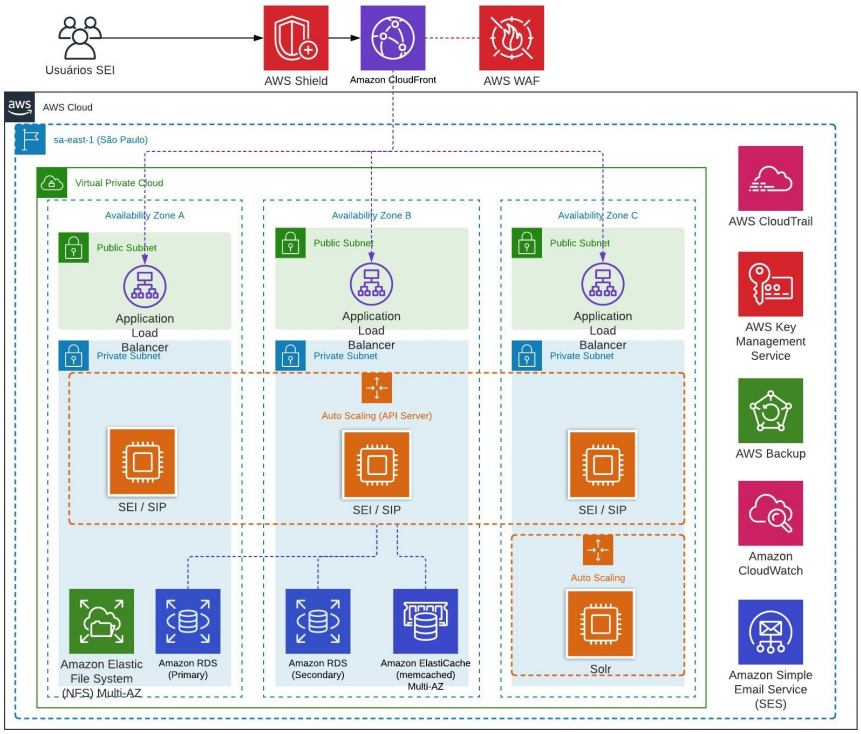
Relatório de análise da solução: tmp67wdmk6l



Análise completa da solução atual

Modelo de cloud:
- Amazon Web Services (AWS)

Lista com os componentes:
- Borda/Proteção: AWS Shield, Amazon CloudFront, AWS WAF
- Rede: VPC na região sa-east-1 (São Paulo), 3 Availability Zones (A, B, C), subnets públicas e privadas, Security Groups e NACLs
- Balanceamento: Application Load Balancer (ALB) em subnets públicas (multi-AZ)
- Camada de aplicação: Auto Scaling Group (API Server) com instâncias EC2 executando SEI/SIP nas AZs A e B
- Armazenamento de arquivos: Amazon Elastic File System (EFS) Multi-AZ
- Banco de dados: Amazon RDS (Primary em AZ A, Secondary em AZ B – Multi-AZ)
- Cache: Amazon ElastiCache (memcached) Multi-AZ
- Busca: Solr em Auto Scaling (indicado na AZ C)
- Observabilidade e governança: AWS CloudTrail, Amazon CloudWatch
- Criptografia e chaves: AWS Key Management Service (KMS)
- Backup: AWS Backup
- E-mail transacional: Amazon Simple Email Service (SES)
- Serviços de gerenciamento da AWS (p.ex., IAM/Systems Manager implícitos no desenho)

Interação entre os componentes:
- Usuários → CloudFront (com AWS Shield e WAF) → ALB público → EC2 (SEI/SIP) em subnets privadas
- EC2 (SEI/SIP) → RDS (leituras/escritas; failover para secundário em B)
- EC2 (SEI/SIP) → EFS (NFS) para conteúdo/artefatos compartilhados
- EC2 (SEI/SIP) → ElastiCache (memcached) para cache de sessão/dados
- EC2 (SEI/SIP) → Solr para indexação e busca
- EC2/ALB/CloudFront/WAF → CloudWatch (métricas/logs) e CloudTrail (trilhas de auditoria)
- Todos os dados em repouso/backup → KMS para chaves de criptografia + AWS Backup para planos de backup
- EC2/Aplicação → SES para envio de e-mails
- Tráfego norte-sul entra pela borda (CloudFront/WAF/Shield), é distribuído via ALB para instâncias EC2, que consomem serviços de dados (RDS/EFS/ElastiCache/Solr) em rede privada

O que esse sistema faz:
- Plataforma web de alta disponibilidade para o sistema SEI/SIP (provável gestão eletrônica de

informações/processos), exposta globalmente via CloudFront, com API/servidores de aplicação escaláveis, banco de dados relacional, cache de alto desempenho, busca full-text (Solr), armazenamento compartilhado de arquivos e envio de e-mails transacionais.

Vulnerabilidades e Solução para cada vulnerabilidade:
- Origin bypass ao ALB (acesso direto sem passar pelo WAF/CloudFront)
  • Solução: restringir Security Group do ALB para aceitar apenas IPs de CloudFront e/ou exigir cabeçalho secreto no ALB (Origin Custom Header) validado no listener/target; usar AWS WAF no ALB também.
- Regras WAF insuficientes para OWASP Top 10 e bots
  • Solução: habilitar AWS Managed Rules (OWASP, Bot Control), regras de rate limiting por IP, validação de payload, Geo/Country blocks conforme necessidade e logging do WAF para análise.
- TLS fraco ou expiração de certificados
  • Solução: forçar TLS 1.2+ no CloudFront/ALB, usar ACM para emissão/rotação automática, HSTS e políticas de ciphers modernas.
- DDoS em camadas L3/L7
  • Solução: AWS Shield (preferir Shield Advanced para detecções/mitigações e resposta 24x7), WAF rate-based rules e arquitetura elástica com auto scaling e limites de burst bem dimensionados.
- Acesso direto ao memcached (sem autenticação nativa)
  • Solução: isolar em subnets privadas; Security Group permitindo apenas tráfego das instâncias de aplicação; avaliar migração para Redis com AUTH e TLS se requisitos de segurança demandarem autenticação.
- EFS sem criptografia em trânsito/permissões frouxas
  • Solução: montar EFS com TLS; habilitar criptografia at-rest com KMS; usar EFS Access Points e POSIX (UID/GID) mínimos necessários; políticas de backup e lifecycle.
- RDS exposto/publicamente ou com credenciais fixas
  • Solução: manter RDS em subnets privadas; Security Groups restritivos; IAM Authentication (quando aplicável), rotação de credenciais no Secrets Manager, TDE/at-rest com KMS, forçar TLS em conexão.
- Solr acessível ou sem hardening
  • Solução: colocar Solr somente em rede privada; autenticação/autorização via security.json; TLS; hardening do SO; restringir para Security Group do app; autoscaling multi-AZ para evitar SPOF.
- Falta de segregação de papéis IAM e privilégio excessivo às instâncias
  • Solução: políticas IAM de menor privilégio; perfis de instância com escopo mínimo; validação de trust policies; uso de SSM para acesso sem chaves; IMDSv2 obrigatório; bloqueio de metadados via egress rules para SSRF.
- Logs e auditoria insuficientes/repudiation possível
  • Solução: CloudTrail org-wide em todas as regiões e contas, logs imutáveis (S3 + Object Lock + KMS), logs do ALB/WAF/CloudFront centralizados, retenção e alarmes de anomalia no CloudWatch.
- Backups inexistentes ou sem testes de restauração
  • Solução: políticas no AWS Backup cobrindo RDS/EFS/EC2, criptografadas em KMS, cópias cross-region, runbooks e testes periódicos de restauração (game days).
- Risco de exfiltração por egress liberado
  • Solução: VPC endpoints privados (S3, STS, CloudWatch, ECR etc.), firewalls/SGs e NACLs de saída restritivos, egress-only internet para IPv6, inspeção com Network Firewall se necessário.
- Entrega de e-mails sem SPF/DKIM/DMARC
  • Solução: configurar SPF, DKIM e DMARC para domínios do SES; gerenciamento de bounces/complaints; reputação/limites.
- Resiliência limitada do Solr (apenas AZ C no desenho)
  • Solução: implantar cluster Solr multi-AZ (A/B/C) com réplica e autoscaling; health checks e failover.
- Observabilidade parcial
  • Solução: métricas customizadas, tracing distribuído (X-Ray/OpenTelemetry), painéis e SLOs; alarmes para saturação de RDS/EFS/CPU, erros 5xx, fila de e-mails, latência de busca.
- Supply chain/patching de AMIs e Solr
  • Solução: pipeline de AMIs endurecidas (EC2 Image Builder), patching automatizado via Systems Manager Patch Manager, varreduras de vulnerabilidade (Inspector).

Gere um Relatório de Modelagem de Ameaças, baseado na metodologia STRIDE:
- Termos de referência (escopo, premissas, stakeholders)
  • Escopo: Front door (CloudFront/WAF/Shield), ALB, EC2 (SEI/SIP), RDS, EFS, ElastiCache, Solr, SES, monitoração/segurança (CloudTrail/CloudWatch/KMS/Backup), VPC e controles de rede.
  • Fora de escopo: estações dos usuários e provedores externos não mostrados.
  • Assunções: DNS gerenciado, IAM centralizado, CI/CD existente.
  • Stakeholders: Arquitetura, Segurança, Dev/DevOps, DBAs, Redes, Operações, Dono do Produto.
- Ativos e dados sensíveis
  • Dados de processos/usuários no RDS; documentos/arquivos no EFS; índices de busca no Solr; segredos/credenciais; logs/auditoria; reputação e disponibilidade do serviço.
- Fronteiras de confiança

  • Internet ↔ CloudFront/WAF (borda)
  • Borda ↔ ALB público
  • ALB público ↔ Subnets privadas (EC2)
  • EC2 ↔ Camada de dados (RDS/EFS/ElastiCache/Solr)
  • Conta AWS ↔ Operadores (IAM/SSM)
- Principais fluxos de dados
  • HTTP(S) usuário → CloudFront/WAF → ALB → API/Aplicação
  • App → RDS/EFS/Cache/Solr
  • App → SES
  • Todos → Logs/Backup/KMS

- STRIDE por categoria (ameaças e mitigação principal)
  1) Spoofing
    • Usuário/Token/API spoofing; origin bypass ao ALB; e-mail spoofing.
    • Mitigações: OIDC/OAuth2 com MFA, TLS mutual para integrações sensíveis, WAF + origin custom header, SG do ALB restrito a IPs do CloudFront, SPF/DKIM/DMARC no SES, validação de JWT e rotação de chaves.
  2) Tampering
    • Alteração de dados em trânsito/repouso (EFS/RDS), manipulação de parâmetros, alteração de AMIs/config.
    • Mitigações: TLS ponta a ponta, KMS para at-rest, controles de integridade (hashing, assinaturas), IaC imutável com revisão (CodePipeline/CodeBuild), EFS Access Points e permissões POSIX, transações e constraints no RDS.
  3) Repudiation
    • Falta de trilhas auditáveis em chamadas críticas; logs não imutáveis.
    • Mitigações: CloudTrail all-regions + S3 Object Lock, logs do WAF/ALB/CF, auditoria no app com correlação (trace-id), clock sync (NTP), retenção e proteção contra exclusão.
  4) Information Disclosure
    • Vazamento via logs, dumps, SGs abertas, memcached sem auth, snapshots não criptografados.
    • Mitigações: mascaramento de PII em logs, SG/NACL least privilege, criptografia em trânsito/repouso, snapshots criptografados, VPC endpoints privados, Secrets Manager para segredos.
  5) Denial of Service
    • DDoS, exaustão de conexões no ALB, tempestades de busca no Solr, saturação do RDS/EFS.
    • Mitigações: Shield Advanced, WAF rate-limit/bot control, limites por cliente, caching, auto scaling com reservas, read replicas (se aplicável), query quotas no Solr, capacity reservations/GP3 throughput, RDS Proxy.
  6) Elevation of Privilege
    • IAM mal configurado, SSRF → IMDS, Solr/OS RCE, privilégios locais elevados.
    • Mitigações: princípio do menor privilégio, IMDSv2 obrigatório e bloqueio de metadados via egress, hardening do SO (CIS), SSM Session Manager sem chaves, segmentação de papéis IAM, varredura contínua (Access Analyzer/Inspector).

- Risco resumido (exemplos)
  • Alto: origin bypass, memcached exposto, Solr sem auth/TLS.
  • Médio: logs não imutáveis, limites de egress, backups sem testes.
  • Baixo: ciphers legados, falta de HSTS.
  • Plano: endereçar riscos altos em 30 dias; médios em 60–90; baixos no ciclo regular.

- Ações priorizadas
  • Bloquear acesso direto ao ALB e exigir cabeçalho de origem do CloudFront.
  • Endurecer memcached (SGs estritos) ou migrar para Redis com TLS/AUTH.
  • Colocar Solr atrás de SG privado + auth/TLS + replicação multi-AZ.
  • Habilitar CloudTrail all-regions com S3 Object Lock e KMS; revisar retenções.
  • Implementar VPC endpoints e políticas de egress.
  • Formalizar backups com testes de restauração e cópias cross-region.
  • Implementar IMDSv2-only e SSM para acesso sem chaves.
  • WAF managed rules + rate limit e bot control.

Additional resourcing needs:
- Network Administrator: validações de VPC, subnets, rotas, endpoints e políticas de egress/ingress.
- Security Officer: revisão de padrões de criptografia, WAF/Shield, gestão de chaves (KMS) e compliance.
- Cloud/DevOps: IaC (CloudFormation/Terraform), CI/CD, auto scaling, observabilidade e automações de segurança.
- Software Developer: autenticação/autorização (OIDC/JWT), hardening do app, validação de entradas, integração com SES/Solr.
- Hardware Developer: não aplicável diretamente; se houver HSM on-prem ou appliances, envolver para proteção de credenciais.
- DBA: configuração RDS (TLS, parâmetros, tuning, failover, backups).

- SecOps/Blue Team: casos de uso de detecção, runbooks, testes de resposta a incidentes.
- Product Owner/Operações: definição de SLO/SLI, janelas de manutenção e requisitos de disponibilidade.

Observações finais:
- Recomenda-se formalizar o modelo em UML/DFD, identificar claramente as fronteiras de confiança e manter o relatório STRIDE vivo, revisando-o a cada mudança material do ambiente.

Sugestões de melhorias

A seguir, entrego:
1) Diagrama mermaid da arquitetura com correções de segurança aplicadas.
2) Relatório de Modelagem de Ameaças (STRIDE) consolidado e pragmático.
3) Script Terraform de referência (único arquivo) para provisionar a solução principal na AWS, incluindo os controles solicitados. Observação: é um baseline opinativo e seguro por padrão; revise tamanhos/valores/custos e adeque às suas políticas internas.

Diagrama mermaid
```mermaid
flowchart TB
  usr[Usuários Globais] --> CF[Amazon CloudFront\n+ AWS WAF (CLOUDFRONT)\n+ AWS Shield]
  subgraph Edge/Borda
    CF -->|TLS 1.2+ / HSTS / Cipher forte| ALB[ALB Público (Multi-AZ)\nSG: Só Prefix List do CloudFront\nListener: exige X-Origin-Secret\n+ AWS WAF (REGIONAL)]
  end

  subgraph VPC sa-east-1 (São Paulo)
    direction TB
    subgraph AZ A
      ALB === PUBA[(Subnet Pública A)]
      APPA[(ASG EC2 SEI/SIP A)]:::app
      EFSMA[(EFS Mount Target A)]
      RDSA[(RDS Primary A)]
      SolrA[(ASG Solr A)]:::solr
      PRIVA[(Subnet Privada A)]
    end
    subgraph AZ B
      ALB === PUBB[(Subnet Pública B)]
      APPB[(ASG EC2 SEI/SIP B)]:::app
      EFSMB[(EFS Mount Target B)]
      RDSB[(RDS Standby B)]
      SolrB[(ASG Solr B)]:::solr
      PRIVB[(Subnet Privada B)]
    end
    subgraph AZ C
      ALB === PUBC[(Subnet Pública C)]
      EFSMC[(EFS Mount Target C)]
      SolrC[(ASG Solr C)]:::solr
      PRIVC[(Subnet Privada C)]
    end

    APPA & APPB -->|HTTP 8080| TG[(Target Group App)]
    ALB -->|Encaminha se header X-Origin-Secret ok| TG

    APPA & APPB -->|NFS/TLS 2049| EFS[(Amazon EFS\nKMS at-rest + TLS in-transit\nAccess Points/UID/GID mínimos)]
    APPA & APPB -->|TLS/5432| RDS[(Amazon RDS Multi-AZ\nPrivado, KMS, IAM Auth, TLS, credenciais no Secrets Manager)]
    APPA & APPB -->|11211| CACHE[(ElastiCache Memcached\nPrivado; SG só do App)]
    APPA & APPB -->|TLS/8983| NLB_SOLR[(NLB Interno 8983)]
    NLB_SOLR --> SolrA & SolrB & SolrC

    APPA & APPB --> SES[(Amazon SES\nSPF/DKIM/DMARC)]
    VPCe[(VPC Endpoints:\nS3, STS, KMS, SSM/EC2Msgs/SSMM, Logs,\nECR, Secrets Manager, etc.)]
    APPA & APPB --> VPCe
```

```
  end

  subgraph Observabilidade/Governança
    CWatch[Amazon CloudWatch\nMétricas/Alarmes/Logs]:::sec
    CTrail[AWS CloudTrail (all-regions)\nS3 com Object Lock + KMS]:::sec
    Backup[AWS Backup\nCópias cross-region]:::sec
    KMS[AWS KMS\nChaves gerenciadas]:::sec
  end

  ALB -.-> CWatch
  CF -.-> CWatch
  APPA & APPB -.-> CWatch
  CTrail -.-> KMS
  EFS -.-> Backup
  RDS -.-> Backup
  APPA & APPB -.-> Backup
  EFS & RDS & EBS[(EBS)] -.-> KMS

  classDef app fill:#e8f5e9,stroke:#2e7d32,color:#1b5e20
  classDef solr fill:#e3f2fd,stroke:#1565c0,color:#0d47a1
  classDef sec fill:#fff8e1,stroke:#f9a825,color:#6d4c41

  %% Controles/mitigações-chave:
  note over CF,ALB: WAF Managed Rules (OWASP + Bot Control + Rate Limit)\nTLS 1.2+, HSTS,
ACM, rotação auto
  note right of ALB: SG do ALB restringido à AWS Managed Prefix
List\ncom.amazonaws.global.cloudfront.origin-facing\nListener 443 com regra: exige header
secreto\nDefault: 403
  note right of CACHE: SG só do App; considerar migração para Redis c/ TLS/AUTH se necessário
  note right of EFS: Montagem com TLS; POSIX mínimo; Access Points; Lifecycle
  note right of RDS: Subnets privadas; IAM Auth; TLS obrigatório; Secrets Manager; rotação
  note right of NLB_SOLR: Solr privado; auth/TLS; hardening; multi-AZ
  note right of VPCe: Minimiza egress; bloquear saída não usada; avaliar AWS Network Firewall
  note right of CTrail: Logs imutáveis (Object Lock);\nS3/KMS; retenções; alarmes
  note right of APPA: IMDSv2-only; SSM Session Manager; AMIs endurecidas\nInspector/SSM Patch;
SGs mínimos
```

Relatório de Modelagem de Ameaças (STRIDE) — resumido e acionável
- Termos de referência
  - Escopo: CloudFront/WAF/Shield, ALB, EC2 (SEI/SIP), RDS, EFS, ElastiCache, Solr, SES, CloudTrail/CloudWatch/KMS/Backup, VPC/SG/NACL/VPC endpoints.
  - Fora de escopo: estações dos usuários, provedores externos não exibidos.
  - Premissas: DNS gerenciado; IAM centralizado; CI/CD existente.
  - Stakeholders: Arquitetura, Segurança, Dev/DevOps, DBAs, Redes, Operações, Dono do Produto, SecOps/Blue Team.
- Ativos/dados sensíveis: dados de processos e usuários (RDS), documentos/arquivos (EFS), índices (Solr), segredos/credenciais, logs/auditoria, disponibilidade e reputação.
- Fronteiras de confiança: Internet↔CloudFront/WAF; Borda↔ALB; ALB↔Subnets privadas; EC2↔RDS/EFS/Cache/Solr; Conta AWS↔Operadores (IAM/SSM).
- Fluxos principais: HTTP(S) usuários→CloudFront/WAF→ALB→App; App→RDS/EFS/Cache/Solr; App→SES; Todos→Logs/Backup/KMS.
- STRIDE por categoria (ameaças→mitigações principais)
  1) Spoofing: origin bypass, spoofing de email/tokens → SG do ALB só para prefix list CloudFront + header secreto obrigatório; WAF; SPF/DKIM/DMARC; validação de JWT / rotação de chaves; MFA/OIDC; TLS mútuo onde necessário.
  2) Tampering: alteração de dados em trânsito/repouso → TLS ponta a ponta; KMS; IaC imutável e revisão; EFS Access Points/UID/GID; constraints no RDS; assinatura/integridade.
  3) Repudiation: logs insuficientes → CloudTrail all-regions em S3 com Object Lock + KMS; logs CF/ALB/WAF centralizados; correlação de IDs; NTP; retenções/alarmes.
  4) Information Disclosure: vazamentos por logs/SGs/snapshots → mascaramento de PII; SG/NACL least privilege; criptografia em trânsito/repouso; snapshots cifrados; Secrets Manager; VPC endpoints e políticas.
  5) DoS: DDoS, exaustão de conexões, tempestades de busca → Shield (preferível Advanced), WAF rate-limit/bot control, autoscaling, quotas por cliente, tuning Solr, RDS Proxy (quando aplicável), capacity reservations.
  6) Elevation of Privilege: IAM excessivo, SSRF→IMDS, RCE → menor privilégio; IMDSv2-only e egress filters; hardening (CIS); SSM Session Manager; segmentação de papéis; varredura contínua (Access Analyzer/Inspector).

- Risco resumido
  - Alto: origin bypass; memcached exposto; Solr sem auth/TLS.
  - Médio: logs não imutáveis; egress aberto; backups sem testes.
  - Baixo: ciphers legados; falta de HSTS.
  - Plano: altos em 30 dias; médios em 60–90; baixos em ciclo regular.
- Ações priorizadas
  - Bloquear acesso direto ao ALB (SG + regra de header); WAF gerenciado + rate limit e bot control.
  - Endurecer memcached (SGs estritos) ou migrar p/ Redis com TLS/AUTH.
  - Solr privado, auth/TLS, cluster multi-AZ, health checks.
  - CloudTrail all-regions com S3 Object Lock + KMS; revisar retenções/alarmes.
  - VPC endpoints + políticas de egress; avaliar Network Firewall.
  - Backups com testes de restauração e cópias cross-region.
  - IMDSv2-only; SSM para acesso sem chaves; patching automatizado.
- Recursos adicionais: Network Admin, Security Officer, Cloud/DevOps, Developers, DBA, SecOps, Product Owner/Operações.

Terraform — baseline seguro e opinativo (main.tf)
Observações:
- Região padrão: sa-east-1 (São Paulo). CloudFront/ACM (us-east-1) via provider alias.
- Variáveis com valores padrão seguros; ajuste tamanhos/quantidades/engine do RDS; custos podem ser significativos (NAT por AZ, Multi-AZ etc.).
- Inclui: VPC (3 AZs), subnets públicas/privadas, NAT por AZ, ALB com SG restrito a CloudFront + header secreto obrigatório, ASG App (AZ A/B), EFS, RDS Multi-AZ, ElastiCache memcached, Solr atrás de NLB interno (multi-AZ), WAF v2 (CloudFront/Regional) com managed rules + rate limit, CloudFront com ACM e logs, CloudTrail (S3 + Object Lock + KMS), AWS Backup, VPC Endpoints, IAM/SSM/IMDSv2, SES (identidade + DKIM, saída de registros). Shield Advanced opcional.
- Para simplificar, certs ACM e domínio CloudFront/SES são variáveis/saídas (DNS externo deve ser ajustado).

```hcl
terraform {
  required_version = ">= 1.6.0"
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = ">= 5.60"
    }
    random = {
      source  = "hashicorp/random"
      version = ">= 3.5"
    }
  }
}

# Providers
provider "aws" {
  region = var.region
}

# CloudFront/ACM devem estar em us-east-1
provider "aws" {
  alias  = "use1"
  region = "us-east-1"
}

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

variable "env_name" {
  type        = string
  description = "Nome do ambiente (tags/nomes)"
  default     = "prod"
}
```

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

variable "az_suffixes" {
  type        = list(string)
  default     = ["a", "b", "c"]
}

variable "alb_cert_arn" {
  type        = string
  description = "ARN do certificado ACM para o ALB (na região primária)."
}

variable "cloudfront_acm_arn_use1" {
  type        = string
  description = "ARN do ACM em us-east-1 para CloudFront."
}

variable "cloudfront_domain_aliases" {
  type        = list(string)
  default     = []
  description = "Aliases (CNAMEs) do CloudFront, ex: [\"www.seu-dominio.com\"]"
}

variable "origin_secret" {
  type        = string
  description = "Segredo para header X-Origin-Secret entre CloudFront e ALB."
  sensitive   = true
}

variable "enable_shield_advanced" {
  type        = bool
  default     = false
}

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

variable "instance_type_solr" {
  type    = string
  default = "t3.large"
}

variable "app_desired_capacity" {
  type    = number
  default = 2
}

variable "solr_desired_capacity" {
  type    = number
  default = 3
}

variable "db_engine" {
  type    = string
  default = "postgres"
}

variable "db_engine_version" {
  type    = string
  default = "15"
}

variable "db_instance_class" {
```

```
  type    = string
  default = "db.t4g.medium"
}

variable "ses_domain" {
  type        = string
  description = "Domínio para SES (opcional)"
  default     = ""
}

locals {
  name         = "${var.env_name}-sei"
  azs          = [for s in var.az_suffixes : "${var.region}${s}"]
  tags_common   = { Environment = var.env_name, System = "SEI-SIP", ManagedBy = "Terraform" }
  app_port      = 8080
  solr_port     = 8983
  rds_port      = var.db_engine == "postgres" ? 5432 : 3306
  memcached_port = 11211
}

######################
# KMS (CMKs)        #
######################
resource "aws_kms_key" "main" {
  description             = "CMK para dados (RDS/EFS/EBS/Logs/Backup)"
  enable_key_rotation     = true
  deletion_window_in_days = 30
  tags                    = local.tags_common
}

resource "aws_kms_alias" "main" {
  name          = "alias/${local.name}-cmk"
  target_key_id = aws_kms_key.main.key_id
}

######################
# VPC e Network     #
######################
resource "aws_vpc" "main" {
  cidr_block           = var.vpc_cidr
  enable_dns_hostnames = true
  enable_dns_support   = true
  tags = merge(local.tags_common, { Name = "${local.name}-vpc" })
}

resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.main.id
  tags   = merge(local.tags_common, { Name = "${local.name}-igw" })
}

# Subnets públicas e privadas (3 AZs)
resource "aws_subnet" "public" {
  for_each = { for idx, az in local.azs : az => az }
  vpc_id                  = aws_vpc.main.id
  cidr_block              = cidrsubnet(var.vpc_cidr, 8, index(local.azs, each.key))
  availability_zone       = each.key
  map_public_ip_on_launch = true
  tags = merge(local.tags_common, {
    Name = "${local.name}-public-${substr(each.key, -1, 1)}"
    Tier = "public"
  })
}

resource "aws_subnet" "private" {
  for_each = { for idx, az in local.azs : az => az }
  vpc_id            = aws_vpc.main.id
  cidr_block        = cidrsubnet(var.vpc_cidr, 8, 100 + index(local.azs, each.key))
  availability_zone = each.key
  tags = merge(local.tags_common, {
```

```
    Name = "${local.name}-private-${substr(each.key, -1, 1)}"
    Tier = "private"
  })
}

# EIPs e NATs (um por AZ)
resource "aws_eip" "nat" {
  for_each = aws_subnet.public
  domain   = "vpc"
  tags     = merge(local.tags_common, { Name = "${local.name}-nat-eip-${substr(each.key, -1, 1)}" })
}

resource "aws_nat_gateway" "nat" {
  for_each      = aws_subnet.public
  subnet_id     = each.value.id
  allocation_id = aws_eip.nat[each.key].id
  tags          = merge(local.tags_common, { Name = "${local.name}-nat-${substr(each.key, -1, 1)}" })
  depends_on    = [aws_internet_gateway.igw]
}

# Route tables
resource "aws_route_table" "public" {
  vpc_id = aws_vpc.main.id
  tags   = merge(local.tags_common, { Name = "${local.name}-rtb-public" })
}

resource "aws_route" "public_default" {
  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
}

resource "aws_route_table" "private" {
  for_each = aws_nat_gateway.nat
  vpc_id   = aws_vpc.main.id
  tags     = merge(local.tags_common, { Name = "${local.name}-rtb-private-${substr(each.key, -1, 1)}"
})
}

resource "aws_route" "private_default" {
  for_each               = aws_route_table.private
  route_table_id         = each.value.id
  destination_cidr_block = "0.0.0.0/0"
  nat_gateway_id         = aws_nat_gateway.nat[each.key].id
}

resource "aws_route_table_association" "private_assoc" {
  for_each       = aws_subnet.private
  subnet_id      = each.value.id
  route_table_id = aws_route_table.private[each.key].id
}

############################
# VPC Endpoints (privado)#
############################
# Gateway endpoint para S3
resource "aws_vpc_endpoint" "s3" {
  vpc_id            = aws_vpc.main.id
  service_name      = "com.amazonaws.${var.region}.s3"
  vpc_endpoint_type = "Gateway"
  route_table_ids   = [for k, v in aws_route_table.private : v.id]
  tags              = merge(local.tags_common, { Name = "${local.name}-vpce-s3" })
}
```

```
# Interface endpoints (principais)
locals {
  interface_services = [
    "ssm", "ec2", "ec2messages", "ssmmessages",
    "logs", "secretsmanager", "kms", "sts", "ecr.api", "ecr.dkr", "monitoring"
  ]
}
resource "aws_security_group" "vpce" {
  name        = "${local.name}-vpce-sg"
  description = "SG para VPC Endpoints (interface)"
  vpc_id      = aws_vpc.main.id
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = local.tags_common
}

resource "aws_vpc_endpoint" "interface" {
  for_each            = toset(local.interface_services)
  vpc_id              = aws_vpc.main.id
  service_name        = "com.amazonaws.${var.region}.${each.key}"
  vpc_endpoint_type   = "Interface"
  private_dns_enabled = true
  security_group_ids  = [aws_security_group.vpce.id]
  subnet_ids          = [for k, v in aws_subnet.private : v.id]
  tags                = merge(local.tags_common, { Name = "${local.name}-vpce-${each.key}" })
}

############################
# Security Groups        #
############################
# Prefix list gerenciada do CloudFront (origns → ALB)
data "aws_prefix_list" "cloudfront" {
  name = "com.amazonaws.global.cloudfront.origin-facing"
}

resource "aws_security_group" "alb" {
  name        = "${local.name}-alb-sg"
  description = "ALB - aceita apenas CloudFront origin-facing"
  vpc_id      = aws_vpc.main.id

  ingress {
    description      = "HTTPS de CloudFront"
    from_port        = 443
    to_port          = 443
    protocol         = "tcp"
    prefix_list_ids  = [data.aws_prefix_list.cloudfront.id]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = local.tags_common
}

resource "aws_security_group" "app" {
  name        = "${local.name}-app-sg"
  description = "Instâncias App SEI/SIP"
  vpc_id      = aws_vpc.main.id

  ingress {
```

```
    description     = "Tráfego do ALB para App"
    from_port       = local.app_port
    to_port         = local.app_port
    protocol        = "tcp"
    security_groups = [aws_security_group.alb.id]
  }

  # Egress: somente necessário (refine conforme política)
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = local.tags_common
}

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

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

  egress { from_port = 0 to_port = 0 protocol = "-1" cidr_blocks = ["0.0.0.0/0"] }
  tags = local.tags_common
}

resource "aws_security_group" "rds" {
  name   = "${local.name}-rds-sg"
  vpc_id = aws_vpc.main.id

  ingress {
    from_port       = local.rds_port
    to_port         = local.rds_port
    protocol        = "tcp"
    security_groups = [aws_security_group.app.id]
    description     = "DB do App"
  }

  egress { from_port = 0 to_port = 0 protocol = "-1" cidr_blocks = ["0.0.0.0/0"] }
  tags = local.tags_common
}

resource "aws_security_group" "cache" {
  name   = "${local.name}-cache-sg"
  vpc_id = aws_vpc.main.id

  ingress {
    from_port       = local.memcached_port
    to_port         = local.memcached_port
    protocol        = "tcp"
    security_groups = [aws_security_group.app.id]
    description     = "Cache do App"
  }

  egress { from_port = 0 to_port = 0 protocol = "-1" cidr_blocks = ["0.0.0.0/0"] }
  tags = local.tags_common
}

resource "aws_security_group" "solr_nlb" {
  name   = "${local.name}-solr-nlb-sg"
  vpc_id = aws_vpc.main.id
```

```
  ingress {
    from_port       = local.solr_port
    to_port         = local.solr_port
    protocol        = "tcp"
    security_groups = [aws_security_group.app.id]
    description     = "Tráfego App para NLB Solr"
  }

  egress { from_port = 0 to_port = 0 protocol = "-1" cidr_blocks = ["0.0.0.0/0"] }
  tags = local.tags_common
}

resource "aws_security_group" "solr_nodes" {
  name   = "${local.name}-solr-nodes-sg"
  vpc_id = aws_vpc.main.id

  ingress {
    from_port       = local.solr_port
    to_port         = local.solr_port
    protocol        = "tcp"
    security_groups = [aws_security_group.solr_nlb.id]
    description     = "NLB → Solr"
  }

  egress { from_port = 0 to_port = 0 protocol = "-1" cidr_blocks = ["0.0.0.0/0"] }
  tags = local.tags_common
}

############################
# IAM roles para EC2       #
############################
data "aws_iam_policy" "ssm_core" {
  arn = "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
}

data "aws_iam_policy" "cw_agent" {
  arn = "arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy"
}

resource "aws_iam_role" "ec2_app" {
  name               = "${local.name}-ec2-app-role"
  assume_role_policy = data.aws_iam_policy_document.ec2_trust.json
  tags               = local.tags_common
}
resource "aws_iam_role" "ec2_solr" {
  name               = "${local.name}-ec2-solr-role"
  assume_role_policy = data.aws_iam_policy_document.ec2_trust.json
  tags               = local.tags_common
}

data "aws_iam_policy_document" "ec2_trust" {
  statement {
    actions = ["sts:AssumeRole"]
    principals { type = "Service" identifiers = ["ec2.amazonaws.com"] }
  }
}

# Política mínima para ler segredo do DB e usar KMS/Logs
resource "aws_iam_policy" "app_inline" {
  name        = "${local.name}-app-inline"
  description = "Permissões mínimas para App (logs, secrets, kms decrypt)"
  policy = jsonencode({
    Version = "2012-10-17",
    Statement = [
      { Effect="Allow",
Action=["logs:CreateLogGroup","logs:CreateLogStream","logs:PutLogEvents"], Resource="*" },
      { Effect="Allow", Action=["secretsmanager:GetSecretValue","secretsmanager:DescribeSecret"],
Resource=[aws_secretsmanager_secret.db.arn] },
      { Effect="Allow", Action=["kms:Decrypt"], Resource=[aws_kms_key.main.arn] }
```

```
    ]
  })
}

resource "aws_iam_role_policy_attachment" "app_ssm" {
  role       = aws_iam_role.ec2_app.name
  policy_arn = data.aws_iam_policy.ssm_core.arn
}
resource "aws_iam_role_policy_attachment" "app_cw" {
  role       = aws_iam_role.ec2_app.name
  policy_arn = data.aws_iam_policy.cw_agent.arn
}
resource "aws_iam_role_policy_attachment" "app_inline_attach" {
  role       = aws_iam_role.ec2_app.name
  policy_arn = aws_iam_policy.app_inline.arn
}

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

# Solr: logs/ssm apenas
resource "aws_iam_role_policy_attachment" "solr_ssm" {
  role       = aws_iam_role.ec2_solr.name
  policy_arn = data.aws_iam_policy.ssm_core.arn
}
resource "aws_iam_role_policy_attachment" "solr_cw" {
  role       = aws_iam_role.ec2_solr.name
  policy_arn = data.aws_iam_policy.cw_agent.arn
}
resource "aws_iam_instance_profile" "ec2_solr" {
  name = "${local.name}-solr-profile"
  role = aws_iam_role.ec2_solr.name
}

############################
# EFS                      #
############################
resource "aws_efs_file_system" "main" {
  encrypted        = true
  kms_key_id       = aws_kms_key.main.arn
  performance_mode  = "generalPurpose"
  throughput_mode   = "bursting"
  tags             = merge(local.tags_common, { Name = "${local.name}-efs" })
}

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

############################
# RDS (Multi-AZ)           #
############################
resource "random_password" "db" {
  length  = 32
  special = true
}

resource "aws_secretsmanager_secret" "db" {
  name = "${local.name}/rds/master"
  kms_key_id = aws_kms_key.main.id
  tags = local.tags_common
}

resource "aws_secretsmanager_secret_version" "db" {
```

```
  secret_id     = aws_secretsmanager_secret.db.id
  secret_string = jsonencode({ username = "dbmaster", password = random_password.db.result })
}

resource "aws_db_subnet_group" "db" {
  name       = "${local.name}-db-subnets"
  subnet_ids = [aws_subnet.private[local.azs[0]].id, aws_subnet.private[local.azs[1]].id]
  tags       = local.tags_common
}

resource "aws_db_parameter_group" "db" {
  name   = "${local.name}-${var.db_engine}-params"
  family = var.db_engine == "postgres" ? "postgres${var.db_engine_version}" :
"mysql${var.db_engine_version}"
  dynamic "parameter" {
    for_each = var.db_engine == "postgres" ? [1] : []
    content {
      name  = "rds.force_ssl"
      value = "1"
    }
  }
  tags = local.tags_common
}

resource "aws_db_instance" "db" {
  identifier                 = "${local.name}-db"
  engine                     = var.db_engine
  engine_version             = var.db_engine_version
  instance_class             = var.db_instance_class
  db_subnet_group_name       = aws_db_subnet_group.db.name
  vpc_security_group_ids     = [aws_security_group.rds.id]
  multi_az                   = true
  allocated_storage          = 100
  storage_type               = "gp3"
  storage_encrypted          = true
  kms_key_id                 = aws_kms_key.main.arn
  username                   = jsondecode(aws_secretsmanager_secret_version.db.secret_string).username
  password                   = jsondecode(aws_secretsmanager_secret_version.db.secret_string).password
  port                       = local.rds_port
  backup_retention_period    = 7
  deletion_protection        = true
  skip_final_snapshot        = false
  apply_immediately          = false
  publicly_accessible        = false
  iam_database_authentication_enabled = var.db_engine == "postgres" ? true : false
  parameter_group_name       = aws_db_parameter_group.db.name
  auto_minor_version_upgrade = true
  monitoring_interval        = 60
  tags                       = local.tags_common
}

############################
# ElastiCache Memcached    #
############################
resource "aws_elasticache_subnet_group" "cache" {
  name       = "${local.name}-cache-subnets"
  subnet_ids = [for k, v in aws_subnet.private : v.id]
}

resource "aws_elasticache_cluster" "memcached" {
  cluster_id           = "${local.name}-memd"
  engine               = "memcached"
  node_type            = "cache.t3.small"
  num_cache_nodes      = 3
  az_mode              = "cross-az"
  preferred_availability_zones = local.azs
  port                 = local.memcached_port
  subnet_group_name    = aws_elasticache_subnet_group.cache.name
  security_group_ids   = [aws_security_group.cache.id]
```

```
  tags              = local.tags_common
}

############################
# CloudWatch Log Group   #
############################
resource "aws_cloudwatch_log_group" "app" {
  name              = "/${local.name}/app"
  retention_in_days = 30
  kms_key_id        = aws_kms_key.main.arn
  tags              = local.tags_common
}

############################
# Launch Templates (EC2) #
############################
data "aws_ami" "al2023" {
  most_recent = true
  owners      = ["amazon"]
  filter {
    name   = "name"
    values = ["al2023-ami-*-x86_64"]
  }
}

# App
resource "aws_launch_template" "app" {
  name_prefix   = "${local.name}-app-"
  image_id      = data.aws_ami.al2023.id
  instance_type = var.instance_type_app
  iam_instance_profile { name = aws_iam_instance_profile.ec2_app.name }
  vpc_security_group_ids = [aws_security_group.app.id]
  block_device_mappings {
    device_name = "/dev/xvda"
    ebs {
      volume_size           = 30
      volume_type           = "gp3"
      encrypted             = true
      kms_key_id            = aws_kms_key.main.arn
      delete_on_termination = true
    }
  }
  metadata_options { http_tokens = "required" } # IMDSv2-only
  user_data = base64encode(<<-EOT
    #!/bin/bash
    set -e
    yum -y update
    yum -y install amazon-efs-utils
    mkdir -p /mnt/efs
    echo "${aws_efs_file_system.main.id}:/ /mnt/efs efs _netdev,tls 0 0" >> /etc/fstab
    mount -a -t efs defaults
    # Placeholder: deploy app SEI/SIP; habilitar CloudWatch Agent; reforçar SO; usar SSM
  EOT
  )
  tag_specifications {
    resource_type = "instance"
    tags          = merge(local.tags_common, { Name = "${local.name}-app" })
  }
}

# Solr
resource "aws_launch_template" "solr" {
  name_prefix   = "${local.name}-solr-"
  image_id      = data.aws_ami.al2023.id
  instance_type = var.instance_type_solr
  iam_instance_profile { name = aws_iam_instance_profile.ec2_solr.name }
  vpc_security_group_ids = [aws_security_group.solr_nodes.id]
  block_device_mappings {
    device_name = "/dev/xvda"
```

```
    ebs {
      volume_size           = 50
      volume_type           = "gp3"
      encrypted             = true
      kms_key_id            = aws_kms_key.main.arn
      delete_on_termination = true
    }
  }
  metadata_options { http_tokens = "required" }
  user_data = base64encode(<<-EOT
    #!/bin/bash
    set -e
    # Placeholder: instalar/Configurar Solr em TLS+auth; hardening; CW Agent; SSM
  EOT
  )
  tag_specifications { resource_type = "instance" tags = merge(local.tags_common, { Name =
"${local.name}-solr" }) }
}

############################
# ALB + Target Group       #
############################
resource "aws_lb" "alb" {
  name               = substr("${local.name}-alb", 0, 32)
  load_balancer_type = "application"
  security_groups    = [aws_security_group.alb.id]
  subnets            = [for k, v in aws_subnet.public : v.id]
  idle_timeout       = 60
  drop_invalid_header_fields = true
  enable_http2       = true
  tags               = local.tags_common
}

resource "aws_lb_target_group" "app" {
  name        = substr("${local.name}-tg", 0, 32)
  port        = local.app_port
  protocol    = "HTTP"
  vpc_id      = aws_vpc.main.id
  health_check {
    protocol            = "HTTP"
    path                = "/health"
    matcher             = "200-399"
    healthy_threshold   = 2
    unhealthy_threshold = 3
    interval            = 15
    timeout             = 5
  }
  tags = local.tags_common
}

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

  # Default nega (403) — só passa se header secreto casar
  default_action {
    type = "fixed-response"
    fixed_response {
      content_type = "text/plain"
      message_body = "Forbidden"
      status_code  = "403"
    }
  }
}

# Regra: exige header X-Origin-Secret para encaminhar ao TG
```

```
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-Secret"
      values           = [var.origin_secret]
    }
  }
}

############################
# Auto Scaling Groups    #
############################
resource "aws_autoscaling_group" "app" {
  name                      = "${local.name}-app-asg"
  desired_capacity          = var.app_desired_capacity
  min_size                  = 2
  max_size                  = 6
  vpc_zone_identifier       = [aws_subnet.private[local.azs[0]].id, aws_subnet.private[local.azs[1]].id]
  health_check_type         = "ELB"
  health_check_grace_period = 60
  target_group_arns         = [aws_lb_target_group.app.arn]
  launch_template {
    id      = aws_launch_template.app.id
    version = "$Latest"
  }
  tag { key = "Name" value = "${local.name}-app" propagate_at_launch = true }
  lifecycle { create_before_destroy = true }
}

# Solr atrás de NLB interno
resource "aws_lb" "solr" {
  name               = substr("${local.name}-solr-nlb", 0, 32)
  load_balancer_type = "network"
  internal           = true
  subnets            = [for k, v in aws_subnet.private : v.id]
  security_groups    = [aws_security_group.solr_nlb.id]
  tags               = local.tags_common
}

resource "aws_lb_target_group" "solr" {
  name     = substr("${local.name}-solr-tg", 0, 32)
  port     = local.solr_port
  protocol = "TCP"
  vpc_id   = aws_vpc.main.id
  health_check {
    protocol = "TCP"
  }
  tags = local.tags_common
}

resource "aws_lb_listener" "solr" {
  load_balancer_arn = aws_lb.solr.arn
  port              = local.solr_port
  protocol          = "TCP"
  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.solr.arn
  }
}

resource "aws_autoscaling_group" "solr" {
  name                      = "${local.name}-solr-asg"
  desired_capacity          = var.solr_desired_capacity
```

```
  min_size                  = 3
  max_size                  = 9
  vpc_zone_identifier       = [for k, v in aws_subnet.private : v.id]
  health_check_type         = "EC2"
  health_check_grace_period = 120
  launch_template {
    id      = aws_launch_template.solr.id
    version = "$Latest"
  }
  tag { key = "Name" value = "${local.name}-solr" propagate_at_launch = true }
  lifecycle { create_before_destroy = true }
}

resource "aws_autoscaling_attachment" "solr_tg" {
  autoscaling_group_name = aws_autoscaling_group.solr.name
  lb_target_group_arn    = aws_lb_target_group.solr.arn
}

############################
# WAFv2 (Regional + CF)  #
############################
# Regional (ALB)
resource "aws_wafv2_web_acl" "regional" {
  name        = "${local.name}-waf-regional"
  description = "WAF para ALB"
  scope       = "REGIONAL"
  default_action { block {} }
  visibility_config {
    cloudwatch_metrics_enabled = true
    metric_name                = "${local.name}-waf-regional"
    sampled_requests_enabled   = true
  }
  rule {
    name     = "AWS-AWSManagedRulesCommonRuleSet"
    priority = 1
    override_action { none {} }
    statement { managed_rule_group_statement { name = "AWSManagedRulesCommonRuleSet"
vendor_name = "AWS" } }
    visibility_config { cloudwatch_metrics_enabled = true metric_name = "common"
sampled_requests_enabled = true }
  }
  rule {
    name     = "AWS-AWSManagedRulesKnownBadInputsRuleSet"
    priority = 2
    override_action { none {} }
    statement { managed_rule_group_statement { name =
"AWSManagedRulesKnownBadInputsRuleSet" vendor_name = "AWS" } }
    visibility_config { cloudwatch_metrics_enabled = true metric_name = "badinputs"
sampled_requests_enabled = true }
  }
  rule {
    name     = "AWS-AWSManagedRulesSQLiRuleSet"
    priority = 3
    override_action { none {} }
    statement { managed_rule_group_statement { name = "AWSManagedRulesSQLiRuleSet"
vendor_name = "AWS" } }
    visibility_config { cloudwatch_metrics_enabled = true metric_name = "sqli"
sampled_requests_enabled = true }
  }
  rule {
    name     = "RateLimit"
    priority = 10
    action { block {} }
    statement { rate_based_statement { limit = 2000 aggregate_key_type = "IP" } }
    visibility_config { cloudwatch_metrics_enabled = true metric_name = "ratelimit"
sampled_requests_enabled = true }
  }
  tags = local.tags_common
}
```

```
resource "aws_wafv2_web_acl_association" "alb_assoc" {
  resource_arn = aws_lb.alb.arn
  web_acl_arn  = aws_wafv2_web_acl.regional.arn
}

# CloudFront (usar provider us-east-1)
resource "aws_wafv2_web_acl" "cloudfront" {
  provider    = aws.use1
  name        = "${local.name}-waf-cf"
  description = "WAF para CloudFront"
  scope       = "CLOUDFRONT"
  default_action { allow {} }
  visibility_config {
    cloudwatch_metrics_enabled = true
    metric_name                = "${local.name}-waf-cf"
    sampled_requests_enabled   = true
  }
  rule {
    name     = "AWS-AWSManagedRulesCommonRuleSet"
    priority = 1
    override_action { none {} }
    statement { managed_rule_group_statement { name = "AWSManagedRulesCommonRuleSet"
vendor_name = "AWS" } }
    visibility_config { cloudwatch_metrics_enabled = true metric_name = "common"
sampled_requests_enabled = true }
  }
  rule {
    name     = "AWS-AWSManagedRulesBotControlRuleSet"
    priority = 2
    override_action { none {} }
    statement { managed_rule_group_statement { name = "AWSManagedRulesBotControlRuleSet"
vendor_name = "AWS" } }
    visibility_config { cloudwatch_metrics_enabled = true metric_name = "bot"
sampled_requests_enabled = true }
  }
  rule {
    name     = "RateLimit"
    priority = 10
    action { block {} }
    statement { rate_based_statement { limit = 2000 aggregate_key_type = "IP" } }
    visibility_config { cloudwatch_metrics_enabled = true metric_name = "ratelimit"
sampled_requests_enabled = true }
  }
  tags = local.tags_common
}

############################
# CloudFront Distribution#
############################
resource "aws_s3_bucket" "cf_logs" {
  bucket = "${local.name}-cf-logs-${data.aws_caller_identity.current.account_id}"
  force_destroy = true
  tags = local.tags_common
}
data "aws_caller_identity" "current" {}

resource "aws_cloudfront_origin_access_control" "oac" {
  name                              = "${local.name}-oac"
  origin_access_control_origin_type = "web"
  signing_behavior                  = "always"
  signing_protocol                  = "sigv4"
  description                       = "OAC para ALB"
}

resource "aws_cloudfront_distribution" "main" {
  provider = aws.use1
  enabled  = true
  aliases  = var.cloudfront_domain_aliases
```

```
  origin {
    domain_name = aws_lb.alb.dns_name
    origin_id   = "alb-origin"
    custom_origin_config {
      origin_protocol_policy = "https-only"
      https_port             = 443
      origin_ssl_protocols   = ["TLSv1.2"]
    }
    origin_custom_header {
      name  = "X-Origin-Secret"
      value = var.origin_secret
    }
    origin_access_control_id = aws_cloudfront_origin_access_control.oac.id
  }

  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","Content-Type","Accept","Accept-Language","Origin","Referer"]
      cookies { forward = "all" }
    }
  }

  restrictions { geo_restriction { restriction_type = "none" } }

  viewer_certificate {
    acm_certificate_arn      = var.cloudfront_acm_arn_use1
    ssl_support_method       = "sni-only"
    minimum_protocol_version = "TLSv1.2_2021"
  }

  logging_config {
    bucket = "${aws_s3_bucket.cf_logs.bucket}.s3.amazonaws.com"
    include_cookies = false
    prefix = "cloudfront/"
  }

  web_acl_id = aws_wafv2_web_acl.cloudfront.arn

  default_root_object = ""
  price_class         = "PriceClass_All"
  tags                = local.tags_common
}

############################
# CloudTrail + S3 Imutável
############################
resource "aws_s3_bucket" "cloudtrail" {
  bucket              = "${local.name}-trail-${data.aws_caller_identity.current.account_id}"
  object_lock_enabled = true
  force_destroy       = false
  tags                = local.tags_common
}

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

resource "aws_s3_bucket_object_lock_configuration" "trail" {
  bucket = aws_s3_bucket.cloudtrail.id
  rule {
    default_retention {
```

```
    mode  = "GOVERNANCE"
    days  = 90
   }
  }
}

resource "aws_s3_bucket_server_side_encryption_configuration" "trail" {
  bucket = aws_s3_bucket.cloudtrail.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                          = "${local.name}-trail"
  s3_bucket_name                = aws_s3_bucket.cloudtrail.bucket
  kms_key_id                    = aws_kms_key.main.arn
  is_multi_region_trail         = true
  include_global_service_events = true
  enable_log_file_validation    = true
  tags                          = local.tags_common
}

############################
# AWS Backup             #
############################
resource "aws_backup_vault" "main" {
  name        = "${local.name}-vault"
  kms_key_arn = aws_kms_key.main.arn
  tags        = local.tags_common
}

resource "aws_backup_plan" "daily" {
  name = "${local.name}-backup"
  rule {
   rule_name         = "daily"
   target_vault_name = aws_backup_vault.main.name
   schedule          = "cron(0 5 * * ? *)"
   lifecycle {
    delete_after = 30
   }
  }
  tags = local.tags_common
}

# Selecione por tag (aplique tag Backup=Yes nos recursos que quer incluir)
resource "aws_backup_selection" "bytag" {
  name         = "tagged-resources"
  iam_role_arn = data.aws_iam_role.service_role_backup.arn
  plan_id      = aws_backup_plan.daily.id

  selection_tag {
   type  = "STRINGEQUALS"
   key   = "Backup"
   value = "Yes"
  }
}
# Função gerenciada para Backup (AWSBackupDefaultServiceRole) – usar data source
data "aws_iam_role" "service_role_backup" {
  name = "AWSBackupDefaultServiceRole"
}

############################
# SES (identidade)       #
############################
resource "aws_ses_domain_identity" "this" {
```

```
  count  = length(var.ses_domain) > 0 ? 1 : 0
  domain = var.ses_domain
}

resource "aws_ses_domain_dkim" "this" {
  count  = length(var.ses_domain) > 0 ? 1 : 0
  domain = aws_ses_domain_identity.this[0].domain
}

output "ses_verification_record" {
  value = length(var.ses_domain) > 0 ? {
    name  = "_amazonses.${var.ses_domain}"
    type  = "TXT"
    value = aws_ses_domain_identity.this[0].verification_token
  } : null
}

output "ses_dkim_records" {
  value = length(var.ses_domain) > 0 ? [
    for t in aws_ses_domain_dkim.this[0].dkim_tokens :
    { name = "${t}._domainkey.${var.ses_domain}", type = "CNAME", value =
"${t}.dkim.amazonses.com" }
  ] : null
}

############################
# Shield Advanced (opt)  #
############################
resource "aws_shield_protection" "alb" {
  count        = var.enable_shield_advanced ? 1 : 0
  name         = "${local.name}-alb"
  resource_arn = aws_lb.alb.arn
}
resource "aws_shield_protection" "cf" {
  provider     = aws.use1
  count        = var.enable_shield_advanced ? 1 : 0
  name         = "${local.name}-cf"
  resource_arn = aws_cloudfront_distribution.main.arn
}

############################
# Outputs principais     #
############################
output "alb_dns_name" {
  value = aws_lb.alb.dns_name
}

output "cloudfront_domain_name" {
  value = aws_cloudfront_distribution.main.domain_name
}

output "efs_id" {
  value = aws_efs_file_system.main.id
}

output "rds_endpoint" {
  value = aws_db_instance.db.address
}

output "memcached_configuration" {
  value =
"${aws_elasticache_cluster.memcached.cache_nodes[0].address}:${aws_elasticache_cluster.memcach
ed.port}"
}

output "solr_nlb_dns" {
  value = aws_lb.solr.dns_name
}
```

Próximos passos
- Preencha variáveis sensíveis/ambiente (origin_secret, certificados ACM, domínios).
- Ajuste tamanhos (instâncias, storage, nós de cache), paths de saúde e regras WAF conforme a aplicação.
- Integre CI/CD, AMIs endurecidas, patching automatizado, e observabilidade (logs do ALB para S3, WAF logs via Firehose).
- Teste:
  - Origin bypass (ALB sem header deve retornar 403; SG não permite tráfego fora do CloudFront).
  - TLS ponta a ponta (ALB/RDS/EFS mount TLS).
  - Failover RDS Multi-AZ; disponibilidade de Solr multi-AZ.
  - Backups e restauração (game days).
- Segurança contínua: Inspector, Access Analyzer, revisão de IAM, egress restrito e, se necessário, AWS Network Firewall.