🏗️ Relatório Completo de Infraestrutura OCI (tf_oci_clusters)

Este documento é um relatório técnico detalhado e exaustivo sobre a arquitetura, provisionamento e configurações estabelecidas através do repositório `tf_oci_clusters` utilizando Terraform na Oracle Cloud Infrastructure (OCI).

🚦 1. Visão Geral da Arquitetura e Ambientes

A infraestrutura segue um modelo multi-ambiente (`DEV` , `HML` e `PROD`), isolados geograficamente ou logicamente através de Compartments da OCI. A gestão é feita via Terraform e a entrega contínua dos manifestos Kubernetes ocorre através do ArgoCD (GitOps).

Característica	Ambiente Não-Produtivo (DEV / HML)	Ambiente Produtivo (PROD)
Padrão de Uso	Foco em otimização de custo (FinOps)	Foco em Alta Disponibilidade (HA) e Escalabilidade
Qtd. de Clusters	3 (Nexus, Barramento, Observabilidade)	3 (Nexus, Barramento, Observabilidade)
Nodes por Cluster	1 a 3 nodes	3 a 12 nodes
Configuração de Nodes	2 OCPUs / 16GB RAM	4 OCPUs / 32GB RAM
Disponibilidade (AD)	Single AD	Multi-AZ (3 Availability Domains)
Autoscaling	Agendado via CronJob (8h às 18h)	Dinâmico baseado em uso (Cluster Autoscaler)
Acesso Kubernetes	Endpoints Privados (via VPN/Bastion)	Endpoints configuráveis (Público/Privado dependendo da Flag)

🛡️ 2. Identidade, Federação e IAM

A base de acesso e permissões foi desenhada para se integrar ativamente ao ecossistema corporativo existente (Azure Active Directory), abstraindo a gestão local da OCI.

2.1. Federação com Azure Active Directory (SSO)

- **Integração SAML2**: O Identity Domains da OCI é configurado para confiar no Azure AD como um Identity Provider (IdP) via SAML2.
- **SSO na Console OCI**: O acesso ao painel da OCI permite que os usuários utilizem as as credenciais corporativas do Azure AD.
- **Just-In-Time Provisioning (JIT)**: Usuários que entrarem pela primeira vez são provisionados automaticamente no Identity Domains da OCI.

2.2. Gestão de Grupos e RBAC do OKE

- É realizado o **Mapeamento de Grupos** de forma que grupos do Azure AD determinem funções dentro da OCI e dentro dos clusters Kubernetes.
- Três níveis de permissionamento base para o OKE:
- `oke-admin`: Permissão total (`manage`) para administração da infraestrutura dos clusters.
- `oke-dev`: Permissão de uso (`use`) para o dia a dia de desenvolvedores.
- `oke-readonly`: Permissão restrita apenas a visualização (`read`).

2.3. Contas de Serviço (Service Accounts)

- Módulo dedicado à criação de usuários padrão IAM da OCI voltados para a integração de sistemas e deploy contínuos (por exemplo, chaves de API para os Azure Pipelines), definidos com e-mails corporativos padronizados.

🌐 3. Topologia de Rede (Network)

Cada ambiente OKE possui a sua própria Virtual Cloud Network (VCN) e arquitetura de sub-redes visando isolamento e segurança.

- **VCN (`vcn-oke`)**: Configurada com `10.110.0.0/16` para DEV. O roteamento conta com a resolução de DNS interna do Kubernetes.
- **Roteamento e Gateways**:
- **Internet Gateway (IGW)**: Tráfego de entrada e saída.
- **NAT Gateway (NGW)**: Garante que pods em sub-redes privadas consigam acessar a internet sem expor IPs públicos.
- **Service Gateway (SGW)**: Configurado na tabela de roteamento privada para que o tráfego OCI-OCI (ex: para o OCI Object Storage e OCI Registry) não saia pela internet, reduzindo custos e latência.
- **Topologia de Subnetting**:
- **Workers (`sbn-workers-1, 2, 3`)**: Sub-redes para instâncias do node pool. Podem bloquear IP público conforme a variável `is_private`. Distribuídas ao longo de múltiplas fault domains.
- **Load Balancers (`sbn-lb-1, 2`)**: Duas sub-redes públicas (para cumprir requisitos do OCI LB) usadas como ponto de entrada das aplicações (ex: via Ingress NGINX).
- **API Gateway (`sbn-api-gateway`)**: Uma subrede isolada e privada especificamente focada nos deployments Serverless do OCI API Gateway.

☸ 4. Infraestrutura Kubernetes (Oracle Kubernetes Engine)

A OCI hospeda a orquestração de containers utilizando o modelo Native Kubernetes Support provisionando três clusters principais: 1. **Core / Nexus**: Execução das aplicações e serviços focados do Invista (ex. frontends e backends em go/node/etc). 2. **Barramento / Integration**: Cluster voltado para integração, APIs de terceiros, gateways, Mensageria (NATS, RabbitMQ). 3. **Observabilidade**: Separado para

hospedar a stack de Prometheus, Grafana, VictoriaMetrics, Jaeger e OpenTelemetry.

Configurações Específicas:

- **Formas de Computação (Shapes)**: Múltiplas arquiteturas suportadas. Capacidade de usar o novo processador Ampere (A1) ARM com detecção em tempo real via script Terraform, ou `VM.Standard.E4.Flex` (AMD x86_64).
- **Hardening do Node Pool**: Bootstrapping injetando as chaves SSH previamente aprovadas, isolando regras de rede.
- **Autoscaling**:
- Em ambientes PROD: O *Cluster Autoscaler* Kubernetes (implantado via ArgoCD) é responsável por crescer/encolher nós dinamicamente monitorando recursos de Pod Pending.
- Em ambientes DEV/HML: O Autoscaler não existe. Ao invés disso, *Kubernetes CronJobs* (Agendamentos) conversam diretamente via OCI CLI para zerar às 18:00h da noite e acordar nós às 8:00h da manhã em dias de semana.

🚀 5. Aplicações e CI/CD

O ciclo de vida do software e infraestrutura obedece a práticas ágeis integrando pipelines com abordagens modernas Cloud Native.

5.1. OCI API Gateway & Micro-Frontends (MFEs)

- Uso de API Gateways públicos mapeando as rotas da web com OCI Object Storage (S3-compatible). O armazenamento serve arquivos estáticos HTML/JS/CSS e o Gateway atua com regras de path fallbacks (`/` voltando o `index.html`) e URLs como `mfe-user-dev.invista.com.br` .
- **Relação de MFEs implementados**: `mfe-user` , `mfe-shell` , `mfe-auth` , `mfe-person` , `mfe-formalization` , e aplicativos de prova de conceito

(`mfe-poc`).

5.2. Gestão de Contêineres (ArgoCD & GitOps)

- O ArgoCD atua no coração dos clusters. É inicializado de forma híbrida: O Terraform gera os aquivos `kubeconfig` no executor, adiciona os templates Helm do ArgoCD no Kubernetes, em seguida mapeia repositórios de manifesto.
- **SSO com ArgoCD OIDC**: Os desenvolvedores não logam com senhas hard-coded no ArgoCD. Ele é configurado com um Client ID OIDC voltado para o Identity Domains, que repassa o fluxo OAuth para o Azure. As credenciais se tornam inteiramente corporativas.

5.3. Pipeline de Integração e Deploy no Azure DevOps

A infraestrutura CI/CD está contida no diretório `.azure/` utilizando Azure Pipelines estruturados: - `azure-pipelines.yml`: Orquestração multi-staged automatizando o Terraform. Fluxos `Plan & Apply`: - **Setup Seguro**: Uso de Service Principals ou Variable Groups para guardar `OCI_PRIVATE_KEY` e Access Keys para comunicação com o backend estático (Object Storage Terraform State). - **Passos**: Terraform Init -> Validate -> Plan (disponibiliza relatório como rate) -> Apply Condicional (executado manualmente ou se a branch o aprovar como stage de release paramétrico). - **Scale Scripts**: Pipeline auxiliar dedicada às funcionalidades de infraestrutura agilizadas (`azure-pipelines-scale.yml`).

📊 6. Observabilidade Nativa

Integração entre Cloud Infrastructure Native Monitoring e Stack interna: - **Log Groups OCI**: Todos clusters e workloads de API enviam logs diretamente ao OCI Logging. - **Notificações & Alarmes**: Tópico de ONS (Oracle Notification Services) alimentado com métricas críticas de rede, saturação de nodes, VCN Gateways, disparando alarmes para times de plantão (via e-mail, slack, ou pager). - **Tracing**

Ativo: A execução de TF gera JSONs em background úteis para auditorias técnicas sob demanda do pipeline.

Relatório Exaustivo gerado automaticamente em 27 de Fevereiro de 2026. Revisão: DevOps & Platform Engineering.