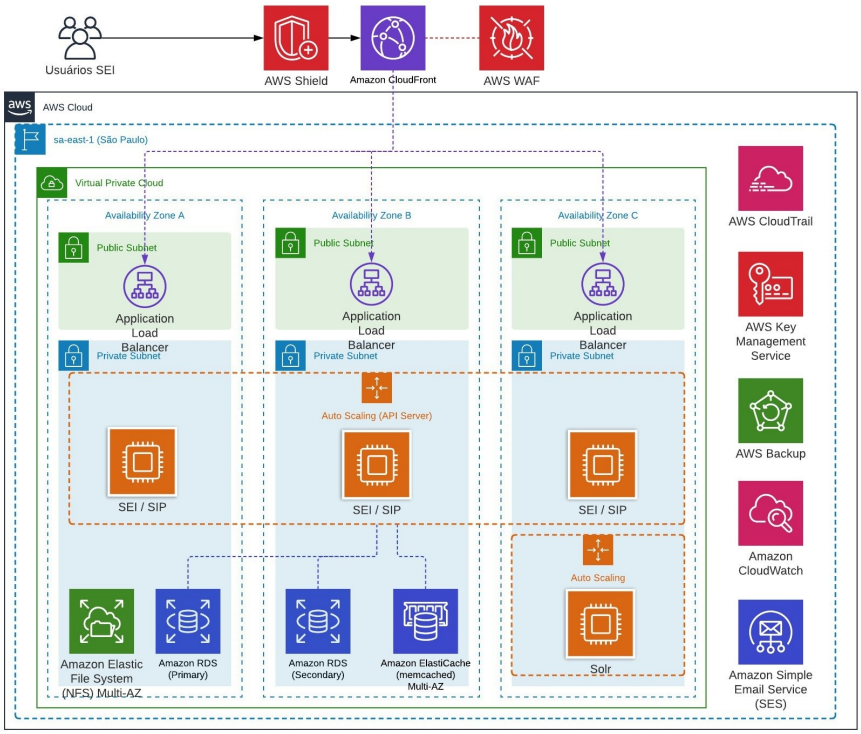
Relatório de análise da solução: tmp63nkblys



Análise completa da solução atual

Modelo de cloud:
- AWS (região sa-east-1 – São Paulo)

Lista com os componentes:
- Usuários (clientes/colaboradores)
- AWS Shield
- Amazon CloudFront
- AWS WAF
- VPC (Virtual Private Cloud) com 3 AZs: A, B e C
- Subnets públicas (em cada AZ)
- Subnets privadas (em cada AZ)
- Application Load Balancer (ALB) distribuído em múltiplas subnets públicas
- Auto Scaling (API Server)
- Servidores de aplicação SEI/SIP (auto scaling) em subnets privadas
- Cluster de busca Solr (auto scaling) em subnets privadas
- Amazon Elastic File System (EFS – NFS) Multi-AZ
- Amazon RDS (Primary em AZ A, Secondary em AZ B)
- Amazon ElastiCache (Memcached) Multi-AZ
- AWS CloudTrail
- AWS Key Management Service (KMS)
- AWS Backup
- Amazon CloudWatch
- Amazon Simple Email Service (SES)
- (Gestão) AWS Systems Manager/“Management Service” para operações/automação

Interação entre os componentes:
- Usuários -> CloudFront -> WAF -> ALB: CloudFront acelera e protege o tráfego; WAF aplica regras L7; Shield mitiga DDoS.
- ALB -> Auto Scaling (API Server) -> SEI/SIP: o balanceador distribui requisições para instâncias privadas que executam a aplicação.
- SEI/SIP -> RDS: leitura/escrita transacional no banco relacional (Multi-AZ).
- SEI/SIP -> EFS: armazenamento de arquivos/artefatos compartilhados (NFS).
- SEI/SIP -> ElastiCache: cache de sessões/objetos (Memcached).
- SEI/SIP <-> Solr: indexação e busca full-text.
- SEI/SIP -> SES: envio de e-mails transacionais/notificações.
- Observabilidade e segurança: CloudWatch (métricas/logs/alarms), CloudTrail (auditoria API), KMS

(chaves de criptografia), AWS Backup (políticas de backup), Systems Manager (patching/parameters/automação).

O que esse sistema faz:
- Hospeda uma aplicação web corporativa de alta disponibilidade com API e busca full-text (Solr), banco relacional (RDS), cache (ElastiCache) e armazenamento de documentos/artefatos compartilhados (EFS). Pelo rótulo SEI/SIP, aparenta ser um sistema de gestão eletrônica de informações/documentos.

Vulnerabilidades e Solução para cara vulnerabilidade:
1) Bypass do WAF/CloudFront acessando diretamente o ALB
- Risco: tráfego malicioso atingir o ALB por IP direto.
- Solução: restringir SG do ALB aos prefixos gerenciados do CloudFront; validar cabeçalho de origem (custom header) no ALB; usar ACLs/NACLs conforme necessário.

2) Criptografia em repouso e em trânsito inconsistente
- Risco: dados em EFS/RDS/volumes sem criptografia; TLS fraco entre camadas.
- Solução: habilitar criptografia com KMS (RDS/EFS/EBS/CloudTrail/CloudWatch Logs/Backups); TLS 1.2+ ponta a ponta; HSTS no CloudFront; rotation e separação de CMKs.

3) Memcached sem autenticação e com suporte limitado a criptografia
- Risco: vazamento/manipulação de dados de cache.
- Solução: isolar por SG/subnet privada; considerar migração para ElastiCache Redis com TLS, AUTH e ACLs; aplicar timeouts/limites de tamanho.

4) Exposição indevida do EFS (NFS)
- Risco: montagem não autorizada e exfiltração.
- Solução: SG do EFS apenas para SG das instâncias; EFS Access Points + POSIX permissons; criptografia em trânsito (TLS para NFS mount helpers) e em repouso; políticas de backup e lifecycle.

5) Falta de hardening nas instâncias de aplicação
- Risco: exploração por vulnerabilidades do SO/app.
- Solução: IMDSv2 obrigatório; SSM para patching e CIS hardening; princípio do menor privilégio em IAM roles; egress control (SG/NAT + VPC egress only); antivírus/EDR; secrets em Secrets Manager/SSM Parameter Store.

6) Solr exposto internamente sem controles
- Risco: RCE/DoS em endpoints de admin; perda de índices.
- Solução: SG restritivo; autenticação/autorização no Solr; TLS interno; limitar handlers perigosos; backups/replicação; cgroup/ulimits contra queries caras.

7) Configuração de WAF insuficiente
- Risco: SQLi/XSS/SSRF/Bot/abuso de credenciais.
- Solução: WAF com AWS Managed Rules + regras custom; rate limiting; Bot Control; geo/rules por URI; logging no Kinesis Firehose + análises.

8) Monitoramento/auditoria inadequados
- Risco: detecção tardia de incidentes.
- Solução: CloudTrail org-level com log file validation; CloudWatch alarms; métricas de saúde do Auto Scaling/ALB; integrar com GuardDuty/Security Hub/Detective.

9) Backups sem imutabilidade ou sem cross-region
- Risco: ransomware/indisponibilidade regional.
- Solução: AWS Backup com Vault Lock (imutável), PITR no RDS, cópias cross-region, testes de restauração, RPO/RTO documentados.

10) Gestão de chaves e segredos frágil
- Risco: vazamento de segredos; chaves sem rotação.
- Solução: AWS KMS com rotation; Secrets Manager com rotação automática e políticas de acesso mínimas; nunca embutir segredos em AMIs/containers.

11) Limitação de alta disponibilidade de DNS/CDN
- Risco: falha de resolução/origin único.
- Solução: Route 53 com health checks e failover; CloudFront com múltiplos origins e failover; ALB cross-zone load balancing.

12) SES mal configurado
- Risco: reputação/entregabilidade e spoofing.
- Solução: SPF, DKIM e DMARC alinhados; feedback loops e tratamento de bounces/complaints;

chaves dedicadas por domínio; quotas monitoradas.

13) Falta de VPC Endpoints
- Risco: tráfego de gerenciamento saindo à internet.
- Solução: VPC endpoints (Gateway/Interface) para S3, SSM, CloudWatch, KMS, Secrets Manager, SES (PrivateLink onde aplicável).

14) Ausência de limites de taxa e proteção contra DoS na aplicação
- Risco: exaustão de recursos no backend.
- Solução: rate limiting no WAF/ALB e no app; filas/bulkheads; circuit breakers; timeouts; auto scaling com limites seguros.

Gere um Relatório de Modelagem de Ameaças, baseado na metodologia STRIDE:
- Spoofing (Falsificação de identidade)
  - Ameaças: falsificação de origem para pular CloudFront; tokens JWT roubados; acesso não autenticado a Solr/ElastiCache.
  - Mitigações: ALB restrito a CloudFront; mTLS interno entre serviços críticos; OAuth2/OIDC com rotinas de revogação; Redis (no lugar de Memcached) com AUTH/TLS; assinatura de URLs no CloudFront; validação de IP/ASN conforme necessário.

- Tampering (Adulteração)
  - Ameaças: alteração de objetos em EFS/RDS; adulteração de logs.
  - Mitigações: criptografia KMS + políticas de acesso mínimas; controle de versão e WORM (Backup Vault Lock); CloudTrail com log integrity validation; checagens de integridade de arquivos (hashing) e trilhas de auditoria no app.

- Repudiation (Repúdio)
  - Ameaças: ações sem trilhas; impossibilidade de atribuir autoria.
  - Mitigações: CloudTrail em todas as contas/regiões, Logs do ALB/CloudFront/WAF; logs de app com correlação (trace IDs); sincronização de tempo; retenção adequada; integração SIEM.

- Information Disclosure (Divulgação de informação)
  - Ameaças: dados sensíveis em trânsito sem TLS; caches vazando dados; headers sensíveis vazados pelo CDN.
  - Mitigações: TLS 1.2+ end-to-end; headers de segurança (HSTS, CSP, XFO) via CloudFront/ALB; segregação de cache por user/session; encriptação de logs; data masking nos logs.

- Denial of Service (Negação de serviço)
  - Ameaças: DDoS na borda; queries pesadas no Solr; bursts que esgotam conexões no RDS/ALB.
  - Mitigações: AWS Shield (Advanced recomendado), WAF rate-based; autoscaling com limites; connection pooling; read replicas/failover; proteção a “slowloris” e timeouts agressivos no ALB.

- Elevation of Privilege (Elevação de privilégio)
  - Ameaças: papéis IAM permissivos; SSRF para acessar IMDS; execução remota em Solr ou servidores.
  - Mitigações: IAM least privilege com SCPs; IMDSv2 + bloquear 169.254.169.254 via egress/iptables; segmentação de segurança por SG/NACL; varreduras SAST/DAST; patching contínuo com SSM; segregação de funções (admin vs runtime).

Additional resourcing needs
- Network Administrator: validar segmentação (VPC, subnets, NACLs, SGs), rotas, endpoints privados, DNS/Route 53 e requisitos de conectividade híbrida.
- Security Officer: revisar conformidade e políticas (IAM, KMS, WAF, Shield, Backup/Retention, registros de auditoria), aprovar baseline de segurança e exceções.
- Cloud/DevOps staff: IaC (CloudFormation/Terraform), pipelines CI/CD, observabilidade (CloudWatch, GuardDuty), automação SSM, controles de egress/ingress e custos.
- Software Developer: contratos de API, autenticação/autorização, validação de entrada, rate limiting, gestão de sessões, instrumentação de logs/traces.
- Hardware Developer: se houver appliances/IoT, revisar armazenamento seguro de credenciais, secure boot, attestation e atualizações OTA.
- Sugestão de artefatos antes/durante STRIDE: diagrama UML/DFD com fronteiras de confiança; inventário de dados (sensibilidade/classificação); fluxos de credenciais; mapa de portas/protocolos; matriz de riscos (probabilidade x impacto) com plano de tratamento; testes de restauração de backup e exercícios de tabletop de incidentes.

Sugestões de melhorias

A seguir apresento:

1) Diagrama mermaid da arquitetura com relações e anotações de correções de segurança
2) Relatório STRIDE (ameaças e mitigações)
3) Terraform “end-to-end” base (fundação) para criar a solução na AWS sa-east-1, com os principais controles de segurança incorporados (WAF, CloudFront, ALB com proteção contra bypass, VPC 3 AZs, subnets públicas/privadas, Auto Scaling, EFS, RDS Multi-AZ, ElastiCache Redis com TLS/AUTH, CloudTrail, KMS, Backup Vault Lock, VPC Endpoints, SSM/Secrets).
Observação: alguns itens exigem ajustes de domínio/conta e/ou habilitação prévia (ex.: certificados ACM, SES, Shield Advanced).

Diagrama mermaid

```mermaid
flowchart LR
  %% Fronteiras de confiança
  subgraph Internet["Internet / Borda (CDN/DDoS)"]
    U[Usuários (Clientes/Colaboradores)]
    CF[Amazon CloudFront\n- TLS 1.2+\n- HSTS/CSP/XFO headers\n- Origin custom header\n- Failover p/ S3 manutenção]
    WAF[AWS WAF (CLOUDFRONT)\n- AWS Managed Rules\n- Rate limiting\n- Bot Control\n- Geo/URI rules\n- Logging]
    SH[AWS Shield (Adv recomendado)]
  end

  subgraph DNS["DNS público"]
    R53[Amazon Route 53\n- Alias para CloudFront\n- SPF/DKIM/DMARC (SES)]
  end

  U -->|HTTPS| CF
  CF -->|evalua| WAF
  SH --- CF
  R53 --- CF

  subgraph VPC["VPC (3 AZs: A/B/C) - sa-east-1"]
    direction TB
    subgraph Public["Subnets Públicas (A/B/C)"]
      ALB[(Application Load Balancer)\n- SG: permite SÓ prefixo CloudFront\n- Listener regra: exige header secreto\n- Cross-zone LB ON]
    end

    subgraph Private["Subnets Privadas (A/B/C)"]
      ASG[Auto Scaling - App SEI/SIP\n- IMDSv2\n- SSM/Hardening\n- SG mínimos]
      SOLR[Cluster Solr (Auto Scaling)\n- SG estrito\n- TLS/auth\n- Handlers limitados]
      EFS[(Amazon EFS - Multi-AZ)\n- SG: só do SG App\n- Access Points\n- Enc. repouso+trânsito]
      RDS[(Amazon RDS - Multi-AZ)\n- Enc. KMS\n- SG: só App\n- PITR]
      REDIS[(ElastiCache Redis - Multi-AZ)\n- TLS+AUTH/ACLs\n- SG: só App]
      VPCE[Interface/GW Endpoints: S3, SSM, SSMMessages,\nEC2Messages, Logs, KMS, Secrets, SES]
      NAT[NAT Gateways (A/B/C)]
    end
    IGW[(Internet Gateway)]
    RTpub[Rotas públicas -> IGW]
    RTpriv[Rotas privadas -> NAT + VPCE]

    IGW <--> RTpub
    NAT <--> RTpriv
    ALB <--> RTpub
    ASG <--> RTpriv
    SOLR <--> RTpriv
    EFS <--> RTpriv
    RDS <--> RTpriv
    REDIS <--> RTpriv
    VPCE <--> RTpriv
  end

  CF -->|HTTPS (Origin)\n+ header secreto| ALB
  ALB -->|HTTP/TLS interno| ASG
  ASG -->|Leit/Escr| RDS
  ASG -->|NFS| EFS
  ASG -->|Cache| REDIS
  ASG <--> SOLR
```

```
  ASG -->|SES API| VPCE

  subgraph Observabilidade_Segurança["Observabilidade / Segurança"]
    CW[CloudWatch\n- Logs/Metrics/Alarms]
    CT[CloudTrail Org/multi-region\n- Log integrity\n- KMS]
    KMS[AWS KMS\n- Chaves dados/logs\n- Rotação]
    BKP[AWS Backup\n- Vault Lock (WORM)\n- Cross-region copy]
    SSM[AWS Systems Manager\n- Patch/Runbooks/Params\n- Session Manager]
    Guard[GuardDuty / Security Hub / Detective]
    Firehose[Kinesis Firehose (WAF logs opcional)]
  end

  WAF -->|Logs| Firehose
  ALB -->|Access Logs| CW
  CF -->|Logs| CW
  ASG -->|App logs/metrics| CW
  RDS -->|Enhanced metrics| CW
  CT -->|S3+KMS| CW
  SSM --- ASG
  KMS --- RDS
  KMS --- EFS
  KMS --- CW
  KMS --- BKP
  Guard --- CT

  subgraph Email["E-mail transacional"]
    SES[Amazon SES\n- SPF/DKIM/DMARC\n- Bounces/Complaints]
  end

  ASG --> SES
  R53 --> SES

  %% Observações de correção de vulnerabilidades
  note right of ALB
    Correções:
    - SG do ALB restrito a prefixo CloudFront
    - Listener exige header secreto (custom)
    - TLS 1.2+ ponta a ponta
  end
  note bottom of Private
    Controles:
    - SG mínimos entre camadas
    - VPC Endpoints p/ mgmt/telemetria
    - Sem acesso direto da Internet
  end
  note left of REDIS
    Redis substitui Memcached:
    - TLS/AUTH/ACLs
    - Timeouts/limites configurados
  end
  note right of EFS
    EFS Access Points + POSIX
    Montagem via TLS
  end
  note right of SOLR
    Admin protegido, TLS/auth,
    handlers perigosos desabilitados
  end
```

Correções principais incluídas na arquitetura
- Bypass WAF/CloudFront: SG do ALB limita origem ao managed prefix list do CloudFront; ALB exige header secreto; CloudFront injeta header secreto; NACLs conforme necessário.
- Criptografia: KMS em RDS/EFS/EBS/CloudTrail/CloudWatch Logs/Backup; TLS 1.2+ CloudFront→ALB→App→RDS/Redis/Solr/EFS; HSTS no CloudFront.
- ElastiCache: uso de Redis com TLS/AUTH/ACLs em vez de Memcached.
- EFS: SG apenas do App; Access Points; TLS no mount helper; backup/lifecycle.
- Hardening: IMDSv2; SSM para patching; least-privilege IAM; egress control; secrets em Secrets Manager/SSM.

- Solr: SG restrito; auth/TLS; handlers limitados; políticas de backup/replicação.
- WAF: Managed rules + custom + rate limit + Bot Control + logging.
- Monitoramento/Auditoria: CloudTrail org/multi-region com validation; CloudWatch métricas/alarms; GuardDuty/Security Hub/Detective.
- Backup: AWS Backup com Vault Lock e cópia cross-region; RDS PITR; testes de restauração.
- Chaves/Segredos: KMS com rotação; Secrets Manager com rotação; nada em AMIs.
- DNS/CDN HA: Route 53 + CloudFront com origin failover; ALB cross-zone.
- SES: SPF/DKIM/DMARC; tratamento de bounces/complaints; monitorar quotas.
- VPC Endpoints: S3, SSM, Logs, KMS, Secrets, SES.
- Proteção contra DoS: WAF rate-based; Shield; timeouts/circuit breakers no app; autoscaling com limites.

Relatório de Modelagem de Ameaças (STRIDE)
- Spoofing (Falsificação)
  - Ameaças: origem falsificada direto no ALB; tokens JWT roubados; acesso a Solr/Redis sem auth.
  - Mitigações: ALB restrito a prefixos CloudFront + header secreto obrigatório; mTLS interno para serviços críticos; OAuth2/OIDC com revogação; Redis com TLS/AUTH/ACL; assinaturas de URL no CloudFront; validação de IP/ASN quando aplicável.
- Tampering (Adulteração)
  - Ameaças: alteração indevida em EFS/RDS; adulteração de logs.
  - Mitigações: criptografia KMS + IAM mínimo; EFS Access Points/POSIX; Backup Vault Lock (WORM); CloudTrail com log file validation; hashing/integridade de arquivos no app.
- Repudiation (Repúdio)
  - Ameaças: ações sem trilha; autoria não atribuível.
  - Mitigações: CloudTrail multi-region/org; logs de ALB/CloudFront/WAF; logs de app com trace-id e correlação; NTP/Time Sync; retenção adequada; integração SIEM.
- Information Disclosure (Divulgação)
  - Ameaças: dados sem TLS; vazamento via cache; headers sensíveis no CDN.
  - Mitigações: TLS 1.2+ ponta a ponta; headers de segurança (HSTS, CSP, X-Frame-Options) no CloudFront; segregação de cache por usuário; criptografia de logs; data masking.
- Denial of Service (DoS)
  - Ameaças: DDoS na borda; queries pesadas no Solr; bursts esgotando RDS/ALB.
  - Mitigações: Shield (Adv); WAF rate-based e Bot Control; autoscaling com limites; connection pooling; RDS Multi-AZ + read replicas; timeouts agressivos no ALB (idle/slowloris).
- Elevation of Privilege (Elevação de privilégio)
  - Ameaças: IAM permissivo; SSRF→IMDS; RCE em Solr/servidores.
  - Mitigações: IAM least-privilege + SCPs; IMDSv2 e bloqueio de 169.254.169.254 via iptables/egress; segmentação por SG/NACL; SAST/DAST; patching contínuo via SSM; separação de funções (admin vs runtime).

Terraform (base de referência)
Notas importantes:
- Ajuste os valores de variáveis em variables.tf (domínio, tamanhos, senhas).
- Pré-requisitos: domínio público no Route 53; subscription do Shield Advanced (opcional); conta habilitada SES em produção; limites/quotas verificados; chaves KMS e roles podem requerer políticas adicionais.
- Para CloudFront, o certificado ACM deve estar em us-east-1. Para o ALB (origin), use ACM em sa-east-1 (domínio interno/externo).
- O código ilustra os principais controles. Pode ser dividido em módulos na sua base.

main.tf
```hcl
terraform {
  required_version = ">= 1.6.0"
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = ">= 5.50"
    }
    random = {
      source  = "hashicorp/random"
      version = ">= 3.5"
    }
  }
}

provider "aws" {
  region = var.region
}
```

```
# Para ACM/CloudFront em us-east-1
provider "aws" {
  alias  = "us_east_1"
  region = "us-east-1"
}

data "aws_caller_identity" "current" {}
data "aws_partition" "current" {}
data "aws_region" "current" {}
data "aws_availability_zones" "available" {
  state = "available"
}

locals {
  name             = var.project_name
  vpc_cidr         = var.vpc_cidr
  azs              = slice(data["aws_availability_zones"]["available"].names, 0, 3)
  public_subnet_cidrs  = var.public_subnet_cidrs
  private_subnet_cidrs = var.private_subnet_cidrs

  # Header secreto CloudFront->ALB; valor em SSM
  waf_bypass_header_name  = "X-Origin-Secret"
  cf_origin_header_name   = local.waf_bypass_header_name
}

# KMS keys
resource "aws_kms_key" "data" {
  description             = "${local.name} data key (RDS/EFS/EBS)"
  enable_key_rotation     = true
  deletion_window_in_days = 30
}

resource "aws_kms_key" "logs" {
  description             = "${local.name} logs key (CloudTrail/Logs/Backup)"
  enable_key_rotation     = true
  deletion_window_in_days = 30
}

# VPC
resource "aws_vpc" "this" {
  cidr_block           = local.vpc_cidr
  enable_dns_support   = true
  enable_dns_hostnames = true
  tags = { Name = "${local.name}-vpc" }
}

resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.this.id
  tags = { Name = "${local.name}-igw" }
}

# Subnets (3 AZs)
resource "aws_subnet" "public" {
  for_each                = { for idx, az in local.azs : az => idx }
  vpc_id                  = aws_vpc.this.id
  availability_zone       = each.key
  cidr_block              = local.public_subnet_cidrs[each.value]
  map_public_ip_on_launch = true
  tags = { Name = "${local.name}-public-${each.key}", Tier = "public" }
}

resource "aws_subnet" "private" {
  for_each          = { for idx, az in local.azs : az => idx }
  vpc_id            = aws_vpc.this.id
  availability_zone = each.key
  cidr_block        = local.private_subnet_cidrs[each.value]
  tags = { Name = "${local.name}-private-${each.key}", Tier = "private" }
}
```

```
# Route tables
resource "aws_route_table" "public" {
  vpc_id = aws_vpc.this.id
  tags = { Name = "${local.name}-public-rt" }
}
resource "aws_route" "public_igw" {
  route_table_id         = aws_route_table.public.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id             = aws_internet_gateway.igw.id
}
resource "aws_route_table_association" "public_assoc" {
  for_each       = aws_subnet.public
  subnet_id      = each.value.id
  route_table_id = aws_route_table.public.id
}

# NAT Gateway por AZ
resource "aws_eip" "nat" {
  for_each = aws_subnet.public
  domain   = "vpc"
  tags     = { Name = "${local.name}-eip-nat-${each.key}" }
}
resource "aws_nat_gateway" "nat" {
  for_each      = aws_subnet.public
  allocation_id = aws_eip.nat[each.key].id
  subnet_id     = each.value.id
  tags          = { Name = "${local.name}-nat-${each.key}" }
  depends_on    = [aws_internet_gateway.igw]
}

resource "aws_route_table" "private" {
  for_each = aws_subnet.private
  vpc_id   = aws_vpc.this.id
  tags     = { Name = "${local.name}-private-rt-${each.key}" }
}
resource "aws_route" "private_nat" {
  for_each               = aws_route_table.private
  route_table_id         = each.value.id
  destination_cidr_block = "0.0.0.0/0"
  nat_gateway_id         = aws_nat_gateway.nat[replace(each.key, "private", "public")].id
}
resource "aws_route_table_association" "private_assoc" {
  for_each       = aws_subnet.private
  subnet_id      = each.value.id
  route_table_id = aws_route_table.private[each.key].id
}

# VPC Endpoints (privados)
resource "aws_vpc_endpoint" "s3" {
  vpc_id           = aws_vpc.this.id
  service_name     = "com.amazonaws.${data.aws_region.current.name}.s3"
  vpc_endpoint_type = "Gateway"
  route_table_ids  = [for rt in aws_route_table.private : rt.id]
  tags = { Name = "${local.name}-vpce-s3" }
}

locals {
  interface_endpoints = toset([
    "ssm",
    "ssmmessages",
    "ec2messages",
    "logs",
    "kms",
    "secretsmanager",
    "ecr.api",
    "ecr.dkr",
    "ses"
  ])
```

```
}

resource "aws_security_group" "vpce" {
  name        = "${local.name}-vpce"
  description = "SG for VPC interface endpoints"
  vpc_id      = aws_vpc.this.id

  ingress {
    description = "From private subnets"
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = [for s in aws_subnet.private : s.cidr_block]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = [aws_vpc.this.cidr_block]
  }
  tags = { Name = "${local.name}-vpce" }
}

resource "aws_vpc_endpoint" "interface" {
  for_each            = local.interface_endpoints
  vpc_id              = aws_vpc.this.id
  service_name        = "com.amazonaws.${data.aws_region.current.name}.${each.key}"
  vpc_endpoint_type   = "Interface"
  subnet_ids          = [for s in aws_subnet.private : s.id]
  security_group_ids  = [aws_security_group.vpce.id]
  private_dns_enabled = true
  tags                = { Name = "${local.name}-vpce-${each.key}" }
}

# Prefix list do CloudFront (proteger ALB)
data "aws_ec2_managed_prefix_list" "cloudfront" {
  name = "com.amazonaws.global.cloudfront.origin-facing"
}

# Security Groups
resource "aws_security_group" "alb" {
  name        = "${local.name}-alb"
  vpc_id      = aws_vpc.this.id
  description = "ALB only from CloudFront origin-facing"

  ingress {
    from_port                = 443
    to_port                  = 443
    protocol                 = "tcp"
    description              = "CloudFront only"
    prefix_list_ids          = [data.aws_ec2_managed_prefix_list.cloudfront.id]
  }

  # Opcional: 80 se necessário para health checks internos
  ingress {
    from_port       = 80
    to_port         = 80
    protocol        = "tcp"
    description     = "CloudFront (HTTP) - se necessário"
    prefix_list_ids = [data.aws_ec2_managed_prefix_list.cloudfront.id]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = [aws_vpc.this.cidr_block]
  }
```

```
  tags = { Name = "${local.name}-alb" }
}

resource "aws_security_group" "app" {
  name   = "${local.name}-app"
  vpc_id = aws_vpc.this.id

  ingress {
    description     = "From ALB"
    from_port       = var.app_port
    to_port         = var.app_port
    protocol        = "tcp"
    security_groups = [aws_security_group.alb.id]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = [aws_vpc.this.cidr_block]
  }
  tags = { Name = "${local.name}-app" }
}

resource "aws_security_group" "solr" {
  name   = "${local.name}-solr"
  vpc_id = aws_vpc.this.id

  ingress {
    description     = "From App"
    from_port       = 8983
    to_port         = 8983
    protocol        = "tcp"
    security_groups = [aws_security_group.app.id]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = [aws_vpc.this.cidr_block]
  }
  tags = { Name = "${local.name}-solr" }
}

resource "aws_security_group" "efs" {
  name   = "${local.name}-efs"
  vpc_id = aws_vpc.this.id

  ingress {
    description     = "NFS from App"
    from_port       = 2049
    to_port         = 2049
    protocol        = "tcp"
    security_groups = [aws_security_group.app.id]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = [aws_vpc.this.cidr_block]
  }
  tags = { Name = "${local.name}-efs" }
}

resource "aws_security_group" "rds" {
  name   = "${local.name}-rds"
```

```
  vpc_id = aws_vpc.this.id

  ingress {
    description     = "DB from App"
    from_port       = var.db_port
    to_port         = var.db_port
    protocol        = "tcp"
    security_groups = [aws_security_group.app.id]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = [aws_vpc.this.cidr_block]
  }
  tags = { Name = "${local.name}-rds" }
}

resource "aws_security_group" "redis" {
  name   = "${local.name}-redis"
  vpc_id = aws_vpc.this.id

  ingress {
    description     = "Redis TLS from App"
    from_port       = 6379
    to_port         = 6379
    protocol        = "tcp"
    security_groups = [aws_security_group.app.id]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = [aws_vpc.this.cidr_block]
  }
  tags = { Name = "${local.name}-redis" }
}

# EFS
resource "aws_efs_file_system" "this" {
  creation_token   = "${local.name}-efs"
  encrypted        = true
  kms_key_id       = aws_kms_key.data.arn
  performance_mode = "generalPurpose"
  throughput_mode  = "bursting"
  tags = { Name = "${local.name}-efs" }
}

resource "aws_efs_mount_target" "mt" {
  for_each        = aws_subnet.private
  file_system_id  = aws_efs_file_system.this.id
  subnet_id       = each.value.id
  security_groups = [aws_security_group.efs.id]
}

resource "aws_efs_access_point" "ap" {
  file_system_id = aws_efs_file_system.this.id
  posix_user {
    gid = 1000
    uid = 1000
  }
  root_directory {
    path = "/app"
    creation_info {
      owner_gid   = 1000
      owner_uid   = 1000
      permissions = "0750"
```

```
    }
  }
  tags = { Name = "${local.name}-efs-ap" }
}

# RDS (Multi-AZ)
resource "random_password" "db" {
  length  = 20
  special = true
}

resource "aws_secretsmanager_secret" "db" {
  name = "${local.name}/db/password"
  kms_key_id = aws_kms_key.logs.id
}

resource "aws_secretsmanager_secret_version" "db" {
  secret_id     = aws_secretsmanager_secret.db.id
  secret_string = random_password.db.result
}

resource "aws_db_subnet_group" "db" {
  name       = "${local.name}-db-subnets"
  subnet_ids = [for s in aws_subnet.private : s.id]
}

resource "aws_db_instance" "db" {
  identifier                = "${local.name}-db"
  engine                    = var.db_engine
  engine_version              = var.db_engine_version
  instance_class              = var.db_instance_class
  allocated_storage           = 100
  max_allocated_storage       = 500
  multi_az                  = true
  db_subnet_group_name        = aws_db_subnet_group.db.name
  vpc_security_group_ids      = [aws_security_group.rds.id]
  username                  = var.db_username
  password                  = random_password.db.result
  port                      = var.db_port
  storage_encrypted           = true
  kms_key_id                = aws_kms_key.data.arn
  backup_retention_period     = 7
  deletion_protection         = true
  apply_immediately           = true
  publicly_accessible         = false
  auto_minor_version_upgrade  = true
  enabled_cloudwatch_logs_exports = ["postgresql", "upgrade"] # ajuste para engine
}

# ElastiCache Redis (TLS/AUTH)
resource "random_password" "redis_auth" {
  length  = 32
  special = false
}

resource "aws_elasticache_subnet_group" "redis" {
  name       = "${local.name}-redis-subnets"
  subnet_ids = [for s in aws_subnet.private : s.id]
}

resource "aws_elasticache_replication_group" "redis" {
  replication_group_id          = "${local.name}-redis"
  description                 = "Redis with TLS/AUTH"
  engine                      = "redis"
  engine_version                = var.redis_engine_version
  node_type                   = var.redis_node_type
  number_cache_clusters         = 2
  automatic_failover_enabled    = true
  multi_az_enabled              = true
```

```
  subnet_group_name          = aws_elasticache_subnet_group.redis.name
  security_group_ids         = [aws_security_group.redis.id]
  at_rest_encryption_enabled = true
  transit_encryption_enabled = true
  auth_token                 = random_password.redis_auth.result
  port                       = 6379
  apply_immediately          = true
}

# Launch Template para App (SEI/SIP)
data "aws_ami" "al2" {
  most_recent = true
  owners      = ["amazon"]
  filter { name = "name"; values = ["amzn2-ami-hvm-*-x86_64-gp2"] }
}

resource "aws_iam_role" "ec2_app" {
  name = "${local.name}-ec2-app"
  assume_role_policy = jsonencode({
    Version = "2012-10-17",
    Statement = [{ Effect="Allow", Principal={Service="ec2.amazonaws.com"},
Action="sts:AssumeRole"}]
  })
}
resource "aws_iam_role_policy_attachment" "ec2_app_ssm" {
  role       = aws_iam_role.ec2_app.name
  policy_arn =
"arn:${data.aws_partition.current.partition}:iam::aws:policy/AmazonSSMManagedInstanceCore"
}
resource "aws_iam_role_policy_attachment" "ec2_app_cloudwatch" {
  role       = aws_iam_role.ec2_app.name
  policy_arn =
"arn:${data.aws_partition.current.partition}:iam::aws:policy/CloudWatchAgentServerPolicy"
}

resource "aws_iam_instance_profile" "ec2_app" {
  name = "${local.name}-ec2-app"
  role = aws_iam_role.ec2_app.name
}

# Header secreto armazenado no SSM
resource "random_password" "origin_header" {
  length  = 32
  special = false
}
resource "aws_ssm_parameter" "origin_header" {
  name  = "/${local.name}/origin/header"
  type  = "SecureString"
  value = random_password.origin_header.result
  key_id = aws_kms_key.logs.id
}

resource "aws_launch_template" "app" {
  name_prefix   = "${local.name}-app-"
  image_id      = data.aws_ami.al2.id
  instance_type = var.app_instance_type
  iam_instance_profile { name = aws_iam_instance_profile.ec2_app.name }
  vpc_security_group_ids = [aws_security_group.app.id]
  metadata_options {
    http_tokens = "required" # IMDSv2
  }
  user_data = base64encode(templatefile("${path.module}/userdata-app.sh", {
    efs_fs_id    = aws_efs_file_system.this.id
    efs_ap_id    = aws_efs_access_point.ap.id
    redis_endpoint = aws_elasticache_replication_group.redis.primary_endpoint_address
    db_endpoint  = aws_db_instance.db.address
    db_port      = var.db_port
  }))
  tag_specifications {
```

```
    resource_type = "instance"
    tags = { Name = "${local.name}-app" }
  }
}

resource "aws_autoscaling_group" "app" {
  name                = "${local.name}-app-asg"
  desired_capacity    = var.app_desired
  min_size            = var.app_min
  max_size            = var.app_max
  vpc_zone_identifier = [for s in aws_subnet.private : s.id]
  health_check_type   = "ELB"
  health_check_grace_period = 120

  launch_template {
    id      = aws_launch_template.app.id
    version = "$Latest"
  }

  tag {
    key                 = "Name"
    value               = "${local.name}-app"
    propagate_at_launch = true
  }
}

# Target Group + ALB + Listener com header secreto obrigatório
resource "aws_lb_target_group" "app" {
  name     = "${local.name}-tg-app"
  port     = var.app_port
  protocol = "HTTP"
  vpc_id   = aws_vpc.this.id
  health_check {
    path                = "/health"
    matcher             = "200-399"
    healthy_threshold   = 2
    unhealthy_threshold = 3
    interval            = 15
    timeout             = 5
  }
}

resource "aws_autoscaling_attachment" "app" {
  autoscaling_group_name = aws_autoscaling_group.app.name
  alb_target_group_arn   = aws_lb_target_group.app.arn
}

resource "aws_lb" "alb" {
  name               = "${local.name}-alb"
  load_balancer_type = "application"
  subnets            = [for s in aws_subnet.public : s.id]
  security_groups    = [aws_security_group.alb.id]
  enable_deletion_protection = true
}

# Certificado do ALB (ajuste o domínio/ACM)
data "aws_acm_certificate" "alb" {
  domain   = var.origin_domain_name
  statuses = ["ISSUED"]
  most_recent = true
}

resource "aws_lb_listener" "https" {
  load_balancer_arn = aws_lb.alb.arn
  port              = 443
  protocol          = "HTTPS"
  ssl_policy        = "ELBSecurityPolicy-TLS13-1-2-2021-06"
  certificate_arn   = data.aws_acm_certificate.alb.arn
```

```
  # Ação padrão: bloquear caso header não presente/errado
  default_action {
    type = "fixed-response"
    fixed_response {
      content_type = "text/plain"
      message_body = "Forbidden"
      status_code  = "403"
    }
  }
}

# Regra que SÓ encaminha se header secreto correto
data "aws_ssm_parameter" "origin_header_val" {
  name = aws_ssm_parameter.origin_header.name
  with_decryption = true
}
resource "aws_lb_listener_rule" "only_cf_with_header" {
  listener_arn = aws_lb_listener.https.arn
  priority     = 10

  condition {
    http_header {
      http_header_name = local.waf_bypass_header_name
      values           = [data.aws_ssm_parameter.origin_header_val.value]
    }
  }

  action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.app.arn
  }
}

# CloudFront + WAF
resource "aws_cloudfront_origin_access_control" "oac" {
  name                              = "${local.name}-oac"
  description                       = "OAC for ALB origin"
  origin_access_control_origin_type = "web"
  signing_behavior                  = "always"
  signing_protocol                  = "sigv4"
}

# Certificado para o domínio público (us-east-1)
data "aws_acm_certificate" "cf" {
  provider = aws.us_east_1
  domain   = var.cloudfront_domain_name
  statuses = ["ISSUED"]
  most_recent = true
}

# WAFv2 WebACL (CLOUDFRONT)
resource "aws_wafv2_web_acl" "cf" {
  name        = "${local.name}-cf-webacl"
  description = "WAF for CloudFront"
  scope       = "CLOUDFRONT"

  default_action { allow {} }

  rule {
    name     = "AWS-AWSManagedRulesCommonRuleSet"
    priority = 1
    override_action { none {} }
    statement {
      managed_rule_group_statement {
        name        = "AWSManagedRulesCommonRuleSet"
        vendor_name = "AWS"
      }
    }
    visibility_config {
```

```
      cloudwatch_metrics_enabled = true
      metric_name                = "common"
      sampled_requests_enabled   = true
    }
  }

  rule {
    name     = "RateLimit"
    priority = 2
    statement {
      rate_based_statement {
        limit              = var.waf_rate_limit
        aggregate_key_type = "IP"
      }
    }
    action { block {} }
    visibility_config {
      cloudwatch_metrics_enabled = true
      metric_name                = "ratelimit"
      sampled_requests_enabled   = true
    }
  }

  visibility_config {
    cloudwatch_metrics_enabled = true
    metric_name                = "cf-webacl"
    sampled_requests_enabled   = true
  }
}

resource "aws_cloudfront_distribution" "this" {
  enabled             = true
  is_ipv6_enabled     = true
  aliases             = [var.cloudfront_domain_name]
  default_root_object = "index.html"

  origins {
    domain_name = aws_lb.alb.dns_name
    origin_id   = "alb-origin"
    origin_path = ""

    custom_origin_config {
      origin_protocol_policy = "https-only"
      https_port             = 443
      origin_ssl_protocols   = ["TLSv1.2"]
    }

    origin_custom_headers {
      name  = local.waf_bypass_header_name
      value = data.aws_ssm_parameter.origin_header_val.value
    }

    origin_access_control_id = aws_cloudfront_origin_access_control.oac.id
  }

  # Failover opcional para S3 (página de manutenção)
  dynamic "origin_group" {
    for_each = var.enable_cf_failover ? [1] : []
    content {
      origin_id = "group1"
      failover_criteria {
        status_codes = [500, 502, 503, 504]
      }
      members {
        items = ["alb-origin", "s3-maintenance"]
      }
    }
  }
```

```
  default_cache_behavior {
    target_origin_id       = var.enable_cf_failover ? "group1" : "alb-origin"
    viewer_protocol_policy = "redirect-to-https"
    allowed_methods        = ["GET","HEAD","OPTIONS","PUT","POST","PATCH","DELETE"]
    cached_methods         = ["GET","HEAD"]
    compress               = true

    forwarded_values {
      query_string = true
      headers      = ["Authorization", "Origin", "Content-Type"]
      cookies { forward = "all" }
    }

    function_associations {
      event_type   = "viewer-response"
      function_arn = var.cf_security_headers_function_arn
    }
  }

  price_class = "PriceClass_200"

  viewer_certificate {
    acm_certificate_arn = data.aws_acm_certificate.cf.arn
    ssl_support_method  = "sni-only"
    minimum_protocol_version = "TLSv1.2_2021"
  }

  restrictions {
    geo_restriction {
      restriction_type = "none"
    }
  }

  web_acl_id = aws_wafv2_web_acl.cf.arn

  depends_on = [aws_lb_listener_rule.only_cf_with_header]
}

# Shield Advanced (opcional; requer assinatura)
# resource "aws_shield_protection" "alb" {
#   name         = "${local.name}-alb-protection"
#   resource_arn = aws_lb.alb.arn
# }
# resource "aws_shield_protection" "cf" {
#   name         = "${local.name}-cf-protection"
#   resource_arn = aws_cloudfront_distribution.this.arn
# }

# Route 53 registros
resource "aws_route53_record" "cf" {
  zone_id = var.public_hosted_zone_id
  name    = var.cloudfront_domain_name
  type    = "A"

  alias {
    name                   = aws_cloudfront_distribution.this.domain_name
    zone_id                = aws_cloudfront_distribution.this.hosted_zone_id
    evaluate_target_health = false
  }
}

# CloudTrail
resource "aws_s3_bucket" "cloudtrail" {
  bucket = "${local.name}-cloudtrail-${data.aws_caller_identity.current.account_id}"
  force_destroy = false
}

resource "aws_s3_bucket_versioning" "cloudtrail" {
  bucket = aws_s3_bucket.cloudtrail.id
```

```
  versioning_configuration { status = "Enabled" }
}

resource "aws_s3_bucket_server_side_encryption_configuration" "cloudtrail" {
  bucket = aws_s3_bucket.cloudtrail.id
  rule {
    apply_server_side_encryption_by_default {
      kms_master_key_id = aws_kms_key.logs.arn
      sse_algorithm     = "aws:kms"
    }
    bucket_key_enabled = true
  }
}

resource "aws_cloudtrail" "this" {
  name                          = "${local.name}-trail"
  s3_bucket_name                = aws_s3_bucket.cloudtrail.id
  include_global_service_events = true
  is_multi_region_trail         = true
  enable_log_file_validation    = true
  kms_key_id                    = aws_kms_key.logs.arn
}

# AWS Backup com Vault Lock
resource "aws_backup_vault" "this" {
  name        = "${local.name}-vault"
  kms_key_arn = aws_kms_key.logs.arn
}

resource "aws_backup_vault_lock_configuration" "lock" {
  backup_vault_name   = aws_backup_vault.this.name
  changeable_for_days = 7
  min_retention_days  = 7
  max_retention_days  = 365
}

resource "aws_backup_plan" "this" {
  name = "${local.name}-plan"
  rule {
    rule_name         = "daily"
    target_vault_name = aws_backup_vault.this.name
    schedule          = "cron(0 5 * * ? *)"
    lifecycle {
      delete_after = 120
    }
    copy_action {
      destination_vault_arn = var.cross_region_backup_vault_arn
    }
  }
}

resource "aws_backup_selection" "efs" {
  name         = "efs-selection"
  iam_role_arn = var.backup_role_arn
  plan_id      = aws_backup_plan.this.id

  resources = [
    aws_efs_file_system.this.arn
  ]
}

resource "aws_backup_selection" "rds" {
  name         = "rds-selection"
  iam_role_arn = var.backup_role_arn
  plan_id      = aws_backup_plan.this.id

  resources = [
    aws_db_instance.db.arn
  ]
```

```
}

# SES (identidade de domínio; requer verificação/produção)
resource "aws_ses_domain_identity" "this" {
  domain = var.email_domain
}

resource "aws_route53_record" "ses_verification" {
  zone_id = var.public_hosted_zone_id
  name    = "_amazonses.${var.email_domain}"
  type    = "TXT"
  ttl     = 600
  records = [aws_ses_domain_identity.this.verification_token]
}

resource "aws_ses_domain_dkim" "this" {
  domain = aws_ses_domain_identity.this.domain
}

resource "aws_route53_record" "ses_dkim" {
  count   = 3
  zone_id = var.public_hosted_zone_id
  name    =
"${aws_ses_domain_dkim.this.dkim_tokens[count.index]}._domainkey.${var.email_domain}"
  type    = "CNAME"
  ttl     = 600
  records = ["${aws_ses_domain_dkim.this.dkim_tokens[count.index]}.dkim.amazonses.com"]
}

# SPF/DMARC
resource "aws_route53_record" "spf" {
  zone_id = var.public_hosted_zone_id
  name    = var.email_domain
  type    = "TXT"
  ttl     = 600
  records = ["v=spf1 include:amazonses.com -all"]
}
resource "aws_route53_record" "dmarc" {
  zone_id = var.public_hosted_zone_id
  name    = "_dmarc.${var.email_domain}"
  type    = "TXT"
  ttl     = 600
  records = ["v=DMARC1; p=quarantine; rua=mailto:${var.dmarc_rua}; ruf=mailto:${var.dmarc_ruf};
fo=1"]
}
```

variables.tf
```hcl
variable "region" {
  description = "AWS region"
  type        = string
  default     = "sa-east-1"
}

variable "project_name" { type = string }

variable "vpc_cidr" {
  type    = string
  default = "10.0.0.0/16"
}

# 3 blocos /24 públicos e 3 privados como exemplo
variable "public_subnet_cidrs" {
  type    = list(string)
  default = ["10.0.0.0/24","10.0.1.0/24","10.0.2.0/24"]
}
variable "private_subnet_cidrs" {
  type    = list(string)
```

```
  default = ["10.0.10.0/24","10.0.11.0/24","10.0.12.0/24"]
}

variable "app_port" {
  type    = number
  default = 8080
}

variable "app_instance_type" {
  type    = string
  default = "t3.medium"
}

variable "app_desired" { type = number, default = 3 }
variable "app_min"     { type = number, default = 3 }
variable "app_max"     { type = number, default = 9 }

variable "db_engine"         { type = string, default = "postgres" }
variable "db_engine_version" { type = string, default = "15.4" }
variable "db_instance_class" { type = string, default = "db.m6g.large" }
variable "db_port"           { type = number, default = 5432 }
variable "db_username"       { type = string, default = "appuser" }

variable "redis_engine_version" { type = string, default = "7.1" }
variable "redis_node_type"      { type = string, default = "cache.t4g.medium" }

# Domínio público (CloudFront) e domínio do origin (ALB) para ACM
variable "public_hosted_zone_id" { type = string }
variable "cloudfront_domain_name" {
  description = "FQDN público (ex.: app.example.com)"
  type        = string
}
variable "origin_domain_name" {
  description = "FQDN do origin/ALB (ex.: origin.example.com) com cert em sa-east-1"
  type        = string
}

# CloudFront Function/Lambda@Edge com headers de segurança (opcional)
variable "cf_security_headers_function_arn" {
  type        = string
  default     = null
}

variable "enable_cf_failover" { type = bool, default = false }

# Backup cross-region (informe o ARN do vault de destino)
variable "cross_region_backup_vault_arn" {
  type    = string
  default = null
}

variable "backup_role_arn" {
  type = string
}

# WAF
variable "waf_rate_limit" {
  type    = number
  default = 2000
}

# SES
variable "email_domain" { type = string }
variable "dmarc_rua"    { type = string, default = "security@example.com" }
variable "dmarc_ruf"    { type = string, default = "security@example.com" }
```

outputs.tf
```hcl
```

```
output "vpc_id" { value = aws_vpc.this.id }
output "alb_dns" { value = aws_lb.alb.dns_name }
output "cloudfront_domain" { value = aws_cloudfront_distribution.this.domain_name }
output "rds_endpoint" { value = aws_db_instance.db.address }
output "redis_primary_endpoint" { value =
aws_elasticache_replication_group.redis.primary_endpoint_address }
output "efs_id" { value = aws_efs_file_system.this.id }
output "origin_secret_header_param" { value = aws_ssm_parameter.origin_header.name }
```

userdata-app.sh (exemplo mínimo; ajuste para sua aplicação)

```bash
#!/bin/bash
set -euo pipefail

# Atualizações e agentes
yum update -y
yum install -y amazon-efs-utils
systemctl enable --now amazon-ssm-agent || true

# Monta EFS com TLS
mkdir -p /mnt/app
echo "${efs_fs_id}:/ /mnt/app efs _netdev,tls,accesspoint=${efs_ap_id} 0 0" >> /etc/fstab
mount -a -t efs defaults

# Exporta variáveis (use SSM/Secrets no app)
cat >/etc/profile.d/app.sh <<EOF
export REDIS_ENDPOINT="${redis_endpoint}"
export DB_ENDPOINT="${db_endpoint}"
export DB_PORT="${db_port}"
EOF

# Aqui: instalar SEI/SIP, configurar serviço, healthcheck /health
# systemctl enable --now sei.service
```

Próximos passos e validações
- Substituir domínios (origin_domain_name e cloudfront_domain_name) e provisionar certificados ACM (sa-east-1 e us-east-1).
- Habilitar SES em produção e concluir verificações DKIM/SPF/DMARC.
- (Opcional) Assinar Shield Advanced e ativar proteções.
- Integrar WAF logging (Kinesis Firehose) e SIEM.
- Configurar Solr com TLS/Auth e regras seguras; ajustar ASG do Solr (similar ao App) e SGs.
- Adicionar policies de rotação no Secrets Manager (DB/Redis).
- Acrescentar alarmes CloudWatch (ALB 5xx, ASG CPU, RDS connections, Redis evictions) e dashboards.
- Realizar testes de restauração de backups e tabletop de incidentes.
- Documentar RPO/RTO, mapa de dados, fluxos de credenciais e matriz de riscos.