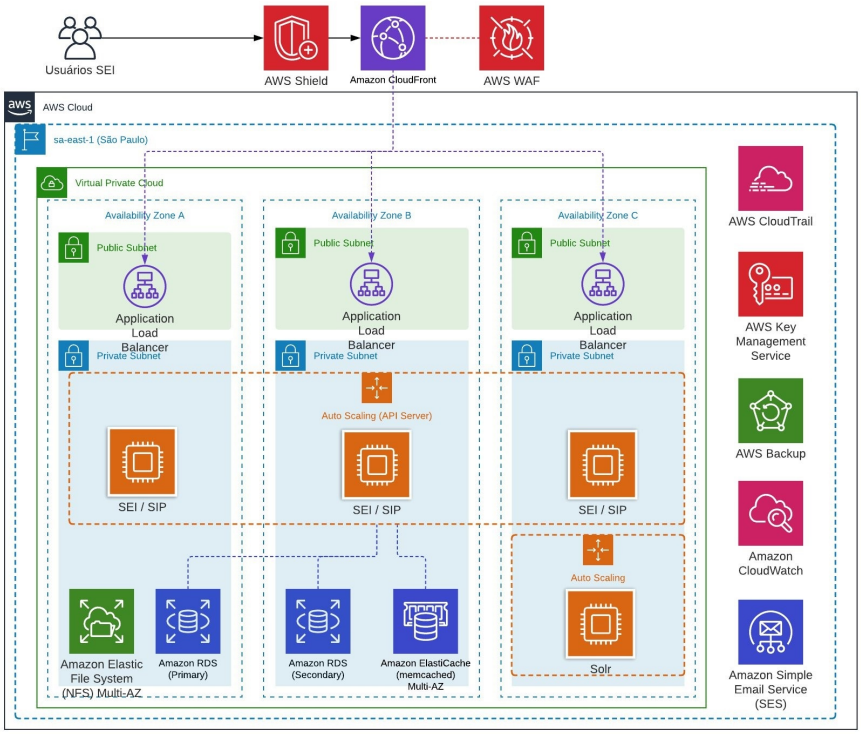
Relatório de análise da solução: tmp6m39cvq0



Análise completa da solução atual

Modelo de cloud:
- AWS (Amazon Web Services), região sa-east-1 (São Paulo), arquitetura multi-AZ com borda global (CloudFront) e camadas de segurança gerenciadas (AWS Shield e AWS WAF).

Lista com os componentes:
- Usuários SEI
- AWS Shield
- Amazon CloudFront
- AWS WAF
- VPC com 3 Availability Zones (A, B, C)
  - Subnets públicas (1 por AZ)
  - Subnets privadas (1 por AZ)
- Application Load Balancer (ALB) em subnets públicas (multi-AZ)
- EC2 Auto Scaling (camada de aplicação API/SEI/SIP) em subnets privadas
- EC2 Auto Scaling para Solr (busca) em subnet privada dedicada
- Amazon RDS:
  - RDS Primary (AZ A)
  - RDS Secondary/standby (AZ B)
- Amazon ElastiCache (Memcached) Multi-AZ
- Amazon Elastic File System (EFS) Multi-AZ (NFS)
- Amazon CloudTrail
- AWS Key Management Service (KMS)
- AWS Backup
- Amazon CloudWatch
- Amazon Simple Email Service (SES)

Interação entre os componentes:
- Fluxo de usuário: Usuários → CloudFront (protegido por AWS Shield) → WAF → ALB → EC2 (SEI/SIP).
- Aplicação:
  - Lê/grava documentos e anexos no EFS via NFS (TLS).
  - Consulta/atualiza dados transacionais no RDS (conexão TLS; primário com standby Multi-AZ).
  - Usa ElastiCache para cache de sessões/consultas (atualmente Memcached).
  - Realiza buscas no cluster Solr (Auto Scaling).
  - Envia e-mails transacionais via SES.
- Observabilidade e segurança:

- CloudWatch recebe métricas e logs (aplicação, sistema, ALB, WAF).
- CloudTrail audita chamadas de API de conta/serviços.
- KMS provê chaves para criptografia em repouso (RDS, EFS, EBS, logs, backups).
- AWS Backup orquestra snapshots/retention (RDS, EFS) e Vault Lock/immutability (quando habilitado).
- Disponibilidade: ALB, EC2, RDS, EFS e ElastiCache distribuídos entre AZs para alta disponibilidade.

O que esse sistema faz:
- Hospeda o SEI/SIP (gestão/processamento eletrônico de informações/documentos), com publicação segura via CDN+WAF, camada web/API escalável, busca full-text (Solr), cache, banco relacional transacional (RDS), armazenamento de arquivos compartilhado (EFS) e envio de e-mails (SES), com auditoria, backup e criptografia gerenciadas.

Vulnerabilidades e Solução para cada vulnerabilidade:
1) Bypass ao CloudFront/WAF acessando o ALB diretamente
- Risco: clientes atacarem o ALB pelo IP público, contornando WAF/Shield.
- Mitigação: restringir Security Group do ALB para aceitar apenas origens do CloudFront (OAC/assinatura via header personalizado) e IPs de gestão; usar AWS WAF no ALB também se necessário.

2) ElastiCache Memcached sem criptografia/autenticação nativa
- Risco: Memcached não provê TLS/ACL; exposição interna facilita sniffing e abuso.
- Mitigação: migrar para ElastiCache Redis com TLS e ACLs; se permanecer Memcached, isolar com SG estrito apenas para a aplicação e VPC Traffic Mirroring desabilitado; segmentar NACL.

3) Falta de egress control/PrivateLink (não há NAT/VPC endpoints no diagrama)
- Risco: instâncias privadas podem precisar sair à Internet para falar com serviços AWS/terceiros; aumenta superfície de ataque.
- Mitigação: criar VPC Endpoints (Interface/Gateway) para CloudWatch, CloudTrail, KMS, S3, SES; se NAT for necessário, usar NAT Gateway com regras de egress filtering (Network Firewall) e deny-all por padrão.

4) EFS excessivamente permissivo
- Risco: acesso amplo a / com NFS pode permitir movimentação lateral e exfiltração.
- Mitigação: EFS Access Points com POSIX, SG dedicados (porta 2049 apenas de subnets/SG da app), encriptação in-transit, políticas de backup e replicação entre regiões.

5) Solr sem proteção robusta
- Risco: endpoints administrativos/executores de query pesadas; exploração RCE históricas.
- Mitigação: Solr atrás de ALB interno; TLS obrigatório; autenticação/RBAC (Basic/OIDC); bloquear API perigosas; WAF interno/rate limit; patching contínuo.

6) Segredos em instâncias/aplicação
- Risco: credenciais de BD/SES em variáveis/arquivos.
- Mitigação: AWS Secrets Manager com rotação automática; IAM roles mínimas; proibir hardcode; varredura de segredos em CI.

7) IMDS/SSRF em EC2
- Risco: exploração SSRF para roubar credenciais de instance profile.
- Mitigação: IMDSv2 obrigatório e hop-limit=1; WAF bloqueando padrões SSRF; egress da app bloqueado ao 169.254.169.254 exceto metadados via IMDSv2.

8) Criptografia/trânsito/repouso inconsistentes
- Risco: dados sensíveis (LGPD) expostos.
- Mitigação: TLS 1.2/1.3 fim-a-fim; HSTS no CloudFront/ALB; KMS para RDS/EFS/EBS/CloudWatch/CloudTrail/Backups; rotação de chaves; políticas de chave mínimas.

9) Logs/auditoria não imutáveis
- Risco: repudiation e destruição de evidências.
- Mitigação: CloudTrail multi-região e org-level; logs para S3 com Object Lock (WORM) e MFA Delete; AWS Backup Vault Lock; integrações com Security Hub/GuardDuty/Detective.

10) DoS contra aplicação e banco
- Risco: esgotamento de conexões/CPU/IO.
- Mitigação: AWS Shield Advanced, WAF rate-based rules e bot control; Auto Scaling; connection pooling (RDS Proxy); read replicas; caching agressivo; limites de payload e timeouts no ALB.

11) Exposição de snapshots/backups

- Risco: snapshots RDS/EBS/EFS públicos ou compartilhados inadvertidamente.
- Mitigação: proibir compartilhamento sem aprovação; criptografia KMS; AWS Config + regras de conformidade; scans periódicos.

12) Envio de e-mail (SES) com spoofing
- Risco: phishing e reputação de domínio.
- Mitigação: SPF, DKIM, DMARC, monitoração de bounces/complaints; segregação de identidades SES por ambiente; quotas.

13) Falta de DR entre regiões
- Risco: indisponibilidade regional.
- Mitigação: RDS cross-region read replica e promoção; EFS Replication cross-region; backups cross-region; runbooks e exercícios de recuperação.

14) IAM excessivamente permissivo
- Risco: elevação lateral e abuso de APIs.
- Mitigação: least privilege, permission boundaries, SCPs (Organizations), Access Analyzer, tags baseadas em ABAC; separação de contas por ambiente.

Gere um Relatório de Modelagem de Ameaças, baseado na metodologia STRIDE:
- Escopo e ativos
  - Escopo: Front (CloudFront/WAF/ALB), App (EC2 Auto Scaling SEI/SIP), Dados (RDS, EFS, ElastiCache, Solr), Operações (SES, CloudWatch, CloudTrail, KMS, Backup), VPC multi-AZ.
  - Ativos: documentos/processos do SEI, credenciais/segredos, disponibilidade do serviço, logs/auditoria, reputação de envio de e-mails, chaves KMS.
  - Principais fronteiras de confiança: Internet→Borda (CloudFront), Borda→VPC/ALB, Subnet pública (ALB)→privada (App), App→Data stores, VPC→Serviços gerenciados (endpoints/egress).

- Spoofing (S)
  - S1: Bypass do CloudFront/WAF para ALB. Mitigação: SG do ALB restrito ao CloudFront, OAC/header de origem, WAF no ALB.
  - S2: Roubo/abuso de sessão do usuário. Mitigação: TLS, cookies Secure/HttpOnly/SameSite, rotacionar tokens, MFA/IdP OIDC, rate limit de login.
  - S3: Spoofing de remetente em SES. Mitigação: SPF, DKIM, DMARC, separação de domínios por ambiente.
  - S4: Abuso de credenciais temporárias da instância via SSRF. Mitigação: IMDSv2, políticas IAM condicionais (aws:SourceVpce, aws:PrincipalTag), firewalls locais.

- Tampering (T)
  - T1: Alteração de dados em trânsito entre ALB↔EC2↔RDS/EFS/Solr. Mitigação: TLS 1.2/1.3 obrigatório, certificados gerenciados/validados.
  - T2: Alteração de dados em repouso (EFS/RDS/logs). Mitigação: criptografia KMS, controle de acesso POSIX/EFS APs, S3 Object Lock para logs.
  - T3: Pipeline/AMI alterada. Mitigação: AMIs “golden” assinadas, CodeBuild/CodePipeline com aprovação, varredura (Inspector, CodeGuru).

- Repudiation (R)
  - R1: Ações sem trilhas de auditoria válidas. Mitigação: CloudTrail org/all-regions, logs do ALB/CloudFront/WAF em S3+Object Lock, Clock sync (Amazon Time Sync), correlação em SIEM.
  - R2: Logs alteráveis por operadores. Mitigação: contas separadas de Log Archive, acesso só de leitura, Vault Lock/controle de retenção.

- Information Disclosure (I)
  - I1: Vazamento de documentos via EFS/backup/snapshots. Mitigação: SG/NACL estritos, EFS APs, criptografia, DLP em saída, revisão de compartilhamentos.
  - I2: Exposição de variáveis de ambiente/segredos. Mitigação: Secrets Manager/Parameter Store criptografado, remoção de segredos do filesystem.
  - I3: Headers/erros verbosos no ALB/app. Mitigação: suprimir stack traces, headers de segurança (HSTS, CSP, X-Frame-Options).

- Denial of Service (D)
  - D1: DDoS de borda. Mitigação: Shield Advanced, WAF managed rules+rate limit, CloudFront caching, circuit breakers.
  - D2: Exaustão de RDS/Solr. Mitigação: RDS Proxy, limites de query, índices, read replicas, auto scaling Solr, filas/retentativas.
  - D3: EFS throughput burst exausto. Mitigação: modo de throughput provisionado/adaptativo, cache local quando possível.

- Elevation of Privilege (E)

  - E1: IAM role escalation. Mitigação: least privilege, permissions boundary, revisão de trust policies, Access Analyzer.
  - E2: Privilege escalation no SO/containers. Mitigação: hardening CIS, patching automatizado (SSM Patch Manager), execução como usuário não-root, SELinux/apparmor.
  - E3: APIs administrativas de Solr. Mitigação: RBAC, isolamento de rede, assinatura de requisições internas, admin endpoints desabilitados.

- Priorização (exemplos)
  - Alto: S1, S4, I1, D1, T1/T2, E1.
  - Médio: D2, Solr hardening, logs imutáveis.
  - Baixo: headers/erros verbosos, otimizações de throughput.
  - Plano: tratar riscos altos em 30–60 dias; médios em 90 dias; baixos em 180 dias, com revisões trimestrais.

- Requisitos de controles principais
  - Rede: SG/NACL mínimos, VPC Endpoints, Network Firewall para egress.
  - Identidade: IAM least privilege, SCPs, MFA, SSO/OIDC.
  - Criptografia: KMS com rotação, TLS end-to-end, HSTS.
  - Observabilidade: CloudTrail org, logs imutáveis, GuardDuty/Detective/Security Hub.
  - Continuidade: Multi-AZ, DR cross-region (RDS/EFS/Backups), testes de restauração.

Additional resourcing needs:
- Gestor do projeto/STRIDE: coordena workshops, priorização de riscos e plano de ação.
- Network Administrator: desenho de subnets/roteamento, SG/NACL, VPC Endpoints, Network Firewall e regras de egress.
- Security Officer: políticas IAM/SCP, classificação de dados (LGPD), padrões de criptografia/KMS, governança de logs/backup.
- Cloud/DevOps: IaC (CloudFormation/Terraform), AMIs “golden”, CI/CD, observabilidade, automação de patching (SSM), WAF/Shield.
- Software Developer: integração de autenticação (OIDC/MFA), gestão de sessões, proteção contra SSRF/OWASP Top 10, uso de RDS Proxy/ElastiCache.
- Hardware/Infra (quando aplicável): proteção de credenciais em appliances, rotação de chaves e boot seguro.
- DBA: modelagem/índices, replicação/DR, RDS parameter groups, RDS Proxy.
- SecOps/Blue team: monitoração em SIEM, regras de detecção, testes de restauração e exercícios de resposta.
- Estimativa de esforço inicial:
  - Descoberta e modelagem (DFD/UML, trust boundaries): 1–2 semanas.
  - Workshops STRIDE por domínios (Borda, App, Dados, Operações): 3–5 sessões de 1–2h.
  - Plano de tratamento e IaC de controles prioritários: 2–4 semanas.
  - Testes (pen test focado em WAF/borda e revisão de IAM): 1–2 semanas.

Observações finais:
- Considere migrar Memcached para Redis com TLS/ACL.
- Garanta que o ALB não seja acessível publicamente exceto por CloudFront.
- Adote VPC Endpoints para tráfego a serviços AWS a partir das subnets privadas.
- Habilite CloudTrail em todas as regiões e logs imutáveis.
- Formalize DR cross-region e treine o runbook de recuperação.

Sugestões de melhorias

A seguir apresento:
1) Diagrama mermaid com a arquitetura, relações e correções aplicadas.
2) Relatório de Modelagem de Ameaças (STRIDE), consolidado e priorizado.
3) Um Terraform de referência (opinionated) para materializar a solução na AWS sa-east-1, incluindo VPC multi-AZ, ALB, ASGs (app e Solr), RDS Multi-AZ com RDS Proxy, ElastiCache Redis com TLS/ACL, EFS com Access Points, CloudFront+WAF+Shield, VPC Endpoints, CloudTrail com S3 Object Lock, AWS Backup Vault Lock, KMS e integrações de observabilidade. Ajuste variáveis, nomes, capacidades e AMIs antes de aplicar.

Diagrama mermaid
```mermaid
flowchart TD
  U[Usuários SEI]:::actor

  subgraph EDGE[CDN/Borda Global]
    CF[Amazon CloudFront]:::edge
    WAF[WebACL AWS WAFv2]:::sec
```

```
  SH[Shield Advanced]:::sec
end

subgraph AWS[Conta AWS - Região sa-east-1]
  subgraph VPC[VPC 10.0.0.0/16 - 3 AZs]
    subgraph PUB[Subnets Públicas (A,B,C)]
      ALB[(Application Load Balancer - HTTPS 443)]:::lb
      note right of ALB
        SG permite APENAS:
        - Prefix List CloudFront origin-facing
        - IPs de gestão (opcional)
        Listener 443:
        - Default 403
        - Regra exige header secreto de origem
      end
    end

    subgraph PRIV[Subnets Privadas (A,B,C)]
      ASG[EC2 Auto Scaling - App SEI/SIP]:::app
      IMDS[IMDSv2 obrigatório (hop-limit=1)]:::sec

      RDSX[(Amazon RDS Multi-AZ - Primary/Standby)]:::data
      RDSP[(RDS Proxy - pooling TLS)]:::data

      EFS[(Amazon EFS - Multi-AZ + Access Points)]:::data

      REDIS[(ElastiCache Redis Replication Group - TLS + ACL)]:::data

      subgraph SOLR[Camada de Busca]
        ALBI[(ALB Interno - TLS)]:::lb
        SOLRASG[EC2 Auto Scaling - Solr]:::app
      end

      VPCe[[VPC Endpoints:
        S3 (Gateway),
        KMS, CW Logs, CloudTrail,
        Secrets Manager, SSM/*,
        SES SMTP, STS (opcional)]]:::net
    end
  end

  subgraph OPS[Operações/Segurança]
    CW[CloudWatch (logs/métricas)]:::ops
    CT[CloudTrail org/all-regions]:::ops
    S3L[(S3 Logs com Object Lock WORM + MFA Delete)]:::sec
    KMS[AWS KMS (chaves p/ RDS/EFS/EBS/Logs/Backup)]:::sec
    BKP[AWS Backup (Vault Lock + planos RDS/EFS)]:::ops
    SES[Amazon SES (SPF/DKIM/DMARC)]:::ops
  end
end

%% Fluxo de usuários
U --> CF --> WAF --> ALB --> ASG

%% Interações da aplicação
ASG -- TLS --> RDSP --> RDSX
ASG -- NFS/TLS --> EFS
ASG -- TLS/ACL --> REDIS
ASG -- TLS --> ALBI --> SOLRASG
ASG -- SMTP TLS --> SES

%% Observabilidade e trilhas
ALB -. logs .-> CW
WAF -. logs .-> CW
ASG -. app/sys logs .-> CW
CT --> S3L
CW -. KMS enc .-> S3L
BKP -. snapshots/retention .-> RDSX
BKP -. backups/retention .-> EFS
```

```
  %% Controles principais
  SH --- CF
  KMS --- RDSX
  KMS --- EFS
  KMS --- CW
  KMS --- BKP

  classDef actor fill:#eef,stroke:#446;
  classDef edge fill:#fff3e6,stroke:#f59e0b;
  classDef lb fill:#e6f7ff,stroke:#0ea5e9;
  classDef app fill:#eef,stroke:#6366f1;
  classDef data fill:#e7f6e7,stroke:#16a34a;
  classDef net fill:#f1f5f9,stroke:#64748b;
  classDef ops fill:#f3e8ff,stroke:#a855f7;
  classDef sec fill:#ffe4e6,stroke:#ef4444;
```

Principais correções aplicadas no desenho
- Bypass CloudFront/WAF: ALB público aceita apenas CloudFront (managed prefix list) e valida header secreto; WAF também aplicado na borda; default action 403.
- Criptografia e segredos: TLS 1.2/1.3 fim-a-fim; KMS para dados/backup/logs; Secrets Manager + RDS Proxy; IMDSv2 obrigatório.
- Cache: migração Memcached → ElastiCache Redis com TLS e ACLs.
- Egress: VPC Endpoints para AWS (S3, KMS, CW Logs, CloudTrail, SES SMTP, SSM, Secrets); NAT opcional com egress estrito.
- EFS: Access Points, SG dedicado porta 2049 somente da app; encriptação em trânsito e repouso.
- Solr: ALB interno com TLS; SG e WAF interno opcional; hardening e RBAC.
- Logs/Auditoria: CloudTrail all-regions para S3 com Object Lock; CloudWatch centralizado; Backup Vault Lock.
- DoS: Shield Advanced em CloudFront; WAF managed + rate-limit; RDS Proxy; auto scaling.

Relatório de Modelagem de Ameaças (STRIDE)
- Escopo e ativos
  - Escopo: Borda (CloudFront/WAF/Shield), App (EC2 ASG), Dados (RDS, Redis, EFS, Solr), Operações (SES, CloudWatch, CloudTrail, KMS, Backup), VPC multi-AZ/endpoints.
  - Ativos: documentos/processos SEI, credenciais/segredos, disponibilidade, logs/auditoria, reputação de e-mail, chaves KMS.
  - Fronteiras: Internet→Borda; Borda→ALB (público); Público→Privado; App→Dados; VPC→Serviços gerenciados (endpoints/egress).
- Spoofing
  - S1: Bypass borda. Mitigação: SG do ALB com prefix list do CloudFront + header secreto + WAF no ALB se necessário.
  - S2: Sessões. TLS, cookies Secure/HttpOnly/SameSite, MFA/IdP OIDC, rate-limit login.
  - S3: SES spoofing. SPF, DKIM, DMARC e segregação por ambiente.
  - S4: Credenciais IMDS. IMDSv2/hop-limit=1; WAF contra SSRF; IAM condicional.
- Tampering
  - T1: Trânsito ALB↔EC2↔RDS/EFS/Solr. TLS 1.2/1.3 obrigatório e certificados válidos.
  - T2: Repouso (EFS/RDS/logs). KMS, EFS APs/POSIX; S3 Object Lock.
  - T3: Pipeline/AMIs. AMIs golden assinadas; CI com aprovações e varreduras.
- Repudiation
  - R1: Sem trilhas. CloudTrail org/all-regions; logs ALB/CF/WAF em S3 com Object Lock; NTP/Time Sync; SIEM.
  - R2: Operadores alteram logs. Conta “log archive”, acesso read-only; Vault Lock/retention.
- Information Disclosure
  - I1: Vazamento EFS/backups/snapshots. SG/NACL mínimos; EFS APs; criptografia; DLP e revisão de compartilhamentos.
  - I2: Segredos. Secrets Manager/Parameter Store KMS; proibir segredos no FS/vars.
  - I3: Erros verbosos/headers. Sanitização e headers (HSTS, CSP, X-Frame-Options).
- Denial of Service
  - D1: DDoS borda. Shield Adv, WAF managed+rate-limit, caching CF, circuit breakers.
  - D2: Saturação RDS/Solr. RDS Proxy, índices/limites, read replicas, autoscaling Solr, filas/retentativas.
  - D3: EFS throughput. Modo provisionado/adaptativo; cache local.
- Elevation of Privilege
  - E1: Escalada IAM. Least privilege, boundaries, revisão de trust, Access Analyzer.
  - E2: Escalada SO. Hardening CIS, patching (SSM), usuário não-root, SELinux/apparmor.
  - E3: Admin Solr. RBAC, isolamento de rede, desabilitar endpoints perigosos.
- Priorização

  - Alto: S1, S4, I1, D1, T1/T2, E1. Médio: D2, Solr hardening, logs imutáveis. Baixo: headers/erros.
  - Prazos: altos 30–60d; médios 90d; baixos 180d; revisões trimestrais.
- Controles principais
  - Rede: SG/NACL mínimos, VPC Endpoints, egress firewall se NAT.
  - Identidade: IAM least privilege, SCPs, MFA/SSO/OIDC.
  - Criptografia: KMS com rotação, TLS end-to-end, HSTS.
  - Observabilidade: CloudTrail org, logs imutáveis, GuardDuty/Detective/Security Hub.
  - Continuidade: Multi-AZ, DR cross-region (RDS/EFS/Backups), testes de restauração.

Terraform de referência
Observações:
- Substitua valores dos certificados ACM: CloudFront usa ACM em us-east-1; o ALB usa ACM em sa-east-1.
- Ajuste tamanhos/capacidades (RDS/Redis/EFS) e AMIs nas launch templates.
- O código foca em controles críticos e pode ser expandido (por exemplo, regras WAF adicionais, Network Firewall).

Arquivo único (main.tf) para iniciar:
```hcl
terraform {
  required_version = ">= 1.5.0"
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = ">= 5.49.0"
    }
    random = {
      source  = "hashicorp/random"
      version = ">= 3.5.1"
    }
  }
}

provider "aws" {
  region = var.region
}

#########################
# Variáveis
#########################
variable "region" {
  type    = string
  default = "sa-east-1"
}
variable "project" {
  type    = string
  default = "sei"
}
variable "vpc_cidr" {
  type    = string
  default = "10.0.0.0/16"
}
variable "enable_nat" {
  type    = bool
  default = false
}
variable "domain_name" {
  description = "Domínio público do SEI (ex: sei.gov.br)"
  type        = string
}
variable "alb_acm_cert_arn" {
  description = "Certificado ACM (sa-east-1) para o ALB (TLS 443)."
  type        = string
}
variable "cloudfront_acm_cert_arn" {
  description = "Certificado ACM (us-east-1) para o CloudFront (CNAME do domínio)."
  type        = string
}
variable "origin_header_secret" {
```

```
  description = "Segredo para header de origem (X-Origin-Verify)."
  type        = string
  sensitive   = true
}
variable "allowed_admin_cidrs" {
  description = "CIDRs de administração (bastion/VPN) para acesso ao ALB se necessário."
  type        = list(string)
  default     = []
}
variable "app_instance_type" {
  type    = string
  default = "t3.large"
}
variable "solr_instance_type" {
  type    = string
  default = "t3.large"
}
variable "app_ami_id" {
  description = "AMI da aplicação (endurecida)."
  type        = string
}
variable "solr_ami_id" {
  description = "AMI do Solr (endurecida)."
  type        = string
}
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 "redis_node_type" {
  type    = string
  default = "cache.t4g.medium"
}

locals {
  tags = {
    Project = var.project
    Owner   = "TI"
    Env     = "prod"
  }
}

#########################
# AZs
#########################
data "aws_availability_zones" "this" {
  state = "available"
}

locals {
  azs = slice(data.aws_availability_zones.this.names, 0, 3)
}

#########################
# VPC e Subnets
#########################
resource "aws_vpc" "this" {
  cidr_block           = var.vpc_cidr
  enable_dns_support   = true
  enable_dns_hostnames = true
  tags = merge(local.tags, { Name = "${var.project}-vpc" })
```

```
}

resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.this.id
  tags   = merge(local.tags, { Name = "${var.project}-igw" })
}

# Subnets públicas e privadas
resource "aws_subnet" "public" {
  for_each = { for idx, az in local.azs : idx => az }
  vpc_id                  = aws_vpc.this.id
  availability_zone       = each.value
  cidr_block              = cidrsubnet(aws_vpc.this.cidr_block, 4, each.key) # /20
  map_public_ip_on_launch = true
  tags = merge(local.tags, { Name = "${var.project}-pub-${each.value}", Tier = "public" })
}

resource "aws_subnet" "private" {
  for_each = { for idx, az in local.azs : idx => az }
  vpc_id            = aws_vpc.this.id
  availability_zone = each.value
  cidr_block        = cidrsubnet(aws_vpc.this.cidr_block, 4, each.key + 8) # /20 outro bloco
  tags = merge(local.tags, { Name = "${var.project}-priv-${each.value}", Tier = "private" })
}

# Rotas públicas
resource "aws_route_table" "public" {
  vpc_id = aws_vpc.this.id
  tags   = merge(local.tags, { Name = "${var.project}-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" {
  for_each       = aws_subnet.public
  subnet_id      = each.value.id
  route_table_id = aws_route_table.public.id
}

# NAT opcional (por AZ)
resource "aws_eip" "nat" {
  for_each = var.enable_nat ? aws_subnet.public : {}
  domain   = "vpc"
  tags     = merge(local.tags, { Name = "${var.project}-eip-nat-${each.key}" })
}
resource "aws_nat_gateway" "nat" {
  for_each      = var.enable_nat ? aws_subnet.public : {}
  allocation_id = aws_eip.nat[each.key].id
  subnet_id     = each.value.id
  tags          = merge(local.tags, { Name = "${var.project}-nat-${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     = merge(local.tags, { Name = "${var.project}-rtb-priv-${each.key}" })
}
resource "aws_route" "private_nat" {
  for_each               = var.enable_nat ? 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[each.key].id
}
resource "aws_route_table_association" "private" {
  for_each       = aws_subnet.private
  subnet_id      = each.value.id
  route_table_id = aws_route_table.private[each.key].id
```

```
}

#########################
# KMS CMK
#########################
resource "aws_kms_key" "main" {
  description             = "KMS para RDS/EFS/EBS/Logs/Backups (${var.project})"
  enable_key_rotation     = true
  deletion_window_in_days = 30
  tags                    = local.tags
}
resource "aws_kms_alias" "main" {
  name          = "alias/${var.project}-main"
  target_key_id = aws_kms_key.main.key_id
}

########################
# CloudWatch Log Groups
########################
resource "aws_cloudwatch_log_group" "app" {
  name              = "/${var.project}/app"
  retention_in_days = 120
  kms_key_id        = aws_kms_key.main.arn
  tags              = local.tags
}
resource "aws_cloudwatch_log_group" "alb" {
  name              = "/${var.project}/alb"
  retention_in_days = 120
  kms_key_id        = aws_kms_key.main.arn
  tags              = local.tags
}
resource "aws_cloudwatch_log_group" "waf" {
  name              = "/${var.project}/waf"
  retention_in_days = 120
  kms_key_id        = aws_kms_key.main.arn
  tags              = local.tags
}

########################
# S3 bucket de logs (Object Lock)
########################
resource "aws_s3_bucket" "logs" {
  bucket              = "${var.project}-logs-${data.aws_caller_identity.me.account_id}"
  object_lock_enabled = true
  tags                = local.tags
}
data "aws_caller_identity" "me" {}

resource "aws_s3_bucket_versioning" "logs" {
  bucket = aws_s3_bucket.logs.id
  versioning_configuration { status = "Enabled" }
}
resource "aws_s3_bucket_server_side_encryption_configuration" "logs" {
  bucket = aws_s3_bucket.logs.id
  rule {
    apply_server_side_encryption_by_default {
      sse_algorithm     = "aws:kms"
      kms_master_key_id = aws_kms_key.main.arn
    }
  }
}
resource "aws_s3_bucket_object_lock_configuration" "logs" {
  bucket = aws_s3_bucket.logs.id
  rule {
    default_retention {
      mode  = "COMPLIANCE"
      days  = 365
    }
  }
```

```
}
resource "aws_s3_bucket_public_access_block" "logs" {
  bucket                  = aws_s3_bucket.logs.id
  block_public_acls       = true
  block_public_policy     = true
  ignore_public_acls      = true
  restrict_public_buckets = true
}

#########################
# CloudTrail org/all-regions
#########################
resource "aws_cloudtrail" "this" {
  name                          = "${var.project}-trail"
  s3_bucket_name                = aws_s3_bucket.logs.id
  include_global_service_events = true
  is_multi_region_trail         = true
  enable_logging                = true
  kms_key_id                    = aws_kms_key.main.arn
  event_selector {
    read_write_type           = "All"
    include_management_events = true
  }
  tags = local.tags
}

#########################
# VPC Endpoints (PrivateLink)
#########################
# SG para endpoints interface
resource "aws_security_group" "vpce" {
  name        = "${var.project}-sg-vpce"
  description = "SG para endpoints de interface"
  vpc_id      = aws_vpc.this.id
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = local.tags
}

# Gateway endpoint S3
resource "aws_vpc_endpoint" "s3" {
  vpc_id            = aws_vpc.this.id
  service_name      = "com.amazonaws.${var.region}.s3"
  vpc_endpoint_type = "Gateway"
  route_table_ids   = concat([aws_route_table.public.id], [for rt in aws_route_table.private : rt.id])
  tags              = local.tags
}

# Interface endpoints mais usados
locals {
  interface_endpoints = [
    "kms",
    "logs",
    "monitoring",
    "events",
    "cloudtrail",
    "secretsmanager",
    "ssm",
    "ssmmessages",
    "ec2messages",
    "email-smtp"
  ]
}
resource "aws_vpc_endpoint" "iface" {
  for_each          = toset(local.interface_endpoints)
```

```
  vpc_id            = aws_vpc.this.id
  service_name      = "com.amazonaws.${var.region}.${each.key}"
  vpc_endpoint_type = "Interface"
  private_dns_enabled = true
  subnet_ids        = [for s in aws_subnet.private : s.id]
  security_group_ids = [aws_security_group.vpce.id]
  tags              = merge(local.tags, { Service = each.key })
}

#########################
# Security Groups principais
#########################
# Prefix list do CloudFront origin-facing
data "aws_ec2_managed_prefix_list" "cloudfront" {
  name = "com.amazonaws.global.cloudfront.origin-facing"
}

resource "aws_security_group" "alb" {
  name        = "${var.project}-sg-alb"
  description = "ALB público (HTTPS) - somente CloudFront e admins"
  vpc_id      = aws_vpc.this.id

  # Egress liberado; o controle é por ingress nos destinos
  egress { from_port = 0, to_port = 0, protocol = "-1", cidr_blocks = ["0.0.0.0/0"] }
  tags = local.tags
}
# Ingress do ALB só do CloudFront (e admins opcionais)
resource "aws_security_group_rule" "alb_ing_cf" {
  type              = "ingress"
  security_group_id = aws_security_group.alb.id
  from_port         = 443
  to_port           = 443
  protocol          = "tcp"
  prefix_list_ids   = [data.aws_ec2_managed_prefix_list.cloudfront.id]
  description       = "CloudFront origin-facing only"
}
resource "aws_security_group_rule" "alb_ing_admin" {
  for_each          = toset(var.allowed_admin_cidrs)
  type              = "ingress"
  security_group_id = aws_security_group.alb.id
  from_port         = 443
  to_port           = 443
  protocol          = "tcp"
  cidr_blocks       = [each.value]
  description       = "Admin CIDR"
}

resource "aws_security_group" "app" {
  name        = "${var.project}-sg-app"
  description = "EC2 App"
  vpc_id      = aws_vpc.this.id
  egress { from_port = 0, to_port = 0, protocol = "-1", cidr_blocks = ["0.0.0.0/0"] }
  tags = local.tags
}
resource "aws_security_group" "solr" {
  name        = "${var.project}-sg-solr"
  description = "EC2 Solr"
  vpc_id      = aws_vpc.this.id
  egress { from_port = 0, to_port = 0, protocol = "-1", cidr_blocks = ["0.0.0.0/0"] }
  tags = local.tags
}
resource "aws_security_group" "alb_internal" {
  name        = "${var.project}-sg-alb-int"
  description = "ALB interno p/ Solr"
  vpc_id      = aws_vpc.this.id
  egress { from_port = 0, to_port = 0, protocol = "-1", cidr_blocks = ["0.0.0.0/0"] }
  tags = local.tags
}
# Permitir app -> ALB interno (443)
```

```
resource "aws_security_group_rule" "albint_ing_app" {
  type                     = "ingress"
  security_group_id        = aws_security_group.alb_internal.id
  from_port                = 443
  to_port                  = 443
  protocol                 = "tcp"
  source_security_group_id = aws_security_group.app.id
}

resource "aws_security_group" "rds" {
  name        = "${var.project}-sg-rds"
  description = "RDS"
  vpc_id      = aws_vpc.this.id
  egress { from_port = 0, to_port = 0, protocol = "-1", cidr_blocks = ["0.0.0.0/0"] }
  tags = local.tags
}
resource "aws_security_group" "rds_proxy" {
  name        = "${var.project}-sg-rds-proxy"
  description = "RDS Proxy"
  vpc_id      = aws_vpc.this.id
  egress { from_port = 0, to_port = 0, protocol = "-1", cidr_blocks = ["0.0.0.0/0"] }
  tags = local.tags
}
# Regras de fluxo DB
resource "aws_security_group_rule" "rds_ing_proxy" {
  type                     = "ingress"
  security_group_id        = aws_security_group.rds.id
  from_port                = var.db_engine == "postgres" ? 5432 : 3306
  to_port                  = var.db_engine == "postgres" ? 5432 : 3306
  protocol                 = "tcp"
  source_security_group_id = aws_security_group.rds_proxy.id
}
resource "aws_security_group_rule" "rdsproxy_ing_app" {
  type                     = "ingress"
  security_group_id        = aws_security_group.rds_proxy.id
  from_port                = var.db_engine == "postgres" ? 5432 : 3306
  to_port                  = var.db_engine == "postgres" ? 5432 : 3306
  protocol                 = "tcp"
  source_security_group_id = aws_security_group.app.id
}

resource "aws_security_group" "efs" {
  name        = "${var.project}-sg-efs"
  description = "EFS"
  vpc_id      = aws_vpc.this.id
  egress { from_port = 0, to_port = 0, protocol = "-1", cidr_blocks = ["0.0.0.0/0"] }
  tags = local.tags
}
resource "aws_security_group_rule" "efs_ing_app" {
  type                     = "ingress"
  security_group_id        = aws_security_group.efs.id
  from_port                = 2049
  to_port                  = 2049
  protocol                 = "tcp"
  source_security_group_id = aws_security_group.app.id
}

resource "aws_security_group" "redis" {
  name        = "${var.project}-sg-redis"
  description = "ElastiCache Redis"
  vpc_id      = aws_vpc.this.id
  egress { from_port = 0, to_port = 0, protocol = "-1", cidr_blocks = ["0.0.0.0/0"] }
  tags = local.tags
}
resource "aws_security_group_rule" "redis_ing_app" {
  type                     = "ingress"
  security_group_id        = aws_security_group.redis.id
  from_port                = 6379
  to_port                  = 6379
```

```
  protocol                 = "tcp"
  source_security_group_id = aws_security_group.app.id
}

#########################
# ALB público (HTTPS)
#########################
resource "aws_lb" "public" {
  name               = "${var.project}-alb"
  internal           = false
  load_balancer_type = "application"
  security_groups    = [aws_security_group.alb.id]
  subnets            = [for s in aws_subnet.public : s.id]
  enable_deletion_protection = true
  tags = local.tags
}

resource "aws_lb_target_group" "app" {
  name        = "${var.project}-tg-app"
  port        = 443
  protocol    = "HTTPS"
  vpc_id      = aws_vpc.this.id
  target_type = "instance"
  health_check {
    protocol            = "HTTPS"
    path                = "/health"
    matcher             = "200-399"
    healthy_threshold   = 2
    unhealthy_threshold = 5
    timeout             = 5
    interval            = 30
  }
  tags = local.tags
}

resource "aws_lb_listener" "https" {
  load_balancer_arn = aws_lb.public.arn
  port              = 443
  protocol          = "HTTPS"
  ssl_policy        = "ELBSecurityPolicy-TLS13-1-2-2021-06"
  certificate_arn   = var.alb_acm_cert_arn
  default_action {
    type = "fixed-response"
    fixed_response {
      content_type = "text/plain"
      message_body = "Forbidden"
      status_code  = "403"
    }
  }
}

# Regra que exige header secreto para encaminhar à app
resource "aws_lb_listener_rule" "origin_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-Origin-Verify"
      values           = [var.origin_header_secret]
    }
  }
}

#########################
# Launch Templates (IMDSv2)
```

```
########################
resource "aws_iam_role" "ec2" {
  name               = "${var.project}-ec2-role"
  assume_role_policy = data.aws_iam_policy_document.ec2_trust.json
  tags               = local.tags
}
data "aws_iam_policy_document" "ec2_trust" {
  statement {
    actions = ["sts:AssumeRole"]
    principals { type = "Service", identifiers = ["ec2.amazonaws.com"] }
  }
}
# Permissões mínimas (SSM, CW logs, Secrets/Get...)
resource "aws_iam_role_policy_attachment" "ssm" {
  role       = aws_iam_role.ec2.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
}
resource "aws_iam_role_policy" "ec2_inline" {
  name = "${var.project}-ec2-inline"
  role = aws_iam_role.ec2.id
  policy = jsonencode({
    Version = "2012-10-17",
    Statement = [
      { Effect = "Allow", Action =
["logs:CreateLogGroup","logs:CreateLogStream","logs:PutLogEvents"], Resource = "*" },
      { Effect = "Allow", Action = ["secretsmanager:GetSecretValue"], Resource = "*" }
    ]
  })
}
resource "aws_iam_instance_profile" "ec2" {
  name = "${var.project}-ec2-profile"
  role = aws_iam_role.ec2.name
}

# User data enxuto (exemplo), exigir TLS p/ EFS no fstab, IMDSv2 enforced pelo LT
locals {
  app_user_data = <<-EOF
    #!/bin/bash
    set -euo pipefail
    yum -y update
    # Exemplo de montagem EFS via TLS (substituir fs-xxxx e AP)
    # amazon-linux-extras install -y efs-utils
    # mkdir -p /mnt/efs
    # echo "fs-XXXXXX:/ /mnt/efs efs _netdev,tls,accesspoint=fsap-XXXX 0 0" >> /etc/fstab
    # mount -a
  EOF

  solr_user_data = <<-EOF
    #!/bin/bash
    set -euo pipefail
    yum -y update
  EOF
}

resource "aws_launch_template" "app" {
  name_prefix   = "${var.project}-lt-app-"
  image_id      = var.app_ami_id
  instance_type = var.app_instance_type
  iam_instance_profile { name = aws_iam_instance_profile.ec2.name }
  vpc_security_group_ids = [aws_security_group.app.id]
  user_data     = base64encode(local.app_user_data)
  metadata_options {
    http_tokens              = "required"
    http_put_response_hop_limit = 1
  }
  tag_specifications {
    resource_type = "instance"
    tags          = merge(local.tags, { Role = "app" })
  }
```

```
  tags = local.tags
}

resource "aws_launch_template" "solr" {
  name_prefix   = "${var.project}-lt-solr-"
  image_id      = var.solr_ami_id
  instance_type = var.solr_instance_type
  iam_instance_profile { name = aws_iam_instance_profile.ec2.name }
  vpc_security_group_ids = [aws_security_group.solr.id]
  user_data     = base64encode(local.solr_user_data)
  metadata_options {
    http_tokens              = "required"
    http_put_response_hop_limit = 1
  }
  tag_specifications {
    resource_type = "instance"
    tags          = merge(local.tags, { Role = "solr" })
  }
  tags = local.tags
}

#########################
# Auto Scaling Groups
#########################
resource "aws_autoscaling_group" "app" {
  name                      = "${var.project}-asg-app"
  desired_capacity          = 3
  min_size                  = 3
  max_size                  = 9
  vpc_zone_identifier       = [for s in aws_subnet.private : s.id]
  health_check_type         = "EC2"
  target_group_arns         = [aws_lb_target_group.app.arn]
  launch_template {
    id      = aws_launch_template.app.id
    version = "$Latest"
  }
  tag { key = "Name", value = "${var.project}-app", propagate_at_launch = true }
}

# ALB interno para Solr
resource "aws_lb" "internal" {
  name               = "${var.project}-alb-int"
  internal           = true
  load_balancer_type = "application"
  security_groups    = [aws_security_group.alb_internal.id]
  subnets            = [for s in aws_subnet.private : s.id]
  tags               = local.tags
}
resource "aws_lb_target_group" "solr" {
  name        = "${var.project}-tg-solr"
  port        = 8983
  protocol    = "HTTP"
  vpc_id      = aws_vpc.this.id
  target_type = "instance"
  health_check { path = "/solr/admin/ping", matcher = "200-399" }
  tags = local.tags
}
resource "aws_lb_listener" "internal_https" {
  load_balancer_arn = aws_lb.internal.arn
  port              = 443
  protocol          = "HTTPS"
  ssl_policy        = "ELBSecurityPolicy-TLS13-1-2-2021-06"
  certificate_arn   = var.alb_acm_cert_arn
  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.solr.arn
  }
}
resource "aws_autoscaling_group" "solr" {
```

```
  name                = "${var.project}-asg-solr"
  desired_capacity    = 3
  min_size            = 3
  max_size            = 6
  vpc_zone_identifier = [for s in aws_subnet.private : s.id]
  target_group_arns   = [aws_lb_target_group.solr.arn]
  launch_template {
    id      = aws_launch_template.solr.id
    version = "$Latest"
  }
  tag { key = "Name", value = "${var.project}-solr", propagate_at_launch = true }
}

#########################
# EFS + Access Point
#########################
resource "aws_efs_file_system" "this" {
  encrypted        = true
  kms_key_id       = aws_kms_key.main.arn
  performance_mode = "generalPurpose"
  throughput_mode  = "elastic"
  tags             = merge(local.tags, { Name = "${var.project}-efs" })
}
resource "aws_efs_mount_target" "mt" {
  for_each       = aws_subnet.private
  file_system_id = aws_efs_file_system.this.id
  subnet_id      = each.value.id
  security_groups = [aws_security_group.efs.id]
}
resource "aws_efs_access_point" "ap" {
  file_system_id = aws_efs_file_system.this.id
  posix_user { gid = 1000, uid = 1000 }
  root_directory {
    path = "/app"
    creation_info { owner_gid = 1000, owner_uid = 1000, permissions = "0750" }
  }
  tags = merge(local.tags, { Name = "${var.project}-efs-ap-app" })
}

#########################
# RDS + RDS Proxy + Secrets
#########################
resource "random_password" "db" {
  length           = 24
  special          = true
  override_characters = "!@#%^*-_=+"
}

resource "aws_secretsmanager_secret" "db" {
  name = "${var.project}/db/credentials"
  kms_key_id = aws_kms_key.main.arn
  tags = local.tags
}
resource "aws_secretsmanager_secret_version" "db" {
  secret_id     = aws_secretsmanager_secret.db.id
  secret_string = jsonencode({
    username = "seiapp",
    password = random_password.db.result
  })
}

resource "aws_db_subnet_group" "this" {
  name       = "${var.project}-db-subnets"
  subnet_ids = [for s in aws_subnet.private : s.id]
  tags       = local.tags
}
resource "aws_db_instance" "this" {
  identifier                = "${var.project}-rds"
  engine                    = var.db_engine
```

```
  engine_version            = var.db_engine_version
  instance_class            = var.db_instance_class
  db_subnet_group_name      = aws_db_subnet_group.this.name
  multi_az                  = true
  allocated_storage         = 100
  max_allocated_storage     = 1000
  storage_encrypted         = true
  kms_key_id                = aws_kms_key.main.arn
  username                  = "admin"
  password                  = random_password.db.result
  vpc_security_group_ids    = [aws_security_group.rds.id]
  backup_retention_period   = 7
  deletion_protection       = true
  skip_final_snapshot       = false
  apply_immediately         = false
  publicly_accessible       = false
  performance_insights_enabled = true
  performance_insights_kms_key_id = aws_kms_key.main.arn
  tags = local.tags
}

resource "aws_iam_role" "rds_proxy" {
  name = "${var.project}-rds-proxy-role"
  assume_role_policy = jsonencode({
    Version="2012-10-17",

Statement=[{Effect="Allow",Principal={Service="rds.amazonaws.com"},Action="sts:AssumeRole"}]
  })
  tags = local.tags
}
resource "aws_iam_role_policy" "rds_proxy" {
  name = "${var.project}-rds-proxy-secret"
  role = aws_iam_role.rds_proxy.id
  policy = jsonencode({
    Version="2012-10-17",

Statement=[{Effect="Allow",Action=["secretsmanager:GetSecretValue"],Resource=[aws_secretsmana
ger_secret.db.arn]}]
  })
}
resource "aws_db_proxy" "this" {
  name                   = "${var.project}-proxy"
  engine_family          = upper(var.db_engine) # POSTGRES/MARIADB
  idle_client_timeout    = 1800
  require_tls            = true
  role_arn               = aws_iam_role.rds_proxy.arn
  vpc_security_group_ids = [aws_security_group.rds_proxy.id]
  vpc_subnet_ids         = [for s in aws_subnet.private : s.id]
  auth {
    auth_scheme = "SECRETS"
    iam_auth    = "DISABLED"
    secret_arn  = aws_secretsmanager_secret.db.arn
  }
  tags = local.tags
}
resource "aws_db_proxy_default_target_group" "this" {
  db_proxy_name = aws_db_proxy.this.name
}
resource "aws_db_proxy_target" "this" {
  db_proxy_name          = aws_db_proxy.this.name
  target_group_name      = aws_db_proxy_default_target_group.this.name
  db_instance_identifier = aws_db_instance.this.id
}

#########################
# ElastiCache Redis (TLS/ACL)
#########################
resource "aws_elasticache_subnet_group" "redis" {
  name       = "${var.project}-redis-subnets"
```

```
  subnet_ids = [for s in aws_subnet.private : s.id]
}
resource "aws_elasticache_user" "app" {
  user_id       = "${var.project}-redis-user"
  user_name     = "app"
  access_string = "on ~* +@all" # restringir por comandos/keys conforme necessidade
  no_password_required = false
  passwords     = [random_password.db.result]
}
resource "aws_elasticache_user_group" "app" {
  engine        = "REDIS"
  user_group_id = "${var.project}-redis-ug"
  user_ids      = [aws_elasticache_user.app.user_id]
}
resource "aws_elasticache_replication_group" "redis" {
  replication_group_id          = "${var.project}-redis"
  description                   = "Redis TLS+AUTH"
  node_type                     = var.redis_node_type
  number_cache_clusters         = 2
  automatic_failover_enabled    = true
  multi_az_enabled              = true
  at_rest_encryption_enabled    = true
  transit_encryption_enabled    = true
  security_group_ids            = [aws_security_group.redis.id]
  subnet_group_name             = aws_elasticache_subnet_group.redis.name
  user_group_ids                = [aws_elasticache_user_group.app.id]
  parameter_group_name          = "default.redis7"
  port                          = 6379
  tags = local.tags
}

##########################
# WAFv2 para CloudFront
##########################
resource "aws_wafv2_web_acl" "cf" {
  name  = "${var.project}-waf-cf"
  scope = "CLOUDFRONT"
  default_action { allow {} }
  rule {
    name     = "AWS-AWSManagedRulesCommonRuleSet"
    priority = 1
    statement { managed_rule_group_statement { name = "AWSManagedRulesCommonRuleSet"
vendor_name = "AWS" } }
    override_action { none {} }
    visibility_config { cloudwatch_metrics_enabled = true, metric_name = "common",
sampled_requests_enabled = true }
  }
  rule {
    name     = "RateLimit"
    priority = 10
    statement { rate_based_statement { limit = 2000, aggregate_key_type = "IP" } }
    action { block {} }
    visibility_config { cloudwatch_metrics_enabled = true, metric_name = "rate",
sampled_requests_enabled = true }
  }
  visibility_config { cloudwatch_metrics_enabled = true, metric_name = "${var.project}-waf",
sampled_requests_enabled = true }
  tags = local.tags
}

##########################
# CloudFront + Shield Advanced
##########################
resource "aws_cloudfront_distribution" "this" {
  enabled             = true
  comment             = "${var.project} distribution"
  aliases             = [var.domain_name]
  is_ipv6_enabled     = true
  price_class         = "PriceClass_200"
```

```
  default_cache_behavior {
    allowed_methods  = ["GET","HEAD","OPTIONS","PUT","POST","PATCH","DELETE"]
    cached_methods   = ["GET","HEAD"]
    target_origin_id = "alb-origin"
    viewer_protocol_policy = "redirect-to-https"
    compress = true
    forwarded_values {
      query_string = true
      headers      = ["Authorization","*"]
      cookies { forward = "all" }
    }
  }
  origin {
    domain_name = aws_lb.public.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"]
    }
    custom_header {
      name  = "X-Origin-Verify"
      value = var.origin_header_secret
    }
  }
  restrictions { geo_restriction { restriction_type = "none" } }
  viewer_certificate {
    acm_certificate_arn = var.cloudfront_acm_cert_arn
    ssl_support_method  = "sni-only"
    minimum_protocol_version = "TLSv1.2_2021"
  }
  web_acl_id = aws_wafv2_web_acl.cf.arn
  tags       = local.tags
}

resource "aws_shield_protection" "cf" {
  resource_arn = aws_cloudfront_distribution.this.arn
  name         = "${var.project}-shield-cf"
}

#########################
# EFS/ RDS Backup (Vault Lock)
#########################
resource "aws_backup_vault" "this" {
  name        = "${var.project}-vault"
  kms_key_arn = aws_kms_key.main.arn
  tags        = local.tags
}
resource "aws_backup_vault_lock_configuration" "this" {
  backup_vault_name = aws_backup_vault.this.name
  changeable_for_days = 3
  max_retention_days  = 3650
  min_retention_days  = 35
}
resource "aws_backup_plan" "this" {
  name = "${var.project}-backup"
  rule {
    rule_name         = "daily"
    target_vault_name = aws_backup_vault.this.name
    schedule          = "cron(0 3 * * ? *)"
    lifecycle { delete_after = 120 }
  }
  tags = local.tags
}
resource "aws_backup_selection" "rds" {
  name         = "rds-selection"
  iam_role_arn = data.aws_iam_role.backup.arn
  backup_plan_id = aws_backup_plan.this.id
```

```
  resources    = [aws_db_instance.this.arn]
}
resource "aws_backup_selection" "efs" {
  name         = "efs-selection"
  iam_role_arn = data.aws_iam_role.backup.arn
  backup_plan_id = aws_backup_plan.this.id
  resources    = [aws_efs_file_system.this.arn]
}
data "aws_iam_role" "backup" {
  name = "AWSBackupDefaultServiceRole"
}

#########################
# SES (Identidade do domínio) - DNS via Route53 não incluso
#########################
resource "aws_ses_domain_identity" "this" {
  domain = var.domain_name
}
resource "aws_ses_domain_dkim" "this" {
  domain = aws_ses_domain_identity.this.domain
}

#########################
# Outputs úteis
#########################
output "alb_dns" { value = aws_lb.public.dns_name }
output "cloudfront_domain" { value = aws_cloudfront_distribution.this.domain_name }
output "s3_logs_bucket" { value = aws_s3_bucket.logs.bucket }
output "rds_endpoint_via_proxy" { value = aws_db_proxy.this.endpoint }
output "redis_primary_endpoint" { value =
aws_elasticache_replication_group.redis.primary_endpoint_address }
```

Como usar
- Pré-requisitos:
  - Certificado ACM em us-east-1 para CloudFront e outro em sa-east-1 para ALB.
  - AMIs endurecidas para app e Solr; health endpoints configurados.
  - DNS (Route53) apontando CNAME do domínio para o CloudFront.
- Passos:
  - Ajuste variáveis (domínio, ARNs dos certificados, AMIs e capacidades).
  - terraform init; terraform plan; terraform apply.
- Próximos incrementos recomendados:
  - Regras WAF adicionais (OWASP, Bot Control), listas de allow/deny.
  - Regras de egress com AWS Network Firewall se habilitar NAT.
  - GuardDuty/Detective/Security Hub e AWS Config com regras de conformidade (snapshots públicos, etc.).
  - DR cross-region: read replica RDS, EFS replication, cópia de backups.