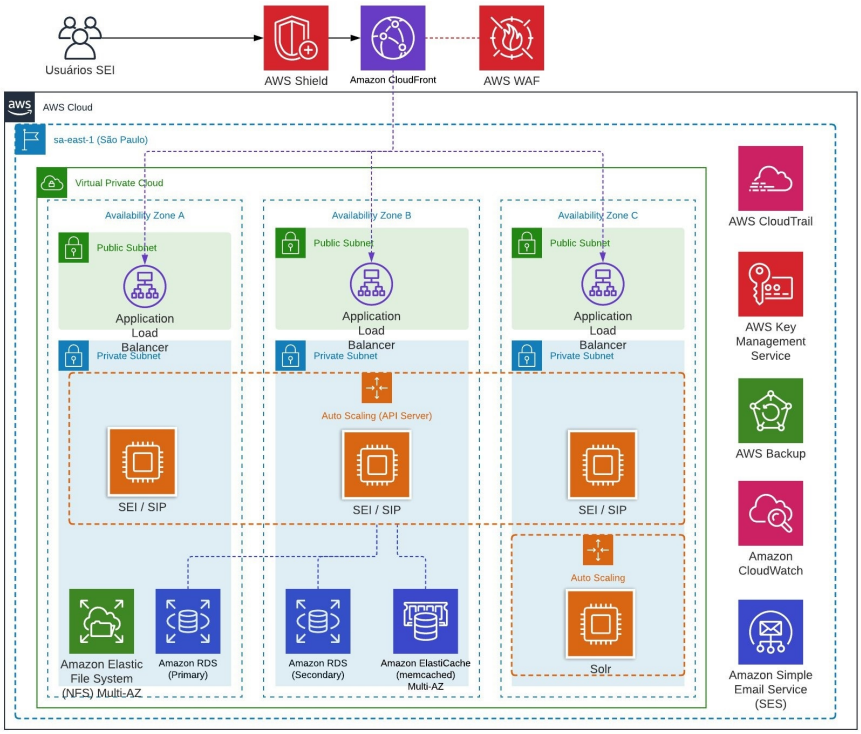
Relatório de análise da solução: tmpih617hi2



Análise completa da solução atual

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

Lista com os componentes:
- Borda/Segurança:
  - AWS Shield (DDoS L3/L4 – em CloudFront)
  - Amazon CloudFront (CDN e cache)
  - AWS WAF (firewall de aplicação)
- Rede:
  - VPC (Virtual Private Cloud) com 3 AZs: A, B e C
  - Subnets públicas (por AZ) para ALBs
  - Subnets privadas (por AZ) para instâncias/app e dados
- Camada de entrada:
  - Application Load Balancer (ALB) em cada AZ (internet-facing)
- Camada de aplicação:
  - Auto Scaling Group de servidores de API (EC2) – identificado como “SEI/SIP”
  - Auto Scaling para cluster de busca Solr (EC2) em AZ dedicada
- Camada de dados:
  - Amazon RDS Multi-AZ (Primary em AZ A, Secondary em AZ B)
  - Amazon Elastic File System (EFS) Multi-AZ (NFS)
  - Amazon ElastiCache (Memcached) Multi-AZ
- Observabilidade, segurança e operações:
  - AWS CloudTrail (auditoria)
  - Amazon CloudWatch (métricas, logs e alarmes)
  - AWS Key Management Service (KMS) para chaves de criptografia
  - AWS Backup (políticas e cofre de backup)
  - Amazon Simple Email Service (SES) para envio de e-mails

Interação entre os componentes:
- Usuários → CloudFront → WAF:
  - Requests HTTPS chegam na CDN (CloudFront), que aplica cache e encaminha apenas tráfego válido para a origem.
  - WAF em CloudFront inspeciona e bloqueia padrões maliciosos (OWASP, bots, rate-limiting).
  - AWS Shield fornece proteção DDoS (Standard).
- CloudFront → ALBs (subnets públicas):
  - CloudFront encaminha para os Application Load Balancers distribuídos nas 3 AZs.

  - ALBs terminam TLS e roteiam para as instâncias de aplicação em subnets privadas.
- ALBs → Aplicação (SEI/SIP) em Auto Scaling (subnets privadas):
  - Instâncias EC2 da aplicação recebem tráfego do ALB.
  - A aplicação consome:
    - Amazon RDS (leitura/escrita no Primary; failover para Secondary).
    - EFS para armazenamento compartilhado de documentos/artefatos.
    - ElastiCache (Memcached) para cache de sessões e dados quentes.
    - Solr (cluster em Auto Scaling) para busca/full-text.
    - SES para envio de notificações e e-mails transacionais.
- Telemetria e segurança:
  - CloudTrail registra chamadas de API.
  - CloudWatch coleta métricas/logs e dispara alarmes.
  - KMS provê chaves para criptografia at-rest (RDS, EFS, EBS, CloudWatch Logs, Backups).
  - AWS Backup orquestra cópias e retenção de snapshots/backups.

O que esse sistema faz:
- É uma arquitetura web multi-AZ de alta disponibilidade para um sistema transacional com forte componente de gestão documental e pesquisa (ex.: SEI/SIP), entregue globalmente via CDN, com proteção WAF/Shield, back-end em EC2 auto-escalável, busca em Solr, persistência relacional em RDS, arquivos compartilhados em EFS e aceleração via cache (ElastiCache). Envia e-mails via SES e possui trilhas de auditoria, monitoramento e backups gerenciados.

Vulnerabilidades e Solução para cada vulnerabilidade:
- Exposição direta dos ALBs à internet (bypass de CloudFront/WAF)
  - Solução: Restringir Security Groups dos ALBs para aceitar somente IPs de saída do CloudFront; usar header de origem assinada (custom header) validado no ALB/Aplicação; considerar AWS WAF também no ALB (se necessário).
- Ausência de TLS end-to-end ou políticas fracas
  - Solução: Enforce TLS 1.2+ em CloudFront/ALB; certificados ACM; TLS entre ALB→EC2 (NLB ou ALB com certificados internos) e montagens EFS com TLS; HSTS e security headers.
- Regras WAF insuficientes
  - Solução: Ativar Managed Rules (AWS e vendors), rate-based rules, bot control, IP reputation lists, validação de payloads, proteções específicas à aplicação (ex.: upload de arquivos).
- DDoS volumétrico e de aplicação
  - Solução: CloudFront + Shield Standard já ajudam; avaliar AWS Shield Advanced (com Response Team), WAF rate-limit, circuit breakers na app, filas/limites por cliente, budgets de custo.
- Credenciais e segredos na AMI/variáveis de ambiente
  - Solução: AWS Secrets Manager ou SSM Parameter Store (com KMS); rotação automática; remover segredos do código/AMI; escopo mínimo de IAM roles.
- RDS exposto ou com configurações fracas
  - Solução: Garantir RDS somente em subnets privadas, SGs restritivos, criptografia at-rest com KMS, IAM DB Auth (quando aplicável), TLS in-transit, parâmetros de endurecimento (log_min_duration_statement, pgaudit/Performance Insights), snapshots criptografados, rotinas de manutenção/patch.
- ElastiCache Memcached sem autenticação/criptografia
  - Solução: Memcached não tem autenticação nativa nem TLS pleno; restringir estritamente por SG e subnet privada; se precisar de criptografia e auth, migrar para Redis com TLS e AUTH (Redis AUTH + ACLs).
- EFS sem controles de acesso finos
  - Solução: EFS Access Points, POSIX permissions, SG estrito, TLS de montagem, backup habilitado, IAM auth (quando aplicável).
- Solr exposto ou desatualizado
  - Solução: Isolar em subnets privadas; SG permitindo apenas tráfego da aplicação; autenticação/admin desabilitado externamente; patch management; considerar serviço gerenciado (ex.: Amazon OpenSearch Service) se aplicável.
- Falta de proteção contra SSRF e acesso ao metadata
  - Solução: IMDSv2 obrigatório, bloqueios de egress (NACL/SG), proxies de saída controlados, validação de URLs de saída na app.
- Logs sensíveis e vazamento de PII
  - Solução: Redação/masking em logs, classificação de dados, CloudWatch Logs com KMS, acesso com ABAC/RBAC, retenções alinhadas a LGPD.
- Auditoria incompleta
  - Solução: CloudTrail multi-região e multi-conta, integridade de logs (S3 + Object Lock + Lake/centralização), alertas SIEM.
- Backups sem isolamento
  - Solução: AWS Backup com Vault Lock (imutabilidade), cofres e chaves em conta separada (cross-account), testes regulares de restauração, cópias cross-region.
- IAM permissivo
  - Solução: Princípio de menor privilégio, SCPs em Organizations, boundaries por função, rotação de

chaves, MFA, acesso Just-In-Time, análise de IAM Access Analyzer.
- Escalabilidade e exaustão de recursos (DB connections, fila de e-mail, etc.)
  - Solução: Connection pooling, limites/quota por cliente, autoscaling baseado em métricas corretas, testes de carga e caos, backpressure/circuit breaker.
- Supply chain e drift de infra
  - Solução: IaC (CloudFormation/Terraform) com revisão de código, assinaturas de artefatos, verificação de AMIs, patching via SSM, controle de drift.

Gere um Relatório de Modelagem de Ameaças, baseado na metodologia STRIDE:
- Escopo e ativos
  - Ativos: dados de usuários/documentos (EFS/RDS), credenciais/segredos, infraestrutura de rede, imagens AMI, logs/auditoria, chaves KMS, endpoints públicos (CloudFront/ALB), filas de e-mail (SES), configuração WAF.
  - Componentes no escopo: CloudFront, WAF, ALB, EC2 (SEI/SIP), Solr, RDS, EFS, ElastiCache, SES, CloudWatch, CloudTrail, KMS, Backup, VPC/Subnets/SG/NACL.
- Limites de confiança (trust boundaries)
  - Internet → CloudFront/WAF (fronteira pública)
  - CloudFront/ALB (borda de rede) → Subnets privadas (aplicação)
  - Aplicação → Dados (RDS/EFS/ElastiCache/Solr)
  - Conta/Região → Serviços de gestão (CloudTrail, KMS, Backup)
- Fluxos de dados principais
  - HTTP(S) Usuário → CloudFront/WAF → ALB → EC2 (SEI/SIP)
  - EC2 → RDS (SQL/TLS), EC2 → EFS (NFS/TLS), EC2 → ElastiCache (Memcached), EC2 → Solr (HTTP), EC2 → SES (SMTP/HTTPS API)
  - Telemetria: EC2/ALB/CloudFront → CloudWatch/CloudTrail; Backups → AWS Backup
- Assunções
  - Todo tráfego externo é TLS 1.2+; instâncias sem IP público; SG minimamente permissivos; CloudTrail/KMS habilitados; backups frequentes.
- Ameaças e mitigação (STRIDE)
  - S – Spoofing (falsificação de identidade)
    - Usuários ou serviços se passando por origens confiáveis; spoofing de origem CloudFront.
    - Mitigações: OIDC/SAML com MFA; assinaturas de sessão; validação de custom header entre CloudFront e ALB; SG dos ALBs só com ranges do CloudFront; mTLS interno opcional.
  - T – Tampering (adulteração)
    - Manipulação de requests ou payloads; alteração de dados em trânsito/repouso; modificação de AMIs/IaC.
    - Mitigações: TLS end-to-end; WAF com validação de entrada; hashing e assinatura de arquivos; KMS e políticas de key separation; pipeline CI/CD assinado e revisado; controles de integridade de AMIs (EC2 Image Builder).
  - R – Repudiation (negação de ações)
    - Falta de trilhas completas de quem fez o quê/quando.
    - Mitigações: CloudTrail org-wide e multi-região, logs imutáveis (S3 Object Lock), correlação de IDs (traceId) na app, sincronização de tempo (NTP), retenções e RBAC de leitura de logs.
  - I – Information Disclosure (exposição de dados)
    - Vazamento por logs, erros detalhados, cache compartilhado, buckets/volumes, snapshots ou e-mails.
    - Mitigações: Masking de PII, políticas de logging seguras, criptografia KMS em todos os artefatos, isolamento de cache por namespace, configuração SES com DKIM/DMARC, respostas de erro genéricas.
  - D – Denial of Service
    - DDoS L3/L7, exaustão de conexões ao RDS, picos de busca no Solr, bursts de upload para EFS.
    - Mitigações: Shield Advanced (avaliar), WAF rate-limit e challenge, CloudFront cache e TTLs, autoscaling com limites, connection pooling e read replicas (se aplicável), filas/limitação por IP/cliente, testes de carga e caos.
  - E – Elevation of Privilege
    - IAM roles excessivos; SSRF levando a metadata; consoles/admin de Solr; scripts de user-data com privilégios.
    - Mitigações: IAM least privilege e boundaries, IMDSv2 obrigatório, egress control, desabilitar/admin não autenticado no Solr, SSM para execução controlada, separação de funções (SoD).
- Riscos residuais e plano
  - Migrar Memcached → Redis com TLS/ACLs se houver requisito de confidencialidade no cache.
  - Considerar DR cross-region (RDS read replica em outra região, backups cross-account e cross-region).
  - Avaliar gerenciamento de segredos unificado (Secrets Manager) e rotação de chaves KMS.
  - Pentest anual e teste de restauração de backups trimestral.
- Requisitos de monitoramento e resposta
  - Alarmes CloudWatch/WAF/Shield/ELB/RDS; SIEM integrando CloudTrail/ALB/CloudFront; playbooks de incidente (DDoS, vazamento, ransomware EFS, indisponibilidade RDS).

Additional resourcing needs:
- Gerente do projeto (liderando o STRIDE) – coordenação das sessões e backlog de riscos.
- Network Administrator – requisitos de roteamento, NAT, VPC endpoints, regras de firewall/SG/NACL e integração com redes locais.
- Security Officer – validação de aderência a políticas, LGPD, classificação de dados e exceções de risco.
- Cloud/DevOps – IaC, pipelines, observabilidade, políticas de backup, hardening de AMIs e resposta a incidentes.
- Software Developer – desenho de APIs, autenticação/autorização, proteção contra OWASP, limites e telemetria de negócios.
- Hardware Developer (se houver dispositivos/IoT) – proteção de credenciais embarcadas e cadeia de suprimentos.
- Banco de Dados/DBA – parâmetros de segurança, desempenho, replicação/failover e plano de restauração.
- SecOps/Blue Team – detecção/alerta, caças a ameaças e exercícios de resposta.
- Compliance/Legal – retenção de logs, bases legais, políticas de e-mail (SES) e privacidade.

Sugestões de melhorias

Segue a entrega em 3 partes:
1) Diagrama mermaid com componentes, relacionamentos e correções aplicadas
2) Script Terraform de referência (sa-east-1), com práticas de segurança e alta disponibilidade
3) Relatório de Modelagem de Ameaças (STRIDE), conforme solicitado

1) Diagrama da arquitetura (mermaid)
```mermaid
flowchart TB
  %% Estilo
  classDef edge fill:#eef,stroke:#336,stroke-width:1px
  classDef data fill:#efe,stroke:#363,stroke-width:1px
  classDef app fill:#ffe,stroke:#663,stroke-width:1px
  classDef sec fill:#f8eefc,stroke:#734a91,stroke-width:1px
  classDef ops fill:#eefcfa,stroke:#2f6f64,stroke-width:1px
  classDef warn fill:#fff0f0,stroke:#cc0000,stroke-width:1px

  %% Usuários e Borda
  U[Usuários Globais]:::edge --> CF[(Amazon CloudFront\nTLS 1.2+, cache, OAC p/ S3 quando aplicável\nCustom Header p/ origem)]:::edge
  CF -- "Associado" --> WAF[AWS WAFv2 (Managed Rules,\nRate-limit, Bot Control, IP Reputation)]:::sec
  CF -- "Proteção DDoS L3/L4" --> SH[AWS Shield Standard]:::sec

  %% VPC e Sub-redes
  subgraph VPC[VPC - sa-east-1 (3 AZs)]
    direction TB
    subgraph PUB[Subnets Públicas (AZ A/B/C)]
      ALB[(ALB - internet-facing\nTLS 1.2+, SG: somente IPs do CloudFront\nListener 443 com regra de Header\nDefault 403)]:::edge
    end

    subgraph PRIVAPP[Subnets Privadas (Aplicação - AZ A/B/C)]
      ASG[ASG EC2 - SEI/SIP (IMDSv2, SSM,\nUserData Hardened)\nProtocolo interno: HTTPS (ALB->EC2)]:::app
    end

    subgraph PRIVDATA[Subnets Privadas (Dados - AZ A/B/C)]
      RDS[(Amazon RDS Multi-AZ\nCriptografia KMS, TLS, SG restrito,\nforce_ssl, logs, PI, backups)]:::data
      EFS[(Amazon EFS Multi-AZ\nKMS, TLS de montagem,\nAccess Points, SG restrito)]:::data
      REDIS[(ElastiCache Redis (recomendado)\nTLS + AUTH, Multi-AZ, SG restrito)]:::data
      SOLR[(Cluster Solr - ASG\nSomente tráfego da App, patching)]:::data
    end
  end

  %% Fluxos
  U -->|HTTPS| CF
  CF -->|HTTPS + Custom Header| ALB
```

```
ALB -->|HTTPS (ALB->EC2)| ASG
ASG -->|SQL/TLS| RDS
ASG -->|NFS/TLS| EFS
ASG -->|Redis TLS| REDIS
ASG -->|HTTP (privado)| SOLR

%% SES
SES[(Amazon SES\nDKIM/DMARC/SPF,\nEnvio transacional)]:::edge
ASG -->|SMTP/HTTPS| SES

%% Observabilidade, segurança e operações
subgraph OPS[Observabilidade, Segurança e Operações]
  CW[CloudWatch (métricas, logs, alarmes\nLogs criptografados KMS)]:::ops
  CT[CloudTrail org/multi-região\nS3 c/ Object Lock + KMS]:::ops
  KMS[KMS (chaves: RDS, EFS, EBS, Logs, Backup)]:::sec
  BAK[AWS Backup (Vault Lock,\nplanos e seleções RDS/EFS)]:::sec
end

CF --> CW
ALB --> CW
ASG --> CW
RDS --> CW
EFS --> CW
REDIS --> CW
SOLR --> CW
U -.auditoria.-> CT

%% Notas de correções de vulnerabilidades
note right of ALB:::warn
  Correções:
  - SG do ALB com IPs do CloudFront (data ip-ranges)
  - Listener 443: regra exigindo header secreto
  - HSTS e políticas TLS modernas
end

note right of ASG:::warn
  Correções:
  - IMDSv2 obrigatório e egress control
  - Segredos em Secrets Manager/SSM, não em AMI
  - IAM least privilege e boundaries
end

note right of REDIS:::warn
  Correções:
  - Migrar de Memcached -> Redis (TLS+AUTH)
  - SG estrito (somente App)
end

note right of RDS:::warn
  Correções:
  - Subnets privadas, SG restrito
  - Criptografia KMS + TLS
  - Backups e snapshots criptografados
  - Parâmetros de endurecimento
end

note right of SOLR:::warn
  Correções:
  - Subnets privadas
  - Admin desabilitado externamente
  - Patching e SG somente App
end

note right of EFS:::warn
  Correções:
  - Montagem com TLS
  - Access Points e POSIX
  - Backup e SG restrito
end
```

```
  note right of WAF:::warn
    Correções:
    - Managed Rules + rate-based
    - Bot control + IP reputation
    - Regras específicas da app
  end
```

2) Terraform (exemplo de referência)
Observações importantes:
- Este é um baseline seguro e modularizável. Ajuste nomes, tamanhos, classes de instância, domínios, zonas, senhas/segredos e quotas conforme seu ambiente.
- Inclui: VPC (3 AZs), NATs, ALB c/ header secreto e SG restrito a IPs do CloudFront, ASGs (App e Solr), RDS Multi-AZ (PostgreSQL), EFS, ElastiCache Redis (recomendado no lugar de Memcached), WAFv2 associado ao CloudFront, CloudFront com TLS 1.2+ e header secreto, CloudTrail com S3 Object Lock e KMS, AWS Backup com Vault Lock, SES (DKIM), KMS, VPC Endpoints essenciais, IMDSv2.
- O CloudFront exige certificado ACM em us-east-1; o ALB exige em sa-east-1.
- Regras de SG com todos os IPs do CloudFront podem atingir limites de regras; mantenha também a verificação do header secreto no ALB. Avalie usar proteção adicional (por exemplo, AWS Shield Advanced).

Estruture em arquivos main.tf, variables.tf e outputs.tf. A seguir um monolítico comentado para facilitar cópia inicial:

```hcl
##############################################
# providers.tf
##############################################
terraform {
  required_version = ">= 1.6.0"
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = ">= 5.50"
    }
    tls = {
      source  = "hashicorp/tls"
      version = ">= 4.0"
    }
    random = {
      source  = "hashicorp/random"
      version = ">= 3.5"
    }
  }
}

provider "aws" {
  region = var.region # "sa-east-1"
}

# Para ACM usado no CloudFront (obrigatório em us-east-1)
provider "aws" {
  alias  = "us_east_1"
  region = "us-east-1"
}

##############################################
# variables.tf
##############################################
variable "project_name" { type = string }
variable "region" { type = string ; default = "sa-east-1" }
variable "vpc_cidr" { type = string ; default = "10.0.0.0/16" }

variable "domain_name" { description = "FQDN público (ex: app.exemplo.gov.br)" ; type = string }
variable "hosted_zone_id" { description = "Route53 Hosted Zone ID para o domain_name" ; type = string }
variable "ses_domain" { description = "Domínio de envio no SES (pode ser igual ao domain_name ou
```

```
subdomínio)" ; type = string }

variable "origin_secret" { description = "Valor do header secreto CloudFront->ALB
(X-Origin-Secret)" ; type = string }
variable "redis_auth_token" { description = "Token AUTH do Redis (mín. 16 chars)" ; type = string ;
sensitive = true }

variable "app_instance_type" { type = string ; default = "t3.medium" }
variable "app_desired_capacity" { type = number ; default = 3 }
variable "solr_instance_type" { type = string ; default = "t3.medium" }
variable "solr_desired_capacity" { type = number ; default = 2 }

variable "db_username" { type = string ; default = "appuser" }
variable "db_name" { type = string ; default = "appdb" }
variable "db_allocated_storage" { type = number ; default = 100 }

################################################
# locals
################################################
locals {
  azs = ["sa-east-1a", "sa-east-1b", "sa-east-1c"]

  # CIDRs por AZ (ajuste conforme necessário)
  public_subnets = [
    cidrsubnet(var.vpc_cidr, 4, 0),
    cidrsubnet(var.vpc_cidr, 4, 1),
    cidrsubnet(var.vpc_cidr, 4, 2)
  ]
  private_app_subnets = [
    cidrsubnet(var.vpc_cidr, 4, 4),
    cidrsubnet(var.vpc_cidr, 4, 5),
    cidrsubnet(var.vpc_cidr, 4, 6)
  ]
  private_data_subnets = [
    cidrsubnet(var.vpc_cidr, 4, 8),
    cidrsubnet(var.vpc_cidr, 4, 9),
    cidrsubnet(var.vpc_cidr, 4, 10)
  ]
  tags = {
    Project = var.project_name
    Owner   = "SEI-SIP"
  }
}

################################################
# KMS - chaves separadas para dados e logs
################################################
resource "aws_kms_key" "data" {
  description         = "${var.project_name} data key (RDS/EFS/EBS/ElastiCache)"
  enable_key_rotation = true
  tags                = local.tags
}

resource "aws_kms_key" "logs" {
  description         = "${var.project_name} logs/backup key"
  enable_key_rotation = true
  tags                = local.tags
}

################################################
# VPC, Subnets, IGW, NAT, Roteamento
################################################
resource "aws_vpc" "main" {
  cidr_block           = var.vpc_cidr
  enable_dns_hostnames = true
  enable_dns_support   = true
  tags                 = merge(local.tags, { Name = "${var.project_name}-vpc" })
}
```

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

# Subnets públicas
resource "aws_subnet" "public" {
  for_each                = { for idx, az in local.azs : az => { az = az, cidr = local.public_subnets[idx] } }
  vpc_id                  = aws_vpc.main.id
  cidr_block              = each.value.cidr
  availability_zone       = each.value.az
  map_public_ip_on_launch = false
  tags = merge(local.tags, { Name = "${var.project_name}-public-${each.value.az}", Tier = "public" })
}

# Subnets privadas de aplicação
resource "aws_subnet" "private_app" {
  for_each          = { for idx, az in local.azs : az => { az = az, cidr = local.private_app_subnets[idx] } }
  vpc_id            = aws_vpc.main.id
  cidr_block        = each.value.cidr
  availability_zone = each.value.az
  tags = merge(local.tags, { Name = "${var.project_name}-privapp-${each.value.az}", Tier =
"private-app" })
}

# Subnets privadas de dados
resource "aws_subnet" "private_data" {
  for_each          = { for idx, az in local.azs : az => { az = az, cidr = local.private_data_subnets[idx] } }
  vpc_id            = aws_vpc.main.id
  cidr_block        = each.value.cidr
  availability_zone = each.value.az
  tags = merge(local.tags, { Name = "${var.project_name}-privdata-${each.value.az}", Tier =
"private-data" })
}

# NAT por AZ (HA)
resource "aws_eip" "nat" {
  for_each = aws_subnet.public
  domain   = "vpc"
  tags     = merge(local.tags, { Name = "${var.project_name}-eip-nat-${each.key}" })
}

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

# Tabelas de rota
resource "aws_route_table" "public" {
  vpc_id = aws_vpc.main.id
  tags   = merge(local.tags, { Name = "${var.project_name}-rt-public" })
}

resource "aws_route" "public_igw" {
  route_table_id         = aws_route_table.public.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id             = aws_internet_gateway.igw.id
}

resource "aws_route_table_association" "public_assoc" {
  for_each       = aws_subnet.public
  subnet_id      = each.value.id
  route_table_id = aws_route_table.public.id
}

# Tabelas privadas app por AZ
```

```
resource "aws_route_table" "private_app" {
  for_each = aws_nat_gateway.nat
  vpc_id   = aws_vpc.main.id
  tags     = merge(local.tags, { Name = "${var.project_name}-rt-privapp-${each.key}" })
}

resource "aws_route" "private_app_nat" {
  for_each               = aws_route_table.private_app
  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_app_assoc" {
  for_each       = aws_subnet.private_app
  subnet_id      = each.value.id
  route_table_id = aws_route_table.private_app[each.key].id
}

# Tabelas privadas dados por AZ
resource "aws_route_table" "private_data" {
  for_each = aws_nat_gateway.nat
  vpc_id   = aws_vpc.main.id
  tags     = merge(local.tags, { Name = "${var.project_name}-rt-privdata-${each.key}" })
}

resource "aws_route" "private_data_nat" {
  for_each               = aws_route_table.private_data
  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_data_assoc" {
  for_each       = aws_subnet.private_data
  subnet_id      = each.value.id
  route_table_id = aws_route_table.private_data[each.key].id
}

################################################
# VPC Endpoints (egress control + disponibilidade)
################################################
resource "aws_vpc_endpoint" "s3" {
  vpc_id           = aws_vpc.main.id
  service_name     = "com.amazonaws.${var.region}.s3"
  vpc_endpoint_type = "Gateway"
  route_table_ids  = [for rt in aws_route_table.private_app : rt.id] ++ [for rt in
aws_route_table.private_data : rt.id]
  tags             = merge(local.tags, { Name = "${var.project_name}-vpce-s3" })
}

locals {
  interface_endpoints = [
    "ssm", "ssmmessages", "ec2messages", "logs", "secretsmanager"
  ]
}
resource "aws_vpc_endpoint" "interfaces" {
  for_each           = toset(local.interface_endpoints)
  vpc_id             = aws_vpc.main.id
  service_name       = "com.amazonaws.${var.region}.${each.key}"
  vpc_endpoint_type  = "Interface"
  subnet_ids         = [for s in aws_subnet.private_app : s.id]
  security_group_ids = []
  private_dns_enabled = true
  tags               = merge(local.tags, { Name = "${var.project_name}-vpce-${each.key}" })
}

################################################
# SGs e IPs do CloudFront
```

```
##############################################
data "aws_ip_ranges" "cloudfront" {
  services = ["CLOUDFRONT"]
}

resource "aws_security_group" "alb" {
  name        = "${var.project_name}-alb-sg"
  description = "Permite 443 a partir dos IPs do CloudFront"
  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
}

# Cria múltiplas regras de ingress para todos os prefixos CloudFront (pode atingir limites; combine com header secreto)
resource "aws_security_group_rule" "alb_ingress_cf_ipv4" {
  for_each          = toset([for p in data.aws_ip_ranges.cloudfront.ipv4_prefixes : p.ip_prefix])
  type              = "ingress"
  security_group_id = aws_security_group.alb.id
  from_port         = 443
  to_port           = 443
  protocol          = "tcp"
  cidr_blocks       = [each.key]
  description       = "CloudFront IPv4"
}

resource "aws_security_group" "app" {
  name        = "${var.project_name}-app-sg"
  description = "App instances"
  vpc_id      = aws_vpc.main.id
  ingress {
    description     = "HTTPS do ALB"
    from_port       = 443
    to_port         = 443
    protocol        = "tcp"
    security_groups = [aws_security_group.alb.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" {
  name        = "${var.project_name}-rds-sg"
  description = "RDS acessível apenas pela App"
  vpc_id      = aws_vpc.main.id
  ingress {
    description     = "PostgreSQL da App"
    from_port       = 5432
    to_port         = 5432
    protocol        = "tcp"
    security_groups = [aws_security_group.app.id]
  }
  egress { from_port = 0, to_port = 0, protocol = "-1", cidr_blocks = ["0.0.0.0/0"] }
  tags = local.tags
}

resource "aws_security_group" "efs" {
  name        = "${var.project_name}-efs-sg"
  description = "EFS NFS 2049 apenas da App"
  vpc_id      = aws_vpc.main.id
  ingress {
    description     = "NFS da App"
    from_port       = 2049
    to_port         = 2049
```

```
    protocol        = "tcp"
    security_groups = [aws_security_group.app.id]
  }
  egress { from_port = 0, to_port = 0, protocol = "-1", cidr_blocks = ["0.0.0.0/0"] }
  tags = local.tags
}

resource "aws_security_group" "redis" {
  name        = "${var.project_name}-redis-sg"
  description = "Redis somente da App"
  vpc_id      = aws_vpc.main.id
  ingress {
    description     = "Redis TLS da App"
    from_port       = 6379
    to_port         = 6379
    protocol        = "tcp"
    security_groups = [aws_security_group.app.id]
  }
  egress { from_port = 0, to_port = 0, protocol = "-1", cidr_blocks = ["0.0.0.0/0"] }
  tags = local.tags
}

resource "aws_security_group" "solr" {
  name        = "${var.project_name}-solr-sg"
  description = "Solr somente da App"
  vpc_id      = aws_vpc.main.id
  ingress {
    description     = "HTTP 8983 da App"
    from_port       = 8983
    to_port         = 8983
    protocol        = "tcp"
    security_groups = [aws_security_group.app.id]
  }
  egress { from_port = 0, to_port = 0, protocol = "-1", cidr_blocks = ["0.0.0.0/0"] }
  tags = local.tags
}

##############################################
# ACM Certificados
##############################################
# Cert p/ CloudFront (em us-east-1)
resource "aws_acm_certificate" "cf" {
  provider          = aws.us_east_1
  domain_name       = var.domain_name
  validation_method = "DNS"
  tags              = local.tags
}

resource "aws_route53_record" "cf_cert_validation" {
  for_each = {
    for dvo in aws_acm_certificate.cf.domain_validation_options : dvo.domain_name => {
      name   = dvo.resource_record_name
      type   = dvo.resource_record_type
      record = dvo.resource_record_value
    }
  }
  zone_id = var.hosted_zone_id
  name    = each.value.name
  type    = each.value.type
  records = [each.value.record]
  ttl     = 60
}

resource "aws_acm_certificate_validation" "cf" {
  provider                = aws.us_east_1
  certificate_arn         = aws_acm_certificate.cf.arn
  validation_record_fqdns = [for r in aws_route53_record.cf_cert_validation : r.fqdn]
}
```

```
# Cert p/ ALB (na região do ALB)
resource "aws_acm_certificate" "alb" {
  domain_name       = var.domain_name
  validation_method = "DNS"
  tags              = local.tags
}

resource "aws_route53_record" "alb_cert_validation" {
  for_each = {
    for dvo in aws_acm_certificate.alb.domain_validation_options : dvo.domain_name => {
      name   = dvo.resource_record_name
      type   = dvo.resource_record_type
      record = dvo.resource_record_value
    }
  }
  zone_id = var.hosted_zone_id
  name    = each.value.name
  type    = each.value.type
  records = [each.value.record]
  ttl     = 60
}

resource "aws_acm_certificate_validation" "alb" {
  certificate_arn         = aws_acm_certificate.alb.arn
  validation_record_fqdns = [for r in aws_route53_record.alb_cert_validation : r.fqdn]
}

################################################
# ALB + Target Group + Listener com header secreto
################################################
resource "aws_lb" "app" {
  name               = "${var.project_name}-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
  idle_timeout               = 60
  tags                       = local.tags
}

# Target group usando HTTPS no backend (TLS fim-a-fim)
resource "aws_lb_target_group" "app" {
  name        = "${var.project_name}-tg"
  port        = 443
  protocol    = "HTTPS"
  vpc_id      = aws_vpc.main.id
  target_type = "instance"
  health_check {
    healthy_threshold   = 3
    unhealthy_threshold = 3
    timeout             = 5
    interval            = 15
    path                = "/health"
    protocol            = "HTTPS"
    matcher             = "200-399"
  }
  tags = local.tags
}

# Listener 80 -> redirect para 443
resource "aws_lb_listener" "http" {
  load_balancer_arn = aws_lb.app.arn
  port              = 80
  protocol          = "HTTP"
  default_action {
    type = "redirect"
    redirect {
      port        = "443"
```

```
      protocol    = "HTTPS"
      status_code = "HTTP_301"
    }
  }
}

# Listener 443 com default 403 e regra de header secreto para encaminhar
resource "aws_lb_listener" "https" {
  load_balancer_arn = aws_lb.app.arn
  port              = 443
  protocol          = "HTTPS"
  ssl_policy        = "ELBSecurityPolicy-TLS13-1-2-2021-06"
  certificate_arn   = aws_acm_certificate_validation.alb.certificate_arn
  default_action {
    type = "fixed-response"
    fixed_response {
      status_code  = "403"
      content_type = "text/plain"
      message_body = "Forbidden"
    }
  }
}

resource "aws_lb_listener_rule" "origin_secret_forward" {
  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]
    }
  }
}

################################################
# IAM Role/Instance Profile para EC2 (SSM, Logs, Secrets)
################################################
data "aws_iam_policy_document" "app_assume" {
  statement {
    actions = ["sts:AssumeRole"]
    principals { type = "Service" ; identifiers = ["ec2.amazonaws.com"] }
  }
}

resource "aws_iam_role" "app" {
  name               = "${var.project_name}-app-role"
  assume_role_policy = data.aws_iam_policy_document.app_assume.json
  tags               = local.tags
}

resource "aws_iam_role_policy_attachment" "ssm_core" {
  role       = aws_iam_role.app.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
}

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

# política mínima para ler segredos do app/*
data "aws_iam_policy_document" "app_secrets" {
  statement {
    actions   = ["secretsmanager:GetSecretValue", "ssm:GetParameter", "ssm:GetParameters"]
    resources = ["*"]
```

```
    condition {
      test     = "StringLike"
      variable = "aws:ResourceTag/Project"
      values   = [var.project_name]
    }
  }
}
resource "aws_iam_policy" "app_secrets" {
  name        = "${var.project_name}-app-secrets"
  description = "Least privilege para segredos do app"
  policy      = data.aws_iam_policy_document.app_secrets.json
}

resource "aws_iam_role_policy_attachment" "app_secrets_attach" {
  role       = aws_iam_role.app.name
  policy_arn = aws_iam_policy.app_secrets.arn
}

resource "aws_iam_instance_profile" "app" {
  name = "${var.project_name}-app-instance-profile"
  role = aws_iam_role.app.name
}

################################################
# Launch Templates e Auto Scaling Groups (App e Solr)
################################################
data "aws_ami" "al2023" {
  most_recent = true
  owners      = ["amazon"]
  filter { name = "name" ; values = ["al2023-ami-*-x86_64"] }
}

# UserData básico: habilita nginx com TLS autoassinado (demonstra TLS ALB->EC2)
locals {
  app_user_data = <<-EOF
    #!/bin/bash
    set -xe
    dnf -y update
    dnf -y install nginx openssl
    mkdir -p /etc/nginx/ssl
    openssl req -x509 -nodes -days 365 -newkey rsa:2048 -subj "/CN=${var.domain_name}" -keyout
/etc/nginx/ssl/server.key -out /etc/nginx/ssl/server.crt
    cat >/etc/nginx/nginx.conf <<NGINX
    events {}
    http {
      server {
        listen              443 ssl;
        ssl_certificate     /etc/nginx/ssl/server.crt;
        ssl_certificate_key /etc/nginx/ssl/server.key;
        location /health { return 200 "ok"; }
        location / {
          return 200 "app";
        }
      }
    }
    NGINX
    systemctl enable nginx && systemctl restart nginx
  EOF

  solr_user_data = <<-EOF
    #!/bin/bash
    set -xe
    dnf -y update
    # Instalação simplificada; em produção use AMI hardened + SSM
    dnf -y install java-17-amazon-corretto-headless
    useradd -m -s /bin/bash solr || true
    su - solr -c "curl -L https://archive.apache.org/dist/solr/solr/9.5.0/solr-9.5.0.tgz | tar -xz"
    su - solr -c "bash ~/solr-9.5.0/bin/install_solr_service.sh ~/solr-9.5.0.tgz -y"
    # Hardening: bind a 0.0.0.0 mas SG já restringe somente App
```

```
  EOF
}

resource "aws_launch_template" "app" {
  name_prefix   = "${var.project_name}-app-"
  image_id      = data.aws_ami.al2023.id
  instance_type = var.app_instance_type
  iam_instance_profile { name = aws_iam_instance_profile.app.name }
  vpc_security_group_ids = [aws_security_group.app.id]
  update_default_version = true
  user_data = base64encode(local.app_user_data)
  metadata_options {
    http_tokens = "required" # IMDSv2
  }
  tag_specifications {
    resource_type = "instance"
    tags          = local.tags
  }
}

resource "aws_autoscaling_group" "app" {
  name                      = "${var.project_name}-app-asg"
  min_size                  = 3
  max_size                  = 10
  desired_capacity          = var.app_desired_capacity
  vpc_zone_identifier       = [for s in aws_subnet.private_app : s.id]
  health_check_type         = "ELB"
  health_check_grace_period = 60
  launch_template {
    id      = aws_launch_template.app.id
    version = "$Latest"
  }
  target_group_arns = [aws_lb_target_group.app.arn]
  tag { key = "Project" ; value = var.project_name ; propagate_at_launch = true }
  lifecycle { ignore_changes = [desired_capacity] }
}

resource "aws_launch_template" "solr" {
  name_prefix   = "${var.project_name}-solr-"
  image_id      = data.aws_ami.al2023.id
  instance_type = var.solr_instance_type
  vpc_security_group_ids = [aws_security_group.solr.id]
  user_data = base64encode(local.solr_user_data)
  metadata_options { http_tokens = "required" }
  tag_specifications { resource_type = "instance" tags = local.tags }
}

resource "aws_autoscaling_group" "solr" {
  name                      = "${var.project_name}-solr-asg"
  min_size                  = 2
  max_size                  = 6
  desired_capacity          = var.solr_desired_capacity
  vpc_zone_identifier       = [for s in aws_subnet.private_data : s.id]
  health_check_type         = "EC2"
  health_check_grace_period = 120
  launch_template {
    id      = aws_launch_template.solr.id
    version = "$Latest"
  }
  tag { key = "Project" ; value = var.project_name ; propagate_at_launch = true }
}

##############################################
# RDS PostgreSQL Multi-AZ
##############################################
resource "random_password" "db" {
  length  = 24
  special = true
}
```

```
resource "aws_db_subnet_group" "main" {
  name       = "${var.project_name}-rds-subnets"
  subnet_ids = [for s in aws_subnet.private_data : s.id]
  tags       = local.tags
}

resource "aws_db_parameter_group" "pg" {
  name        = "${var.project_name}-pg"
  family      = "postgres14"
  description = "force SSL, logging"
  parameter {
    name  = "rds.force_ssl"
    value = "1"
  }
}

resource "aws_db_instance" "pg" {
  identifier                = "${var.project_name}-pg"
  engine                    = "postgres"
  engine_version              = "14.12"
  instance_class            = "db.m6g.large"
  username                  = var.db_username
  password                  = random_password.db.result
  db_name                   = var.db_name
  allocated_storage           = var.db_allocated_storage
  storage_encrypted           = true
  kms_key_id                = aws_kms_key.data.arn
  multi_az                  = true
  db_subnet_group_name        = aws_db_subnet_group.main.name
  vpc_security_group_ids      = [aws_security_group.rds.id]
  publicly_accessible         = false
  backup_retention_period     = 7
  copy_tags_to_snapshot       = true
  deletion_protection         = true
  apply_immediately           = false
  performance_insights_enabled = true
  performance_insights_kms_key_id = aws_kms_key.logs.arn
  enabled_cloudwatch_logs_exports = ["postgresql", "upgrade"]
  parameter_group_name        = aws_db_parameter_group.pg.name
  maintenance_window          = "Sun:03:00-Sun:04:00"
  backup_window               = "03:00-04:00"
  monitoring_interval         = 60
}

################################################
# EFS (com SG restrito e KMS)
################################################
resource "aws_efs_file_system" "main" {
  encrypted        = true
  kms_key_id       = aws_kms_key.data.arn
  performance_mode = "generalPurpose"
  throughput_mode  = "bursting"
  tags             = merge(local.tags, { Name = "${var.project_name}-efs" })
}

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

################################################
# ElastiCache Redis (recomendado, Multi-AZ, TLS+AUTH)
################################################
resource "aws_elasticache_subnet_group" "redis" {
  name       = "${var.project_name}-redis-subnets"
  subnet_ids = [for s in aws_subnet.private_data : s.id]
```

```
}

resource "aws_elasticache_replication_group" "redis" {
  replication_group_id          = "${var.project_name}-redis"
  description                   = "Redis com TLS + AUTH"
  engine                        = "redis"
  engine_version                = "7.1"
  node_type                     = "cache.t4g.medium"
  number_cache_clusters         = 2
  automatic_failover_enabled    = true
  multi_az_enabled              = true
  transit_encryption_enabled    = true
  at_rest_encryption_enabled    = true
  auth_token                    = var.redis_auth_token
  security_group_ids            = [aws_security_group.redis.id]
  subnet_group_name             = aws_elasticache_subnet_group.redis.name
  kms_key_id                    = aws_kms_key.data.arn
  port                          = 6379
  maintenance_window            = "sun:04:00-sun:05:00"
  tags                          = local.tags
}

################################################
# CloudFront + WAF (CLOUDFRONT scope)
################################################
resource "aws_wafv2_web_acl" "cf" {
  name        = "${var.project_name}-waf-cf"
  description = "Managed rules + rate limit + bot"
  scope       = "CLOUDFRONT"
  default_action { allow {} }
  visibility_config {
    cloudwatch_metrics_enabled = true
    metric_name                = "${var.project_name}-waf-cf"
    sampled_requests_enabled   = true
  }

  rule {
    name     = "AWSManagedCommon"
    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     = "BadInputs"
    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     = "IPReputation"
    priority = 3
    override_action { none {} }
    statement { managed_rule_group_statement { name =
"AWSManagedRulesAmazonIpReputationList" vendor_name = "AWS" } }
    visibility_config { cloudwatch_metrics_enabled = true, metric_name = "IPRep",
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 }
  }

  # Se licenciado: Bot Control (pode haver custo)
  # rule {
  #   name     = "BotControl"
  #   priority = 11
  #   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 }
  # }
}

resource "aws_cloudfront_distribution" "this" {
  enabled = true
  aliases = [var.domain_name]

  origin {
    domain_name = aws_lb.app.dns_name
    origin_id   = "alb-origin"
    custom_origin_config {
      origin_protocol_policy = "https-only"
      http_port              = 80
      https_port             = 443
      origin_ssl_protocols   = ["TLSv1.2"]
    }
    custom_header {
      name  = "X-Origin-Secret"
      value = var.origin_secret
    }
  }

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

  restrictions {
    geo_restriction { restriction_type = "none" }
  }

  viewer_certificate {
    acm_certificate_arn            = aws_acm_certificate_validation.cf.certificate_arn
    ssl_support_method             = "sni-only"
    minimum_protocol_version       = "TLSv1.2_2021"
  }

  web_acl_id = aws_wafv2_web_acl.cf.arn

  default_root_object = ""
  price_class         = "PriceClass_All"
  http_version        = "http2and3"
  is_ipv6_enabled     = true
  tags                = local.tags
}
```

```
# DNS para o CloudFront (CNAME)
resource "aws_route53_record" "app_cname" {
  zone_id = var.hosted_zone_id
  name    = var.domain_name
  type    = "CNAME"
  ttl     = 300
  records = [aws_cloudfront_distribution.this.domain_name]
}

################################################
# CloudTrail com S3 Object Lock + KMS
################################################
resource "aws_s3_bucket" "trail" {
  bucket              = "${var.project_name}-trail-${data.aws_caller_identity.me.account_id}"
  object_lock_enabled = true
  tags                = local.tags
}
data "aws_caller_identity" "me" {}

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

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

resource "aws_s3_bucket_object_lock_configuration" "trail" {
  bucket = aws_s3_bucket.trail.id
  rule { default_retention { mode = "GOVERNANCE" days = 90 } }
}

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

################################################
# CloudWatch Log Groups (exemplo)
################################################
resource "aws_cloudwatch_log_group" "app" {
  name              = "/${var.project_name}/app"
  retention_in_days = 90
  kms_key_id        = aws_kms_key.logs.arn
  tags              = local.tags
}

################################################
# AWS Backup (Vault Lock) + planos
################################################
resource "aws_backup_vault" "main" {
  name        = "${var.project_name}-vault"
  kms_key_arn = aws_kms_key.logs.arn
  tags        = local.tags
}

resource "aws_backup_vault_lock_configuration" "main" {
  backup_vault_name = aws_backup_vault.main.name
```

```
  min_retention_days = 30
  max_retention_days = 365
  changeable_for_days = 3
}

resource "aws_backup_plan" "main" {
  name = "${var.project_name}-plan"
  rule {
    rule_name         = "daily"
    target_vault_name = aws_backup_vault.main.name
    schedule          = "cron(0 3 * * ? *)"
    lifecycle { delete_after = 120 }
  }
  tags = local.tags
}

resource "aws_backup_selection" "rds" {
  name         = "rds-selection"
  iam_role_arn =
"arn:aws:iam::${data.aws_caller_identity.me.account_id}:role/service-role/AWSBackupDefaultServic
eRole"
  plan_id      = aws_backup_plan.main.id
  resources    = [aws_db_instance.pg.arn]
}

resource "aws_backup_selection" "efs" {
  name         = "efs-selection"
  iam_role_arn =
"arn:aws:iam::${data.aws_caller_identity.me.account_id}:role/service-role/AWSBackupDefaultServic
eRole"
  plan_id      = aws_backup_plan.main.id
  resources    = [aws_efs_file_system.main.arn]
}

##############################################
# SES (verificação de domínio + DKIM)
##############################################
resource "aws_ses_domain_identity" "ses" {
  domain = var.ses_domain
}

resource "aws_route53_record" "ses_verification" {
  zone_id = var.hosted_zone_id
  name    = "_amazonses.${var.ses_domain}"
  type    = "TXT"
  ttl     = 1800
  records = [aws_ses_domain_identity.ses.verification_token]
}

resource "aws_ses_domain_dkim" "dkim" {
  domain = aws_ses_domain_identity.ses.domain
}

resource "aws_route53_record" "ses_dkim" {
  count   = 3
  zone_id = var.hosted_zone_id
  name    = "${element(aws_ses_domain_dkim.dkim.dkim_tokens,
count.index)}._domainkey.${var.ses_domain}"
  type    = "CNAME"
  ttl     = 1800
  records = ["${element(aws_ses_domain_dkim.dkim.dkim_tokens,
count.index)}.dkim.amazonses.com"]
}

# SPF e DMARC (exemplo simples; ajuste política)
resource "aws_route53_record" "spf" {
  zone_id = var.hosted_zone_id
  name    = var.ses_domain
  type    = "TXT"
```

```
  ttl     = 300
  records = ["v=spf1 include:amazonses.com -all"]
}

resource "aws_route53_record" "dmarc" {
  zone_id = var.hosted_zone_id
  name    = "_dmarc.${var.ses_domain}"
  type    = "TXT"
  ttl     = 300
  records = ["v=DMARC1; p=quarantine; rua=mailto:dmarc@${var.ses_domain}"]
}

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

3) Relatório de Modelagem de Ameaças (STRIDE)

- Escopo e ativos
  - Ativos: dados de usuários/documentos (RDS/EFS), segredos/credenciais (Secrets Manager/SSM), infraestrutura VPC/SG/NACL, AMIs, logs (CloudWatch/ALB/CloudFront/Trail), chaves KMS, endpoints públicos (CloudFront/ALB), SES (filas/identidade), configuração WAF.
  - Componentes no escopo: CloudFront, WAF, ALB, EC2 (SEI/SIP), Solr, RDS, EFS, ElastiCache, SES, CloudWatch, CloudTrail, KMS, Backup, VPC/Subnets/SG/NACL.

- Limites de confiança
  - Internet → CloudFront/WAF (borda pública)
  - CloudFront/ALB (subnets públicas) → Subnets privadas (aplicação)
  - Aplicação → Dados (RDS/EFS/ElastiCache/Solr)
  - Conta/Região → Serviços de gestão (CloudTrail, KMS, Backup)

- Fluxos de dados principais
  - HTTP(S) Usuário → CloudFront/WAF → ALB → EC2 (SEI/SIP)
  - EC2 → RDS (SQL/TLS), EC2 → EFS (NFS/TLS), EC2 → ElastiCache (Redis TLS), EC2 → Solr (HTTP privado), EC2 → SES (SMTP/HTTPS API)
  - Telemetria: EC2/ALB/CloudFront → CloudWatch/CloudTrail; Backups → AWS Backup (Vault Lock)

- Assunções
  - TLS 1.2+ em todo tráfego externo; instâncias sem IP público; SG minimamente permissivos; CloudTrail/KMS habilitados; backups frequentes; app com health endpoint.

- Ameaças e mitigação (STRIDE)
  - Spoofing
    - Risco: fonte não autorizada diretamente no ALB; falsificação de identidade de usuário/serviço.
    - Mitigações: SG do ALB aceitando apenas IPs CloudFront; header secreto X-Origin-Secret validado no ALB; IAM com menor privilégio; MFA/OIDC/SAML no app; IMDSv2 obrigatório.
  - Tampering
    - Risco: adulteração em trânsito; alteração de AMIs/IaC.
    - Mitigações: TLS fim-a-fim (CloudFront→ALB e ALB→EC2); WAF com validação; KMS em repouso (RDS/EFS/EBS/Logs/Backups); pipeline assinado e revisão de código; EC2 Image Builder e controle de versões de AMI.
  - Repudiation
    - Risco: negação de ações sem trilha.
    - Mitigações: CloudTrail multi-região com S3 Object Lock e KMS; correlação de traceId; logs de acesso do ALB/CloudFront/ASG; retenção e RBAC controlados.
  - Information Disclosure
    - Risco: dados sensíveis em logs; erros verbosos; snapshots; cache.
    - Mitigações: masking/redação de PII; respostas genéricas; criptografia KMS; Redis com TLS+AUTH e namespaces; SES com DKIM/DMARC; políticas de acesso a logs com KMS; retenção adequada à LGPD.
  - Denial of Service

  - Risco: DDoS L3/L7; exaustão de conexões DB; picos de Solr/EFS.
  - Mitigações: CloudFront + Shield Standard; WAF rate-limit; cache agressivo; autoscaling com limites; connection pooling; alarmes e circuit breakers; avaliar Shield Advanced.
- Elevation of Privilege
  - Risco: permissões IAM excessivas; SSRF→metadata; consoles/admin abertos (Solr).
  - Mitigações: IAM least privilege + boundaries; IMDSv2; controle de egress + VPC endpoints; desabilitar/admin externo do Solr; uso de SSM em vez de chaves SSH; separação de funções.

- Riscos residuais e plano
- Migrar Memcached→Redis com TLS/ACLs (neste baseline já usamos Redis).
- Considerar DR cross-region (read replica/Global Database; backups cross-region e cross-account).
- Gestão unificada de segredos e rotação automática (Secrets Manager).
- Pentest anual e exercícios de restauração trimestrais.

- Requisitos de monitoramento e resposta
- Alarmes: WAF (bloqueios, rate-limit), ALB (5xx, Latência), ASG (capacidade), RDS (CPU, Conexões, FreeStorage), EFS (BurstCreditBalance), Redis (Evictions, CPU), Solr (métricas app), CloudFront (4xx/5xx).
- Integração com SIEM: CloudTrail, VPC Flow Logs (opcional, com KMS), logs ALB/CloudFront.
- Playbooks de incidente: DDoS, vazamento, ransomware em EFS (testar restauração), indisponibilidade RDS.

Observações finais
- Custos: NAT por AZ, CloudFront, Redis Multi-AZ, RDS Multi-AZ, WAF geram custos significativos. Planeje budgets e alarms.
- Quotas: Regras de SG para todos IPs do CloudFront podem se aproximar dos limites; mantenha a verificação do header no ALB e, se necessário, automatize atualização dos IPs com pipeline.
- Endurecimento adicional recomendado: NACLs restritivas, boundary policies no IAM, política de tagging, S3 Access Logs, e controles de DR.