "__main__":
    enable_pxgrid_service(
        os.environ['ISE_HOST'],
        os.environ['ISE_USER'],
        os.environ['ISE_PASS']
    )
```

4. Security Group Tags (SGT) Implementation

4.1 SGT Propagation Methods Analysis

Inline Tagging vs. SXP (SGT eXchange Protocol):

Method	Requirements	Performance	Coverage	Automation Level
Inline Tagging	Hardware support, IOS 16.9+	Highest	Best	High
SXP	Any device with SXP support	Good	Limited	Medium
Static Mapping	Manual configuration	N/A	Point	Low

4.2 Dynamic SGT Assignment Policies

```
# ansible/roles/ise_security_groups/dynamic_policies.yml
---
- name: Dynamic SGT Assignment Policy Automation
  hosts: ise_servers
  vars:
    dynamic_sgt_policies:
      - name: "Dynamic_by_LDAP_Group"
        ldap_group: "CN=Engineers,OU=Groups,DC=corp,DC=com"
        assigned_sgt: "Engineering_Team"
        priority: 100

      - name: "Dynamic_by_Machine_Certificate"
        certificate_issuer: "CN=Corporate_CA,DC=corp,DC=com"
        assigned_sgt: "Corporate_Machines"
        priority: 200

      - name: "Dynamic_by_MAC_OUI"
        oui_patterns: ["00:1A:2B:*", "AC:1F:6B:*"]
        assigned_sgt: "Corporate_Devices"
        priority: 300

  tasks:
    - name: Create dynamic SGT assignment policy
      cisco.ise.authorization_policy:
        ise_hostname: "{{ ise_hostname }}"
        state: present
        name: "{{ item.name }}"
        rule:
          type: "Dynamic"
          condition: "Security_Group:Dynamic"
          assignments:
            - type: "SGT"
              value: "{{ item.assigned_sgt }}"
          priority: "{{ item.priority }}"
        stateful_conditions:
          - "{{ item.ldap_group | default(omit) }}"
          - "{{ item.certificate_issuer | default(omit) }}"
          - "{{ item.oui_patterns | default(omit) }}"
      loop: "{{ dynamic_sgt_policies }}"
```

4.3 TrustSec Matrix Configuration

```
# terraform/modules/trustsec-matrix/main.tf
resource "ciscoise_trustsec_egress_policy" "matrix_policies" {
  for_each = var.trustsec_matrix

  source_sgt_id = ciscoise_sgt.security_groups[each.value.source_sgt].i
  destination_sgt_id = ciscoise_sgt.security_groups[each.value.destinat
  matrix_cell_status = each.value.status
  default_rule = each.value.default_rule

  sgacl = each.value.sgacl_enabled ? {
    name = "${each.key}-SGACL"
    content = templatefile("${path.module}/templates/sgacl_${each.key}.
      source_sgt = each.value.source_sgt
      destination_sgt = each.value.destination_sgt
      actions = each.value.actions
    })
  } : null

  contract = each.value.contract_enabled ? {
    name = "${each.key}-Contract"
    scope = "STANDARD"
    payload = each.value.contract_payload
  } : null
}

# Example TrustSec Matrix Configuration
variable "trustsec_matrix" {
  type = map(object({
    source_sgt      = string
    destination_sgt = string
    status         = string  # "ALLOW", "DENY", "MONITOR"
    default_rule   = string
    sgacl_enabled  = bool
    sgacl_content  = string
    contract_enabled = bool
    contract_payload = string
  }))

  default = {
    "employees_to_servers" = {
      source_sgt      = "Employees"
      destination_sgt = "Servers"
      status         = "ALLOW"
      default_rule   = "PERMIT_IP"
      sgacl_enabled  = true
      sgacl_content  = "permit ip tag 100 tag 200"
      contract_enabled = false
      contract_payload = ""
    }
    "guests_to_internet" = {
      source_sgt      = "Guests"
      destination_sgt = "Internet"
      status          = "MONITOR"
      default_rule    = "DENY_IP"
      sgacl_enabled   = false
      sgacl_content   = ""
      contract_enabled = true
      contract_payload = "RESTRICT_INTERNET_ACCESS"
    }
  }
}
```

5. Catalyst Center Integration Deep Dive

5.1 ISE-Catalyst Center Integration Requirements

Prerequisites for Integration: - ERS (External RESTful Services) enabled on ISE - pxGrid service enabled on ISE nodes - Valid SSL certificates (or certificate acceptance) - FQDN resolution between systems - Network connectivity on required ports: - 443 - HTTPS for REST APIs - 8910 - pxGrid communication - 9060 - ERS API endpoint - 22 - SSH for device onboarding (optional)

Integration Architecture:

```
┌──────────────────────────────────────────────────────────────┐
│                  Catalyst Center 2.3.7+                       │
│                                                             │
│  ┌──────────────┐  ┌──────────────┐  ┌────────────────┐     │
│  │ Assurance    │  │ Fabric       │  │ Device         │     │
│  │              │  │ Controller   │  │ Onboarding     │     │
│  └──────┬───────┘  └──────┬───────┘  └────────┬───────┘     │
│         │                 │                   │            │
│─────────┼─────────────────┼───────────────────┼────────────│
│         ▼                 ▼                   ▼            │
│  ┌──────────────┐  ┌──────────────┐  ┌────────────────┐     │
│  │ AAA Server   │  │ SD-Access    │  │ Network       │     │
│  │ Integration  │  │ Policies     │  │ Discovery    │    │
│  └──────┬───────┘  └──────┬───────┘  └──────┬───────┘     │
└─────────┼─────────────────┼───────────────────┼─────────┘
          │                 │                   │
          │ pxGrid + API    │ REST API          │ NetFlow
          ▼                 ▼                   ▼
┌─────────────────────────────────────────────────────────────┐
│                    Cisco ISE 3.3+                            │
│                                                              │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐     │
│  │ pxGrid       │  │ RADIUS/      │  │ Endpoint     │     │
│  │ Publisher    │  │ TACACS+      │  │ Database     │    │
│  └──────┬───────┘  └──────┬───────┘  └──────┬───────┘     │
└─────────┴─────────────────┴───────────────────┴─────────┘
```

5.2 Automated Integration Workflow

```
 # terraform/catalyst-integration.tf
# Complete ISE-Catalyst Center integration automation

# Create SDA fabric site
resource "catalystcenter_sda_fabric_site" "main_fabric" {
  site_name_hierarchy = "Global/Region1/Site1"
  fabric_type         = "FABRIC_SITE"
}

# Configure ISE as AAA server
resource "catalystcenter_aaa_server_ise" "ise_integration" {
  hostname         = var.ise_primary_hostname
  fqdn             = var.ise_primary_fqdn
  username         = var.ise_username
  password         = var.ise_password
  shared_secret    = var.radius_shared_secret
  subscriber_name  = "pxgrid_client_${var.deployment_name}"

  # Enable pxGrid integration
  pxgrid_enabled   = true
  pxgrid_node_role = "ALL"  # PAN, MNT, PSN options

  # Certificate management
  certificate_status = "TRUSTED"
  auto_accept_certs  = true

  # Integration timeout
  integration_timeout = 300
}

# Create integration workflow
resource "catalystcenter_ise_integration_workflow" "main_workflow" {
  ise_server_id = catalystcenter_aaa_server_ise.ise_integration.id
  workflow_type = "COMPLETE_INTEGRATION"

  validation_steps = [
    "ERS_CHECK",
    "PXGRID_CHECK",
    "CERTIFICATE_VALIDATION",
    "NODE_DISCOVERY",
    "SGT_SYNC"
  ]

  retry_count            = 3
  retry_interval_seconds = 30

  lifecycle {
    create_before_destroy = true
  }
}

# Synchronize SGTs from ISE to Catalyst Center
resource "catalystcenter_ise_sgt_sync" "sgt_synchronization" {
  depends_on = [catalystcenter_ise_integration_workflow.main_workflow]

  ise_server_id = catalystcenter_aaa_server_ise.ise_integration.id
  sync_type     = "FULL_SYNC"
  filters {
    include_sgt_range = [5, 65535]  # Valid SGT range
    exclude_sgt_tags  = [1, 2, 3]   # Reserved tags
  }

  schedule {
    enabled      = true
    interval_minutes = 60
    timezone     = "UTC"
  }
}
```

5.3 SD-Access Fabric Integration

```
# terraform/sda-fabric-integration.tf
# Complete SD-Access fabric provisioning with ISE integration

# Create SDA fabric control plane
resource "catalystcenter_sda_fabric_control_plane" "control_plane" {
  fabric_name = var.fabric_name
  site_name_hierarchy = var.site_hierarchy

  control_plane_nodes = [
    {
      device_ip = var.fabric_edge_node_1_ip
      device_name = var.fabric_edge_node_1_name
      roles = ["EDGE", "CONTROL_PLANE"]
    },
    {
      device_ip = var.fabric_edge_node_2_ip
      device_name = var.fabric_edge_node_2_name
      roles = ["EDGE", "CONTROL_PLANE"]
    }
  ]
  authentication_profile {
    aaa_servers = [catalystcenter_aaa_server_ise.ise_integration.id]
    default_role = "ACCESSIBLE"
    security_groups = data.ciscoise_sgt.all_sgts.value
  }
}

# Configure virtual networks with SGT integration
resource "catalystcenter_sda_virtual_network" "vn_production" {
  fabric_name = catalystcenter_sda_fabric_control_plane.control_plane.f
  virtual_network_name = "PROD_VN"

  security_groups = [
    "Employees",
    "Contractors",
    "Guests",
    "Servers",
    "IoT_Devices"
  ]

  ip_pool_info {
    ip_pool_name = "PROD_IPPOOL"
    scalable_group_info {
      name = "Employees"
      id   = 100
    }
  }

  # Enable TrustSec enforcement
  enable_trustsec = true
  sgacl_enforcement = "STRICT"
}

# Configure fabric sites with ISE policies
resource "catalystcenter_sda_fabric_site" "fabric_sites" {
  for_each = var.fabric_sites
  site_name_hierarchy = each.value.hierarchy
  fabric_type = "FABRIC_SITE"

  ise_integration_config {
    ise_server_id = catalystcenter_aaa_server_ise.ise_integration.id
    use_pxgrid = true
    use_rest_api = true

    policy_integration {
      enable_authorization_policies = true
      enable_security_groups = true
      enable_device_compliance = true
    }
  }
}
```

6. Production Deployment Strategies

6.1 High Availability Configurations

```
# terraform/ha-configuration.tf
# Multi-node ISE HA deployment with Terraform

# Primary Administration Node (PAN)
module "ise_primary_pan" {
  source = "./modules/ise-node"

  node_name = "ise-pan-primary"
  node_type = "ADMIN"
  ip_address = "10.1.1.10"
  domain = "corp.example.com"

  personas = ["Admin", "Policy Service", "Monitoring"]
  is_primary = true
  # pxGrid configuration
  enable_pxgrid = true
  pxgrid_certificate_arn = aws_acm_certificate.pxgrid_cert.arn

  # Load balancer configuration
  load_balancer_id = aws_lb.ise_alb.id
  target_group_ids = [
    aws_lb_target_group.pxgrid.id,
    aws_lb_target_group.radius.id,
    aws_lb_target_group.ers.id
  ]

  # Auto-scaling for high availability
  auto_scaling = {
    enabled = true
    min_size = 2
    max_size = 4
    desired_capacity = 3
  }
}

# Secondary PAN with automatic failover
module "ise_secondary_pan" {
  source = "./modules/ise-node"

  node_name = "ise-pan-secondary"
  node_type = "ADMIN"
  ip_address = "10.1.1.11"
  domain = "corp.example.com"

  personas = ["Admin", "Policy Service"]
  is_primary = false
  primary_node_ip = "10.1.1.10"

  # Automatic failover configuration
  failover_config = {
    enabled = true
    priority = 1
    replication_timeout = 300
  }
}

# Multiple PSN nodes for load balancing
module "ise_psn_nodes" {
  source = "./modules/ise-node"
  count  = var.psn_count  # Typically 3-5 nodes

  node_name = "ise-psn-${count.index + 1}"
  node_type = "PSN"
  ip_address = "10.1.1.${20 + count.index}"
  domain = "corp.example.com"

  personas = ["Policy Service"]
  enable_pxgrid = true

  # Load balancer registration
  register_with_elb = true
  target_groups = [
    "pxgrid-tg",
    "radius-tg",
    "ersapi-tg"
  ]
}
```

6.2 Disaster Recovery Planning

```
 # terraform/dr-configuration.tf
# Cross-region disaster recovery deployment

# Primary Region (us-west-2)
module "ise_primary_region" {
  source = "./modules/ise-infrastructure"
  region = "us-west-2"
  environment = "production"
  deployment_type = "HA_PRIMARY"

  network_config = {
    vpc_cidr = "10.1.0.0/16"
    subnets = [
      { cidr = "10.1.1.0/24", az = "us-west-2a", type = "public" },
      { cidr = "10.1.2.0/24", az = "us-west-2b", type = "private" }
    ]
  }

  ise_deployment = {
    pan_primary = { ip = "10.1.1.10", hostname = "ise-pan-west.example.
    pan_secondary = { ip = "10.1.1.11", hostname = "ise-pan-west2.examp
    psn_nodes = 3
    mnt_primary = { ip = "10.1.1.20", hostname = "ise-mnt-west.example.
  }
}

# DR Region (us-east-1)
module "ise_dr_region" {
  source = "./modules/ise-infrastructure"

  region = "us-east-1"
  environment = "production"
  deployment_type = "DR_STANDBY"

  network_config = {
    vpc_cidr = "10.2.0.0/16"
    subnets = [
      { cidr = "10.2.1.0/24", az = "us-east-1a", type = "public" },
      { cidr = "10.2.2.0/24", az = "us-east-1b", type = "private" }
    ]
  }
  ise_deployment = {
    pan_primary = { ip = "10.2.1.10", hostname = "ise-pan-east.example.
    psn_nodes = 2
    mnt_primary = { ip = "10.2.1.20", hostname = "ise-mnt-east.example.
  }
}

# Cross-region replication
resource "aws_rds_replication" "ise_db_replication" {
  source_db_instance_identifier = module.ise_primary_region.database_id
  target_db_instance_identifier = module.ise_dr_region.database_id
  replication_type = "cross-region"
  auto_start = true

  # Automated failover threshold
  failover_criteria {
    threshold = 5
    duration = 300
    metric = "replication_lag"
  }
}

# DNS failover configuration
resource "aws_route53_health_check" "ise_health_check" {
  fqdn = "ise.example.com"
  port = 443
  type = "HTTPS"
  resource_path = "/admin/API/mnt/HealthCheck"
  failure_threshold = "3"
  request_interval = "30"

  tags = {
    Name = "ISE-Health-Check"
  }
}

resource "aws_route53_record" "ise_dns_failover" {
  zone_id = data.aws_route53_zone.main.zone_id
  name = "ise.example.com"
  type = "A"

  set_identifier = "Primary"
  failover_routing_policy {
    type = "PRIMARY"
  }

  alias {
    name = module.ise_primary_region.load_balancer_dns
    zone_id = module.ise_primary_region.load_balancer_zone_id
    evaluate_target_health = true
  }

  health_check_id = aws_route53_health_check.ise_health_check.id
}

resource "aws_route53_record" "ise_dns_failover_secondary" {
  zone_id = data.aws_route53_zone.main.zone_id
  name = "ise.example.com"
  type = "A"

  set_identifier = "Secondary"
  failover_routing_policy {
    type = "SECONDARY"
  }

  alias {
    name = module.ise_dr_region.load_balancer_dns
    zone_id = module.ise_dr_region.load_balancer_zone_id
    evaluate_target_health = true
  }
}
```

6.3 Secrets Management with Vault Integration

```
# terraform/secrets-management.tf
# HashiCorp Vault integration for secure credentials

# Configure Vault for ISE secrets
resource "vault_mount" "ise_secrets" {
  path = "ise"
  type = "kv-v2"
  description = "Cisco ISE secrets storage"

  max_versions = 10
  delete_version_after = "30d"  # Automatic cleanup
}

# Store ISE credentials
resource "vault_kv_secret_v2" "ise_credentials" {
  mount = vault_mount.ise_secrets.path
  name  = "ise-primary"

  data_json = jsonencode({
    username                         = var.ise_admin_username
    password                         = var.ise_admin_password
    ers_api_user                     = var.ise_ers_username
    ers_api_password                 = var.ise_ers_password
    pxgrid_password                  = var.ise_pxgrid_password
    snmp_communities                 = var.snmp_communities
    radius_shared_secrets            = var.radius_shared_secrets
  })

  cas_required = true  # Require Check-And-Set for updates
}

# Dynamic provider authentication
data "vault_kv_secret_v2" "current_ise_creds" {
  depends_on = [vault_kv_secret_v2.ise_credentials]
  mount     = vault_mount.ise_secrets.path
  name      = "ise-primary"
}

# Use Vault secrets in provider configuration
provider "ciscoise" {
  username  = data.vault_kv_secret_v2.current_ise_creds.data["username"
  password  = data.vault_kv_secret_v2.current_ise_creds.data["password"
  base_url  = "https://${var.ise_primary_hostname}"
  ssl_verify = false
}

# Ansible Vault integration for playbooks
resource "null_resource" "ansible_vault_setup" {
  triggers = {
    vault_addr = var.vault_address
    vault_path = vault_kv_secret_v2.ise_credentials.name
  }

  provisioner "local-exec" {
    command = <<-EOT
      # Create encrypted Ansible Vault file from Vault data
      vault kv get -format=json ${vault_mount.ise_secrets.path}/data/${
      jq -r '.data' > /tmp/ise_credentials.json

      # Create encrypted vault file
      echo -n '${var.vault_password}' | ansible-vault create --vault-pa
        ${path.root}/ansible/group_vars/ise_servers/vault.yml

      # Update vault ID in Ansible configuration
      echo "vault_password_file = $(pwd)/.vault_pass" >> ansible/ansibl
    EOT

    environment = {
      VAULT_ADDR = var.vault_address
      VAULT_TOKEN = var.vault_token
    }
  }
}
```

6.4 CI/CD Pipeline Integration

```
# .github/workflows/terraform-validate.yml
name: Terraform Validation
on:
  push:
    paths: ['terraform/**']
  pull_request:
    paths: ['terraform/**']

env:
  TF_VERSION: "1.5.0"

jobs:
  terraform-validate:
    name: Validate Terraform
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v4

    - name: Setup Terraform
      uses: hashicorp/setup-terraform@v3
      with:
        terraform_version: ${{ env.TF_VERSION }}

    - name: Terraform Format Check
      working-directory: terraform
      run: terraform fmt -check -recursive
      continue-on-error: true

    - name: Terraform Init
```

```
      working-directory: terraform
      run: |
        terraform init -backend=false
        terraform version
```

```
# .github/workflows/ansible-validation.yml
name: Ansible Validation
on:
  push:
    paths: ['ansible/**']
  pull_request:
    paths: ['ansible/**']

jobs:
  ansible-lint:
    name: Ansible Lint
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v4

    - name: Setup Python
      uses: actions/setup-python@v4
      with:
        python-version: '3.9'

    - name: Install dependencies
      run: |
        pip install ansible ansible-lint ciscoisesdk

    - name: Install Ansible collections
      run: ansible-galaxy collection install cisco.ise

    - name: Run Ansible syntax check
      working-directory: ansible
      run: |
        ansible-playbook --syntax-check site.yml
        ansible-lint playbooks/ roles/ --skip-list 301 303 305
```

7. Cloud Deployment Strategies

7.1 AWS Multi-Node ISE Deployment

```
# terraform/environments/aws/main.tf
# Complete AWS ISE deployment with best practices

module "ise_vpc" {
  source = "terraform-aws-modules/vpc/aws"

  name = "${var.deployment_name}-ise-vpc"
  cidr = "10.0.0.0/16"

  azs             = ["us-west-2a", "us-west-2b", "us-west-2c"]
  private_subnets = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]
  public_subnets  = ["10.0.101.0/24", "10.0.102.0/24", "10.0.103.0/24"]

  enable_nat_gateway = true
  enable_vpn_gateway = true
  enable_dns_hostnames = true
  enable_dns_support = true

  tags = {
    Terraform = "true"
    Environment = var.environment
    Service = "Cisco ISE"
  }
}

# Security Groups for ISE
resource "aws_security_group" "ise_sg" {
  name = "${var.deployment_name}-ise-sg"
  description = "Security group for Cisco ISE nodes"
  vpc_id = module.ise_vpc.vpc_id

  # pxGrid communication
  ingress {
    from_port   = 8910
    to_port     = 8910
    protocol    = "tcp"
    cidr_blocks = ["10.0.0.0/16"]
    description = "pxGrid communication"
  }

  # RADIUS authentication
  ingress {
    from_port   = 1812
    to_port     = 1813
    protocol    = "udp"
    cidr_blocks = ["10.0.0.0/8"]
    description = "RADIUS authentication"
  }

  # Administrative access
  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["10.0.0.0/16", "192.168.0.0/16"]  # Management netwo
    description = "HTTPS administrative access"
  }

  # ERS API access
  ingress {
    from_port   = 9060
    to_port     = 9060
    protocol    = "tcp"
    cidr_blocks = ["10.0.0.0/16"]
    description = "ERS API access"
  }
}

# ISE Nodes Deployment
module "ise_pan_primary" {
  source = "../../../modules/ise-node-aws"

  node_name = "ise-pan-primary"
  node_type = "PAN_PRIMARY"
  instance_type = "c5.2xlarge"  # ISE 3.3+ requirement

  vpc_id = module.ise_vpc.vpc_id
  subnet_id = module.ise_vpc.private_subnets[0]
  security_group_id = aws_security_group.ise_sg.id

  # EBS configuration for persistent storage
  root_volume_size = 200  # GB
  root_volume_type = "gp3"
  root_volume_iops = 3000

  # pxGrid certificate
  pxgrid_certificate_arn = aws_acm_certificate.pxgrid_cert.arn

  tags = {
    Environment = var.environment
    NodeType = "PAN_PRIMARY"
    Terraform = "true"
  }
}

# Load Balancer for high availability
resource "aws_lb" "ise_alb" {
  name               = "${var.deployment_name}-ise-alb"
  internal           = true
  load_balancer_type = "application"
  security_groups    = [aws_security_group.ise_sg.id]
  subnets            = module.ise_vpc.public_subnets

  enable_deletion_protection = true

  tags = {
    Environment = var.environment
    Terraform = "true"
  }
}
```

7.2 Azure Multi-Region Deployment

```
 # terraform/environments/azure/main.tf
# Azure-based ISE deployment with disaster recovery

# Resource Groups
resource "azurerm_resource_group" "ise_rg" {
  name     = "${var.deployment_name}-ise-rg"
  location = var.primary_region
  tags     = var.common_tags
}

# Virtual Networks
resource "azurerm_virtual_network" "ise_vnet" {
  name                = "${var.deployment_name}-ise-vnet"
  address_space       = ["10.0.0.0/16"]
  location            = azurerm_resource_group.ise_rg.location
  resource_group_name = azurerm_resource_group.ise_rg.name
  tags                = var.common_tags
}

# Subnets for ISE
resource "azurerm_subnet" "ise_subnet" {
  name                 = "ise-subnet"
  resource_group_name  = azurerm_resource_group.ise_rg.name
  virtual_network_name = azurerm_virtual_network.ise_vnet.name
  address_prefixes     = ["10.0.1.0/24"]

  service_endpoints = ["Microsoft.Storage"]
}

# Network Security Groups
resource "azurerm_network_security_group" "ise_nsg" {
  name                = "${var.deployment_name}-ise-nsg"
  location            = azurerm_resource_group.ise_rg.location
  resource_group_name = azurerm_resource_group.ise_rg.name

  security_rule {
    name                       = "pxgrid"
    priority                   = 100
    direction                  = "Inbound"
    access                     = "Allow"
    protocol                   = "Tcp"
    source_port_range          = "*"
    destination_port_range     = "8910"
    source_address_prefix      = "10.0.0.0/16"
    destination_address_prefix = "*"
  }

  security_rule {
    name                       = "radius"
    priority                   = 110
    direction                  = "Inbound"
    access                     = "Allow"
    protocol                   = "Udp"
    source_port_range          = "*"
    destination_port_range     = "1812-1813"
    source_address_prefix      = "*"
    destination_address_prefix = "*"
  }

  tags = var.common_tags
}
```

```
# Load Balancer
resource "azurerm_lb" "ise_private_lb" {
  name                = "${var.deployment_name}-ise-lb"
  location            = azurerm_resource_group.ise_rg.location
  resource_group_name = azurerm_resource_group.ise_rg.name

  frontend_ip_configuration {
    name      = "private"
    subnet_id = azurerm_subnet.ise_subnet.id
    private_ip_address_allocation = "Dynamic"
  }

  tags = var.common_tags
}

# ISE Virtual Machines
resource "azurerm_linux_virtual_machine" "ise_nodes" {
  for_each = var.ise_nodes

  name                = "ise-${each.key}-vm"
  location            = azurerm_resource_group.ise_rg.location
  resource_group_name = azurerm_resource_group.ise_rg.name
  availability_set_id = azurerm_availability_set.ise_as.id
  size                = "Standard_D8s_v3"

  admin_username = "iseadmin"
  admin_password = var.ise_admin_password

  network_interface_ids = [azurerm_network_interface.ise_nics[each.key]

  os_disk {
    caching              = "ReadWrite"
    storage_account_type = "Premium_LRS"
    disk_size_gb         = 200
  }
  source_image_reference {
    publisher = "cisco"
    offer     = "cisco-ise"
    sku       = "ise-3x"
    version   = "latest"
  }

  tags = merge(var.common_tags, {
    NodeType = each.value.type
    NodeRole = each.value.role
  })
}

# Backup and Monitoring
resource "azurerm_recovery_services_vault" "ise_backup" {
  name                = "${var.deployment_name}-ise-backup"
  location            = azurerm_resource_group.ise_rg.location
  resource_group_name = azurerm_resource_group.ise_rg.name
  sku                = "Standard"

  soft_delete_enabled = true
  encryption {
    key_id = azurerm_key_vault_key.backup_key.id
  }
}
```

⚡ 8. Performance and Optimization

8.1 API Rate Limiting and Bulk Operations

```
 # scripts/ise_bulk_operations.py
# Bulk operations with rate limiting and optimization

import asyncio
import aiohttp
from aiohttp import BasicAuth
from collections import deque
import time
from typing import List, Dict
import logging

class ISEAsyncBulkOperations:
    def __init__(self, host: str, username: str, password: str, max_con
        self.host = host
        self.auth = BasicAuth(username, password)
        self.max_concurrent = max_concurrent
        self.rate_limiter = asyncio.Semaphore(max_concurrent)
        self.session = None
        self.rate_limit_queue = deque(maxlen=100)

    async def __aenter__(self):
        connector = aiohttp.TCPConnector(limit=100, limit_per_host=20)
        timeout = aiohttp.ClientTimeout(total=120, connect=30)
        self.session = aiohttp.ClientSession(
            connector=connector,
            timeout=timeout,
            auth=self.auth
        )
        return self

    async def __aexit__(self, exc_type, exc_val, exc_tb):
        if self.session:
            await self.session.close()

    async def _rate_limited_request(self, method: str, url: str, **kwar
        """Make API request with rate limiting"""
        async with self.rate_limiter:
            # Rate limiting with exponential backoff
            await asyncio.sleep(0.1)  # Base delay

            # Circuit breaker pattern
            if len(self.rate_limit_queue) >= 90:  # 90% capacity
                oldest_request = self.rate_limit_queue[0]
                if time.time() - oldest_request < 1.0:
                    await asyncio.sleep(0.5)  # Exponential backoff

            self.rate_limit_queue.append(time.time())

            try:
                async with self.session.request(method, url, **kwargs)
                    if response.status == 429:  # Rate limit exceeded
                        retry_after = int(response.headers.get('Retry-A
                        logging.warning(f"Rate limit exceeded, waiting
                        await asyncio.sleep(retry_after)
                        return await self._rate_limited_request(method,

                    response.raise_for_status()
                    return await response.json()

            except aiohttp.ClientError as e:
                logging.error(f"Request failed: {e}")
                raise

    async def bulk_create_sgts(self, sgts: List[Dict]) -> List[Dict]:
        """Bulk create Security Group Tags with concurrent processing""
        tasks = []

        for sgt in sgts:
            # Batch processing with optimal chunk size
            url = f"https://{self.host}/ers/config/sgt"
            payload = {
                "Sgt": {
                    "name": sgt['name'],
                    "description": sgt['description'],
                    "value": sgt['value'],
                    "propogateToApic": sgt.get('propagate_to_apic', Fal
                }
            }
            task = self._rate_limited_request('POST', url, json=payload
            tasks.append(task)

        # Process with concurrency control
        results = await asyncio.gather(*tasks, return_exceptions=True)

        successful = []
        failed = []

        for i, result in enumerate(results):
            if isinstance(result, Exception):
                logging.error(f"Failed to create SGT {sgts[i]['name']}:
                failed.append({'sgt': sgts[i], 'error': str(result)})
            else:
                successful.append({'sgt': sgts[i], 'response': result})

        return {
            'successful': successful,
            'failed': failed,
            'total_processed': len(sgts),
            'success_rate': len(successful) / len(sgts)
        }

# Usage example
async def main():
    sgts = [
        {"name": f"SGT_{i}", "description": f"Test SGT {i}", "value": i
        for i in range(1000, 1100)
    ]

    bulk_results = await bulk_create_sgts(sgts)
    print(f"Success rate: {bulk_results['success_rate']:.2%}")

if __name__ == "__main__":
    asyncio.run(main())
```

8.2 Terraform Execution Optimization

```
 # terraform/performance-optimizations.tf
# Performance optimizations for large-scale deployments

# Parallel resource creation
resource "ciscoise_network_device" "network_devices" {
  for_each = { for device in var.network_devices : device.name => devic

  name                 = each.key
  description          = each.value.description
  network_device_ip     = each.value.ip_address
  mask                 = each.value.mask
  network_device_groups = each.value.groups

  # Parallel execution strategy
  lifecycle {
    create_before_destroy = true
    replace_triggered_by  = [each.value.replace_trigger]

    # Resource dependency optimization
    depends_on = [
      ciscoise_sgt.security_groups,  # Create SGTs first
      ciscoise_active_directory.ad  # Ensure AD is available
    ]
  }
}

# Dynamic resource limits based on deployment size
variable "resource_limits" {
  description = "Resource creation limits for performance"
  type = object({
    max_parallel_resources = number
    api_rate_limit = number
    timeout_seconds = number
  })
  default = {
    max_parallel_resources = 20  # Adjust based on ISE performance
    api_rate_limit = 100  # Requests per minute
    timeout_seconds = 300  # 5 minutes timeout
  }
}

# Use data sources to minimize API calls
data "ciscoise_sgt_list" "all_sgts" {
  # Cache SGT list to avoid multiple lookups
}

data "ciscoise_network_device_list" "all_devices" {
  # Cache device list for policy references
}

# Batch policy creation
resource "ciscoise_authorization_policy" "batch_policies" {
  for_each = { for policy in chunklist(var.authorization_policies, 10)[
               policy.name => policy }
  count = length(chunklist(var.authorization_policies, 10))

  name = each.value.name
  condition = each.value.condition
  profiles = each.value.profiles
  security_group = each.value.security_group

  # Batch processing with delays
  lifecycle {
    create_before_destroy = true
  }
}

resource "time_sleep" "api_cooldown" {
  for_each = toset(range(length(chunklist(var.authorization_policies, 1

  create_duration = "30s"
  triggers = {
    batch_id = each.key
  }
}
```

8.3 Ansible Playbook Performance Tuning

```
# ansible/ansible.cfg - Performance optimizations
[defaults]
# Performance settings
strategy = mitogen_linear  # Faster than default
strategy_plugins = ~/.ansible/plugins/strategy:/usr/share/ansible/plugi
gathering = smart
fact_caching = jsonfile
fact_caching_connection = /tmp/ansible_facts_cache
fact_caching_timeout = 86400

# Parallel execution
forks = 50
serial = 0  # No limit on parallel execution
timeout = 30

# Connection optimization
pipelining = True
ssh_args = -o ControlMaster=auto -o ControlPersist=60s -o ControlPath=/
transport = smart

# Variable and task optimization
host_key_checking = False
retry_files_enabled = False
allow_world_readable_tmpfiles = True
remote_tmp = /tmp/.ansible-${USER}/tmp

# Callback plugins for performance monitoring
stdout_callback = yaml
callback_whitelist = timer, profile_tasks, profile_roles
```

```
# Plugin optimization
action_plugins = ~/.ansible/plugins/action:/usr/share/ansible/plugins/a
callback_plugins = ~/.ansible/plugins/callback:/usr/share/ansible/plugi
connection_plugins = ~/.ansible/plugins/connection:/usr/share/ansible/p
lookup_plugins = ~/.ansible/plugins/lookup:/usr/share/ansible/plugins/l
vars_plugins = ~/.ansible/plugins/vars:/usr/share/ansible/plugins/vars
```

```yaml

# ansible/playbooks/high-performance-site.yml

---

- name: High-Performance ISE Configuration Deployment hosts: ise_servers gather_facts: no serial: "{{ ansible_play_hosts | length }}" # All nodes in parallel

vars: # Chunk large configurations for better performance chunk_size: 50 max_workers: 4

tasks: - name: Configure SGTs with batch processing include_tasks: batch_sgt_process.yml vars: sgt_batch: "{{ security_groups | batch(chunk_size) }}" loop: "{{ sgt_batch }}" loop_control: label: "Processing batch {{ ansible_loop.index }}" loop_var: batch

```
 - name: Configure policies with async execution
   cisco.ise.authorization_policy:
     ise_hostname: "{{ ise_hostname }}"
     state: present
     name: "{{ item.name }}"
     rule: "{{ item.rule }}"
     profiles: "{{ item.profiles }}"
```
```

```
    loop: "{{ authorization_policies }}"
    async: 300  #
```