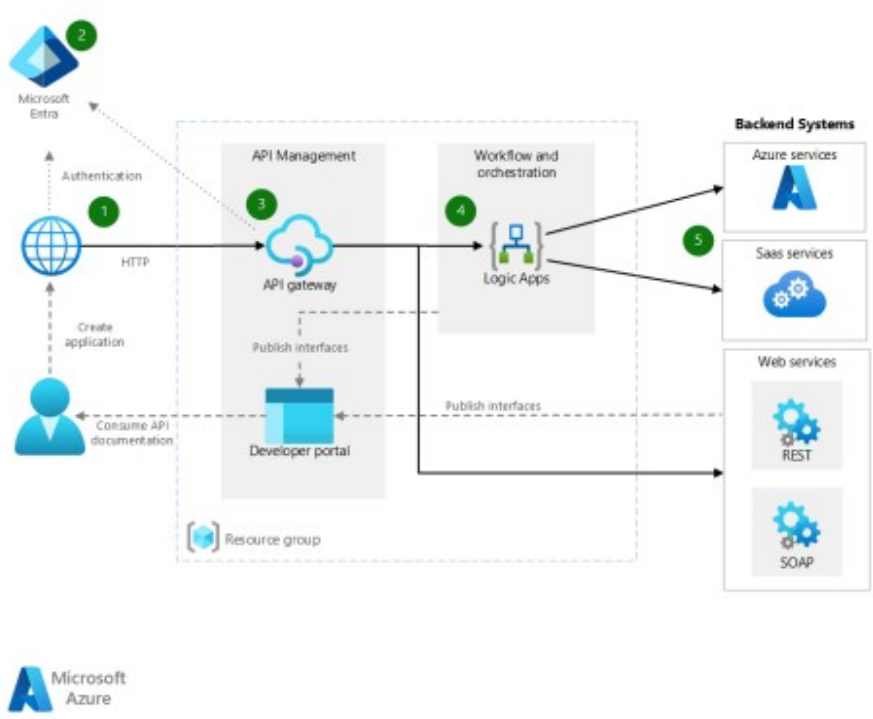
Relatório de análise da solução: Cloud_azure



Análise completa da solução atual

Modelo de cloud:
- Microsoft Azure

Lista com os componentes:
- Consumidor/Cliente da API (externo, via Internet)
- Microsoft Entra ID (antigo Azure AD) – autenticação e autorização
- Azure API Management (APIM)
  - API Gateway
  - Developer Portal
- Azure Logic Apps – workflow e orquestração
- Backend Systems
  - Azure services (ex.: Functions, App Services, Service Bus, Storage, SQL/ Cosmos DB)
  - SaaS services (ex.: Dynamics 365, Salesforce, Office 365)
  - Web services externos (REST e SOAP)
- Resource Group (escopo de implantação dos recursos)

Interação entre os componentes:
- 1. O Cliente faz requisições HTTP para o API Gateway no APIM, após registrar sua aplicação no Developer Portal e obter credenciais/subscription key.
- 2. O APIM delega a autenticação/autorização ao Microsoft Entra ID (OAuth2/OIDC, validação de tokens, escopos/roles).
- 3. O APIM aplica políticas (rate limit, validate-jwt, validação de esquema, transformação) e publica interfaces padronizadas.
- 4. O APIM aciona Logic Apps para orquestrar fluxos (chamadas sequenciais/paralelas, transformação de dados, enrichment, retries).
- 5. O Logic Apps integra-se com os Backends: serviços Azure (via private endpoints/managed identity), SaaS (conectores), e Web services REST/SOAP externos.

O que esse sistema faz:
- Fornece uma camada de API façade/integração para expor serviços internos e de terceiros com autenticação centralizada (Entra ID), governança e segurança via APIM, e orquestrações de negócios via Logic Apps. Permite que desenvolvedores descubram, assinem e consumam APIs de forma controlada pelo Developer Portal.

Vulnerabilidades e Solução para cada vulnerabilidade:
- V1. Spoofing de identidade (tokens falsos/roubados)

  - Solução: validate-jwt no APIM com issuer/audience/assinatura; OAuth2/OIDC com PKCE para públicos; lifetimes curtos + refresh tokens rotativos; condicionar por Conditional Access (MFA, device compliance).
- V2. Exposição de subscription keys no cliente
  - Solução: preferir OAuth2/OIDC e managed identity; se usar keys, rotacionar automaticamente, restringir por IP, usar APIM Certificates/mTLS; nunca embutir em apps móveis; usar vault/secret manager do lado servidor.
- V3. Injeção e payload malicioso nas APIs
  - Solução: políticas de validação de esquema (OpenAPI, JSON/XML schema), content-size limit, block-Regex, sanitização e canonicalização; WAF em frente (App Gateway + WAF).
- V4. Falta de rate limiting levando a DoS
  - Solução: quotas e rate-limit por produto/assinatura no APIM; DDoS Protection Standard na VNet; autoscale do APIM/Logic Apps; backpressure e circuit breakers.
- V5. Conexões inseguras com backends externos
  - Solução: TLS 1.2+ obrigatório, certificate pinning/mTLS quando suportado; permitir apenas cipher suites fortes; validar hostname e revogação (OCSP).
- V6. Vazamento de dados sensíveis em logs
  - Solução: mascaramento/redação no APIM (política set-variable/mask), segregação de logs com RBAC, retenção mínima necessária, Private Link para Log Analytics; criptografia em repouso.
- V7. Segredos nas definições do Logic Apps/APIM
  - Solução: Azure Key Vault com RBAC, purge protection e soft-delete; Managed Identity para APIM/Logic Apps; desabilitar segredos inline em parâmetros.
- V8. Exposição pública indevida do Developer Portal/APIs
  - Solução: restringir por IP/Network (VNet, Private Link, self-hosted gateway), CORS estrito, desabilitar listagem pública, revisão de produtos/APIs “unlimited”.
- V9. Falta de versionamento e mudança breaking
  - Solução: versionamento de APIs no APIM (path/header/query), depreciação com prazos e comunicação; testes de contrato (consumer-driven).
- V10. Falhas de autorização entre serviços
  - Solução: ABAC/RBAC fino por escopos/roles no token; validação de claims no APIM/Logic Apps; separation of duties nas identidades gerenciadas.
- V11. SSRF via conectores/HTTP no Logic Apps
  - Solução: usar VNet Integration/ISE (ou Standard) com regras de saída, deny by default, allowlist de endpoints; desabilitar resolução interna desnecessária.
- V12. Supply chain de conectores/artefatos
  - Solução: políticas de “conectores aprovados”, revisão de versões, assinaturas e provenance; Defender for Cloud e CI/CD com verificação de segurança.
- V13. Configuração fraca de TLS/CSP/CORS
  - Solução: TLS 1.2/1.3 only, HSTS, CSP restritiva para portais, CORS por origem exata e métodos necessários.

Relatório de Modelagem de Ameaças (STRIDE):
- Escopo e fluxos principais
  - Trust boundaries: Internet ↔ APIM; APIM ↔ Entra ID; APIM ↔ Logic Apps; Logic Apps ↔ Backends; Dev Portal ↔ Desenvolvedores.
  - Dados: tokens OAuth/JWT, subscription keys, payloads de negócio, segredos de conexão, logs e telemetria.

- S – Spoofing
  - Ameaças: uso de tokens roubados, impersonação de cliente/serviço, sequestro de sessão do portal.
  - Mitigações: OAuth2/OIDC com PKCE e MFA; validate-jwt e verificação de audience/scope no APIM; mTLS onde aplicável; Managed Identity para chamadas internas; timeout e revogação de sessão; Conditional Access.

- T – Tampering
  - Ameaças: alteração de payloads em trânsito, manipulação de políticas do APIM, pipelines maliciosos.
  - Mitigações: TLS 1.2+ end-to-end; assinaturas de mensagem (quando SOAP); controle de alterações com IaC e revisões (PRs) + RBAC; APIM policy fragments versionados; hashing/ETag; verificação de integridade.

- R – Repudiation
  - Ameaças: negação de autoria de chamadas, ausência de trilhas de auditoria.
  - Mitigações: logging imutável no Log Analytics/Storage com retenção e bloqueio legal; correlation IDs; clock sync; assinatura de logs; acesso a logs via RBAC e PIM.

- I – Information Disclosure
  - Ameaças: vazamento de PII/segredos em headers/logs, erro detalhado exposto, exfiltração via conectores.

  - Mitigações: mascaramento/redação de logs; políticas de remove-headers; páginas de erro genéricas; Key Vault; VNet/Private Link; DLP e egress control para Logic Apps.

- D – Denial of Service
  - Ameaças: flood de requisições ao APIM/Logic Apps, esgotamento de conexões a backends.
  - Mitigações: quotas/rate-limit/burst no APIM; caching onde possível; autoscale; DDoS Protection Standard; circuit breaker e retries exponenciais; filas (Service Bus) para desacoplamento.

- E – Elevation of Privilege
  - Ameaças: permissões excessivas em Managed Identities, bypass de políticas no APIM, abuso do portal.
  - Mitigações: princípio do menor privilégio; segmentação por RG/Subscription; PIM para roles privilegiadas; escopos específicos por API; validação de claims de role/scope; revisão periódica de acessos.

- Controles recomendados adicionais
  - Rede: APIM/Logic Apps integrados a VNet, private endpoints para backends, Private DNS Zones, NSG/UDR com egress control.
  - Segurança de borda: Application Gateway + WAF em frente ao APIM; CORS estrito; HSTS.
  - Gestão de segredos: Key Vault com purge protection; rotação automática; customer-managed keys onde aplicável.
  - Observabilidade: Azure Monitor, Log Analytics, distributed tracing; alertas operacionais e de segurança (Defender for Cloud).
  - Ciclo de vida: IaC (Bicep/Terraform) com validações, DevSecOps (SAST/DAST), testes de contrato, azul/verde/canary.
  - Conformidade: classificação de dados, criptografia em repouso, GDPR/Lei Geral de Proteção de Dados quando aplicável.

Additional resourcing needs:
- Administrador de Rede: validar requisitos de VNet, Private Link, DNS privado, NSG/UDR e DDoS.
- Security Officer: garantir aderência a padrões de segurança, políticas de acesso, logging e retenção.
- Equipe de Cloud/DevOps: definir arquitetura de hosting, CI/CD, IaC, observabilidade e governança (naming, tagging, RBAC).
- Desenvolvedor de Software: definir contratos de API, segurança de chamadas, versionamento, testes e SDKs.
- Desenvolvedor de Hardware (se houver dispositivos): proteger credenciais em hardware, provisionamento seguro e attestation.

Resumo
- Tipo de solução: API-led integration com API Gateway (APIM) e orquestração (Logic Apps) sobre Azure, expondo serviços internos e de terceiros de forma segura e governada.
- Principais controles a priorizar no início: OAuth2/OIDC com validate-jwt no APIM, quotas/rate-limit, VNet/Private Link, Managed Identity + Key Vault, WAF em frente ao APIM, logging/monitoramento centralizado e RBAC/PIM.

Sugestões de melhorias

Segue a entrega solicitada: diagrama Mermaid da arquitetura (com correções embutidas nos pontos de vulnerabilidade) e um script Terraform de referência para provisionamento no Azure. O Terraform foca no esqueleto seguro: APIM em modo interno atrás de Application Gateway WAF v2, integração de rede/VNet + Private Link, Managed Identities + Key Vault e políticas básicas no APIM (validate-jwt, rate-limit, CORS). Ajuste domínios, certificados e integrações específicas do backend conforme seu ambiente.

Diagrama Mermaid (arquitetura, relações e correções)
```mermaid
flowchart TD
  %% Trust Boundaries
  subgraph TB1[Trust Boundary: Internet]
    C[Cliente/Consumidor da API]
    Devs[Desenvolvedores (acesso ao Portal)]
    ExtWS[Web Services Externos (REST/SOAP)]
  end

  subgraph TB2[Trust Boundary: Microsoft Entra ID]
    AAD[Entra ID (OAuth2/OIDC, Conditional Access, MFA)]
  end
```

```
  subgraph AZ[Azure Subscription / Resource Group]
    DDoS[DDoS Protection Standard]

    subgraph NET[VNet + NSG/UDR + Private DNS Zones]
      subgraph Edge[Borda e Proteção]
        WAF[Application Gateway WAF v2<br/>HSTS, TLS 1.2/1.3, CSP/CORS estrito]
      end

      subgraph APIMSG[API Management (Internal/VNet)]
        APIM[API Gateway<br/>Políticas: validate-jwt, rate-limit, quotas,<br/>schema validation,
transformação, remove-headers]
        DevPortal[Developer Portal (custom domain)]
      end

      subgraph ORQ[Orquestração]
        LA[Logic Apps Standard<br/>VNet Integration, Managed Identity]
      end

      KV[Azure Key Vault<br/>RBAC, purge protection, soft-delete]
      LAW[Log Analytics + Azure Monitor]
      AFW[Azure Firewall (egress allowlist p/ SSRF)]

      subgraph BEs[Backends]
        subgraph AZS[Serviços Azure via Private Endpoint]
          FN[Functions]
          AS[App Service]
          SB[Service Bus]
          ST[(Storage)]
          SQL[(Azure SQL)]
          COS[(Cosmos DB)]
        end
        SAAS[SaaS (D365, Salesforce, O365) - conectores aprovados]
      end
    end
  end

  %% Fluxos principais
  C -->|HTTPS| WAF -->|HTTPS (SNI p/ domínio do gateway)| APIM
  Devs -->|HTTPS| WAF --> DevPortal

  APIM <-->|OIDC/OAuth2 (tokens/escopos/roles)| AAD

  APIM -->|Chama| LA
  LA -->|Managed Identity + Private Link| AZS
  LA -->|Conectores aprovados| SAAS
  LA -->|HTTPS egress via allowlist| ExtWS

  %% Observabilidade e segredos
  APIM -.->|Logs mascarados/telemetria| LAW
  LA -.-> LAW
  WAF -.-> LAW
  APIM -.->|Managed Identity (Get Secret)| KV
  LA -.->|Managed Identity (Get Secret)| KV

  %% Notas de segurança (mapeiam vulnerabilidades -> controles)
  note right of APIM
    V1: validate-jwt + OIDC/PKCE + MFA/CA
    V2: Preferir OAuth2/MI; rotacionar keys; mTLS opcional
    V3: Schema validation; WAF OWASP; size limits
    V4: rate-limit/quotas; autoscale
    V5: TLS 1.2+; mTLS/pinning se suportado
    V6: Redação de logs; remove-headers
    V7: Segredos no Key Vault
    V8: APIM interno; WAF na borda; CORS estrito
    V9: Versionamento no APIM
    V10: Claims/roles/escopos verificados
    V11: Egress control (Firewall/UDR/deny-by-default)
    V12: Conectores aprovados + DevSecOps
    V13: TLS/HSTS/CSP/CORS
```

```
  end
```

Correção das vulnerabilidades (resumo objetivo)
- V1 Spoofing: validate-jwt no APIM; OAuth2/OIDC com PKCE; MFA/Conditional Access; tokens curtos e refresh rotativo.
- V2 Keys expostas: usar OAuth2/Managed Identity; rotação automática; mTLS quando possível; nunca embutir keys em cliente; segredos no Key Vault.
- V3 Injeção/payload: validação de esquema; limite de tamanho; WAF OWASP; sanitização/regex; transformação no APIM.
- V4 DoS: rate-limit e quotas por produto/assinatura; autoscale; DDoS Std; circuit breaker/backpressure.
- V5 Conexões inseguras: TLS 1.2+ end-to-end; pinning/mTLS; validação de hostname/revogação.
- V6 Vazamento em logs: mascaramento/redação; remover headers; RBAC nos logs; criptografia e retenção mínima.
- V7 Segredos em IaC/definições: Key Vault + MI; purge/soft-delete; proibir segredos inline.
- V8 Exposição indevida: APIM em VNet (Internal) atrás de WAF; CORS estrito; revisar produtos/APIs; desabilitar hostnames default.
- V9 Versionamento: version sets no APIM; políticas de depreciação e comunicação.
- V10 Autorização entre serviços: roles/escopos no token; validação de claims no APIM/Logic Apps; privilégio mínimo.
- V11 SSRF: VNet Integration; UDR/Firewall com deny-by-default e allowlist; desabilitar resoluções desnecessárias.
- V12 Supply chain: conectores aprovados; verificação de versões/assinaturas; Defender for Cloud; CI/CD com checagens.
- V13 TLS/CSP/CORS: TLS 1.2/1.3 only; HSTS; CSP restritiva no portal; CORS por origem exata e métodos necessários.

Terraform de referência
Observações:
- Ajuste variáveis marcadas TODO: domínios, certificados, ranges de IP, SKU, etc.
- O APIM é implantado em modo Internal na VNet e publicado por um Application Gateway WAF v2. É necessário ter certificados válidos para os domínios do gateway e do portal guardados em um Key Vault (ou forneça o Secret ID).
- O backend Logic Apps Standard é criado com Managed Identity e integração de rede. O vínculo do APIM para o endpoint específico do workflow é mostrado como exemplo via backend de URL; substitua pelo seu endpoint real.
- Inclui Private Endpoint de exemplo para Storage e Log Analytics para diagnósticos.

```hcl
terraform {
  required_version = ">= 1.6.0"
  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "~> 3.113" # ou mais novo
    }
  }
}

provider "azurerm" {
  features {}
}

data "azurerm_client_config" "current" {}

##########################
# Variáveis principais #
##########################

variable "prefix" {
  description = "Prefixo de nomeação dos recursos"
  type        = string
  default     = "apilayer"
}

variable "location" {
  description = "Região do Azure"
  type        = string
```

```
  default     = "brazilsouth"
}

variable "apim_gateway_host" {
  description = "Hostname customizado do gateway do APIM (ex.: api.example.com)"
  type        = string
}

variable "apim_portal_host" {
  description = "Hostname customizado do Developer Portal (ex.: portal-api.example.com)"
  type        = string
}

variable "apim_gateway_cert_secret_id" {
  description = "Key Vault Secret ID do certificado PFX do gateway (api.example.com)"
  type        = string
}

variable "apim_portal_cert_secret_id" {
  description = "Key Vault Secret ID do certificado PFX do portal"
  type        = string
}

variable "allowed_cors_origins" {
  description = "Lista de origens CORS permitidas"
  type        = list(string)
  default     = ["https://app.example.com"]
}

variable "logicapp_allowed_egress" {
  description = "Destinos permitidos de saída (egress) para Logic Apps (ex.: APIs externas)"
  type        = list(string)
  default     = ["api.partner.com", "api2.partner.com"]
}

variable "logic_app_storage_sku" {
  type        = string
  default     = "Standard_LRS"
}

variable "apim_sku" {
  description = "SKU do APIM (Premium para VNet Internal)"
  type        = string
  default     = "Premium"
}

#####################
# Resource Group   #
#####################

resource "azurerm_resource_group" "rg" {
  name     = "${var.prefix}-rg"
  location = var.location
  tags = {
    env   = "prod"
    owner = "integration"
  }
}

##########################
# Log Analytics + Logs  #
##########################

resource "azurerm_log_analytics_workspace" "law" {
  name                = "${var.prefix}-law"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  sku                 = "PerGB2018"
  retention_in_days   = 30
```

```
}

##################
# Networking    #
##################

resource "azurerm_ddos_protection_plan" "ddos" {
  name                = "${var.prefix}-ddos"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
}

resource "azurerm_virtual_network" "vnet" {
  name                = "${var.prefix}-vnet"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  address_space       = ["10.40.0.0/16"]
  ddos_protection_plan {
    id     = azurerm_ddos_protection_plan.ddos.id
    enable = true
  }
  tags = {
    env = "prod"
  }
}

resource "azurerm_subnet" "snet_appgw" {
  name                 = "snet-appgw"
  resource_group_name  = azurerm_resource_group.rg.name
  virtual_network_name = azurerm_virtual_network.vnet.name
  address_prefixes     = ["10.40.0.0/24"]
}

resource "azurerm_subnet" "snet_apim" {
  name                 = "snet-apim"
  resource_group_name  = azurerm_resource_group.rg.name
  virtual_network_name = azurerm_virtual_network.vnet.name
  address_prefixes     = ["10.40.1.0/24"]
}

resource "azurerm_subnet" "snet_la" {
  name                 = "snet-logicapps"
  resource_group_name  = azurerm_resource_group.rg.name
  virtual_network_name = azurerm_virtual_network.vnet.name
  address_prefixes     = ["10.40.2.0/24"]
  delegation {
    name = "del-web"
    service_delegation {
      name = "Microsoft.Web/serverFarms"
      actions = [
        "Microsoft.Network/virtualNetworks/subnets/action",
      ]
    }
  }
}

resource "azurerm_subnet" "snet_pe" {
  name                 = "snet-private-endpoints"
  resource_group_name  = azurerm_resource_group.rg.name
  virtual_network_name = azurerm_virtual_network.vnet.name
  address_prefixes     = ["10.40.3.0/24"]
  private_endpoint_network_policies_enabled = false
}

############################
# Application Gateway WAF  #
############################

resource "azurerm_public_ip" "appgw_pip" {
```

```
  name                = "${var.prefix}-agw-pip"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  allocation_method   = "Static"
  sku                 = "Standard"
}

resource "azurerm_web_application_firewall_policy" "wafpol" {
  name                = "${var.prefix}-wafpol"
  resource_group_name = azurerm_resource_group.rg.name
  location            = azurerm_resource_group.rg.location

  policy_settings {
    enabled                     = true
    mode                        = "Prevention"
    request_body_check          = true
    file_upload_limit_in_mb     = 10
    max_request_body_size_in_kb = 128
  }

  managed_rules {
    managed_rule_set {
      type    = "OWASP"
      version = "3.2"
      rule_group_override {
        rule_group_name = "REQUEST-913-SCANNER-DETECTION"
        disabled_rules  = []
      }
    }
  }
}

resource "azurerm_application_gateway" "appgw" {
  name                = "${var.prefix}-agw"
  resource_group_name = azurerm_resource_group.rg.name
  location            = azurerm_resource_group.rg.location
  firewall_policy_id  = azurerm_web_application_firewall_policy.wafpol.id

  sku {
    name     = "WAF_v2"
    tier     = "WAF_v2"
    capacity = 2
  }

  autoscale_configuration {
    min_capacity = 2
    max_capacity = 10
  }

  gateway_ip_configuration {
    name      = "gwipc"
    subnet_id = azurerm_subnet.snet_appgw.id
  }

  frontend_port {
    name = "fe-https"
    port = 443
  }

  frontend_ip_configuration {
    name                 = "feip"
    public_ip_address_id = azurerm_public_ip.appgw_pip.id
  }

  backend_address_pool {
    name  = "be-apim-gateway"
    fqdns = [var.apim_gateway_host] # Resolve via Private DNS para IP privado do APIM
  }
```

```
  backend_http_settings {
    name                                = "be-https-443"
    protocol                            = "Https"
    port                                = 443
    cookie_based_affinity               = "Disabled"
    request_timeout                     = 60
    pick_host_name_from_backend_address = true
  }

  http_listener {
    name                           = "lis-apim"
    frontend_ip_configuration_name = "feip"
    frontend_port_name             = "fe-https"
    protocol                       = "Https"
    ssl_certificate_name           = "agw-ssl"
    host_name                      = var.apim_gateway_host
  }

  ssl_certificate {
    name = "agw-ssl"
    # Recomendado: usar Key Vault reference via SSL profiles; aqui simplificado com PFX
    data     = filebase64("certs/${var.apim_gateway_host}.pfx")   # TODO: fornecer PFX
    password = var.pfx_password                                   # TODO: definir variável
  }

  request_routing_rule {
    name                       = "rr-apim"
    rule_type                  = "Basic"
    http_listener_name         = "lis-apim"
    backend_address_pool_name  = "be-apim-gateway"
    backend_http_settings_name = "be-https-443"
    priority                   = 10
  }

  tags = {
    env = "prod"
  }
}

variable "pfx_password" {
  description = "Senha do PFX do certificado usado no App Gateway"
  type        = string
  sensitive   = true
}

#####################################
# Private DNS para domínios APIM
#####################################

# Zona privada para o domínio base (ex.: example.com) para resolver o host do APIM para IP privado
# Alternativamente, use um split-horizon DNS corporativo.
resource "azurerm_private_dns_zone" "zone_custom" {
  name                = replace(var.apim_gateway_host, "/^[^.]+\\./", "") # pega o domínio base
  resource_group_name = azurerm_resource_group.rg.name
}

resource "azurerm_private_dns_zone_virtual_network_link" "zone_link" {
  name                  = "${var.prefix}-zone-link"
  resource_group_name   = azurerm_resource_group.rg.name
  private_dns_zone_name = azurerm_private_dns_zone.zone_custom.name
  virtual_network_id    = azurerm_virtual_network.vnet.id
  registration_enabled  = false
}

# Registros A para gateway e portal apontando para IP privado do APIM (preenchidos após criação do
APIM)
resource "azurerm_private_dns_a_record" "apim_gateway_a" {
  name                = chomp(replace(var.apim_gateway_host,
"/\\.${azurerm_private_dns_zone.zone_custom.name}$/", ""))
```

```
  zone_name           = azurerm_private_dns_zone.zone_custom.name
  resource_group_name = azurerm_resource_group.rg.name
  ttl                 = 300
  records             = [element(azurerm_api_management.apim.private_ip_addresses, 0)]
  depends_on          = [azurerm_api_management.apim]
}

############################
# Key Vault (suportes)   #
############################

resource "azurerm_key_vault" "kv" {
  name                        = "${var.prefix}kv${substr(replace(uuid(), "-", ""), 0, 6)}"
  location                    = azurerm_resource_group.rg.location
  resource_group_name         = azurerm_resource_group.rg.name
  tenant_id                   = data.azurerm_client_config.current.tenant_id
  sku_name                    = "standard"
  purge_protection_enabled    = true
  soft_delete_retention_days = 90
  public_network_access_enabled = true
}

##############################
# Logic Apps Standard (VNet)
##############################

resource "azurerm_storage_account" "la_sa" {
  name                     = lower(replace("${var.prefix}lsa${substr(replace(uuid(), "-", ""), 0, 8)}", "-", ""))
  resource_group_name      = azurerm_resource_group.rg.name
  location                 = azurerm_resource_group.rg.location
  account_tier             = "Standard"
  account_replication_type = "LRS"
  min_tls_version          = "TLS1_2"
}

resource "azurerm_service_plan" "la_plan" {
  name                = "${var.prefix}-la-plan"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  os_type             = "Linux"
  sku_name            = "WS1" # Workflow Standard (ajuste conforme carga)
}

resource "azurerm_logic_app_standard" "la" {
  name                       = "${var.prefix}-la"
  location                   = azurerm_resource_group.rg.location
  resource_group_name        = azurerm_resource_group.rg.name
  service_plan_id            = azurerm_service_plan.la_plan.id
  storage_account_name       = azurerm_storage_account.la_sa.name
  storage_account_access_key = azurerm_storage_account.la_sa.primary_access_key

  identity {
    type = "SystemAssigned"
  }

  # Disponível em provedores recentes: integração VNet direta
  # Caso seu provedor não tenha, use swift_connection abaixo.
  virtual_network_subnet_id = azurerm_subnet.snet_la.id

  app_settings = {
    WEBSITE_RUN_FROM_PACKAGE = "1"
    FUNCTIONS_WORKER_RUNTIME = "node"
  }

  depends_on = [azurerm_virtual_network.vnet]
}

# Alternativa para integração VNet (se necessário):
# resource "azurerm_app_service_virtual_network_swift_connection" "la_vnet" {
```

```
#   app_service_id = azurerm_logic_app_standard.la.id
#   subnet_id      = azurerm_subnet.snet_la.id
# }

##############################
# API Management (Internal) #
##############################

resource "azurerm_api_management" "apim" {
  name                = "${var.prefix}-apim"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  publisher_name      = "Org"
  publisher_email     = "api-admin@example.com"

  sku_name = "${var.apim_sku}_1" # Premium_1 recomendado p/ VNet Internal

  virtual_network_type = "Internal"
  virtual_network_configuration {
    subnet_id = azurerm_subnet.snet_apim.id
  }

  identity {
    type = "SystemAssigned"
  }

  # Endurecimento TLS
  protocols {
    enable_http2 = true
  }

  # Desabilita TLS 1.0/1.1
  custom_properties = {
    "Microsoft.WindowsAzure.ApiManagement.Gateway.Security.Protocols.Tls10" = "false"
    "Microsoft.WindowsAzure.ApiManagement.Gateway.Security.Protocols.Tls11" = "false"
  }

  # Hostnames customizados com certificados no Key Vault
  hostname_configuration {
    proxy {
      host_name    = var.apim_gateway_host
      key_vault_id = var.apim_gateway_cert_secret_id
      negotiate_client_certificate = false
    }
    portal {
      host_name    = var.apim_portal_host
      key_vault_id = var.apim_portal_cert_secret_id
    }
  }

  tags = {
    env = "prod"
  }
}

# Provedor OIDC (Entra ID)
resource "azurerm_api_management_openid_connect_provider" "oidc" {
  name                = "entra-id"
  api_management_name = azurerm_api_management.apim.name
  resource_group_name = azurerm_resource_group.rg.name
  display_name        = "Microsoft Entra ID"
  metadata_endpoint   =
"https://login.microsoftonline.com/${data.azurerm_client_config.current.tenant_id}/v2.0/.well-known/
openid-configuration"
  client_id           = "REPLACE_WITH_APP_ID"      # TODO: App Registration (cliente público/
confidencial)
  client_secret       = "REPLACE_WITH_CLIENT_SECRET_IF_CONFIDENTIAL" # ou use
credencial via Key Vault reference
}
```

```
# Produto com quotas/limites
resource "azurerm_api_management_product" "product" {
  product_id          = "std"
  api_management_name = azurerm_api_management.apim.name
  resource_group_name = azurerm_resource_group.rg.name
  display_name        = "Standard"
  subscription_required = true
  approval_required   = true
  published           = true
}

# Política do produto para quotas e rate-limit
resource "azurerm_api_management_product_policy" "product_policy" {
  api_management_name = azurerm_api_management.apim.name
  resource_group_name = azurerm_resource_group.rg.name
  product_id          = azurerm_api_management_product.product.product_id

  xml_content = <<POL
<policies>
  <inbound>
    <base />
    <rate-limit calls="100" renewal-period="60" />
    <quota calls="10000" renewal-period="604800" /> <!-- semana -->
  </inbound>
  <backend><base /></backend>
  <outbound><base /></outbound>
  <on-error><base /></on-error>
</policies>
POL
}

# API de exemplo apontando para Logic Apps (substitua o backend_url real do workflow)
resource "azurerm_api_management_api" "api" {
  name                = "sample-api"
  resource_group_name = azurerm_resource_group.rg.name
  api_management_name = azurerm_api_management.apim.name
  revision            = "1"
  display_name        = "Sample Orchestration API"
  path                = "sample/v1"
  protocols           = ["https"]
  subscription_required = true
}

# Política da API: validate-jwt, CORS estrito, forwards
resource "azurerm_api_management_api_policy" "api_policy" {
  api_name            = azurerm_api_management_api.api.name
  api_management_name = azurerm_api_management.apim.name
  resource_group_name = azurerm_resource_group.rg.name

  xml_content = <<POL
<policies>
  <inbound>
    <base />
    <!-- CORS estrito -->
    <cors allow-credentials="false">
      <allowed-origins>
        ${join("\n        ", [for o in var.allowed_cors_origins : "<origin>${o}</origin>"])}
      </allowed-origins>
      <allowed-methods>
        <method>GET</method>
        <method>POST</method>
        <method>PUT</method>
      </allowed-methods>
      <allowed-headers>
        <header>authorization</header>
        <header>content-type</header>
      </allowed-headers>
      <expose-headers>
```

```
        <header>request-id</header>
      </expose-headers>
    </cors>

    <!-- Validate JWT -->
    <validate-jwt header-name="Authorization" failed-validation-httpcode="401"
require-expiration-time="true" require-scheme="Bearer">
      <openid-config
url="${azurerm_api_management_openid_connect_provider.oidc.metadata_endpoint}" />
      <audiences>
        <audience>api://${var.apim_gateway_host}</audience>
      </audiences>
      <issuers>
        <issuer>https://sts.windows.net/${data.azurerm_client_config.current.tenant_id}/</issuer>

<issuer>https://login.microsoftonline.com/${data.azurerm_client_config.current.tenant_id}/v2.0</issue
r>
      </issuers>
      <required-claims>
        <claim name="scp">
          <value>api.read</value>
          <value>api.write</value>
        </claim>
      </required-claims>
    </validate-jwt>

    <!-- Exemplo de validação de conteúdo (JSON Schema) -->
    <!-- Ajuste com seu schema -->
    <!--
    <validate-content unspecified-content-type-action="prevent">
      <json-content max-size="1048576">
        <schema ref="https://example.com/schemas/request.json" />
      </json-content>
    </validate-content>
    -->

    <!-- Remover headers sensíveis -->
    <set-header name="Server" exists-action="delete" />
    <set-header name="X-Powered-By" exists-action="delete" />

    <!-- Encaminhar para backend do Logic Apps -->
    <set-backend-service
base-url="https://REPLACE-LA-ENDPOINT.azurewebsites.net/api/workflows/REPLACE/trigger/RE
PLACE" />
  </inbound>
  <backend>
    <base />
  </backend>
  <outbound>
    <base />
  </outbound>
  <on-error>
    <base />
  </on-error>
</policies>
POL
}

# Associação do produto à API
resource "azurerm_api_management_product_api" "product_api" {
  api_name            = azurerm_api_management_api.api.name
  product_id          = azurerm_api_management_product.product.product_id
  api_management_name = azurerm_api_management.apim.name
  resource_group_name = azurerm_resource_group.rg.name
}

##############################
# Private Endpoint (exemplo)
##############################
```

```
# Storage backend com PE + Private DNS
resource "azurerm_storage_account" "be_storage" {
  name                     = lower(replace("${var.prefix}be${substr(replace(uuid(), "-", ""), 0, 8)}", "-", ""))
  resource_group_name      = azurerm_resource_group.rg.name
  location                 = azurerm_resource_group.rg.location
  account_tier             = "Standard"
  account_replication_type = "LRS"
  min_tls_version          = "TLS1_2"
}

resource "azurerm_private_dns_zone" "pdns_blob" {
  name                = "privatelink.blob.core.windows.net"
  resource_group_name = azurerm_resource_group.rg.name
}

resource "azurerm_private_dns_zone_virtual_network_link" "pdns_blob_link" {
  name                  = "${var.prefix}-blob-link"
  resource_group_name   = azurerm_resource_group.rg.name
  private_dns_zone_name = azurerm_private_dns_zone.pdns_blob.name
  virtual_network_id    = azurerm_virtual_network.vnet.id
}

resource "azurerm_private_endpoint" "pe_storage" {
  name                = "${var.prefix}-pe-storage"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  subnet_id           = azurerm_subnet.snet_pe.id

  private_service_connection {
    name                           = "${var.prefix}-pe-storage-conn"
    private_connection_resource_id = azurerm_storage_account.be_storage.id
    is_manual_connection           = false
    subresource_names              = ["blob"]
  }

  private_dns_zone_group {
    name                 = "blob-zone-group"
    private_dns_zone_ids = [azurerm_private_dns_zone.pdns_blob.id]
  }
}

####################################
# Diagnóstico (APIM, AppGW, LA)
####################################

resource "azurerm_monitor_diagnostic_setting" "diag_apim" {
  name                       = "${var.prefix}-apim-diag"
  target_resource_id         = azurerm_api_management.apim.id
  log_analytics_workspace_id = azurerm_log_analytics_workspace.law.id

  enabled_log {
    category = "GatewayLogs"
  }
  enabled_log {
    category = "WebSocketConnectionLogs"
  }
  enabled_log {
    category = "AuditLogs"
  }
  metric {
    category = "AllMetrics"
  }
}

resource "azurerm_monitor_diagnostic_setting" "diag_appgw" {
  name                       = "${var.prefix}-agw-diag"
  target_resource_id         = azurerm_application_gateway.appgw.id
  log_analytics_workspace_id = azurerm_log_analytics_workspace.law.id
```

```
  enabled_log { category = "ApplicationGatewayAccessLog" }
  enabled_log { category = "ApplicationGatewayPerformanceLog" }
  enabled_log { category = "ApplicationGatewayFirewallLog" }
  metric { category = "AllMetrics" }
}

resource "azurerm_monitor_diagnostic_setting" "diag_la" {
  name                       = "${var.prefix}-la-diag"
  target_resource_id         = azurerm_logic_app_standard.la.id
  log_analytics_workspace_id = azurerm_log_analytics_workspace.law.id

  enabled_log { category = "WorkflowRuntime" }
  enabled_log { category = "FunctionAppLogs" }
  metric      { category = "AllMetrics" }
}

#####################################
# Acesso ao Key Vault via MI (MI)
#####################################

# APIM -> KV (get segredos)
resource "azurerm_key_vault_access_policy" "kv_apim" {
  key_vault_id = azurerm_key_vault.kv.id
  tenant_id    = data.azurerm_client_config.current.tenant_id
  object_id    = azurerm_api_management.apim.identity[0].principal_id

  secret_permissions = ["Get", "List"]
}

# Logic Apps -> KV (get segredos)
resource "azurerm_key_vault_access_policy" "kv_la" {
  key_vault_id = azurerm_key_vault.kv.id
  tenant_id    = data.azurerm_client_config.current.tenant_id
  object_id    = azurerm_logic_app_standard.la.identity[0].principal_id

  secret_permissions = ["Get", "List"]
}

##################
# Saídas úteis  #
##################

output "apim_private_ips" {
  value = azurerm_api_management.apim.private_ip_addresses
}

output "app_gateway_public_ip" {
  value = azurerm_public_ip.appgw_pip.ip_address
}

output "developer_portal_url" {
  value = "https://${var.apim_portal_host}"
}
```

Próximos passos e notas de implementação
- DNS público: aponte o FQDN do gateway (api.suaempresa.com) para o IP público do Application Gateway. No DNS privado, o mesmo FQDN resolve para o IP privado do APIM (A record criado).
- Certificados: carregue certificados válidos (PFX) para os hostnames do gateway e portal; no APIM, o Terraform referencia os secrets do Key Vault; no Application Gateway, o exemplo usa PFX local (recomenda-se integrar com Key Vault via SSL profile).
- OIDC/Entra ID: crie um App Registration para a API (audience) e clientes; considere PKCE e Conditional Access (MFA, device compliance).
- Backend Logic Apps: substitua a URL do set-backend-service pelo endpoint real do seu workflow; aplique conectores aprovados e egress allowlist via UDR/Firewall.
- WAF: ajuste regras customizadas conforme payloads esperados; monitore e ajuste falsos positivos.
- Segurança adicional: considere Azure Firewall com UDR “deny by default” para a subnet do Logic Apps; Defender for Cloud; rotação de subscription keys; mTLS se os clientes suportarem.

- Versionamento: utilize apiVersionSet no APIM e políticas de depreciação.
- CI/CD: trate este Terraform como baseline; estruture módulos, naming/ tagging e pipelines de validação (tfsec/checkov) e testes de contrato.