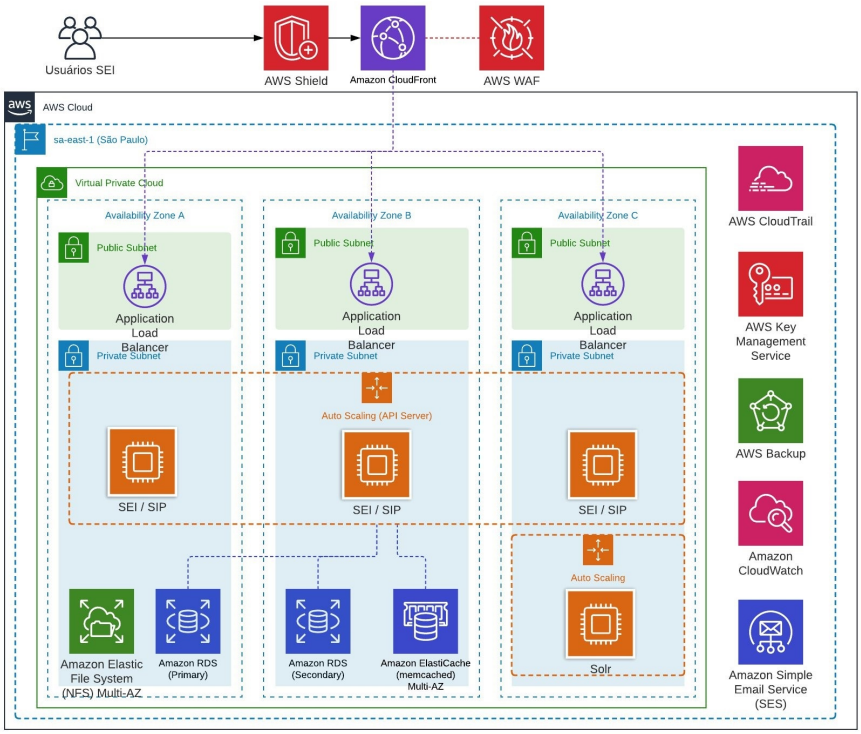
Relatório de análise da solução: Cloud_aws



Análise completa da solução atual

Modelo de cloud:
- AWS (Amazon Web Services). A imagem mostra serviços e ícones oficiais como AWS Shield, Amazon CloudFront, AWS WAF, VPC, ALB, Auto Scaling, Amazon RDS, ElastiCache, EFS, CloudTrail, KMS, Backup, CloudWatch e SES na região sa-east-1 (São Paulo).

Lista com os componentes:
- Usuários SEI (clientes/usuários finais na internet)
- Borda e proteção:
  - AWS Shield (DDoS)
  - Amazon CloudFront (CDN)
  - AWS WAF (firewall de aplicação)
- Camada de rede:
  - VPC na região sa-east-1 com 3 AZs (A, B, C)
  - Subnets públicas (uma por AZ) com Application Load Balancer (ALB)
  - Subnets privadas (A e B) para aplicação SEI/SIP em Auto Scaling (EC2)
  - Subnet privada (C) com Auto Scaling do Solr (EC2)
- Camada de dados:
  - Amazon RDS Multi-AZ (Primary em AZ A, Secondary em AZ B)
  - Amazon ElastiCache Memcached Multi-AZ
  - Amazon Elastic File System (EFS) Multi-AZ via NFS
- Observabilidade e segurança:
  - AWS CloudTrail (auditoria de APIs)
  - AWS Key Management Service (KMS)
  - AWS Backup
  - Amazon CloudWatch (métricas, logs, alarmes)
- Serviços de suporte:
  - Amazon Simple Email Service (SES)
  - (Implícito/gerencial) AWS Systems Manager/Management Services para operação

Interação entre os componentes:
- Usuários → CloudFront (protegido por AWS Shield) → WAF → ALB nas subnets públicas.
- ALB → instâncias EC2 da aplicação SEI/SIP (Auto Scaling nas subnets privadas A e B).
- Aplicação → RDS (leitura/escrita; failover para Secondary) → KMS para criptografia.
- Aplicação → ElastiCache Memcached (cache) e → EFS (arquivos compartilhados NFS).
- Aplicação → Solr (busca) hospedado em EC2 na AZ C.
- Aplicação → SES para envio de e-mails transacionais.

- CloudWatch coleta métricas/logs; CloudTrail registra chamadas de APIs; AWS Backup orquestra backups; KMS provê chaves para RDS/EBS/EFS/Logs.

O que esse sistema faz:
- Arquitetura web multi-AZ, escalável e altamente disponível para o SEI/SIP (aplicação transacional) com:
  - Entrega via CDN e WAF, balanceamento por ALB, escalabilidade horizontal por Auto Scaling.
  - Persistência relacional (RDS), cache de alto desempenho (ElastiCache), storage compartilhado (EFS) e motor de busca (Solr).
  - Observabilidade, auditoria, backups e criptografia gerenciada por serviços nativos da AWS.
  - Envio de e-mails via SES.

Vulnerabilidades e Solução para cada vulnerabilidade:
1) Exposição indevida do ALB (acesso direto, bypass do CloudFront/WAF)
   - Solução: Restringir o SG do ALB para aceitar somente o AWS Managed Prefix List “CloudFront origin-facing”; exigir header secreto entre CloudFront→ALB; aplicar WAF no CloudFront e no ALB conforme necessário.

2) Ausência de egress control/NAT por AZ e VPC Endpoints
   - Solução: Criar NAT Gateway por AZ ou preferir VPC Interface Endpoints (CloudWatch Logs, CloudTrail, KMS, SSM, Secrets Manager, SES) para tráfego privado; NACLs/Security Groups de saída restritivos.

3) Memcached sem criptografia
   - Solução: Migrar para ElastiCache Redis com TLS e criptografia at-rest; evitar colocar dados sensíveis no cache; rotacionar keys/dados com TTLs curtos.

4) Solr em única AZ (risco de indisponibilidade)
   - Solução: Distribuir o cluster Solr em pelo menos 2 AZs com replicação; health checks e políticas de Auto Scaling multi-AZ.

5) Segredos e credenciais armazenados no código/ambiente
   - Solução: AWS Secrets Manager/SSM Parameter Store com rotação automática (RDS/SMTP/APP); IAM Roles for EC2 com least privilege e boundary policies; MFA para operadores.

6) Tráfego sem criptografia entre camadas
   - Solução: TLS 1.2/1.3 fim-a-fim (CloudFront↔ALB, ALB↔App, App↔RDS/Redis/Solr quando suportado); NFS over TLS para EFS; Security Groups “deny by default”.

7) Falta de hardening/patching nas EC2
   - Solução: AMIs endurecidas (CIS), SSM Patch Manager, desligar SSH público, Session Manager, IMDSv2 obrigatório, EBS criptografado por padrão.

8) Injeções e XSS no app
   - Solução: Sanitização server-side, ORM parametrizado, CSP/HSTS, validação de entrada, regras WAF gerenciadas (SQLi/XSS), testes SAST/DAST em pipeline.

9) Auditoria incompleta e possibilidade de repúdio
   - Solução: CloudTrail org-wide, logs imutáveis (S3 + Object Lock + KMS), ALB/CloudFront access logs, VPC Flow Logs, trilhas de auditoria do RDS; clock sync (NTP) e correlação por request IDs.

10) Backup/DR insuficiente e risco regional
   - Solução: Backups automáticos (RDS/EFS) com retenção e testes de restauração; snapshots cross-Region; arquitetura de recuperação (pilot-light ou warm standby) em segunda região.

11) Proteção DDoS apenas básica
   - Solução: Considerar AWS Shield Advanced (deteção e resposta 24x7, proteções adicionais), rate-limiting no WAF, caching no CloudFront, Auto Scaling com limites e reservas.

12) Configuração drift e concessões IAM excessivas
   - Solução: AWS Config, SCPs e IAM Access Analyzer; Security Hub/GuardDuty; revisão periódica de privilégios e segregação de funções.

Gere um Relatório de Modelagem de Ameaças, baseado na metodologia STRIDE:
- Escopo e limites de confiança:
  - Borda pública: Internet ↔ CloudFront/WAF/Shield.
  - Perímetro da VPC: CloudFront ↔ ALB (subnets públicas).
  - Camada de aplicação: ALB ↔ EC2 (SEI/SIP) em subnets privadas.
  - Camada de dados: App ↔ RDS/ElastiCache/EFS/Solr.

  - Saída/control plane: App ↔ SES e serviços gerenciados (CloudWatch, KMS, Backup, CloudTrail) via endpoints/NAT.
  - Plano de controle AWS (APIs) ↔ contas/equipes operacionais.

- STRIDE por categoria (principais ameaças e mitigações):
  1) Spoofing (falsificação de identidade)
    - Risco: Uso de credenciais roubadas em APIs/admin; origem falsa batendo no ALB.
    - Mitigações: MFA/SSO (IAM Identity Center), OAuth/OIDC para usuários; SG do ALB restrito a CloudFront + header secreto; mTLS interno quando aplicável; chaves KMS com políticas de recurso; IMDSv2; WAF bot control.
  2) Tampering (adulteração de dados/tráfego)
    - Risco: Alteração de payloads, pacotes NFS, código na pipeline.
    - Mitigações: TLS 1.2/1.3 em todos os saltos; EFS criptografado + NFS over TLS; assinatura/integração CI/CD com code signing e verificação; controles de mudança (Change Manager) e least privilege.
  3) Repudiation (negação de autoria)
    - Risco: Ações administrativas sem trilha confiável; logs perdidos.
    - Mitigações: CloudTrail multi-região e organização; logs imutáveis (S3 Object Lock, MFA-Delete); ALB/CloudFront access logs; VPC Flow Logs; correlação com IDs de requisição e tempo sincronizado.
  4) Information Disclosure (exposição de dados)
    - Risco: Vazamento por cache/headers, Memcached sem criptografia, snapshots acessíveis.
    - Mitigações: PII tokenization/masking; Redis com TLS em vez de Memcached; políticas KMS e de backup restritivas; headers de segurança (CSP, HSTS, X-Content-Type-Options); WAF para data leak; segredo no Secrets Manager; revisão de TTL e regras de cache no CloudFront.
  5) Denial of Service (DoS/DDoS)
    - Risco: Saturação de ALB/app, exploração de endpoints caros, ataques de camada 7.
    - Mitigações: Shield Advanced, WAF rate-limit e challenge (CAPTCHA), caching no CloudFront, circuit-breakers e timeouts no app, Auto Scaling com limites, filas assíncronas para tarefas pesadas.
  6) Elevation of Privilege (elevação de privilégios)
    - Risco: Funções IAM amplas nas EC2/CI, LPE por sistema operacional, acesso lateral entre sub-redes.
    - Mitigações: IAM least privilege + permission boundaries; segmentação por Security Groups/NACL; desabilitar sudo/SSH; SSM Session Manager; varreduras e hardening CIS; rota de privilégios separada e break-glass auditada.

- Riscos notáveis e priorização (alto impacto/alta probabilidade):
  - Bypass do CloudFront/WAF para o ALB → tratar imediatamente (SG/headers/WAF).
  - Memcached sem criptografia → migrar para Redis ou remover dados sensíveis do cache.
  - Solr single-AZ → multi-AZ e automação de failover.
  - Segredos fora do Secrets Manager → migração e rotação automática.
  - Falta de egress control/endpoints → endpoints privados e/ou NAT por AZ com políticas.

- Controles recomendados adicionais:
  - Route 53 + ACM para TLS e HSTS; políticas de senhas/MFA e SSO centralizado.
  - GuardDuty, Security Hub, AWS Config, Detective para detecção e governança.
  - Pipeline CI/CD com SAST/DAST/IAST, dependabot, verificação de imagens/AMIs.
  - Testes regulares de restauração de backup e de caos (AZ/instância).

Additional resourcing needs:
- Network Administrator: validar sub-redes, NAT/Endpoints, NACLs, conectividade híbrida e requisitos de latência.
- Security Officer: revisar conformidade, políticas KMS, WAF, logging imutável, STRIDE e tratamento de riscos.
- Cloud/DevOps: desenhar Terraform/CloudFormation, CI/CD seguro, Auto Scaling, observabilidade, endpoints privados.
- Software Developer: mapear fluxos e APIs do SEI/SIP, validar autenticação/autorização, sanitização e performance.
- Hardware/Plataforma (se aplicável): hardening de AMIs, integrações com HSM/KMS, proteção de credenciais em runtime.
- Operações de Banco de Dados: parâmetros do RDS, PITR, cross-Region, criptografia, plano de failover e testes.
- FinOps: modelar custos de Shield Advanced, NAT por AZ, logs e armazenamento de backups.
- Gestão do Projeto: condução das sessões STRIDE (sprints por componente), definição de escopo, stakeholders e cadência de revisão.

Sugestões de melhorias

A seguir você encontra:
1) Diagrama Mermaid dos componentes, relações e correções aplicadas.
2) Relatório de Modelagem de Ameaças (STRIDE) para este escopo.
3) Um script Terraform de referência para provisionar a arquitetura na AWS sa-east-1 com as correções propostas. Observação: alguns recursos exigem pré-requisitos/inscrições (por exemplo, Shield Advanced) ou variáveis (ARNs de certificados ACM, domínios SES). O código foi escrito para ser seguro por padrão e servir como base editável.

Diagrama Mermaid

```mermaid
flowchart TD
  %% Borda e Proteção
  U[Usuários SEI] -->|HTTPS| CF[CloudFront\n+ AWS Shield (Adv opcional)\n+ WAFv2 (managed rules)]
  CF -->|Origin HTTPS + Header Secreto| ALB[Application Load Balancer (ALB)\nSubnets Públicas (AZ A/B/C)\nSG: Permite SOMENTE Prefix List CloudFront]

  %% VPC e sub-redes
  subgraph VPC[ VPC sa-east-1 ]
    direction TB

    subgraph AZA[AZ A]
      PUBA[Subnet Pública A]
      PRIVA_APP[Subnet Privada App A]
    end
    subgraph AZB[AZ B]
      PUBB[Subnet Pública B]
      PRIVB_APP[Subnet Privada App B]
    end
    subgraph AZC[AZ C]
      PUBC[Subnet Pública C]
      PRIVC_SOLR[Subnet Privada Solr C]
    end

    ALB --- PUBA
    ALB --- PUBB
    ALB --- PUBC

    %% Aplicação SEI/SIP
    ALB -->|HTTPS (TLS 1.2+)| ASGAPP[EC2 Auto Scaling SEI/SIP\nMulti-AZ (A/B)\nIMDSv2, SSM, sem SSH público]
    ASGAPP --- PRIVA_APP
    ASGAPP --- PRIVB_APP

    %% Solr
    ASGAPP -->|HTTPS/TLS| SOLR[EC2 Auto Scaling Solr\nMulti-AZ (>=2)\n(antes: apenas C)]
    SOLR --- PRIVC_SOLR
    SOLR -. opcional .- PRIVB_APP

    %% Dados
    ASGAPP -->|TLS| RDS[(Amazon RDS Multi-AZ\n(A prim/B sec)\nSSE-KMS; SG restrito)]
    ASGAPP -->|TLS| REDIS[(ElastiCache Redis\nc/ TLS + at-rest enc)\n(substitui Memcached)]
    ASGAPP -->|NFS over TLS| EFS[(Amazon EFS Multi-AZ\nSSE-KMS; SG restrito)]

    %% Suporte/saída controlada
    ASGAPP -->|PrivLink (VPC Endpoints)| SVC[(KMS, CloudWatch Logs,\nCloudTrail, SSM, Secrets Manager,\nSES API/SMTP)]
  end

  %% Observabilidade e Segurança Organizacional
  CT[CloudTrail org/multi-região\nLogs imutáveis (S3 + Object Lock + KMS)]
  CW[CloudWatch Métricas/Logs/Alarmes]
  BAK[AWS Backup\nRDS/EFS com retenção e cópia cross-Region]
  KMS[AWS KMS (chaves dedicadas)]

  %% Ligações de Observabilidade
  ALB -. access logs .-> CW
  CF -. access logs .-> CW
  ASGAPP -. app/infra logs .-> CW
```

```
  RDS -. audit .-> CW
  VPCFL[VPC Flow Logs] -.-> CW
  CT -. trilhas API .-> CW
  BAK -. jobs/relatórios .-> CW

  %% Correções de vulnerabilidades (principais)
  ALB:::fix -->|SG restrito a Prefix List CloudFront + header secreto| CF
  REDIS:::fix -->|migrado de Memcached p/ Redis c/ TLS + enc at-rest| ASGAPP
  SOLR:::fix -->|agora Multi-AZ + health checks| ASGAPP
  SVC:::fix -->|VPC Endpoints + egress control/NAT por AZ| ASGAPP
  EFS:::fix -->|NFS over TLS + SSE-KMS| ASGAPP

  classDef fix fill=#e6ffed,stroke=#087443,color=#087443;
```

Relatório de Modelagem de Ameaças (STRIDE) — Resumo
- Escopo e limites de confiança:
  - Borda pública: Internet ↔ CloudFront/WAF/Shield.
  - Perímetro da VPC: CloudFront ↔ ALB (subnets públicas).
  - Camada de aplicação: ALB ↔ EC2 (SEI/SIP) em subnets privadas A/B.
  - Camada de dados: App ↔ RDS/Redis/EFS/Solr.
  - Saída/control plane: App ↔ SES + serviços gerenciados via VPC Endpoints/NAT por AZ.
  - Plano de controle AWS: APIs ↔ contas/equipes operacionais.

- Ameaças e mitigações:
  1) Spoofing
     - Riscos: Credenciais roubadas; bypass direto ao ALB.
     - Mitigações: MFA/SSO (IAM Identity Center), OAuth/OIDC; SG do ALB restrito ao Managed Prefix List CloudFront + header secreto; IMDSv2; WAF bot control; políticas KMS por recurso.
  2) Tampering
     - Riscos: Alteração de payloads/NFS/código.
     - Mitigações: TLS 1.2/1.3 fim-a-fim; EFS criptografado + NFS over TLS; code signing na pipeline; least privilege e controle de mudanças.
  3) Repudiation
     - Riscos: Ações sem trilha confiável.
     - Mitigações: CloudTrail multi-região/org → S3 com Object Lock + KMS; logs de acesso ALB/CloudFront; VPC Flow Logs; IDs de correlação e tempo sincronizado.
  4) Information Disclosure
     - Riscos: Vazamento via cache/headers; snapshots; Memcached sem criptografia.
     - Mitigações: Redis com TLS e enc at-rest; KMS e políticas restritivas; headers de segurança (CSP/HSTS/etc.); WAF data leak; Secrets Manager; revisão de TTL/regras no CloudFront.
  5) Denial of Service
     - Riscos: Saturação de ALB/app; L7 caro.
     - Mitigações: Shield Advanced; WAF rate-limit e challenge; caching no CloudFront; timeouts/circuit-breakers; Auto Scaling com limites; filas assíncronas para tarefas pesadas.
  6) Elevation of Privilege
     - Riscos: IAM amplo; LPE SO; lateralidade.
     - Mitigações: IAM least privilege + permission boundaries; segmentação via SG/NACL; SSH desabilitado, Session Manager; hardening CIS e patching; rota de privilégios separada (break-glass auditado).

- Riscos prioritários (alto impacto/probabilidade):
  - Bypass de CloudFront/WAF para o ALB.
  - Cache sem criptografia (migração Memcached→Redis).
  - Solr single-AZ.
  - Segredos fora do Secrets Manager.
  - Egress sem controle/endpoints privados.

Script Terraform de Referência (main.tf)
Observações importantes:
- Substitua valores de variáveis marcados como TODO (por exemplo, ARNs de certificados ACM, tamanhos de instância, nomes de domínio).
- O certificado ACM para CloudFront deve estar em us-east-1. O do ALB deve estar em sa-east-1. Aqui foram deixados como variáveis.
- Recursos como Shield Advanced exigem assinatura prévia. O recurso aws_shield_protection é criado apenas se enable_shield_advanced = true.
- Para simplificar, este arquivo único contém comentários extensos e recursos essenciais. Em produção, modularize por domínio (rede, segurança, app, dados).

```hcl
terraform {
  required_version = ">= 1.5.0"
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = ">= 5.45"
    }
    random = {
      source  = "hashicorp/random"
      version = ">= 3.6"
    }
  }
}

provider "aws" {
  region = var.region
}

# Provider adicional para ACM de CloudFront (certificado deve estar em us-east-1)
provider "aws" {
  alias  = "us_east_1"
  region = "us-east-1"
}

##############################
# Variáveis
##############################
variable "region" {
  type        = string
  default     = "sa-east-1"
  description = "Região principal"
}

variable "project_name" {
  type        = string
  default     = "sei-sip"
  description = "Prefixo de nome para recursos"
}

variable "vpc_cidr" {
  type        = string
  default     = "10.20.0.0/16"
}

variable "azs" {
  type        = list(string)
  default     = ["sa-east-1a", "sa-east-1b", "sa-east-1c"]
}

# Subnets (ajuste CIDRs conforme necessidade)
variable "public_subnet_cidrs" {
  type        = list(string)
  default     = ["10.20.0.0/24", "10.20.1.0/24", "10.20.2.0/24"]
}
variable "private_app_subnet_cidrs" {
  type        = list(string)
  default     = ["10.20.10.0/24", "10.20.11.0/24"] # AZs A e B
}
variable "private_solr_subnet_cidrs" {
  type        = list(string)
  default     = ["10.20.12.0/24"] # AZ C (pode adicionar outra para multi-AZ)
}

# Certificados ACM
variable "alb_acm_certificate_arn" {
  type        = string
  description = "ARN do certificado ACM em sa-east-1 para o ALB"
}
```

```
variable "cloudfront_acm_certificate_arn" {
  type        = string
  description = "ARN do certificado ACM em us-east-1 para o CloudFront"
}

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

# ASG/EC2
variable "app_instance_type" {
  type    = string
  default = "t3.medium"
}
variable "solr_instance_type" {
  type    = string
  default = "t3.medium"
}
variable "app_desired_capacity" {
  type    = number
  default = 2
}
variable "solr_desired_capacity" {
  type    = number
  default = 2
}

# RDS
variable "rds_engine" {
  type    = string
  default = "postgres"
}
variable "rds_engine_version" {
  type    = string
  default = "15.4"
}
variable "rds_instance_class" {
  type    = string
  default = "db.t3.medium"
}

# Redis (ElastiCache)
variable "redis_node_type" {
  type    = string
  default = "cache.t3.small"
}

# Shield Advanced
variable "enable_shield_advanced" {
  type    = bool
  default = false
}

##############################
# KMS - Chave gerenciada para logs/EBS/RDS/EFS/S3
##############################
resource "aws_kms_key" "main" {
  description             = "${var.project_name}-kms"
  enable_key_rotation     = true
  deletion_window_in_days = 30
}

resource "aws_kms_alias" "main" {
  name          = "alias/${var.project_name}-kms"
  target_key_id = aws_kms_key.main.id
}
```

```
##############################
# Rede - VPC, Subnets, IGW, NAT por AZ, RTs
##############################
resource "aws_vpc" "this" {
  cidr_block           = var.vpc_cidr
  enable_dns_hostnames = true
  enable_dns_support   = true
  tags = { Name = "${var.project_name}-vpc" }
}

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

# Subnets Públicas (3)
resource "aws_subnet" "public" {
  for_each = {
    a = { cidr = var.public_subnet_cidrs[0], az = var.azs[0] }
    b = { cidr = var.public_subnet_cidrs[1], az = var.azs[1] }
    c = { cidr = var.public_subnet_cidrs[2], az = var.azs[2] }
  }
  vpc_id                  = aws_vpc.this.id
  cidr_block              = each.value.cidr
  availability_zone       = each.value.az
  map_public_ip_on_launch = false
  tags = { Name = "${var.project_name}-public-${each.key}" }
}

# Subnets Privadas App (A/B)
resource "aws_subnet" "private_app" {
  for_each = {
    a = { cidr = var.private_app_subnet_cidrs[0], az = var.azs[0] }
    b = { cidr = var.private_app_subnet_cidrs[1], az = var.azs[1] }
  }
  vpc_id            = aws_vpc.this.id
  cidr_block        = each.value.cidr
  availability_zone = each.value.az
  tags = { Name = "${var.project_name}-priv-app-${each.key}" }
}

# Subnet Privada Solr (C) - adicione outra se desejar multi-AZ completo
resource "aws_subnet" "private_solr" {
  for_each = {
    c = { cidr = var.private_solr_subnet_cidrs[0], az = var.azs[2] }
  }
  vpc_id            = aws_vpc.this.id
  cidr_block        = each.value.cidr
  availability_zone = each.value.az
  tags = { Name = "${var.project_name}-priv-solr-${each.key}" }
}

# Tabela de rota pública
resource "aws_route_table" "public" {
  vpc_id = aws_vpc.this.id
  tags   = { Name = "${var.project_name}-rtb-public" }
}
resource "aws_route" "public_inet" {
  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 por AZ
```

```
resource "aws_eip" "nat" {
  for_each = aws_subnet.public
  vpc      = true
  tags     = { Name = "${var.project_name}-eip-nat-${each.key}" }
}
resource "aws_nat_gateway" "nat" {
  for_each      = aws_subnet.public
  allocation_id = aws_eip.nat[each.key].id
  subnet_id     = aws_subnet.public[each.key].id
  tags          = { Name = "${var.project_name}-nat-${each.key}" }
  depends_on    = [aws_internet_gateway.igw]
}

# RT privadas (cada subnet privada sai por NAT da mesma AZ)
resource "aws_route_table" "private" {
  for_each = merge(aws_subnet.private_app, aws_subnet.private_solr)
  vpc_id   = aws_vpc.this.id
  tags     = { Name = "${var.project_name}-rtb-priv-${each.key}" }
}
resource "aws_route" "private_egress" {
  for_each               = aws_route_table.private
  route_table_id         = each.value.id
  destination_cidr_block = "0.0.0.0/0"
  nat_gateway_id         = contains(keys(aws_nat_gateway.nat), each.key) ?
aws_nat_gateway.nat[each.key].id : aws_nat_gateway.nat["c"].id
}
resource "aws_route_table_association" "private_app_assoc" {
  for_each       = aws_subnet.private_app
  subnet_id      = each.value.id
  route_table_id = aws_route_table.private[each.key].id
}
resource "aws_route_table_association" "private_solr_assoc" {
  for_each       = aws_subnet.private_solr
  subnet_id      = each.value.id
  route_table_id = aws_route_table.private[each.key].id
}

##############################
# VPC Endpoints (Interface e Gateway) + SG
##############################
resource "aws_security_group" "endpoints" {
  name        = "${var.project_name}-sg-endpoints"
  description = "Permite HTTPS dos apps/solr para endpoints"
  vpc_id      = aws_vpc.this.id
  ingress {
    from_port       = 443
    to_port         = 443
    protocol        = "tcp"
    security_groups = [aws_security_group.app.id, aws_security_group.solr.id]
  }
  egress { from_port = 0 to_port = 0 protocol = "-1" cidr_blocks = ["0.0.0.0/0"] }
  tags = { Name = "${var.project_name}-sg-endpoints" }
}

locals {
  interface_endpoints = [
    "com.amazonaws.${var.region}.logs",
    "com.amazonaws.${var.region}.monitoring",
    "com.amazonaws.${var.region}.kms",
    "com.amazonaws.${var.region}.ssm",
    "com.amazonaws.${var.region}.ssmmessages",
    "com.amazonaws.${var.region}.ec2messages",
    "com.amazonaws.${var.region}.secretsmanager",
    "com.amazonaws.${var.region}.cloudtrail",
    "com.amazonaws.${var.region}.email-smtp",
    "com.amazonaws.${var.region}.email"
  ]
}
```

```
resource "aws_vpc_endpoint" "interface" {
  for_each            = toset(local.interface_endpoints)
  vpc_id              = aws_vpc.this.id
  service_name        = each.value
  vpc_endpoint_type   = "Interface"
  private_dns_enabled = true
  security_group_ids  = [aws_security_group.endpoints.id]
  subnet_ids          = concat(values(aws_subnet.private_app)[*].id,
values(aws_subnet.private_solr)[*].id)
  tags                = { Name = "${var.project_name}-vpce-${replace(each.value,
"com.amazonaws.${var.region}.", "")}" }
}

# Gateway Endpoints (S3 e DynamoDB)
resource "aws_vpc_endpoint" "s3_gateway" {
  vpc_id            = aws_vpc.this.id
  service_name      = "com.amazonaws.${var.region}.s3"
  vpc_endpoint_type = "Gateway"
  route_table_ids   = [aws_route_table.private["a"].id, aws_route_table.private["b"].id,
aws_route_table.private["c"].id]
}
resource "aws_vpc_endpoint" "dynamodb_gateway" {
  vpc_id            = aws_vpc.this.id
  service_name      = "com.amazonaws.${var.region}.dynamodb"
  vpc_endpoint_type = "Gateway"
  route_table_ids   = [aws_route_table.private["a"].id, aws_route_table.private["b"].id,
aws_route_table.private["c"].id]
}

##############################
# Security Groups principais
##############################

# Prefix List CloudFront origin-facing
data "aws_prefix_list" "cloudfront_origin" {
  name = "com.amazonaws.global.cloudfront.origin-facing"
}

resource "aws_security_group" "alb" {
  name   = "${var.project_name}-sg-alb"
  vpc_id = aws_vpc.this.id
  description = "ALB exposto somente a CloudFront"
  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    prefix_list_ids = [data.aws_prefix_list.cloudfront_origin.id]
    description = "Somente CloudFront origin-facing"
  }
  egress { from_port = 0 to_port = 0 protocol = "-1" cidr_blocks = ["0.0.0.0/0"] }
  tags = { Name = "${var.project_name}-sg-alb" }
}

resource "aws_security_group" "app" {
  name   = "${var.project_name}-sg-app"
  vpc_id = aws_vpc.this.id
  ingress { from_port = 443 to_port = 443 protocol = "tcp" security_groups =
[aws_security_group.alb.id] description = "Tráfego do ALB" }
  egress  { from_port = 0 to_port = 0 protocol = "-1" cidr_blocks = ["0.0.0.0/0"] }
  tags = { Name = "${var.project_name}-sg-app" }
}

resource "aws_security_group" "solr" {
  name   = "${var.project_name}-sg-solr"
  vpc_id = aws_vpc.this.id
  ingress { from_port = 8983 to_port = 8983 protocol = "tcp" security_groups =
[aws_security_group.app.id] description = "App -> Solr" }
  egress  { from_port = 0 to_port = 0 protocol = "-1" cidr_blocks = ["0.0.0.0/0"] }
  tags = { Name = "${var.project_name}-sg-solr" }
```

```
}

resource "aws_security_group" "rds" {
  name   = "${var.project_name}-sg-rds"
  vpc_id = aws_vpc.this.id
  ingress { from_port = 5432 to_port = 5432 protocol = "tcp" security_groups =
[aws_security_group.app.id] description = "App -> RDS" }
  egress  { from_port = 0 to_port = 0 protocol = "-1" cidr_blocks = ["0.0.0.0/0"] }
  tags = { Name = "${var.project_name}-sg-rds" }
}

resource "aws_security_group" "redis" {
  name   = "${var.project_name}-sg-redis"
  vpc_id = aws_vpc.this.id
  ingress { from_port = 6379 to_port = 6379 protocol = "tcp" security_groups =
[aws_security_group.app.id] description = "App -> Redis (TLS)" }
  egress  { from_port = 0 to_port = 0 protocol = "-1" cidr_blocks = ["0.0.0.0/0"] }
  tags = { Name = "${var.project_name}-sg-redis" }
}

resource "aws_security_group" "efs" {
  name   = "${var.project_name}-sg-efs"
  vpc_id = aws_vpc.this.id
  ingress { from_port = 2049 to_port = 2049 protocol = "tcp" security_groups =
[aws_security_group.app.id] description = "App -> EFS (TLS no cliente)" }
  egress  { from_port = 0 to_port = 0 protocol = "-1" cidr_blocks = ["0.0.0.0/0"] }
  tags = { Name = "${var.project_name}-sg-efs" }
}

##############################
# ALB + Target Group + Listener (header secreto exigido)
##############################
resource "aws_lb" "app" {
  name               = "${var.project_name}-alb"
  load_balancer_type = "application"
  security_groups    = [aws_security_group.alb.id]
  subnets            = values(aws_subnet.public)[*].id
  idle_timeout       = 60
  drop_invalid_header_fields = true
  desync_mitigation_mode     = "defensive"
}

resource "aws_lb_target_group" "app" {
  name        = "${var.project_name}-tg"
  port        = 443
  protocol    = "HTTPS"
  vpc_id      = aws_vpc.this.id
  target_type = "instance"
  health_check {
    enabled             = true
    protocol            = "HTTPS"
    path                = "/health"
    matcher             = "200-399"
    interval            = 30
    unhealthy_threshold = 3
  }
}

resource "aws_lb_listener" "https" {
  load_balancer_arn = aws_lb.app.arn
  port              = 443
  protocol          = "HTTPS"
  ssl_policy        = "ELBSecurityPolicy-TLS13-1-2-2021-06"
  certificate_arn   = var.alb_acm_certificate_arn

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.app.arn
  }
```

```
}

# Listener rule para exigir header secreto (se ausente, 403)
resource "aws_lb_listener_rule" "secret_header" {
  listener_arn = aws_lb_listener.https.arn
  priority     = 10
  action {
    type = "forward"
    target_group_arn = aws_lb_target_group.app.arn
  }
  condition {
    http_header {
      http_header_name = "X-Edge-Secret"
      values           = ["${random_password.edge_secret.result}"]
    }
  }
}

resource "aws_lb_listener_rule" "deny_without_header" {
  listener_arn = aws_lb_listener.https.arn
  priority     = 5
  action {
    type = "fixed-response"
    fixed_response {
      content_type = "text/plain"
      message_body = "Forbidden"
      status_code  = "403"
    }
  }
  condition {
    http_header {
      http_header_name = "X-Edge-Secret"
      values           = ["*invalid*"] # nunca casa; age como fallback antes do default
    }
  }
  depends_on = [aws_lb_listener_rule.secret_header]
}

resource "random_password" "edge_secret" {
  length  = 32
  special = false
}

##############################
# CloudFront + WAFv2 (CLOUDFRONT scope)
##############################
resource "aws_wafv2_web_acl" "cf" {
  name        = "${var.project_name}-waf-cf"
  description = "WAF CloudFront"
  scope       = "CLOUDFRONT"

  default_action { allow {} }

  rule {
    name     = "AWS-AWSManagedRulesCommonRuleSet"
    priority = 1
    statement { managed_rule_group_statement { name = "AWSManagedRulesCommonRuleSet"
vendor_name = "AWS" } }
    visibility_config { cloudwatch_metrics_enabled = true metric_name = "common"
sampled_requests_enabled = true }
    override_action { none {} }
  }
  rule {
    name     = "AWS-AWSManagedRulesSQLiRuleSet"
    priority = 2
    statement { managed_rule_group_statement { name = "AWSManagedRulesSQLiRuleSet"
vendor_name = "AWS" } }
    visibility_config { cloudwatch_metrics_enabled = true metric_name = "sqli"
sampled_requests_enabled = true }
```

```
    override_action { none {} }
  }
  rule {
    name     = "AWS-AWSManagedRulesKnownBadInputsRuleSet"
    priority = 3
    statement { managed_rule_group_statement { name =
"AWSManagedRulesKnownBadInputsRuleSet" vendor_name = "AWS" } }
    visibility_config { cloudwatch_metrics_enabled = true metric_name = "badinputs"
sampled_requests_enabled = true }
    override_action { none {} }
  }
  rule {
    name     = "RateLimit"
    priority = 10
    statement { rate_based_statement { limit = var.waf_rate_limit aggregate_key_type = "IP" } }
    visibility_config { cloudwatch_metrics_enabled = true metric_name = "ratelimit"
sampled_requests_enabled = true }
    action { block {} }
  }

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

resource "aws_cloudfront_distribution" "this" {
  enabled             = true
  is_ipv6_enabled     = true
  comment             = "${var.project_name}-cf"
  price_class         = "PriceClass_200"

  origin {
    domain_name = aws_lb.app.dns_name
    origin_id   = "alb-origin"
    custom_origin_config {
      http_port              = 80
      https_port             = 443
      origin_protocol_policy = "https-only"
      origin_ssl_protocols   = ["TLSv1.2", "TLSv1.3"]
    }
    origin_header {
      name  = "X-Edge-Secret"
      value = random_password.edge_secret.result
    }
  }

  default_cache_behavior {
    allowed_methods  = ["GET", "HEAD", "OPTIONS", "PUT", "POST", "PATCH", "DELETE"]
    cached_methods   = ["GET", "HEAD", "OPTIONS"]
    target_origin_id = "alb-origin"
    viewer_protocol_policy = "redirect-to-https"
    forwarded_values {
      query_string = true
      headers      = ["Authorization", "Content-Type", "X-Requested-With"]
      cookies { forward = "all" }
    }
    compress = true
  }

  restrictions { geo_restriction { restriction_type = "none" } }

  viewer_certificate {
    acm_certificate_arn            = var.cloudfront_acm_certificate_arn
    ssl_support_method             = "sni-only"
    minimum_protocol_version       = "TLSv1.2_2021"
  }
```

```
  web_acl_id = aws_wafv2_web_acl.cf.arn

  depends_on = [aws_lb.app]
}

# Proteções Shield Advanced (opcionais)
resource "aws_shield_protection" "alb" {
  count        = var.enable_shield_advanced ? 1 : 0
  name         = "${var.project_name}-shield-alb"
  resource_arn = aws_lb.app.arn
}
resource "aws_shield_protection" "cf" {
  count        = var.enable_shield_advanced ? 1 : 0
  name         = "${var.project_name}-shield-cf"
  resource_arn = aws_cloudfront_distribution.this.arn
}

##############################
# IAM Roles para EC2 (SSM, Logs) e perfis
##############################
data "aws_iam_policy_document" "ec2_assume" {
  statement {
    actions = ["sts:AssumeRole"]
    principals { type = "Service" identifiers = ["ec2.amazonaws.com"] }
  }
}

resource "aws_iam_role" "ec2_role" {
  name               = "${var.project_name}-ec2-role"
  assume_role_policy = data.aws_iam_policy_document.ec2_assume.json
}

resource "aws_iam_role_policy_attachment" "ssm_managed" {
  role       = aws_iam_role.ec2_role.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
}
resource "aws_iam_role_policy_attachment" "cw_agent" {
  role       = aws_iam_role.ec2_role.name
  policy_arn = "arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy"
}

resource "aws_iam_instance_profile" "ec2_profile" {
  name = "${var.project_name}-ec2-profile"
  role = aws_iam_role.ec2_role.name
}

##############################
# Launch Templates (IMDSv2, sem SSH público)
##############################
data "aws_ssm_parameter" "al2023_ami" {
  name = "/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-6.1-x86_64"
}

locals {
  common_user_data = base64encode(<<-EOF
    #!/bin/bash
    set -euo pipefail
    dnf -y update
    dnf -y install amazon-efs-utils
    # Exemplo: montar EFS com TLS quando ID disponível via SSM/Tags
    # mkdir -p /mnt/efs && echo "${aws_efs_file_system.main.dns_name}:/ /mnt/efs efs _netdev,tls 0
0" >> /etc/fstab && mount -a
    # habilitar cloudwatch agent via ssm
  EOF
  )
}

resource "aws_launch_template" "app" {
  name_prefix   = "${var.project_name}-lt-app-"
```

```
  image_id      = data.aws_ssm_parameter.al2023_ami.value
  instance_type = var.app_instance_type
  iam_instance_profile { name = aws_iam_instance_profile.ec2_profile.name }
  update_default_version = true

  network_interfaces {
    associate_public_ip_address = false
    security_groups             = [aws_security_group.app.id]
  }

  metadata_options {
    http_tokens = "required" # IMDSv2
  }

  user_data = local.common_user_data

  block_device_mappings {
    device_name = "/dev/xvda"
    ebs {
      volume_size = 30
      volume_type = "gp3"
      encrypted   = true
      kms_key_id  = aws_kms_key.main.arn
    }
  }
}

resource "aws_launch_template" "solr" {
  name_prefix   = "${var.project_name}-lt-solr-"
  image_id      = data.aws_ssm_parameter.al2023_ami.value
  instance_type = var.solr_instance_type
  iam_instance_profile { name = aws_iam_instance_profile.ec2_profile.name }
  update_default_version = true

  network_interfaces {
    associate_public_ip_address = false
    security_groups             = [aws_security_group.solr.id]
  }

  metadata_options { http_tokens = "required" }
  user_data = local.common_user_data

  block_device_mappings {
    device_name = "/dev/xvda"
    ebs {
      volume_size = 30
      volume_type = "gp3"
      encrypted   = true
      kms_key_id  = aws_kms_key.main.arn
    }
  }
}

##############################
# Auto Scaling Groups
##############################
resource "aws_autoscaling_group" "app" {
  name                      = "${var.project_name}-asg-app"
  max_size                  = 6
  min_size                  = 2
  desired_capacity          = var.app_desired_capacity
  vpc_zone_identifier       = [aws_subnet.private_app["a"].id, aws_subnet.private_app["b"].id]
  health_check_type         = "EC2"
  health_check_grace_period = 60

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

```
  target_group_arns = [aws_lb_target_group.app.arn]
  tag { key = "Name" value = "${var.project_name}-app" propagate_at_launch = true }
}

# Solr em pelo menos 1 AZ adicional (C); adicione B se quiser completo
resource "aws_autoscaling_group" "solr" {
  name                = "${var.project_name}-asg-solr"
  max_size            = 4
  min_size            = 2
  desired_capacity    = var.solr_desired_capacity
  vpc_zone_identifier = [aws_subnet.private_solr["c"].id]
  health_check_type   = "EC2"
  health_check_grace_period = 60

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

  tag { key = "Name" value = "${var.project_name}-solr" propagate_at_launch = true }
}

##############################
# EFS (Multi-AZ)
##############################
resource "aws_efs_file_system" "main" {
  encrypted  = true
  kms_key_id = aws_kms_key.main.arn
  tags = { Name = "${var.project_name}-efs" }
}

# Mount targets nas subnets privadas que hospedam as instâncias
resource "aws_efs_mount_target" "mt_app_a" {
  file_system_id  = aws_efs_file_system.main.id
  subnet_id       = aws_subnet.private_app["a"].id
  security_groups = [aws_security_group.efs.id]
}
resource "aws_efs_mount_target" "mt_app_b" {
  file_system_id  = aws_efs_file_system.main.id
  subnet_id       = aws_subnet.private_app["b"].id
  security_groups = [aws_security_group.efs.id]
}
resource "aws_efs_mount_target" "mt_solr_c" {
  file_system_id  = aws_efs_file_system.main.id
  subnet_id       = aws_subnet.private_solr["c"].id
  security_groups = [aws_security_group.efs.id]
}

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

resource "aws_secretsmanager_secret" "db" {
  name = "${var.project_name}/rds/master"
  kms_key_id = aws_kms_key.main.arn
}

resource "aws_secretsmanager_secret_version" "db" {
  secret_id     = aws_secretsmanager_secret.db.id
  secret_string = jsonencode({ username = "seiadmin", password = random_password.db_master.result
})
}

resource "aws_db_subnet_group" "rds" {
```

```
  name       = "${var.project_name}-rds-subnetgrp"
  subnet_ids = [aws_subnet.private_app["a"].id, aws_subnet.private_app["b"].id]
}

resource "aws_db_instance" "rds" {
  identifier              = "${var.project_name}-rds"
  engine                  = var.rds_engine
  engine_version            = var.rds_engine_version
  instance_class            = var.rds_instance_class
  multi_az                = true
  allocated_storage         = 50
  storage_encrypted         = true
  kms_key_id              = aws_kms_key.main.arn
  username                = "seiadmin"
  password                = random_password.db_master.result
  db_subnet_group_name      = aws_db_subnet_group.rds.name
  vpc_security_group_ids    = [aws_security_group.rds.id]
  backup_retention_period   = 7
  deletion_protection       = true
  publicly_accessible       = false
  skip_final_snapshot       = false
  auto_minor_version_upgrade = true
  performance_insights_enabled = true
  performance_insights_kms_key_id = aws_kms_key.main.arn
}

##############################
# ElastiCache Redis (TLS + At-rest)
##############################
resource "aws_elasticache_subnet_group" "redis" {
  name       = "${var.project_name}-redis-subnetgrp"
  subnet_ids = [aws_subnet.private_app["a"].id, aws_subnet.private_app["b"].id]
}

resource "aws_elasticache_replication_group" "redis" {
  replication_group_id       = "${var.project_name}-redis"
  replication_group_description = "Redis TLS/enc for ${var.project_name}"
  engine                     = "redis"
  engine_version               = "7.0"
  node_type                  = var.redis_node_type
  number_cache_clusters        = 2
  automatic_failover_enabled   = true
  multi_az_enabled             = true
  security_group_ids           = [aws_security_group.redis.id]
  subnet_group_name            = aws_elasticache_subnet_group.redis.name
  at_rest_encryption_enabled   = true
  transit_encryption_enabled   = true
}

##############################
# CloudTrail + S3 (Object Lock + KMS) + VPC Flow Logs
##############################
resource "aws_s3_bucket" "trail" {
  bucket            = "${var.project_name}-trail-${random_password.edge_secret.result}"
  force_destroy     = false
  object_lock_enabled = true
}

resource "aws_s3_bucket_versioning" "trail" {
  bucket = aws_s3_bucket.trail.id
  versioning_configuration { status = "Enabled" }
}

resource "aws_s3_bucket_object_lock_configuration" "trail" {
  bucket = aws_s3_bucket.trail.id
  rule {
    default_retention {
      mode  = "COMPLIANCE"
      days  = 90
```

```
    }
  }
}

resource "aws_s3_bucket_server_side_encryption_configuration" "trail" {
  bucket = aws_s3_bucket.trail.id
  rule {
    apply_server_side_encryption_by_default {
      sse_algorithm     = "aws:kms"
      kms_master_key_id = aws_kms_key.main.arn
    }
  }
}

resource "aws_cloudtrail" "main" {
  name                          = "${var.project_name}-trail"
  s3_bucket_name                = aws_s3_bucket.trail.id
  include_global_service_events = true
  is_multi_region_trail         = true
  enable_log_file_validation    = true
  kms_key_id                    = aws_kms_key.main.arn
}

resource "aws_cloudwatch_log_group" "vpc_flow" {
  name              = "/aws/vpc/${var.project_name}/flowlogs"
  retention_in_days = 90
  kms_key_id        = aws_kms_key.main.arn
}

data "aws_iam_policy_document" "flowlogs_assume" {
  statement {
    actions = ["sts:AssumeRole"]
    principals { type = "Service" identifiers = ["vpc-flow-logs.amazonaws.com"] }
  }
}

resource "aws_iam_role" "flowlogs" {
  name               = "${var.project_name}-role-flowlogs"
  assume_role_policy = data.aws_iam_policy_document.flowlogs_assume.json
}

resource "aws_iam_role_policy" "flowlogs" {
  name = "${var.project_name}-pol-flowlogs"
  role = aws_iam_role.flowlogs.id
  policy = jsonencode({
    Version = "2012-10-17",
    Statement = [{
      Effect   = "Allow",
      Action   = ["logs:CreateLogStream", "logs:PutLogEvents", "logs:DescribeLogGroups",
"logs:DescribeLogStreams"],
      Resource = "*"
    }]
  })
}

resource "aws_flow_log" "vpc" {
  log_destination_type = "cloud-watch-logs"
  log_group_name       = aws_cloudwatch_log_group.vpc_flow.name
  iam_role_arn         = aws_iam_role.flowlogs.arn
  traffic_type         = "ALL"
  vpc_id               = aws_vpc.this.id
}

##############################
# AWS Backup (RDS/EFS)
##############################
resource "aws_backup_vault" "main" {
  name        = "${var.project_name}-backup-vault"
  kms_key_arn = aws_kms_key.main.arn
```

```
}

resource "aws_backup_plan" "main" {
  name = "${var.project_name}-backup-plan"
  rule {
    rule_name         = "daily-backup"
    target_vault_name = aws_backup_vault.main.name
    schedule          = "cron(0 5 * * ? *)"
    lifecycle {
      cold_storage_after = 30
      delete_after       = 120
    }
  }
}

# Seleção por ARNs (poderia ser por tags)
resource "aws_backup_selection" "main" {
  name         = "${var.project_name}-backup-selection"
  iam_role_arn = aws_iam_role.flowlogs.arn # usar um papel dedicado em prod
  plan_id      = aws_backup_plan.main.id

  resources = [
    aws_efs_file_system.main.arn,
    aws_db_instance.rds.arn
  ]
}

##############################
# SES (Observações)
##############################
# - Configure verificação de domínio, DKIM e políticas de envio fora deste script.
# - O app deve usar SDK (via endpoints privados) ou SMTP (email-smtp) pelas interface endpoints
criadas.

##############################
# Saídas úteis
##############################
output "cloudfront_domain_name" { value = aws_cloudfront_distribution.this.domain_name }
output "alb_dns_name"           { value = aws_lb.app.dns_name }
output "rds_endpoint"           { value = aws_db_instance.rds.address }
output "redis_primary_endpoint" { value =
aws_elasticache_replication_group.redis.primary_endpoint_address }
output "efs_id"                 { value = aws_efs_file_system.main.id }
```

Como aplicar e validar rapidamente
- Preencha os ARNs dos certificados ACM (ALB em sa-east-1 e CloudFront em us-east-1).
- terraform init; terraform plan; terraform apply.
- Configure o CloudFront para apontar seu domínio (Route 53) e habilite HSTS no aplicativo.
- Testes de segurança:
  - Confirme que o ALB não responde sem o header X-Edge-Secret nem fora do prefix list do CloudFront.
  - Verifique conexões TLS fim-a-fim (CF→ALB, ALB→App, App→RDS/Redis; EFS montado com tls).
  - Valide endpoints privados (ex.: curl para KMS/Logs/SSM hostname resolve para IPs privados).
  - Inspecione WAF métricas/contadores e regras gerenciadas.
  - Garanta que Solr opere em multi-AZ (adicione sub-rede extra para B se quiser 2 AZs).
  - Cheque CloudTrail no bucket S3 com Object Lock ativado e KMS.

Pontos de atenção e próximos passos
- Integre CI/CD com SAST/DAST e code signing; automatize Patching via SSM Patch Manager.
- Configure GuardDuty, Security Hub, AWS Config e SCPs no nível organizacional.
- Planeje DR cross-Region (Backups e, se necessário, replicação de dados/infra).
- Migre segredos de app para AWS Secrets Manager com rotação (RDS/SMTP etc.).
- Estime custos de NAT por AZ, CloudFront, WAF, Shield Advanced, logs e Backup (FinOps).